Laboratory Manual
2023-24

Sub:- Big Data and Data Analytics
B.Tech Final Year(CSE)

Prepared By
Prof.Thorat N.A.

G.K. Gujar Memorial Charitable Trust's,
Dr.Ashok Gujar Technical Institute's
Dr. Daulatrao Aher College of Engineering, Karad.
Vidyanagar Ext. Banawadi, Tal. Karad 415124, Dist. Satara, Maharashtra INDIA

Prepared By:- Prof. Thorat N.A.	Approved By: Head of Department

Student Responsibilities

1. In the very beginning of the laboratory work, the student has to do programming individually. For this reason, regular attendance is strictly required.

2. Every laboratory session is divided into two parts. In the first part, the instructor will be lecturing on the test objective, procedure and data collection. In the second part, the students, organized in groups (individual student), are required to do the programming. In order to perform the experiment within the assigned period, and to gain the maximum benefit from the experiment, the students must familiarize themselves with the purpose, objective, and procedure of the experiment before coming to the laboratory. Relevant lecture notes and laboratory manual should be studied carefully and thoroughly.

3. At the end of the experiment, every student should submit the written algorithm & programs & testes result for approval by the instructor.

4. It should be understood that laboratory facilities and equipment's (Hardware. & Software) are provided to enhance the learning process.

5. The equipment's must be properly cared after every laboratory session. Also, students should always take precautions to avoid any possible hazards. Students must follow laboratory regulations provided at the end of this section.

G.K. Gujar Memorial Charitable Trust's,
Dr.Ashok Gujar Technical Institute's
Dr. Daulatrao Aher College of Engineering, Karad.
Vidyanagar Ext. Banawadi, Tal. Karad 415124, Dist. Satara, Maharashtra INDIA

Department of Computer Science & Engineering

EXPERIMENT LIST

Department: Computer Science and Engineering | **Academic Year:-2023-24**

Class:-Final Year(B.Tech) | **Semester:- VIII**

Subject:-Big Data and Analytics | **Semester Duration:-6 months**

Exp No.	Experiment Name
1	**Installation of Hadoop and R.**
2	**Building Hadoop MapReduce application for counting frequency of word/phrase in simple text file.**
3	**Study of Hadoop YARN Administration command and User commands.**
4	**Study of Hadoop Hive DDL commands, like create database, Viewing database, Dropping database, Altering database, creating tables, Dropping and altering tables.**
5	**Study of Hadoop Hive DML commands like Insert, Delete, Update, Data Retrieval queries and Join- inner and outer.**
6	**Working with operations in Pig-FOREACH,ASSERT, FILTER, GROUP, ORDER BY, DISTINCT, JOIN, LIMIT, SAMPLE, SPLIT, FLATTEN.**
7	**Study of R-declaring variables, expressions, functions and executing R-script.**
8	**Working with R with data sets-create, read, write and R Tables-create, read, write.**
9	**Manipulating and processing data in R-merging datasets, sorting data, putting data into shape, managing data using matrices, managing data using data frames.**
10	**Installation and configuration of Apache Spark on Local Machine.**

Subject In charge
Prof.Thorat N.A.

HOD
Prof.Kakade S.P.

Experiment 1

Aim: Installation of Hadoop and R.

Basics of Hadoop:

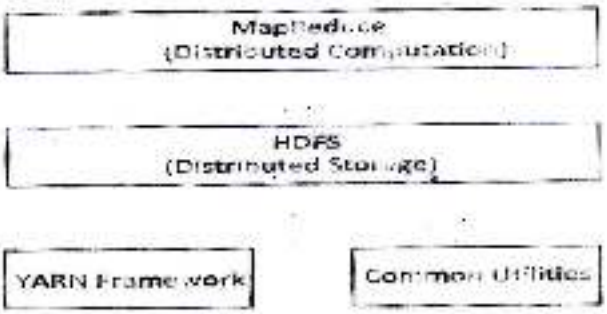
Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Hadoop Environment:

Hadoop framework includes following four modules:

- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System (HDFS™):** A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

Hadoop framework.



Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above but also to the collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark etc.

MapReduce

Hadoop **MapReduce** is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).

- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Typically both the input and the output are stored in a file system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master **JobTracker** and one slave TaskTracker per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically.

The JobTracker is a single point of failure for the Hadoop MapReduce service which means if JobTracker goes down, all running jobs are halted.

Hadoop Distributed File System

Hadoop can work directly with any mountable distributed file system such as Local FS, HFTP FS, S3 FS, and others, but the most common file system used by Hadoop is the Hadoop Distributed File System (HDFS).

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerant manner.

HDFS uses a master/slave architecture where master consists of a single **NameNode** that manages the file system metadata and one or more slave **DataNodes** that store the actual data.

A file in an HDFS namespace is split into several blocks and those blocks are stored in a set of DataNodes. The NameNode determines the mapping of blocks to the DataNodes. The DataNodes takes care of read and write operation with the file system. They also take care of block creation, deletion and replication based on instruction given by NameNode.

Advantages of Hadoop

- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

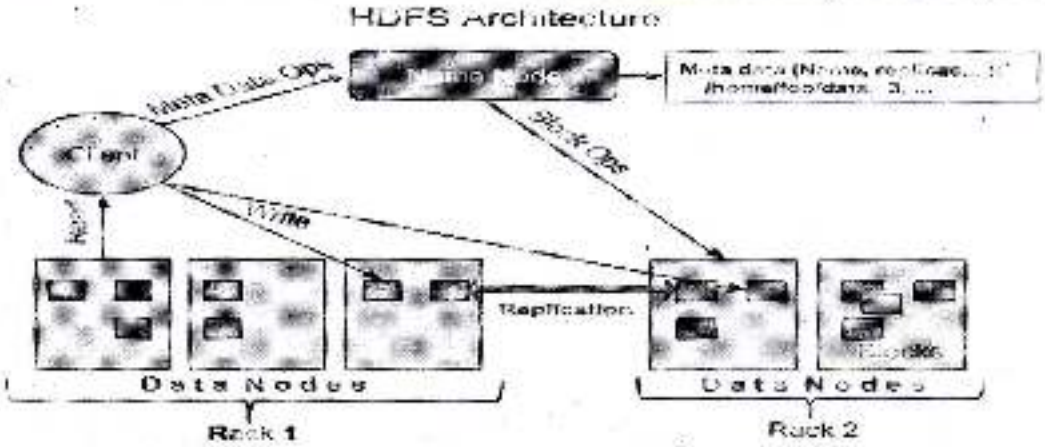
HDFS architecture:

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly faulttolerant and designed using low cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

HDFS Architecture:

Given below is the architecture of a Hadoop File System.



HDFS follows the master-slave architecture and it has the following elements.

Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

Datanode

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

R programming language:

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac.

R is free software distributed under a GNU-style copy left, and an official part of the GNU project called **GNU S.**

Evolution of R

R was initially written by **Ross Ihaka** and **Robert Gentleman** at the Department of Statistics of the University of Auckland in Auckland, New Zealand. R made its first appearance in 1993.

- A large group of individuals has contributed to R by sending code and bug reports.
- Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code archive.

Features of R

As stated earlier, R is a programming language and software environment for statistical analysis, graphics representation and reporting. The following are the important features of R −

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility,
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

Merits

1) Open Source

An open-source language is a language on which we can work without any need for a license or a fee. R is an open source language. We can contribute to the development of R by optimizing our packages, developing new ones, and resolving issues.

2) Platform Independent

R is a platform-independent language or cross-platform programming language which means its code can run on all operating systems. R enables programmers to develop software for several competing platforms by writing a program only once. R can run quite easily on Windows, Linux, and Mac.

3) Machine Learning Operations

R allows us to do various machine learning operations such as classification and regression. For this purpose, R provides various packages and features for developing the artificial neural network. R is used by the best data scientists in the world.

4) Exemplary support for data wrangling

R allows us to perform data wrangling. R provides packages such as dplyr, readr which are capable of transforming messy data into a structured form.

5) Quality plotting and graphing

R simplifies quality plotting and graphing. R libraries such as ggplot2 and plotly advocates for visually appealing and aesthetic graphs which set R apart from other programming languages.

6) The array of packages

R has a rich set of packages. R has over 10,000 packages in the CRAN repository which are constantly growing. R provides packages for data science and machine learning operations.

7) Statistics

R is mainly known as the language of statistics. It is the main reason why R is predominant than other programming languages for the development of statistical tools.

8) Continuously Growing

R is a constantly evolving programming language. Constantly evolving means when something evolves, it changes or develops over time, like our taste in music and clothes, which evolve as we get older. R is a state of the art which provides updates whenever any new feature is added.

Demerits

1) Data Handling

In R, objects are stored in physical memory. It is in contrast with other programming languages like Python. R utilizes more memory as compared to Python. It requires the entire data in one single place which is in the memory. It is not an ideal option when we deal with Big Data.

2) Basic Security

R lacks basic security. It is an essential part of most programming languages such as Python. Because of this, there are many restrictions with R as it cannot be embedded in a web-application.

3) Complicated Language

R is a very complicated language, and it has a steep learning curve. The people who don't have prior knowledge or programming experience may find it difficult to learn R.

4) Weak Origin

The main disadvantage of R is, it does not have support for dynamic or 3D graphics. The reason behind this is its origin. It shares its origin with a much older programming language "S."

5) Lesser Speed

R programming language is much slower than other programming languages such as MATLAB and Python. In comparison to other programming language, R packages are much slower.

In R, algorithms are spread across different packages. The programmers who have no prior knowledge of packages may find it difficult to implement algorithms.

Conclusion : Thus we have studied installation of hadoop and R.

Steps for installation of Hadoop

1. Prerequisites

First, we need to make sure that the following prerequisites are installed:

1. Java 8 runtime environment (JRE): Hadoop 3 requires a Java 8 installation. I prefer using the offline installer.
2. Java 8 development Kit (JDK)
3. To unzip downloaded Hadoop binaries, we should install 7zip.
4. I will create a folder "E:\hadoop-env" on my local machine to store downloaded files.

2. Download Hadoop binaries

The first step is to download Hadoop binaries from the official website. The binary package size is about 342 MB.

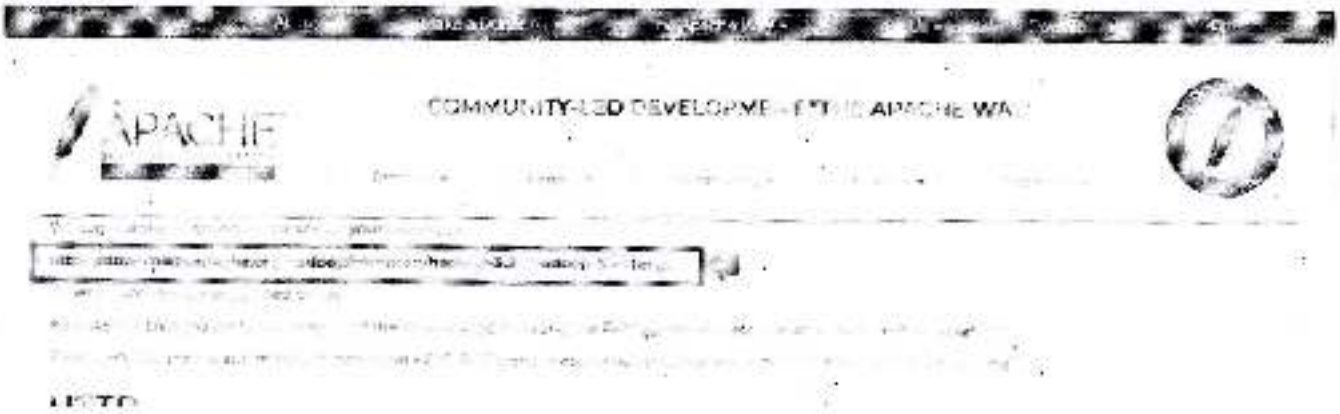


Figure 1 Hadoop binaries download link

After finishing the file download, we should unpack the package using 7zip int two steps. First, we should extract the hadoop-3.2.1.tar.gz library, and then, we should unpack the extracted tar file:

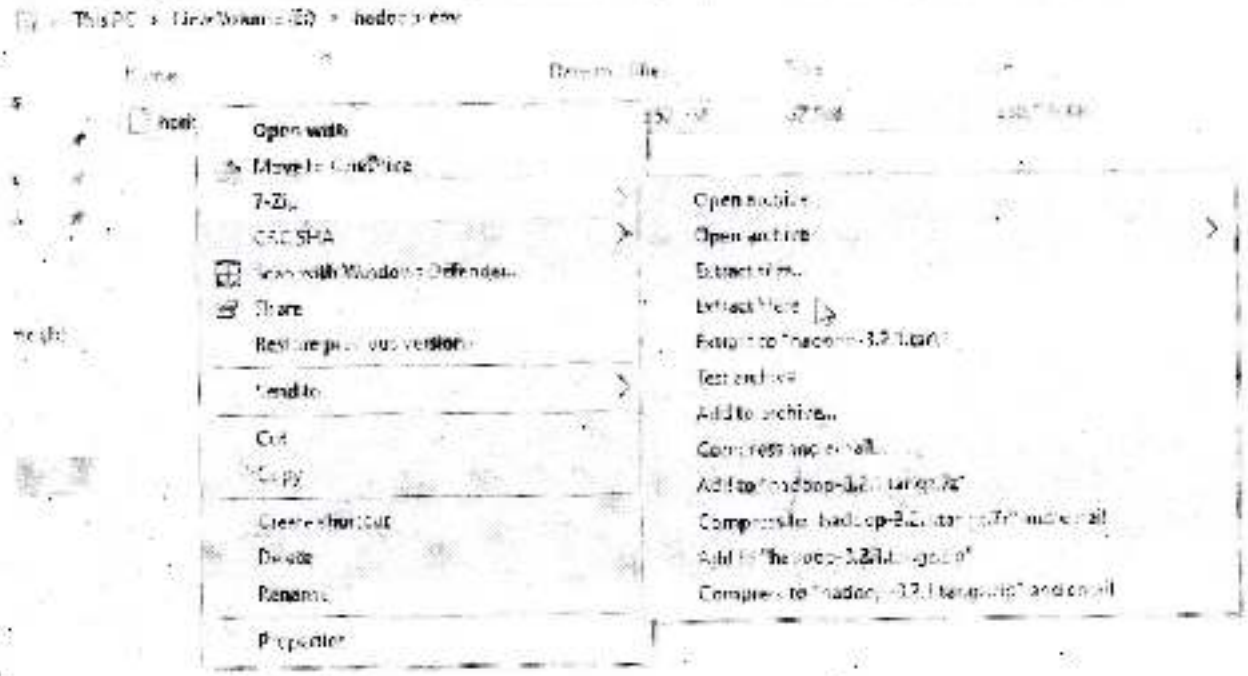


Figure 2 — Extracting hadoop-3.2.1.tar.gz package using 7zip

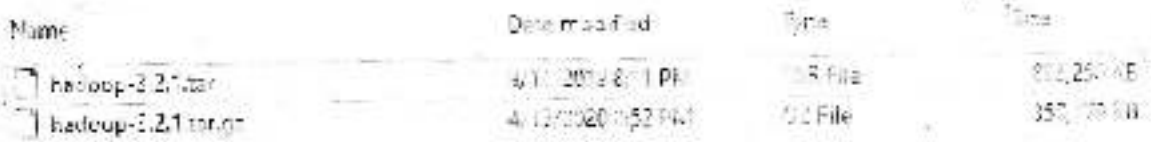


Figure 3 — Extracted hadoop-3.2.1.tar file

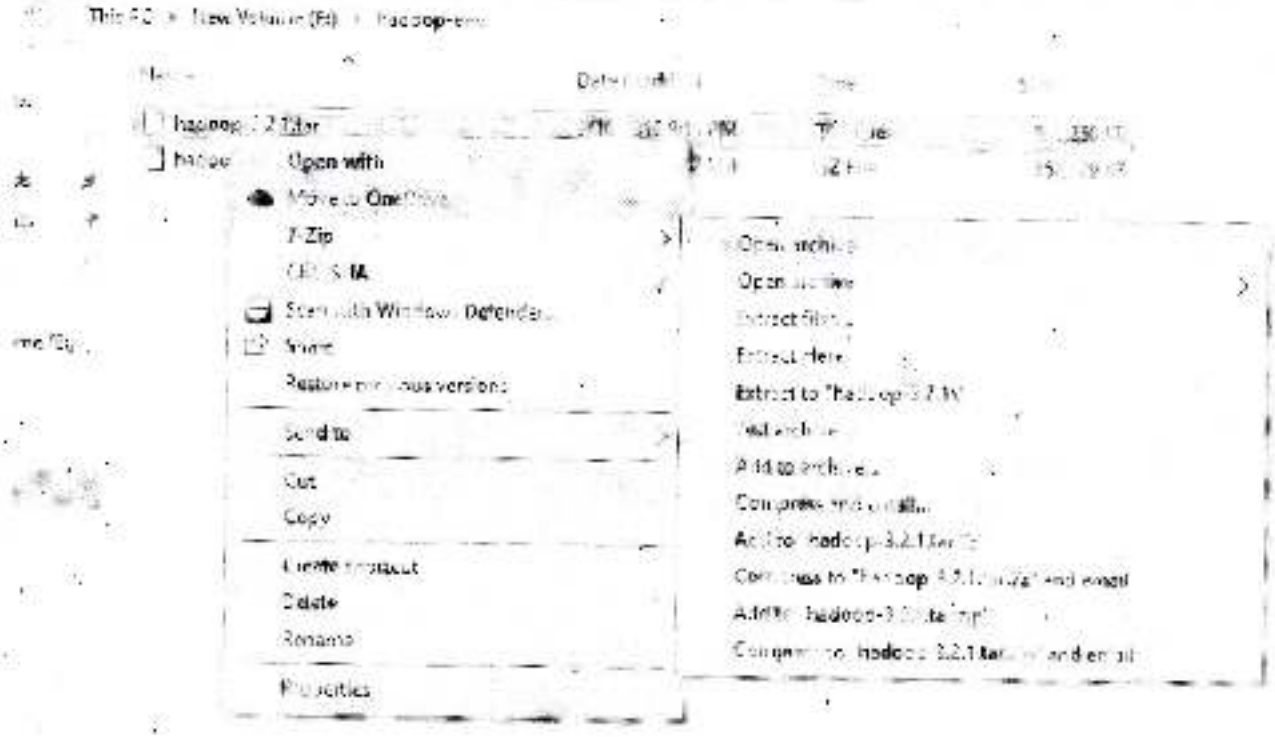


Figure 4 Extracting the hadoop-3.2.1.tar file

The tar file extraction may take some minutes to finish. In the end, you may see some warnings about symbolic link creation. Just ignore these warnings since they are not related to windows.

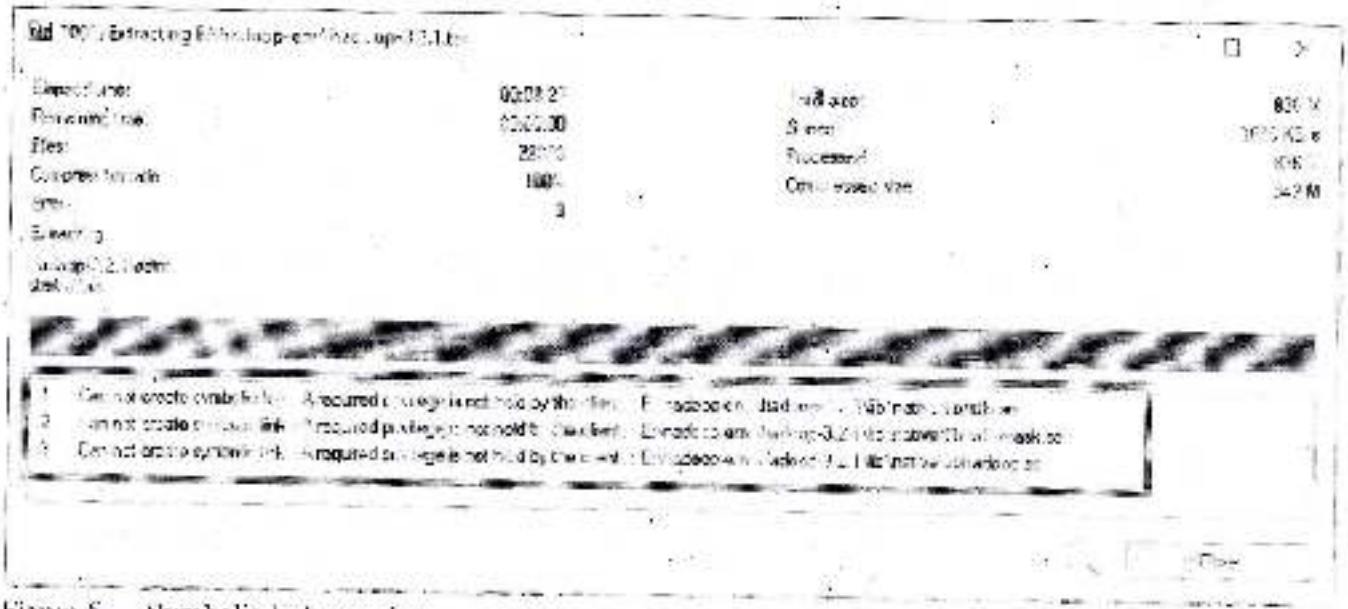


Figure 5 — Symbolic link warnings

After unpacking the package, we should add the Hadoop native IO libraries, which can be found in the following GitHub repository: https://github.com/cdarlint/winutils.

Since we are installing Hadoop 3.2.1, we should download the files located in https://github.com/cdarlint/winutils/tree/master/hadoop-3.2.1/bin and copy them into the "hadoop-3.2.1\bin" directory.

3. Setting up environment variables

After installing Hadoop and its prerequisites, we should configure the environment variables to define Hadoop and Java default paths.

To edit environment variables, go to Control Panel > System and Security > System (or right-click > properties on My Computer icon) and click on the "Advanced system settings" link.

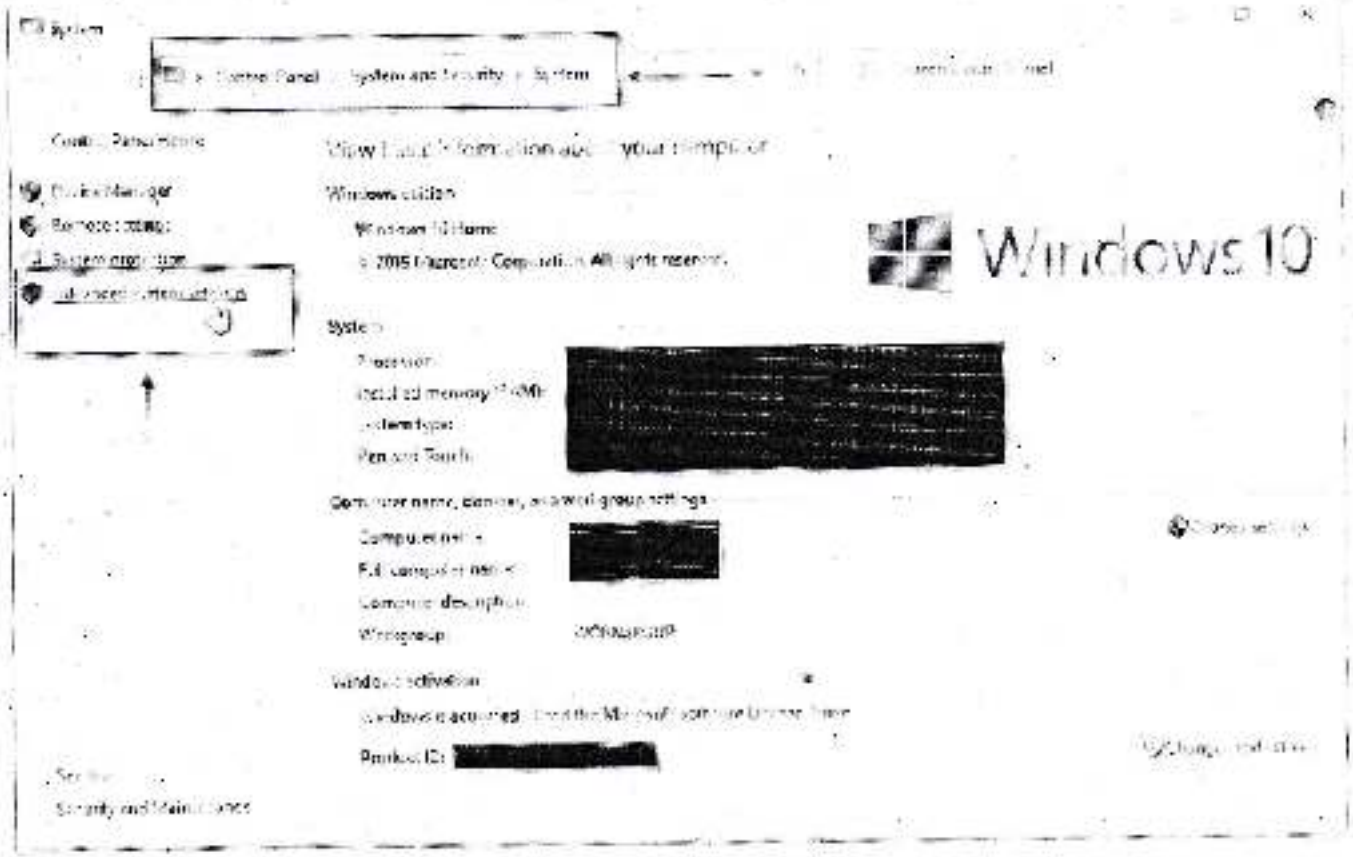


Figure 6 — Opening advanced system settings

When the "Advanced system settings" dialog appears, go to the "Advanced" tab and click on the "Environment variables" button located on the bottom of the dialog.

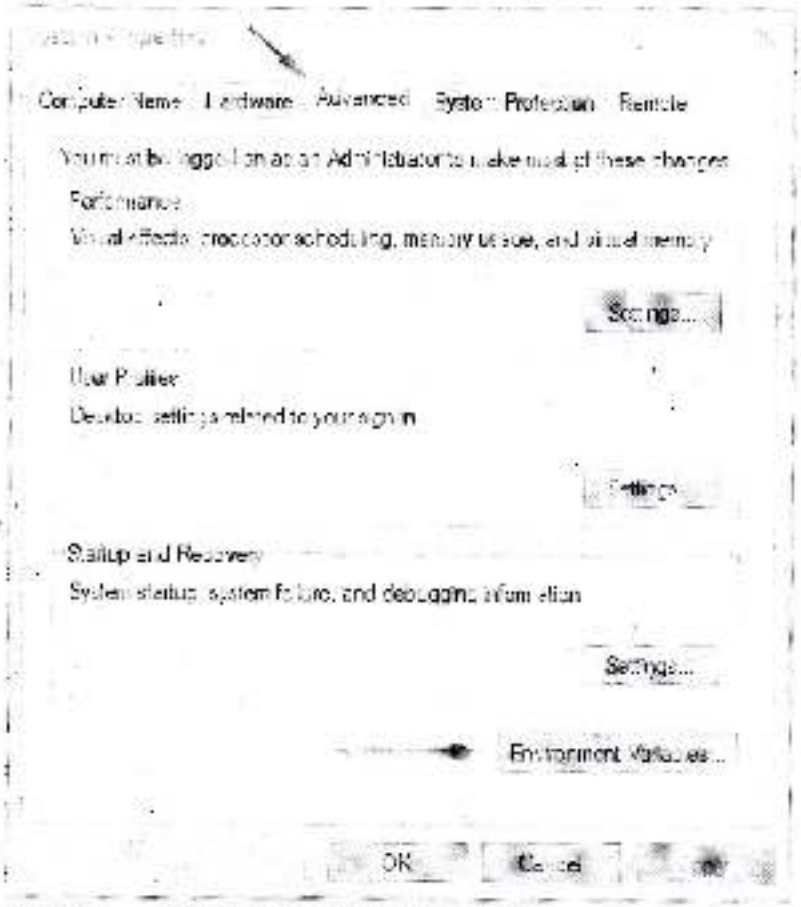


Figure 7 — Advanced system settings dialog

In the "Environment Variables" dialog, press the "New" button to add a new variable.

Note: In this guide, we will add user variables since we are configuring Hadoop for a single user. If you are looking to configure Hadoop for multiple users, you can define System variables instead.

There are two variables to define:

1. JAVA_HOME: JDK installation folder path
2. HADOOP_HOME: Hadoop installation folder path

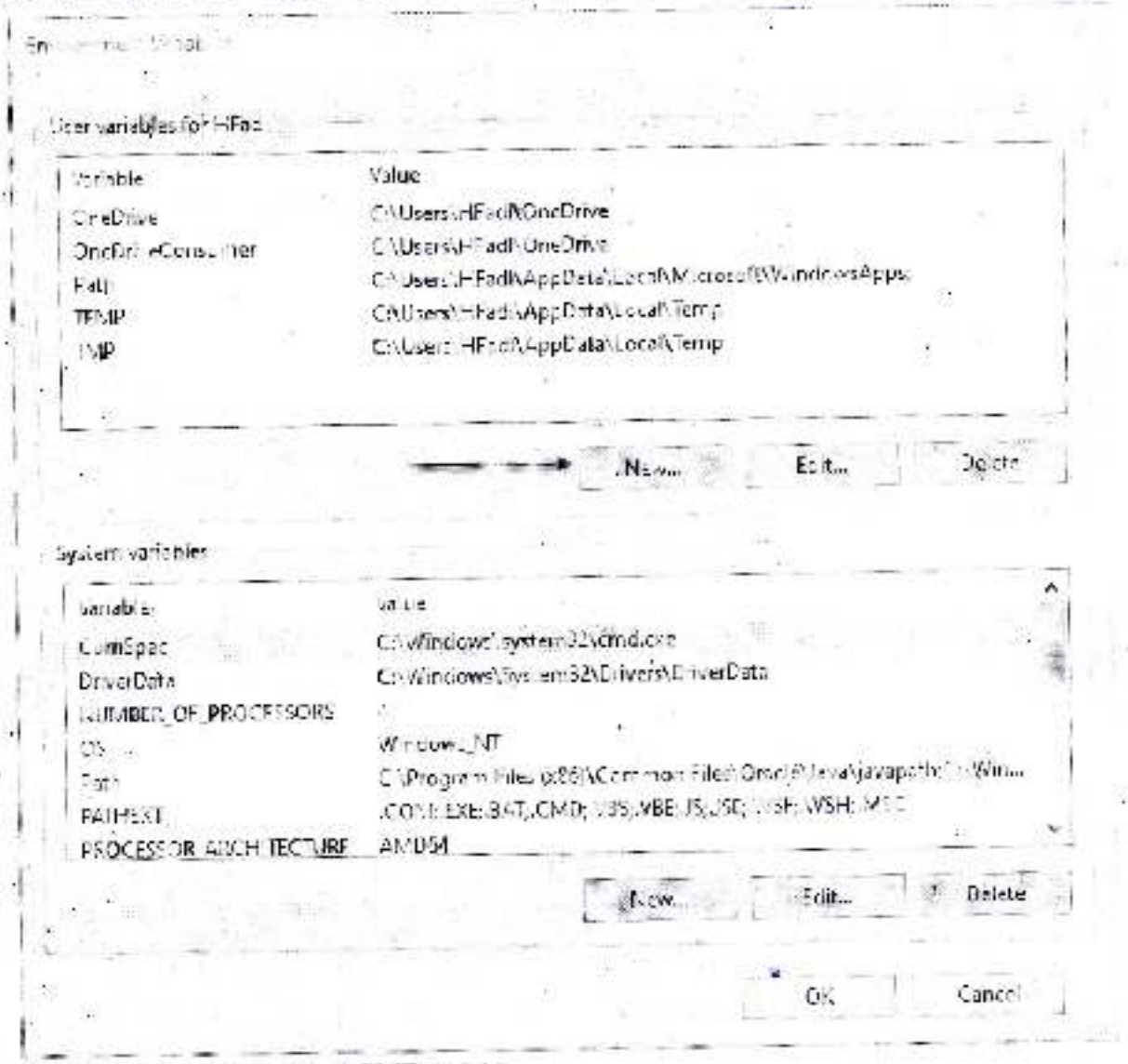


Figure 8 — Adding JAVA_HOME variable

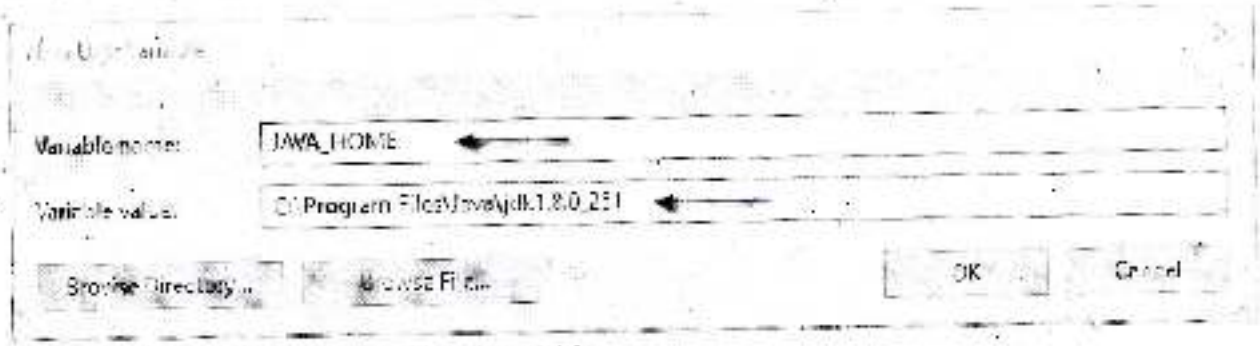


Figure 9 — Adding HADOOP_HOME variable

Now, we should edit the PATH variable to add the Java and Hadoop binaries paths as shown in the following screenshots.

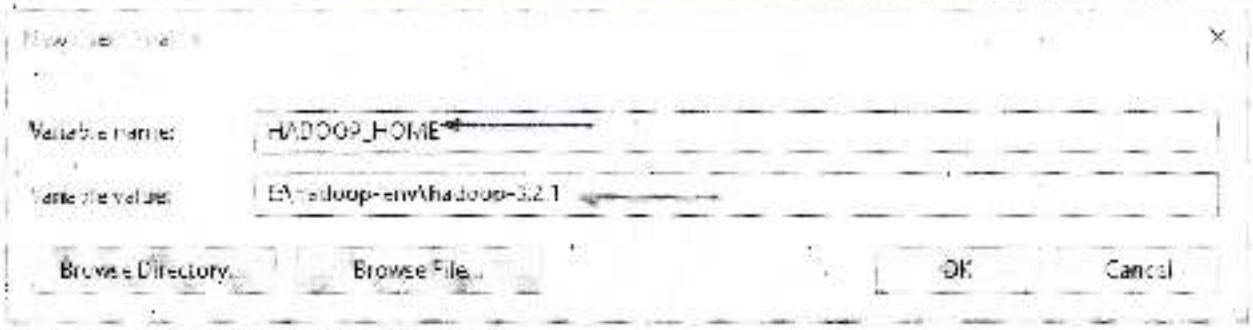


Figure 10 — Editing the PATH variable

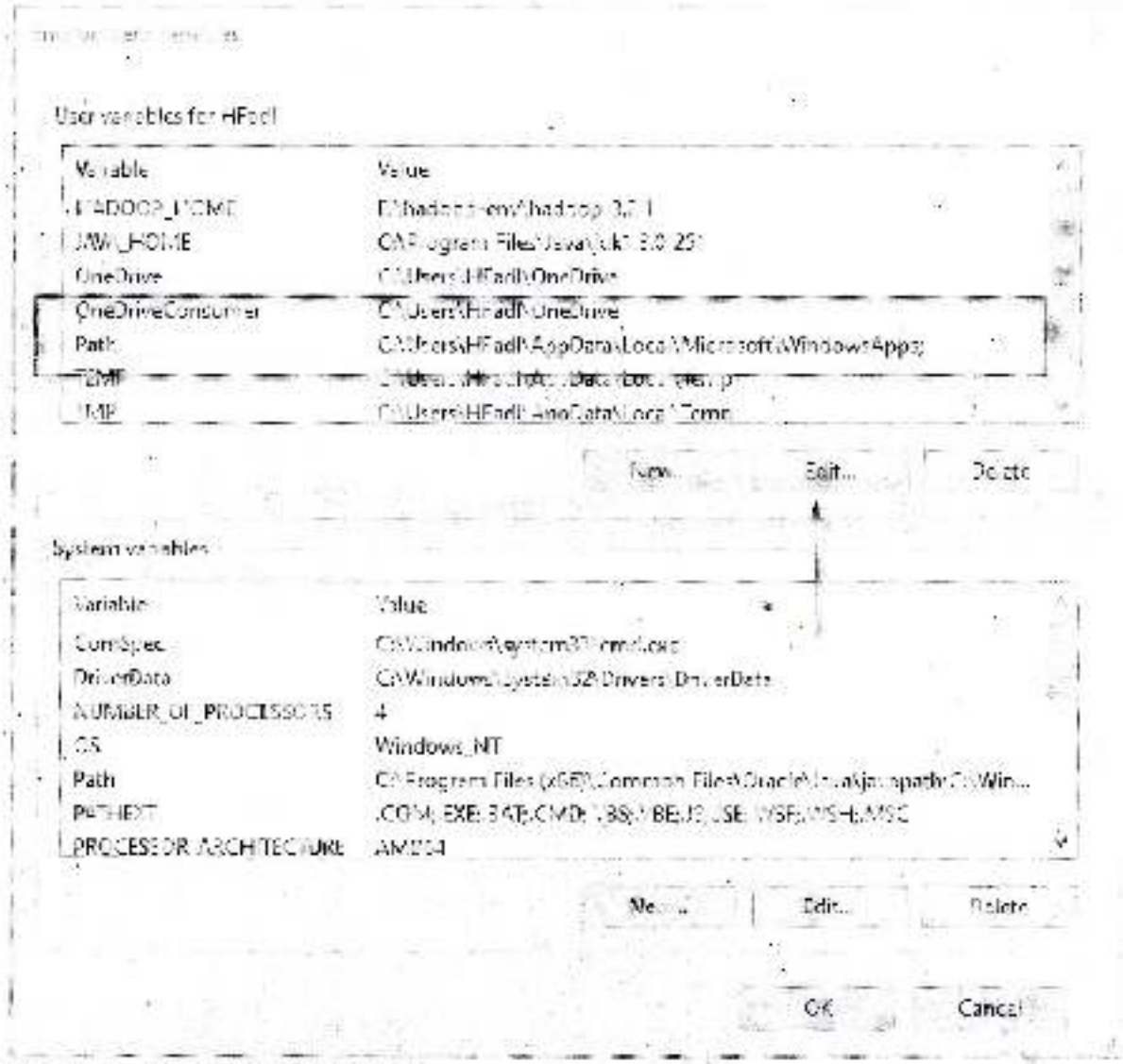


Figure 11 — Editing PATH variable

Figure 12 — Adding new paths to the PATH variable

3.1. JAVA_HOME is incorrectly set error

Now, let's open PowerShell and try to run the following command:

hadoop -version

In this example, since the JAVA_HOME path contains spaces, I received the following error:

JAVA_HOME is incorrectly set

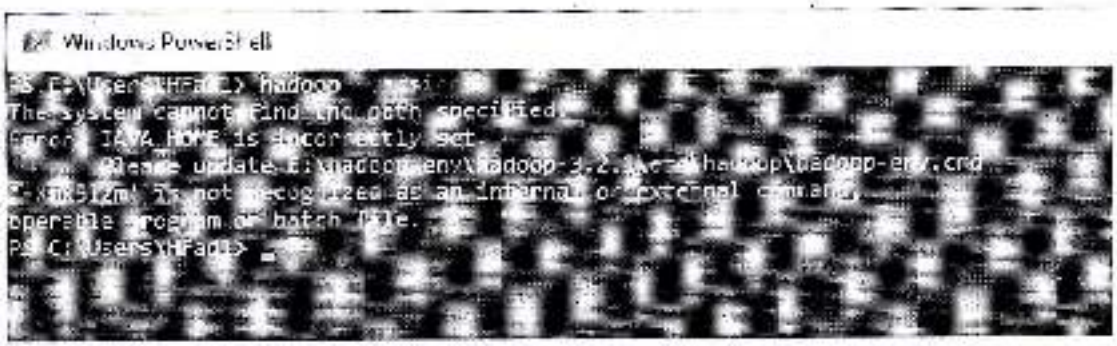


Figure 13 — JAVA_HOME error

To solve this issue, we should use the windows 8.3 path instead. As an example:

- Use "Progra~1" instead of "Program Files"
- Use "Progra~2" instead of "Program Files(x86)"

After replacing "Program Files" with "Progra~1", we closed and reopened PowerShell and tried the same command. As shown in the screenshot below, it runs without errors.

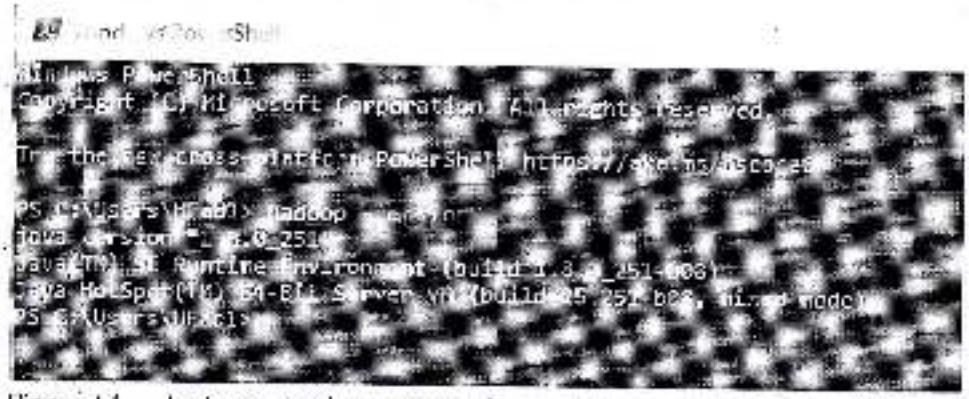


Figure 14 — hadoop -version command executed successfully

4. Configuring Hadoop cluster

There are four files we should alter to configure Hadoop cluster:

1. %HADOOP_HOME%\etc\hadoop\hdfs-site.xml
2. %HADOOP_HOME%\etc\hadoop\core-site.xml
3. %HADOOP_HOME%\etc\hadoop\mapred-site.xml
4. %HADOOP_HOME%\etc\hadoop\yarn-site.xml

4.1. HDFS site configuration

As we know, Hadoop is built using a master slave paradigm. Before altering the HDFS configuration file, we should create a directory to store all master node (name node) data and another one to store data (data node). In this example, we created the following directories:

- E:\hadoop-env\hadoop-3.2.1\data\dfs\namenode
- E:\hadoop-env\hadoop-3.2.1\data\dfs\datanode

Now, let's open "hdfs-site.xml" file located in "%HADOOP_HOME%\etc\hadoop" directory, and we should add the following properties within the <configuration></configuration> element:

<property><name>dfs.replication</name><value>1</value></property><property><name>dfs.namenode.name.dir</name><value>file:///E:/hadoop-env/hadoop-3.2.1/data/dfs/namenode</value></property><property><name>dfs.datanode.data.dir</name><value>file:///E:/hadoop-env/hadoop-3.2.1/data/dfs/datanode</value></property>

Note that we have set the replication factor to 1 since we are creating a single node cluster.

4.2. Core site configuration

Now, we should configure the name node URL adding the following XML code into the <configuration></configuration> element within "core-site.xml":

<property><name>fs.default.name</name><value>hdfs://localhost:9820</value></property>

4.3. Map Reduce site configuration

Now, we should add the following XML code into the <configuration></configuration> element within "mapred-site.xml":

<property><name>mapreduce.framework.name</name><value>yarn</value><description>MapReduce framework name</description></property>

4.4. Yarn site configuration

Now, we should add the following XML code into the <configuration></configuration> element within "yarn-site.xml":

<property><name>yarn.nodemanager.aux-services</name><value>mapreduce_shuffle</value><description>Yarn Node Manager Aux Service</description></property>

5. Formatting Name node

After finishing the configuration, let's try to format the name node using the following command:

hdfs namenode -format

-Due to a bug in the Hadoop 3.2.1 release, you will receive the following error:

2020-04-17 22:04:01,503 ERROR namenode.NameNode: Failed to start namenode.java.lang.UnsupportedOperationExceptionat java.nio.file.Files.setPosixFilePermissions(Files.java:2044)at org.apache.hadoop.hdfs.server.common.Storage$StorageDirectory.clearDirectory(Storage.java:452)at org.apache.hadoop.hdfs.server.namenode.NNStorage.format(NNStorage.java:591)at org.apache.hadoop.hdfs.server.namenode.NNStorage.format(NNStorage.java:613)at org.apache.hadoop.hdfs.server.namenode.FSImage.format(FSImage.java:188)at org.apache.hadoop.hdfs.server.namenode.NameNode.format(NameNode.java:1206)at org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(NameNode.java:1649)at org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java:1759)2020-04-17 22:04:01,511 INFO util.ExitUtil: Exiting with status 1: java.lang.UnsupportedOperationException2020-04-17 22:04:01,518 INFO namenode.NameNode: SHUTDOWN_MSG:

This issue will be solved within the next release. For now, you can fix it temporarily using the following steps (reference):

1. Download hadoop-hdfs-3.2.1.jar file from the following link.
2. Rename the file name hadoop-hdfs-3.2.1.jar to hadoop-hdfs-3.2.1.bak in folder %HADOOP_HOME%\share\hadoop\hdfs
3. Copy the downloaded hadoop-hdfs-3.2.1.jar to folder %HADOOP_HOME%\share\hadoop\hdfs

Now, if we try to re-execute the format command (Run the command prompt or PowerShell as administrator), you need to approve file system format.

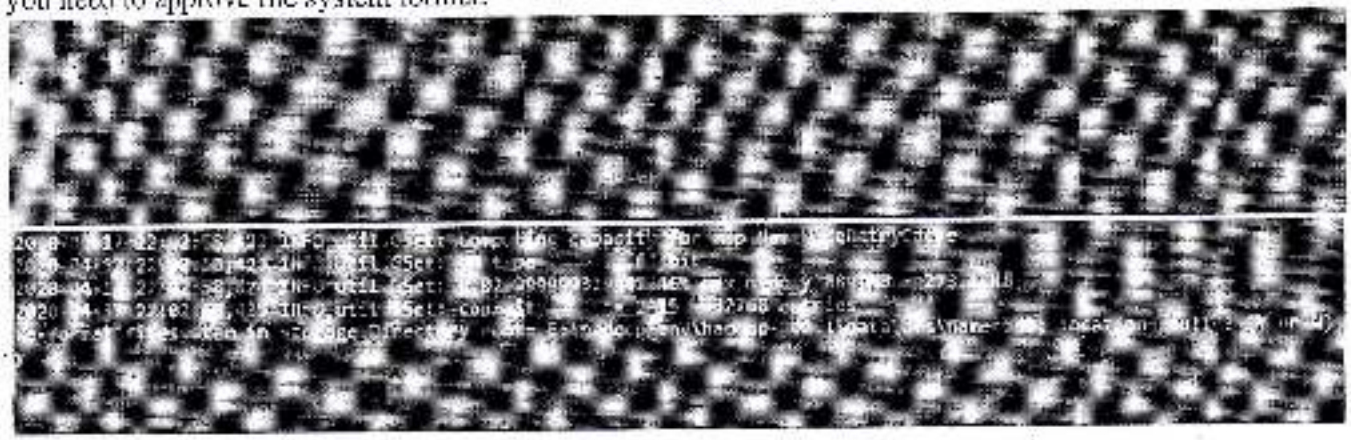


Figure 15 — File system format approval

And the command is executed successfully:

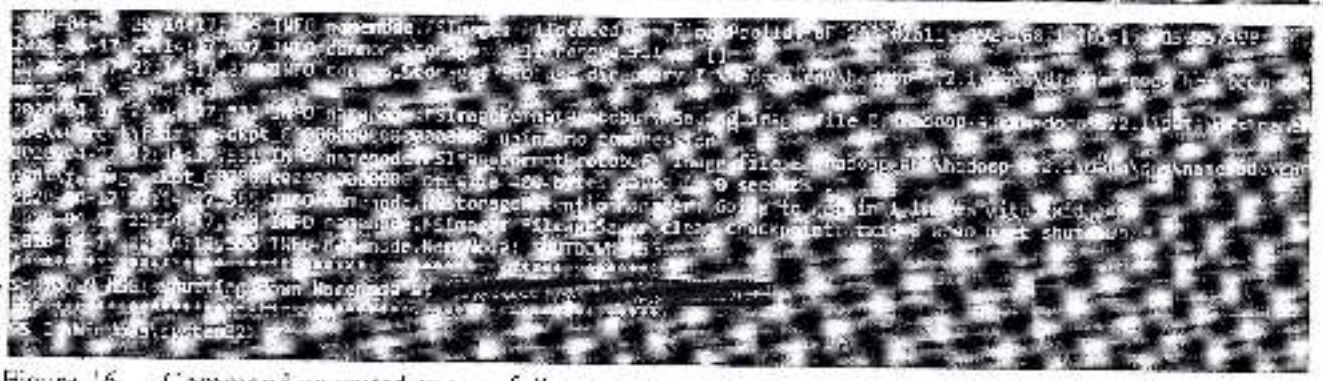


Figure 16 — Command executed successfully.

6. Starting Hadoop services

Now, we will open PowerShell, and navigate to "%HADOOP_HOME%\sbin" directory. Then we will run the following command to start the Hadoop nodes:

./start-dfs.cmd

Figure 17 — StartingHadoop nodes

Two command prompt windows will open (one for the name node and one for the data node) as follows:

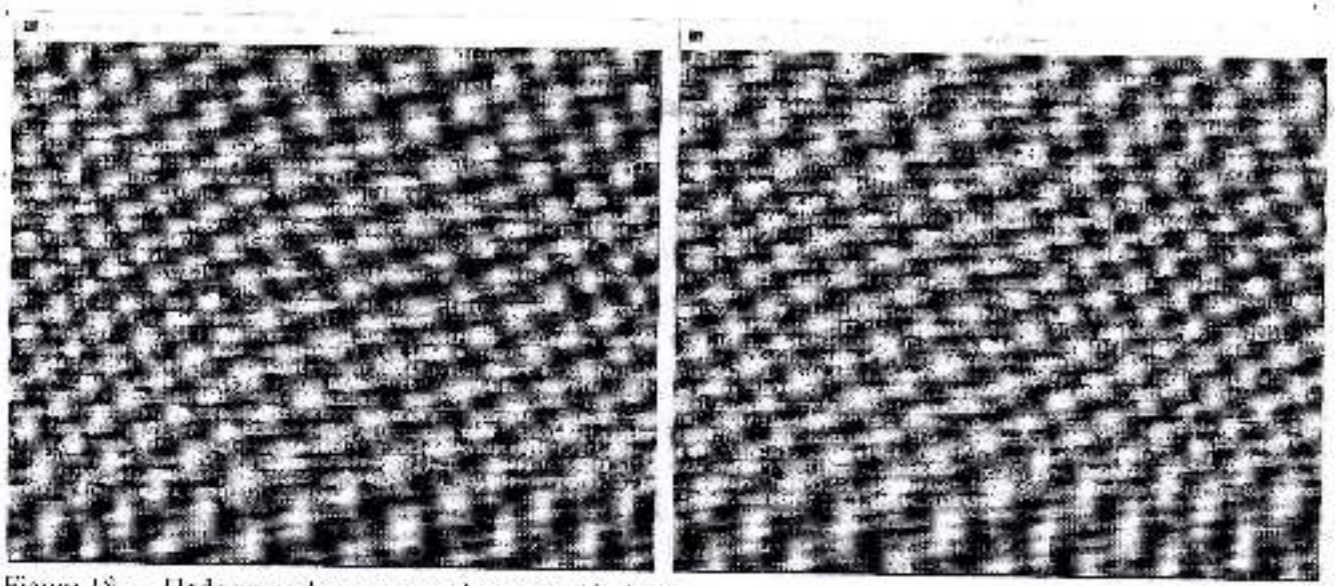


Figure 18 — Hadoop nodes command prompt windows

Next, we must start the Hadoop Yarn service using the following command:

./start-yarn.cmd

Figure 19 — Starting Hadoop Yarn services

Two command prompt windows will open (one for the resource manager and one for the node manager) as follows:

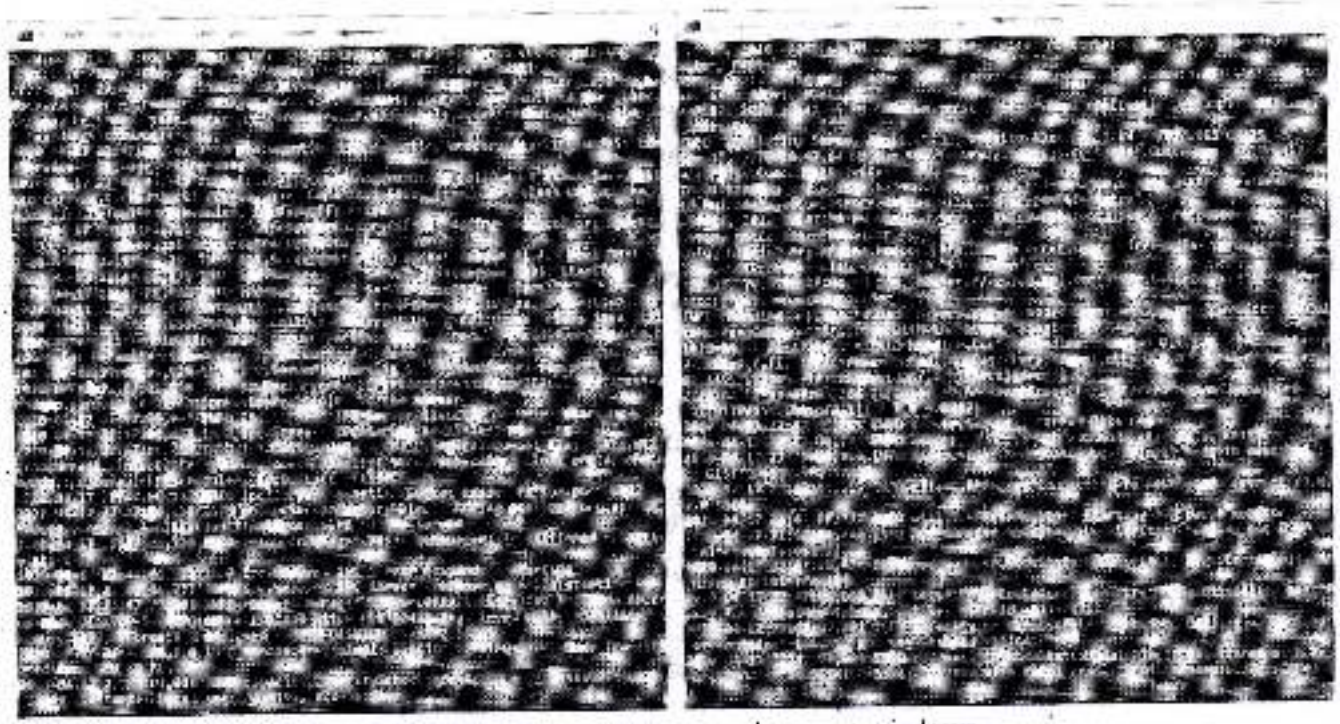


Figure 20— Node manager and Resource manager command prompt windows

To make sure that all services started successfully, we can run the following command:

jps

It should display the following services:

14560 DataNode

4960 ResourceManager

5936 NameNode

768 NodeManager

14636 Jps

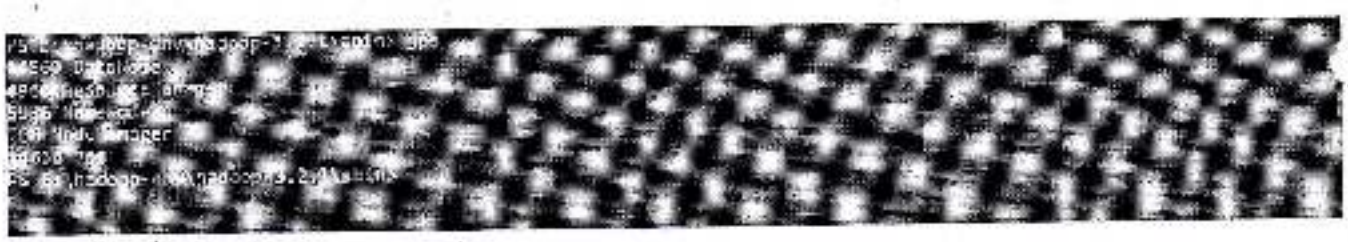


Figure 21 — Executing jps command

Installation of R

Obtaining R

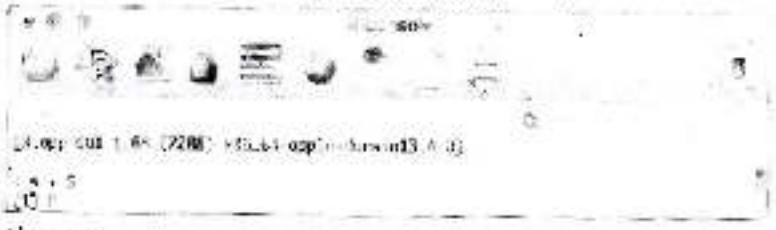
- R is available for Linux, MacOS, and Windows. Software can be downloaded from
- The Comprehensive R Archive Network (CRAN).

Startup

- After R is downloaded and installed, simply find and launch R from your Applications folder.

Entering Commands

- R is a command line driven program. The user enters commands at the prompt (> by default) and each command is executed one at a time.



The Workspace

- The workspace is your current R working environment and includes any user-defined objects (vectors, matrices, data frames, lists, functions). At the end of an R session, the user can save an image of the current workspace that is automatically reloaded the next time R is started.

Graphic User Interfaces

- Aside from the built in R console, RStudio is the most popular R code editor, and it interfaces with R for Windows, MacOS, and Linux platforms.

R GUI

Experiment No.: 2

Aim: Building Hadoop MapReduce application for counting frequency of word/phrase in simple text file.

What is MapReduce?

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
    - **Map stage** : The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
    - **Reduce stage** : This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable

interface to facilitate sorting by the framework. Input and Output types of a MapReduce job: (Input) <k1, v1> -> map -> <k2, v2> > reduce -> <k3, v3>(Output).

	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Terminology

- **PayLoad** - Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** - Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** - Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** - Node where data is presented in advance before any processing takes place.
- **MasterNode** - Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** - Node where Map and Reduce program runs.
- **JobTracker** - Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** - Tracks the task and reports status to JobTracker.
- **Job** - A program is an execution of a Mapper and Reducer across a dataset.
- **Task** - An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** - A particular instance of an attempt to execute a task on a SlaveNode.

Example Scenario

Given below is the data regarding the electrical consumption of an organization. It contains the monthly electrical consumption and the annual average for various years.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

If the above data is given as input, we have to write applications to process it and produce results such as finding the year of maximum usage, year of minimum usage, and so on. This is a walkover for the programmers with finite number of records. They will simply write the logic to produce the required output, and pass the data to the application written.

But, think of the data representing the electrical consumption of all the largescale industries of a particular state, since its formation.

When we write applications to process such bulk data,

- They will take a lot of time to execute.
- There will be a heavy network traffic when we move data from source to network server and so on.

To solve these problems, we have the MapReduce framework.

Conclusion : Thus we have studied Map reduce framework.

(Take printouts)

Input Data

The above data is saved as **sample.txt**and given as input. The input file looks as shown below.

```
1979  23  23  2  43  24  25  26  26  26  26  25  26  25
1980  26  27  28  28  28  30  31  31  31  30  30  30  29
1981  31  32  32  32  33  34  35  36  36  34  34  34  34
1984  39  38  39  39  39  41  42  43  40  39  38  38  40
1985  38  39  39  39  39  41  41  41  00  40  39  39  45
```

Example Program

Given below is the program to the sample data using MapReduce framework.

```
package hadoop;

import java.util.*;

import java.io.IOException;
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ProcessUnits
{
 //Mapper class
 public static class E_EMapper extends MapReduceBase implements
 Mapper<LongWritable ,/*Input key Type */
 Text,            /*Input value Type*/
 Text,            /*Output key Type*/
 IntWritable>     /*Output value Type*/
 {

   //Map function
   public void map(LongWritable key, Text value,
   OutputCollector<Text, IntWritable> output,
   Reporter reporter) throws IOException
   {
     String line = value.toString();
     String lasttoken = null;
     StringTokenizer s = new StringTokenizer(line,"\t");
     String year = s.nextToken();
```

```
        while(s.hasMoreTokens())
          {
            lasttoken=s.nextToken();
          }

        int avgprice = Integer.parseInt(lasttoken);
        output.collect(new Text(year), new IntWritable(avgprice));
      }
    }


    //Reducer class
    public static class E_EReduce extends MapReduceBase implements
    Reducer< Text, IntWritable, Text, IntWritable >
    {

      //Reduce function
      public void reduce( Text key, Iterator <IntWritable> values,
        OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException
        {
        int maxavg=30;
          int val=Integer.MIN_VALUE;

          while (values.hasNext())
          {
            if((val=values.next().get())>maxavg)
            {
              output.collect(key, new IntWritable(val));
            }
          }

        }
    }

    //Main function
    public static void main(String args[])throws Exception
    {
      JobConf conf = new JobConf(ProcessUnits.class);

      conf.setJobName("max_electricityunits");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      conf.setMapperClass(E_EMapper.class);
```

```
    conf.setCombinerClass(E_EReduce.class);
    conf.setReducerClass(E_EReduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
  }
```

Save the above program as **ProcessUnits.java**. The compilation and execution of the program is explained below.

Compilation and Execution of Process Units Program

Let us assume we are in the home directory of a Hadoop user (e.g. /home/hadoop).

Follow the steps given below to compile and execute the above program.

Step 1

The following command is to create a directory to store the compiled java classes.

$ mkdir units

Step 2

Download **Hadoop-core-1.2.1.jar,** which is used to compile and execute the MapReduce program. Visit the following link http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1 to download the jar. Let us assume the downloaded folder is **/home/hadoop/.**

Step 3

The following commands are used for compiling the **ProcessUnits.java** program and creating a jar for the program.

$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java

$ jar -cvf units.jar -C units/ .

Step 4

The following command is used to create an input directory in HDFS.

$HADOOP_HOME/bin/hadoop fs -mkdir input_dir

Step 5

The following command is used to copy the input file named **sample.txt**in the input directory of HDFS.

$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir

Step 6

The following command is used to verify the files in the input directory.

$HADOOP_HOME/bin/hadoop fs -ls input_dir/

Step 7

The following command is used to run the Eleunit_max application by taking the input files from the input directory.

$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits input_dir output_dir

Wait for a while until the file is executed. After execution, as shown below, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc.

INFO mapreduce.Job: Job job_1414748220717_0002
completed successfully
14/10/31 06:02:52
INFO mapreduce.Job: Counters: 49
File System Counters

FILE: Number of bytes read=61
FILE: Number of bytes written=279400
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=546
HDFS: Number of bytes written=40

HDFS: Number of read operations=9
HDFS: Number of large read operations=0
HDFS: Number of write operations=2 Job Counters

Launched map tasks=2
Launched reduce tasks=1
Data-local map tasks=2
Total time spent by all maps in occupied slots (ms)=146137
Total time spent by all reduces in occupied slots (ms)=441
Total time spent by all map tasks (ms)=14613
Total time spent by all reduce tasks (ms)=44120
Total vcore-seconds taken by all map tasks=146137

Total vcore-seconds taken by all reduce tasks=44120
Total megabyte-seconds taken by all map tasks=149644288
Total megabyte-seconds taken by all reduce tasks=45178880

Map-Reduce Framework

Map input records=5
Map output records=5
Map output bytes=45
Map output materialized bytes=67
Input split bytes=208
Combine input records=5
Combine output records=5
Reduce input groups=5
Reduce shuffle bytes=6
Reduce input records=5
Reduce output records=5
Spilled Records=10
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=948
CPU time spent (ms)=5160
Physical memory (bytes) snapshot=47749120
Virtual memory (bytes) snapshot=2899349504
Total committed heap usage (bytes)=277684224

File Output Format Counters

Bytes Written=40

Step 8

The following command is used to verify the resultant files in the output folder.

$HADOOP_HOME/bin/hadoop fs -ls output_dir/

Step 9

The following command is used to see the output in **Part-00000** file. This file is generated by HDFS.

$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000

Below is the output generated by the MapReduce program.

1981 34

1984 40

1985 45

Step 10

The following command is used to copy the output folder from HDFS to the local file system for analyzing.

$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop dfs get output_dir /home/hadoop

Experiment No : 3

Aim : Study of Hadoop YARN Administration command and User commands.

Apache Hadoop YARN (Yet Another Resource Negotiator) is a cluster management technology.

YARN is one of the key features in the second generation Hadoop 2 version of the Apache Software Foundation's open source distributed processing framework. Originally described by Apache as a redesigned resource manager, YARN is now characterized as a large-scale, distributed operating system for big data applications.

In 2012, YARN became a sub-project of the larger Apache Hadoop project. Sometimes called MapReduce 2.0, YARN is a software rewrite that decouples MapReduce's resource management and scheduling capabilities from the data processing component, enabling Hadoop to support more varied processing approaches and a broader array of applications. For example, Hadoop clusters can now run interactive querying and streaming data applications simultaneously with MapReduce batch jobs. The original incarnation of Hadoop closely paired the Hadoop Distributed File System (HDFS) with the batch-oriented MapReduce programming framework, which handles resource management and job scheduling on Hadoop systems and supports the parsing and condensing of data sets in parallel.

YARN combines a central resource manager that reconciles the way applications use Hadoop system resources with node manager agents that monitor the processing operations of individual cluster nodes. Running on commodity hardware clusters, Hadoop has attracted particular interest as a staging area and data store for large volumes of structured and unstructured data intended for use in analytics applications. Separating HDFS from MapReduce with YARN makes the Hadoop environment more suitable for operational applications that can't wait for batch jobs to finish.

Overview

Yarn commands are invoked by the bin/yarn script. Running the yarn script without any arguments prints the description for all commands.

Usage: yarn [--config confdir] COMMAND

Yarn has an option parsing framework that employs parsing generic options as well as running classes.

COMMAND_OPTIONS	Description
--config confdir	Overwrites the default Configuration directory. Default is $[HADOOP_PREFIX]/conf.
COMMAND COMMAND_OPTIONS	Various commands with their options are described in the following sections. The commands have been grouped into User Commands and Administration Commands.

User Commands

Commands useful for users of a Hadoop cluster.

jar

Runs a jar file. Users can bundle their Yarn code in a jar file and execute it using this command.

Usage: yarn jar <jar> [mainClass] args...

application

Prints application(s) report/kill application

Usage: yarn application <options>

COMMAND_OPTIONS	Description

-list	Lists applications from the RM. Supports optional use of -appTypes to filter applications based on application type, and -appStates to filter applications based on application state
-appStates States	Works with -list to filter applications based on input comma-separated list of application states. The valid application state can be one of the following: ALL, NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED
-appTypes Types	Works with -list to filter applications based on input comma-separated list of application types.
-status ApplicationId	Prints the status of the application.
-kill ApplicationId	Kills the application.

node

Prints node report(s)

Usage: yarn node <options>

COMMAND_OPTIONS	Description
-list	Lists all running nodes. Supports optional use of -states to filter nodes based on node state, and -all to list all nodes.
-states States	Works with -list to filter nodes based on input comma-separated list of node states.
-all	Works with -list to list all nodes.
-status NodeId	Prints the status report of the node.

logs

Dump the container logs

Usage: yarn logs -applicationId <application ID> <options>

COMMAND_OPTIONS	Description
-applicationId <application ID>	Specifies an application id
-appOwner AppOwner	AppOwner (assumed to be current user if not specified)
-containerId ContainerId	ContainerId (must be specified if node address is specified)
-nodeAddress NodeAddress	NodeAddress in the format nodename:port (must be specified if container id is specified)

classpath

Prints the class path needed to get the Hadoop jar and the required libraries

Usage: yarn classpath

version

Prints the version.

Usage: yarn version

Administration Commands

Commands useful for administrators of a Hadoop cluster.

resourcemanager

Start the ResourceManager

Usage: yarn resourcemanager

nodemanager

Start the NodeManager

Usage: yarn nodemanager

proxyserver

Start the web proxy server

Usage: yarn proxyserver

rmadmin

Runs ResourceManager admin client

Usage: yarn rmadmin [-refreshQueues] [-refreshNodes] [-refreshUserToGroupsMapping]
[-refreshSuperUserGroupsConfiguration] [-refreshAdminAcls]
[-refreshServiceAcl] [getGroups [username]] [-help [cmd]]
[-transitionToActive <serviceId>]
[-transitionToStandby <serviceId>]
[-getServiceState <serviceId>]
[-checkHealth <serviceId>]

COMMAND_OPTIONS	Description
-refreshQueues	Reload the queues' acls, states and scheduler specific properties. ResourceManager will reload the mapred-queues configuration file.
-refreshNodes	Refresh the hosts information at the ResourceManager.
-refreshUserToGroupsMappings	Refresh user to groups mappings.
-refreshSuperUserGroupsConfiguration	Refresh superuser proxy groups mappings.
-refreshAdminAcls	Refresh acls for administration of ResourceManager
-refreshServiceAcl	Reload the service-level authorization policy file ResourceManager will reload the authorization policy file.
-getGroups [username]	Get groups the specified user belongs to.
-help [cmd]	Displays help for the given command or all commands if none is specified.

-transitionToActive <serviceId>	Transitions the service into Active state.
-transitionToStandby <serviceId>	Transitions the service into Standby state.
getServiceState <serviceId>	Returns the state of the service.
-checkHealth <serviceId>	Requests that the service perform a health check. The RMAdmin tool will exit with a non-zero exit code if the check fails.

Daemonlog

Get/Set the log level for each daemon.

Usage: yarn daemonlog -getlevel <host:port> <name>

Usage: yarn daemonlog -setlevel <host:port> <name> <level>

COMMAND_OPTIONS	Description
-getlevel <host:port> <name>	Prints the log level of the daemon running at <host:port>. This command internally connects to http://<host:port>/logLevel?log=<name>
-setlevel <host:port> <name> <level>	Sets the log level of the daemon running at <host:port>. This command internally connects to http://<host:port>/logLevel?log=<name>

Conclusion: Thus we have studied Yarn commands.

Experiment No :4

Aim: Study of Hadoop Hive DDL commands, like create database, Viewing database, Droppung database, Altering database, creating tables, Dropping and altering tables.

What is Hive

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive is not

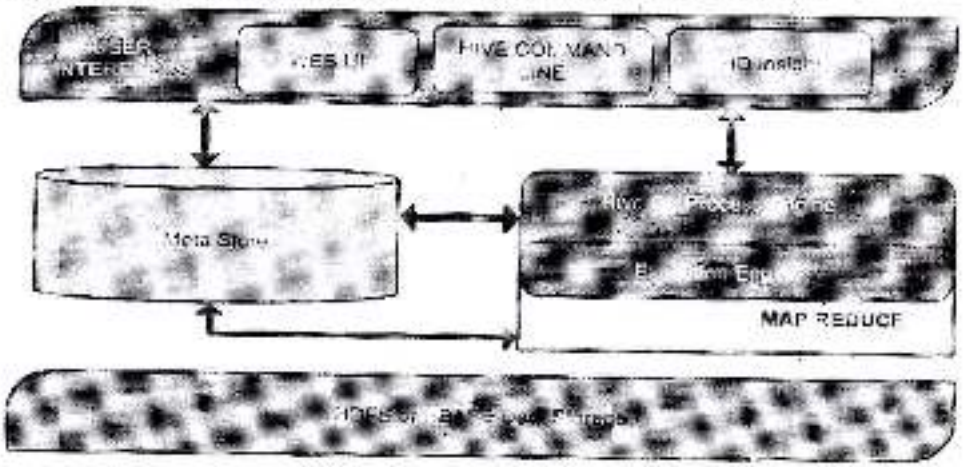
- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Features of Hive

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Architecture of Hive

The following component diagram depicts the architecture of Hive:



Hive DDL Commands

- **Create Database Statement**

Syntax:

CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command. The following query is executed to create a database named **userdb**:

hive> CREATE DATABASE [IF NOT EXISTS] userdb;

or

hive> CREATE SCHEMA userdb;

- **Viewing database**

The following query is used to verify a databases list:

hive> SHOW DATABASES;
default
userdb

- **Drop database:**

  Drop Database is a statement that drops all the tables and deletes the database. Its syntax is as follows:

  Syntax:

DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];

The following queries are used to drop a database. Let us assume that the database name is **userdb**.

hive> DROP DATABASE IF EXISTS userdb;

The following query drops the database using **CASCADE**. It means dropping respective tables before dropping the database.

hive> DROP DATABASE IF EXISTS userdb CASCADE;

The following query drops the database using **SCHEMA**.

hive> DROP SCHEMA userdb;
hive> DROP DATABASE IF EXISTS userdb;

- **Creating Hive Tables**

1. **Create Table Statement**

Create Table is a statement used to create a table in Hive. The syntax and example are as follows:

Syntax

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name

[(col_name data_type [COMMENT col_comment], ...)]
[COMMENT table_comment]
[ROW FORMAT row_format]
[STORED AS file_format]

Example

Let us assume you need to create a table named **employee** using **CREATE TABLE** statement. The following table lists the fields and their data types in employee table:

Sr.No	Field Name	Data Type
1	Eid	int
2	Name	String
3	Salary	Float
4	Designation	string

STORED IN TEXT FILE

The following query creates a table named **employee** using the above data.

hive> CREATE TABLE IF NOT EXISTS employee (eid int, name String,
salary String, destination String)
COMMENT 'Employee details'

ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

If you add the option IF NOT EXISTS, Hive ignores the statement in case the table already exists.

On successful creation of table, you get to see the following response:

OK
Time taken: 5.905 seconds
hive>

2. Browse the table

hive> Show tables;

3. Altering Tables

hive> ALTER TABLE Sonoo RENAME TO Kafka;
hive> ALTER TABLE Kafka ADD COLUMNS (col INT);
hive> ALTER TABLE HIVE_TABLE ADD COLUMNS (col1 INT COMMENT 'a comment');
hive> ALTER TABLE HIVE_TABLE REPLACE COLUMNS (col2 INT, weight STRING, baz INT CO
MMENT 'baz replaces new_col1')

4. Drop Table Statement

Syntax:

DROP TABLE [IF EXISTS] table_name;

The following query drops a table named **employee**:

hive> DROP TABLE IF EXISTS employee;

On successful execution of the query, you get to see the following response:

OK
Time taken: 5.3 seconds
hive>

Conclusion : Thus we have studied HIVE DDL commands.

Experiment No: 5

Aim: Study of Hadoop Hive DML commands like Insert, delete, update, data retrieval queries and Join-inner and outer.

What is Hive?

Apache Hive is a Data warehouse system which is built to work on Hadoop. It is used to querying and managing large datasets residing in distributed storage. Before becoming a open source project of Apache Hadoop, Hive was originated in Facebook. It provides a mechanism to project structure onto the data in Hadoop and to query that data using a SQL-like language called HiveQL (HQL).

Hive is used because the tables in Hive are similar to tables in a relational database. If you are familiar with SQL, it's a cakewalk. Many users can simultaneously query the data using Hive QL.

What is HQL?

Hive defines a simple SQL-like query language to querying and managing large datasets called Hive-QL (HQL). It's easy to use if you're familiar with SQL Language. Hive allows programmers who are familiar with the language to write the custom MapReduce framework to perform more sophisticated analysis.

Uses of Hive:

1. The Apache Hive distributed storage.
2. Hive provides tools to enable easy data extract/transform/load (ETL)
3. It provides the structure on a variety of data formats.
4. By using Hive, we can access files stored in Hadoop Distributed File System (HDFS is used to querying and managing large datasets residing in) or in other data storage systems such as Apache HBase.

Limitations of Hive:

- Hive is not designed for Online transaction processing (OLTP), it is only used for the Online Analytical Processing.
- Hive supports overwriting or apprehending data, but not updates and deletes.
- In Hive, sub queries are not supported.

Why Hive is used inspite of Pig?

The following are the reasons why Hive is used in spite of Pig's availability:

- Hive QL is a declarative language like SQL, PigLatin is a data flow language.
- Pig: a data-flow language and environment for exploring very large datasets.
- Hive: a distributed data warehouse.

Components of Hive:

Metastore :

Hive stores the schema of the Hive tables in a Hive Metastore. Metastore is used to hold all the information about the tables and partitions that are in the warehouse. By default, the metastore is run in the same process as the Hive service and the default Metastore is DerBy Database.

SerDe :

Serializer, Deserializer gives instructions to hive on how to process a record.

Data Manipulation Language (DML)

DML statements are used to retrieve, store, modify, delete, insert and update data in the database.

1. **LOAD**

   Syntax :

LOAD data <LOCAL> inpath <file path> into table [tablename]

The Load operation is used to move the data into corresponding Hive table. If the keyword local is specified, then in the load command will give the local file system path. If the keyword local is not specified we have to use the HDFS path of the file.

2. INSERT ... VALUES Statement

The INSERT ... VALUES statement is revised to support adding multiple values into table columns directly from SQL statements. A valid INSERT ... VALUES statement must provide values for each column in the table. However, users may assign null values to columns for which they do not want to assign a value. In addition, the PARTITION clause must be included in the DML.

Syntax :

INSERT INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)] VALUES values_row [, values_row...]

In this syntax, values_row is (value [, value]) and where value is either NULL or any SQL literal.

The following example SQL statements demonstrate several usage variations of this statement:

CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3,2)) CLUSTERED BY (age) INTO 2 BUCKETS STORED AS ORC;

INSERT INTO TABLE students VALUES ('fred flintstone', 35, 1.28), ('barney rubble', 32, 2.32);

CREATE TABLE pageviews (userid VARCHAR(64), link STRING, from STRING) PARTITIONED BY (datestamp STRING) CLUSTERED BY (userid) INTO 256 BUCKETS STORED AS ORC;

INSERT INTO TABLE pageviews PARTITION (datestamp = '2014-09-23') VALUES ('jsmith', 'mail.com', 'sports.com'), ('jdoe', 'mail.com', null);

INSERT INTO TABLE pageviews PARTITION (datestamp) VALUES ('tjohnson', 'sports.com', 'finance.com', '2014-09-23'), ('tlee', 'finance.com', null, '2014-09-21');

3. UPDATE Statement

Use the UPDATE statement to modify data already written to Apache Hive. Depending on the condition specified in the optional WHERE clause, an UPDATE statement may affect every row in a table. You must have both the SELECT and UPDATE privileges to use this statement.

Syntax :

UPDATE tablename SET column = value [, column = value ...] [WHERE expression];

The UPDATE statement has the following limitations:

- The expression in the WHERE clause must be an expression supported by a Hive SELECT clause.
- Partition and bucket columns cannot be updated.
- Query vectorization is automatically disabled for UPDATE statements. However, updated tables can still be queried using vectorization.
- Subqueries are not allowed on the right side of the SET statement.

The following example demonstrates the correct usage of this statement:

UPDATE students SET name = null WHERE gpa <= 1.0;

4. DELETE Statement

Use the DELETE statement to delete data already written to Apache Hive.

Syntax :

DELETE FROM tablename [WHERE expression];

The DELETE statement has the following limitation: query vectorization is automatically disabled for the DELETE operation. However, tables with deleted data can still be queried using vectorization.
The following example demonstrates the correct usage of this statement:
DELETE FROM students WHERE gpa <= 1.0;

HIVE JOIN

Let's see two tables Employee and EmployeeDepartment that are going to be joined.

Employee			Employee Department	
EMP ID	Emp Name	Address	Emp ID	Department
1	Jose	US	1	IT
2	Fred	US	2	IT
3	Jess	In	3	Eng
4	Ajay	In	4	Admin

Inner joins

Select * from employee join employeedepartment ON (employee.empid=employeedepartment.empId)

Left outer joins

Select e.empId, empName, department from employee e Left outer join employeedepartment ed on(e.empId =ed.empId);

Right outer joins

Select e.empId, empName, department from employee e Right outer join employeedepartment ed on(e.empId=ed.empId);

Full outer joins

Select e.empId, empName, department from employee e FULL outer join employeedepartment ed on(e.empId=ed.empId);

Conclusion: Thus we have studied HIVE DML commands.

Experiment No :-6

Aim: Working with operators in Pig- FOREACH, ASSERT, FILTER, GROUP, ORDER BY, DISTINCT, JOIN, LIMIT, SAMPE, SPLIT, FLATTEN.

Pig (programming tool):

Apache Pig is a high-level platform for creating programs that run on Apache Hadoop. The language for this platform is called **Pig Latin**.Pig can execute its Hadoop jobs in MapReduce, Apache Tez, or Apache Spark. Pig Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for RDBMSs. Pig Latin can be extended using User Defined Functions (UDFs) which the user can write in Java, Python, JavaScript, Ruby or Groovy and then call directly from the language.

History:

Developer(s)	Apache Software Foundation, Yahoo Research
Initial release	September 11, 2008; 8 years ago
Stable release	v0.16.0 / June 8, 2016; 8 months ago
Development status	Active
Operating system	Microsoft Windows, OS X, Linux
Type	Data analytics
License	Apache License 2.0
Website	pig.apache.org

Pig vs SQL

In comparison to SQL, Pig

1. Uses lazy evaluation (In programming language theory, lazy evaluation, or call-by-need is an evaluation strategy which delays the evaluation of an expression until its value is needed),
2. Uses extract, transform, load (ETL),
3. Is able to store data at any point during a pipeline,
4. Declares execution plans,
5. Supports pipeline splits, thus allowing workflows to proceed along DAGs(In mathematics and computer science, a directed acyclic graph is a finite directed graph with no directed cycles.) instead of strictly sequential pipelines.

Relational Operators

1. ASSERT

Assert a condition on the data.

Syntax

ASSERT alias BY expression [, message];

Terms

alias The name of the relation.

BY Required keyword.
expression A boolean expression.
message Error message when assertion fails.

Usage

Use assert to ensure a condition is true on your data. Processing fails if any of the records violate the condition.

Examples

Suppose we have relation A.

```
A = LOAD 'data' AS (a0:int,a1:int,a2:int);
DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)
```

Now, you can assert that a0 column in your data is >0, fail if otherwise

```
ASSERT A by a0 > 0, 'a0 should be greater than 0';
```

2. FOREACH

Generates data transformations based on columns of data.

Syntax

```
alias = FOREACH { block | nested_block };
```

Terms

alias	The name of relation (outer bag).
block	FOREACH...GENERATE block used with a relation (outer bag). Use this syntax: alias = FOREACH alias GENERATE expression [AS schema] [expression [AS schema]....];

nested_block	Nested FOREACH...GENERATE block used with a inner bag. Use this syntax:
	alias = FOREACH nested_alias {
	alias = {nested_op | nested_exp}; [{alias = {nested_op | nested_exp}; ...]
	GENERATE expression [AS schema] [expression [AS schema]....]
	};
	Where:
	The nested block is enclosed in opening and closing brackets { ... }.
	The GENERATE keyword must be the last statement within the nested block.
	Macros are NOT allowed inside a nested block.

expression	An expression.
nested_alias	The name of the inner bag.
nested_op	Allowed operations are CROSS, DISTINCT, FILTER, FOREACH, LIMIT, and ORDER BY.

	Note: FOREACH statements can be nested to two levels only. FOREACH statements that are nested to three or more levels will result in a grammar error. You can also perform projections within the nested block.
nested_exp	Any arbitrary, supported expression.
AS	Keyword
schema	A schema using the AS keyword If the FLATTEN operator is used, enclose the schema in parentheses. • If the FLATTEN operator is not used, don't enclose the schema in parentheses.

Usage

Use the FOREACH...GENERATE operation to work with columns of data (if you want to work with tuples or rows of data, use the FILTER operation).

FOREACH...GENERATE works with relations (outer bags) as well as inner bags:

If A is a relation (outer bag), a FOREACH statement could look like this.

```
X = FOREACH A GENERATE f1;
```

If A is an inner bag, a FOREACH statement could look like this.

```
X = FOREACH B {
    S = FILTER A BY 'xyz';
    GENERATE COUNT (S.$0);
}
```

Example: Projection

In this example the asterisk (*) is used to project all fields from relation A to relation X. Relation A and X are identical.

```
X = FOREACH A GENERATE *;

DUMP X;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)
```

In this example two fields from relation A are projected to form relation X.

```
X = FOREACH A GENERATE a1, a2;

DUMP X;
(1,2)
(4,2)
(8,3)
(4,3)
(7,2)
```

(8,4)

3. FILTER

Selects tuples from a relation based on some condition.

Syntax

alias = FILTER alias BY expression;

Terms

alias	The name of the relation.
BY	Required keyword.
expression	A boolean expression.

Usage

Use the FILTER operator to work with tuples or rows of data (if you want to work with columns of data, use the FOREACH...GENERATE operation).

FILTER is commonly used to select the data that you want; or, conversely, to filter out (remove) the data you don't want.

Examples

Suppose we have relation A.

A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)

In this example the condition states that if the third field equals 3, then include the tuple with relation X.

X = FILTER A BY f3 == 3;

DUMP X;
(1,2,3)
(4,3,3)
(8,4,3)

In this example the condition states that if the first field equals 8 or if the sum of fields f2 and f3 is not greater than first field, then include the tuple relation X.

X = FILTER A BY (f1 == 8) OR (NOT (f2+f3 > f1));

DUMP X;
(4,2,1)
(8,3,4)
(7,2,5)

(8,4,3)

4. DISTINCT

Removes duplicate tuples in a relation.

Syntax

alias = DISTINCT alias [PARTITION BY partitioner] [PARALLEL n];

Terms

alias	The name of the relation.
PARTITION BY partitioner	Use this feature to specify the Hadoop Partitioner. The partitioner controls the partitioning of the keys of the intermediate map-outputs.
PARALLEL n	Increase the parallelism of a job by specifying the number of reduce tasks, n.

Usage

Use the DISTINCT operator to remove duplicate tuples in a relation. DISTINCT does not preserve the original order of the contents (to eliminate duplicates, Pig must first sort the data). You cannot use DISTINCT on a subset of fields: to do this, use FOREACH and a nested block to first select the fields and then apply DISTINCT (see Example: Nested Block).

Example

Suppose we have relation A.

A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;
(8,3,4)
(1,2,3)
(4,3,3)
(4,3,3)
(1,2,3)

In this example all duplicate tuples are removed.

X = DISTINCT A;

DUMP X;
(1,2,3)
(4,3,3)
(8,3,4)

5. GROUP

Groups the data in one or more relations.

Syntax

alias = GROUP alias { ALL | BY expression} [, alias ALL | BY expression ...] [USING 'collected' | 'merge'] [PARTITION BY partitioner] [PARALLEL n];

Terms

alias	The name of a relation. You can COGROUP up to but no more than 127 relations at a time.
ALL	Keyword. Use ALL if you want all tuples to go to a single group; for example, when doing aggregates across entire relations. B = GROUP A ALL;
BY	Keyword. Use this clause to group the relation by field, tuple or expression. B = GROUP A BY f1;
expression	A tuple expression. This is the group key or key field. If the result of the tuple expression is a single field, the key will be the value of the first field rather than a tuple with one field. To group using multiple keys, enclose the keys in parentheses: B = GROUP A BY (key1,key2);
USING	Keyword
'collected'	Use the 'collected' clause with the GROUP operation (works with one relation only). The following conditions apply: • The loader must implement the {CollectableLoader} interface. • Data must be sorted on the group key. If your data and loaders satisfy these conditions, use the 'collected' clause to perform an optimized version of GROUP; the operation will execute on the map side and avoid running the reduce phase.
'merge'	Use the 'merge' clause with the COGROUP operation (works with two or more relations only). The following conditions apply: • No other operations can be done between the LOAD and COGROUP statements. • Data must be sorted on the COGROUP key for all tables in ascending (ASC) order. • Nulls are considered smaller than everything. If data contains null keys, they should occur before anything else. • Left-most loader must implement the {CollectableLoader} interface as well as {OrderedLoadFunc} interface. • All other loaders must implement IndexableLoadFunc. • Type information must be provided in the schema for all the loaders. If your data and loaders satisfy these conditions, the 'merge' clause to perform an optimized version of COGROUP; the operation will execute on the map side and avoid running the reduce phase.
PARTITION BY partitioner	Use this feature to specify the Hadoop Partitioner. The partitioner controls the partitioning of the keys of the intermediate map-outputs.
PARALLEL n	Increase the parallelism of a job by specifying the number of reduce tasks, n.

Usage

The GROUP operator groups together tuples that have the same group key (key field). The key field will be a tuple if the group key has more than one field, otherwise it will be the same type as that of the group key. The result of a GROUP operation is a relation that includes one tuple per group. This tuple contains two fields:

- The first field is named "group" (do not confuse this with the GROUP operator) and is the same type as the group key.
- The second field takes the name of the original relation and is type bag.
- The names of both fields are generated by the system as shown in the example below.

Example

Suppose we have relation A.

```
A = load 'student' AS (name:chararray,age:int,gpa:float);

DESCRIBE A;
A: {name: chararray,age: int,gpa: float}

DUMP A:
(John,18,4.0F)
(Mary,19,3.8F)
(Bill,20,3.9F)
(Joe,18,3.8F)
```

Now, suppose we group relation A on field "age" for form relation B. We can use the DESCRIBE and ILLUSTRATE operators to examine the structure of relation B. Relation B has two fields. The first field is named "group" and is type int, the same as field "age" in relation A. The second field is name "A" after relation A and is type bag.

```
B = GROUP A BY age;

DESCRIBE B;
B: {group: int, A: {name: chararray,age: int,gpa: float}}

ILLUSTRATE B;
etc ...
-------------------------------------------------------------------
| B    | group: int | A: bag({name: chararray,age: int,gpa: float})|
-------------------------------------------------------------------
|      | 18         | {(John, 18, 4.0), (Joe, 18, 3.8)}            |
|      | 20         | {(Bill, 20, 3.9)}                            |
-------------------------------------------------------------------

DUMP B:
(18,{(John,18,4.0F),(Joe,18,3.8F)})
(19,{(Mary,19,3.8F)})
(20,{(Bill,20,3.9F)})
```

6. **ORDER BY**

Sorts a relation based on one or more fields.

Syntax

alias = ORDER alias BY { * [ASC|DESC] | field_alias [ASC|DESC] [, field_alias [ASC|DESC] ...] }
[PARALLEL n];

Terms

alias	The name of a relation.
*	The designator for a tuple.
field_alias	A field in the relation. The field must be a simple type.
ASC	Sort in ascending order.
DESC	Sort in descending order.
PARALLEL n	Increase the parallelism of a job by specifying the number of reduce tasks, n.

Usage

Note: ORDER BY is NOT stable; if multiple records have the same ORDER BY key, the order in which these records are returned is not defined and is not guaranteed to be the same from one run to the next.

In Pig, relations are unordered:

- If you order relation A to produce relation X (X = ORDER A BY * DESC;) relations A and X still contain the same data.
- If you retrieve relation X (DUMP X;) the data is guaranteed to be in the order you specified (descending).
- However, if you further process relation X (Y = FILTER X BY $0 > 1;) there is no guarantee that the data will be processed in the order you originally specified (descending).

Pig currently supports ordering on fields with simple types or by tuple designator (*). You cannot order on fields with complex types or by expressions.

```
A = LOAD 'mydata' AS (x: int, y: map[]);
B = ORDER A BY x; -- this is allowed because x is a simple type
B = ORDER A BY y; -- this is not allowed because y is a complex type
B = ORDER A BY y#'id'; -- this is not allowed because y#'id' is an expression
```

Examples

Suppose we have relation A.

```
A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)
```

In this example relation A is sorted by the third field, f3 in descending order. Note that the order of the three tuples ending in 3 can vary.

X = ORDER A BY a3 DESC;

DUMP X;
(7,2,5)
(8,3,4)
(1,2,3)
(4,3,3)
(8,4,3)
(4,2,1)

7. JOIN (inner)

Performs an inner join of two or more relations based on common field values.

Syntax

alias = JOIN alias BY {expression|'('expression [, expression ...]')'} (, alias BY {expression|'('expression [, expression ...]')'} ...) [USING 'replicated' | 'skewed' | 'merge' 'merge-sparse'] [PARTITION BY partitioner] [PARALLEL n];

Terms

alias	The name of a relation.
BY	Keyword
expression	A field expression. Example: X = JOIN A BY fieldA, B BY fieldB, C BY fieldC;
USING	Keyword
'replicated'	Use to perform replicated joins
'skewed'	Use to perform skewed joins
'merge'	Use to perform merge joins
'merge-sparse'	Use to perform merge-sparse joins
PARTITION BY partitioner	Use this feature to specify the Hadoop Partitioner. The partitioner controls the partitioning of the keys of the intermediate map-outputs.
PARALLEL n	Increase the parallelism of a job by specifying the number of reduce tasks, n.

Usage

Use the JOIN operator to perform an inner, equijoin join of two or more relations based on common field values. Inner joins ignore null keys, so it makes sense to filter them out before the join.

8. Self Joins

To perform self joins in Pig load the same data multiple times, under different aliases, to avoid naming conflicts.

In this example the same data is loaded twice using aliases A and B.

grunt> A = load 'mydata';

grunt> B = load 'mydata';
grunt> C = join A by $0, B by $0;
grunt> explain C;

Example

Suppose we have relations A and B.

A = LOAD 'data1' AS (a1:int,a2:int,a3:int);

DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)

B = LOAD 'data2' AS (b1:int,b2:int);

DUMP B;
(2,4)
(8,9)
(1,3)
(2,7)
(2,9)
(4,6)
(4,9)

In this example relations A and B are joined by their first fields.

X = JOIN A BY a1, B BY b1;

DUMP X;
(1,2,3,1,3)
(1,2,1,4,6)
(4,3,3,4,6)
(4,2,1,4,9)
(4,3,3,4,9)
(8,3,4,8,9)
(8,4,3,8,9)

9. JOIN (outer)

Performs an outer join of two relations based on common field values.

Syntax

alias = JOIN left alias BY left-alias-column [LEFT|RIGHT|FULL] [OUTER], right-alias BY right alias column [USING 'replicated' | 'skewed' | 'merge'] [PARTITION BY partitioner] [PARALLEL n];

Terms

alias	The name of a relation. Applies to alias, left-alias and right alias.
alias-column	The name of the join column for the corresponding relation. Applies to left-alias-column and right-alias-column.
BY	Keyword
LEFT	Left outer join.
RIGHT	Right outer join.
FULL	Full outer join.
OUTER	(Optional) Keyword
USING	Keyword
'replicated'	Use to perform replicated joins (see Replicated Joins). Only left outer join is supported for replicated joins.
'skewed'	Use to perform skewed joins (see Skewed Joins).
'merge'	Use to perform merge joins.
PARTITION BY partitioner	Use this feature to specify the Hadoop Partitioner. The partitioner controls the partitioning of the keys of the intermediate map-outputs.
PARALLEL n	Increase the parallelism of a job by specifying the number of reduce tasks, n.

Usage

Use the JOIN operator with the corresponding keywords to perform left, right, or full outer joins. The keyword OUTER is optional for outer joins; the keywords LEFT, RIGHT and FULL will imply left outer, right outer and full outer joins respectively when OUTER is omitted. The Pig Latin syntax closely adheres to the SQL standard.

Please note the following:

- Outer joins will only work provided the relations which need to produce nulls (in the case of non-matching keys) have schemas.
- Outer joins will only work for two-way joins; to perform a multi-way outer join, you will need to perform multiple two-way outer join statements.

Examples

This example shows a left outer join.

```
A = LOAD 'a.txt' AS (n:chararray, a:int);
B = LOAD 'b.txt' AS (n:chararray, m:chararray);
C = JOIN A by $0 LEFT OUTER, B BY $0;
```

This example shows a full outer join.

```
A = LOAD 'a.txt' AS (n:chararray, a:int);
B = LOAD 'b.txt' AS (n:chararray, m:chararray);
C = JOIN A BY $0 FULL, B BY $0;
```

This example shows a replicated left outer join.

A = LOAD 'large';
B = LOAD 'tiny';
C= JOIN A BY $0 LEFT, B BY $0 USING 'replicated';
This example shows a skewed full outer join.
A = LOAD 'studenttab' as (name, age, gpa);
B = LOAD 'votertab' as (name, age, registration, contribution);
C = JOIN A BY name FULL, B BY name USING 'skewed';

10. LIMIT

Limits the number of output tuples.

Syntax

alias = LIMIT alias n;

Terms

alias	The name of a relation.
n	The number of output tuples, either: • a constant (for example, 3) • a scalar used in an expression (for example, c.sum/100) Note: The expression can consist of constants or scalars; it cannot contain any columns from the input relation. Note: Using a scalar instead of a constant in LIMIT automatically disables most optimizations (only push-before-foreach is performed).

Usage

Use the LIMIT operator to limit the number of output tuples.
If the specified number of output tuples is equal to or exceeds the number of tuples in the relation, all tuples in the relation are returned.
If the specified number of output tuples is less than the number of tuples in the relation, then n tuples are returned. There is no guarantee which n tuples will be returned, and the tuples that are returned can change from one run to the next. A particular set of tuples can be requested using the ORDER operator followed by LIMIT.
Note: The LIMIT operator allows Pig to avoid processing all tuples in a relation. In most cases a query that uses LIMIT will run more efficiently than an identical query that does not use LIMIT. It is always a good idea to use limit if you can.

Examples

In this example the limit is expressed as a scalar.
a = load 'a.txt';
b = group a all;
c = foreach b generate COUNT(a) as sum;
d = order a by $0;
e = limit d c.sum/100;
Suppose we have relation A.
A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)

In this example output is limited to 3 tuples. Note that there is no guarantee which three tuples will be output.

X = LIMIT A 3;

DUMP X;
(1,2,3)
(4,3,3)
(7,2,5)

In this example the ORDER operator is used to order the tuples and the LIMIT operator is used to output the first three tuples.

B = ORDER A BY f1 DESC, f2 ASC;

DUMP B;
(8,3,4)
(8,4,3)
(7,2,5)
(4,2,1)
(4,3,3)
(1,2,3)

X = LIMIT B 3;

DUMP X;
(8,3,4)
(8,4,3)
(7,2,5)

11. SAMPLE

Selects a random sample of data based on the specified sample size.

Syntax

SAMPLE alias size;

Terms

alias	The name of a relation.
size	Sample size, either • a constant, range 0 to 1 (for example, enter 0.1 for 10%) • a scalar used in an expression

	Note: The expression can consist of constants or scalars; it cannot contain any columns from the input relation.

Usage

Use the SAMPLE operator to select a random data sample with the stated sample size. SAMPLE is a probabalistic operator; there is no guarantee that the exact same number of tuples will be returned for a particular sample size each time the operator is used.

Example

In this example relation X will contain 1% of the data in relation A.

```
A = LOAD 'data' AS (f1:int,f2:int,f3:int);

X = SAMPLE A 0.01;
```

In this example, a scalar expression is used (it will sample approximately 1000 records from the input).

```
a = load 'a.txt';
b = group a all;
c = foreach b generate COUNT(a) as num_rows;
e = sample a 1000/c.num_rows;
```

12. SPLIT

Partitions a relation into two or more relations.

Syntax

SPLIT alias INTO alias IF expression, alias IF expression [, alias IF expression ...] [, alias OTHERWISE];

Terms

alias	The name of a relation.
INTO	Required keyword.
IF	Required keyword.
expression	An expression.
OTHERWISE	Optional keyword. Designates a default relation.

Usage

Use the SPLIT operator to partition the contents of a relation into two or more relations based on some expression. Depending on the conditions stated in the expression:

- A tuple may be assigned to more than one relation.
- A tuple may not be assigned to any relation.

Example

In this example relation A is split into three relations, X, Y, and Z.

```
A = LOAD 'data' AS (f1:int,f2:int,f3:int);

DUMP A;
(1,2,3)
(4,5,6)
(7,8,9)
```

SPLIT A INTO X IF f1<7, Y IF f2==5, Z IF (f3<6 OR f3>6);

DUMP X;
(1,2,3)
(4,5,6)

DUMP Y;
(4,5,6)

DUMP Z;
(1,2,3)
(7,8,9)

Flatten Operator

The FLATTEN operator looks like a UDF syntactically, but it is actually an operator that changes the structure of tuples and bags in a way that a UDF cannot. Flatten un-nests tuples as well as bags. The idea is the same, but the operation and result is different for each type of structure.

For tuples, flatten substitutes the fields of a tuple in place of the tuple. For example, consider a relation that has a tuple of the form (a, (b, c)). The expression GENERATE $0, flatten($1), will cause that tuple to become (a, b, c).

For bags, the situation becomes more complicated. When we un-nest a bag, we create new tuples. If we have a relation that is made up of tuples of the form ({(b,c),(d,e)}) and we apply GENERATE flatten($0), we end up with two tuples (b,c) and (d,e). When we remove a level of nesting in a bag, sometimes we cause a cross product to happen. For example, consider a relation that has a tuple of the form (a, {(b,c), (d,e)}), commonly produced by the GROUP operator. If we apply the expression GENERATE $0, flatten($1) to this tuple, we will create new tuples: (a, b, c) and (a, d, e).

Also note that the flatten of empty bag will result in that row being discarded; no output is generated.

grunt> cat empty.bag
{} 1
grunt> A = LOAD 'empty.bag' AS (b : bag{}, i : int);
grunt> B = FOREACH A GENERATE flatten(b), i;
grunt> DUMP B;
grunt>

Null Operators

Description

Operator	Symbol	Notes
is null	is null	
is not null	is not null	

Examples

In this example, values that are not null are obtained.

X = FILTER A BY f1 is not null;

Types Table

The null operators can be applied to all data types

Sign Operators

Description

Operator	Symbol	Notes
positive	+	Has no effect.
negative (negation)	-	Changes the sign of a positive or negative number.

Examples

In this example, the negation operator is applied to the "x" values.

A = LOAD 'data' as (x, y, z);

B = FOREACH A GENERATE -x, y;

Types Table: negative (-) operator

bag	error
tuple	error
map	error
int	int
long	long
float	float
double	double
chararray	error
bytearray	double (as double)
datetime	error
biginteger	biginteger
bigdecimal	bigdecimal

Conclusion: Thus we have studied Pig commands.

Experiment No :7

Aim : Study of R-declaring variables, expressions, functions and executing R script

R - Overview

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac.

R is free software distributed under a GNU-style copy left, and an official part of the GNU project called **GNU S.**

Features of R

As stated earlier, R is a programming language and software environment for statistical analysis, graphics representation and reporting. The following are the important features of R −

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility,
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

Local Environment Setup

If you are still willing to set up your environment for R, you can follow the steps given below.

Windows Installation

You can download the Windows installer version of R from R-3.2.2 for Windows (32/64 bit) and save it in a local directory.

As it is a Windows installer (.exe) with a name "R-version-win.exe". You can just double click and run the installer accepting the default settings. If your Windows is 32-bit version, it installs the 32-bit version. But if your windows is 64-bit, then it installs both the 32-bit and 64-bit versions.

After installation you can locate the icon to run the Program in a directory structure "R\R3.2.2\bin\i386\Rgui.exe" under the Windows Program Files. Clicking this icon brings up the R-GUI which is the R console to do R Programming.

R Command Prompt

Once you have R environment setup, then it's easy to start your R command prompt by just typing the following command at your command prompt −

$ R

This will launch R interpreter and you will get a prompt > where you can start typing your program as follows −

```
> myString <- "Hello, World!"
```

> print (myString)
[1] "Hello, World!"
Here first statement defines a string variable myString, where we assign a string "Hello, World!" and then next statement print() is being used to print the value stored in variable myString.

R Script File

Usually, you will do your programming by writing your programs in script files and then you execute those scripts at your command prompt with the help of R interpreter called **Rscript**. So let's start with writing following code in a text file called test.R as under –

```
# My first program in R Programming
myString <- "Hello, World!"

print ( myString)
```

Save the above code in a file test.R and execute it at Linux command prompt as given below. Even if you are using Windows or other system, syntax will remain same.

```
$ Rscript test.R
```

When we run the above program, it produces the following result.

```
[1] "Hello, World!"
```

Comments

Comments are like helping text in your R program and they are ignored by the interpreter while executing your actual program. Single comment is written using # in the beginning of the statement as follows –

```
# My first program in R Programming
```

R - Data Types

In contrast to other programming languages like C and java in R, the variables are not declared as some data type. The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable. There are many types of R-objects. The frequently used ones are – .

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

The simplest of these objects is the **vector object** and there are six data types of these atomic vectors, also termed as six classes of vectors. The other R-Objects are built upon the atomic vectors.

The simplest of these objects is the **vector object** and there are six data types of these atomic vectors, also termed as six classes of vectors. The other R-Objects are built upon the atomic vectors.

1. Logical
2. Numeric
3. Integer
4. Complex
5. Character
6. Raw

Vectors

When you want to create vector with more than one element, you should use c() function which means to combine the elements into a vector.

```
# Create a vector.
apple <- c('red','green',"yellow");
print(apple)

# Get the class of the vector.
print(class(apple))
```

When we execute the above code, it produces the following result −

```
[1] "red"    "green"  "yellow"
[1] "character"
```

Lists

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```
# Create a list.
list1 <- list(c(2,5,3),21.3,sin)

# Print the list.
print(list1)
```

When we execute the above code, it produces the following result −

```
[[1]]
[1] 2 5 3

[[2]]
[1] 21.3

[[3]]
function (x)  .Primitive("sin")
```

Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
# Create a matrix.
M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
print(M)
```

When we execute the above code, it produces the following result −

```
     [,1] [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"
```

Arrays

While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimension. In the below example we create an array with two elements which are 3x3 matrices each.

```
# Create an array.
a <- array(c('green','yellow'),dim = c(3,3,2))
print(a)
```

When we execute the above code, it produces the following result :

```
, , 1

     [,1]   [,2]   [,3]
[1,] "green"  "yellow" "green"
[2,] "yellow" "green"  "yellow"
[3,] "green"  "yellow" "green"

, , 2

     [,1]   [,2]   [,3]
[1,] "yellow" "green"  "yellow"
[2,] "green"  "yellow" "green"
[3,] "yellow" "green"  "yellow"
```

Factors

Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling.

Factors are created using the **factor()** function. The **nlevels** functions gives the count of levels.

```
# Create a vector.
apple_colors <- c('green','green','yellow','red','red','red','green')

# Create a factor object.
factor_apple <- factor(apple_colors)

# Print the factor.
print(factor_apple)
print(nlevels(factor_apple))
```

When we execute the above code, it produces the following result

```
[1] green  green  yellow red    red    red    yellow green
Levels: green red yellow
# applying the nlevels function we can know the number of distinct values
[1] 3
```

Data Frames

Data frames are tabular data objects. Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.

Data Frames are created using the **data.frame()** function.

```
# Create the data frame.
BMI <-   data.frame(
```

```
  gender = c("Male", "Male","Female"),
  height = c(152, 171.5, 165),
  weight = c(81,93, 78),
  Age = c(42,38,26)
)
print(BMI)
```

When we execute the above code, it produces the following result −

```
  gender height weight Age
1   Male  152.0     81  42
2   Male  171.5     93  38
3 Female  165.0     78  26
```

Variable Assignment

The variables can be assigned values using leftward, rightward and equal to operator. The values of the variables can be printed using **print()** or **cat()**function. The **cat()** function combines multiple items into a continuous print output.

```
# Assignment using equal operator.
var.1 = c(0,1,2,3)

# Assignment using leftward operator.
var.2 <- c("learn","R")

# Assignment using rightward operator.
c(TRUE,1) -> var.3

print(var.1)
cat ("var.1 is ", var.1 ,"\n")
cat ("var.2 is ", var.2 ,"\n")
cat ("var.3 is ", var.3 ,"\n")
```

When we execute the above code, it produces the following result

```
[1] 0 1 2 3
var.1 is  0 1 2 3
var.2 is  learn R
var.3 is  1 1
```

R - Decision making

R provides the following types of decision making statements. Click the following links to check their detail.

Sr.No.	Statement & Description
1	if statement An **if** statement consists of a Boolean expression followed by one or more statements.
2	if...else statement An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.

3	switch statement A switch statement allows a variable to be tested for equality against a list of values.

R - Loops

Sr. No.	Loop Type & Description
1	repeat loop Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
2	while loop Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
3	for loop Like a while statement, except that it tests the condition at the end of the loop body.

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

R supports the following control statements. Click the following links to check their detail.

Sr.No.	Control Statement & Description
1	break statement Terminates the **loop** statement and transfers execution to the statement immediately following the loop.
2	Next statement The **next** statement simulates the behavior of R switch.

R - Functions

A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions.

Function Definition

An R function is created by using the keyword **function**. The basic syntax of an R function definition is as follows

```
function_name <- function(arg_1, arg_2, ...) {
  Function body
}
```

Function Components

The different parts of a function are –

- **Function Name** – This is the actual name of the function. It is stored in R environment as an object with this name.
- **Arguments** – An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.
- **Function Body** – The function body contains a collection of statements that defines what the function does.

- **Return Value** – The return value of a function is the last expression in the function body to be evaluated.

R has many **in-built** functions which can be directly called in the program without defining them first. We can also create and use our own functions referred as **user defined** functions.

Built-in Function

Simple examples of in-built functions are **seq()**, **mean()**, **max()**, **sum(x)** and **paste(...)** etc. They are directly called by user written programs.

```
# Create a sequence of numbers from 32 to 44.
print(seq(32,44))

# Find mean of numbers from 25 to 82.
print(mean(25:82))

# Find sum of numbers frm 41 to 68.
print(sum(41:68))
```

When we execute the above code, it produces the following result:

```
[1] 32 33 34 35 36 37 38 39 40 41 42 43 44
[1] 53.5
[1] 1526
```

User-defined Function

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below is an example of how a function is created and used.

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
  for(i in 1:a) {
    b <- i^2
    print(b)
  }
}
```

Calling a Function

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
  for(i in 1:a) {
    b <- i^2
    print(b)
  }
}

# Call the function new.function supplying 6 as an argument.
new.function(6)
```

When we execute the above code, it produces the following result:

```
[1] 1
[1] 4
```

[1] 9
[1] 16
[1] 25
[1] 36

Calling a Function without an Argument

```
# Create a function without an argument.
new.functinn <- function() {
  for(i in 1:5) {
    print(i^2)
  }
}

# Call the function without supplying an argument.
new.function()
```

When we execute the above code, it produces the following result −

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```

Calling a Function with Argument Values (by position and by name)

The arguments to a function call can be supplied in the same sequence as defined in the function or they can be supplied in a different sequence but assigned to the names of the arguments.

```
# Create a function with arguments.
new.function <- function(a,b,c) {
  result <- a * b + c
  print(result)
}

# Call the function by position of arguments.
new.function(5,3,11)

# Call the function by names of the arguments.
new.function(a = 11, b = 5, c = 3)
```

When we execute the above code, it produces the following result −

```
[1] 26
[1] 58
```

Calling a Function with Default Argument

We can define the value of the arguments in the function definition and call the function without supplying any argument to get the default result. But we can also call such functions by supplying new values of the argument and get non default result.

```
# Create a function with arguments.
new.function <- function(a = 3, b = 6) {
```

```
  result <- a * b
  print(result)
}

# Call the function without giving any argument.
new.function()

# Call the function with giving new values of the argument.
new.function(9,5)
```

When we execute the above code, it produces the following result

```
[1] 18
[1] 45
```

Conclusion: Thus we have studied R-declaring variables, expressions, functions and executing R script.

Experiment no -8

Aim : Working with R data sets- create, read, write.

Overview

Here, we will learn to create read data from a csv file and then write data into a csv file. The file should be present in current working directory so that R can read it. Of course we can also set our own directory and read files from there.

Getting and Setting the Working Directory

You can check which directory the R workspace is pointing to using the **getwd** function. You can also set a new working directory using **setwd** function.

```
# Get and print current working directory.
print(getwd())

# Set current working directory.
setwd("/web/com")

# Get and print current working directory.
print(getwd())
```

When we execute the above code, it produces the following result −

```
[1] "/web/com/1441086124_2016"
[1] "/web/com"
```

This result depends on your OS and your current directory where you are working.

Creating (Input as) CSV File

The csv file is a text file in which the values in the columns are separated by a comma. Let's consider the following data present in the file named **input.csv**.

You can create this file using windows notepad by copying and pasting this data. Save the file as **input.csv** using the save As All files*.* option in notepad.

```
id,name,salary,start_date,dept
1,Rick,623.3,2012-01-01,IT
2,Dan,515.2,2013-09-23,Operations
3,Michelle,611,2014-11-15,IT
4,Ryan,729,2014-05-11,HR
5,Gary,843.25,2015-03-27,Finance
6,Nina,578,2013-05-21,IT
7,Simon,632.8,2013-07-30,Operations
8,Guru,722.5,2014-06-17,Finance
```

Reading a CSV File

Following is a simple example of **read.csv** function to read a CSV file available in your current working directory −

```
data <- read.csv("input.csv")
print(data)
```

When we execute the above code, it produces the following result –

```
  id, name, salary, start_date, dept
1  1  Rick  623.30  2012-01-01  IT
2  2  Dan  515.20  2013-09-23  Operations
3  3  Michelle 611.00  2014-11-15  IT
4  4  Ryan  729.00  2014-05-11  HR
5  NA  Gary  843.25  2015-03-27  Finance
6  6  Nina  578.00  2013-05-21  IT
7  7  Simon  632.80  2013-07-30  Operations
8  8  Guru  722.50  2014-06-17  Finance
```

Analyzing the CSV File

By default the **read.csv** function gives the output as a data frame. This can be easily checked as follows. Also we can check the number of columns and rows.

```
data <- read.csv("input.csv")

print(is.data.frame(data))
print(ncol(data))
print(nrow(data))
```

When we execute the above code, it produces the following result –

```
[1] TRUE
[1] 5
[1] 8
```

Once we read data in a data frame, we can apply all the functions applicable to data frames as explained in subsequent section.

1. Get the maximum salary

```
# Create a data frame.
data <- read.csv("input.csv")

# Get the max salary from data frame.
sal <- max(data$salary)
print(sal)
```

When we execute the above code, it produces the following result –

```
[1] 843.25
```

2. Get the details of the person with max salary

We can fetch rows meeting specific filter criteria similar to a SQL where clause.

```
# Create a data frame.
data <- read.csv("input.csv")

# Get the max salary from data frame.
sal <- max(data$salary)

# Get the person detail having max salary.
```

```
retval <- subset(data, salary == max(salary))
print(retval)
```

When we execute the above code, it produces the following result –

```
  id  name  salary  start_date   dept
5  NA  Gary  843.25  2015-03-27  Finance
```

3. Get all the people working in IT department

```
# Create a data frame.
data <- read.csv("input.csv")

retval <- subset( data, dept == "IT")
print(retval)
```

When we execute the above code, it produces the following result –

```
  id  name    salary  start_date  dept
1  1  Rick    623.3   2012-01-01  IT
3  3  Michelle 611.0  2014-11-15  IT
6  6  Nina    578.0   2013-05-21  IT
```

4. Get the persons in IT department whose salary is greater than 600

```
# Create a data frame.
data <- read.csv("input.csv")

info <- subset(data, salary > 600 & dept == "IT")
print(info)
```

When we execute the above code, it produces the following result –

```
  id  name    salary  start_date  dept
1  1  Rick    623.3   2012-01-01  IT
3  3  Michelle 611.0  2014-11-15  IT
```

5. Get the people who joined on or after 2014

```
# Create a data frame.
data <- read.csv("input.csv")

retval <- subset(data, as.Date(start_date) > as.Date("2014-01-01"))
print(retval)
```

When we execute the above code, it produces the following result –

```
  id  name    salary  start_date  dept
3  3  Michelle 611.00  2014-11-15  IT
4  4  Ryan    729.00  2014-05-11  HR
5  NA  Gary    843.25  2015-03-27  Finance
8  8  Guru    722.50  2014-06-17  Finance
```

Writing into a CSV File

R can create csv file form existing data frame. The **write.csv** function is used to create the csv file. This file gets created in the working directory.

```
# Create a data frame.
data <- read.csv("input.csv")
```

```
retval <- subset(data, as.Date(start_date) > as.Date("2014-01-01"))

# Write filtered data into a new file.
write.csv(retval,"output.csv")
newdata <- read.csv("output.csv")
print(newdata)
```

When we execute the above code, it produces the following result −

```
  X  id  name    salary  start_date   dept
1 3   3   Michelle  611.00  2014-11-15   IT
2 4   4   Ryan    729.00  2014-05-11   HR
3 5   NA  Gary    843.25  2015-03-27   Finance
4 8   8   Guru    722.50  2014-06-17   Finance
```

Here the column X comes from the data set newper. This can be dropped using additional parameters while writing the file.

```
# Create a data frame.
data <- read.csv("input.csv")
retval <- subset(data, as.Date(start_date) > as.Date("2014-01-01"))

# Write filtered data into a new file.
write.csv(retval,"output.csv", row.names = FALSE)
newdata <- read.csv("output.csv")
print(newdata)
```

When we execute the above code, it produces the following result −

```
   id   name    salary  start_date   dept
1   3   Michelle  611.00  2014-11-15   IT
2   4   Ryan    729.00  2014-05-11   HR
3   NA  Gary    843.25  2015-03-27   Finance
4   8   Guru    722.50  2014-06-17   Finance
```

Conclusion: Thus we have studied and implemented R data sets

Experiment No: 9

Aim: Manipulating and processing data in R- merging datasets, sorting data, putting data into shape, managing data using matrices managing data using data frames.

Overview

R provides a different way to sort the data either in ascending or descending order; Data-analysts, and Data scientists use order(), sort() and packages like dplyr to sort data depending upon the structure of the obtained data.

order() can sort vector, matrix, and also a dataframe can be sorted in ascending and descending order with its help, which is shown in the final section of this tutorial.

Syntax of order()

The syntax of order() is shown below:

order(x, decreasing = TRUE or FALSE, na.last = TRUE or FLASE, method = c("auto", "shell", "quick", "radix"))

The argument above in order() states that:

x: data-frames, matrices, or vectors

- **decreasing:** boolean value; TRUE then sort in descending order or FALSE then sort in ascending order.
- **na.last:** boolean value; TRUE then NA indices are put at last or FLASE THEN NA indices are put first.
- **method:** sorting method to be used.

Order() in R

Let's look at an example of order() in action.

Below the code contains variable x, which includes a vector with a list of numbers. The numbers are ordered according to its index by using order(x).

```
y = c(4,12,6,7,2,9,5)
order(y)
```

The above code gives the following output:

5 1 7 3 4 6 2

Here the order() will sort the given numbers according to its index in the ascending order. Since number 2 is the smallest, which has an index as five and number 4 is index 1, and similarly, the process moves forward in the same pattern.

```
y = c(4,12,6,7,2,9,5)
y[order(y)]
```

The above code gives the following output:

2 4 5 6 7 9 12

Here the indexing of order is done where the actual values are printed in the ascending order. The values are ordered according to the index using order() then after each value accessed using y[some value].

Sorting vector using different parameters in order()

Let's look at an example where the datasets contain the value as symbol NA(Not available).

order(x,na.last=TRUE)

```
x <- c(8,2,4,1,-4,NA,46,8,9,5,3)
order(x,na.last = TRUE)
```

The above code gives the following output:

5 4 2 11 3 10 1 8 9 7 6

Here the order() will also sort the given list of numbers according to its index in the ascending order. Since NA is present, its index will be placed last, where 6 will be placed last because of na.last=TRUE.

order(x,na.last=FALSE)

order(x,na.last=FALSE)

The above code gives the following output:

6 5 4 2 11 3 10 1 8 9 7

Here the order() will also sort the given list of numbers according to its index in the ascending order. Since NA is present, it's index, which is 6, will be placed first because of na.last=FALSE.

order(x,decreasing=TRUE,na.last=TRUE)

order(x,decreasing=TRUE,na.last=TRUE)

The above code gives the following output:

,7 9 1 8 10 3 11 2 4 5 6

Here order() will sort a given list of numbers according to its index in the descending order because of decreasing=TRUE: -6. The largest is placed at index 7, and the other values are arranged in a decreasing manner. Since NA is present, index 6 will be placed last because of na.last=TRUE.

order(x,decreasing=FALSE,na.last=FALSE)

order(x,decreasing=FALSE,na.last=FALSE)

The above code gives the following output:

6 5 4 2 11 3 10 1 8 9 7

Here NA is present which index is 6 will be placed at first because of na.last=FALSE. order() will sort a given list of numbers according to its index in the ascending order because of decreasing=FALSE: -4, which is smallest placed at index 5, and the other values are arranged increasingly.

Sorting a dataframe by using order()

Let's create a dataframe where the population value is 10. The variable gender consists of vector values 'male' and 'female' where 10 sample values could be obtained with the help of sample(), whereas replace = TRUE will generate only the unique values. Similarly, the age consists of value from 25 to 75, along with a degree of possible value as c("MA," "ME," "BE," "BSCS"), which again will generate unique values.

Task: To sort the given data in the ascending order based on the given population's age.

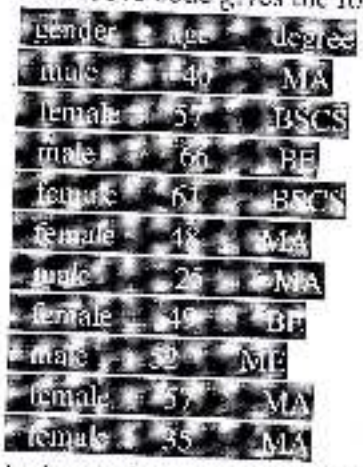
Note: The sample data shown may differ while you're trying to use it in your local machine because each time running a code will create a unique dataframe.

```
population = 10
gender=sample(c("male","female"),population,replace=TRUE)
age = sample(25:75, population, replace=TRUE)
degree = sample(c("MA","ME", "BE","BSCS"), population, replace=TRUE)
(final.data = data.frame(gender=gender, age=age, degree=degree))
```

	gender	age	degree
	male	40	MA

gender	age	degree
female	57	BSCS
male	66	BE
female	61	BSCS
female	48	MA
male	25	MA
female	49	BE
male	52	ME
female	57	MA
female	35	MA

The above code gives the following output, which shows a newly created dataframe.



Let's sort the dataframe in the ascending order by using order() based on the variable age.

The merge function in R allows you to combine two data frames, much like the join function that is used in SQL to combine data tables. Merge, however, does not allow for more than two data frames to be joined at once, requiring several lines of code to join multiple data frames.

The merge function

As described, merge is essentially the "join" of the R world. Whilst this post is not about the fine workings of merge.

Merge takes two data frames, x and y, and combines them based on one or more shared columns. Rows are combined where the data of these shared columns are equal, meaning we can combine columns from different data frames that refer to the same piece of data. For instance, take the following two data frames:

```
height <- data.frame("Character" = c("Luke", "Han", "Leia"),
                     "height" = c("1.75m", "1.85m", "1.5m"))
gender <- data.frame("Character" = c("Luke", "Han", "Leia"),
                     "gender" = c("m", "m", "f"))
```

It is clear that the two data frames are referring to the same characters, however it may be more useful to us if the two were combined into a single data frame. This is where merge comes in. Merge takes the following structure:

```
merge(x = height, y = gender, by = "Character")
```

Here, we are looking to combine the height and gender data frames where the character columns are equal. To continue the SQL analogy, x is the left-hand table, y is the right-hand table, and merge is the LEFT JOIN operation. The "by" component is our "ON" clause. For example:

```
SELECT *
FROM height
LEFT JOIN gender
ON height.column1 = gender.column1
```

Running this merge function gives us the following output:

	Character	Height	Gender
1	Han	1.85m	m
2	Leia	1.5m	f
3	Luke	1.75m	m

This is the result we were expecting, but what if we introduce a third data frame?

```
eyeColour <- data.frame("Character" = c("Luke","Han","Leia"),
                        "eye_colour" = c("Brown", "Blue", "Brown"))
```

Sadly, merge does not allow us to simply add our eyeColour data frame as a third input (we only have x and y parameters available). That's where Reduce comes in.

The Reduce function

Reduce takes a function and sequentially applies it to a given list of inputs, in our case a list of data frames. For example, imagine we have a function f which accepts two arguments, and a list of objects (a, b, c). Then Reduce(x, list(a, b, c)) would perform the following action:

f(a, f(b, c))

where the function x is first applied to data frames b and c, and is then applied to data frame a and the output of the first application of x. This allows us to avoid running and saving x(b, c), like this:

output_1 <- x(b, c)

output_2 <- x(a, output_1)

Applying Reduce to merge

In merge we have an example of a function that performs an action on two inputs. Reduce takes two parameters: f which stands for function and x which represents a vector. Reduce will sequentially apply the function f to the list x.

In our example, the function that we want to apply is merge, and the vector which we want to apply it to is a list of our data frames. First off, let's try the following:

Reduce(merge, list(height, gender, eyeColour))

	Character	height	gender	eye_colour
1	Han	1.65m	m	Blue
2	Leia	1.5m	f	Brown
3	Luke	1.75m	m	Brown

Perfect! But what if we wanted to specify the parameters within our merge function call? Well, we could define our own function which merges two data frames with specified parameters:

```
Reduce(function(x,y) merge(x = x, y = y, by = "Character"),
    list(height, gender, eyeColour))
```

Here, we have specified our f as a custom function, which takes two parameters and applies the merge function to them. Within this custom function, we have specified our by parameter, which may be necessary for longer or more complex uses of Reduce.

Conclusion: Thus we have studied Manipulating and processing data in R.

Expt No 10

Title: Installation and configuration of Apache Spark on Local Machine.

Aim: **Installation and configuration of Apache Spark on Local Machine.**

Introduction

Apache Spark is a framework used in cluster computing environments for **analyzing big data.** This platform became widely popular due to its ease of use and the improved data processing speeds over Hadoop.

Apache Spark is able to distribute a workload across a group of computers in a cluster to more effectively process large sets of data. This **open-source engine** supports a wide array of programming languages. This includes Java, Scala, Python, and R.

In this tutorial, you will **learn how to install Spark on an Ubuntu machine.** The guide will show you how to start a master and slave server and how to load Scala and Python shells. It also provides the most important Spark commands.

Prerequisites

- An Ubuntu system.
- Access to a terminal or command line.
- A user with sudo or **root** permissions.

Install Packages Required for Spark

Before downloading and setting up Spark, you need to install necessary dependencies. This step includes installing the following packages:

- JDK
- Scala
- Git

Open a terminal window and run the following command to install all three packages at once:

```
sudo apt install default-jdk scala git -y
```

You will see which packages will be installed.

Once the process completes, **verify the installed dependencies** by running these commands:

```
java -version; javac -version; scala -version; git --version
```

The output prints the versions if the installation completed successfully for all packages.

Download and Set Up Spark on Ubuntu

Now, **you need to download the version of Spark you want** form their website. We will go for *Spark 3.0.1* with *Hadoop 2.7* as it is the latest version at the time of writing this article.

Use the **wget** command and the direct link to download the Spark archive:

```
wget https://downloads.apache.org/spark/spark-3.0.1/spark-3.0.1-bin-hadoop2.7.tgz
```

When the download completes, you will see the *saved* message.

Note: If the URL does not work, please go to the Apache Spark download page to check for the latest version. Remember to replace the Spark version number in the subsequent commands if you change the download URL.

Now, extract the saved archive using tar:

```
tar xvf spark-*
```

Let the process complete. The output shows the files that are being unpacked from the archive.

Finally, move the unpacked directory *spark-3.0.1-bin-hadoop2.7* to the ***opt/spark*** directory.

Use the **mv** command to do so:

```
sudo mv spark-3.0.1-bin-hadoop2.7 /opt/spark
```

The terminal returns no response if it successfully moves the directory. If you mistype the name, you will get a message similar to:

```
mv: cannot stat 'spark-3.0.1-bin-hadoop2.7': No such file or directory
```

Configure Spark Environment

Before starting a master server, you need to configure environment variables. There are a few Spark home paths you need to add to the user profile.

Use the **echo** command to add these three lines to *.profile*:

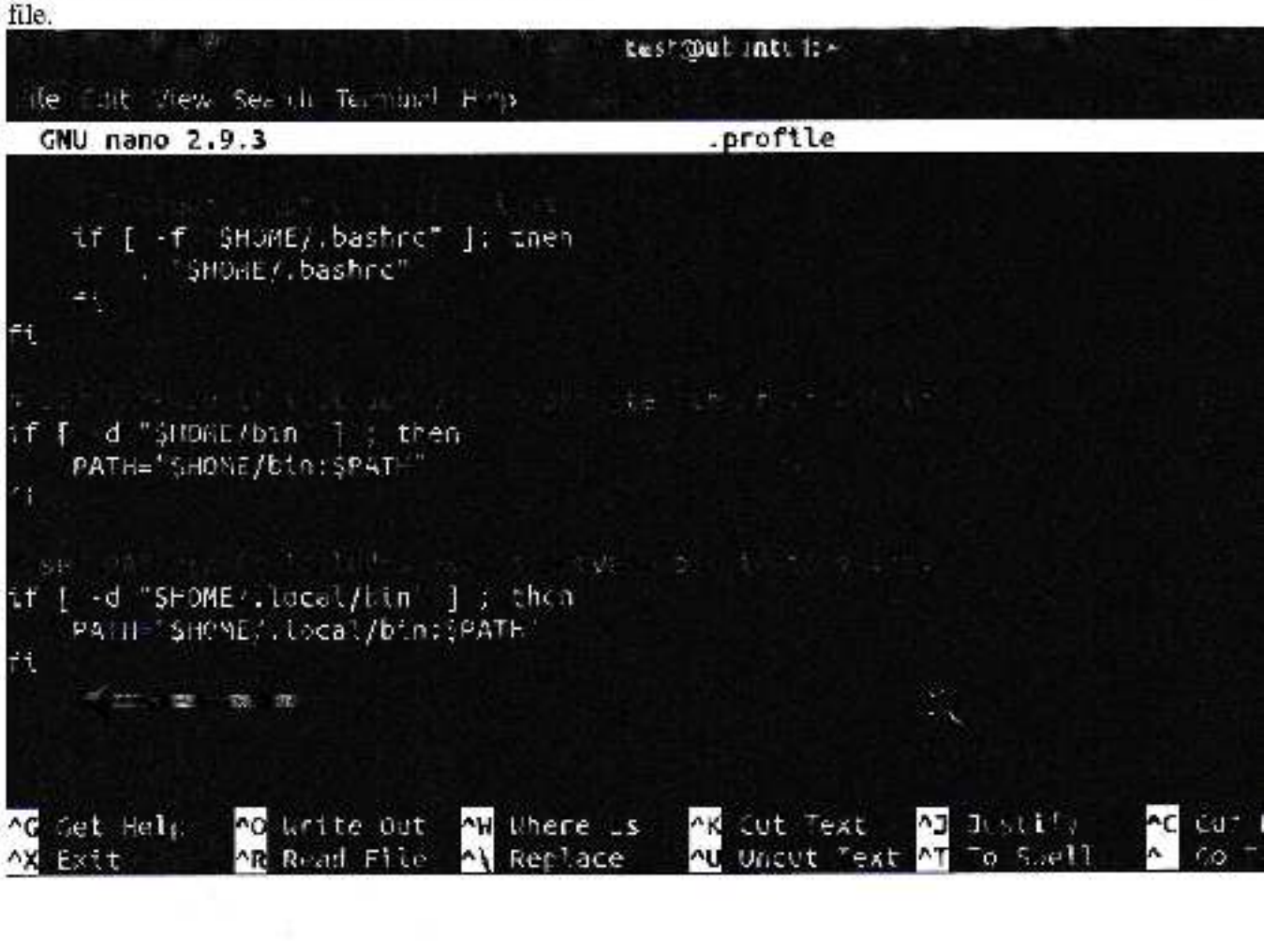
```
echo "export SPARK_HOME=/opt/spark" >> ~/.profile
echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin" >> ~/.profile
echo "export PYSPARK_PYTHON=/usr/bin/python3" >> ~/.profile
```

You can also add the export paths by editing the *.profile* file in the editor of your choice, such as nano or vim.

For example, to use nano, enter:

```
nano .profile
```

When the profile loads, scroll to the bottom of the file.



Then, add these three lines:

```
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
export PYSPARK_PYTHON=/usr/bin/python3
```

Exit and save changes when prompted.

When you finish adding the paths, load the *.profile* file in the command line by typing:

```
source ~/.profile
```

Start Standalone Spark Master Server

Now that you have completed configuring your environment for Spark, you can start a master server.

In the terminal, type:

```
start-master.sh
```

To view the Spark Web user interface, open a web browser and enter the localhost IP address on port 8080.

```
http://127.0.0.1:8080/
```

The page shows your **Spark URL**, status information for workers, hardware resource utilization, etc.

The URL for Spark Master is the name of your device on port 8080. In our case, this is ***ubuntu1:8080***. So, there are three possible ways to load Spark Master's Web UI:

1. 127.0.0.1:8080
2. localhost:8080
3. *deviceName*:8080

Note: Learn how to automate the deployment of Spark clusters on Ubuntu servers by reading our Automated Deployment Of Spark Cluster On Bare Metal Cloud article.

Start Spark Slave Server (Start a Worker Process)

In this single-server, standalone setup, we will start one slave server along with the master server.

To do so, run the following command in this format:

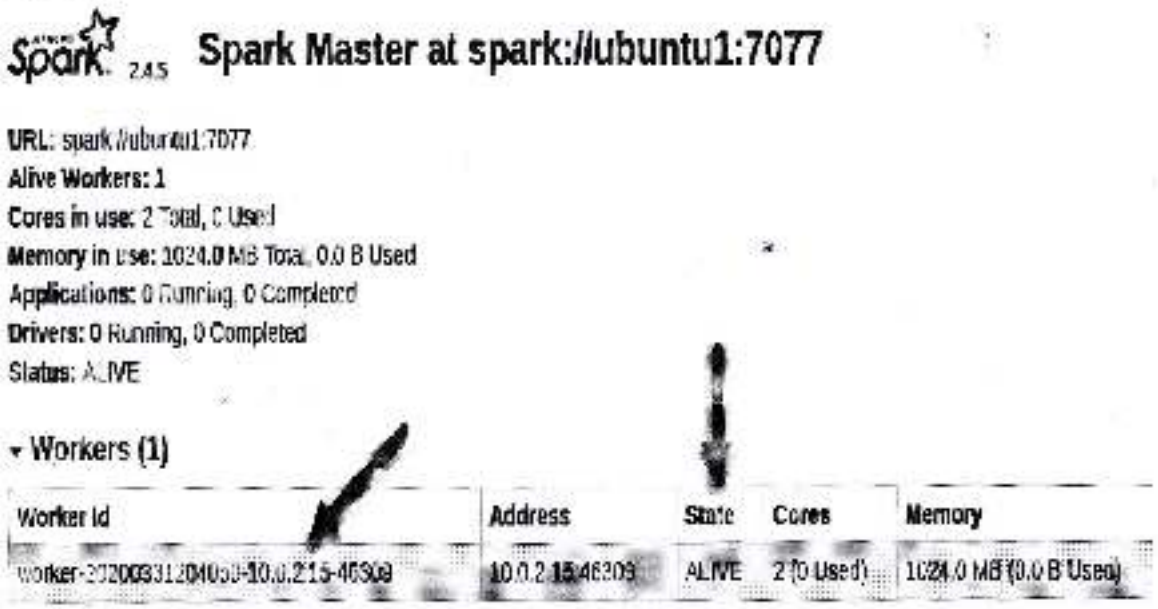
```
start-slave.sh spark://master:port
```

The **master** in the command can be an IP or hostname.

In our case it is **ubuntu1**:

```
start-slave.sh spark://ubuntu1:7077
```

Now that a worker is up and running, if you reload Spark Master's Web UI, you should see it on the list:



Specify Resource Allocation for Workers

The default setting when starting a worker on a machine is to use all available CPU cores. You can specify the number of cores by passing the -c flag to the **start-slave** command.

For example, to start a worker and assign only **one CPU core** to it, enter this command:

```
start-slave.sh -c 1 spark://ubuntu1:7077
```

Reload Spark Master's Web UI to confirm the worker's configuration.

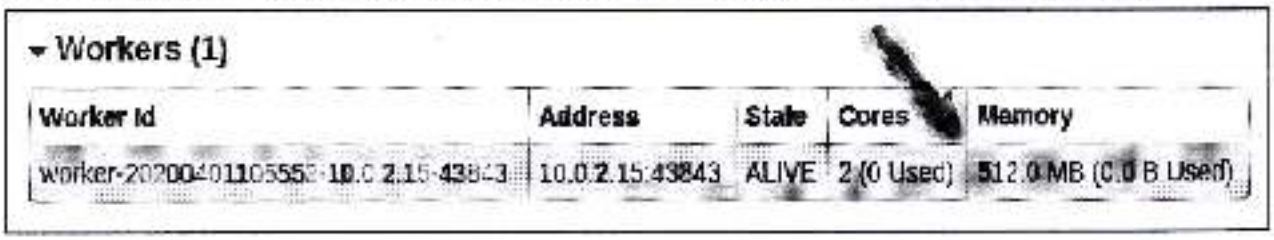
Similarly, you can assign a specific amount of memory when starting a worker. The default setting is to use whatever amount of RAM your machine has, minus 1GB.

To start a worker and assign it a specific amount of memory, add the -m option and a number. For gigabytes, use G and for megabytes, use M.

For example, to start a worker with 512MB of memory, enter this command:

```
start-slave.sh -m 512M spark://ubuntu1:7077
```

Reload the Spark Master Web UI to view the worker's status and confirm the configuration.



Test Spark Shell

After you finish the configuration and start the master and slave server, test if the Spark shell

Conclusion: Hence we studied how to **Install** and configure of Apache Spark on Local Machine.

Laboratory Manual

2023-24

Sub:- **Mobile Application Development(MAD)**

B.Tech Final Year (CSE)

Prepared By

Prof. Karande A.A.

Prof. Sarawade J.P.

G.K. Gujar Memorial Charitable Trust's,

Dr.Ashok Gujar Technical Institute's

Dr. Daulatrao Aher College of Engineering, Karad.

Vidyanagar Ext. Banawadi, Tal. Karad 415124, Dist. Satara, Maharashtra INDIA

Prepared By:-	Approved By:
Prof. Karande A.A. Prof. Sarawade J.P.	Head of Department

Laboratory Manual

Mobile Application Development (Android)

For

BTECH

(Computer Science and Engineering)

Prepared By: Prof A. A. Karande
Prof. J. P. Sarawade

LABORATORY MANUAL CONTENTS

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Programme Outcomes (POs):

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities withan understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

LAB INDEX

DOs and DON'Ts in Laboratory:

1. Make entry in the Log Book as soon as you enter the Laboratory.
2. All the students should sit according to their roll numbers starting from their left to right.
3. All the students are supposed to enter the terminal number in the log book.
4. Do not change the terminal on which you are working.
5. All the students are expected to get at least the algorithm of the program/concept to be implement.
6. Strictly follow the instructions given by the teacher/Lab Instructor.

Sr. No	Title of Experiment
1	Installation of Android SDK, emulator
2	To create simple project and study of android project structure and installing apk on mobile device/tablet, configuring mobile device/tablet in Android Studio with developer option and running app directly on mobile device/table
3	Introduction to android and its architecture
4	Study and implementation of layouts of UI in Android
5	Study of Activity LifeCycle
6	Study and implementation of Intents in android
7	Write a program to use of Intents for SMS and Telephony.
8	Write a program to study and demonstrate Broadcast Receiver.
9	Program to demonstrate Buttons, Text Fields, Checkboxes, Radio Buttons, and Toggle Buttons with their events handler
10	Program to demonstrate Spinners, Touch Mode, Alerts, Popups, and Toasts with their events handler.
11	Program to demonstrate Touch Mode, Menus with their events handler.
12	Program to demonstrate notification with their action.
13	Develop a native calculator application
14	Implement an application that writes data to the SD card.
15	Write a mobile application that creates alarm clock.
16	Implement an application that implements Multi-threading
17	Write a program to study and use of SQLite database.
18	Study of publishing app to the Android Market.

Experiment no 1

Aim: Installation of Android SDK, emulator
Objective: Student should get the knowledge of installation of Android SDK, emulator.

Outcome: Student will be aware of the android operating system and install android studio

Theory:

Step 1 - System Requirements

The required tools to develop Android applications are open source and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.

Java JDK5 or later version

Java Runtime Environment (JRE)

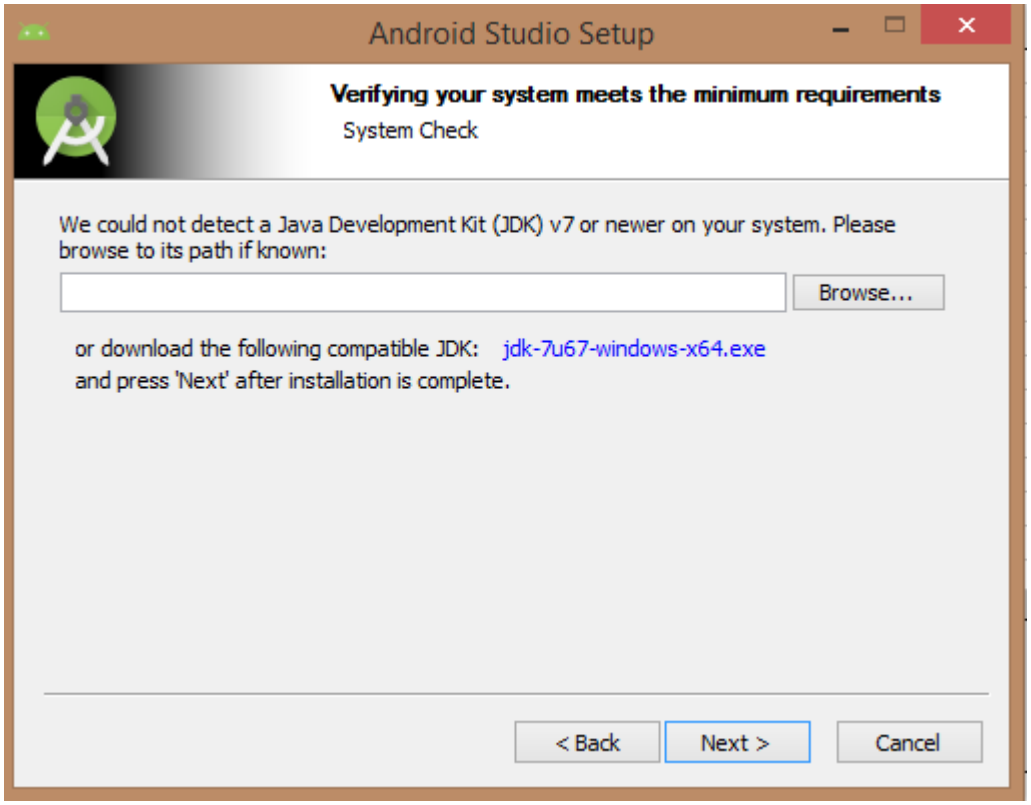
6Android Studio

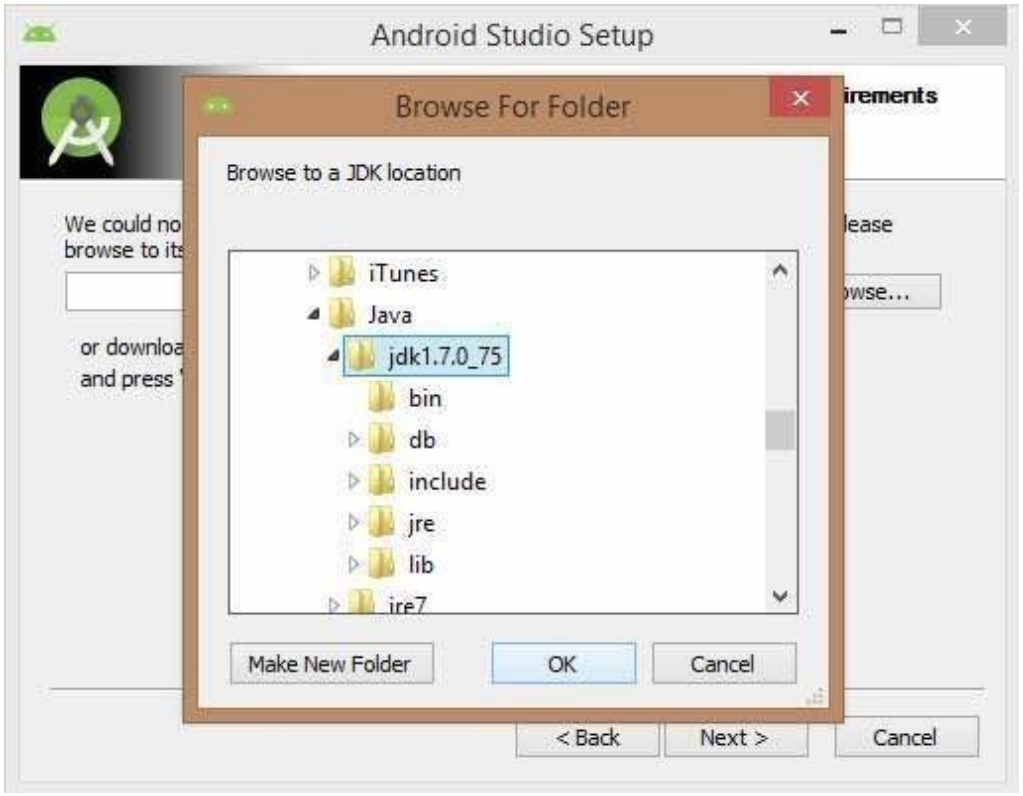
Step 2 - Setup Android Studio

Android Studio is the official IDE for android application development. It works based on IntelliJ IDEA, You can download the latest version of android studio from Android Studio 2.2 Download, If you are new to installing Android Studio on windows, you will find a file, which is named as android-studio-bundle-143.3101438-windows.exe.So just download and run on windows machine according to android studio wizard guideline.

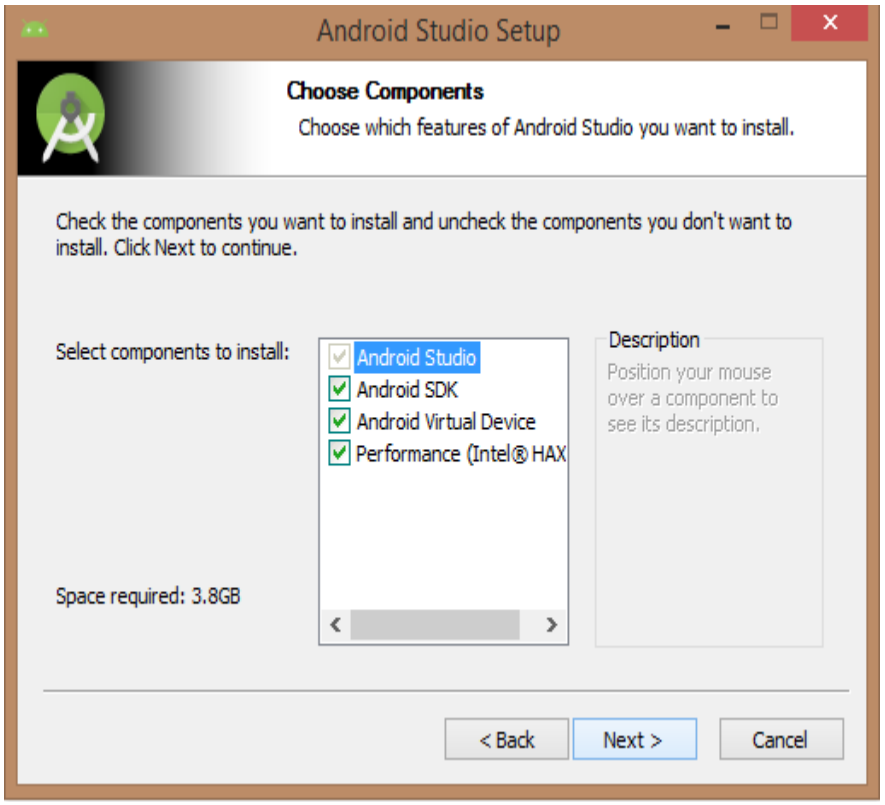
If you are installing Android Studio on Mac or Linux, You can download the latest version from Android Studio Mac Download,or Android Studio Linux Download, check the instructions provided along with the downloaded file for Mac OS and Linux. This tutorial will consider that you are going to setup your environment on Windows machine having Windows 8.1 operating system. Installation So let's launch Android Studio.exe, Make sure before launch Android Studio, Our Machine should required installed Java JDK. To install Java JDK, take a reference of Android environment setup

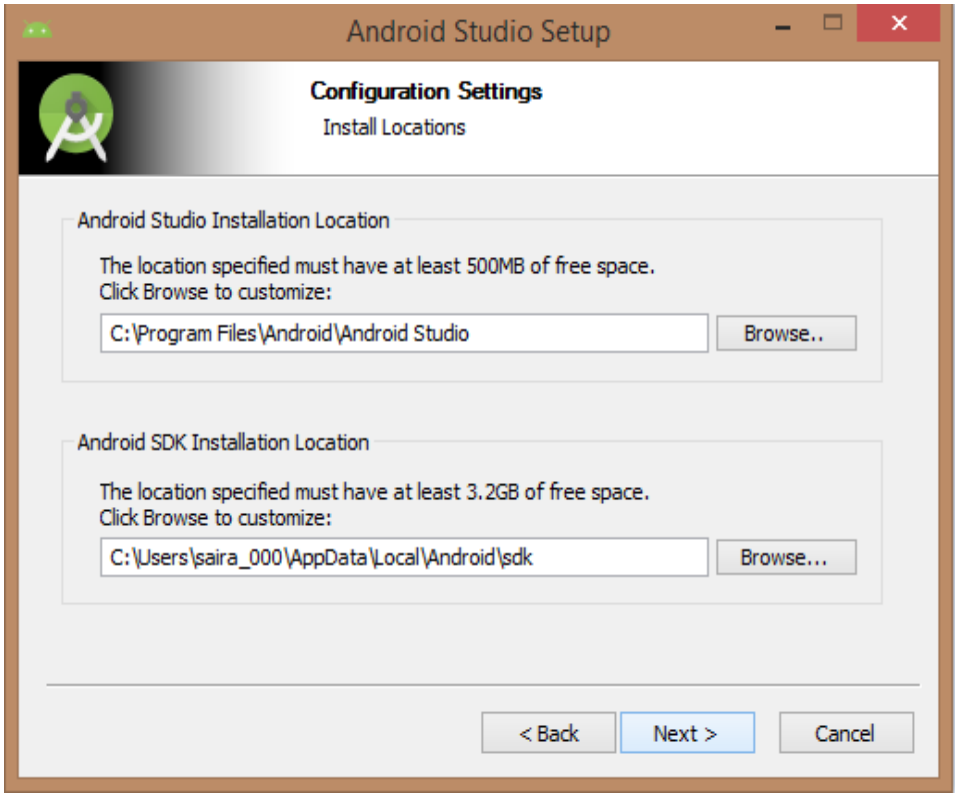
Once you launched Android Studio, its time to mention JDK7 path or later version in androidstudio installer.



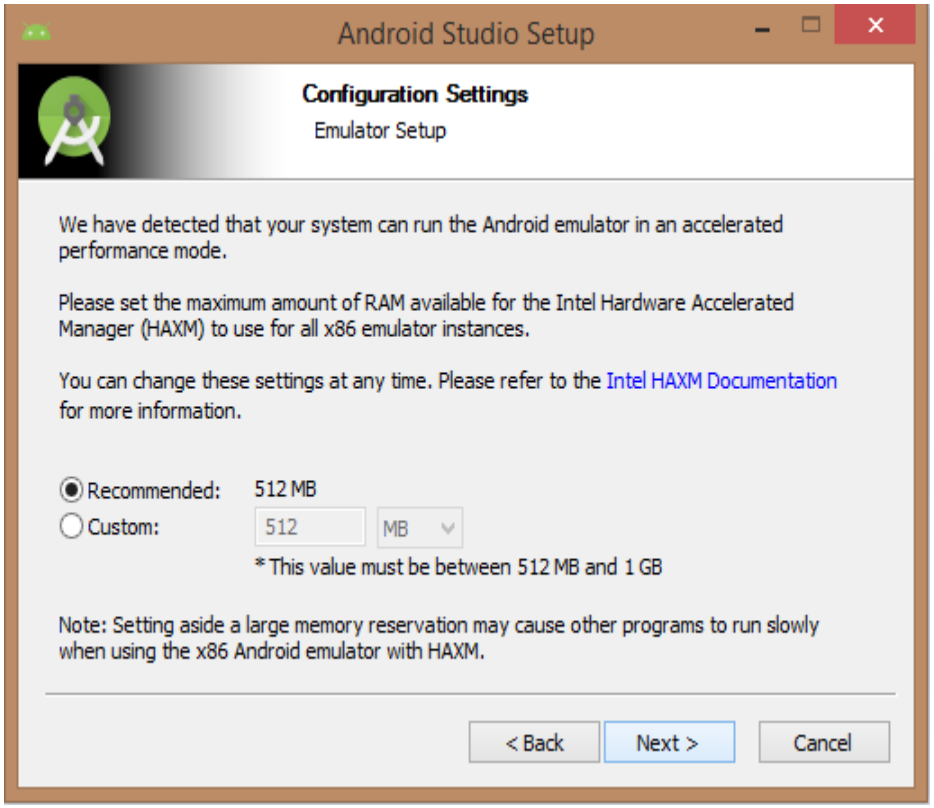


Need to check the components, which are required to create applications, below the image has selected Android Studio, Android SDK, Android Virtual Machine and performance(Intel chip).

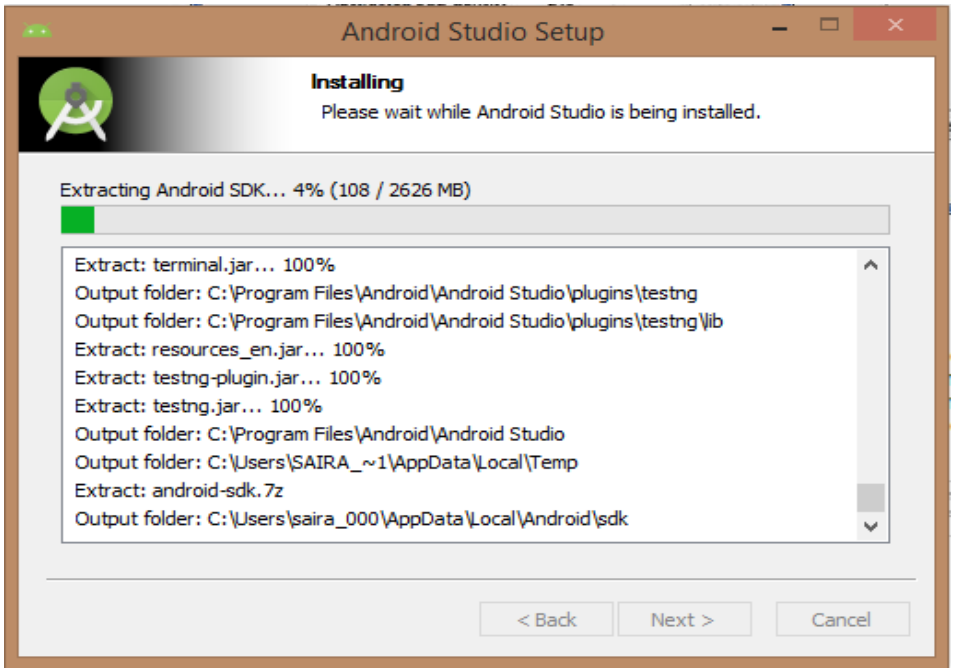




Need to specify the ram space for Android emulator by default it would take 512MB of local machine RAM.

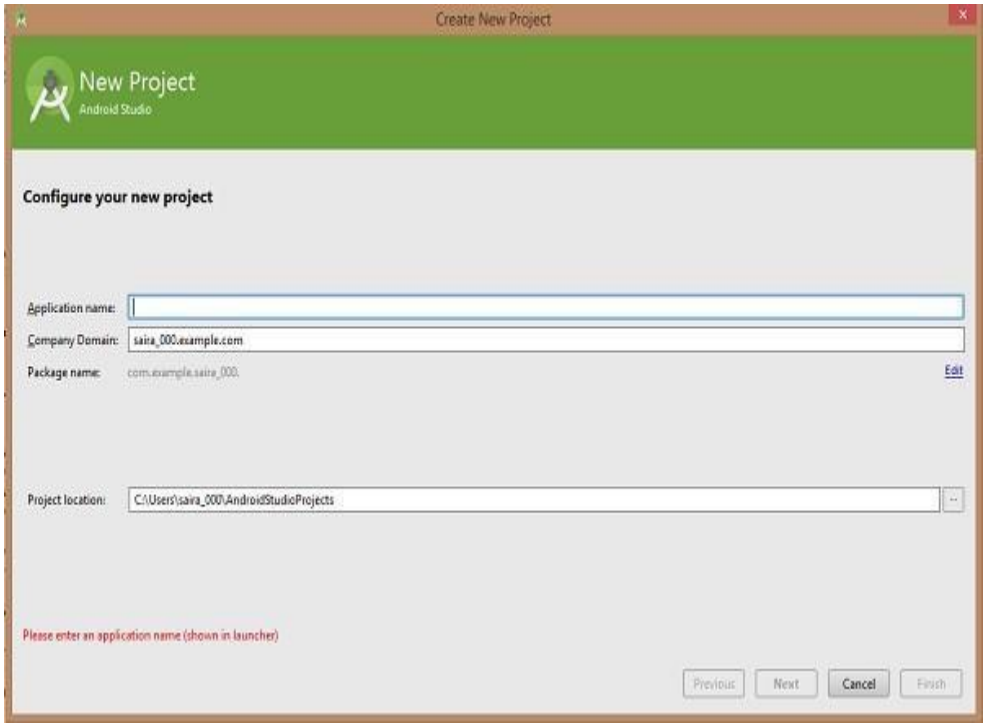


At final stage, it would extract SDK packages into our local machine, it would take a while time to finish the task and would take 2626MB of Hard disk space.

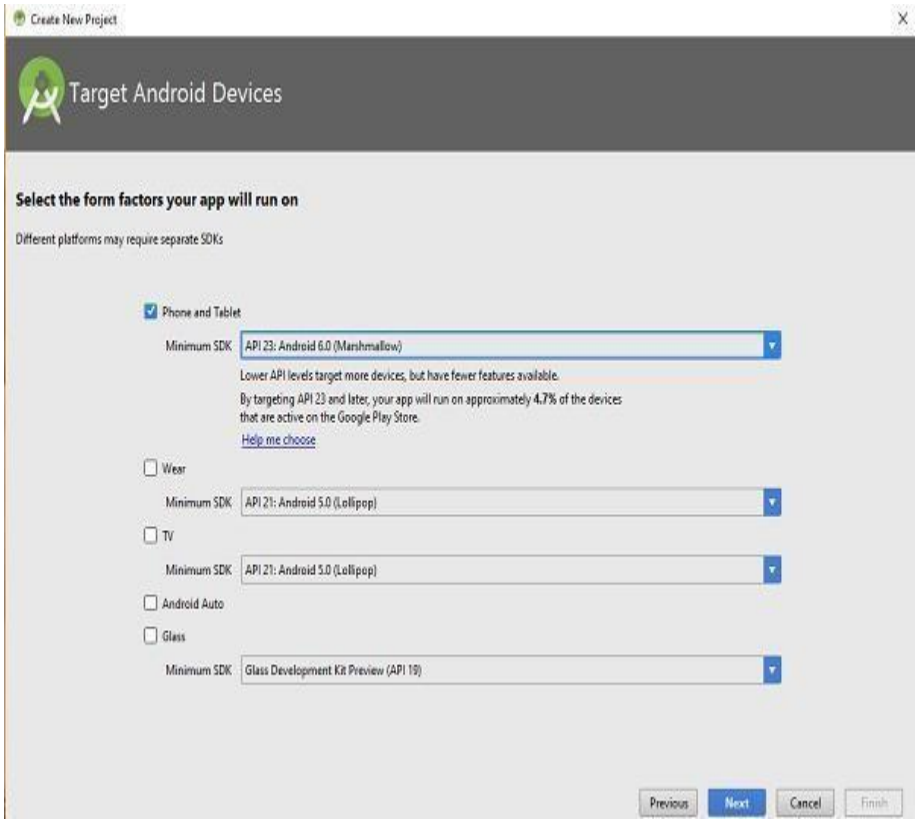


After done all above steps perfectly, you must get finish button and it gonna be open androidstudio project with Welcome to android studio message as shown below

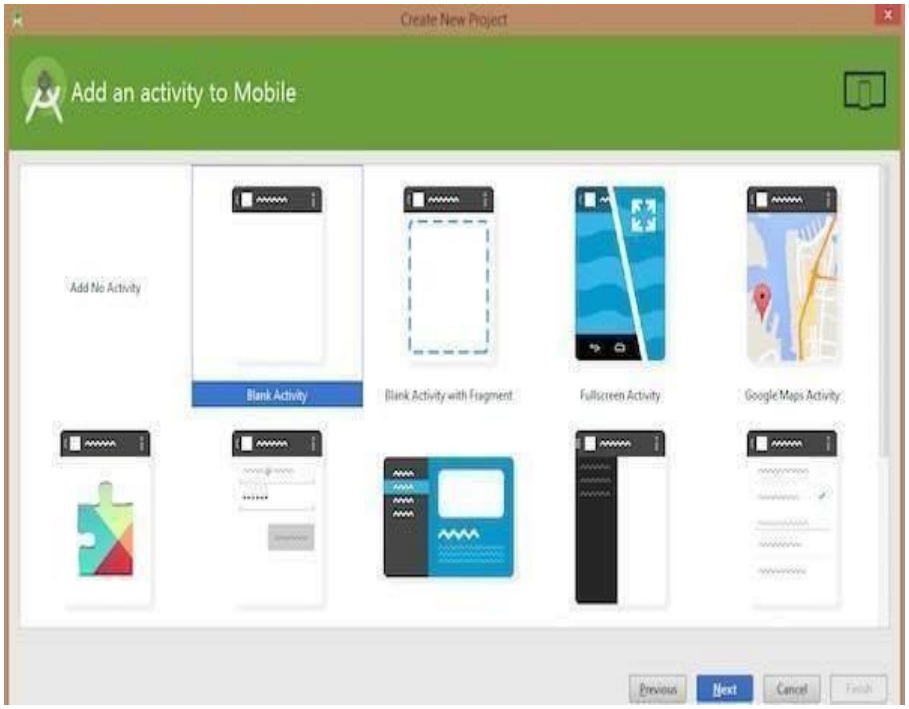
You can start your application development by calling start a new android studio project. in a new installation frame should ask Application name, package information and location of the project.



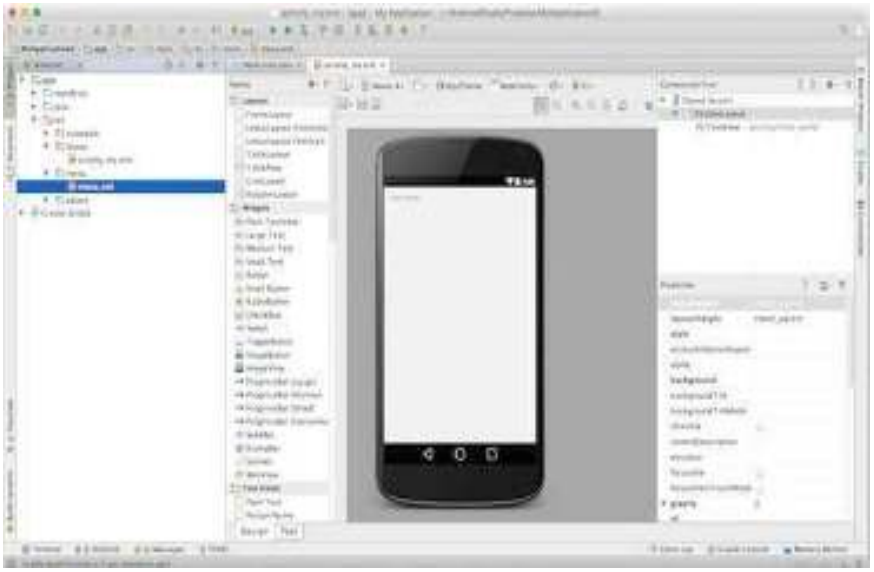
After entered application name, it going to be called select the form factors your application runson, here need to specify Minimum SDK, in our tutorial, I have declared as API23: Android 6.0(Mashmallow)



The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications



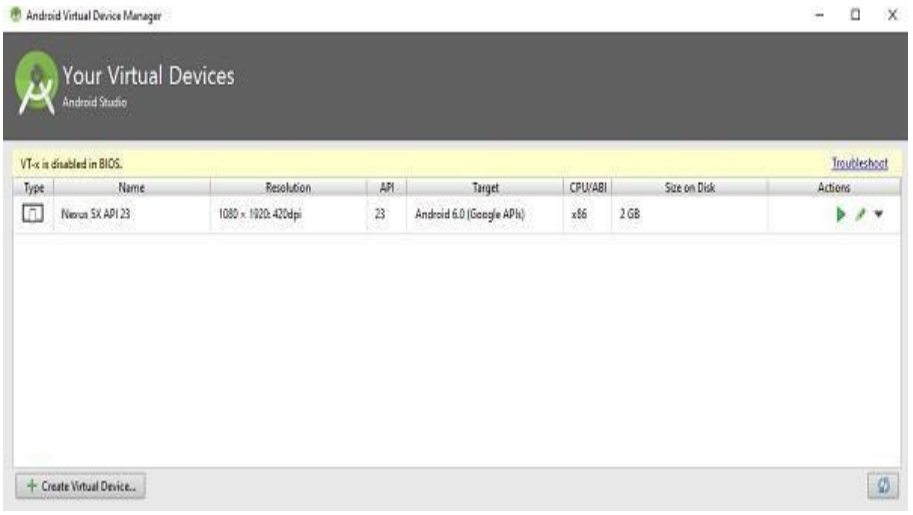
At the final stage it going to be open development tool to write the application code.



Step 3 - <u>Create Android Virtual Device</u>

To test your Android applications, you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device. Launch Android AVD Manager Clicking AVD_Manager icon as shown below

After Click on a virtual device icon, it going to be shown by default virtual devices which arepresent on your SDK, or else need to create a virtual device by clicking Create new Virtual device button



If your AVD is created successfully it means your environment is ready for Android application development. If you like, you can close this window using top-right cross button. Better you re- start your machine and once you are done with this last step, you are ready to proceed for your first Android example but before that we will see few more important concepts related to AndroidApplication Development.

Conclusion: Thus we have successfully installed android studio

Experiment no 2

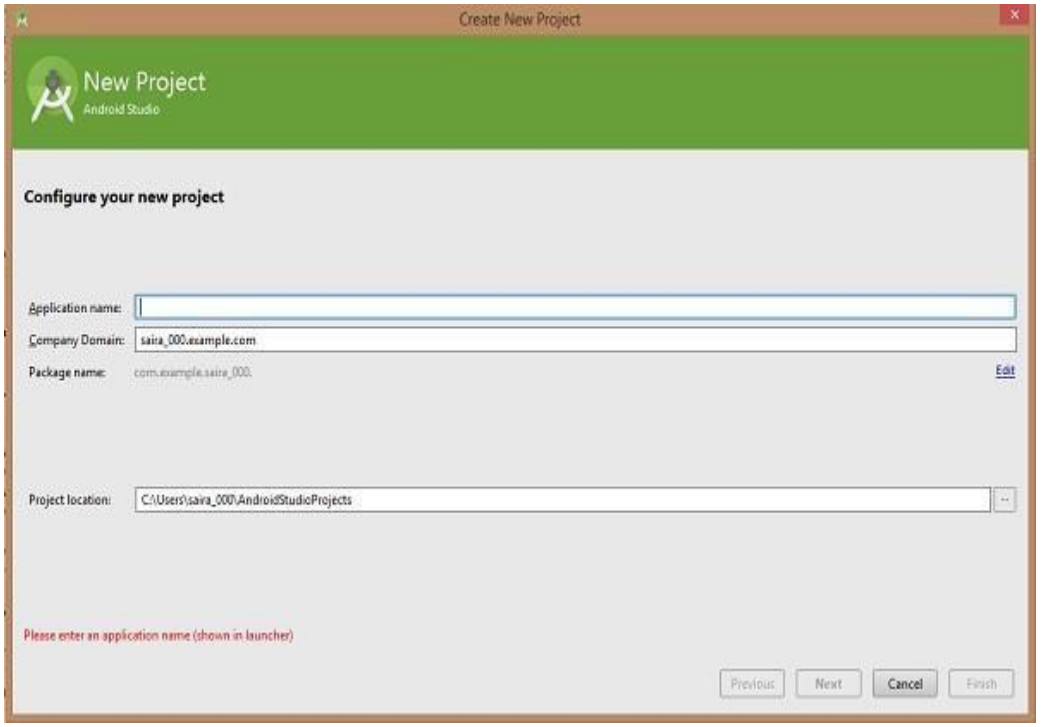
Aim: To create simple project and study of android project structure and installing apk on mobile device/tablet, configuring mobile device/tablet in Android Studio with developer option and running app directly on mobile device/table

Objective: Student should get the knowledge of creating and running android application.

Outcome: Student will be aware of the android operating system and develop their first apk

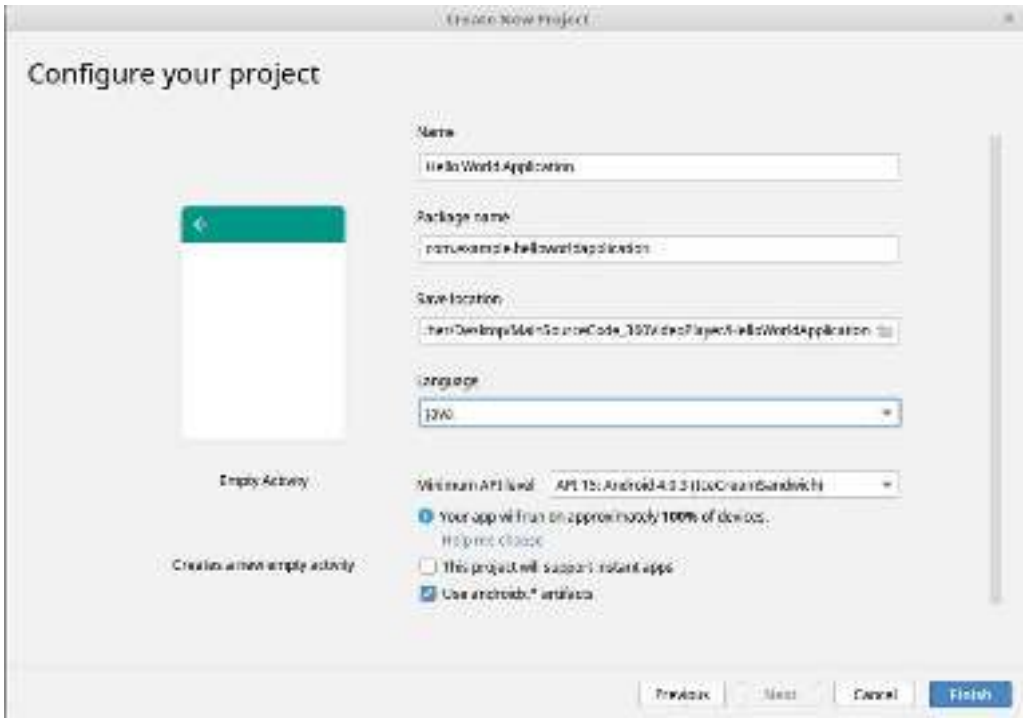
Theory:

You can start your application development by calling start a new android studio project. in a newinstallation frame should ask Application name, package information and location of the project.

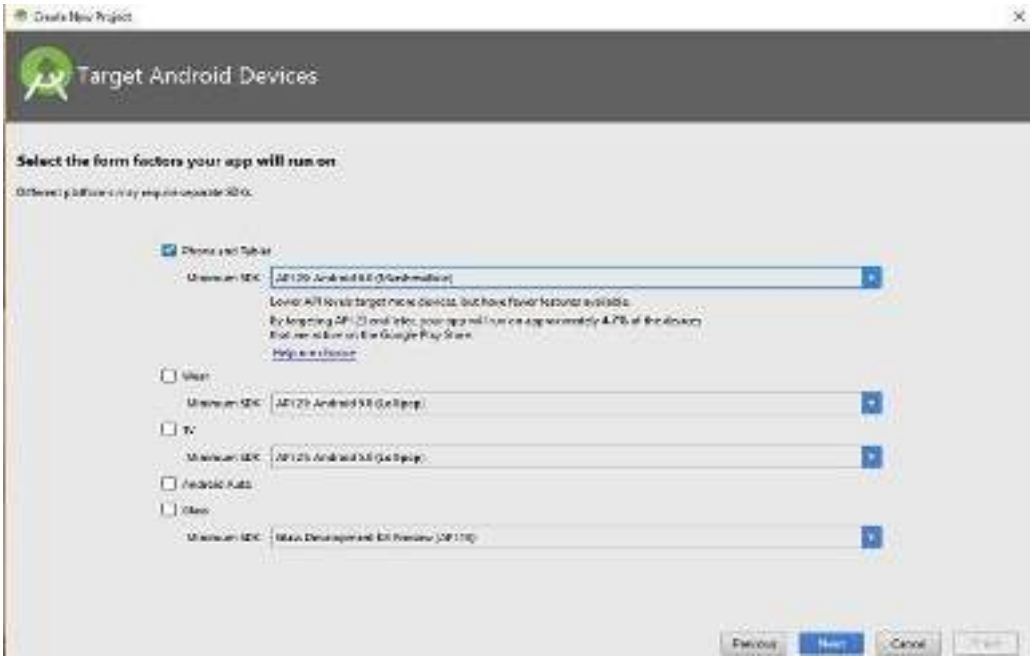


Configure the Hello World Project Details

We'll finish creating the project by configuring some details about its name, location, and the API version it



Change the name of the application. Change the default **Project location** to your preferreddirectory or just leave it as the default location.On the **minimum API level**, ensure that **API 15: Android 4.0.3 IceCreamSandwich** is set as theMinimum SDK. This ensures that your application runs on almost all devices.



The next level of installation should contain selecting the activity to mobile, it specifies the defaultlayout for Applications.

SOURCE CODE :

The Main Activity File

The main activity code is a Java file MainActivity.java. This is the actual application file whichultimately gets converted to a Dalvik executable and runs your application

```
package com.example.helloworldapplication;

importandroidx.appcompat.app.AppCompatActvity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

The Layout File

The **activity_main.xml** is a layout file available in res/layout directory, that is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following contentrelated to default layout −

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent" android:layout_height="match_parent"
  >

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:padding="@dimen/padding_medium"
    android:text="@string/hello_world"
    tools:context=".MainActivity" />
```

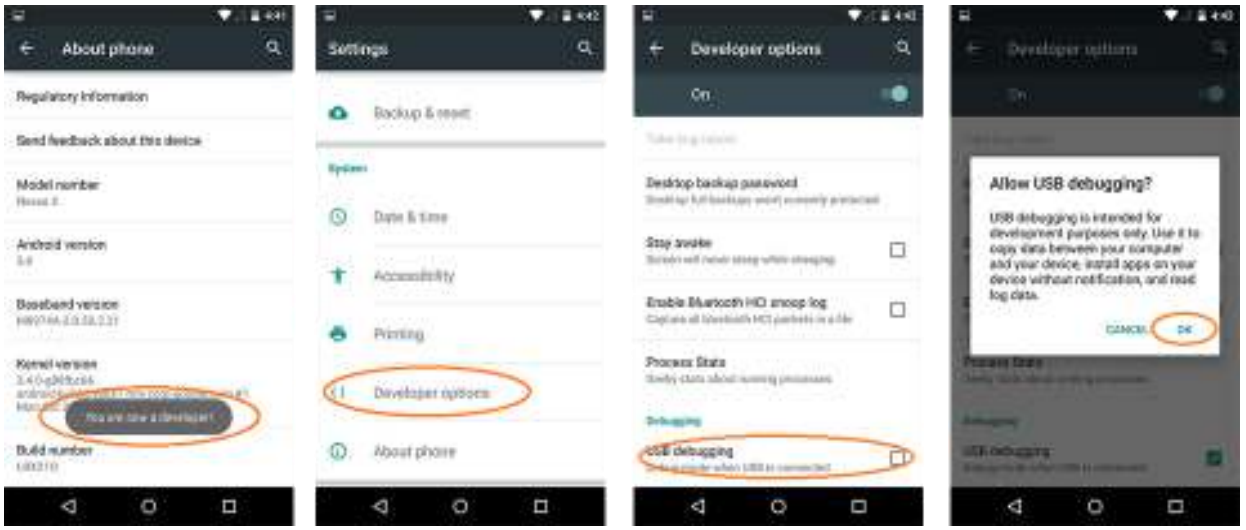
I)Running app on Phone:

Connect your Phone to Computer

Plug in your device to your computer with a USB cable. If you're developing on Windows, you might needto install this universal ADB USB driver or find your specific USB driver for your device.

Enable USB Debugging

The next step is to enable USB debugging so your phone can interact with your computer in a developer mode.



The following steps are needed:

1. (Windows Only) Install this ADB Driver
2. Plug-in your Android Device to Computer via USB
3. Open the "Settings" App on the Device
4. Scroll down to bottom to find "About phone" item
5. Scroll down to bottom to find "Build number" section
6. Tap on "Build Number" 7 times in quick succession
7. You should see the message "You are now a developer!"
8. Go back to main "Settings" page
9. Scroll down bottom to find "Developer options" item
10. Turn on "USB Debugging" switch and hit "OK"
11. Unplug and re-plug the device
12. Dialog appears "Allow USB Debugging?"
13. Check "Always allow from this computer" and then hit "OK"

Running your App

Now, we can launch apps from Android Studio onto our device:

1. Select one of your projects and click "Run" from the toolbar.
2. In the "Choose Device" window that appears, select the "Choose a running device" radio button, select the device, and click OK.

II) Running app on Emulator(AVD)

To run the app from Android studio, open one of your project's activity files and click Run icon from the tool bar. Android studio installs the app on your AVD and starts it and if everything is fine with your set-up and application, it will display following Emulator window −Once Gradle finishes building, Android Studio should installthe app on your connected device and start it.

Conclusion: Thus we have studied background and learnt procedure for running android application

Experiment no 3

Aim: Introduction to android

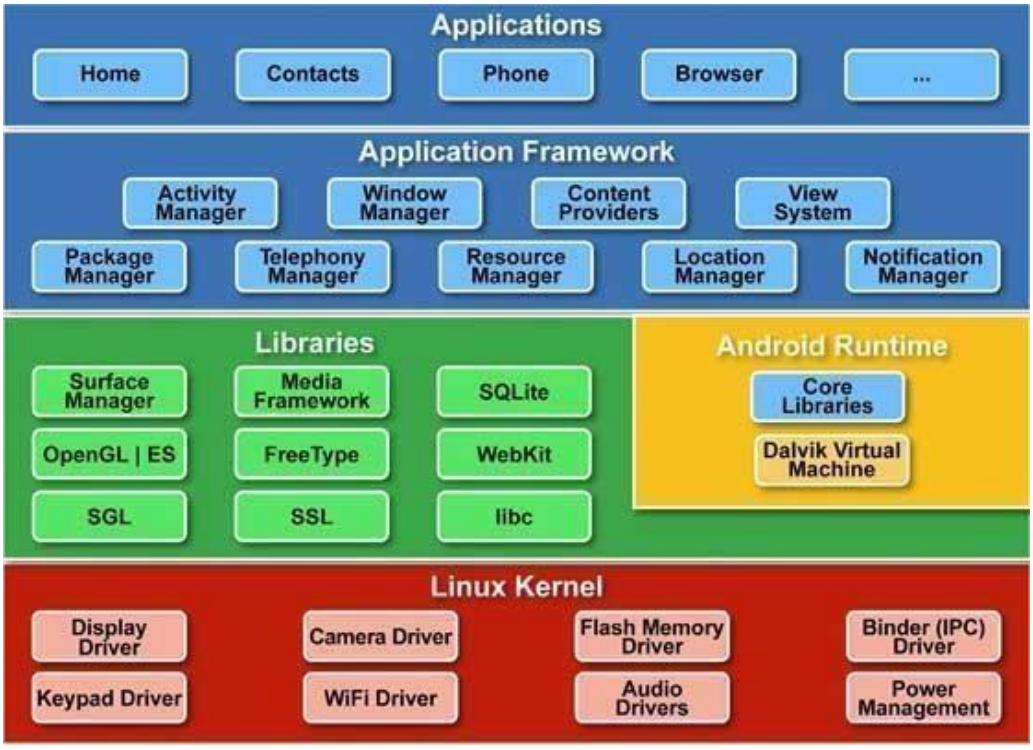
Objective: Student should get the knowledge of android operating system background.

Outcome: Student will be aware of the android operating system.

Theory:

Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecturediagram.



Linux kernel

At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

Libraries

On top of Linux kernel there is a set of libraries including open -source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play andrecord audio and video, SSL libraries responsible for Internet security etc.

Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VMenables every Android application to run in its own process, with its own instance ofthe Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

Android UI

An Android application user interface is everything that the user can see and interact with. You have learned about the various layouts that you can use to position your views in an activity. This chapter will give you detail on various views.

A **View** is an object that draws something on the screen that the user can interact with and a **ViewGroup**is an object that holds other View (and ViewGroup) objectsin order to define the layout of the user interface.

You define your layout in an XML file which offers a human-readable structure for the layout, similar to HTML. For example, a simple vertical layout with a textview and a button looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <TextView android:id="@+id/text"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"android:text="I
  am a TextView" />
  <Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="I am a Button" />
```

```
</LinearLayout>
```

Android UI Controls

There are number of UI controls provided by Android that allow you to build the graphical user interface for your app.

Sr No.	UI Control & Description
1	TextView This control is used to display text to the user.
2	EditText EditText is a predefined subclass of TextView that includes richediting capabilities.
3	AutoCompleteTextView The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.
4	Button A push-button that can be pressed, or clicked, by the user toperform an action.
5	ImageButton AbsoluteLayout enables you to specify theexact location of itschildren.
6	CheckBox An on/off switch that can be toggled by the user. You shoulduse checkboxes when presenting users with a group of selectable options that are not mutually exclusive.
7	ToggleButton An on/off button with a light indicator.

8	RadioButton The RadioButton has two states: either checked or unchecked.
9	RadioGroup A RadioGroup is used to group together one or more RadioButtons.
10	ProgressBar The ProgressBar view provides visual feedback about some ongoing tasks, such as when you are performing a task in thebackground.
11	Spinner A drop-down list that allows users to select one value from a set.
12	TimePicker The TimePicker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode.
13	DatePicker The DatePicker view enables users to select a date of the day.

Create UI Controls

As explained in previous chapter, a view object may have a unique ID assigned to it which will identify the View uniquely within the tree. The syntax for an ID, inside an XML tag is:

```
android:id="@+id/text_id"
```

To create a UI Control/View/Widget you will have to define a view/widget in thelayout file and assign it a unique ID as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <TextView android:id="@+id/text_id"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="I am a TextView" />
</LinearLayout>
```

Then finally create an instance of the Control object and capture it from thelayout, use the following:

```
TextView myText = (TextView) findViewById(R.id.text_id);
```

Android Event Handling

Events are a useful way to collect data about a user's interaction with interactive components of your app, like button presses or screen touch etc. The Android framework maintains an event queue into which events are placed as they occur and then each event is removed from the queue on a first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate actionas per requirements.

There are following three concepts related to Android Event Management:

- **Event Listeners:**

  The **View** class is mainly involved in building up a Android GUI,same View class provides a number of Event Listeners. The Event Listener is theobject that receives notification when an event happes.

- **Event Listeners Registration:** Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is calledwhen the Event Listener fires the event.
- **Event Handlers:** When an event happens and we have registered and event listener fo the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Event Listeners & Event Handlers

Event Handler	Event Listener & Description
onClick()	**OnClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button,text, image etc. You will use onClick() event handler to handle such event.
onLongClick()	**OnLongClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button,text, image etc. for one or more seconds. You willuse onLongClick() event handler to handle such event.
onFocusChange()	**OnFocusChangeListener()** This is called when the widget looses its focus ie.user goes away from the view item. You will use onFocusChange() event handler to handle such event.
onKey()	**OnFocusChangeListener()** This is called when the user is focused on the item and presses or releases a hardware keyon

	the device. You will use onKey() event handler to handle such event.
onTouch()	**OnTouchListener()** This is called when the user presses the key, releases the key, or any movement gesture on thescreen. You will use onTouch() event handler to handle such event.
onMenuItemClick()	**OnMenuItemClickListener()** This is called when the user selects a menu item.You will use onMenuItemClick() event handler tohandle such event.

There are many more event listeners available as a part of **View** class like OnHoverListener, OnDragListener etc which may be needed for your application.So I recommend to refer official documentation for Android application development in case you are going to develop a sophisticated apps.

Conclusion: Thus we have studied background and architecture of android

:

Experiment no 4

Aim: Study and implementation of layouts of UI in Android

Objective: Student should be able to design their own UI for android application using XML.

Outcome: Student will demonstrate the basic application using UI in android.

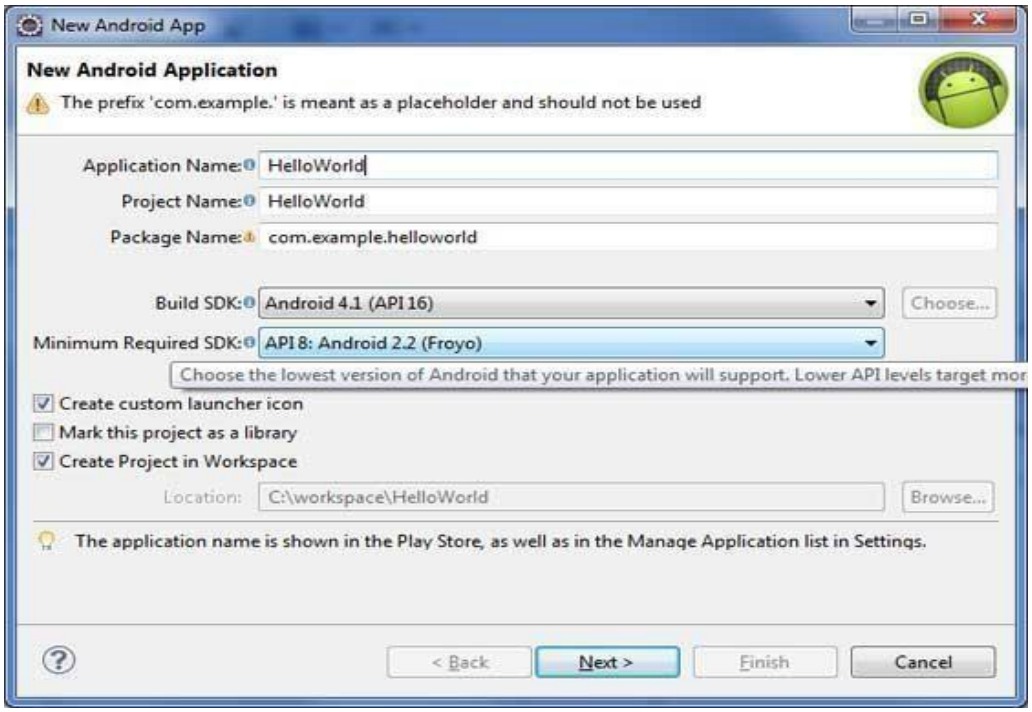
Theory

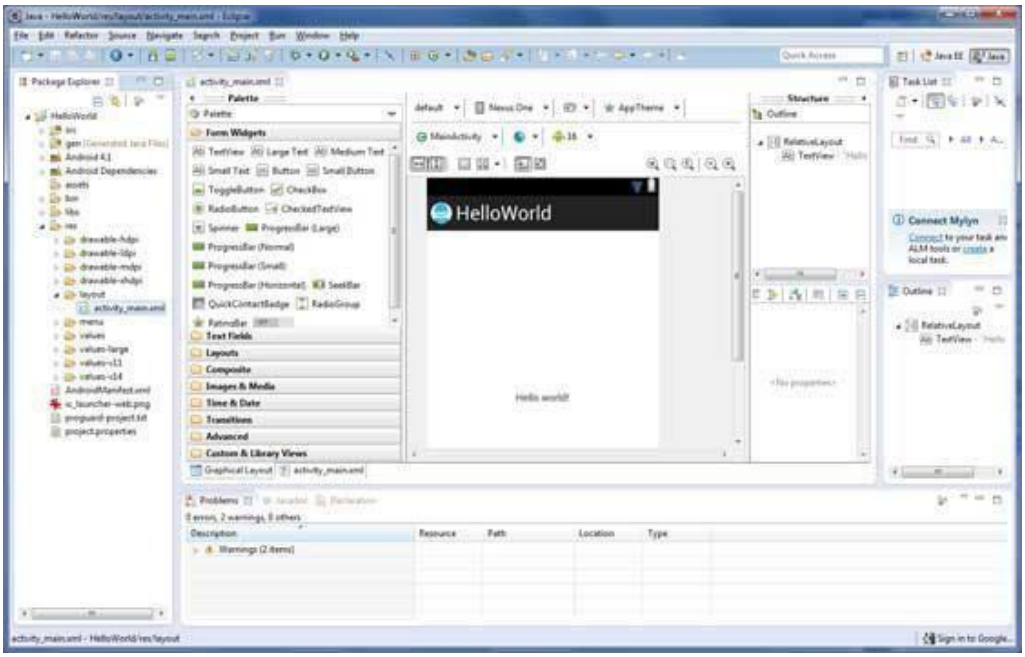
Create Android Application

The first step is to create a simple Android Application using Eclipse IDE. Follow the option

File New Project and finally select Android New Application wizard fromthe wizard list. Now name your application as HelloWorld using the wizard window as follows:

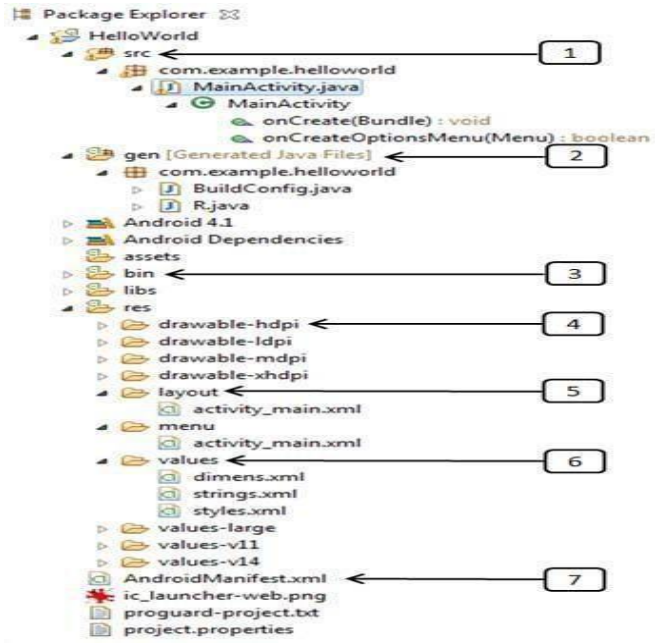
Next, follow the instructions provided and keep all other entries as default till the final step. Once your project is created successfully, you will have following project screen:





Anatomy of Android Application

Before you run your app, you should be aware of a few directories and files inthe Android project:



Sr. No.	Folder, File & Description
1	src This contains the .java source files for your project. By default, it
	includes an MainActivity.java source file having an activity classthat runs when your app is launched using the app icon.
2	gen This contains the .R file, a compiler-generated file that referencesall the resources found in your project. You should not modify this file.
3	bin This folder contains the Android package files .apk built by the ADT during the build process and everything else needed to run an Android application.
4	res/drawable-hdpi This is a directory for drawable objects that are designed for high-density screens.
5	res/layout This is a directory for files that define your app's user interface.
6	res/values This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.
7	AndroidManifest.xml This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

Following section will give a brief overview few of the important application files.The

Main Activity File

The main activity code is a Java file MainActivity.java. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the default code generated by the applicationwizard for Hello World! application:

```
package com.example.helloworld;
import android.os.Bundle; import
android.app.Activity; import
android.view.Menu; import
android.view.MenuItem;
import android.support.v4.app.NavUtils; public
class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu); return
        true;
    }
}
```

Here, R.layout.activity_main refers to the activity_main.xml file located in the res/layout folder. TheonCreate() method is one of many methods that are fi red when an activity is loaded.

The Manifest File

Whatever component you develop as a part of your application, you must declareall its components in a manifest file called AndroidManifest.xml which ressides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file:

```
<manifest                xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.helloworld"
  android:versionCode="1"
  android:versionName="1.0" >
```

```
   <uses-sdk
      android:minSdkVersion="8"
          android:targetSdkVersion="15"  />
   <application android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:theme="@style/AppTheme" >
       <activity
           android:name=".MainActivity"
           android:label="@string/title_activity_main" >
           <intent-filter>
               <action  android:name="android.intent.action.MAIN"  />
               <category  android:name="android.intent.category.LAUNCHER"/>
           </intent-filter>
       </activity>
   </application>
</manifest>
```

Here <application>...</application> tags enclosed the components related to the application. Attributeandroid:icon will point to the application icon available under res/drawable-hdpi. The application uses the image named ic_launcher.png located in the drawable folders

The <activity> tag is used to specify an activity and android:name attribute specifies the fully qualified class name of the Activity subclass and the android:label attributes specifies a string to use as the label for the activity. You can specify multiple activities using <activity> tags.

The action for the intent filter is named android.intent.action.MAIN to indicate that this activity serves as the entry point for the application. The category for the intent-filter is namedandroid.intent.category.LAUNCHER to indicate that the application can be launched from the device's launcher icon.

The @string refers to the strings.xml file explained below. Hence, @string/app_name refers to theapp_name string defined in the strings.xml fi le, which is "HelloWorld". Similar way, other strings getpopulated in the application.

Following is the list of tags which you will use in your manifest file to specify different Android application components:

<activity>elements for activities

<service> elements for services

<receiver> elements for broadcast receivers

<provider> elements for content providersThe

Strings File

The strings.xml file is located in the res/values folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file:

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string  name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources> The
```

R File

The gen/com.example.helloworld/R.java file is the glue between the activityJava files likeMainActivity.java and the resources like strings.xml. It is an automatically generated file and you should not modify the content of the R.java file. Following is a sample of R.java file:

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found. It
 * should not be modified by hand.
 */

package com.example.helloworld;

public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int padding_large=0x7f040002; public
        static final int padding_medium=0x7f040001;public static
        final int padding_small=0x7f040000;
    }
```

```
    public static final class drawable {
        public static final int ic_action_search=0x7f020000;public
        static final int ic_launcher=0x7f020001;
    }
    public static final class id {
        public static final int menu_settings=0x7f080000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int activity_main=0x7f070000;
    }
    public static final class string {
        public static final int app_name=0x7f050000; public
        static final int hello_world=0x7f050001; public static
        final int menu_settings=0x7f050002;
        public static final int title_activity_main=0x7f050003;
    }
    public static final class style {
        public static final int AppTheme=0x7f060000;
    }
}
```

The Layout File

The activity_main.xml is a layout file available in res/layout directory, that is referenced by your application when building its interface. You will modify thisfile very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"android:layout_width="match_parent"
  android:layout_height="match_parent" >

  <TextView android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:padding="@dimen/padding_medium"
      android:text="@string/hello_world"
      tools:context=".MainActivity" /
</RelativeLayout>
```

This is an example of simple RelativeLayout which we will study in a separate chapter. The TextView is an Android control used to build the GUI and it have various attribuites like android:layout_width,android:layout_height etc which are being used to set its width and height etc. The @string refers to the strings.xmlfile located in the res/values folder. Hence, @string/hello_world refers to the hello string defined in the strings.xml fi le, which is "Hello World!".

Running the Application

Conclusion: Thus we a have studied and implemented UI application using relative layout

Experiment no 5

Aim: Study of Activity LifeCycle

Objective: Student should get the knowledge of android activity lifecycle.

Outcome: Student will demonstrate the Activity lifecycle through the application.

Theory:

Activity life cycle State of an activity

An activity can be in different states depending how it is interacting with theuser. These states are described by the following table.

Activity state

State	Description
Running	Activity is visible and interacts with the user.
Paused	Activity is still visible but partially obscured, instance is runningbut might be killed by the system.
Stopped	Activity is not visible, instance is running but might be killed bythe system.
Killed	Activity has been terminated by the system of by a call toits finish() method.

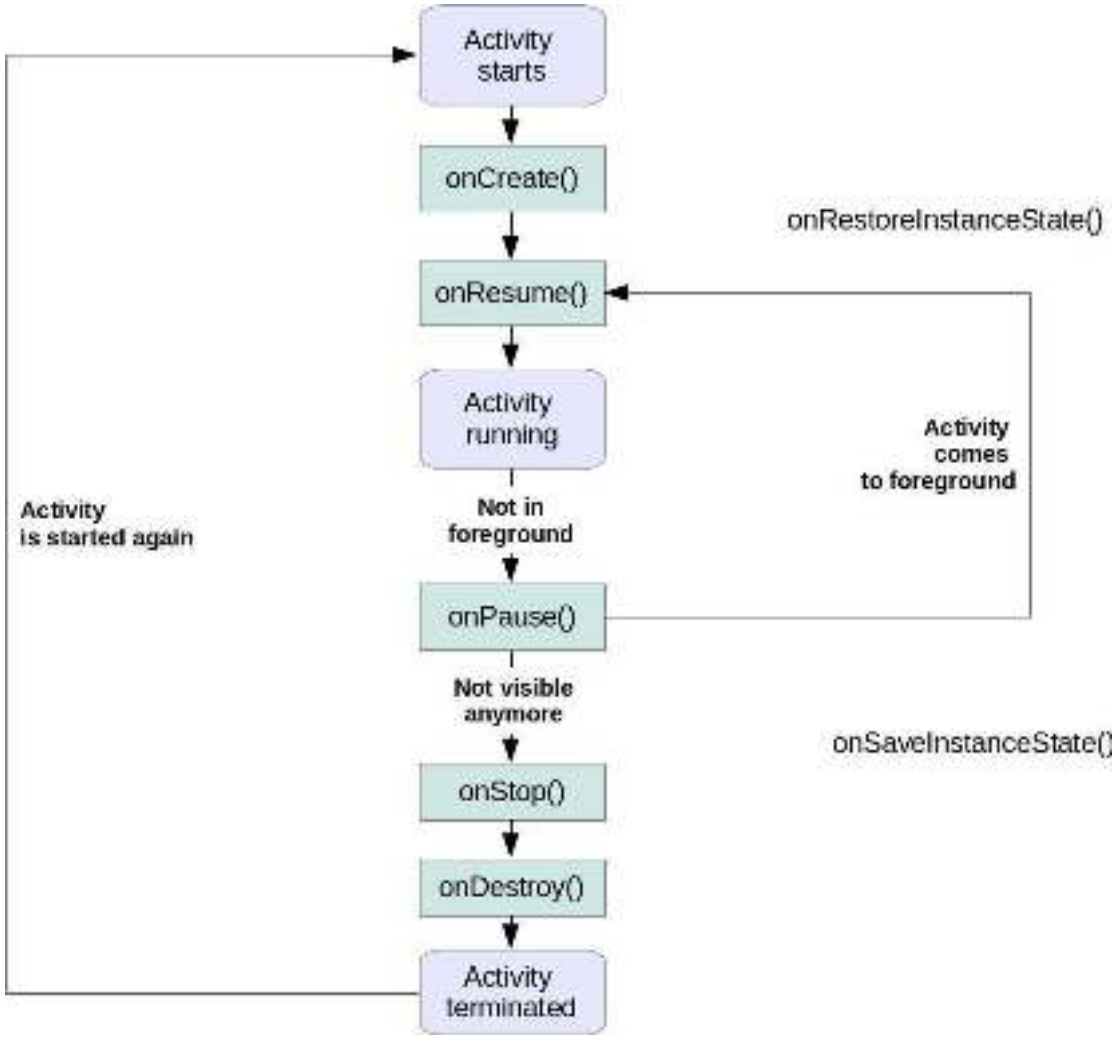
The live cycle methods

The Android system defines a life-cycle for activities via predefined life-cycle methods. The most important methods are:

Important Activity lifecycle methods

Method	Purpose
onCreate()	Called then the activity is created. Used to initialize the activity, for example create the user interface.
onResume()	Called if the activity get visible again and the user starts interacting with the activity again. Used to initialize fields, register listeners, bind to services, etc.
onPause()	Called once another activity gets into the foreground. Always called before the activity is not visible anymore. Used to release resources or save application data. For example you unregister listeners, intent receivers, unbind from services or remove system service listeners.
onStop()	Called once the activity is no longer visible. Time or CPU intensive shut-down operations, such as writing information to a database should be down in the onStop() method. This method is guaranteed to be called as of API 11.

The life cycle of an activity with its most important methods is displayed in the following diagram.



Android has more life cycle methods but not all of these methods are guaranteedto be called. The onDestroy() method is not guaranteed to be called, hence you typically do not use it.

Activity instance state

Instance state of an activity which is required to restore the activity to the statein which the user left it. This is non-persistent application data that needs tobe passed between activities restarts during a configuration change to restore user selections. The application is responsible for restoring its instance state.

Assume for example the user scrolled through a ListView with thousands of items and the activity is recreated. Loosing the position in the list is annoying for the user, hence the position should be restored.The onSaveInstanceState() can be used to store this instance state as a Bundle. A Bundle can contain primitive data types, arrays, String and objects which areof the Parcelable or Serialisable type.

The persisted Bundle data is passed at restart of the activity to the onCreate() method and onRestoreInstanceState() as parameter.If you override onSaveInstanceState() and onRestoreInstanceState() you should call the super implementation of it, because the default views of Android store their data via a call to View.onSaveInstanceState from the onSaveInstanceState() method of the activity. For example EditText stores its content via the default call of this method.The onRestoreInstanceState() or the onCreate() methods can be used to recreate the instance scope of an activity if it is restarted.

Conclusion: Thus we have studied activity life cycle

Experiment no 6

Aim: Study and implementation of Intents in android

Objective: Student should know what is intent in android and what are the types of it.

Outcome: Student will develop a good application using the intents and its properties efficiently

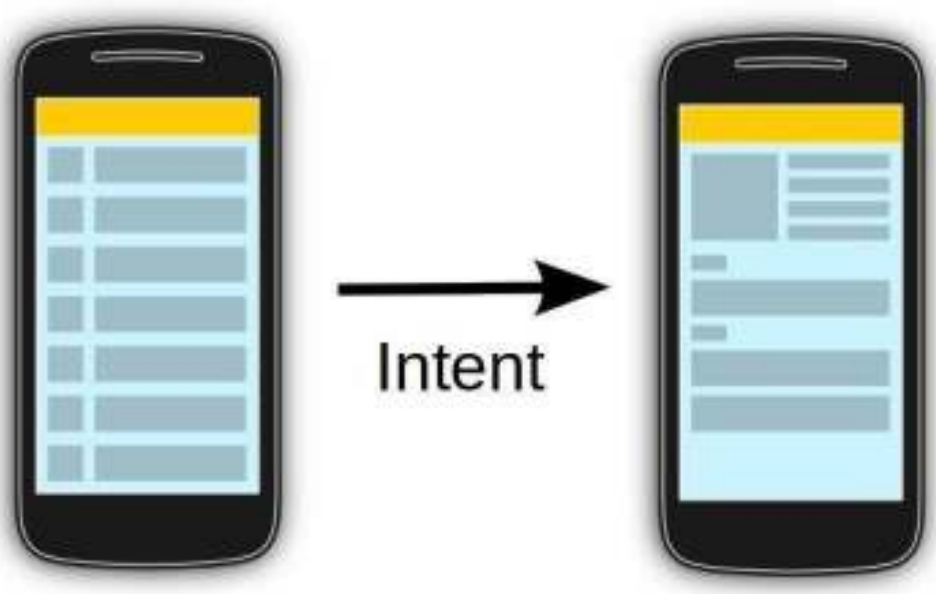
Theory:

Intents and intent filter

What are intents?

Intents are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

Intents are objects of the android.content.Intent type. Your code can send them to the Android system defining the components you are targeting. For example, via the startActivity() method you can define that the intent should be used to start an activity. An intent can contain data via a Bundle. This data can be used by the receiving component. Starting activities. To start an activity, use the method startActivity(intent). This method is defined on the Context object whichactivity extends.



The following code demonstrates how you can start another activity via anintent.

```
# Start the activity connect to the
# specified class

Intent i = new Intent(this, ActivityTwo.class);
startActivity(i);
```

Sub-activities

Activities which are started by other Android activities are called sub-activities.This wording makes it easier to describe which activity is meant.

Starting services

You can also start services via intents. Use the startService(Intent) method callfor that.

Different types of intents

Android supports explicit and implicit intents. An application can define the target component directly in the intent (explicit intent) or ask the Android system to evaluate registered components based on the intent data (implicit intents).

Explicit Intents

Explicit intents explicitly define the component which should be called by the Android system, by using the Java class as identifier.

The following shows how to create an explicit intent and send it to the Android system. If the class specified in the intent represents an activity, the Android system starts it.

```
Intent i = new Intent(this, ActivityTwo.class);
i.putExtra("Value1", "This value one for ActivityTwo ");
i.putExtra("Value2", "This value two ActivityTwo");
```

Explicit intents are typically used within on application as the classes in an application are controlled by the application developer.

Implicit Intents

Implicit intents specify the action which should be performed and optionally data which provides content for the action.

For example, the following tells the Android system to view a webpage. All installed web browsers should be registered to the corresponding intent data via an intent filter.

```
Intent i = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.exercise.com"));
```

```
startActivity(i);
```

If an implicit intent is sent to the Android system, it searches for all componentswhich are registered for the specific action and the fitting data type.

If only one component is found, Android starts this component directly. If several components are identified by the Android system, the user will get a selection dialog and can decide which component should be used for the intent.

Data transfer between activities

Data transfer to the target component: An intent contains certain header data, e.g., the desired action, the type, etc. Optionally an intent can also contain additional data based on an instance of the Bundle class which can be retrieved from the intent via the getExtras() method.

You can also add data directly to the Bundle via the overloaded putExtra()methods of the Intent objects. Extras are key/value pairs. The key is always of type String. As value you can use the primitive data types (int, float, ...) plus objects of type String, Bundle, Parceable and Serializable.

The receiving component can access this information via the getAction() and getData() methods on the Intent object. ThisIntent object can be retrieved via thegetIntent() method.

The component which receives the intent can use the getIntent().getExtras() method call to get the extra data. That is demonstrated in the following code snippet.

```
Bundle extras = getIntent().getExtras();
if (extras == null) {
  return;
}
```

```
// get data via the key
String value1 = extras.getString(Intent.EXTRA_TEXT);
if (value1 != null) {
  // do something with the data
}
```

Example: Using the share intent

Lots of Android applications allow you to share some data with other people, e.g., the Facebook, G+, Gmail and Twitter application. You can send data to one of these components. The following code snippet demonstrates the usage of such an intent within your application.

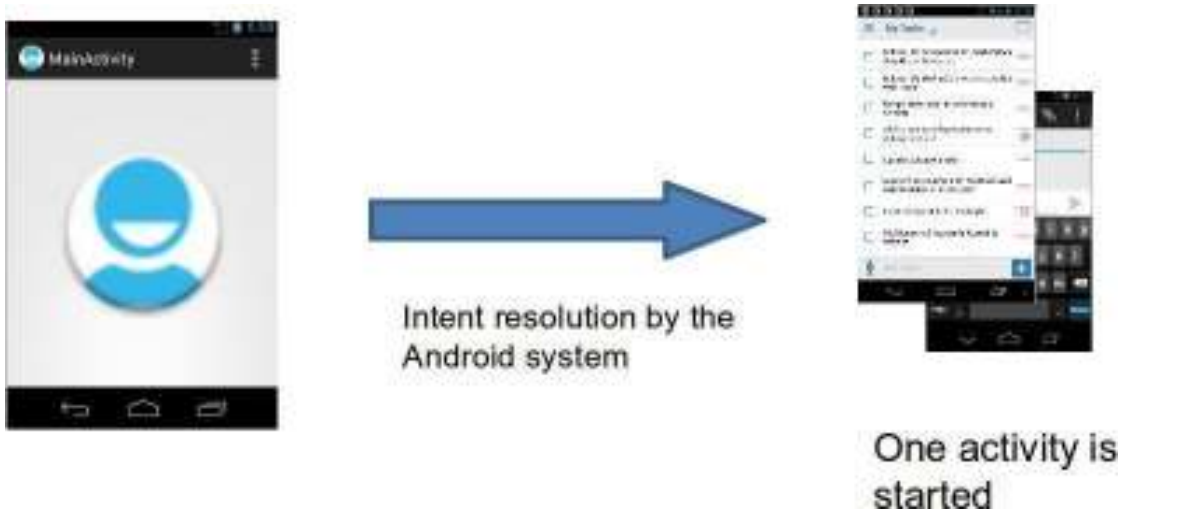
```
// this runs, for example, af ter a button click
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(android.content.Intent.EXTRA_TEXT, "News for you!");
startActivity(intent);
```

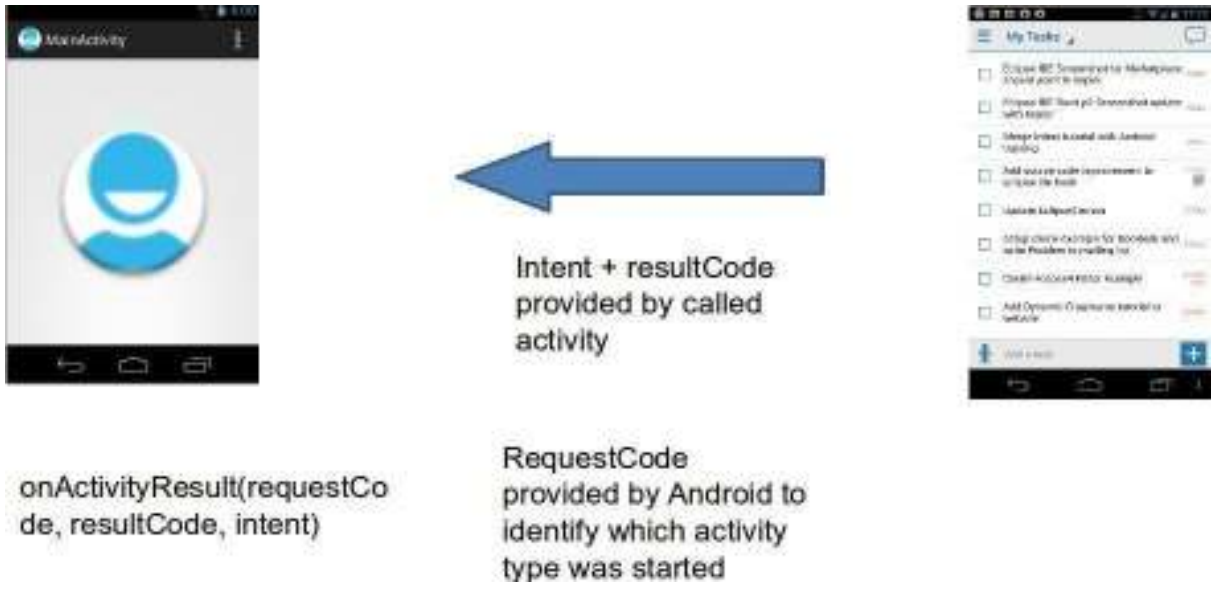
Retrieving result data from a sub-activity

An activity can be closed via the back button on the phone. In this case the finish() method is performed. If the activity was started with the startActivity(Intent) method call, the caller requires no result or feedback from the activity which now isclosed.

If you start the activity with the startActivityForResult() method call, you expect feedback from the sub-activity. Once the sub-activity ends, the onActivityResult() method on the sub-activity is called and you can perform actions based on the result.

In the startActivityForResult() method call you can specify a result code to determine which activity you started. This result code is returned to you. The started activity can also set a result code which the caller can use to determine if the activity was canceled or not.





The sub-activity uses the finish() method to create a new intent and to put data into it. It also sets a result via the setResult()method call.

The following example code demonstrates how to trigger an intent with the startActivityForResult() method.

```
public void onClick(View view) {
  Intent i = new Intent(this, ActivityTwo.class);
  i.putExtra("Value1", "This value one for ActivityTwo ");
  i.putExtra("Value2", "This value two ActivityTwo");
  // set the request code to any code you like,
  // you can identif y the callback via this code
  startActivityForResult(i, REQUEST_CODE);
}
```

If you use the startActivityForResult() method, then the started activity is calleda sub-activity.

If the sub-activity is finished, it can send data back to its caller via an Intent.This is done in the finish() method.

```
@Override
public void finish() {
  // Prepare data intent
  Intent data = new Intent();
  data.putExtra("returnKey1", "Swinging on a star. ");
  data.putExtra("returnKey2", "You could be better then you are. ");
  // Activity f inished ok, return the data
  setResult(RESULT_OK, data);
  super.finish();
}
```

Once the sub-activity finishes, the onActivityResult() method in the callingactivity is called.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
  if (resultCode == RESULT_OK && requestCode == REQUEST_CODE) {
    if (data.hasExtra("returnKey1"))  {
      Toast.makeText(this, data.getExtras().getString("returnKey1"),
        Toast.LENGTH_SHORT).show();
    }
  }
}
```

Defining intent filters

Intent filter

Intents are used to signal to the Android system that a certain event hasoccurred. Intents often describe the action which should be performed and provide data upon which such an action should be done. For example, your application can start a browser component for a certain URL via an intent. Thisis demonstrated by the following example.

```
String url = "http://www.exercise.com";
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(url));
startActivity(i);
```

A component can register itself via an intent filter for a specific action and specific data. An intent filter specifies the types of intents to which an activity, service, or broadcast receiver can respond to by declaring the capabilities of a component
Android components register intent filters either statically in the AndroidManifest.xml or in case of a broadcast receiver also dynamically via code. An intent filter is defined by its category, action and data filters. It can also contain additional meta-data.

If an intent is sent to the Android system, the Android platform runs a receiver determination. It uses the data included in the intent. If several components have registered for the same intent filter, the user can decide which componentshould be started.

Defining intent filter

You can register your Android components via intent filters for certain events. If a component does not define one, it can only be called by explicit intents. This chapter gives an example for registering a component for an intent. The key for this registration is that your component registers for the correct action, mime-type and specifies the correct meta-data.If you send such an intent to your system, the Android system determines all registered Android components for this intent. If several components have registered for this intent, the user can select which one should be used.

Conclusion: Thus we have studied and implemented concept of Intent

Experiment no 6

Aim: Implementation of SMS and telephony in android

Objective: Student should be able to implement service of SMS and telephony

Outcome: Student will develop a android application which will include service of SMS and telephony

Theory

In Android, you can use SmsManager API or devices Built-in SMS application to send SMS's. In this two basic examples to send SMS message are given

SmsManager API

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage("phoneNo", null, "sms message", null, null);
```

Built-in SMS application

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.putExtra("sms_body", "default content");
sendIntent.setType("vnd.android-dir/mms-sms");
startActivity(sendIntent);
```

Of course, both need **SEND_SMS permission**.

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

Apart from the above method, there are few other important functions available in SmsManager class. These methods are listed below

Sr.No.	Method & Description
1	**ArrayList<String> divideMessage(String text)** This method divides a message text into several fragments, none bigger than the maximum SMS message size.
2	**static SmsManager getDefault()** This method is used to get the default instance of the SmsManager

3	**void sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)** This method is used to send a data based SMS to a specific application port.
4	**void sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList<PendingIntent> deliveryIntents)** Send a multi-part text based SMS.
5	**void sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)** Send a text based SMS.

Example

Following example shows you in practical how to use SmsManager object to send an SMS to the given mobile number. To experiment with this example, you will need actual Mobile device equipped with latest Android OS, otherwise you will have to struggle with emulator which may not work.

Step	Description
1	You will use Android Studio IDE to create an Android application and name it as tutorialspoint under a package com.example.tutorialspoint.
2	Modify src/MainActivity.java file and add required code to take care of sending sms.
3	Modify layout XML file res/layout/activity_main.xml add any GUI component if required. I'm adding a simple GUI to take mobile number and SMS text to be sent and a simple button to send SMS.
4	No need to define default string constants at res/values/strings.xml. Android studio takes care of default constants.
5	Modify AndroidManifest.xml as shown below
6	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.tutorialspoint/MainActivity.java**.

```
package com.example.tutorialspoint;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.app.Activity;

import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.telephony.SmsManager;

import android.util.Log;
import android.view.Menu;
import android.view.View;

import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {
   private static final int MY_PERMISSIONS_REQUEST_SEND_SMS =0 ;
   Button sendBtn;
   EditText txtphoneNo;
   EditText txtMessage;
   String phoneNo;
   String message;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      sendBtn = (Button) findViewById(R.id.btnSendSMS);
      txtphoneNo = (EditText) findViewById(R.id.editText);
      txtMessage = (EditText) findViewById(R.id.editText2);

      sendBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            sendSMSMessage();
         }
      });
   }

   protected void sendSMSMessage() {
      phoneNo = txtphoneNo.getText().toString();
      message = txtMessage.getText().toString();

      if (ContextCompat.checkSelfPermission(this,
         Manifest.permission.SEND_SMS)
```

```
        != PackageManager.PERMISSION_GRANTED) {
          if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.SEND_SMS)) {
          } else {
             ActivityCompat.requestPermissions(this,
              new String[]{Manifest.permission.SEND_SMS},
              MY_PERMISSIONS_REQUEST_SEND_SMS);
          }
      }
    }

@Override
  public void onRequestPermissionsResult(int requestCode,String permissions[], int[] grantResults) {
    switch (requestCode) {
      case MY_PERMISSIONS_REQUEST_SEND_SMS: {
        if (grantResults.length > 0
          && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(phoneNo, null, message, null, null);
            Toast.makeText(getApplicationContext(), "SMS sent.",
              Toast.LENGTH_LONG).show();
        } else {
          Toast.makeText(getApplicationContext(),
            "SMS faild, please try again.", Toast.LENGTH_LONG).show();
          return;
        }
      }
    }

  }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

Here abc indicates about tutorialspoint logo

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="MainActivity">

 <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sending SMS Example"
```

```
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:textSize="30dp" />

  <TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Tutorials point "
    android:textColor="#ff87ff09"
    android:textSize="30dp"
    android:layout_below="@+id/textView1"
    android:layout_alignRight="@+id/imageButton"
    android:layout_alignEnd="@+id/imageButton" />

<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageButton"
    android:src="@drawable/abc"
    android:layout_below="@+id/textView2"
    android:layout_centerHorizontal="true" />

  <EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText"
    android:hint="Enter Phone Number"
    android:phoneNumber="true"
    android:textColorHint="@color/abc_primary_text_material_dark"
    android:layout_below="@+id/imageButton"
    android:layout_centerHorizontal="true" />

  <EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText2"
    android:layout_below="@+id/editText"
    android:layout_alignLeft="@+id/editText"
    android:layout_alignStart="@+id/editText"
    android:textColorHint="@color/abc_primary_text_material_dark"
    android:layout_alignRight="@+id/imageButton"
    android:layout_alignEnd="@+id/imageButton"
    android:hint="Enter SMS" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Send Sms"
    android:id="@+id/btnSendSMS"
    android:layout_below="@+id/editText2"
    android:layout_centerHorizontal="true"
```

```
    android:layout_marginTop="48dp" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">tutorialspoint</string>
</resources>
```

Following is the default content of **AndroidManifest.xml** −

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.tutorialspoint" >

  <uses-permission android:name="android.permission.SEND_SMS" />

  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
      android:name="com.example.tutorialspoint.MainActivity"
      android:label="@string/app_name" >

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>

  </application>
</manifest>
```

Now you can enter a desired mobile number and a text message to be sent on that number. Finally click on **Send SMS** button to send your SMS. Make sure your GSM/CDMA connection is working fine to deliver your SMS to its recipient.

You can take a number of SMS separated by comma and then inside your program you will have to parse them into an array string and finally you can use a loop to send message to all the given numbers. That's how you can write your own SMS client. Next section will show you how to use existing SMS client to send SMS.

Using Built-in Intent to send SMS

You can use Android Intent to send SMS by calling built-in SMS functionality of the Android. Following section explains different parts of our Intent object required to send an SMS.

Intent Object - Action to send SMS

You will use **ACTION_VIEW** action to launch an SMS client installed on your Android device. Following is simple syntax to create an intent with ACTION_VIEW action.

```
Intent smsIntent = new Intent(Intent.ACTION_VIEW);
```

Intent Object - Data/Type to send SMS

To send an SMS you need to specify **smsto:** as URI using setData() method and data type will be to **vnd.android-dir/mms-sms** using setType() method as follows –

```
smsIntent.setData(Uri.parse("smsto:"));
smsIntent.setType("vnd.android-dir/mms-sms");
```

Intent Object - Extra to send SMS

Android has built-in support to add phone number and text message to send an SMS as follows −

```
smsIntent.putExtra("address"  , new String("0123456789;3393993300"));
smsIntent.putExtra("sms_body"  , "Test SMS to Angilla");
```

Here address and sms_body are case sensitive and should be specified in small characters only. You can specify more than one number in single string but separated by semi-colon (;).

Example

Following example shows you in practical how to use Intent object to launch SMS client to send an SMS to the given recipients.

To experiment with this example, you will need actual Mobile device equipped with latest Android OS, otherwise you will have to struggle with emulator which may not work.

Step	Description
1	You will use Android studio IDE to create an Android application and name it as tutorialspoint under a package com.example.tutorialspoint.
2	Modify src/MainActivity.java file and add required code to take care of sending SMS.
3	Modify layout XML file res/layout/activity_main.xml add any GUI component if required. I'm adding a simple button to launch SMS Client.
4	No need to define default constants.Android studio takes care of default constants.
5	Modify AndroidManifest.xml as shown below
6	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.tutorialspoint/MainActivity.java**.

```
package com.example.tutorialspoint;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button startBtn = (Button) findViewById(R.id.button);
      startBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            sendSMS();
         }
      });
   }

   protected void sendSMS() {
      Log.i("Send SMS", "");
      Intent smsIntent = new Intent(Intent.ACTION_VIEW);
      smsIntent.setData(Uri.parse("smsto:"));
      smsIntent.setType("vnd.android-dir/mms-sms");
      smsIntent.putExtra("address"  , new String ("01234"));
      smsIntent.putExtra("sms_body"  , "Test ");

      try {
         startActivity(smsIntent);
         finish();
         Log.i("Finished sending SMS...", "");
      } catch (android.content.ActivityNotFoundException ex) {
         Toast.makeText(MainActivity.this,
         "SMS faild, please try again later.", Toast.LENGTH_SHORT).show();
      }
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

Here abc indicates about tutorialspoint logo

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Drag and Drop Example"
    android:id="@+id/textView"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:textSize="30dp" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Tutorials Point "
    android:id="@+id/textView2"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true"
    android:textSize="30dp"
    android:textColor="#ff14be3c" />

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:src="@drawable/abc"
    android:layout_marginTop="48dp"
    android:layout_below="@+id/textView2"
    android:layout_centerHorizontal="true" />

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Compose SMS"
    android:id="@+id/button"
    android:layout_below="@+id/imageView"
    android:layout_alignRight="@+id/textView2"

    android:layout_alignEnd="@+id/textView2"
    android:layout_marginTop="54dp"
```

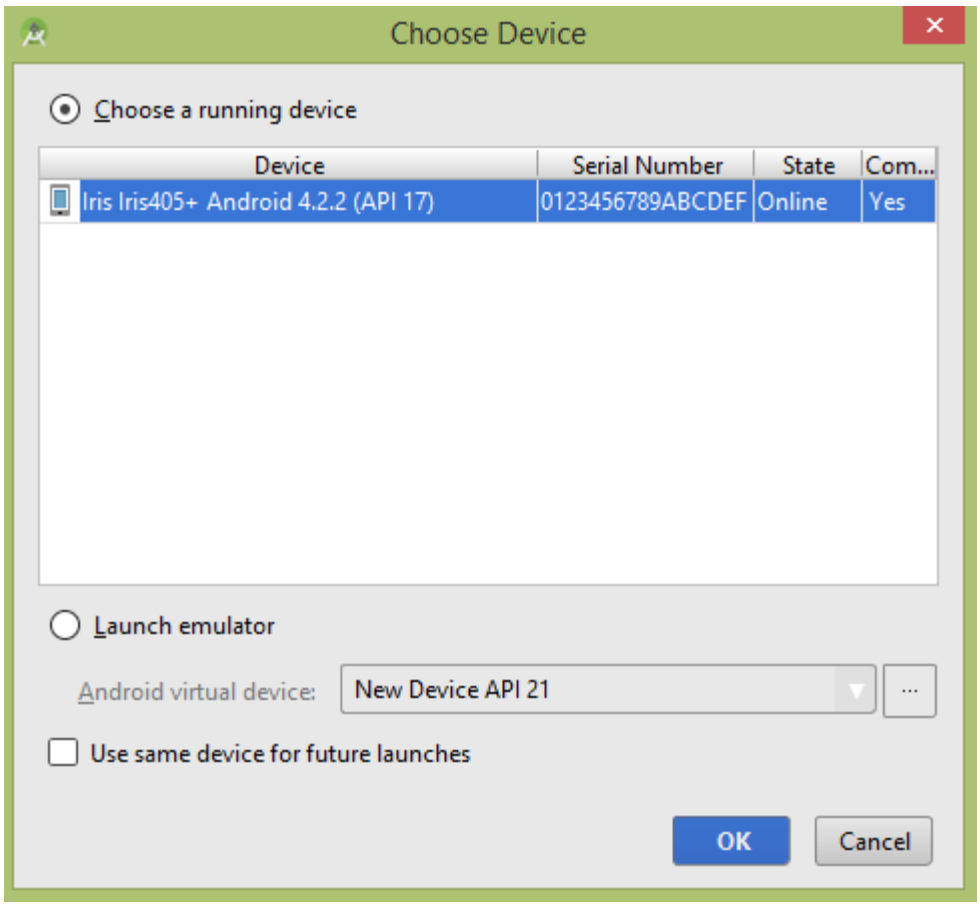
```
    android:layout_alignLeft="@+id/imageView"
    android:layout_alignStart="@+id/imageView" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">tutorialspoint</string>
</resources>
```

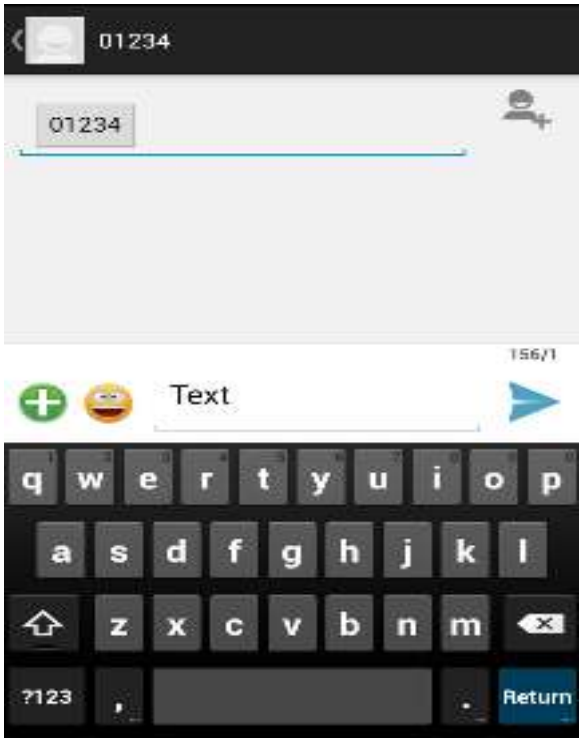
Following is the default content of **AndroidManifest.xml** −

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.tutorialspoint" >

  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
      android:name="com.example.tutorialspoint.MainActivity"
      android:label="@string/app_name" >

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>

  </application>
</manifest>
```



Select your mobile device as an option and then check your mobile device which will display following screen –

Now use **Compose SMS** button to launch Android built-in SMS clients which is shown below −



You can modify either of the given default fields and finally use send SMS button to send your SMS to the mentioned recipient.

Telephony Service

Android provides Built-in applications for phone calls, in some occasions we may need to make a phone call through our application. This could easily be done by using implicit Intent with appropriate actions. Also, we can use PhoneStateListener and TelephonyManager classes, in order to monitor the changes in some telephony states on the device.

Intent Object - Action to make Phone Call

You will use **ACTION_CALL** action to trigger built-in phone call functionality available in Android device. Following is simple syntax to create an intent with ACTION_CALL action

```
Intent phoneIntent = new Intent(Intent.ACTION_CALL);
```

You can use **ACTION_DIAL** action instead of ACTION_CALL, in that case you will have option to modify hardcoded phone number before making a call instead of making a direct call.

Intent Object - Data/Type to make Phone Call

To make a phone call at a given number 91-000-000-0000, you need to specify **tel:** as URI using setData() method as follows −

```
phoneIntent.setData(Uri.parse("tel:91-000-000-0000"));
```

The interesting point is that, to make a phone call, you do not need to specify any extra data or data type.

Example

Step	Description
1	You will use Android studio IDE to create an Android application and name it as My Application under a package com.example.saira_000.myapplication.
2	Modify src/MainActivity.java file and add required code to take care of making a call.
3	Modify layout XML file res/layout/activity_main.xml add any GUI component if required. I'm adding a simple button to Call 91-000-000-0000 number
4	No need to define default string constants.Android studio takes care of default constants.
5	Modify AndroidManifest.xml as shown below
6	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/MainActivity.java**.

```
package com.example.saira_000.myapplication;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {
  private Button button;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    button = (Button) findViewById(R.id.buttonCall);
```

```
    button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View arg0) {
          Intent callIntent = new Intent(Intent.ACTION_CALL);
          callIntent.setData(Uri.parse("tel:0377778888"));

          if (ActivityCompat.checkSelfPermission(MainActivity.this,
            Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
              return;
            }
            startActivity(callIntent);
        }
      });

  }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <Button
    android:id="@+id/buttonCall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="call 0377778888" />

</LinearLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

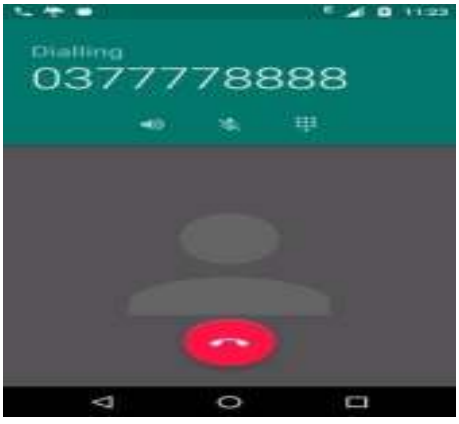
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">My Application</string>
</resources>
```

Following is the default content of **AndroidManifest.xml** −

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.saira_000.myapplication" >

  <uses-permission android:name="android.permission.CALL_PHONE" />

  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
```

```
android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
      android:name="com.example.saira_000.myapplication.MainActivity"
      android:label="@string/app_name" >

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>

  </application>
</manifest>
```

Now use **Call** button to make phone call as shown below −



<u>Conclusion:</u> Thus we have implemented the concept of SMS and telephony

EXPERIMENT NO 6

Aim: Implementation of broadcasts receiver in android

Objective: Student should be able to implement broadcasts receiver in android

Outcome: Student will develop a android application which will include broadcasts receiver

Theory

Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

There are following two important steps to make BroadcastReceiver works for the system broadcasted intents

- Creating the Broadcast Receiver.
- Registering Broadcast Receiver

There is one additional step in case you are going to implement your custom intents then you will have to create and broadcast those intents.

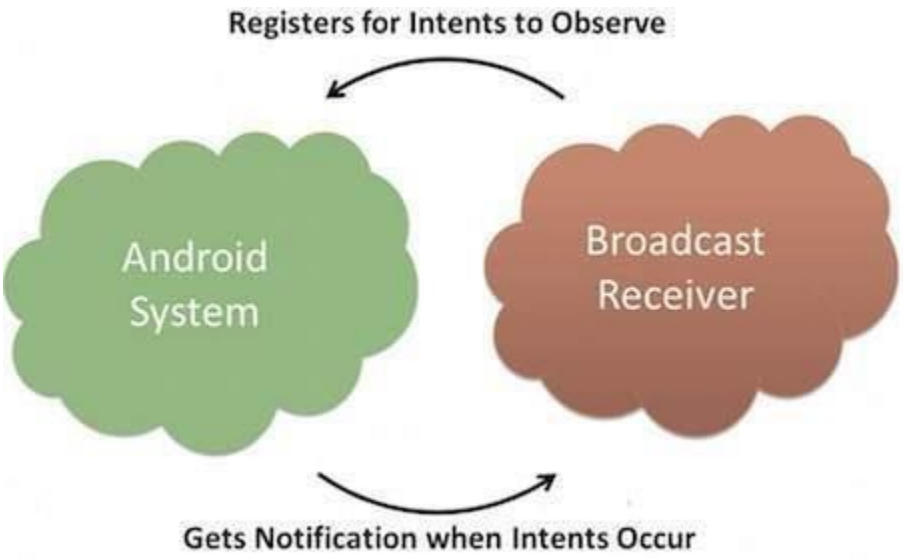
Creating the Broadcast Receiver

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and overriding the onReceive() method where each message is received as a **Intent** object parameter.

```
public class MyReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
   }
}
```

Registering Broadcast Receiver

An application listens for specific broadcast intents by registering a broadcast receiver in *AndroidManifest.xml* file. Consider we are going to register *MyReceiver* for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.



Broadcast-Receiver

```
<application
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name"
  android:theme="@style/AppTheme" >
  <receiver android:name="MyReceiver">

    <intent-filter>
      <action android:name="android.intent.action.BOOT_COMPLETED">
      </action>
    </intent-filter>

  </receiver>
</application>
```

Now whenever your Android device gets booted, it will be intercepted by BroadcastReceiver *MyReceiver* and implemented logic inside *onReceive()* will be executed.There are several system generated events defined as final static fields in the **Intent** class. The following table lists a few important system events.

Sr.No	Event Constant & Description
1	**android.intent.action.BATTERY_CHANGED** Sticky broadcast containing the charging state, level, and other information about the battery.
2	**android.intent.action.BATTERY_LOW** Indicates low battery condition on the device.
3	**android.intent.action.BATTERY_OKAY**

	Indicates the battery is now okay after being low.
4	**android.intent.action.BOOT_COMPLETED** This is broadcast once, after the system has finished booting.
5	**android.intent.action.BUG_REPORT** Show activity for reporting a bug.
6	**android.intent.action.CALL** Perform a call to someone specified by the data.
7	**android.intent.action.CALL_BUTTON** The user pressed the "call" button to go to the dialer or other appropriate UI for placing a call.
8	**android.intent.action.DATE_CHANGED** The date has changed.
9	**android.intent.action.REBOOT** Have the device reboot.

Broadcasting Custom Intents

If you want your application itself should generate and send custom intents then you will have to create and send those intents by using the *sendBroadcast()* method inside your activity class. If you use the *sendStickyBroadcast(Intent)* method, the Intent is **sticky**, meaning the *Intent* you are sending stays around after the broadcast is complete.

```
public void broadcastIntent(View view) {
   Intent intent = new Intent();
   intent.setAction("com.tutorialspoint.CUSTOM_INTENT");
   sendBroadcast(intent);
}
```

This intent *com.tutorialspoint.CUSTOM_INTENT* can also be registered in similar way as we have registered system generated intent.

```
<application
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name"
  android:theme="@style/AppTheme" >
  <receiver android:name="MyReceiver">

    <intent-filter>
      <action android:name="com.tutorialspoint.CUSTOM_INTENT">
      </action>
    </intent-filter>

  </receiver>
</application>
```

Example

This example will explain you how to create *BroadcastReceiver* to intercept custom intent. Once you are familiar with custom intent, then you can program your application to intercept system generated intents.

Step	Description
1	You will use Android studio to create an Android application and name it as *My Application* under a package *com.example.tutorialspoint7.myapplication* as explained in the *Hello World Example* chapter.
2	Modify main activity file *MainActivity.java* to add *broadcastIntent()* method.
3	Create a new java file called *MyReceiver.java* under the package *com.example.tutorialspoint7.myapplication* to define a BroadcastReceiver.
4	An application can handle one or more custom and system intents without any restrictions. Every intent you want to intercept must be registered in your *AndroidManifest.xml* file using <receiver.../> tag
5	Modify the default content of *res/layout/activity_main.xml* file to include a button to broadcast intent.
6	No need to modify the string file, Android studio take care of string.xml file.
7	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **MainActivity.java**. This file can include each of the fundamental life cycle methods. We have added *broadcastIntent()* method to broadcast a custom intent.

```
package com.example.tutorialspoint7.myapplication;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends Activity {

  /** Called when the activity is first created. */
  @Override

  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }


  // broadcast a custom intent.

  public void broadcastIntent(View view){
    Intent intent = new Intent();
    intent.setAction("com.tutorialspoint.CUSTOM_INTENT"); sendBroadcast(intent);
  }
}
```

Following is the content of **MyReceiver.java**:

```
package com.example.tutorialspoint7.myapplication;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

/**
 * Created by TutorialsPoint7 on 8/23/2016.
 */
public class MyReceiver extends BroadcastReceiver{
  @Override
  public void onReceive(Context context, Intent intent) {
    Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
  }
}
```

Following will the modified content of *AndroidManifest.xml* file. Here we have added <receiver.../> tag to include our service:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.tutorialspoint7.myapplication">

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <receiver android:name="MyReceiver">
      <intent-filter>
        <action android:name="com.tutorialspoint.CUSTOM_INTENT">
        </action>
      </intent-filter>

    </receiver>
  </application>

</manifest>
```

Following will be the content of **res/layout/activity_main.xml** file to include a button to broadcast our custom intent −

```
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">
```

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Example of Broadcast"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:textSize="30dp" />

  <TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Tutorials point "
    android:textColor="#ff87ff09"
    android:textSize="30dp"
    android:layout_above="@+id/imageButton"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="40dp" />

  <ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageButton"
    android:src="@drawable/abc"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/button2"
    android:text="Broadcast Intent"
    android:onClick="broadcastIntent"
    android:layout_below="@+id/imageButton"
    android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Now to broadcast our custom intent, let's click on **Broadcast Intent** button, this will broadcast our custom intent "com.tutorialspoint.CUSTOM_INTENT" which will be intercepted by our registered BroadcastReceiver i.e. MyReceiver and as per our implemented logic a toast will appear on the bottom of the the simulator as follows −

Conclusion: Thus we have studied and implemented the concept of broadcast receiver.

EXPERIMENT NO 7

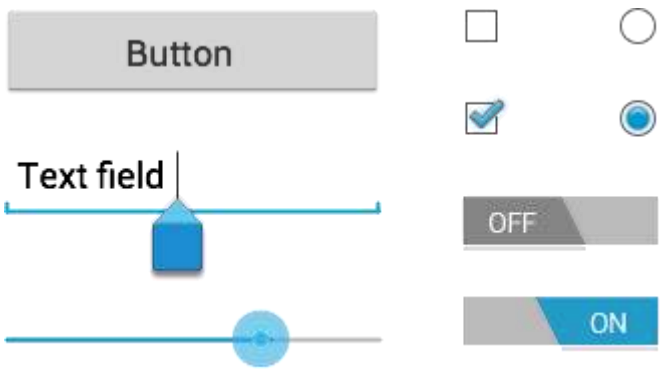
Aim: Implementation of program to demonstrate Buttons, Text Fields, Checkboxes, Radio Buttons, and Toggle Buttons with their events handler.

Objective: Student should be able to implement program to demonstrate Buttons, Text Fields, Checkboxes, Radio Buttons, and Toggle Buttons with their events handler.

Outcome: Student will develop a android application which will include Buttons, Text Fields, Checkboxes, Radio Buttons, and Toggle Buttons with their events handler.

Theory:

In android **UI** or **input** controls are the interactive or View components that are used to design the user interface of an application. In android we have a wide variety of UI or input controls available, those are TextView, EditText, Buttons, Checkbox, Progressbar, Spinners, etc.
Following is the pictorial representation of user interface (UI) or input controls in android application.



Generally, in android the user interface of an app is made with a collection of **View** and **ViewGroup** objects. The **View** is a base class for all UI components in android and it is used to create interactive UI components such as TextView, EditText, Checkbox, Radio Button, etc. and it is responsible for event handling and drawing.The **ViewGroup** is a subclass of **View** and it will act as a base class for layouts and layout parameters. The ViewGroup will provide invisible containers to hold other Views or ViewGroups and to define the layout properties.
To know more about View and ViewGroup in android applications, check this Android View and ViewGroup.

In android, we can define a UI or input controls in two ways, those are

- Declare UI elements in XML
- Create UI elements at runtime

The android framework will allow us to use either or both of these methods to define our application's UI.

Declare UI Elements in XML

In android, we can create layouts same as web pages in HTML by using default **Views** and **ViewGroups** in the XML file. The layout file must contain only one root element, which must be a **View** or **ViewGroup** object. Once we define the root element, then we can add additional layout objects or widgets as a child elements to build View hierarchy that defines our layout.

Following is the example of defining UI controls (TextView, EditText, Button) in the XML file (**activity_main.xml**) using LinearLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView

android:id="@+id/fstTxt"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Enter Name" />
   <EditText
      android:id="@+id/name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:ems="10"/>
   <Button
      android:id="@+id/getName"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Get Name" />
</LinearLayout>
```

In android, each input control is having a specific set of events and these events will be raised when the user performs particular action like, entering the text or touches the button.

Note: we need to create a user interface (**UI**) layout files in **/res/layout** project directory, then only the layout files will compile properly.

Load XML Layout File from an Activity

Once we are done with the creation of layout with UI controls, we need to load the XML layout resource from our activity **onCreate()** callback method like as shown below.

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
}
```

If you observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name**. Here our xml file name is **activity_main.xml** so we used file name **activity_main**.

Generally, during the launch of our activity, **onCreate()** callback method will be called by android framework to get the required layout for an activity.

Create UI Element at Runtime

If we want to create UI elements at runtime, we need to create our own custom **View** and **ViewGroup** objects programmatically with required layouts.

Following is the example of creating UI elements (TextView, EditText, Button) in LinearLayout using custom **View** and **ViewGroup** objects in an activity programmatically.

```
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      TextView textView1 = new TextView(this);
      textView1.setText("Name:");
      EditText editText1 = new EditText(this);
      editText1.setText("Enter Name");
      Button button1 = new Button(this);
      button1.setText("Add Name");

 LinearLayout linearLayout = new LinearLayout(this);
      linearLayout.addView(textView1);
      linearLayout.addView(editText1);
      linearLayout.addView(button1);
      setContentView(linearLayout);
   }
}
```

By creating a custom **View** and **ViewGroups** programmatically, we can define UI controls in layouts based on our requirements in android applications.

Width and Height

When we define a UI controls in a layout using an XML file, we need to set width and height for every **View** and **ViewGroup** elements using **layout_width** and **layout_height** attributes.

Following is the example of setting width and height for **View** and **ViewGroup** elements in the XML layout file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:id="@+id/fstTxt"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
    android:text="Enter Name" />
</LinearLayout>
```

If you observe above example, we used different values to set layout_width and layout_height, those are

- match_parent
- wrap_content

If we set value **match_parent**, then the **View** or **ViewGroup** will try to match with parent width or height.

If we set value **wrap_content**, then the **View** or **ViewGroup** will try to adjust its width or height based on the content.

Android Different Types of UI Controls

We have a different type of UI controls available in android to implement the user interface for our android applications.

Following are the commonly used UI or input controls in android applications.

- TextView
- EditText
- AutoCompleteTextView
- Button
- ImageButton
- ToggleButton
- CheckBox
- RadioButton
- RadioGroup
- ProgressBar
- Spinner
- TimePicker
- DatePicker
- SeekBar
- AlertDialog
- Switch
- RatingBar

Android TextView

In android, **TextView** is a user interface control that is used to display the text to the user.

Android EditText

In android, **EditText** is a user interface control which is used to allow the user to enter or modify the text.

To know more about EditText, check this Android EditText with Examples. Android AutoCompleteTextView

In android, **AutoCompleteTextView** is an editable text view which is used to show the list of suggestions based on the user typing text. The list of suggestions will be shown as a dropdown menu from which the user can choose an item to replace the content of the textbox.

To know more about AutoCompleteTextView, check this Android AutoCompleteTextView with Examples.

Android Button

In android, **Button** is a user interface control that is used to perform an action when the user clicks or tap on it. To know more about Button in android check this, Android Buttons with Examples.

Android Image Button

In android, **Image Button** is a user interface control that is used to display a button with an image to perform an action when the user clicks or tap on it. Generally, the Image button in android looks similar as regular Button and perform the actions same as regular button but only difference is for image button we will add an image instead of text. To know more about Image Button in android check this, Android Image Button with Examples.

Android Toggle Button

In android, **Toggle Button** is a user interface control that is used to display ON (Checked) or OFF (Unchecked) states as a button with a light indicator.
To know more about Toggle Button in android check this, Android Toggle Button with Examples.

Android CheckBox

In android, **Checkbox** is a two-states button that can be either checked or unchecked.To know more about CheckBox in android check this, Android CheckBox with Examples.

Android Radio Button

In android, **Radio Button** is a two-states button that can be either checked or unchecked and it cannot be unchecked once it is checked.

Android Radio Group

In android, **Radio Group** is used to group one or more radio buttons into separate groups based on our requirements.In case if we group radio buttons using the radio group, at a time only one item can be selected from the group of radio buttons.

Android ProgressBar

In android, **ProgressBar** is a user interface control which is used to indicate the progress of an operation.

Android Spinner

In android, **Spinner** is a drop-down list which allows a user to select one value from the list.

Android TimePicker
In android, **TimePicker** is a widget for selecting the time of day, either in 24-hour or AM/PM mode.

.**Android DatePicker**
In android, DatePicker is a widget for selecting a date.

Three TextViews labelled: (1) Unrestricted Text, (2) Phone input, and (3) Password input

1. Three EditText fields that go along with the TextViews above
2. Check Box
3. RadioGroup which has a couple of radio buttons, one for Male and other for Female
4. Button
5. Toggle Button

The code behind for the Mainactivity.xml file looks as shown:

```
<RelativeLayout xmlns_android="http://schemas.android. com/apk/res/android"

  xmlns:tools="http://schemas.android.com/tools"

  android:layout_width="match_parent"

  android:layout_height="match_parent"

 android:paddingLeft="@dimen/activity_horizontal_margin"

 android:paddingRight="@dimen/activity_horizontal_margin"

 android:paddingTop="@dimen/activity_vertical_margin"

 android:paddingBottom="@dimen/activity_vertical_margin"

 tools:context=".MainActivity">

  <TextView

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:textAppearance="?android:attr/textAppearanceSmall"

    android:text="Unrestricted Text"

    android:id="@+id/textView"
```

```
    android:layout_alignParentTop="true"
   android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editTextUnrestrictedText"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="Phone input"
    android:id="@+id/textView2"
    android:layout_below="@+id/editTextUnrestrictedText"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="phone"
    android:ems="10"
```

```
    android:id="@+id/editTextPhone"
    android:layout_below="@+id/editTextUnrestrictedText"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="60dp"
    android:layout_marginEnd="60dp" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="Password input"
    android:id="@+id/textView3"
    android:layout_below="@+id/editTextPhone"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="35dp" />
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:id="@+id/editTextPassword"
    android:layout_alignBottom="@+id/textView3"
```

```
    android:layout_centerHorizontal="true" />
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New CheckBox"
    android:id="@+id/checkBoxSelect"
    android:layout_below="@+id/textView3"
    android:layout_toLeftOf="@+id/editTextUnrestrictedText"
    android:layout_toStartOf="@+id/editTextUnrestrictedText"
    android:layout_marginTop="33dp"
    android:onClick="onClick" />
<RadioGroup
    android:id="@+id/radiogroup"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_below="@+id/checkBoxSelect">
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Male"
    android:id="@+id/radioButtonMale"
    android:layout_centerVertical="true"
```

```
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:onClick="onClick" />
  <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Female"
    android:id="@+id/radioButtonFemale"
    android:layout_below="@+id/radioButtonMale"
    android:layout_alignRight="@+id/radioButtonMale"
    android:layout_alignEnd="@+id/radioButtonMale"
    android:onClick="onClick" />
    </RadioGroup>
  <ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click here to toggle"
    android:id="@+id/toggleButton"
    android:layout_below="@+id/radiogroup"
    android:layout_alignRight="@+id/checkBoxSelect"
    android:layout_alignEnd="@+id/checkBoxSelect"
    android:textOn="Started"
    android:textOff="Stopped"
```

```
    android:layout_marginTop="43dp"
    android:onClick="onClick" />
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a button"
    android:id="@+id/button"
    android:layout_below="@+id/toggleButton"
    android:layout_alignRight="@+id/editTextUnrestrictedText"
    android:layout_alignEnd="@+id/editTextUnrestrictedText"
    android:onClick="onClick" />
</RelativeLayout>
```

We then update the MainActivity.java class to introduce member variables for the various input controls.

```
//Mainactivity.java
public class MainActivity extends ActionBarActivity {
  ToggleButton toggleButton;
  Button button;
  EditText editTextUnrestrictedText, editTextPhone,
   editTextPassword;
  RadioGroup radioGroup;
  RadioButton radioButtonMale, radioButtonFemale;
  CheckBox checkbox;
  @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

In the OnCreate method, we initialize these member variables.

```
@Override

protected voidonCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);

  setContentView(R.layout.activity_main);

  toggleButton =

    (ToggleButton)findViewById(R.id.toggleButton);

  button = (Button)findViewById(R.id.button);

  editTextUnrestrictedText =

    (EditText)findViewById(R.id.editTextUnrestrictedText);

  editTextPhone = (EditText)findViewById(R.id.editTextPhone);

  editTextPassword = (EditText)findViewById(R.id.editTextPassword);

  radioGroup = (RadioGroup)findViewById(R.id.radiogroup);

  radioButtonMale = (RadioButton)findViewById(R.id.radioButtonMale);

  radioButtonFemale= (RadioButton)findViewById(R.id.radioButtonFemale);

  checkbox = (CheckBox)findViewById(R.id.checkBoxSelect);

}
```

Finally, we implement the generic onClick event handler method.

```
public void onClick(View v) {

if (v.getId() == R.id.toggleButton) {

  if(toggleButton.isChecked())

    Toast.makeText(getApplicationContext(), "Toggle - Started",
```

```
        Toast.LENGTH_SHORT).show();
    else
      Toast.makeText(getApplicationContext(), "Toggle - Stopped",
        Toast.LENGTH_SHORT).show();
}
if(v.getId() == R.id.checkBoxSelect) {
    if(checkbox.isChecked())
      Toast.makeText(getApplicationContext(), "CheckBox - Checked",
        Toast.LENGTH_SHORT).show();
    else
      Toast.makeText(getApplicationContext(), "CheckBox - Unchecked",
        Toast.LENGTH_SHORT).show();
}
    if(v.getId() == R.id.radioButtonFemale) {
      if(radioButtonFemale.isChecked())
        Toast.makeText(getApplicationContext(), "RadioButton - Female",
          Toast.LENGTH_SHORT).show();
}
    if(v.getId() == R.id.radioButtonMale) {
      if(radioButtonMale.isChecked())
        Toast.makeText(getApplicationContext(), "RadioButton - Male",
          Toast.LENGTH_SHORT).show();
}
```

```
 if(v.getId() == R.id.button) {

     Toast.makeText(getApplicationContext(), "Button - Clicked",

        Toast.LENGTH_SHORT).show();

   }

}
```

Our application is now complete. When we run our application, you will notice that the input keyboard changes the available keys when we enter text for the various EditText fields.When the various checkbox, radio buttons, toggle buttons, and button controls are clicked, a toast is shown to the user indicating that there was interaction with the control.

```
public class TimePickerFrag extends DialogFragment

   implements TimePickerDialog.OnTimeSetListener {
```

Next, we implement the onCreateDialog method and onTimeSet methods.

```
//timepickerfrag.java

package com.example.vipul.pickercontrolsdemo;

import android.app.Dialog;

import android.app.TimePickerDialog;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v4.app.DialogFragment;

import android.text.format.DateFormat;

import android.widget.TimePicker;

import java.util.Calendar;

public class TimePickerFrag extends DialogFragment

implements TimePickerDialog.OnTimeSetListener {
```

```
@NonNull
  @Override
  public Dialog onCreateDialog(Bundle savedInstanceState) {
    final Calendar calendar = Calendar.getInstance();
    int hh = calendar.get(Calendar.HOUR_OF_DAY);
    int mm = calendar.get(Calendar.MINUTE);
    return new TimePickerDialog(getActivity(), this, hh, mm,
    DateFormat.is24HourFormat(getActivity()));
  }
  @Override
  public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    // do something
  }
}
```

Similarly, we implement the DialogFragment for Date Picker. Add another Java class, called DatePickerFrag, to the project.

```
//DatepickerFrag.java
package com.example.vipul.pickercontrolsdemo;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.DialogFragment;
import android.widget.DatePicker;
```

```
import java.util.Calendar;

public class DatePickerFrag extends DialogFragment

    implements DatePickerDialog.OnDateSetListener {

  @NonNull

  @Override

  public Dialog onCreateDialog(Bundle savedInstanceState) {

final Calendar calendar = Calendar.getInstance();

    int yy = calendar.get(Calendar.YEAR);

    int mm = calendar.get(Calendar.MONTH);

    int dd = calendar.get(Calendar.DAY_OF_MONTH);

    return new DatePickerDialog(getActivity(),

       this, yy, mm, dd);

 }

  @Override

  public void onDateSet(DatePicker view, int year,

    int monthOfYear, int dayOfMonth) {

  }

}
```

Lastly, we add two buttons to MainActivity.xml and wire up the click events to instantiate the dialogs.

```
<RelativeLayout xmlns_android="http://schemas.android.com

    /apk/res/android"

  xmlns:tools="http://schemas.android.com/tools"
```

```
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin"
  tools:context=".MainActivity" >
<TextView android_text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView" />
 <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Time Picker"
    android:id="@+id/buttonTimePicker"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="70dp"
    android:onClick="onButtonTimePickerClick" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="Date Picker"
    android:id="@+id/buttonDatePicker"
    android:layout_centerVertical="true"
    android:layout_alignRight="@+id/buttonTimePicker"
    android:layout_alignEnd="@+id/buttonTimePicker" />
</RelativeLayout>
```

In MainActivity.java, we implement the click event handlers.

```
public void onButtonTimePickerClick(View view){
  android.support.v4.app.DialogFragment timepickerFrag =
    new TimePickerFrag();
  timepickerFrag.show(getSupportFragmentManager(),
    "mytimepicker");
}
public void onButtonDatePickerClick(View view){
  android.support.v4.app.DialogFragment datepickerFrag =
    new DatePickerFrag();
  datepickerFrag.show(getSupportFragmentManager(),
    "mydatepicker");
}
```

Our application is now complete. When we launch the application and click the buttons, we can see the pickers in action.

Conclusion: Thus we have studied program to demonstrate Buttons, Text Fields, Checkboxes, Radio Buttons, and Toggle Buttons with their events handler.

Experiment no 8

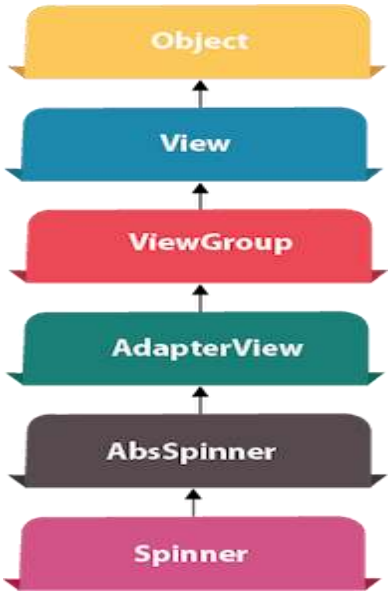
Title: Program to demonstrate Spinners, Touch Mode, Alerts, Popups, and Toasts with their events handler.

Objective: To develop android application to demonstrate Spinners, Touch Mode, Alerts, Popups, and Toasts with their events

Outcome: Students will be able to implement program to Spinners, Touch Mode, Alerts, Popups, and Toasts with their events

Theory

Android Spinner is like the combox box of AWT or Swing. It can be used to display the multiple options to the user in which only one item can be selected by the user. Android spinner is like the drop down menu with multiple values from which the end user can select only one value. Android spinner is associated with AdapterView. So you need to use one of the adapter classes with spinner. Android Spinner class is the subclass of AsbSpinner class.



Android Spinner Example

In this example, we are going to display the country list. You need to use **ArrayAdapter** class to store the country list.

activity_main.xml

Drag the Spinner from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
    <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:context="example.javatpoint.com.spinner.MainActivity">

        <Spinner
          android:id="@+id/spinner"
          android:layout_width="149dp"
          android:layout_height="40dp"
          android:layout_marginBottom="8dp"
          android:layout_marginEnd="8dp"
          android:layout_marginStart="8dp"
          android:layout_marginTop="8dp"
          app:layout_constraintBottom_toBottomOf="parent"
          app:layout_constraintEnd_toEndOf="parent"
          app:layout_constraintHorizontal_bias="0.502"
          app:layout_constraintStart_toStartOf="parent"
          app:layout_constraintTop_toTopOf="parent"
          app:layout_constraintVertical_bias="0.498" />

    </android.support.constraint.ConstraintLayout>
```

Activity class

Let's write the code to display item on the spinner and perform event handling.

File: MainActivity.java

```
package example.javatpoint.com.spinner;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity implements
    AdapterView.OnItemSelectedListener {
  String[] country = { "India", "USA", "China", "Japan", "Other"};

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
   //Getting the instance of Spinner and applying OnItemSelectedListener on it
    Spinner spin = (Spinner) findViewById(R.id.spinner);
    spin.setOnItemSelectedListener(this);

    //Creating the ArrayAdapter instance having the country list
    ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,country);
    aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    //Setting the ArrayAdapter data on the Spinner
    spin.setAdapter(aa);

  }

  //Performing action onItemSelected and onNothing selected
  @Override



public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id) {
    Toast.makeText(getApplicationContext(),country[position] , Toast.LENGTH_LONG).show();
  }
  @Override
  public void onNothingSelected(AdapterView<?> arg0) {
    // TODO Auto-generated method stub
  }
}
```

Output:

Conclusion: Thus we have studied program to demonstrate Spinners, Touch Mode, Alerts, Popups, and Toasts with their events handler.

Experiment no 9

Title: Implementation of program to demonstrate Touch Mode, Menus with their events handler.

Objective: To develop android application to demonstrate Touch Mode, Menus with their events handler.

Outcome: Students will be able to implement program to demonstrate Touch Mode, Menus with their events handler.

Theory:

Events are a useful way to collect data about a user's interaction with interactive components of Applications. Like button presses or screen touch etc. The Android framework maintains an event queue as first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate action as per requirements.

There are following three concepts related to Android Event Management −

- **Event Listeners** − An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.
- **Event Listeners Registration** − Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.
- **Event Handlers** − When an event happens and we have registered an event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Event Listeners & Event Handlers

Event Handler	Event Listener & Description
onClick()	**OnClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.
onLongClick()	**OnLongClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.
onFocusChange()	**OnFocusChangeListener()** This is called when the widget looses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.
onKey()	**OnFocusChangeListener()**

	This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.
onTouch()	**OnTouchListener()** This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.
onMenuItemClick()	**OnMenuItemClickListener()** This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.
onCreateContextMenu()	**onCreateContextMenuItemListener()** This is called when the context menu is being built(as the result of a sustained "long click)

There are many more event listeners available as a part of **View** class like OnHoverListener, OnDragListener etc which may be needed for your application. So I recommend to refer official documentation for Android application development in case you are going to develop a sophisticated apps.

Event Listeners Registration

Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event. Though there are several tricky ways to register your event listener for any event, but I'm going to list down only top 3 ways, out of which you can use any of them based on the situation.

- Using an Anonymous Inner Class
- Activity class implements the Listener interface.
- Using Layout file activity_main.xml to specify event handler directly.

Below section will provide you detailed examples on all the three scenarios −

Touch Mode

Users can interact with their devices by using hardware keys or buttons or touching the screen.Touching the screen puts the device into touch mode. The user can then interact with it by touching the on-screen virtual buttons, images, etc.You can check if the device is in touch mode by calling the View class's isInTouchMode() method.

Focus

A view or widget is usually highlighted or displays a flashing cursor when it's in focus. This indicates that it's ready to accept input from the user.

- **isFocusable()** − it returns true or false
- **isFocusableInTouchMode()** − checks to see if the view is focusable in touch mode. (A view may be focusable when using a hardware key but not when the device is in touch mode)

```
android:foucsUp="@=id/button_l"
```

onTouchEvent()

```
public boolean onTouchEvent(motionEvent event){
  switch(event.getAction()){
    case TOUCH_DOWN:
    Toast.makeText(this,"you have clicked down Touch button",Toast.LENTH_LONG).show();
    break();

 case TOUCH_UP:
    Toast.makeText(this,"you have clicked up touch button",Toast.LENTH_LONG).show();
    break;

    case TOUCH_MOVE:
    Toast.makeText(this,"you have clicked move touch button"Toast.LENTH_LONG).show();
    break;
  }
  return super.onTouchEvent(event) ;
}
```

Event Handling Examples

Event Listeners Registration Using an Anonymous Inner Class

Here you will create an anonymous implementation of the listener and will be useful if each class is applied to a single control only and you have advantage to pass arguments to event handler. In this approach event handler methods can access private data of Activity. No reference is needed to call to Activity.

But if you applied the handler to more than one control, you would have to cut and paste the code for the handler and if the code for the handler is long, it makes the code harder to maintain.

Following are the simple steps to show how we will make use of separate Listener class to register and capture click event. Similar way you can implement your listener for any other required event type.

Step	Description
1	You will use Android studio IDE to create an Android application and name it as *myapplication* under a package *com.example.myapplication* as explained in the *Hello World Example* chapter.
2	Modify *src/MainActivity.java* file to add click event listeners and handlers for the two buttons defined.
3	Modify the detault content of *res/layout/activity_main.xml* file to include Android UI controls.

4	No need to declare default string constants.Android studio takes care default constants.
5	Run the application to launch Android emulator and verify the result of the changes done in the aplication.

Following is the content of the modified main activity file **src/com.example.myapplication/MainActivity.java**. This file can include each of the fundamental lifecycle methods.

```
package com.example.myapplication;


import android.app.ProgressDialog;
import android.os.Bundle;

import android.support.v7.app.ActionBarActivity;

import android.view.View;
import android.widget.Button;

import android.widget.TextView;

public class MainActivity extends ActionBarActivity {
  private ProgressDialog progress;
  Button b1,b2;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    progress = new ProgressDialog(this);

    b1=(Button)findViewById(R.id.button);
    b2=(Button)findViewById(R.id.button2);
    b1.setOnClickListener(new View.OnClickListener() {

      @Override
      public void onClick(View v) {
        TextView txtView = (TextView) findViewById(R.id.textView);
        txtView.setTextSize(25);
      }
    });

    b2.setOnClickListener(new View.OnClickListener() {
```

```
@Override
        public void onClick(View v) {
          TextView txtView = (TextView) findViewById(R.id.textView);
          txtView.setTextSize(55);
         }
       });
   }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

Here abc indicates about tutorialspoint logo

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">


 <TextView
     android:id="@+id/textView1"
     android:layout_width="wrap_content"

 android:layout_height="wrap_content"

 android:text="Event Handling "
     android:layout_alignParentTop="true"
     android:layout_centerHorizontal="true"
     android:textSize="30dp"/>

   <TextView
     android:id="@+id/textView2"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Tutorials point "
     android:textColor="#ff87ff09"
     android:textSize="30dp"
     android:layout_above="@+id/imageButton"
     android:layout_centerHorizontal="true"
     android:layout_marginBottom="40dp" />
```

```
  <ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageButton"
    android:src="@drawable/abc"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Small font"
    android:id="@+id/button"
    android:layout_below="@+id/imageButton"
    android:layout_centerHorizontal="true" />

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Large Font"
    android:id="@+id/button2"
    android:layout_below="@+id/button"
    android:layout_alignRight="@+id/button"
    android:layout_alignEnd="@+id/button" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:id="@+id/textView"
    android:layout_below="@+id/button2"
    android:layout_centerHorizontal="true"
    android:textSize="25dp" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">myapplication</string>
</resources>
```

Following is the default content of **AndroidManifest.xml** −

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.myapplication" >

  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
      android:name="com.example.myapplication.MainActivity"
      android:label="@string/app_name" >

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>

  </application>
</manifest>
```

Android Option Menu Example

Android Option Menus are the primary menus of android. They can be used for settings, search, delete item etc.Here, we are going to see two examples of option menus. First, the simple option menus and second, options menus with images.Here, we are inflating the menu by calling the **inflate()** method of **MenuInflater** class. To perform event handling on menu items, you need to override **onOptionsItemSelected()** method of Activity class.

Android Option Menu Example

Let's see how to create menu in android. Let's see the simple option menu example that contains three menu items.

activity_main.xml

We have only one textview in this file.

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
  <android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/ap
  k/res/android"
     xmlns:app="http://schemas.android.com/apk/res-auto"
     xmlns:tools="http://schemas.android.com/tools"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     tools:context="example.javatpoint.com.optionmenu.MainActivity">
      <android.support.design.widget.AppBarLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
          android:id="@+id/toolbar"
          android:layout_width="match_parent"
          android:layout_height="?attr/actionBarSize"
          android:background="?attr/colorPrimary"
          app:popupTheme="@style/AppTheme.PopupOverlay" />
    </android.support.design.widget.AppBarLayout>
     <include layout="@layout/content_main" />
    </android.support.design.widget.CoordinatorLayout>
```

File: context_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/
android"    xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context="example.javatpoint.com.optionmenu.MainActivity"
   tools:showIn="@layout/activity_main">

 <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
  </android.support.constraint.ConstraintLayout>
```

menu_main.xml

It contains three items as show below. It is created automatically inside the res/menu directory.

File: menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools"
      tools:context="example.javatpoint.com.optionmenu.MainActivity">
    <item android:id="@+id/item1"
      android:title="Item 1"/>
     <item android:id="@+id/item2"
      android:title="Item 2"/>
     <item android:id="@+id/item3"
      android:title="Item 3"
      app:showAsAction="withText"/>
</menu>
```

Activity class

This class displays the content of menu.xml file and performs event handling on clicking the menu items.

File: MainActivity.java

```
package example.javatpoint.com.optionmenu;
  import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

setSupportActionBar(toolbar);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
      int id = item.getItemId();
        switch (id){
            case R.id.item1:
                Toast.makeText(getApplicationContext(),"Item 1 Selected",Toast.LENGTH_LONG).show();


return true;
            case R.id.item2:
```

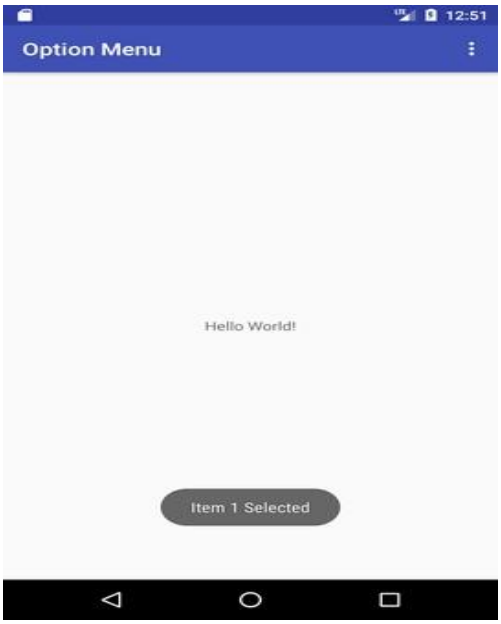
```
            Toast.makeText(getApplicationContext(),"Item 2 Selected",Toast.LENGTH_LONG).show();
                return true;
            case R.id.item3:
                Toast.makeText(getApplicationContext(),"Item 3 Selected",Toast.LENGTH_LONG).show();

                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Output:

Output without clicking on the menu button.

Output after clicking on the menu button.

Output after clicking on the second menu item .



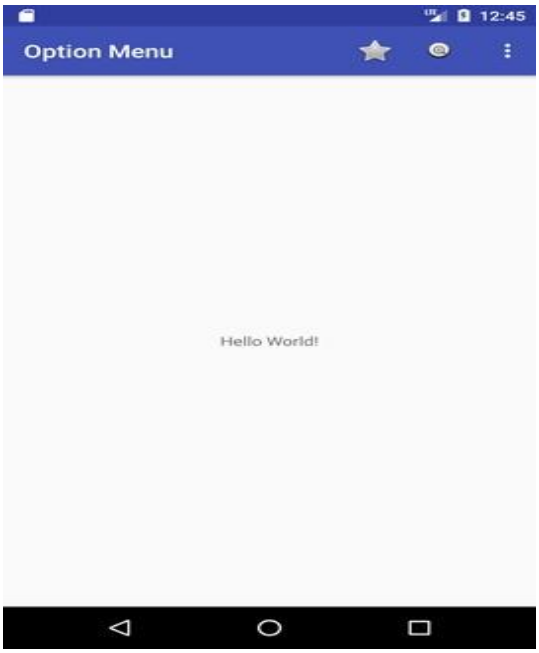
Option Menu with Icon

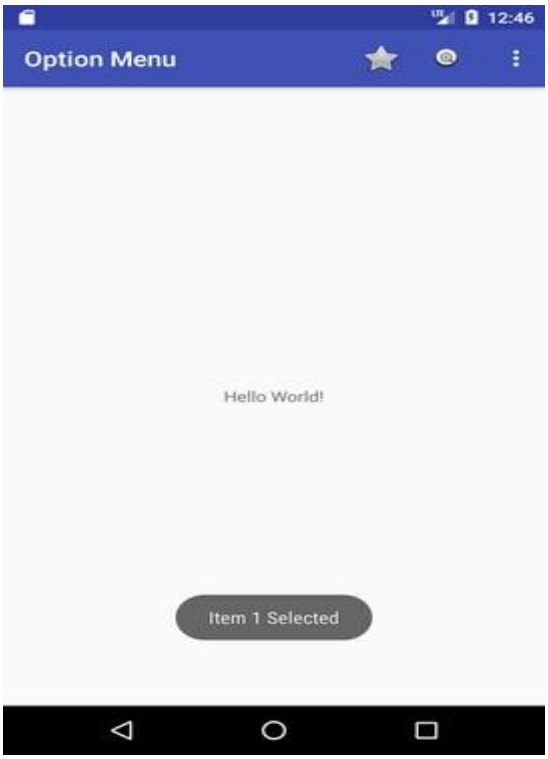
You need to have icon images inside the res/drawable directory. The android:icon element is used to display the icon on the option menu. You can write the string information in the strings.xml file. But we have written it inside the menu_main.xml file

File: menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="example.javatpoint.com.optionmenu.MainActivity">
```

```
    <item  android:id="@+id/item1"
        android:title="Item 1"
        app:showAsAction="always"
        android:icon="@android:drawable/btn_star"/>
    <item  android:id="@+id/item2"
        android:title="Item 2"
        app:showAsAction="ifRoom"
        android:icon="@android:drawable/btn_plus"/>
    <item  android:id="@+id/item3"
        android:title="Item 3"
        app:showAsAction="withText"
        android:icon="@android:drawable/btn_plus"/>
</menu>
```





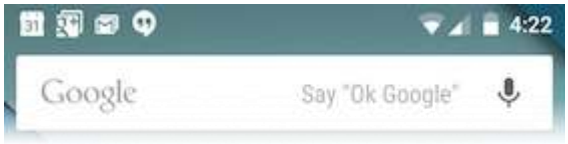
Conclusion: Thus we have studied program to demonstrate Touch Mode, Menus with their events handler

Experiment no 10

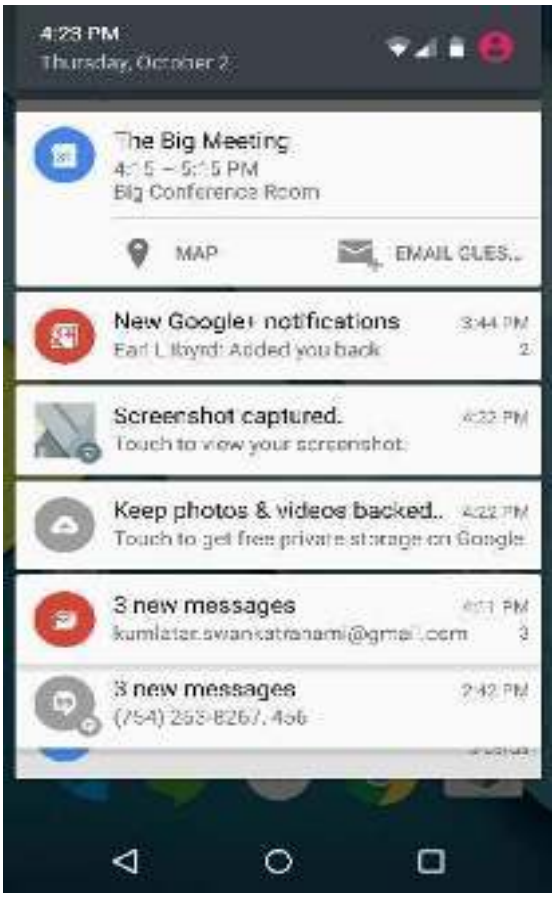
Title: Implementation of an android application displaying notification
Objective: To develop android application displaying notification
Outcome: Students will be able to implement program displaying notification
Theory:

A **notification** is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

Android **Toast** class provides a handy way to show users alerts but problem is that these alerts are not persistent which means alert flashes on the screen for a few seconds and then disappears.



To see the details of the notification, you will have to select the icon which will display notification drawer having detail about the notification. While working with emulator with virtual device, you will have to click and drag down the status bar to expand it which will give you detail as follows. This will be just **64 dp** tall and called normal view.



Above expanded form can have a **Big View** which will have additional detail about the notification. You can add upto six additional lines in the notification. The following screen shot shows such notification.

Create and Send Notifications

You have simple way to create a notification. Follow the following steps in your application to create a notification −

Step 1 - <u>Create Notification Builder</u>

As a first step is to create a notification builder using *NotificationCompat.Builder.build()*. You will use Notification Builder to set various Notification properties like its small and large icons, title, priority etc.

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
```

Step 2 - Setting Notification Properties

Once you have **Builder** object, you can set its Notification properties using Builder object as per your requirement. But this is mandatory to set at least following −

- A small icon, set by **setSmallIcon()**
- A title, set by **setContentTitle()**
- Detail text, set by **setContentText()**

```
mBuilder.setSmallIcon(R.drawable.notification_icon);
mBuilder.setContentTitle("Notification Alert, Click Me!");
mBuilder.setContentText("Hi, This is Android Notification Detail!");
```

You have plenty of optional properties which you can set for your notification. To learn more about them, see the reference documentation for NotificationCompat.Builder.

Step 3 - Attach Actions

This is an optional part and required if you want to attach an action with the notification. An action allows users to go directly from the notification to an **Activity** in your application, where they can look at one or more events or do further work.

The action is defined by a **PendingIntent** containing an **Intent** that starts an Activity in your application. To associate the PendingIntent with a gesture, call the appropriate method of *NotificationCompat.Builder*. For example, if you want to start Activity when the user clicks the notification text in the notification drawer, you add the PendingIntent by calling **setContentIntent()**.

A PendingIntent object helps you to perform an action on your applications behalf, often at a later time, without caring of whether or not your application is running.

We take help of stack builder object which will contain an artificial back stack for the started Activity. This ensures that navigating backward from the Activity leads out of your application to the Home screen.

```
Intent resultIntent = new Intent(this, ResultActivity.class);
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(ResultActivity.class);

// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
stackBuilder.getPendingIntent(0,PendingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(resultPendingIntent);
```

Step 4 - Issue the notification

Finally, you pass the Notification object to the system by calling NotificationManager.notify() to send your notification. Make sure you call **NotificationCompat.Builder.build()** method on builder object before notifying it. This method combines all of the options that have been set and return a new **Notification** object.

```
NotificationManager mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

// notificationID allows you to update the notification later on.
mNotificationManager.notify(notificationID, mBuilder.build());
```

The NotificationCompat.Builder Class

The NotificationCompat.Builder class allows easier control over all the flags, as well as help constructing the typical notification layouts. Following are few important and most frequently used methods available as a part of NotificationCompat.Builder class.

Sr.No.	Constants & Description
1	**Notification build()** Combine all of the options that have been set and return a new Notification object.
2	**NotificationCompat.Builder setAutoCancel (boolean autoCancel)** Setting this flag will make it so the notification is automatically canceled when the user clicks it in the panel.
3	**NotificationCompat.Builder setContent (RemoteViews views)** Supply a custom RemoteViews to use instead of the standard one.
4	**NotificationCompat.Builder setContentInfo (CharSequence info)** Set the large text at the right-hand side of the notification.
5	**NotificationCompat.Builder setContentIntent (PendingIntent intent)** Supply a PendingIntent to send when the notification is clicked.
6	**NotificationCompat.Builder setContentText (CharSequence text)** Set the text (second row) of the notification, in a standard notification.

7	**NotificationCompat.Builder setContentTitle (CharSequence title)** Set the text (first row) of the notification, in a standard notification.
8	**NotificationCompat.Builder setDefaults (int defaults)** Set the default notification options that will be used.
9	**NotificationCompat.Builder setLargeIcon (Bitmap icon)** Set the large icon that is shown in the ticker and notification.
10	**NotificationCompat.Builder setNumber (int number)** Set the large number at the right-hand side of the notification.
11	**NotificationCompat.Builder setOngoing (boolean ongoing)** Set whether this is an ongoing notification.
12	**NotificationCompat.Builder setSmallIcon (int icon)** Set the small icon to use in the notification layouts.
13	**NotificationCompat.Builder setStyle (NotificationCompat.Style style)** Add a rich notification style to be applied at build time.
14	**NotificationCompat.Builder setTicker (CharSequence tickerText)** Set the text that is displayed in the status bar when the notification first arrives.
15	**NotificationCompat.Builder setVibrate (long[] pattern)** Set the vibration pattern to use.
16	**NotificationCompat.Builder setWhen (long when)** Set the time that the event occurred. Notifications in the panel are sorted by this time.

Example

Following example shows the functionality of a Android notification using a **NotificationCompat.Builder** Class which has been introduced in Android

Step	Description
1	You will use Android studio IDE to create an Android application and name it as *tutorialspoint* under a package *com.example.notificationdemo*.
2	Modify *src/MainActivity.java* file and add the code to notify(""), if user click on the button,it will call android notification service.
3	Create a new Java file *src/NotificationView.java*, which will be used to display new layout as a part of new activity which will be started when user will click any of the notifications
4	Modify layout XML file *res/layout/activity_main.xml* to add Notification button in relative layout.
5	Create a new layout XML file *res/layout/notification.xml*. This will be used as layout file for new activity which will start when user will click any of the notifications.
6	No need to change default string constants. Android studio takes care of default string constants
7	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.notificationdemo/MainActivity.java**. This file can include each of the fundamental lifecycle methods.

```
package com.example.notificationdemo;

import android.app.Activity;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
```

```
import android.support.v4.app.NotificationCompat;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
   Button b1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      b1 = (Button)findViewById(R.id.button);
      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            addNotification();
         }
      });
   }


private void addNotification() {
      NotificationCompat.Builder builder =
         new NotificationCompat.Builder(this)
         .setSmallIcon(R.drawable.abc)
         .setContentTitle("Notifications Example")
         .setContentText("This is a test notification");

      Intent notificationIntent = new Intent(this, MainActivity.class);
      PendingIntent contentIntent = PendingIntent.getActivity(this, 0, notificationIntent,
         PendingIntent.FLAG_UPDATE_CURRENT);
      builder.setContentIntent(contentIntent);

      // Add as notification
      NotificationManager manager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
      manager.notify(0, builder.build());
   }
}
```

Following will be the content of **res/layout/notification.xml** file −

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent" >
```

```
   <TextView
      android:layout_width="fill_parent"
      android:layout_height="400dp"
      android:text="Hi, Your Detailed notification view goes here ... " />
</LinearLayout>
```

Following is the content of the modified main activity file **src/com.example.notificationdemo/NotificationView.java**.

```
package com.example.notificationdemo;

import android.os.Bundle;
import android.app.Activity;

public class NotificationView extends Activity{
   @Override
   public void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);
      setContentView(R.layout.notification);
   }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="MainActivity">

<TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Notification Example"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp" />
```

```
   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point "
      android:textColor="#ff87ff09"
      android:textSize="30dp"
      android:layout_below="@+id/textView1"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="48dp" />

   <ImageButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageButton"
      android:src="@drawable/abc"
      android:layout_below="@+id/textView2"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="42dp" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Notification"
      android:id="@+id/button"
      android:layout_marginTop="62dp"
      android:layout_below="@+id/imageButton"
      android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="action_settings">Settings</string>
   <string name="app_name">tutorialspoint </string>
</resources>
```

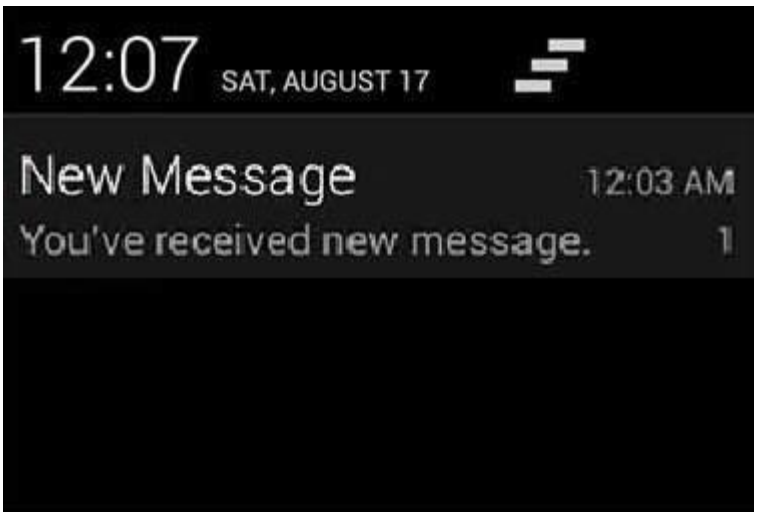
Following is the default content of **AndroidManifest.xml** −

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.notificationdemo" >

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
```

```
    <activity
      android:name="com.example.notificationdemo.MainActivity"
      android:label="@string/app_name" >

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>

    <activity android:name=".NotificationView"
      android:label="Details of notification"
      android:parentActivityName=".MainActivity">
      <meta-data
      android:name="android.support.PARENT_ACTIVITY"
      android:value=".MainActivity"/>
    </activity>

  </application>
</manifest>
```

Now click **button**, you will see at the top a message "New Message Alert!" will display momentarily and after that you will have following screen having a small icon at the top left corner.

Now lets expand the view, long click on the small icon, after a second it will display date information and this is the time when you should drag status bar down without releasing mouse. You will see status bar will expand and you will get following screen −



Conclusion: Thus we have studied and implemented program to demonstrate Touch Mode, Menus with their events handler.

Experiment no 11

Title: Implementation of an android application for calculator

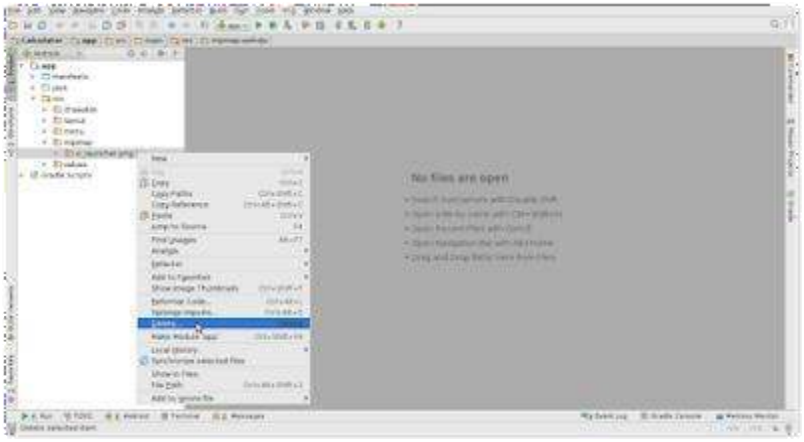
Objective: To develop android application for calculator

Outcome: Students will be able to implement program for calculator

Theory:

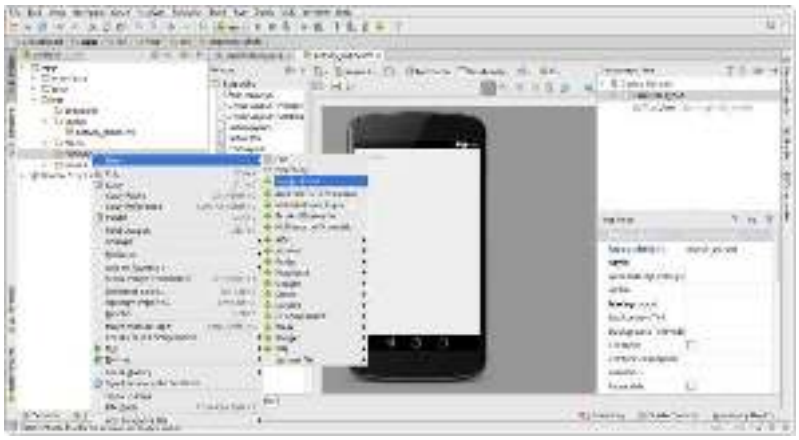
Step 1:
Create a new Android application project with an application name: "*Calculator*" and package name: "com.javahelps.calculator".

Step 2:
By default, Android uses a green Android robot icon. In this project, we are going to use a custom application icon. Therefore, delete the default ic_launcher icon set from the "mipmap" folder.

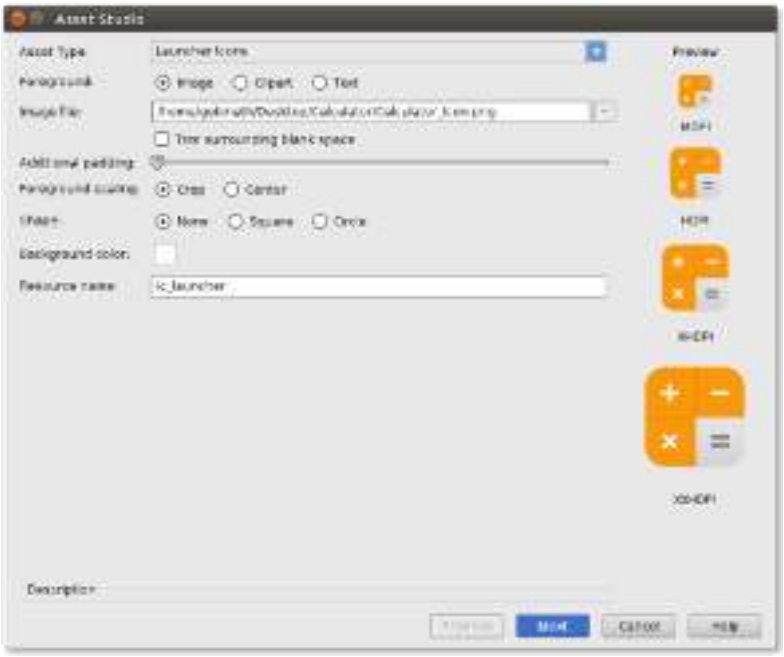


Step 3:
Get any PNG image file for the application icon. (It is recommended to have a minimum size 256x256 pixels). This icon is used to display in the Google Play as well as in the applications menu of Android devices.

Step 4:
Right click on the "*mipmap*" folder and select New → Image Asset



Step 5:
Browse and select your icon as the image file and click on Next → Finish buttons. (Make sure that the resource name is: *ic_launcher*)



Step 6:
Replace the content of the activity_main.xml file by the following code. This code creates a TextView as the calculator number screen and some necessary buttons. TextView is used instead of EditText, in order to prevent manual user input using the default keypad of Android. In this code, some common properties of Buttons are not provided to reduce the length of this tutorial. In your code make sure that you have included these four attributes for all the Buttons.

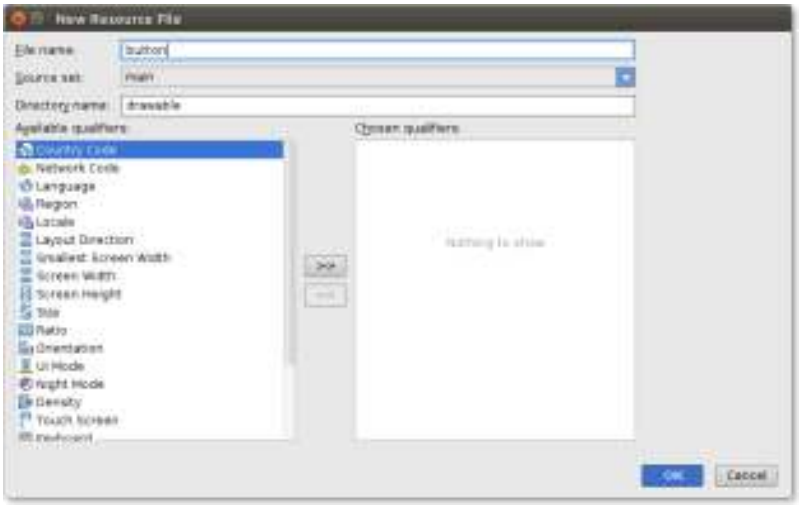
```
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:textSize="30sp"
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txtScreen"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:gravity="right|center_vertical"
        android:maxLength="16"
        android:padding="10dp"
        android:textAppearance="?android:attr/textAppearanceLarge"
```

```
    android:textSize="30sp"
    android:typeface="serif" />

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@+id/txtScreen"
    android:orientation="vertical">

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="1">
      <Button
        android:id="@+id/btnSeven"
        android:text="7" />
      <Button
        android:id="@+id/btnEight"
        android:text="8" />
      <Button
        android:id="@+id/btnNine"
        android:text="9"/>
      <Button
        android:id="@+id/btnDivide"
        android:text="/"/>
    </LinearLayout>

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="1">
      <Button
        android:id="@+id/btnFour"
        android:text="4"/>
      <Button
        android:id="@+id/btnFive"
        android:text="5" />
      <Button
        android:id="@+id/btnSix"
        android:text="6" />
      <Button
        android:id="@+id/btnMultiply"
        android:text="*" />
    </LinearLayout>

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="1">
      <Button
        android:id="@+id/btnOne"
        android:text="1" />
```

```
        <Button
            android:id="@+id/btnTwo"
            android:text="2" />
        <Button
            android:id="@+id/btnThree"
            android:text="3" />
        <Button
            android:id="@+id/btnSubtract"
            android:text="-" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1">
        <Button
            android:id="@+id/btnDot"
            android:text="." />
        <Button
            android:id="@+id/btnZero"
            android:text="0" />
        <Button
            android:id="@+id/btnClear"
            android:text="C" />
        <Button
            android:id="@+id/btnAdd"
            android:text="+" />
    </LinearLayout>

    <Button
        android:id="@+id/btnEqual"
        android:text="=" />
  </LinearLayout>
</RelativeLayout>
```

Step 7:
Right click on the "*drawable*" folder and select New → Drawable resource file.

Step 8:
Create a *drawable* file with a name button.



Step 9:
Replace the content of *button.xml* by the following code. This drawable resource is used to decorate the buttons of the calculator. There are two gradient shapes in this code; one is for button pressed state and another for the normal state.

```
<?xml version="1.0" encoding="utf-8" ?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:state_pressed="true">
      <shape>
         <gradient android:angle="90" android:endColor="#FFFFFF" android:startColor="#9EB8FF"
android:type="linear" />
         <padding android:bottom="0dp" android:left="0dp" android:right="0dp" android:top="0dp" />
         <size android:width="60dp" android:height="60dp" />
         <stroke android:width="1dp" android:color="#ff3da6ef" />
      </shape>
   </item>
   <item>
      <shape>
         <gradient android:angle="90" android:endColor="#FFFFFF" android:startColor="#ffd9d9d9"
android:type="linear" />
         <padding android:bottom="0dp" android:left="0dp" android:right="0dp" android:top="0dp" />
         <size android:width="60dp" android:height="60dp" />
         <stroke android:width="0.5dp" android:color="#ffcecece" />
      </shape>
   </item>
</selector>
```

Step 10:
For all the buttons in the *activity_main.xml*, add a property "*android:background*".

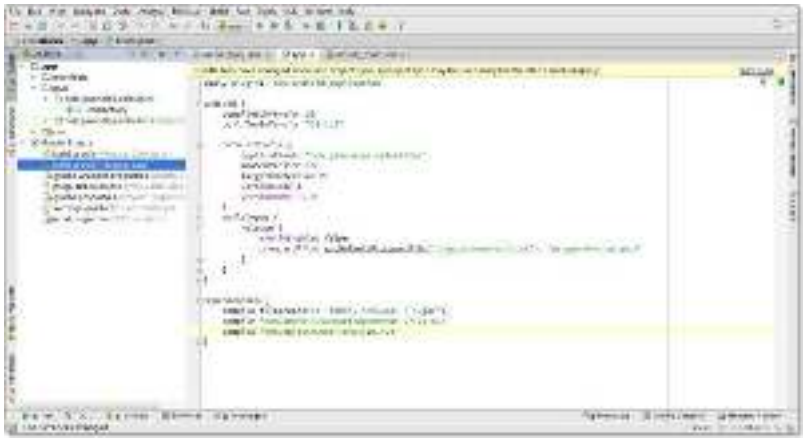
```
android:background="@drawable/button"
```

After the modification, *activity_main.xml* must be like this.

Step 11:
To evaluate the arithmetic expressions, the exp4J library is used in this project. Open the "*build.gradle (Module: app)*" file from the Gradle scripts. Add a dependency '*net.objecthunter:exp4j:0.4.4*' to the project as shown below.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile 'net.objecthunter:exp4j:0.4.4'
}
```

Once you save the file, Android Studio will ask to sync the project. Allow it to sync by clicking on the link appeared on the top left corner. (You need an active Internet connection to download the libraries by Gradle)



Step 12:
Modify the MainActivity.java as provided below. Complete description of the code is provided in comments.

```
package com.javahelps.calculator;

import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import net.objecthunter.exp4j.Expression;
import net.objecthunter.exp4j.ExpressionBuilder;


public class MainActivity extends ActionBarActivity {
    // IDs of all the numeric buttons
    private int[] numericButtons = {R.id.btnZero, R.id.btnOne, R.id.btnTwo, R.id.btnThree, R.id.btnFour,
R.id.btnFive, R.id.btnSix, R.id.btnSeven, R.id.btnEight, R.id.btnNine};
    // IDs of all the operator buttons
    private int[] operatorButtons = {R.id.btnAdd, R.id.btnSubtract, R.id.btnMultiply, R.id.btnDivide};
```

```
    // TextView used to display the output
    private TextView txtScreen;
    // Represent whether the lastly pressed key is numeric or not
    private boolean lastNumeric;
    // Represent that current state is in error or not
    private boolean stateError;
    // If true, do not allow to add another DOT
    private boolean lastDot;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Find the TextView
        this.txtScreen = (TextView) findViewById(R.id.txtScreen);
        // Find and set OnClickListener to numeric buttons
        setNumericOnClickListener();


// Find and set OnClickListener to operator buttons, equal button and decimal point button
        setOperatorOnClickListener();
    }

    /**
     * Find and set OnClickListener to numeric buttons.
     */
    private void setNumericOnClickListener() {
        // Create a common OnClickListener
        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Just append/set the text of clicked button
                Button button = (Button) v;
                if (stateError) {
                    // If current state is Error, replace the error message
                    txtScreen.setText(button.getText());
                    stateError = false;
                } else {
                    // If not, already there is a valid expression so append to it
                    txtScreen.append(button.getText());
                }
                // Set the flag
                lastNumeric = true;
            }
        };
        // Assign the listener to all the numeric buttons
        for (int id : numericButtons) {
            findViewById(id).setOnClickListener(listener);
        }
    }

    /**
```

```
    * Find and set OnClickListener to operator buttons, equal button and decimal point button.
    */
    private void setOperatorOnClickListener() {
        // Create a common OnClickListener for operators
        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // If the current state is Error do not append the operator
                // If the last input is number only, append the operator
                if (lastNumeric && !stateError) {
                    Button button = (Button) v;
                    txtScreen.append(button.getText());
                    lastNumeric = false;
                    lastDot = false;    // Reset the DOT flag
                }
            }
        };
        // Assign the listener to all the operator buttons
        for (int id : operatorButtons) {
            findViewById(id).setOnClickListener(listener);
        }
        // Decimal point
        findViewById(R.id.btnDot).setOnClickListener(new View.OnClickListener() {


    @Override
            public void onClick(View v) {
                if (lastNumeric && !stateError && !lastDot) {
                    txtScreen.append(".");
                    lastNumeric = false;
                    lastDot = true;
                }
            }
        });
        // Clear button
        findViewById(R.id.btnClear).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                txtScreen.setText("");  // Clear the screen
                // Reset all the states and flags
                lastNumeric = false;
                stateError = false;
                lastDot = false;
            }
        });
        // Equal button
        findViewById(R.id.btnEqual).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onEqual();
            }
        });
```

```
    }

    /**
     * Logic to calculate the solution.
     */
    private void onEqual() {
        // If the current state is error, nothing to do.
        // If the last input is a number only, solution can be found.
        if (lastNumeric && !stateError) {
            // Read the expression
            String txt = txtScreen.getText().toString();
            // Create an Expression (A class from exp4j library)
            Expression expression = new ExpressionBuilder(txt).build();
            try {
                // Calculate the result and display
                double result = expression.evaluate();
                txtScreen.setText(Double.toString(result));
                lastDot = true; // Result contains a dot
            } catch (ArithmeticException ex) {
                // Display an error message
                txtScreen.setText("Error");
                stateError = true;
                lastNumeric = false;
            }
        }
    }
}
```

Step 13:
Save all the changes and run the application.

Conclusion: Thus we have studied and implemented naïve calculator application

Experiment no 12

Title: Implementation of an android application that writes data to SD card
Objective: To develop android application writes data to SD card
Outcome: Students will be able to implement program that writes data to SD card
Theory:
Android external storage can be used to write and save data, read configuration files etc.

External storage such as SD card can also store application data, there's no security enforced upon files you save to the external storage.
In general there are two types of External Storage:

- **Primary External Storage**: In built shared storage which is "accessible by the user by plugging in a USB cable and mounting it as a drive on a host computer". Example: When we say Nexus 5 32 GB.
- **Secondary External Storage**: Removable storage. Example: SD Card

All applications can read and write files placed on the external storage and the user can remove them. We need to check if the SD card is available and if we can write to it. Once we've checked that the external storage is available only then we can write to it else the save button would be disabled.

Android External Storage Example

Like internal storage, we are able to save or read data from the device external memory such as sdcard. The FileInputStream and FileOutputStream classes are used to read and write data into the file.

Example of reading and writing data in the android external storage
activity_main.xml

Drag the 2 edittexts, 2 textviews and 2 buttons from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.externalstorage.MainActivity">
   <EditText
```

```
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:layout_marginRight="20dp"
    android:layout_marginTop="24dp"
    android:ems="10" >
      <requestFocus />
</EditText>
   <EditText
   android:id="@+id/editText2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
    android:layout_alignRight="@+id/editText1"
    android:layout_below="@+id/editText1"
    android:layout_marginTop="24dp"
    android:ems="10" />

  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/editText1"
    android:layout_alignBottom="@+id/editText1"
    android:layout_alignParentLeft="true"
    android:text="File Name:" />
  <TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/editText2"
    android:layout_alignBottom="@+id/editText2"
    android:layout_alignParentLeft="true"
    android:text="Data:" />
```

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText2"
    android:layout_below="@+id/editText2"
    android:layout_marginLeft="70dp"
    android:layout_marginTop="16dp"
    android:text="save" />

  <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/button1"
    android:layout_alignBottom="@+id/button1"
    android:layout_toRightOf="@+id/button1"
    android:text="read" />
</RelativeLayout>
```

You need to provide the WRITE_EXTERNAL_STORAGE permission.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

File: Activity_Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="example.javatpoint.com.externalstorage">
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

File: MainActivity.java

```
package example.javatpoint.com.externalstorage;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {
    EditText editTextFileName,editTextData;
    Button saveButton,readButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editTextFileName=findViewById(R.id.editText1);
        editTextData=findViewById(R.id.editText2);
        saveButton=findViewById(R.id.button1);
```

```
 readButton=findViewById(R.id.button2);


//Performing action on save button
saveButton.setOnClickListener(new View.OnClickListener(){

    @Override
    public void onClick(View arg0) {
        String filename=editTextFileName.getText().toString();
        String data=editTextData.getText().toString();

        FileOutputStream fos;
        try {
            File myFile = new File("/sdcard/"+filename);
      myFile.createNewFile();
            FileOutputStream fOut = new FileOutputStream(myFile);
            OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
            myOutWriter.append(data);
            myOutWriter.close();
            fOut.close();
      Toast.makeText(getApplicationContext(),filename + "saved",Toast.LENGTH_LONG).show();

        } catch (FileNotFoundException e) {e.printStackTrace();}
        catch (IOException e) {e.printStackTrace();}
    }
});


//Performing action on Read Button
readButton.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View arg0) {
        String filename=editTextFileName.getText().toString();
        StringBuffer stringBuffer = new StringBuffer();
        String aDataRow = "";
  String aBuffer = "";
        try {
            File myFile = new File("/sdcard/"+filename);
            FileInputStream fIn = new FileInputStream(myFile);
```

```
                BufferedReader myReader = new BufferedReader(
                        new InputStreamReader(fIn));
                while ((aDataRow = myReader.readLine()) != null) {
                aBuffer += aDataRow + "\n";
                }
                myReader.close();
                } catch (IOException e) {
                e.printStackTrace();
                }
                Toast.makeText(getApplicationContext(),aBuffer,Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

Conclusion: Thus we have studied and implemented technique for writing data to sd card

Experiment no 13

Title: Implementation of an android application for alarm clock
Objective: To develop android application dealing with alarm clock
Outcome: Students will be able to implement program of alarm clock
Theory:
The AlarmClock provider contains an Intent action and extras that can be used to start an Activity to set new alarm or timer in an alarm clock application. Applications that wish to receive the ACTION_SET_ALARM and ACTION_SET_TIMER Intents should create an activity to handle the Intent that requires the permission com.android.alarm.permission.SET_ALARM. Applications that wish to create a new alarm or timer should use Context.startActivity() so that the user has the option of choosing which alarm clock application to use. Android TV devices may not support the alarm intents.

ACTION_DISMISS_ALARM

public static final **String** ACTION_DISMISS_ALARM

Activity Action: Dismiss an alarm.

The alarm to dismiss can be specified or searched for in one of the following ways:

1. The Intent's data URI, which represents a deeplink to the alarm.
2. The extra EXTRA_ALARM_SEARCH_MODE to determine how to search for the alarm.

If neither of the above are given then:

- If exactly one active alarm exists, it is dismissed.
- If more than one active alarm exists, the user is prompted to choose the alarm to dismiss.

If the extra EXTRA_ALARM_SEARCH_MODE is used, and the search results contain two or more matching alarms, then the implementation should show an UI with the results and allow the user to select the alarm to dismiss. If the implementation supports Intent.CATEGORY_VOICE and the activity is started in Voice Interaction mode (i.e. check Activity.isVoiceInteraction()), then the implementation should additionally use VoiceInteractor.PickOptionRequest to start a voice interaction follow-on flow to help the user disambiguate the alarm by voice.

If the specified alarm is a single occurrence alarm, then dismissing it effectively disables the alarm; it will never ring again unless explicitly re-enabled.

If the specified alarm is a repeating alarm, then dismissing it only prevents the upcoming instance from ringing. The alarm remains enabled so that it will still ring on the date and time of the next instance (i.e. the instance after the upcoming one).

ACTION_DISMISS_TIMER

public static final **String** ACTION_DISMISS_TIMER

Activity Action: Dismiss a timer.

The timer to dismiss should be specified using the Intent's data URI, which represents a deep link to the timer.

If no data URI is provided, dismiss all expired timers.

Constant Value: "android.intent.action.DISMISS_TIMER"

ACTION_SET_ALARM

public static final **String** ACTION_SET_ALARM

Activity Action: Set an alarm.

Activates an existing alarm or creates a new one.

This action requests an alarm to be set for a given time of day. If no time of day is specified, an implementation should start an activity that is capable of setting an alarm (EXTRA_SKIP_UI is ignored in this case). If a time of day is specified, and EXTRA_SKIP_UI is true, and the alarm is not repeating, the implementation should remove this alarm after it has been dismissed. If an identical alarm exists matching all parameters, the implementation may re-use it instead of creating a new one (in this case, the alarm should not be removed after dismissal).

This action always enables the alarm.

This activity could also be started in Voice Interaction mode. The activity should check Activity.isVoiceInteraction(), and if true, the implementation should report a deeplink of the created/enabled alarm using VoiceInteractor.CompleteVoiceRequest. This allows follow-on voice actions such as ACTION_DISMISS_ALARM to dismiss the alarm that was just enabled.

Request parameters

- EXTRA_HOUR *(optional)*: The hour of the alarm being set.
- EXTRA_MINUTES *(optional)*: The minutes of the alarm being set.
- EXTRA_DAYS *(optional)*: Weekdays for repeating alarm.
- EXTRA_MESSAGE *(optional)*: A custom message for the alarm.
- EXTRA_RINGTONE *(optional)*: A ringtone to play with this alarm.
- EXTRA_VIBRATE *(optional)*: Whether or not to activate the device vibrator for this alarm.
- EXTRA_SKIP_UI *(optional)*: Whether or not to display an activity for setting this alarm.

Constant Value: "android.intent.action.SET_ALARM"

ACTION_SET_TIMER

public static final **String** ACTION_SET_TIMER

Activity Action: Set a timer.

Activates an existing timer or creates a new one.

This action requests a timer to be started for a specific length of time. If no length is specified, the implementation should start an activity that is capable of setting a timer (EXTRA_SKIP_UI is ignored in this case). If a length is specified, and EXTRA_SKIP_UI is true, the implementation should remove this timer after it has been dismissed. If an identical, unused timer exists matching both parameters, an implementation may re- use it instead of creating a new one (in this case, the timer should not be removed after dismissal). This action always starts the timer.

Request parameters

- EXTRA_LENGTH *(optional)*: The length of the timer being set.
- EXTRA_MESSAGE *(optional)*: A custom message for the timer.
- EXTRA_SKIP_UI *(optional)*: Whether or not to display an activity for setting this timer.

Constant Value: "android.intent.action.SET_TIMER"

ACTION_SHOW_ALARMS

public static final **String** ACTION_SHOW_ALARMS

Activity Action: Show the alarms.

This action opens the alarms page.

Constant Value: "android.intent.action.SHOW_ALARMS"

ACTION_SHOW_TIMERS

public static final **String** ACTION_SHOW_TIMERS

Activity Action: Show the timers.

This action opens the timers page.

Constant Value: "android.intent.action.SHOW_TIMERS"

ACTION_SNOOZE_ALARM

public static final **String** ACTION_SNOOZE_ALARM

Activity Action: Snooze a currently ringing alarm.

Snoozes the currently ringing alarm. The extra EXTRA_ALARM_SNOOZE_DURATION can be optionally set to specify the snooze duration; if unset, the implementation should use a reasonable default, for example 10 minutes. The alarm should ring again after the snooze duration.

Note: setting the extra EXTRA_ALARM_SNOOZE_DURATION does not change the default snooze duration; it's only applied to the currently ringing alarm.

If there is no currently ringing alarm, then this is a no-op.

ALARM_SEARCH_MODE_ALL

public static final **String** ALARM_SEARCH_MODE_ALL

Selects all alarms.

Constant Value: "android.all"

ALARM_SEARCH_MODE_LABEL

public static final **String** ALARM_SEARCH_MODE_LABEL

Search by alarm label. Should return alarms that contain the word or phrase in given label.

ALARM_SEARCH_MODE_NEXT

public static final **String** ALARM_SEARCH_MODE_NEXT

Selects the alarm that will ring next, or the alarm that is currently ringing, if any.

ALARM_SEARCH_MODE_TIME

public static final **String** ALARM_SEARCH_MODE_TIME

Search for the alarm that is most closely matched by the search parameters EXTRA_HOUR, EXTRA_MINUTES, EXTRA_IS_PM. In this search mode, at least one of these additional extras are required.

- EXTRA_HOUR - The hour to search for the alarm.
- EXTRA_MINUTES - The minute to search for the alarm.
- EXTRA_IS_PM - Whether the hour is AM or PM.

EXTRA_ALARM_SEARCH_MODE

public static final **String** EXTRA_ALARM_SEARCH_MODE

Bundle extra: Specify the type of search mode to look up an alarm.

For example, used by ACTION_DISMISS_ALARM to identify the alarm to dismiss.

This extra is only used when the alarm is not already identified by a deeplink as specified in the Intent's data URI.

The value of this extra is a String, restricted to the following set of supported search modes:

- *Time* - ALARM_SEARCH_MODE_TIME: Selects the alarm that is most closely matched by the search parameters EXTRA_HOUR, EXTRA_MINUTES, EXTRA_IS_PM.
- *Next alarm* - ALARM_SEARCH_MODE_NEXT: Selects the alarm that will ring next, or the alarm that is currently ringing, if any.
- *All alarms* - ALARM_SEARCH_MODE_ALL: Selects all alarms.
- *Label* - ALARM_SEARCH_MODE_LABEL: Search by alarm label. Should return alarms that contain the word or phrase in given label.

EXTRA_ALARM_SNOOZE_DURATION

public static final **String** EXTRA_ALARM_SNOOZE_DURATION

Bundle extra: The snooze duration of the alarm in minutes.

Used by ACTION_SNOOZE_ALARM. This extra is optional and the value is an Integer that specifies the duration in minutes for which to snooze the alarm.

EXTRA_DAYS

public static final **String** EXTRA_DAYS

Bundle extra: Weekdays for repeating alarm.

Used by ACTION_SET_ALARM.

The value is an ArrayList<Integer>. Each item can be:

- Calendar.SUNDAY,
- Calendar.MONDAY,
- Calendar.TUESDAY,
- Calendar.WEDNESDAY,
- Calendar.THURSDAY,
- Calendar.FRIDAY,
- Calendar.SATURDAY

Constant Value: "android.intent.extra.alarm.DAYS"

EXTRA_HOUR

public static final **String** EXTRA_HOUR

Bundle extra: The hour of the alarm.

Used by ACTION_SET_ALARM.

This extra is optional. If not provided, an implementation should open an activity that allows a user to set an alarm with user provided time.

The value is an Integer and ranges from 0 to 23.

EXTRA_IS_PM

Added in API level 23

public static final **String** EXTRA_IS_PM

Bundle extra: The AM/PM of the alarm.

Used by ACTION_DISMISS_ALARM.

This extra is optional and only used when EXTRA_ALARM_SEARCH_MODE is set to ALARM_SEARCH_MODE_TIME. In this search mode, the EXTRA_IS_PM is used together with EXTRA_HOUR and EXTRA_MINUTES. The implementation should look up the alarm that is most closely matched by these search parameters. If EXTRA_IS_PM is missing, then the AM/PM of the specified EXTRA_HOUR is ambiguous and the implementation should ask for clarification from the user.

The value is a Boolean, where false=AM and true=PM.

EXTRA_LENGTH

public static final **String** EXTRA_LENGTH

Bundle extra: The length of the timer in seconds.

Used by ACTION_SET_TIMER.

This extra is optional. If not provided, an implementation should open an activity that allows a user to set a timer with user provided length.

The value is an Integer and ranges from 1 to 86400 (24 hours).

EXTRA_MESSAGE

public static final **String** EXTRA_MESSAGE

Bundle extra: A custom message for the alarm or timer.

Used by ACTION_SET_ALARM and ACTION_SET_TIMER.

The value is a String.

EXTRA_MINUTES

public static final **String** EXTRA_MINUTES

Bundle extra: The minutes of the alarm.

Used by ACTION_SET_ALARM.

The value is an Integer and ranges from 0 to 59. If not provided, it defaults to 0.

EXTRA_RINGTONE

public static final **String** EXTRA_RINGTONE

Bundle extra: A ringtone to be played with this alarm.

Used by ACTION_SET_ALARM.

This value is a String and can either be set to VALUE_RINGTONE_SILENT or to a content URI of the media to be played. If not specified or the URI doesn't exist, "content://settings/system/alarm_alert will be used.

EXTRA_SKIP_UI

public static final **String** EXTRA_SKIP_UI

Bundle extra: Whether or not to display an activity after performing the action.

Used by ACTION_SET_ALARM and ACTION_SET_TIMER.

If true, the application is asked to bypass any intermediate UI. If false, the application may display intermediate UI like a confirmation dialog or settings.

The value is a Boolean. The default is false.

EXTRA_VIBRATE

public static final **String** EXTRA_VIBRATE

Bundle extra: Whether or not to activate the device vibrator.

Used by ACTION_SET_ALARM.

The value is a Boolean. The default is true.

VALUE_RINGTONE_SILENT

public static final **String** VALUE_RINGTONE_SILENT

Bundle extra value: Indicates no ringtone should be played.

Used by ACTION_SET_ALARM, passed in through EXTRA_RINGTONE.

Code for AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.exno11" >

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme" >
      <activity android:name=".MainActivity" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <receiver android:name=".AlarmReceiver" >
      </receiver>
   </application>

</manifest>
```

Code for Activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">

   <TimePicker
      android:id="@+id/timePicker"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center" />

   <ToggleButton
      android:id="@+id/toggleButton"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:layout_margin="20dp"
      android:checked="false"
      android:onClick="OnToggleClicked" />

</LinearLayout>
```

Java Coding for the Android Application:

Java Coding for Main Activity:

- Click on **app -> java -> com.example.exno11 -> MainActivity.**
- Then delete the code which is there and type the code as given below

```
package com.example.exno11;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.TimePicker;
import android.widget.Toast;
import android.widget.ToggleButton;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity
{
   TimePicker alarmTimePicker;
   PendingIntent pendingIntent;
   AlarmManager alarmManager;

   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      alarmTimePicker = (TimePicker) findViewById(R.id.timePicker);
      alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
   }
   public void OnToggleClicked(View view)
   {
long time;
      if (((ToggleButton) view).isChecked())
      {
         Toast.makeText(MainActivity.this, "ALARM ON", Toast.LENGTH_SHORT).show();
         Calendar calendar = Calendar.getInstance();
         calendar.set(Calendar.HOUR_OF_DAY, alarmTimePicker.getCurrentHour());
         calendar.set(Calendar.MINUTE, alarmTimePicker.getCurrentMinute());
         Intent intent = new Intent(this, AlarmReceiver.class);
         pendingIntent = PendingIntent.getBroadcast(this, 0, intent, 0);

         time=(calendar.getTimeInMillis()-(calendar.getTimeInMillis()%60000));
         if(System.currentTimeMillis()>time)
         {
            if (calendar.AM_PM == 0)
```

```
                time = time + (1000*60*60*12);
            else
                time = time + (1000*60*60*24);
            }
            alarmManager.setRepeating(AlarmManager.RTC_WAKEUP, time, 10000, pendingIntent);
        }
        else
        {
            alarmManager.cancel(pendingIntent);
            Toast.makeText(MainActivity.this, "ALARM OFF", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Java Coding for Alarm Receiver:

- Click on **app -> java -> com.example.exno11 -> AlarmReceiver.**
- Then delete the code which is there and type the code as given below.

Code for AlarmReceiver.java:

```
package com.example.exno11;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.media.Ringtone;

import android.media.RingtoneManager;
import android.net.Uri;
import android.widget.Toast;

public class AlarmReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        Toast.makeText(context, "Alarm! Wake up! Wake up!", Toast.LENGTH_LONG).show();
        Uri alarmUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);
        if (alarmUri == null)
        {
            alarmUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
        }
        Ringtone ringtone = RingtoneManager.getRingtone(context, alarmUri);
        ringtone.play();
    }
}
```

Conclusion: Thus we have studied and implemented alarm clock android

Experiment no 14

Title: Implementation an application using concept Multi-threading
Objective: To develop android application dealing with Multi-threading
Outcome: Students will be able to implement program using Multi-threading
Theory:

A thread is a lightweight sub-process, it going to do background operations without interrupt to ui. This example demonstrate about How to use multiple threads in android.

Android can use multiple CPU cores for multithreading, but the kernel and JVM handle that process, not the developer himself. An internal multithreading design will improve the program's basic performance, but the device upon which it actually runs will determine its speed.

Multithreading in Android

Example
This means when a process is broken, the equivalent number of threads are available. For example, Autocorrect is the process where the software looks for the mistakes in the current word being typed. Endlessly checking for the mistake and providing suggestions at the same time is an example of a Multi-Threaded process.

When an application is launched in Android, it creates the first thread of execution, known as the “main” thread. The main thread is responsible for dispatching events to the appropriate user interface widgets as well as communicating with components from the Android UI toolkit. To keep your application responsive, it is essential to avoid using the main thread to perform any operation that may end up keeping it blocked.

Network operations and database calls, as well as loading of certain components, are common examples of operations that one should avoid in the main thread. When they are called in the main thread, they are called synchronously, which means that the UI will remain completely unresponsive until the operation completes. For this reason, they are usually performed in separate threads, which thereby avoids blocking the UI while they are being performed (i.e., they are performed asynchronously from the UI).

Android provides many ways of creating and managing threads, and many third-party libraries exist that make thread management a lot more pleasant. However, with so many different approaches at hand, choosing the right one can be quite confusing

Threading in Android

In Android, you can categorize all threading components into two basic categories:

1. **Threads that *are* attached to an activity/fragment:** These threads are tied to the lifecycle of the activity/fragment and are terminated as soon as the activity/fragment is destroyed.
2. **Threads that *are not* attached to any activity/fragment:** These threads can continue to run beyond the lifetime of the activity/fragment (if any) from which they were spawned.

Threading Components that Attach to an Activity/Fragment

ASYNCTASK

`AsyncTask` is the most basic Android component for threading. It's simple to use and can be good for basic scenario

Threading Components that Don't Attach to an Activity/Fragment

SERVICE

`Service` is a component that is useful for performing long (or potentially long) operations without any UI. `Service` runs in the main thread of its hosting process; the service does not create its own thread and does not run in a separate process unless you specify otherwise.

INTENTSERVICE

Like `Service`, `IntentService` runs on a separate thread, and stops itself automatically after it completes its work. `IntentService` is usually used for short tasks that don't need to be attached to any UI

Step 1 − Create a new project in Android Studio, go to File ⇒ New Project and fill all required details to create a new project.

Step 2 − Add the following code to res/layout/activity_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:gravity="center_horizontal"
  android:layout_marginTop="100dp"
  tools:context=".MainActivity">
  <EditText
    android:id="@+id/edit_query"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter string" />
  <Button
    android:id="@+id/click"
    android:layout_marginTop="50dp"
    style="@style/Base.TextAppearance.AppCompat.Widget.Button.Borderless.Colored"
    android:layout_width="wrap_content"
    android:background="#c1c1c1"
    android:textColor="#FFF"
    android:layout_height="wrap_content"
    android:text="Button" />
  <TextView
    android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
  <TextView
    android:id="@+id/text1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
```

In the above code, we have taken edittext and textview. When user enter some text into edittext, it going to wait till 5000ms and update both textview's with thread name.

Step 3 − Add the following code to src/MainActivity.java

```
package com.example.myapplication;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
  EditText edit_query;
  TextView textView;

TextView text1;
  boolean twice = false;
  Thread t = null;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    edit_query = findViewById(R.id.edit_query);
    textView = findViewById(R.id.text);
    text1 = findViewById(R.id.text1);
    findViewById(R.id.click).setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        runthread();
        runthread1();
      }
    });
  }
```

```
  private void runthread1() {
     runOnUiThread(new Runnable() {
        @Override
        public void run() {
          try {
            Thread.sleep(5000);
          } catch (InterruptedException e) {
            e.printStackTrace();
          }
          text1.setText("tutorialspoint.com");
        }
     });
  }

private void runthread() {
     twice = true;
        if (twice) {

final String s1 = edit_query.getText().toString();
          t = new Thread(new Runnable() {
          @Override
          public void run() {
            runOnUiThread(new Runnable() {
              @Override
              public void run() {
                textView.setText(t.getName());
                twice = false;
              }
            });
          }
        });
        t.start();
        t.setName(s1);
        t.setPriority(Thread.MAX_PRIORITY);
     }
  }
}
```

Let's try to run your application. I assume you have connected your actual Android Mobile device with your computer. To run the app from android studio, open one of your project's activity files and click Run icon from the toolbar. Select your mobile device as an option and then check your mobile device which will display your default screen −

In the above result, Enter some text in edit text and click on button, It will append the data to thread and take the data from thread as get name method and append to textview in thread 1. Second thread

works in the background and update textview as tuorialspoint.com.

Conclusion: Thus we have studied and implemented the concept of multithreading in android

Experiment no 15

Title: Implementation of program to study and use of SQLite database
Objective: To develop android application dealing with SQLite database
Outcome: Students will be able to implement program using SQLite database
Theory:

SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c

Database - Package

The main package is android.database.sqlite that contains the classes to manage your own databases

Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object.Its syntax is given below

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",MODE_PRIVATE,null);
```

Apart from this , there are other functions available in the database package , that does this job. They are listed below

Sr.No	Method & Description
1	**openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)** This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN_READWRITE OPEN_READONLY
2	**openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)** It is similar to the above method as it also opens the existing database but it does not define any handler to handle the errors of databases
3	**openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)** It not only opens but create the database if it not exists. This method is equivalent to openDatabase method.

4	**openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)** This method is similar to above method but it takes the File object as a path rather then a string. It is equivalent to file.getPath()

Database - Insertion

We can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(Username VARCHAR,Password VARCHAR);");
mydatabase.execSQL("INSERT INTO TutorialsPoint VALUES('admin','admin');");
```

This will insert some values into our table in our database. Another method that also does the same job but take some additional parameter is given below

Sr.No	Method & Description
1	**execSQL(String sql, Object[] bindArgs)** This method not only insert data , but also used to update or modify already existing data in database using bind arguments

Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

```
Cursor resultSet = mydatbase.rawQuery("Select * from TutorialsPoint",null);
resultSet.moveToFirst();
String username = resultSet.getString(0);
String password = resultSet.getString(1);
```

There are other functions available in the Cursor class that allows us to effectively retrieve the data. That includes

Sr.No	Method & Description
1	**getColumnCount()** This method return the total number of columns of the table.
2	**getColumnIndex(String columnName)** This method returns the index number of a column by specifying the name of the column
3	**getColumnName(int columnIndex)** This method returns the name of the column by specifying the index of the column
4	**getColumnNames()** This method returns the array of all the column names of the table.
5	**getCount()** This method returns the total number of rows in the cursor
6	**getPosition()** This method returns the current position of the cursor in the table
7	**isClosed()** This method returns true if the cursor is closed and return false otherwise

Database - Helper class

For managing all the operations related to the database , an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database. Its syntax is given below

```
public class DBHelper extends SQLiteOpenHelper {
  public DBHelper(){
    super(context,DATABASE_NAME,null,1);
  }
  public void onCreate(SQLiteDatabase db) {}
  public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}
}
```

Example

Here is an example demonstrating the use of SQLite Database. It creates a basic contacts application that allows insertion, deletion and modification of contacts.

To experiment with this example, you need to run this on an actual device on which camera is supported.

Steps	Description
1	You will use Android studio to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to get references of all the XML components and populate the contacts on listView.
3	Create new src/DBHelper.java that will manage the database work
4	Create a new Activity as DisplayContact.java that will display the contact on the screen
5	Modify the res/layout/activity_main to add respective XML components
6	Modify the res/layout/activity_display_contact.xml to add respective XML components
7	Modify the res/values/string.xml to add necessary string components
8	Modify the res/menu/display_contact.xml to add necessary menu components
9	Create a new menu as res/menu/mainmenu.xml to add the insert contact option
10	Run the application and choose a running android device and install the application on it and verify the results.

Following is the content of the modified **MainActivity.java**.

```
package com.example.sairamkrishna.myapplication;

import android.content.Context;
import android.content.Intent;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;

import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends ActionBarActivity {
  public final static String EXTRA_MESSAGE = "MESSAGE";
  private ListView obj;
  DBHelper mydb;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mydb = new DBHelper(this);
    ArrayList array_list = mydb.getAllCotacts();
    ArrayAdapter arrayAdapter=new ArrayAdapter(this,android.R.layout.simple_list_item_1, array_list);

    obj = (ListView)findViewById(R.id.listView1);
    obj.setAdapter(arrayAdapter);

obj.setOnItemClickListener(new OnItemClickListener(){
      @Override
      public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,long arg3) {
        // TODO Auto-generated method stub
        int id_To_Search = arg2 + 1;

        Bundle dataBundle = new Bundle();
```

```
        dataBundle.putInt("id", id_To_Search);

        Intent intent = new Intent(getApplicationContext(),DisplayContact.class);

        intent.putExtras(dataBundle);
        startActivity(intent);
      }
    });
  }


  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item){
    super.onOptionsItemSelected(item);

    switch(item.getItemId()) {
      case R.id.item1:Bundle dataBundle = new Bundle();
      dataBundle.putInt("id", 0);

      Intent intent = new Intent(getApplicationContext(),DisplayContact.class);
      intent.putExtras(dataBundle);

      startActivity(intent);
      return true;
      default:
      return super.onOptionsItemSelected(item);
    }
  }

  public boolean onKeyDown(int keycode, KeyEvent event) {
    if (keycode == KeyEvent.KEYCODE_BACK) {
      moveTaskToBack(true);
    }
    return super.onKeyDown(keycode, event);
  }
}
```

Following is the modified content of display contact activity **DisplayContact.java**

```
package com.example.sairamkrishna.myapplication;

import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;
import android.database.Cursor;

import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class DisplayContact extends Activity {
  int from_Where_I_Am_Coming = 0;
  private DBHelper mydb ;

  TextView name ;
  TextView phone;
  TextView  email;
  TextView  street;
  TextView place;
  int id_To_Update = 0;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_display_contact);
    name = (TextView) findViewById(R.id.editTextName);
    phone = (TextView) findViewById(R.id.editTextPhone);
    email = (TextView) findViewById(R.id.editTextStreet);
    street = (TextView) findViewById(R.id.editTextEmail);
    place = (TextView) findViewById(R.id.editTextCity);

    mydb = new DBHelper(this);
```

```
Bundle extras = getIntent().getExtras();
    if(extras !=null) {
      int Value = extras.getInt("id");

      if(Value>0){
        //means this is the view part not the add contact part.
        Cursor rs = mydb.getData(Value);
        id_To_Update = Value;
        rs.moveToFirst();

        String nam = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_NAME));
          String phon = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_PHONE));
          String emai = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_EMAIL));
          String stree = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_STREET));
        String plac = rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_CITY));

        if (!rs.isClosed())  {
          rs.close();
        }
        Button b = (Button)findViewById(R.id.button1);
        b.setVisibility(View.INVISIBLE);

        name.setText((CharSequence)nam);
        name.setFocusable(false);
        name.setClickable(false);

        phone.setText((CharSequence)phon);
        phone.setFocusable(false);
        phone.setClickable(false);

        email.setText((CharSequence)emai);
        email.setFocusable(false);
        email.setClickable(false);

        street.setText((CharSequence)stree);
        street.setFocusable(false);
        street.setClickable(false);

        place.setText((CharSequence)plac);
        place.setFocusable(false);
        place.setClickable(false);
      }
    }
  }
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  // Inflate the menu; this adds items to the action bar if it is present.
  Bundle extras = getIntent().getExtras();

  if(extras !=null) {
    int Value = extras.getInt("id");
    if(Value>0){
      getMenuInflater().inflate(R.menu.display_contact, menu);
    } else{
      getMenuInflater().inflate(R.menu.menu_main menu);
    }
  }
  return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
  super.onOptionsItemSelected(item);
  switch(item.getItemId()) {
    case R.id.Edit_Contact:
    Button b = (Button)findViewById(R.id.button1);
    b.setVisibility(View.VISIBLE);
    name.setEnabled(true);
    name.setFocusableInTouchMode(true);
    name.setClickable(true);

    phone.setEnabled(true);
    phone.setFocusableInTouchMode(true);
    phone.setClickable(true);

    email.setEnabled(true);
    email.setFocusableInTouchMode(true);
    email.setClickable(true);

    street.setEnabled(true);
    street.setFocusableInTouchMode(true);
    street.setClickable(true);

    place.setEnabled(true);
    place.setFocusableInTouchMode(true);
    place.setClickable(true);

    return true;
    case R.id.Delete_Contact:
```

```
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage(R.string.deleteContact)
      .setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
          mydb.deleteContact(id_To_Update);
          Toast.makeText(getApplicationContext(), "Deleted Successfully",
            Toast.LENGTH_SHORT).show();
          Intent intent = new Intent(getApplicationContext(),MainActivity.class);
          startActivity(intent);
        }
    })
    .setNegativeButton(R.string.no, new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int id) {
        // User cancelled the dialog
      }
     });

    AlertDialog d = builder.create();
    d.setTitle("Are you sure");
    d.show();

    return true;
    default:
    return super.onOptionsItemSelected(item);

  }
}

public void run(View view) {
  Bundle extras = getIntent().getExtras();
  if(extras !=null) {
    int Value = extras.getInt("id");
    if(Value>0){
      if(mydb.updateContact(id_To_Update,name.getText().toString(),
        phone.getText().toString(), email.getText().toString(),
                                          street.getText().toString(), place.getText().toString())){
        Toast.makeText(getApplicationContext(), "Updated", Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(getApplicationContext(),MainActivity.class);
        startActivity(intent);
      } else{
        Toast.makeText(getApplicationContext(), "not Updated", Toast.LENGTH_SHORT).show();
      }
    } else{
      if(mydb.insertContact(name.getText().toString(), phone.getText().toString(),
                                          email.getText().toString(), street.getText().toString(),
```

```
                                        place.getText().toString())){
            Toast.makeText(getApplicationContext(), "done",
                                                    Toast.LENGTH_SHORT).show();
        } else{
          Toast.makeText(getApplicationContext(), "not done",
                                                Toast.LENGTH_SHORT).show();
        }
        Intent intent = new Intent(getApplicationContext(),MainActivity.class);
        startActivity(intent);
      }
    }
  }
}
```

Following is the content of Database class **DBHelper.java**

```
package com.example.sairamkrishna.myapplication;

import  java.util.ArrayList;
import  java.util.HashMap;
import java.util.Hashtable;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;

public class DBHelper extends SQLiteOpenHelper {


public static final String DATABASE_NAME = "MyDBName.db";
   public static final String CONTACTS_TABLE_NAME = "contacts";
   public static final String CONTACTS_COLUMN_ID = "id";
   public static final String CONTACTS_COLUMN_NAME = "name";
   public static final String CONTACTS_COLUMN_EMAIL = "email";
   public static final String CONTACTS_COLUMN_STREET = "street";
   public static final String CONTACTS_COLUMN_CITY = "place";
   public static final String CONTACTS_COLUMN_PHONE = "phone";
   private HashMap hp;


 public DBHelper(Context context) {
     super(context, DATABASE_NAME , null, 1);
   }
```

```
  @Override
  public void onCreate(SQLiteDatabase db) {
    // TODO Auto-generated method stub
    db.execSQL(
      "create table contacts " +
      "(id integer primary key, name text,phone text,email text, street text,place text)"
    );
  }

  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // TODO Auto-generated method stub
    db.execSQL("DROP TABLE IF EXISTS contacts");
    onCreate(db);
  }

  public boolean insertContact (String name, String phone, String email, String street,String place) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("name", name);
    contentValues.put("phone", phone);
    contentValues.put("email", email);
    contentValues.put("street", street);
    contentValues.put("place", place);
    db.insert("contacts", null, contentValues);
    return true;
  }

  public Cursor getData(int id) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor res =  db.rawQuery( "select * from contacts where id="+id+"", null );
    return res;
  }

 public int numberOfRows(){
    SQLiteDatabase db = this.getReadableDatabase();
    int numRows = (int) DatabaseUtils.queryNumEntries(db, CONTACTS_TABLE_NAME);
    return numRows;
  }

  public boolean updateContact (Integer id, String name, String phone, String email, String street,String place)
{
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
```

```
    contentValues.put("name", name);
    contentValues.put("phone", phone);
    contentValues.put("email", email);
    contentValues.put("street", street);
    contentValues.put("place", place);
    db.update("contacts", contentValues, "id = ? ", new String[] { Integer.toString(id) } );
    return true;
  }

  public Integer deleteContact (Integer id) {
    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete("contacts",
    "id = ? ",
    new String[] { Integer.toString(id) });
  }

  public ArrayList<String> getAllCotacts() {
    ArrayList<String> array_list = new ArrayList<String>();

    //hp = new HashMap();
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor res =  db.rawQuery( "select * from contacts", null );
    res.moveToFirst();

    while(res.isAfterLast() == false){
      array_list.add(res.getString(res.getColumnIndex(CONTACTS_COLUMN_NAME)));
      res.moveToNext();
    }
    return array_list;
  }
}
```

Following is the content of the **res/layout/activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:id="@+id/textView"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:textSize="30dp"
    android:text="Data Base" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Tutorials Point"
    android:id="@+id/textView2"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true"
    android:textSize="35dp"
    android:textColor="#ff16ff01" />

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:layout_below="@+id/textView2"
    android:layout_centerHorizontal="true"
    android:src="@drawable/logo"/


<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/scrollView"
    android:layout_below="@+id/imageView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true">

    <ListView
      android:id="@+id/listView1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true" >
    </ListView>

  </ScrollView>
```

```
</RelativeLayout>
```

Following is the content of the **res/layout/activity_display_contact.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/scrollView1"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  tools:context=".DisplayContact" >

  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="370dp"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <EditText
      android:id="@+id/editTextName"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_marginTop="5dp"
      android:layout_marginLeft="82dp"
      android:ems="10"
      android:inputType="text" >
    </EditText>

<EditText
      android:id="@+id/editTextEmail"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/editTextStreet"
      android:layout_below="@+id/editTextStreet"
      android:layout_marginTop="22dp"
      android:ems="10"
      android:inputType="textEmailAddress" />

    <TextView
```

```
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/editTextName"
        android:layout_alignParentLeft="true"
        android:text="@string/name"
        android:textAppearance="?android:attr/textAppearanceMedium" /


<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editTextCity"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="28dp"
        android:onClick="run"
        android:text="@string/save" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/editTextEmail"
        android:layout_alignLeft="@+id/textView1"
        android:text="@string/email"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/editTextPhone"
        android:layout_alignLeft="@+id/textView1"
        android:text="@string/phone"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/editTextEmail"
        android:layout_alignLeft="@+id/textView5"
        android:text="@string/street"
        android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
    <EditText
      android:id="@+id/editTextCity"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignRight="@+id/editTextName"
      android:layout_below="@+id/editTextEmail"
      android:layout_marginTop="30dp"
      android:ems="10"
      android:inputType="text" />

    <TextView
      android:id="@+id/textView3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/editTextCity"
      android:layout_alignBottom="@+id/editTextCity"
      android:layout_alignParentLeft="true"
      android:layout_toLeftOf="@+id/editTextEmail"
      android:text="@string/country"
      android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
      android:id="@+id/editTextStreet"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/editTextName"
      android:layout_below="@+id/editTextPhone"
      android:ems="10"
      android:inputType="text" >

      <requestFocus />
    </EditText>

    <EditText
      android:id="@+id/editTextPhone"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/editTextStreet"
      android:layout_below="@+id/editTextName"
      android:ems="10"
      android:inputType="phone|text" />

  </RelativeLayout>
</ScrollView>
```

Following is the content of the **res/value/string.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">Address Book</string>
   <string name="action_settings">Settings</string>
   <string name="hello_world">Hello world!</string>
   <string name="Add_New">Add New</string>
   <string name="edit">Edit Contact</string>
   <string name="delete">Delete Contact</string>
   <string name="title_activity_display_contact">DisplayContact</string>
   <string name="name">Name</string>
   <string name="phone">Phone</string>
   <string name="email">Email</string>
   <string name="street">Street</string>
   <string name="country">City/State/Zip</string>
   <string name="save">Save Contact</string>
   <string name="deleteContact">Are you sure, you want to delete it.</string>
   <string name="yes">Yes</string>
   <string name="no">No</string>
</resources>
```

Following is the content of the **res/menu/main_menu.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

   <item android:id="@+id/item1"
      android:icon="@drawable/add"
      android:title="@string/Add_New" >
   </item>

</menu>
```

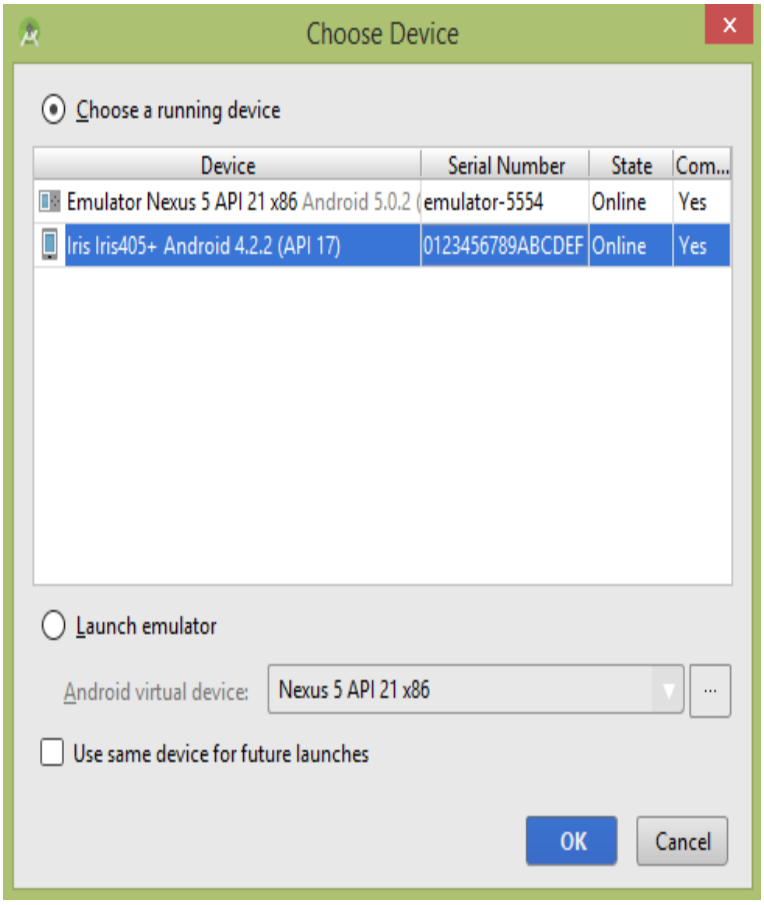
Following is the content of the **res/menu/display_contact.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
   <item
      android:id="@+id/Edit_Contact"
      android:orderInCategory="100"
      android:title="@string/edit"/>

   <item
      android:id="@+id/Delete_Contact"
```

```
    android:orderInCategory="100"
    android:title="@string/delete"/>

</menu>
```

This is the defualt **AndroidManifest.xml** of this project

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.sairamkrishna.myapplication" >

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
      android:name=".MainActivity"
      android:label="@string/app_name" >

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

    </activity>

    <activity android:name=".DisplayContact"/>

  </application>
</manifest>
```

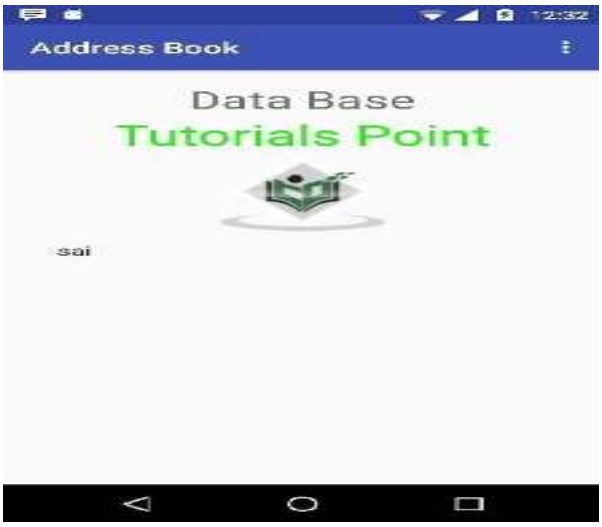


Select your mobile device as an option and then check your mobile device which will display following screen

Now open your optional menu, it will show as below image: **Optional menu appears different places on different versions**

Click on the add button of the menu screen to add a new contact. It will display the following screen −

It will display the following fields. Please enter the required information and click on save contact. It will bring you back to main screen.



Now our contact sai has been added. In order to see that where is your database is created. Open your android studio, connect your mobile. Go **tools/android/android device monitor**. Now browse the file explorer tab. Now browse this folder **/data/data/<your.package.name>/databases<database-name>**

Conclusion:

Thus we have studied and implemented android application using SQLite Database

Experiment no 16

Title: Study of publishing app to the Android Market

Objective: To publish android application in app market or Playstore

Outcome: Students will be able to deploy the android application on Playstore or android app market

Theory:

Step 1: Create a Google Developer account

This is something you can do at the beginning of the app development process. Without registering a Google Developer Account, you can't publish your app on the Play Market.

You can use any of your current Google accounts or create another one to sign up for a Google Developer Account. It doesn't matter whether it's a private or corporate account. You may easily transfer your app to another one in the future.

The creation process includes signing the Google Play Developer distribution agreement, adding some personal information, and paying a one-time registration fee of $25. There is nothing complicated. Just follow the instructions.

Google Play Developer distribution agreement

Usually, it takes no more than two days to get approval from Google. Don't worry if you forget to add some information. You can edit your account later.

Step 2: Add a Merchant Account

If you plan to sell paid apps or in-app purchases, you have to create a Google Merchant Account. There you can manage app sales and your monthly payouts, as well as analyze sales reports.

Once you finish creating the Merchant profile, the developer account gets automatically linked to it.

Step 3: -Prepare the Documents

Paperwork always requires much effort, especially when it comes to any kind of legal documents. Based on our experience, we highly recommend starting to prepare the End User License Agreement (EULA) and Privacy Policy in advance.

You can take the documents from similar apps as references and create your own based on them, or ask a lawyer to make everything from scratch.

EULA is an agreement between you as an owner and a user of your product. In brief, it contains:

- What the users can do with the app, and what they aren't allowed to do
- Licensing fees
- Intellectual property information, etc.

Terms of Use or Terms and Conditions explain what services you offer the users and how you expect them to behave in return. Though Google doesn't demand Terms of Use, it's better to publish them. You can create one document, adding there Privacy Policy and Terms of Use chapters.

Pay special attention to include in the Privacy Policy the following information:

- A complete list of personal data that is collected, processed and used through the app
- Technical information that is collected about the device and the installed OS
- Functional features of the app, its paid and free functionality
- Place of registration of the company and/or location of the copyright holder of the application
- The chosen legal system and legislation that will be applied in resolving disputes and regulating legal relations
- The terms of subscription
- Citizenship (residence) of the overwhelming majority of application users
- Age criteria, the presence of specific content

Step 4: Study Google Developer Policies

We guess you already made up your product concept. Now it's time to make sure that every feature you will implement in the app is aligned with the Google Developer Policies. These documents explain how apps need to be developed, updated, and promoted to support the store's high-quality standards.

If Google decides that your product violates some policy chapters, it may be rejected, blocked, or even deleted from the Play Store. Besides, numerous and repetitive violations may lead to the developer account termination. So study all the available information carefully about:

- Restricted content definition
- Store listing and promotion
- Impersonation and intellectual property
- Rules for monetization and ads
- Privacy, security and deception regulation
- Spam and minimum functionality

Google is constantly working on its policies, and it's important to monitor the changes and stay up to date even after your app is released.

Step 5: Technical Requirements

You went through the development process, endless testing, and bug fixing, and finally, the "X-day" comes. Before moving on to the upload process, you need to check the following things:

- **Unique Bundle ID**

The package name should be suitable over the life of your application. You cannot change it after the distribution. You can set the package name in the application's manifest file.

- **Signed App Release With a Signing Certificate**

Every application should be digitally signed with a developer's certificate. The certificate is used to identify the author of an app and can't be generated again.

- **The App Size**

Google set the limit size of the uploaded file: 100MB for Android 2.3 and higher (API level 9-10, 14 and higher) and 50MB for lower Android versions.

If your app exceeds this limit, you can always switch to APK Expansion Files.

- **The File Format**

Two possible release formats are accepted by Google: app bundle and .apk. However, .aab is the preferred one. To use this format, you need to enroll in app signing by Google Play.

You may learn more about app file technical requirements in the Developer Documents, Prepare for the release guide.

Step 6: Creating the App on the Google Console

Now you have the file that is ready for uploading. It's time to get to the fun part. Let's create a new app in your Developer Account:

- Reach to All applications tab in the menu
- Now select Create Application
- Choose the app's default language from the drop-down menu
- Add a brief app description (you can change it later)

- Tap on Create

After this, you will be taken to the store entry page, where we will add the complete data about the app.

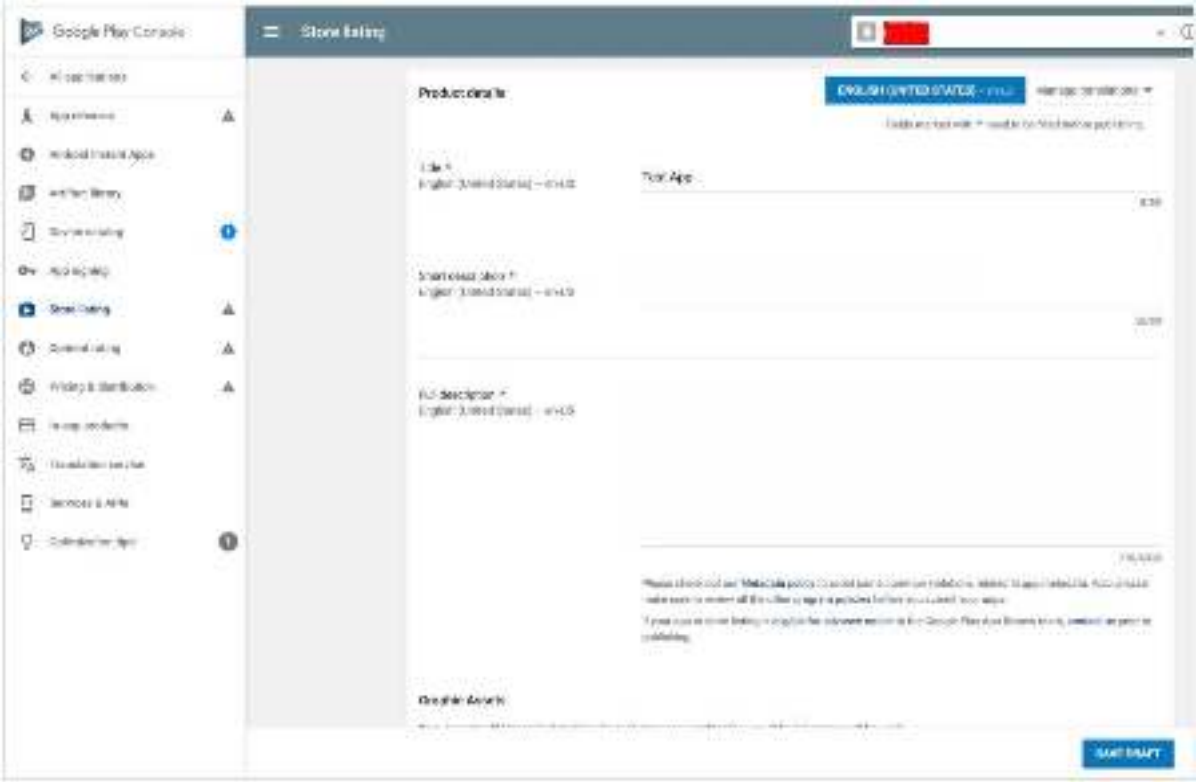
Step 7: Store Listing

First, let's prepare the Store listing. It contains the most important information useful for app store optimization (ASO) and gives the users more details about your app before downloading. The mandatory sections are marked with *.

You may need some designer and copywriter efforts, so it's better to start preparing the following materials in advance.

- **Product description**

It contains a title of your app (up to 50 symbols), a brief description (up to 80 symbols), and a full description (up to 4000 symbols). Control yourself and do not overdo the keywords.



Store listing

- **Screenshots**

You may add from 2 to 8 screenshots. Choose the ones that show the app functionality and value the most.

The requirements are the following:

- JPEG or 24-bit PNG (no alpha)

- from 320px to 3840 px

the ratio of the long side to the short side should not be more than 2:1

Store listing - Product details

- **Icon**

The requirements are the following:

- 512px by 512px

- 32-bit PNG (with alpha)

- **Maximum file size: 1024KB**

- Feature graphic

It is an optional marketing tool displayed in various places on the Play Store, for example, on the homepage.

The requirements are the following:

- JPEG or 24-bit PNG (no alpha)

- 1024px x 500px

- **Promo video**

If you have any promo video, you may add a link to your YouTube channel. This video will be shown before the screenshots on the app's page.

- **Tags**

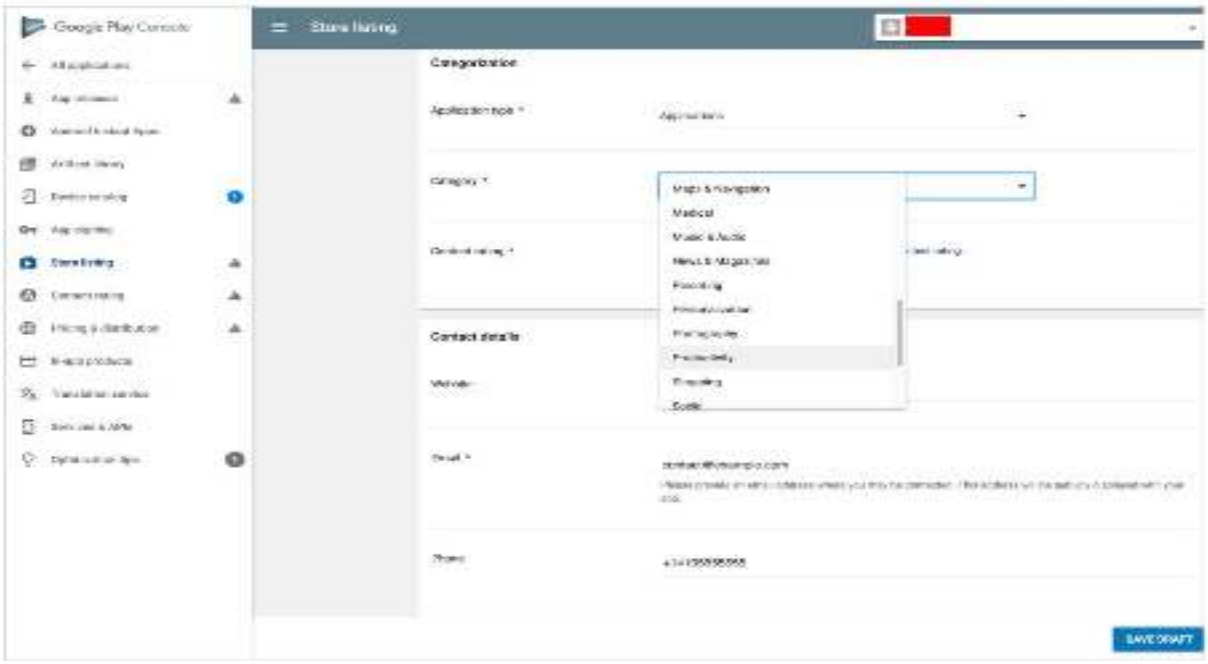
You may choose from the list the most relevant to your app keywords for better ASO. There is no possibility to add any custom tags.

- **Localization**

If your app supports several languages, mention all of them and add translations of your app's information. It's highly recommended to include localized screenshots and images.

- **Application type and categorization**

First, through the drop-down menu, select the application type: game or app. Then pick the category that your app fits into. You can also add a section to rate your content after uploading APK to Google Play.



App categories on Google Play

- **Contact details**

Here you should provide the support service contacts. By filling the website URL, email, and phone, you make it easier for the users to contact you if necessary.

- **Privacy Policy**

Google requires you to add a link to the Privacy Policy that we discussed above.

While editing the Store Listing, you can take a break at any moment, click Save Draft, and complete this stage later.

Step 8: Content Rating

In order not to be marked as an Unrated App (that may lead to app removal), pass a rating questionnaire. You can easily find this section on the left-side menu.

The information provided in the questionnaire must be accurate. Any misrepresentation of your app's content might lead to suspension or removal of the Play Store account.

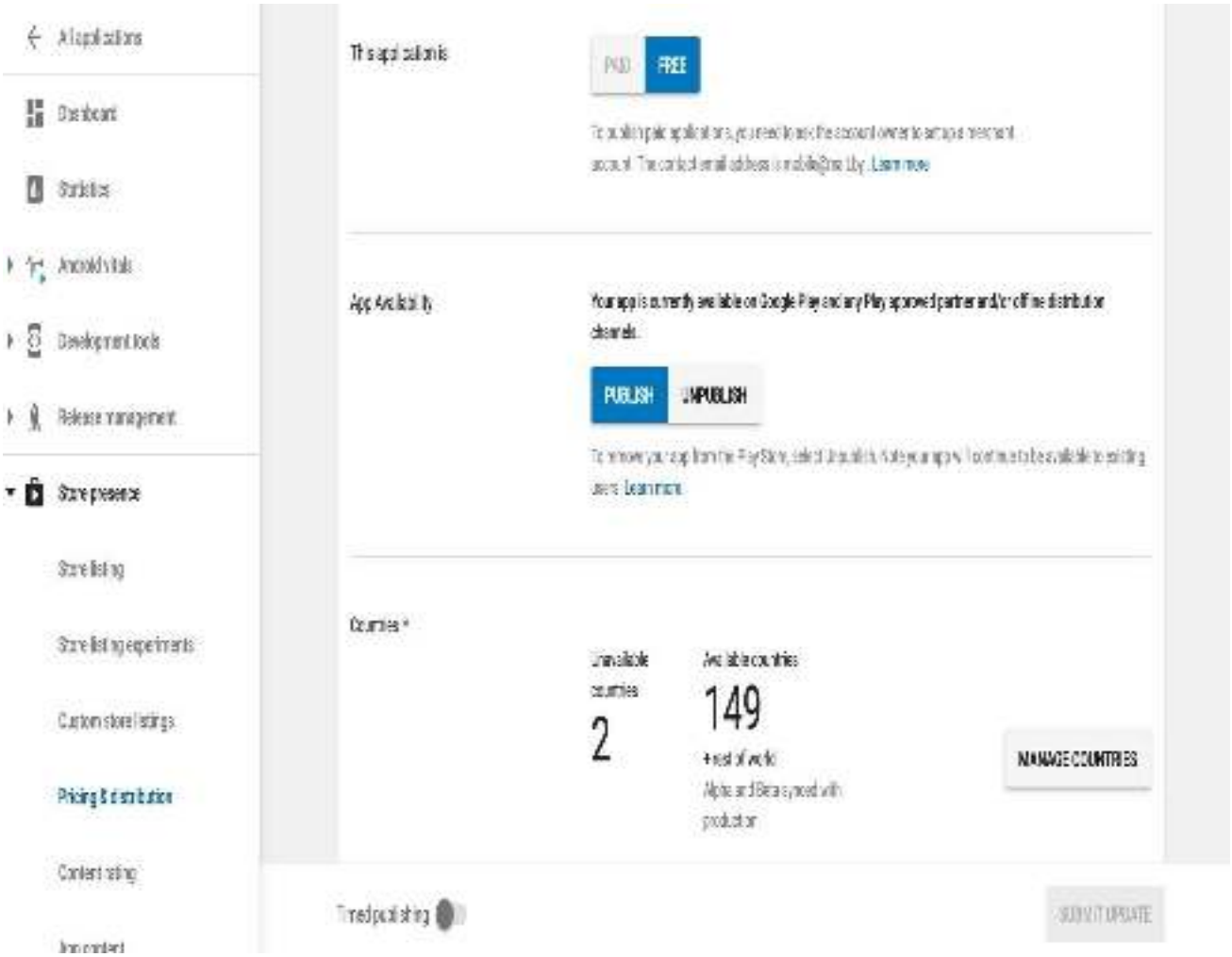
- Click on Save Questionnaire once you complete the survey
- Click on Calculate Rating
- In the end, click on Apply Rating to confirm the rating and move forward with the pricing & distribution plan

Step 9: Pricing the Application

In the Pricing and distribution section, you need to fill the following information:

- Whether your app is free or paid
- Where the app will be available (just choose the countries from the list)
- Whether your app will be available only on the specific devices
- Whether the app has sensitive content and is not suitable for children under the age of 13
- Whether your app contains ads

Remember that you can change your paid app to a free one later, but you cannot do the vice versa. If you decide later that you want to distribute it for money, you'll have to create another app.



Step 10: Upload APK and Send for Review

Finally, you are ready to upload your app file. That's the most exciting moment ever.

Let's go to the App Releases section on the left panel. Here you will find three options for publishing the app: Production, Beta and Alpha tracks.

We highly recommend starting with Alpha or Beta versions. In this case, after passing the review process, your app will not be available to everyone on the Play Store.

The Alpha version assumes closed testing and is available only to those who you invite as testers. The Beta version means that anyone can join your testing program and send feedback to you.

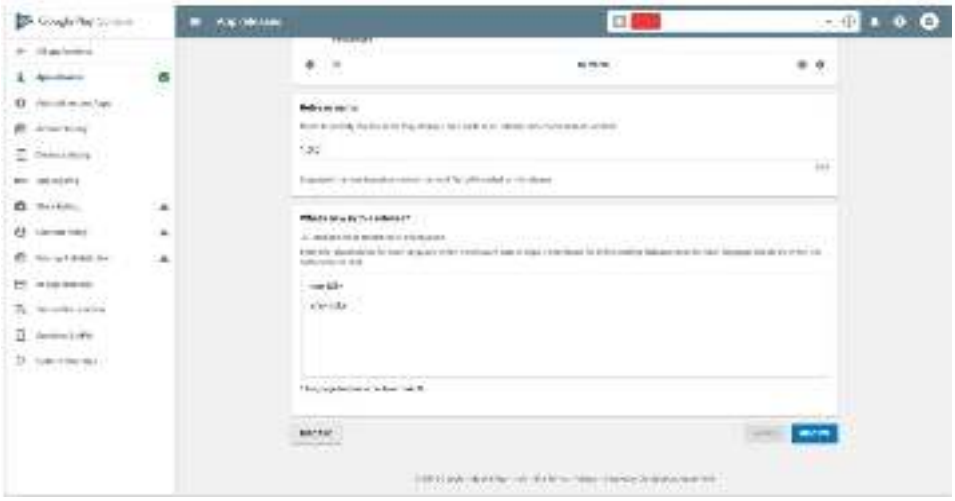
Pre-release testing allows you to gather people's opinions, test your app in a broader audience, and fix issues before making the app public.

Note that if you decide later to change the Alpha or Beta version to Production type, it will take time to go through another review round.

Once you choose the type of release, follow the steps:

- Choose Manage (Production/Beta/Alpha)
- Click on Edit Release
- Upload an APK or app bundle

The release name will be added automatically. For the first time, you may delete the text from the What's new in this release field.



- Click on Review to confirm the changes and send your app to the review by pressing Start rollout to production.

Don’t worry that you may forget to add some information. All the way, Google will show you the instructions and tips. Actually, you won’t manage to send the app to the review if something important is missed.

Remember that with the very first version, there is no opportunity to set manual publishing. The app will be released right after it passes the review. Usually, it takes up to 2 days. Google says the review process could take up to 7 days or even longer.

Once the app is reviewed, you’ll receive a notification on Google Console Dashboard.

Conclusion:

Thus we have studied requirements for deploying android application in app market