

Geetanjali Institute of Technical Studies

(Approved by AICTE, New Delhi and Affiliated to Rajasthan Technical University Kota (Raj.))

DABOK, UDAIPUR, RAJASTHAN 313022

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B. Tech - VI SEMESTER



ACADEMIC YEAR – 2021-2022

MOBILE APPLICATION DEVELOPMENT LAB 6CS4-24

Submitted by:
(Name of the Student)
(Roll No.)
(Section-A)

INDEX

S. N.	Experiment Name	Date of Conduct	Date of Submission	Signature
1	To study Android and Android Studio installation. Create “Hello World” application.	1/4/2022	22/4/2022	
2	To understand Activity, Intent. Create sample application with login module. (Check username and password)	22/4/2022	26/4/2022	
3	Design simple GUI application with activity and intents e.g. calculator.	26/4/2022	27/4/2022	
4	Develop an application that makes use of RSS Feed.	27/4/2022	29/4/2022	
5	Write an application that draws basic graphical primitives on the screen.	29/4/2022	5/5/2022	
6	Create an android app for database creation using SQLite database.	5/5/2022	7/5/2022	
7	Develop a native application that uses GPS location information.	7/5/2022	9/5/2022	
8	Implement an application that writes data to the SD card.	9/5/2022	10/5/2022	
9	Design a gaming application.	10/5/2022	11/5/2022	
10	Create an application to handle images and video according to size.	11/5/2022	12/5/2022	

COURSE SYLLABUS



RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

III Year-VI Semester: B.Tech. Computer Science and Engineering

6CS4-24: Mobile Application Development Lab

Credit: 1.5

Max. Marks: 75(IA:45, ETE:30)

OL+OT+3P

End Term Exam: 2 Hours

SN	List of Experiments
1	To study Android Studio and android studio installation. Create "Hello World" application.
2	To understand Activity, Intent, Create sample application with login module.(Check username and password).
3	Design simple GUI application with activity and intents e.g. calculator.
4	Develop an application that makes use of RSS Feed.
5	Write an application that draws basic graphical primitives on the screen
6	Create an android app for database creation using SQLite Database.
7	Develop a native application that uses GPS location information
8	Implement an application that writes data to the SD card.
9	Design a gaming application
10	Create an application to handle images and videos according to size.

Office of Dean Academic Affairs
Rajasthan Technical University, Kota

Syllabus of 3rd Year B. Tech. (CS) for students admitted in Session 2017-18 onwards. Page 14

6CS4-24: Mobile Application Development Lab

Credit: 1.5

Max. Marks: 75 (IA:45, ETE:30)

0L+0T+3P End Term Exam: 2 Hours

Sr. No.	LIST OF EXPERIMENT	CO MAPPING
1.	To study Android and Android Studio installation. Create “Hello World” application.	CO1
2.	To understand Activity, Intent. Create sample application with login module. (Check username and password)	CO1, CO2
3.	Design simple GUI application with activity and intents e.g. calculator.	CO2
4.	Develop an application that makes use of RSS Feed.	CO3
5.	Write an application that draws basic graphical primitives on the screen.	CO2, CO3
6.	Create an android app for database creation using SQLite database.	CO4
7.	Develop a native application that uses GPS location information.	CO5
8.	Implement an application that writes data to the SD card.	CO4, CO5
9.	Design a gaming application.	CO5
10.	Create an application to handle images and video according to size.	CO4

Prerequisites:

- Students are expected to have knowledge of core Java.
- Students are expected to have basic knowledge of HTML.

Software requirements:

- Android Studio

EXPERIMENT NO. 1

AIM: Experiment-1: To study Android studio and android studio installation. Create "Hello World" application.

Solution: Android is an open source operating system which is based on Linux. This operating system is currently available for smart mobile devices and tablet computers. It was initially developed by Google and Open Handset Alliance in 2008. Well, here, we will be providing you the tutorials for basic Android programming and its advanced concepts.

What Are The Required Prerequisites?

There are no particular required prerequisites to learn Android programming. But, you will definitely get an upper hand if you are already familiar with the Java programming. If you are thinking “why Java programming?” then you must know that Android programming is based on Java programming.

Android offers a brought together way to deal with application advancement for cell phones which implies designers require produce for Android, and their applications ought to have the capacity to keep running on various gadgets fueled by Android.

The first beta variant of the Android Software Development Kit (SDK) was launched by Google in 2007 where as the primary business version, Android 1.0, was launched in September 2008. On June 27, 2012, at the Google I/O gathering, Google declared the Android versions, 4.1 Jelly Bean.

Android Applications

Android applications are generally created in the Java programming language utilizing the Android Software Development Kit.

Once created, Android applications can be bundled effortlessly and sold out either through an application store like Google Play Store, Opera Mobile Store, Amazon App store etc..

Android powers a huge number of cell phones in more than 190 nations around the globe. It's the biggest introduced base of any versatile stage and developing quicker. Consistently more than 1 million new Android gadgets are initiated around the world.

This instructional exercise has been composed with an intent to show you how to create and bundle Android application. We will begin with condition setup for Android application programming and after that penetrate down to investigate different parts of Android applications.

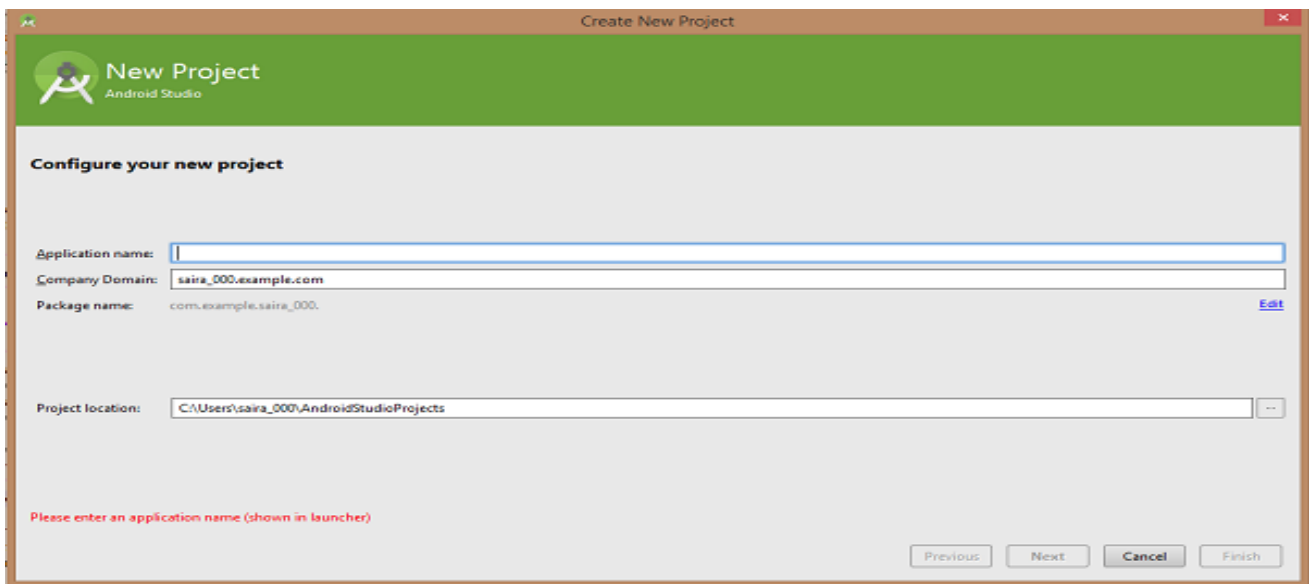
How to create an Android application?

Follow the step-by-step guide of ours to understand how an Android application is created.

Step 1: We are using Android Studio to create the Android application. Once you launch this software, you will get a window which will appear on the following screen.

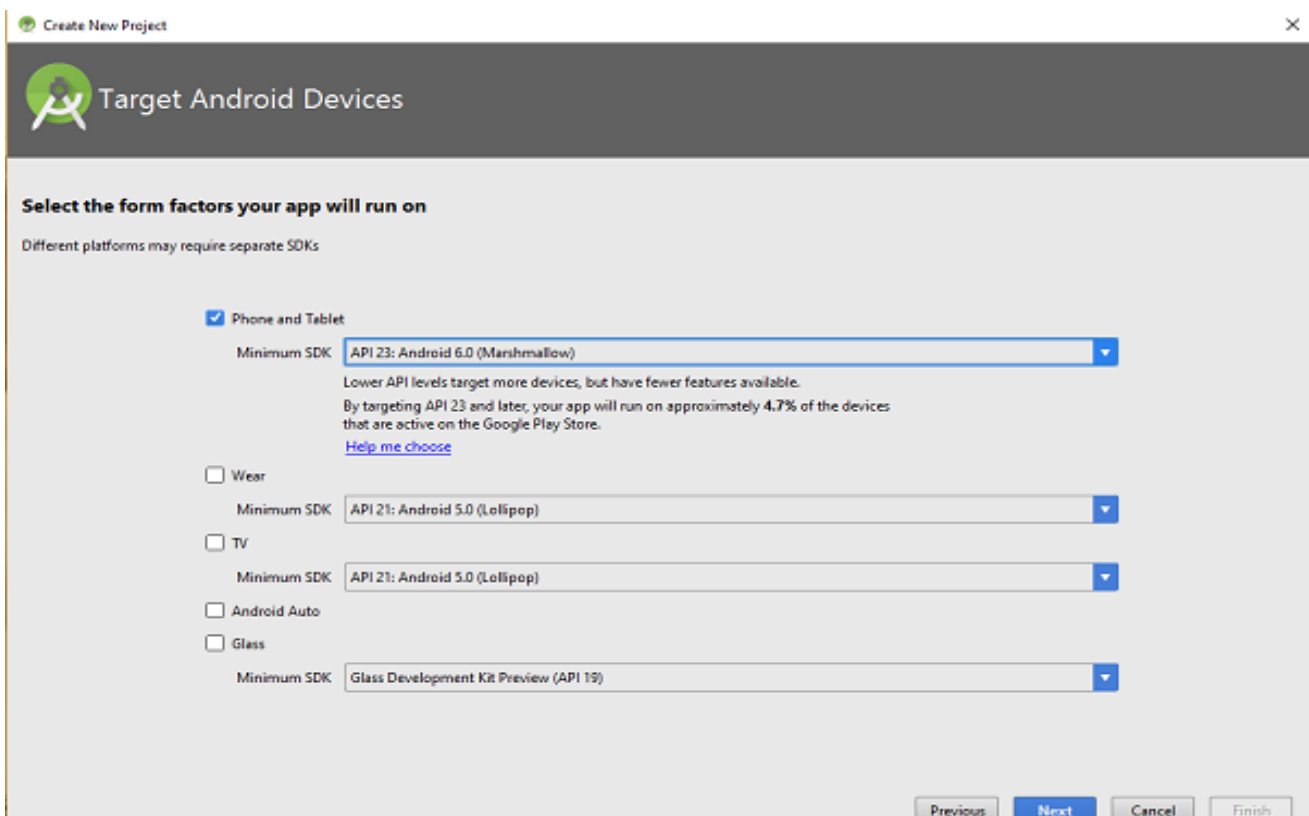


Step 2: Here, click on *Start a new Android Studio project*. After this, you will be redirected to another window titled as *Create New Project*. This window will ask you to enter a few details which include the Application name, Company domain, Package name, and Project location.



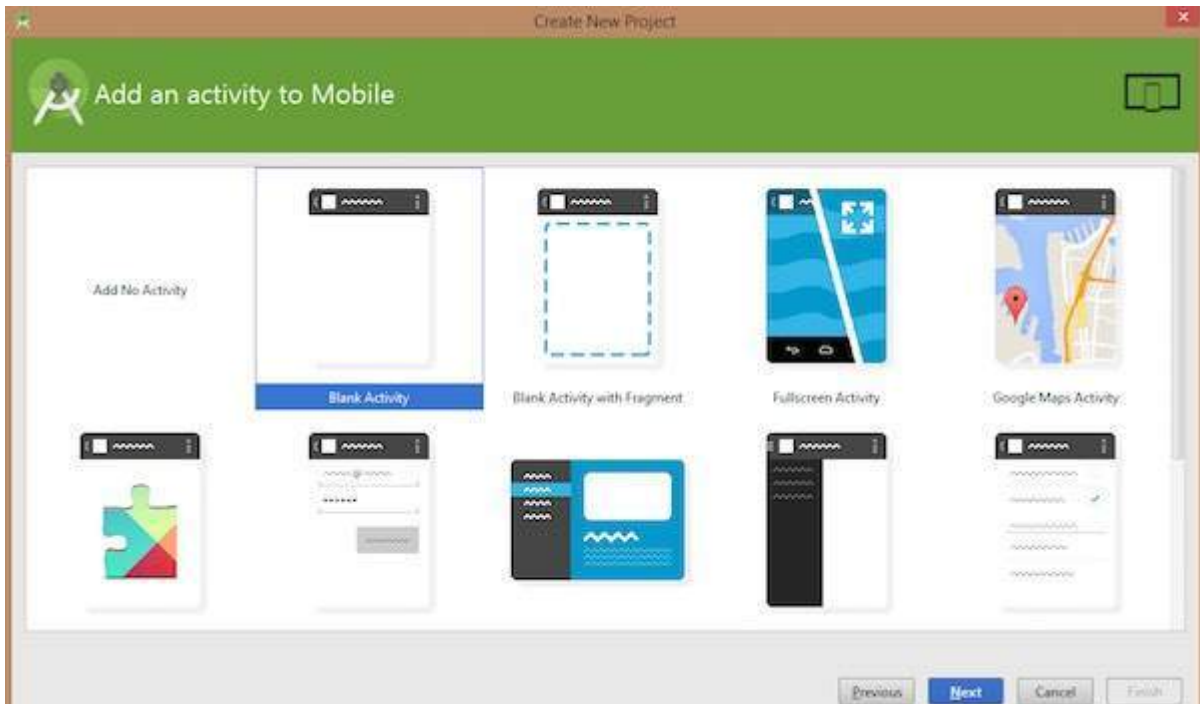
The screenshot shows the 'Create New Project' dialog in Android Studio. The title bar says 'Create New Project'. The main header is 'New Project' with the Android Studio logo. Below it, the text 'Configure your new project' is displayed. There are four input fields: 'Application name' (empty), 'Company Domain' (saira_000.example.com), 'Package name' (com.example.saira_000), and 'Project location' (C:\Users\saira_000\AndroidStudioProjects). A red error message at the bottom left says 'Please enter an application name (shown in launcher)'. At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

Step 3: After entering the required details, hit the *Next* button. On this window, you will be called to select a few form factors. This new window will appear like the given screen. Well, we are using API23: Android 6.0 (Marshmallow) which we have declared in the *Minimum SDK* of *Phone and Tablet*.

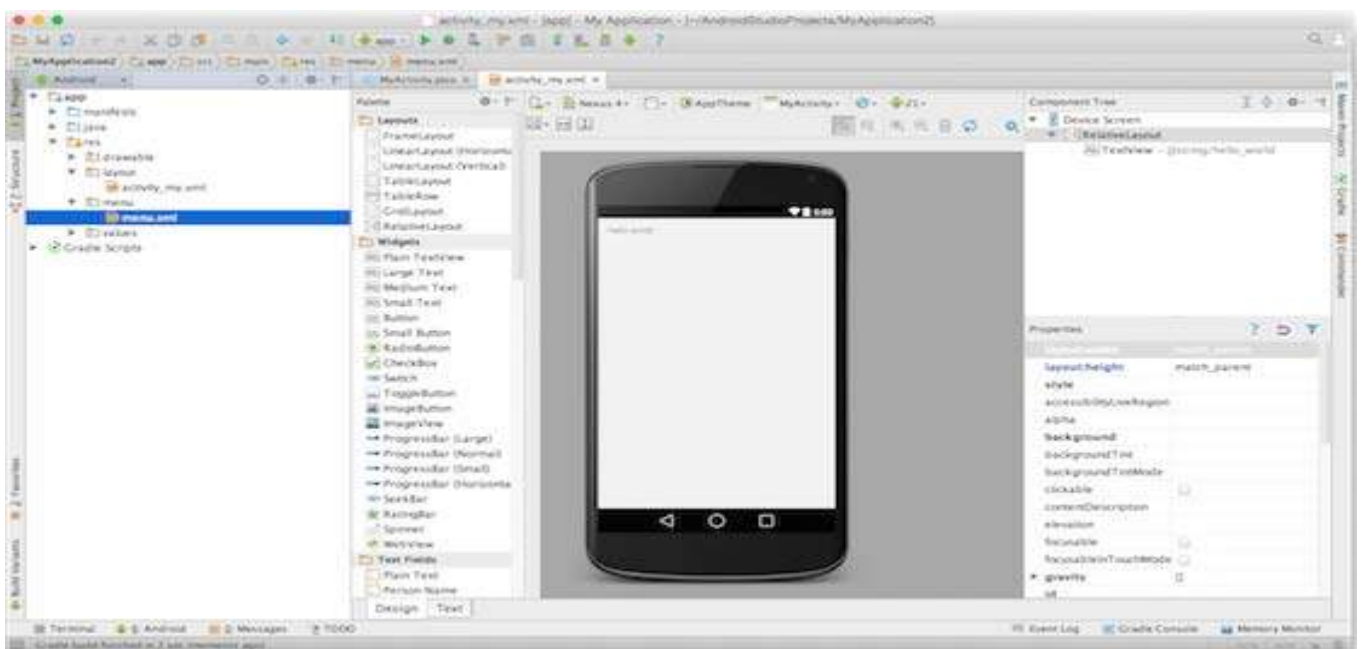


The screenshot shows the 'Target Android Devices' dialog in Android Studio. The title bar says 'Create New Project'. The main header is 'Target Android Devices' with the Android Studio logo. Below it, the text 'Select the form factors your app will run on' is displayed. A note says 'Different platforms may require separate SDKs'. There are five options: 'Phone and Tablet' (checked), 'Wear', 'TV', 'Android Auto', and 'Glass'. Each option has a 'Minimum SDK' dropdown menu. For 'Phone and Tablet', the dropdown is set to 'API 23: Android 6.0 (Marshmallow)'. Below this dropdown, there is a note: 'Lower API levels target more devices, but have fewer features available. By targeting API 23 and later, your app will run on approximately 4.7% of the devices that are active on the Google Play Store.' and a link 'Help me choose'. At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

Step 4: This level will ask you to add activities to mobile. On this window, you will get a number of default layouts for your Android application.

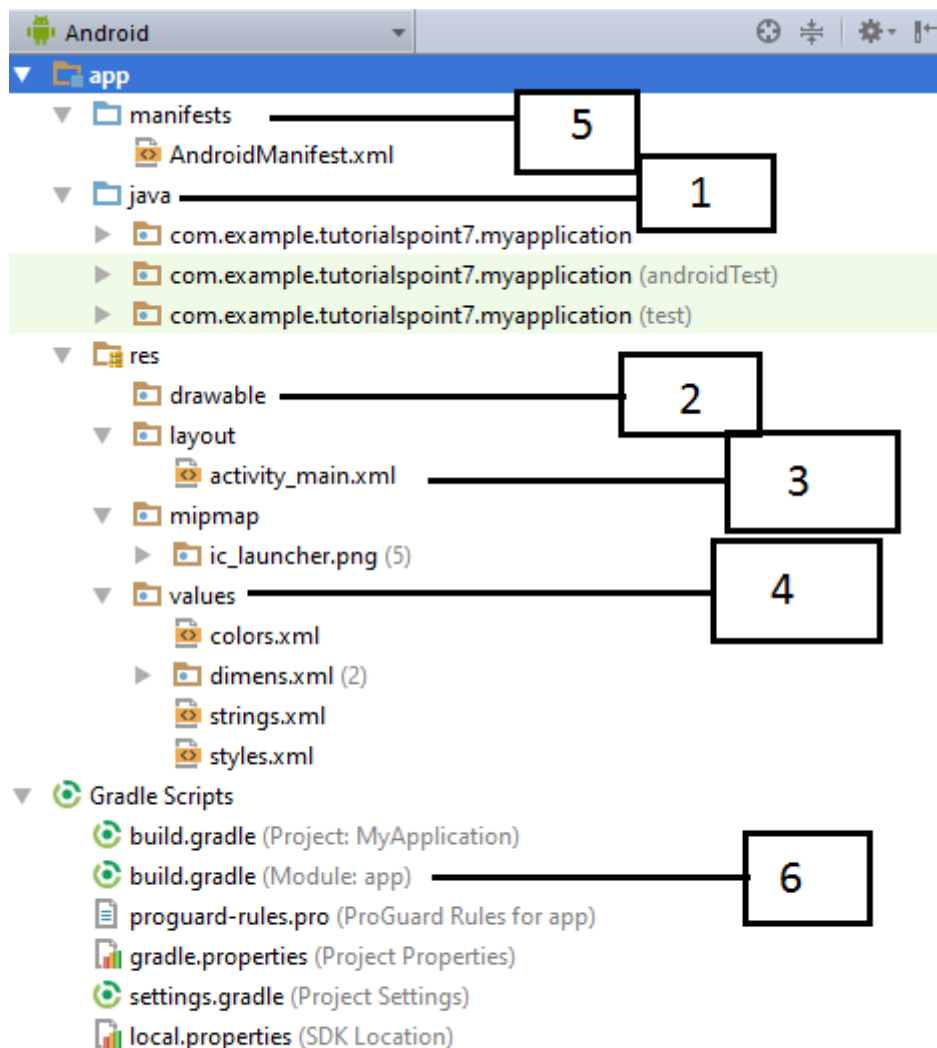


Step 5: Finally, you have arrived at the application development tool where you can write the code for your first application.



Closer look at Android Application

Before we go straight to the application coding, you must understand that there are various files and directories used for certain purposes. Let's take a closer look at all of these files and directories of the Android project.



S.No.	Files and Directories	Description
1	<i>java</i>	<ul style="list-style-type: none"> * Contains .java source files used in Android projects * Contains MainActivity.java file which automatically runs, once your application is launched. This Java file includes an Activity class in it
2	<i>res/drawable</i>	<ul style="list-style-type: none"> * Contains drawable objects especially used for high-density screen
3	<i>res/layout</i>	<ul style="list-style-type: none"> * Files used to help you to define an application's user interface
4	<i>res/values</i>	<ul style="list-style-type: none"> * Contains numerous XML files * These files keep a collection of strings and color definitions
5	<i>manifests/AndroidManifest.xml</i>	<ul style="list-style-type: none"> * Describes basic features of an application * Contains definition of each and every component of an application
6	<i>Gradle Scripts/build.gradle</i>	<ul style="list-style-type: none"> * An auto-generated file * Contains compileSdkVersion, buildToolsVersion,

applicationId, minSdkVersion, targetSdkVersion,
versionCode and versionName

Here, you will get to know a bit more about the major application files.

Main Activity File – MainActivity.java

As you can already see *MainActivity.java* is a Java file. This is the real application file and helps in running your application. This file first gets converted into Dalvik executable file after which it helps your application to run. Check the given default code for Hello World! app.

```
package com.example.helloworld;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Here, *onCreate()* method is one of those methods which is used when you want to load an activity. Other than that, *R.layout.activity_main* is used to refer to the *activity_main.xml* file from the *res/layout* directory.

Manifest File – manifest.xml

All the components used in an application should be defined in *manifest.xml*. This file is located at the root of the application project directory. *manifest.xml* is an important file as it acts as an interface between your operating system and the developed Android application. So, in any case, if you forget to declare your application's component in this file then the operating system won't be able to recognize them. The given code will help you to understand how a *manifest.xml* file usually looks like.

```
<?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>
    </application>
</manifest>
```

Here, all the important application's components are enclosed between *<application>* and *</application>* tags. Other than that, *android:icon* is an attribute which is used to point the application icon present in *res/drawable-hdpi* directory. In this drawable directory, you will find an image under the name *ic_launcher.png* which is also used in the application. And the *<activity>* and *</activity>* tags enclose the

specification of an activity. The *action* performed by the intent filter is stored in *android.intent.action.MAIN*. This file points out an activity as the entry point for your application. While *android.intent.action.LAUNCHER* is the *category* for the intent filter. The inclusion of this file shows that your application is available to be launched with the help of device's launcher icon.

To refer to strings.xml file, we have used `@string`. While `@string/app_name` is already defined in the strings.xml file under the name of `app_name`. The definition of the `@string/app_name` is "HelloWorld". Similarly, other strings are also declared in the strings.xml file.

To specify various components, you need to use the following tags in your manifest file. Take a look at the following list.

- * **<activity> and </activity>** : to denote activities
- * **<service>** : to denote services
- * **<receiver> and </receiver>** : to denote broadcast receivers
- * **<provider>** : to denote content providers

Strings File – strings.xml

The strings file is denoted by *strings.xml* which is located in the *res/values* directory. All the text which are being used by your application is already declared in this file. From the names of the buttons to the labels, everything is being defined in the strings.xml file. In totality, this file is having the responsibility of handling all the textual content of the application. Well, the strings.xml file usually contains the code which looks like the following.

```
<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>
```

[post_middile_section_ad]

Layout File – activity_main.xml

To represent the layout file, the Android Studio is using *activity_main.xml*. You can locate this file in the *res/layout* directory. This file will be used when you want to build a user interface for your application. The default layout for the "Hello World!" application will appear as the given code.

```
<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>
```

The above code is a basic structure for the *RelativeLayout*. The Android control, *TextView* own various attributes which include *android:layout_width*, *android:layout_height*, *android:layout_centerHorizontal*,

android:layout_centerVertical, and a few others. All of these attributes help in building the user interface of your application. In the above lines of code, @string is used to denote strings.xml which resides in the res/values directory. And so, the @string/hello_world is used to denote the hello string which already defined in strings.xml file. This string will be responsible for displaying the “Hello World!”.

Running The “Hello World!” Application

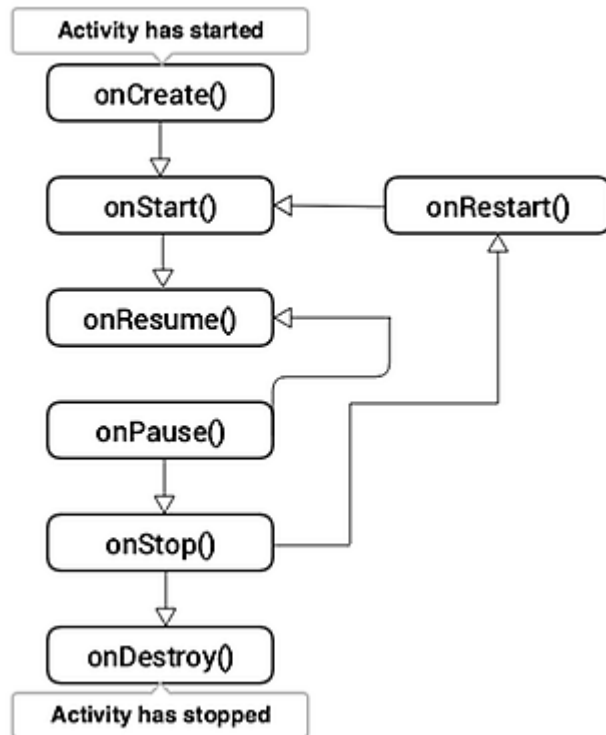
Now, let’s run your first Android application. We have already created AVD to test the application. So, to run your recently created application, you need to open the activity file of the project in the Android Studio. After which, go to the toolbar of the software and click on the Run button. The process will further help the Android Studio to install your application on the AVD. Then, the application will run to display the following output only if there are no syntax errors in the coding part. You will be able to see this output on the Emulator window.



Experiment-2: To understand Activity, Intent. Create application with login module. (Check username and password).

Solution:

Android Activities: If you are familiar with Object Oriented Programming languages which include C, C++, Java, and a few others then you must be aware that every program starts with `main()` function. Similarly, when we talk about an Android program then the initiation takes place with the launched **Activity**. Here, the first method to be called is `onCreate()` callback method.



In the provided flow chart, you can easily see that a series of callback methods (`onCreate()` method -> `onStart()` method -> `onResume()` method) initiate an Activity. While another series of callback methods (`onPause()` method -> `onStop()` method -> `onDestroy()` method) is responsible for shutting down the same Activity. The above Activity life cycle diagram will help you understand every phase of an Activity Life Cycle.

Well, an Activity class contains the definitions for various callbacks which are known as events. These events are tabulated in the table provided below. So, because of these events, you don't need to implement all the callback methods every now and then. But, it is really important that you understand the functioning and implementation of each and every callback method. With the help of this knowledge, you will be able to make sure that your application behaves according to the expectation of the end users.

S. No.	Callback Method	Description
1	<code>onCreate()</code>	<ul style="list-style-type: none">* This is the first callback method in the Activity Life Cycle.* This method is called whenever an Activity creation takes place.
2	<code>onStart()</code>	<ul style="list-style-type: none">* This callback method comes into existence when an Activity becomes visible to an end user.
3	<code>onResume()</code>	<ul style="list-style-type: none">* This method gets called whenever a user begins its interaction with the application.
4	<code>onPause()</code>	<ul style="list-style-type: none">* This method doesn't accept inputs from the user as the functioning of the program gets paused.* At the paused phase, the program doesn't even execute the code.

		* This callback method is called whenever the program pauses and the last activity gets resumed.
5	<i>onStop()</i>	* This method is called whenever the activity is no more visible to the user.
6	<i>onDestroy()</i>	* This callback method is called every time before the activity needs to be destroyed by the system.
7	<i>onRestart()</i>	* This method is called when the activity needs to be resumed after it has been stopped once.

For an instance, follow the given steps for the better understanding of an Android application activity life cycle. These steps will lead you to make a few changes in the already discussed Hello World application.

Step 1: You need to create an Android application on the Android Studio. Name this application as HelloWorld, which will be placed under the package com.example.helloworld. This has been already mentioned in the chapter, Hello World Example.

Step 2: Now, you need to modify the main activity file, which is MainActivity.java file. But make sure that you leave the rest of the files as it is. You can see these changes happening in the below section.

Step 3: At the final step, run the newly created application so that the Android Emulator gets launched. After this, check for the final output after the changes have been implemented on the application.

Below is the content present in the modified main activity file. You can locate this file at *src/com.example.helloworld/MainActivity.java*. Also, you must be aware of the fact that this *MainActivity.java* file contains all the basic activity lifecycle methods.

```
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;

public class MainActivity extends Activity
{
    String msg = "Android : "

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(msg, "The onCreate() event");
    }

    /** Called when the activity is about to become visible. */
    @Override
    protected void onStart()
    {
        super.onStart();
        Log.d(msg, "The onStart() event");
    }
}
```

```

/** Called when the activity has become visible. */
@Override
protected void onResume()
{
    super.onResume();
    Log.d(msg, "The onResume() event");
}

/** Called when another activity is taking focus. */
@Override
protected void onPause()
{
    super.onPause();
    Log.d(msg, "The onPause() event");
}

/** Called when the activity is no longer visible. */
@Override
protected void onStop()
{
    super.onStop();
    Log.d(msg, "The onStop() event");
}

/** Called just before the activity is destroyed. */
@Override
public void onDestroy()
{
    super.onDestroy();
    Log.d(msg, "The onDestroy() event");
}
}

```

Here, **Log.d()** method is used to generate log messages.

Well, an Activity class loads various UI components with the help of available XML file. This XML file can be found under the folder *res/layout*. The below command is responsible for the loading of the UI components from the file *res/layout/activity_main.xml*.

```

setContentView(R.layout.activity_main);

```

It is possible that a single application can have more than one activities without any constraints applied. But, on the other hand, it is important that you define each and every activity of your Android application in the *AndroidManifest.xml* file. You also need to declare the main activity in the manifest file. This declaration needs to be done with the help of the *<intent-filter>*. This *<intent-filter>* contains **MAIN action** and the **LAUNCHER category** too. You can clearly see this in the content of the following manifest file.

```

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

    package="com.example.w3school.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>
</application>

</manifest>

```

In any case, if you miss declaring either the MAIN action or the LAUNCHER category for one of your application's activities then it is definite that your application icon won't appear on the home screen of the system.

Now, let's run this modified Hello World! application. Here, we are assuming that you have already created the AVD during the process of environment setup. As mentioned earlier, we will be using Android Studio to run our application. So, after you run the Android Studio, open any of the activity files of your project and hit on the Run icon from the toolbar of the Android Studio. If everything is in place then you will be displayed with an Emulator window. On this window, you can see the log messages generated by the application. These log messages will be displayed on the **LogCat window** on the Android Studio.

```

08-23 10:32:07.682 4480-4480/com.example.helloworld D/Android :: The onCreate() event
08-23 10:32:07.683 4480-4480/com.example.helloworld D/Android :: The onStart() event
08-23 10:32:07.685 4480-4480/com.example.helloworld D/Android :: The onResume() event

```



Now, locate the lock screen button on the Emulator. Once, you find it, click on the button to generate the below-provided event messages. These event messages will get displayed again on the LogCat window.

```

08-23 10:32:53.230 4480-4480/com.example.helloworld D/Android :: The onPause() event
08-23 10:32:53.294 4480-4480/com.example.helloworld D/Android :: The onStop() event

```

After this, try unlocking the screen. It will generate the following event messages on the LogCat window.

```

08-23 10:34:41.390 4480-4480/com.example.helloworld D/Android :: The onStart() event
08-23 10:34:41.392 4480-4480/com.example.helloworld D/Android :: The onResume() event

```

At last, try clicking on the Back button on the Emulator. And the following event messages will be generated as a result of that.

```

08-23 10:37:24.806 4480-4480/com.example.helloworld D/Android :: The onPause() event
08-23 10:37:25.668 4480-4480/com.example.helloworld D/Android :: The onStop() event
08-23 10:37:25.669 4480-4480/com.example.helloworld D/Android :: The onDestroy() event

```

With this, the whole Activity Life Cycle of an Android application completes itself.

Android - Intents and Filters

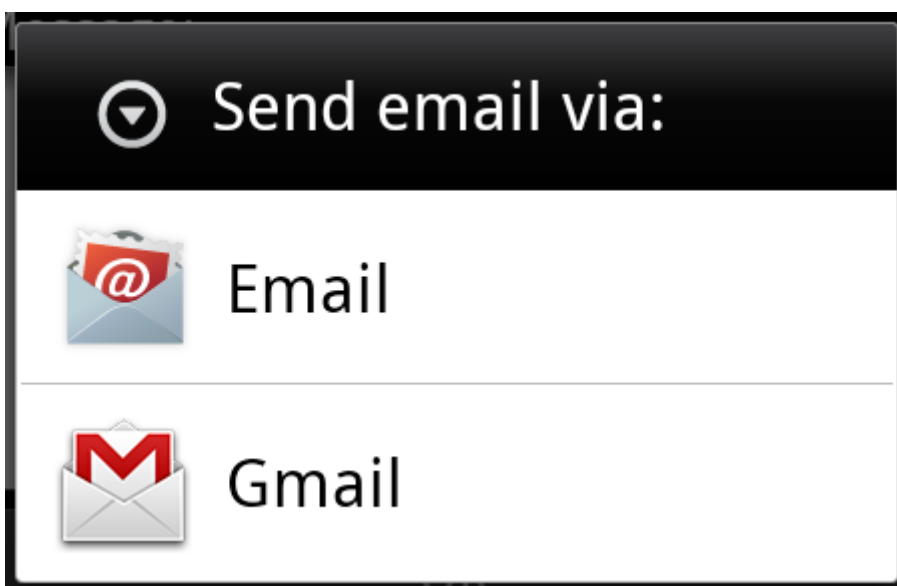
An Android **Intent** is an abstract description of an operation to be performed. It can be used with **startActivity** to launch an Activity, **broadcastIntent** to send it to any interested BroadcastReceiver components, and **startService(Intent)** or **bindService(Intent, ServiceConnection, int)** to communicate with a background Service.

The intent itself, an Intent object, is a passive data structure holding an abstract description of an operation to be performed.

For example, let's assume that you have an Activity that needs to launch an email client and sends an email using your Android device. For this purpose, your Activity would send an ACTION_SEND along with appropriate **chooser**, to the Android Intent Resolver. The specified chooser gives the proper interface for the user to pick how to send your email data.

```
Intent email = new Intent(Intent.ACTION_SEND, Uri.parse("mailto:"));
email.putExtra(Intent.EXTRA_EMAIL, recipients);
email.putExtra(Intent.EXTRA_SUBJECT, subject.getText().toString());
email.putExtra(Intent.EXTRA_TEXT, body.getText().toString());
startActivity(Intent.createChooser(email, "Choose an email client
from..."));
```

Above syntax is calling startActivity method to start an email activity and result should be as shown below
—



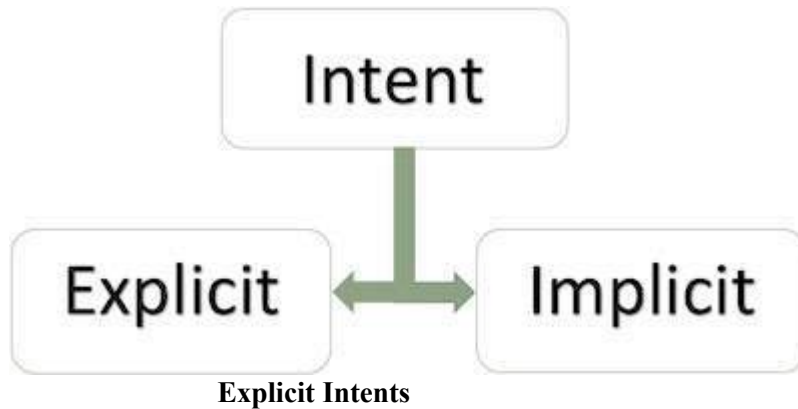
For example, assume that you have an Activity that needs to open URL in a web browser on your Android device. For this purpose, your Activity will send ACTION_WEB_SEARCH Intent to the Android Intent Resolver to open given URL in the web browser. The Intent Resolver parses through a list of Activities and chooses the one that would best match your Intent, in this case, the Web Browser Activity. The Intent Resolver then passes your web page to the web browser and starts the Web Browser Activity.

```
String q = "tutorialspoint";
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
intent.putExtra(SearchManager.QUERY, q);
startActivity(intent);
```


Above example will search as **tutorialspoint** on android search engine and it gives the result of tutorialspoint in your an activity.

Types of Intents

There are following two types of intents supported by Android



Explicit intent going to be connected internal world of application, suppose if you wants to connect one activity to another activity, we can do this quote by explicit intent, below image is connecting first activity to second activity by clicking button.



These intents designate the target component by its name and they are typically used for application-internal messages - such as an activity starting a subordinate service or launching a sister activity. For example –

```
// Explicit Intent by specifying its class name
Intent i = new Intent(FirstActivity.this, SecondActivity.class);
```

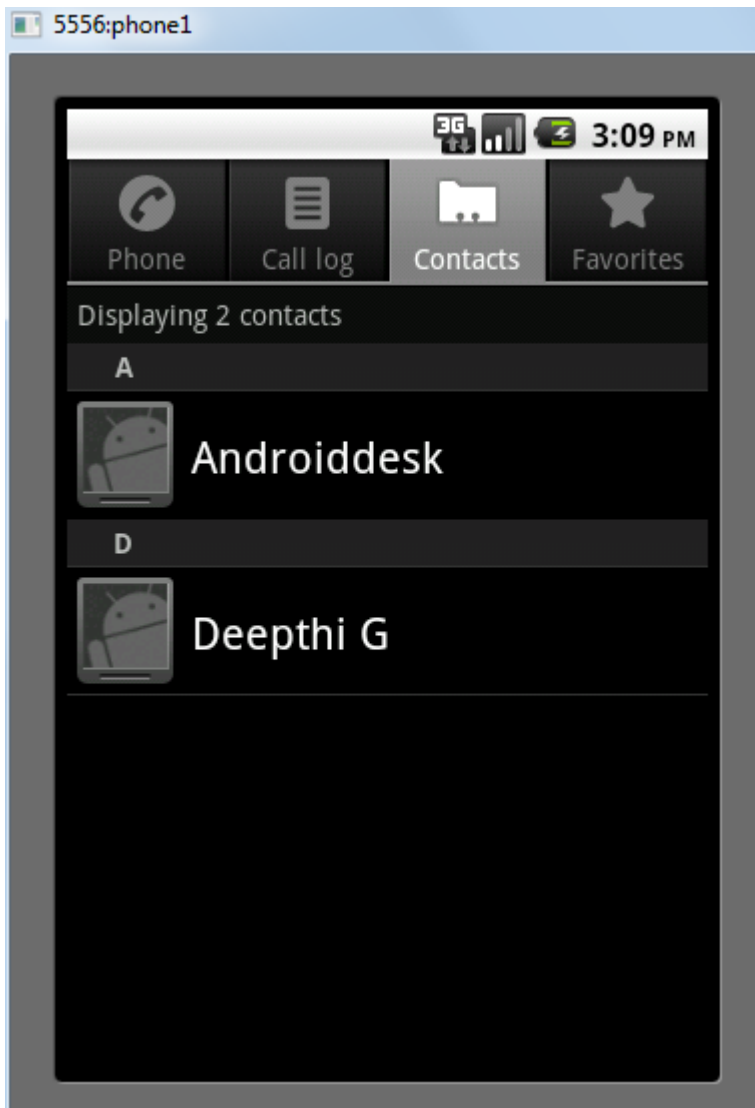
```
// Starts TargetActivity
startActivity(i);
```

Implicit Intents

These intents do not name a target and the field for the component name is left blank. Implicit intents are often used to activate components in other applications. For example –

```
Intent read1=new Intent();  
read1.setAction(android.content.Intent.ACTION_VIEW);  
read1.setData(ContactsContract.Contacts.CONTENT_URI);  
startActivity(read1);
```

Above code will give result as shown below



The target component which receives the intent can use the **getExtras()** method to get the extra data sent by the source component. For example –

```
// Get bundle object at appropriate place in your code  
Bundle extras = getIntent().getExtras();  
  
// Extract data using passed keys  
String value1 = extras.getString("Key1");  
String value2 = extras.getString("Key2");
```

Android - Login Screen

A login application is the screen asking your credentials to login to some particular application. You might have seen it when logging into facebook, twitter e.t.c

This chapter explains, how to create a login screen and how to manage security when false attempts are made.

First you have to define two TextView asking username and password of the user. The password TextView must have **inputType** set to password. Its syntax is given below –

```
<EditText
    android:id = "@+id/editText2"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:inputType = "textPassword" />
```

```
<EditText
    android:id = "@+id/editText1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
/>
```

Define a button with login text and set its **onClick** Property. After that define the function mentioned in the onClick property in the java file.

```
<Button
    android:id = "@+id/button1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:onClick = "login"
    android:text = "@string/Login"
/>
```

In the java file, inside the method of onClick get the username and passwords text using **getText()** and **toString()** method and match it with the text using **equals()** function.

```
EditText username = (EditText)findViewById(R.id.editText1);
EditText password = (EditText)findViewById(R.id.editText2);
```

```
public void login(View view){
    if(username.getText().toString().equals("admin") && password.getText().toString().equals("admin")){

        //correct password
```

```

    }else{
        //wrong password
    }

```

The last thing you need to do is to provide a security mechanism, so that unwanted attempts should be avoided. For this initialize a variable and on each false attempt, decrement it. And when it reaches to 0, disable the login button.

```

int counter = 3;
counter--;

if(counter==0){
    //disble the button, close the application e.t.c
}

```

Example

Here is an example demonstrating a login application. It creates a basic application that gives you only three attempts to login to an application.

To experiment with this example, you can run this on an actual device or in an emulator.

Steps	Description
1	You will use Android studio to create an Android application under a package com.example.sairamkrishna.myapplication.
3	Modify src/MainActivity.java file to add necessary code.
4	Modify the res/layout/activity_main to add respective XML components
5	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 main activity file **src/MainActivity.java**.

```

package com.example.sairamkrishna.myapplication;

```

```

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;

```

```

import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

```

```

public class MainActivity extends Activity {
    Button b1,b2;
    EditText ed1,ed2;

    TextView tx1;
    int counter = 3;

    @Override

```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    b1 = (Button)findViewById(R.id.button);
    ed1 = (EditText)findViewById(R.id.editText);
    ed2 = (EditText)findViewById(R.id.editText2);

    b2 = (Button)findViewById(R.id.button2);
    tx1 = (TextView)findViewById(R.id.textView3);
    tx1.setVisibility(View.GONE);

    b1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(ed1.getText().toString().equals("admin") &&
                ed2.getText().toString().equals("admin")) {
                Toast.makeText(getApplicationContext(),
                    "Redirecting...",Toast.LENGTH_SHORT).show();
            }else{
                Toast.makeText(getApplicationContext(), "Wrong
                    Credentials",Toast.LENGTH_SHORT).show();

                tx1.setVisibility(View.VISIBLE);
                tx1.setBackgroundColor(Color.RED);
                counter--;
                tx1.setText(Integer.toString(counter));

                if (counter == 0) {
                    b1.setEnabled(false);
                }
            }
        }
    });

    b2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish();
        }
    });
}

```

Following is the modified content of the xml **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:text = "Login" android:layout_width="wrap_content"
        android:layout_height = "wrap_content"
        android:id = "@+id/textview"

```

```
    android:textSize = "35dp"
    android:layout_alignParentTop = "true"
    android:layout_centerHorizontal = "true" />
```

```
<TextView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "Tutorials point"
    android:id = "@+id/textView"
    android:layout_below = "@+id/textview"
    android:layout_centerHorizontal = "true"
    android:textColor = "#ff7aff24"
    android:textSize = "35dp" />
```

```
<EditText
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:id = "@+id/editText"
    android:hint = "Enter Name"
    android:focusable = "true"
    android:textColorHighlight = "#ff7eff15"
    android:textColorHint = "#ffff25e6"
    android:layout_marginTop = "46dp"
    android:layout_below = "@+id/imageView"
    android:layout_alignParentLeft = "true"
    android:layout_alignParentStart = "true"
    android:layout_alignParentRight = "true"
    android:layout_alignParentEnd = "true" />
```

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:src="@drawable/abc"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true" />
```

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:id="@+id/editText2"
    android:layout_below="@+id/editText"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignRight="@+id/editText"
    android:layout_alignEnd="@+id/editText"
    android:textColorHint="#ffff299f"
    android:hint="Password" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Attempts Left:"
```

```

        android:id="@+id/textView2"
        android:layout_below="@+id/editText2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textSize="25dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Text"
    android:id="@+id/textView3"
    android:layout_alignTop="@+id/textView2"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignBottom="@+id/textView2"
    android:layout_toEndOf="@+id/textview"
    android:textSize="25dp"
    android:layout_toRightOf="@+id/textview" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="login"
    android:id="@+id/button"
    android:layout_alignParentBottom="true"
    android:layout_toLeftOf="@+id/textview"
    android:layout_toStartOf="@+id/textview" />

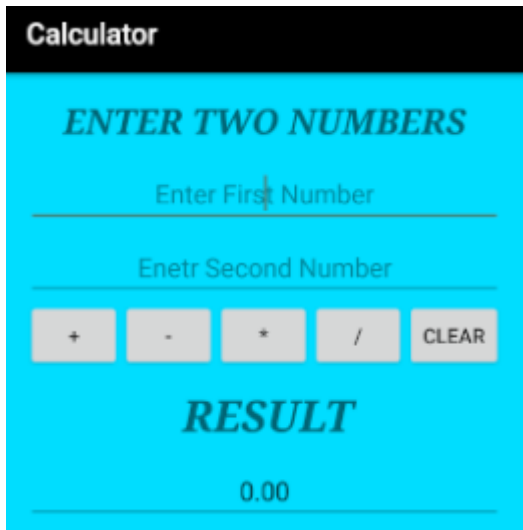
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cancel"
    android:id="@+id/button2"
    android:layout_alignParentBottom="true"
    android:layout_toRightOf="@+id/textview"
    android:layout_toEndOf="@+id/textview" />

</RelativeLayout>

```

Experiment-3: Design simple GUI application with activity and intents e.g.calculator.

Solution:



The complete interface code of 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:id="@+id/activity_main"
    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="abhiandroid.com.calculater.MainActivity"
    android:orientation="vertical"
    android:gravity="top"
    android:textAlignment="center"
    android:background="@android:color/holo_blue_bright"
    android:weightSum="1">

    <TextView
        android:text="@string/enter_two_numbers"
        android:layout_width="match_parent"
        android:id="@+id/textView"
        android:layout_height="30dp"
        android:gravity="center_horizontal"
        android:textColorLink="?android:attr/editTextColor"
        tools:textStyle="bold|italic"
        android:textStyle="bold|italic"
        android:fontFamily="serif"
        android:visibility="visible"
        android:textSize="24sp"
        android:layout_weight="0.07" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
```



```
android:id="@+id/editOp1"  
android:textSize="18sp"  
android:gravity="center_horizontal"  
android:layout_marginBottom="5dp"  
android:visibility="visible" />
```

```
<EditText  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:inputType="number"  
    android:ems="10"  
    android:id="@+id/editOp2"  
    android:textSize="18sp"  
    android:gravity="center_horizontal"  
    android:elevation="1dp" />
```

```
<LinearLayout  
    android:orientation="horizontal"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">
```

```
<Button  
    android:text="+"  
    android:layout_width="78dp"  
    android:layout_height="wrap_content"  
    android:id="@+id/btnadd"  
    android:layout_weight="0.03" />
```

```
<Button  
    android:text="-"  
    android:layout_width="78dp"  
    android:layout_height="wrap_content"  
    android:id="@+id/btnsub"  
    android:layout_weight="0.03" />
```

```
<Button  
    android:text="*"  
    android:layout_width="78dp"  
    android:layout_height="wrap_content"  
    android:id="@+id/btnmul"  
    android:layout_weight="0.03"/>
```

```
<Button  
    android:text="/" "  
    android:layout_height="wrap_content"  
    android:id="@+id/btndiv"  
    android:layout_width="78dp"  
    android:layout_weight="0.03" />
```

```
<Button  
    android:text="Clear"  
    android:layout_width="80dp"  
    android:layout_height="wrap_content"  
    android:id="@+id/btnclr"  
    android:layout_weight="0.03" />
```

```

</LinearLayout>

<TextView
    android:text="@string/result"
    android:layout_width="332dp"
    android:id="@+id/textView1"
    android:layout_marginTop="10dp"
    android:layout_height="50dp"
    android:gravity="center_horizontal"
    android:textColorLink="?android:attr/editTextColor"
    tools:textStyle="bold|italic"
    android:textStyle="bold|italic"
    android:fontFamily="serif"
    android:visibility="visible"
    android:textSize="30sp" />

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/result"
    android:textSize="18sp"
    android:text="0.00"
    android:gravity="center_horizontal" />
</LinearLayout>

```

MainActivity.java: The interface part of the application is over, let's focus on adding functionality to the application. This calculator app basically perform five operations i.e addition, subtraction, multiplication, division and reset. So for that we need to define these operation over button click. For that we use `setOnClickListener()` function.

`parseDouble()` is used to convert String value to double. By default the value is String and we need to convert it into Double to perform operation over it.

If person doesn't enter the value and directly click on the any button then a Toast message will appear on the screen telling user to enter the required numbers.

```

package abhiandroid.com.calculator;
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.TextView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    private EditText opr1;
    private EditText opr2;
    private Button btnadd;
    private Button btnsub;
    private Button btnmul;
    private Button btndiv;
    private Button btnclr;
    private TextView txtresult;

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    opr1 = (EditText) findViewById(R.id.editOp1);
    opr2 = (EditText) findViewById(R.id.editOp2);
    btnadd = (Button) findViewById(R.id.btnAdd);
    btnsub = (Button) findViewById(R.id.btnsub);
    btnmul = (Button) findViewById(R.id.btnmul);
    btndiv = (Button) findViewById(R.id.btndiv);
    btnclr = (Button) findViewById(R.id.btnclr);
    txtresult = (TextView) findViewById(R.id.result);
    // Addition
    btnadd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if((opr1.getText().length()>0) && (opr2.getText().length()>0))
            {
                double oper1 = Double.parseDouble(opr1.getText().toString());
                double oper2 = Double.parseDouble(opr2.getText().toString());
                double result = oper1 + oper2;
                txtresult.setText(Double.toString(result));
            }
            else{
                Toast toast= Toast.makeText(MainActivity.this,"Enter The Required
Numbers",Toast.LENGTH_LONG);
                toast.show();
            }
        }
    });
    //Subtraction
    btnsub.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if((opr1.getText().length()>0) && (opr2.getText().length()>0))
            {
                double oper1 = Double.parseDouble(opr1.getText().toString());
                double oper2 = Double.parseDouble(opr2.getText().toString());
                double result = oper1 - oper2;
                txtresult.setText(Double.toString(result));
            }
            else{
                Toast toast= Toast.makeText(MainActivity.this,"Enter The Required
Numbers",Toast.LENGTH_LONG);
                toast.show();
            }
        }
    });
    // Multiplication
    btnmul.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if((opr1.getText().length()>0) && (opr2.getText().length()>0))
            {

```

```

        double oper1 = Double.parseDouble(opr1.getText().toString());
        double oper2 = Double.parseDouble(opr2.getText().toString());
        double result = oper1 * oper2;
        txtresult.setText(Double.toString(result));
    }
    else{
        Toast toast= Toast.makeText(MainActivity.this,"Enter The Required
Numbers",Toast.LENGTH_LONG);
        toast.show();
    }
}
});
// Division
btndiv.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if((opr1.getText().length()>0) && (opr2.getText().length()>0))
        {
            double oper1 = Double.parseDouble(opr1.getText().toString());
            double oper2 = Double.parseDouble(opr2.getText().toString());
            double result = oper1 / oper2;
            txtresult.setText(Double.toString(result));
        }
        else{
            Toast toast= Toast.makeText(MainActivity.this,"Enter The Required
Numbers",Toast.LENGTH_LONG);
            toast.show();
        }
    }
});
// Reset Feilds
btnclr.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        opr1.setText("");
        opr2.setText("");
        txtresult.setText("0.00");
        opr1.requestFocus();
    }
});
}
}

```

Experiment-4: Develop an application that makes use of RSS Feed.

Solution:

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

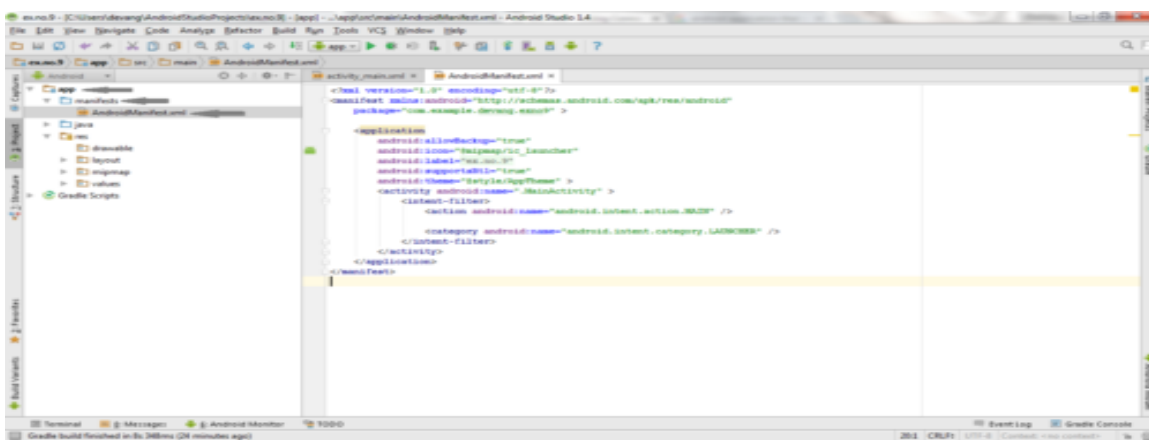
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

</LinearLayout>

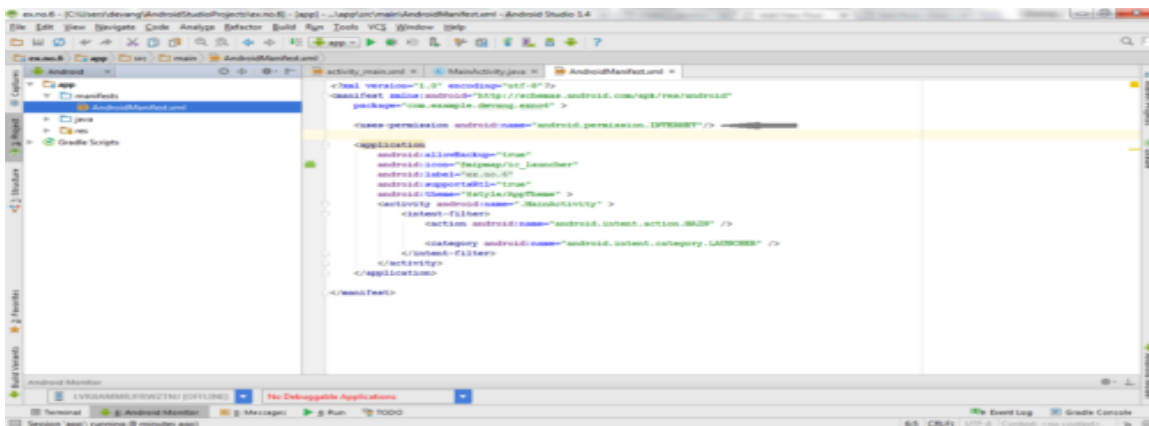
```

Adding permissions in Manifest for the Android Application:

- Click on **app** -> **manifests** -> **AndroidManifest.xml**



- Now include the **INTERNET** permissions in the AndroidManifest.xml file as shown below



Code for AndroidManifest.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.exno6" >

    <uses-permission android:name="android.permission.INTERNET"/>

    <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>
    </application>

</manifest>

```

So now the Permissions are added in the Manifest.

Code for MainActivity.java:

```

package com.example.exno6;
import android.app.ListActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends ListActivity
{
    List headlines;
    List links;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        new MyAsyncTask().execute();
    }
    class MyAsyncTask extends AsyncTask<Object,Void,ArrayAdapter>
    {
        @Override
        protected ArrayAdapter doInBackground(Object[] params)
        {
            headlines = new ArrayList();

```

```

links = new ArrayList();
try
{
    URL url = new URL("https://codingconnect.net/feed");
    XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
    factory.setNamespaceAware(false);
    XmlPullParser xpp = factory.newPullParser();

    // We will get the XML from an input stream
    xpp.setInput(getInputStream(url), "UTF_8");
    boolean insideItem = false;
    // Returns the type of current event: START_TAG, END_TAG, etc..
    int eventType = xpp.getEventType();
    while (eventType != XmlPullParser.END_DOCUMENT)
    {
        if (eventType == XmlPullParser.START_TAG)
        {
            if (xpp.getName().equalsIgnoreCase("item"))
            {
                insideItem = true;
            }
            else if (xpp.getName().equalsIgnoreCase("title"))
            {
                if (insideItem)
                    headlines.add(xpp.nextText()); //extract the headline
            }
            else if (xpp.getName().equalsIgnoreCase("link"))
            {
                if (insideItem)
                    links.add(xpp.nextText()); //extract the link of article
            }
        }
        else if(eventType==XmlPullParser.END_TAG && xpp.getName().equalsIgnoreCase("item"))
        {
            insideItem=false;
        }
        eventType = xpp.next(); //move to next element
    }
}
catch (MalformedURLException e)
{
    e.printStackTrace();
}
catch (XmlPullParserException e)
{
    e.printStackTrace();
}
catch (IOException e)
{
    e.printStackTrace();
}
return null;
}
protected void onPostExecute(ArrayAdapter adapter)
{

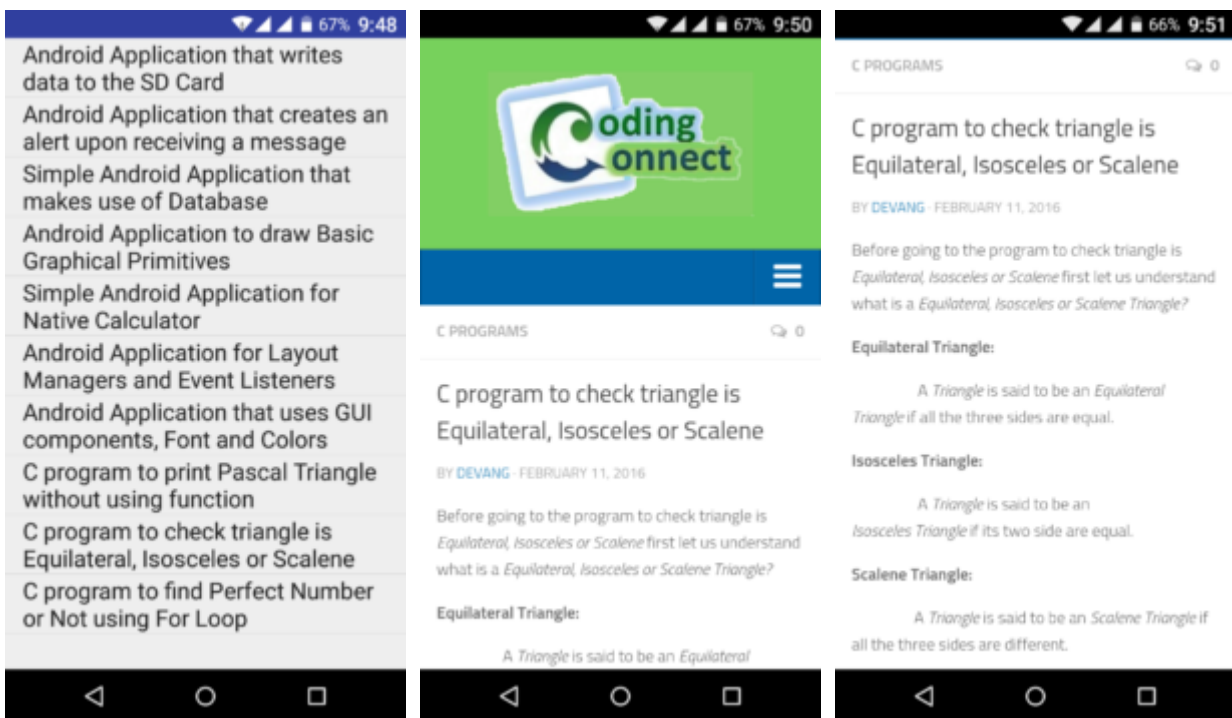
```

```

        adapter = new ArrayAdapter(MainActivity.this, android.R.layout.simple_list_item_1, headlines);
        setListAdapter(adapter);
    }
}
@Override
protected void onItemClick(ListView l, View v, int position, long id)
{
    Uri uri = Uri.parse((links.get(position)).toString());
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    startActivity(intent);
}
public InputStream getInputStream(URL url)
{
    try
    {
        return url.openConnection().getInputStream();
    }
    catch (IOException e)
    {
        return null;
    }
}
}
}

```

Output:



Experiment-5: Write an application that draws basic graphical primitives on the screen.

Solution:

Code for Activity_main.xml:


```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/imageView" />
</RelativeLayout>

```

Code for MainActivity.java:

```

package com.example.exno4;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.widget.ImageView;

public class MainActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Creating a Bitmap
        Bitmap bg = Bitmap.createBitmap(720, 1280, Bitmap.Config.ARGB_8888);

        //Setting the Bitmap as background for the ImageView
        ImageView i = (ImageView) findViewById(R.id.imageView);
        i.setBackgroundDrawable(new BitmapDrawable(bg));

        //Creating the Canvas Object
        Canvas canvas = new Canvas(bg);

        //Creating the Paint Object and set its color & TextSize
        Paint paint = new Paint();
        paint.setColor(Color.BLUE);
        paint.setTextSize(50);

        //To draw a Rectangle
        canvas.drawText("Rectangle", 420, 150, paint);
        canvas.drawRect(400, 200, 650, 700, paint);

        //To draw a Circle
        canvas.drawText("Circle", 120, 150, paint);
        canvas.drawCircle(200, 350, 150, paint);

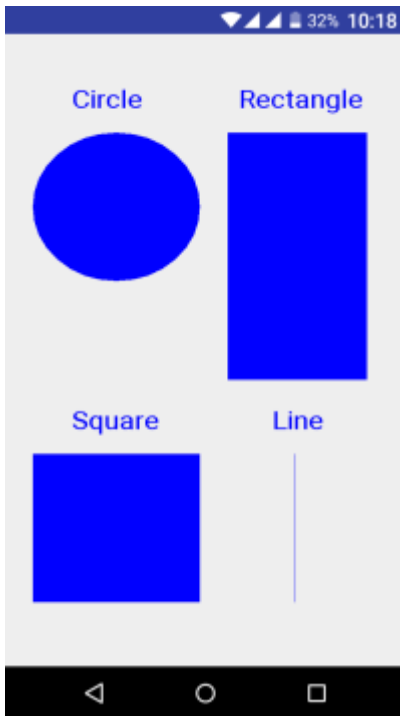
        //To draw a Square

```

```
canvas.drawText("Square", 120, 800, paint);
canvas.drawRect(50, 850, 350, 1150, paint);

//To draw a Line
canvas.drawText("Line", 480, 800, paint);
canvas.drawLine(520, 850, 520, 1150, paint);
}
```

Output:



Experiment-6: Create an android app for database creation using SQLite Database.

Solution:

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
	openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)
1	This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN_READWRITE OPEN_READONLY openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)
2	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 openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)
3	It not only opens but create the database if it not exists. This method is equivalent to openDatabase method. openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)
4	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 TutorialPoint(Username  
VARCHAR,Password VARCHAR);");  
mydatabase.execSQL("INSERT INTO TutorialPoint 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
-------	----------------------

	execSQL(String sql, Object[] bindArgs)
--	---

- | | |
|---|---|
| 1 | 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 = mydatabase.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
-------	----------------------

	getColumnCount()
--	-------------------------

- | | |
|---|--|
| 1 | This method return the total number of columns of the table. |
|---|--|

	getColumnIndex(String columnName)
--	--

- | | |
|---|---|
| 2 | This method returns the index number of a column by specifying the name of the column |
|---|---|

	getColumnName(int columnIndex)
--	---------------------------------------

- | | |
|---|--|
| 3 | This method returns the name of the column by specifying the index of the column |
|---|--|

	getColumnNames()
--	-------------------------

- | | |
|---|---|
| 4 | This method returns the array of all the column names of the table. |
|---|---|

	getCount()
--	-------------------

- | | |
|---|--|
| 5 | This method returns the total number of rows in the cursor |
|---|--|

	getPosition()
--	----------------------

- | | |
|---|---|
| 6 | This method returns the current position of the cursor in the table |
|---|---|

	isClosed()
--	-------------------

- | | |
|---|---|
| 7 | 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 applications 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.

Experiment-7: Develop a native application that uses GPS location information.

Solution:

Android location APIs make it easy for you to build location-aware applications, without needing to focus on the details of the underlying location technology.

This becomes possible with the help of **Google Play services**, which facilitates adding location awareness to your app with automated location tracking, geofencing, and activity recognition. This tutorial shows you how to use Location Services in your APP to get the current location, get periodic location updates, look up addresses etc.

The Location Object

The **Location** object represents a geographic location which can consist of a latitude, longitude, time stamp, and other information such as bearing, altitude and velocity. There are following important methods which you can use with Location object to get location specific information –

Sr.No.	Method & Description
1	float distanceTo(Location dest) Returns the approximate distance in meters between this location and the given location.
2	float getAccuracy() Get the estimated accuracy of this location, in meters.
3	double getAltitude() Get the altitude if available, in meters above sea level.
4	float getBearing() Get the bearing, in degrees.
5	double getLatitude() Get the latitude, in degrees.
6	double getLongitude() Get the longitude, in degrees.
7	float getSpeed() Get the speed if it is available, in meters/second over ground.
8	boolean hasAccuracy() True if this location has an accuracy.
9	boolean hasAltitude() True if this location has an altitude.
10	boolean hasBearing() True if this location has a bearing.
11	boolean hasSpeed() True if this location has a speed.
12	void reset() Clears the contents of the location.
13	void setAccuracy(float accuracy) Set the estimated accuracy of this location, meters.
14	void setAltitude(double altitude) Set the altitude, in meters above sea level.
15	void setBearing(float bearing) Set the bearing, in degrees.
16	void setLatitude(double latitude) Set the latitude, in degrees.
17	void setLongitude(double longitude) Set the longitude, in degrees.
18	void setSpeed(float speed) Set the speed, in meters/second over ground.
19	String toString() Returns a string containing a concise, human-readable description of this object.

Get the Current Location

To get the current location, create a location client which is **LocationClient** object, connect it to Location Services using **connect()** method, and then call its **getLastLocation()** method. This method returns the

most recent location in the form of **Location** object that contains latitude and longitude coordinates and other information as explained above. To have location based functionality in your activity, you will have to implement two interfaces –

- `GooglePlayServicesClient.ConnectionCallbacks`
- `GooglePlayServicesClient.OnConnectionFailedListener`

These interfaces provide following important callback methods, which you need to implement in your activity class –

Sr.No.	Callback Methods & Description
	abstract void onConnected(Bundle connectionHint)
1	This callback method is called when location service is connected to the location client successfully. You will use connect() method to connect to the location client.
	abstract void onDisconnected()
2	This callback method is called when the client is disconnected. You will use disconnect() method to disconnect from the location client.
	abstract void onConnectionFailed(ConnectionResult result)
3	This callback method is called when there was an error connecting the client to the service.

You should create the location client in **onCreate()** method of your activity class, then connect it in **onStart()**, so that Location Services maintains the current location while your activity is fully visible. You should disconnect the client in **onStop()** method, so that when your app is not visible, Location Services is not maintaining the current location. This helps in saving battery power up-to a large extent.

Get the Updated Location

If you are willing to have location updates, then apart from above mentioned interfaces, you will need to implement **LocationListener** interface as well. This interface provide following callback method, which you need to implement in your activity class –

Sr.No.	Callback Method & Description
	abstract void onLocationChanged(Location location)
1	This callback method is used for receiving notifications from the LocationClient when the location has changed.

Location Quality of Service

The **LocationRequest** object is used to request a quality of service (QoS) for location updates from the **LocationClient**. There are following useful setter methods which you can use to handle QoS. There are equivalent getter methods available which you can check in Android official documentation.

Sr.No.	Method & Description
	setExpirationDuration(long millis)
1	Set the duration of this request, in milliseconds.
	setExpirationTime(long millis)
2	Set the request expiration time, in millisecond since boot.
	setFastestInterval(long millis)
3	Explicitly set the fastest interval for location updates, in milliseconds.
	setInterval(long millis)
4	Set the desired interval for active location updates, in milliseconds.
	setNumUpdates(int numUpdates)
5	Set the number of location updates.
	setPriority(int priority)
6	Set the priority of the request.

Now for example, if your application wants high accuracy location it should create a location request with **setPriority(int)** set to `PRIORITY_HIGH_ACCURACY` and **setInterval(long)** to 5 seconds. You can also use bigger interval and/or other priorities like `PRIORITY_LOW_POWER` for to request "city" level accuracy or `PRIORITY_BALANCED_POWER_ACCURACY` for "block" level accuracy.

Activities should strongly consider removing all location request when entering the background (for example at `onPause()`), or at least swap the request to a larger interval and lower quality to save power

consumption.

Displaying a Location Address

Once you have **Location** object, you can use **Geocoder.getFromLocation()** method to get an address for a given latitude and longitude. This method is synchronous, and may take a long time to do its work, so you should call the method from the **doInBackground()** method of an **AsyncTask** class.

The **AsyncTask** must be subclassed to be used and the subclass will override **doInBackground(Params...)** method to perform a task in the background and **onPostExecute(Result)** method is invoked on the UI thread after the background computation finishes and at the time to display the result. There is one more important method available in **AsyncTask** which is **execute(Params... params)**, this method executes the task with the specified parameters.

Example

Following example shows you in practical how to use Location Services in your app to get the current location and its equivalent addresses etc.

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.

Create Android Application

Step	Description
1	You will use Android studio IDE to create an Android application and name it as <i>Tutorialspoint</i> under a package <i>com.example.tutorialspoint7.myapplication</i> .
2	add <i>src/GPSTracker.java</i> file and add required code.
3	Modify <i>src/MainActivity.java</i> file and add required code as shown below to take care of getting current location and its equivalent address.
4	Modify layout XML file <i>res/layout/activity_main.xml</i> to add all GUI components which include three buttons and two text views to show location/address.
5	Modify <i>res/values/strings.xml</i> to define required constant values
6	Modify <i>AndroidManifest.xml</i> as shown below
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**.
package com.example.tutorialspoint7.myapplication;

```
import android.Manifest;
import android.app.Activity;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.test.mock.MockPackageManager;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
```

```
public class MainActivity extends Activity {
```

```
    Button btnShowLocation;
    private static final int REQUEST_CODE_PERMISSION = 2;
    String mPermission = Manifest.permission.ACCESS_FINE_LOCATION;
```

```
// GPSTracker class
GPSTracker gps;
```

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



```

try {
    if (ActivityCompat.checkSelfPermission(this, mPermission)
        != MockPackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions(this, new String[]{mPermission},
            REQUEST_CODE_PERMISSION);

        // If any permission above not allowed by user, this condition will
        // execute every time, else your else part will work
    }
} catch (Exception e) {
    e.printStackTrace();
}

btnShowLocation = (Button) findViewById(R.id.button);

// show location button click event
btnShowLocation.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        // create class object
        gps = new GPSTracker(MainActivity.this);

        // check if GPS enabled
        if(gps.canGetLocation()){

            double latitude = gps.getLatitude();
            double longitude = gps.getLongitude();

            // \n is for new line
            Toast.makeText(getApplicationContext(), "Your Location is - \nLat: "
                + latitude + "\nLong: " + longitude, Toast.LENGTH_LONG).show();
        }else{
            // can't get location
            // GPS or Network is not enabled
            // Ask user to enable GPS/network in settings
            gps.showSettingsAlert();
        }
    }
});
}

```

Following is the content of the modified main activity file **GPSTracker.java**.

```

package com.example.tutorialspoint7.myapplication;

```

```

import android.app.AlertDialog;
import android.app.Service;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;

```

```

import android.location.LocationManager;
import android.os.Bundle;
import android.os.IBinder;
import android.provider.Settings;
import android.util.Log;

public class GPSTracker extends Service implements LocationListener {

    private final Context mContext;

    // flag for GPS status
    boolean isGPSEnabled = false;

    // flag for network status
    boolean isNetworkEnabled = false;

    // flag for GPS status
    boolean canGetLocation = false;

    Location location; // location
    double latitude; // latitude
    double longitude; // longitude

    // The minimum distance to change Updates in meters
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10 meters

    // The minimum time between updates in milliseconds
    private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute

    // Declaring a Location Manager
    protected LocationManager locationManager;

    public GPSTracker(Context context) {
        this.mContext = context;
        getLocation();
    }

    public Location getLocation() {
        try {
            locationManager = (LocationManager) mContext.getSystemService(LOCATION_SERVICE);

            // getting GPS status
            isGPSEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);

            // getting network status
            isNetworkEnabled = locationManager
                .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

            if (!isGPSEnabled && !isNetworkEnabled) {
                // no network provider is enabled
            } else {
                this.canGetLocation = true;
                // First get location from Network Provider
                if (isNetworkEnabled) {
                    locationManager.requestLocationUpdates(

```

```

        locationManager.NETWORK_PROVIDER,
        MIN_TIME_BW_UPDATES,
        MIN_DISTANCE_CHANGE_FOR_UPDATES, this);

    Log.d("Network", "Network");
    if (locationManager != null) {
        location = locationManager
            .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

        if (location != null) {
            latitude = location.getLatitude();
            longitude = location.getLongitude();
        }
    }
}

// if GPS Enabled get lat/long using GPS Services
if (isGPSEnabled) {
    if (location == null) {
        locationManager.requestLocationUpdates(
            LocationManager.GPS_PROVIDER,
            MIN_TIME_BW_UPDATES,
            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);

        Log.d("GPS Enabled", "GPS Enabled");
        if (locationManager != null) {
            location = locationManager
                .getLastKnownLocation(LocationManager.GPS_PROVIDER);

            if (location != null) {
                latitude = location.getLatitude();
                longitude = location.getLongitude();
            }
        }
    }
}

} catch (Exception e) {
    e.printStackTrace();
}

return location;
}

/**
 * Stop using GPS listener
 * Calling this function will stop using GPS in your app
 */

public void stopUsingGPS(){
    if(locationManager != null){
        locationManager.removeUpdates(GPSTracker.this);
    }
}

```

```

/**
 * Function to get latitude
 * */

public double getLatitude(){
    if(location != null){
        latitude = location.getLatitude();
    }

    // return latitude
    return latitude;
}

/**
 * Function to get longitude
 * */

public double getLongitude(){
    if(location != null){
        longitude = location.getLongitude();
    }

    // return longitude
    return longitude;
}

/**
 * Function to check GPS/wifi enabled
 * @return boolean
 * */

public boolean canGetLocation() {
    return this.canGetLocation;
}

/**
 * Function to show settings alert dialog
 * On pressing Settings button will launch Settings Options
 * */

public void showSettingsAlert(){
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);

    // Setting Dialog Title
    alertDialog.setTitle("GPS is settings");

    // Setting Dialog Message
    alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?");

    // On pressing Settings button
    alertDialog.setPositiveButton("Settings", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog,int which) {
            Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
            mContext.startActivity(intent);
        }
    });
}

```

```

    }
});

// on pressing cancel button
AlertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        dialog.cancel();
    }
});

// Showing Alert Message
AlertDialog.show();
}

@Override
public void onLocationChanged(Location location) {
}

@Override
public void onProviderDisabled(String provider) {
}

@Override
public void onProviderEnabled(String provider) {
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
}

@Override
public IBinder onBind(Intent arg0) {
    return null;
}
}

```

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/button"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "getLocation"/>

```

```

</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">Tutorialspoint</string>
</resources>


```

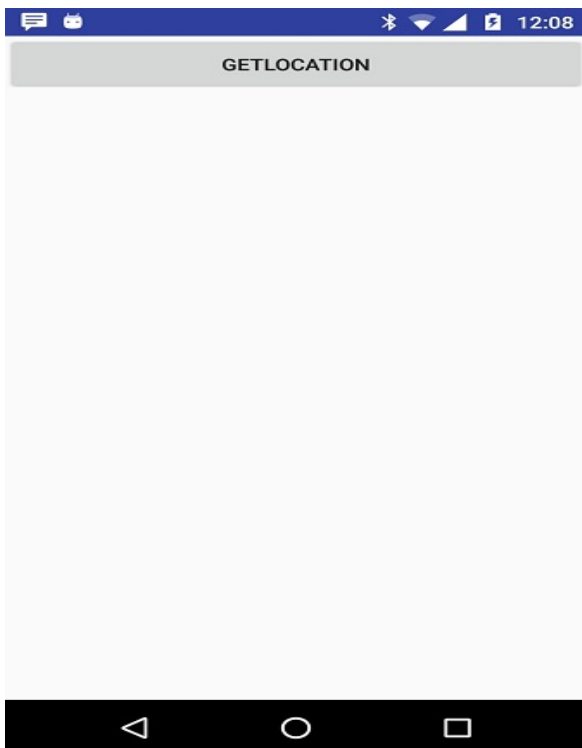
Following is the default content of **AndroidManifest.xml** –

```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "com.example.tutorialspoint7.myapplication">
    <uses-permission android:name = "android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name = "android.permission.INTERNET" />
    <application
        android:allowBackup = "true"
        android:icon = "@mipmap/ic_launcher"
        android:label = "@string/app_name"
        android:supportsRtl = "true"
        android:theme = "@style/AppTheme">

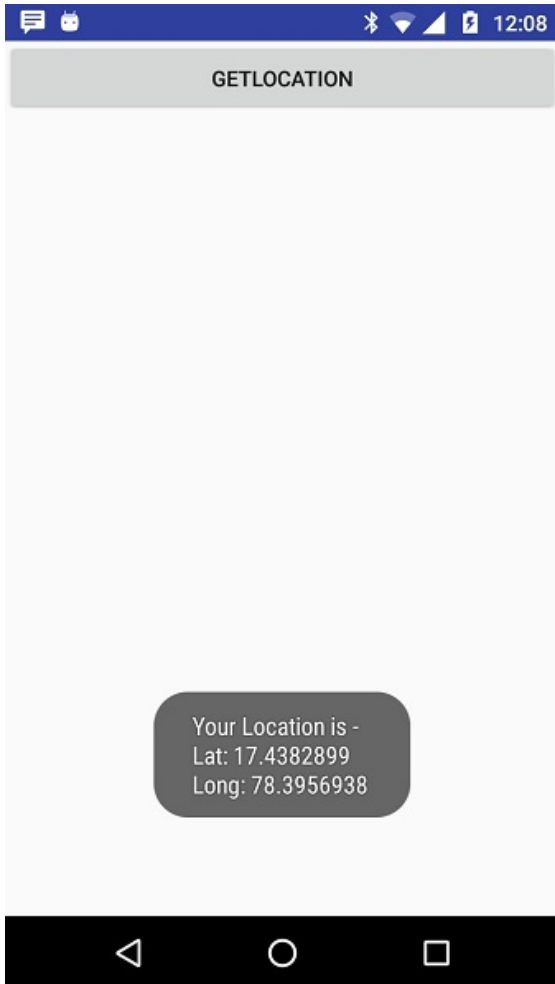
        <activity android:name = ".MainActivity">
            <intent-filter>
                <action android:name = "android.intent.action.MAIN" />

                <category android:name = "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Let's try to run your **Tutorialspoint** application. I assume that, 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. Before starting your application, Android studio installer will display following window to select an option where you want to run your Android application.



Now to see location select Get Location Button which will display location information as follows –



Experiment-8: Implement an application that writes data to the SD card.

Solution:

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:layout_margin="20dp"
    android:orientation="vertical">

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:singleLine="true"
        android:textSize="30dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
```

```
    android:text="Write Data"
    android:textSize="30dp" />
```

```
<Button
    android:id="@+id/button2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:text="Read data"
    android:textSize="30dp" />
```

```
<Button
    android:id="@+id/button3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:text="Clear"
    android:textSize="30dp" />
```

```
</LinearLayout>
```

Code for AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.exno9" >
```

```
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-
permission>
```

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="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>
```

Code for MainActivity.java:

```
package com.example.exno9;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
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.FileOutputStream;
import java.io.InputStreamReader;

public class MainActivity extends AppCompatActivity
{
    EditText e1;
    Button write, read, clear;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        e1= (EditText) findViewById(R.id.editText);
        write= (Button) findViewById(R.id.button);
        read= (Button) findViewById(R.id.button2);
        clear= (Button) findViewById(R.id.button3);

        write.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                String message=e1.getText().toString();
                try
                {
                    File f=new File("/sdcard/myfile.txt");
                    f.createNewFile();
                    FileOutputStream fout=new FileOutputStream(f);
                    fout.write(message.getBytes());
                    fout.close();
                    Toast.makeText(getApplicationContext(),"Data Written in
SDCARD",Toast.LENGTH_LONG).show();
                }
                catch (Exception e)
                {
                    Toast.makeText(getApplicationContext(),e.getMessage(),Toast
.LENGTH_LONG).show();
                }
            }
        });

        read.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                String message;
                String buf = "";
                try
                {
                    File f = new File("/sdcard/myfile.txt");
                    FileInputStream fin = new FileInputStream(f);
                    BufferedReader br = new BufferedReader(new
InputStreamReader(fin));
                    while ((message = br.readLine()) != null)

```

```

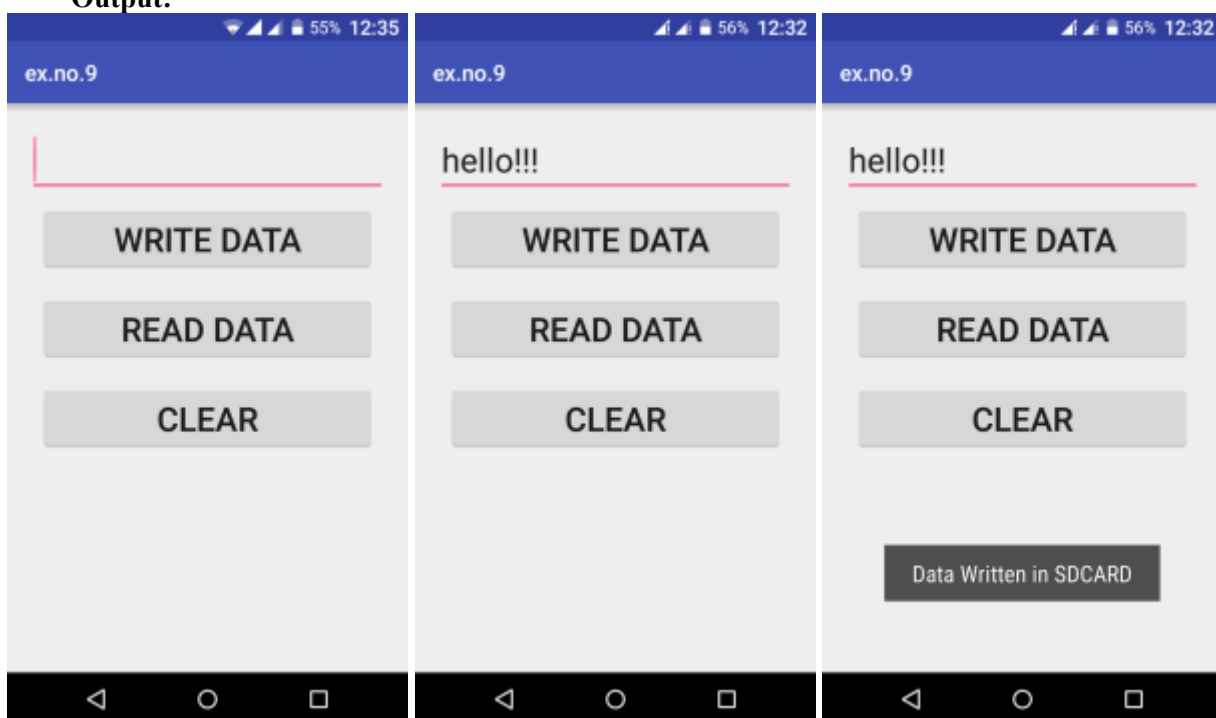
        {
            buf += message;
        }
        e1.setText(buf);
        br.close();
        fin.close();
        Toast.makeText(getApplicationContext(), "Data Recived from
SDCARD", Toast.LENGTH_LONG).show();
    }
    catch (Exception e)
    {
        Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
    }
    }
    });

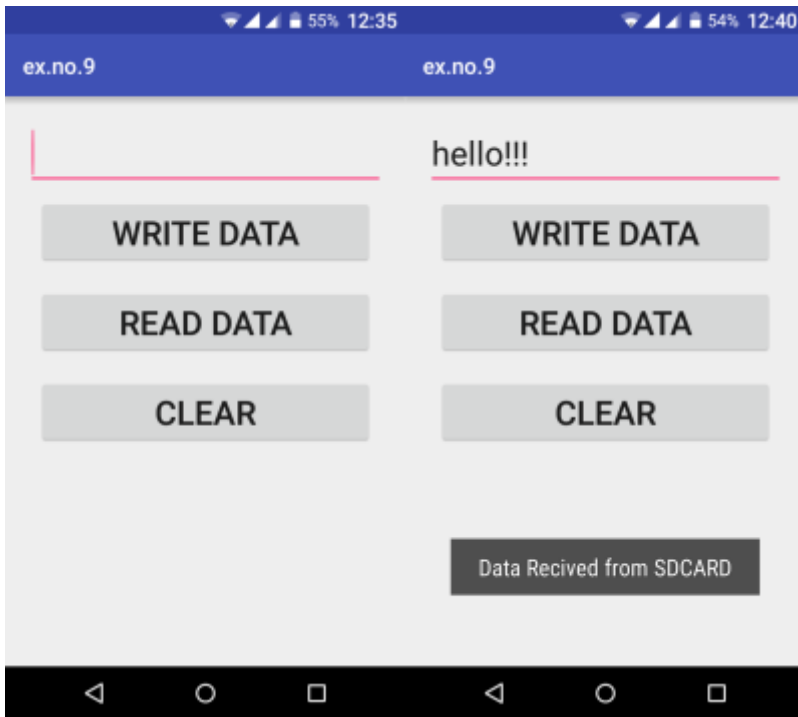
clear.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        e1.setText("");
    }
});
}
}

```

- So now the Coding part is also completed.
- Now run the application to see the output.

Output:





Experiment-9: Design a gaming application.

Solution:

Designing the Start Screen



- Above you can see the first screen of the game. It has a nice background image with two **ImageButton**s. You already downloaded the images used in this screen.
- As you can see it is a full screen activity. So to make your application full screen, you need to go **res->values->styles.xml** and modify it as the following code.

styles.xml

```
<resources>

<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <item name="windowNoTitle">true</item>
    <item name="windowActionBar">false</item>
    <item name="android:windowFullscreen">true</item>
    <item name="android:windowContentOverlay">@null</item>
</style>

</resources>
```

- Inside **activity_main.xml** write the following xml code.

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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@drawable/splash"
    tools:context="net.simplifiedcoding.simplegame.MainActivity">

    <ImageButton
        android:id="@+id/buttonPlay"
        android:background="@drawable/playnow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
android:layout_above="@+id/buttonScore"
android:layout_centerHorizontal="true" />
```

```
<ImageButton
    android:id="@+id/buttonScore"
    android:background="@drawable/highscore"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true" />
```

</RelativeLayout>

- When we tap the **Play Now** button our Game Activity will start.
- Now come inside **MainActivity.java** and write the following code.

MainActivity.java

```
package net.simplifiedcoding.simplegame;

import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.media.Image;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;

public class MainActivity extends AppCompatActivity implements View.OnClickListener{

    //image button
    private ImageButton buttonPlay;

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

        //setting the orientation to landscape
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

        //getting the button
        buttonPlay = (ImageButton) findViewById(R.id.buttonPlay);

        //adding a click listener
        buttonPlay.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {

        //starting game activity
        startActivity(new Intent(this, GameActivity.class));
    }
}
```

- Now you need to create a new activity named **GameActivity**. To create a new activity **right click on the package name -> new -> activity -> empty activity**

Building Game View

Now its time to build our Game View. We will be using [SurfaceView](#) for building our game view. Surfaceview provides a dedicated drawing surface.

- Create a new class named **GameView** and write the following code.

GameView.java

```
public class GameView extends SurfaceView implements Runnable {

    //boolean variable to track if the game is playing or not
    volatile boolean playing;
    //the game thread
    private Thread gameThread = null;

    //Class constructor
    public GameView(Context context) {
        super(context);
    }

    @Override
    public void run() {
        while (playing) {
            //to update the frame
            update();
            //to draw the frame
            draw();
            //to control
            control();
        }
    }

    private void update() {

    }

    private void draw() {

    }

    private void control() {
        try {
            gameThread.sleep(17);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public void pause() {
        //when the game is paused
        //setting the variable to false
        playing = false;
        try {
            //stopping the thread
            gameThread.join();
        }
    }
}
```

```

        } catch (InterruptedException e) {
        }
    }
    public void resume() {
//when the game is resumed
//starting the thread again
        playing = true;
        gameThread = new Thread(this);
        gameThread.start();
    }
}

```

- The above class is our **GameView** class. It is the actual game panel where we will play the game. The class is implementing **Runnable interface**. We have a **volatile boolean type variable running** that will track whether the game is running or not. After that we have our **gameThread**, it is the main game loop. Then we have the constructor to the class. We are not doing anything inside the constructor right now. Then we have the overridden method **run()**, here we are running a loop until the playing variable **running** is true. Inside the loop we are calling the following methods.
update() -> Here we will update the coordinate of our characters.
draw() -> Here we will draw the characters to the canvas.
control() -> This method will control the frames per seconds drawn. Here we are calling the delay method of Thread. And this is actually making our frame rate to around 60fps.
After these we have two more methods.
pause() -> To pause the game, we are stopping the **gameThread** here.
resume() -> To resume the game, here we are starting the **gameThread**.

Adding GameView to GameActivity

- After tapping the play now button we are launching the GameActivity. We will set our GameView to the content of this activity. So go to **GameActivity.java** and modify the code as below.

I will be adding the comments only above the new code added so that you can understand what is new in this code so that you can modify your code easily.

GameActivity.java

```

package net.simplifiedcoding.spacefighter;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class GameActivity extends AppCompatActivity {

    //declaring gameview
    private GameView gameView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //Initializing game view object
        gameView = new GameView(this);

        //adding it to contentview
        setContentView(gameView);
    }

    //pausing the game when activity is paused
    @Override
    protected void onPause() {
        super.onPause();
    }
}

```

```

        gameView.pause();
    }

    //running the game when activity is resumed
    @Override
    protected void onResume() {
        super.onResume();
        gameView.resume();
    }
}

```

Creating Player

- Create a new class Player inside your package and write the following code.
- package net.simplifiedcoding.spacefighter;
- import android.content.Context;
- import android.graphics.Bitmap;
- import android.graphics.BitmapFactory;
- /**
- * Created by Belal on 6/24/2016.
- */
- public class Player {
- //Bitmap to get character from image
- private Bitmap bitmap;
- //coordinates
- private int x;
- private int y;
-
- //motion speed of the character
- private int speed = 0;
-
- //constructor
- public Player(Context context) {
- x = 75;
- y = 50;
- speed = 1;
-
- //Getting bitmap from drawable resource
- bitmap = BitmapFactory.decodeResource(context.getResources(), R.drawable.player);
- }
-
- //Method to update coordinate of character
- public void update(){
- //updating x coordinate
- x++;
- }
-
- /**
- * These are getters you can generate it automatically!
- * right click on editor -> generate -> getters
- */
- public Bitmap getBitmap() {
- return bitmap;
- }
-
- public int getX() {

- return x;
- }
-
- public int getY() {
- return y;
- }
-
- public int getSpeed() {
- return speed;
- }
- } }

Drawing Player to GameView

- To draw the player to our GameView you need to come back to the GameView.java class and modify it as below.

- public class GameView extends SurfaceView implements Runnable {

-
- volatile boolean playing;
- private Thread gameThread = null;
-
- //adding the player to this class
- private Player player;
-
- //These objects will be used for drawing
- private Paint paint;
- private Canvas canvas;
- private SurfaceHolder surfaceHolder;
-

- public GameView(Context context) {
- super(context);
-
- //initializing player object
- player = new Player(context);
-
- //initializing drawing objects
- surfaceHolder = getHolder();
- paint = new Paint();
- }

-
- @Override
- public void run() {
- while (playing) {
- update();
- draw();
- control();
- }
- }

-
- private void update() {
- //updating player position
- player.update();
- }

-
- private void draw() {
- //checking if surface is valid
- if (surfaceHolder.getSurface().isValid()) {

- //locking the canvas
- canvas = surfaceHolder.lockCanvas();
- //drawing a background color for canvas
- canvas.drawColor(Color.BLACK);
- //Drawing the player
- canvas.drawBitmap(
- player.getBitmap(),
- player.getX(),
- player.getY(),
- paint);
- //Unlocking the canvas
- surfaceHolder.unlockCanvasAndPost(canvas);
- }
- }
-
- private void control() {
- try {
- gameThread.sleep(17);
- } catch (InterruptedException e) {
- e.printStackTrace();
- }
- }
-
- public void pause() {
- playing = false;
- try {
- gameThread.join();
- } catch (InterruptedException e) {
- }
- }
-
- public void resume() {
- playing = true;
- gameThread = new Thread(this);
- gameThread.start();
- }
- }

- Now try executing your application. Put your emulator in landscape mode. When you tap the play now button in the first activity you will see the moving Space Jet as shown below.



If you are getting the output as shown in the above image then Hurray!. You might have got the concept about drawing images in canvas using SurfaceView. And the movement is the magic of the coordinates. We are changing the x coordinate so it is moving horizontally ahead.

Adding Controls

We will now add control to the player's Space Jet. When the player will tap on the screen the Space Jet will boost up and after releasing the screen the Space Jet will boost down. The movement is inspired by the most popular game **Flappy Bird**.

To add this control we will need to detect touches of the screen. So let's begin.

- Come inside GameView.java file and override the method **onTouchEvent()**.
- **@Override**
- `public boolean onTouchEvent(MotionEvent motionEvent) {`
- `switch (motionEvent.getAction() & MotionEvent.ACTION_MASK) {`
- `case MotionEvent.ACTION_UP:`
- `//When the user presses on the screen`
- `//we will do something here`
- `break;`
- `case MotionEvent.ACTION_DOWN:`
- `//When the user releases the screen`
- `//do something here`
- `break;`
- `}`
- `return true;`
- `}`

Currently we need these two events only which are **ACTION_UP** and **ACTION_DOWN**. What we need

to do is we need to boost up the Space Jet on **ACTION_UP** and boost down the Space Jet on **ACTION_DOWN**.

Adding Booster to Space Jet

Now we will add the booster to our **Space Jet** so that our player can control the **Space Jet**. Follow these steps to do this.

- **Modify the code of `Player.java` file as follows.**

- `public class Player {`
- `private Bitmap bitmap;`
- `private int x;`
- `private int y;`
- `private int speed = 0;`
-
- `//boolean variable to track the ship is boosting or not`
- `private boolean boosting;`
-
- `//Gravity Value to add gravity effect on the ship`
- `private final int GRAVITY = -10;`
-
- `//Controlling Y coordinate so that ship won't go outside the screen`
- `private int maxY;`
- `private int minY;`
-
- `//Limit the bounds of the ship's speed`
- `private final int MIN_SPEED = 1;`
- `private final int MAX_SPEED = 20;`
-
- `public Player(Context context) {`
- `x = 75;`
- `y = 50;`
- `speed = 1;`
- `bitmap = BitmapFactory.decodeResource(context.getResources(), R.drawable.player);`
-
- `//setting the boosting value to false initially`
- `boosting = false;`
- `}`
-
-
- `//setting boosting true`
- `public void setBoosting() {`
- `boosting = true;`
- `}`
-
- `//setting boosting false`
- `public void stopBoosting() {`
- `boosting = false;`
- `}`
-
- `public void update() {`
- `//if the ship is boosting`
- `if (boosting) {`
- `//speeding up the ship`
- `speed += 2;`
- `} else {`
- `//slowing down if not boosting`
- `speed -= 5;`

```

•    }
•    //controlling the top speed
•    if (speed > MAX_SPEED) {
•        speed = MAX_SPEED;
•    }
•    //if the speed is less than min speed
•    //controlling it so that it won't stop completely
•    if (speed < MIN_SPEED) {
•        speed = MIN_SPEED;
•    }
•
•    //moving the ship down
•    y -= speed + GRAVITY;
•
•    //but controlling it also so that it won't go off the screen
•    if (y < minY) {
•        y = minY;
•    }
•    if (y > maxY) {
•        y = maxY;
•    }
• }
•
• public Bitmap getBitmap() {
•     return bitmap;
• }
•
• public int getX() {
•     return x;
• }
•
• public int getY() {
•     return y;
• }
•
• public int getSpeed() {
•     return speed;
• }
• }

```

We also need to detect the size of screen so go inside **GameActivity.java** file and add the following code inside **onCreate()**.

@Override

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
    //Getting display object
    Display display = getWindowManager().getDefaultDisplay();
```

```
    //Getting the screen resolution into point object
    Point size = new Point();
    display.getSize(size);
```

```
    //Initializing game view object
    //this time we are also passing the screen size to the GameView constructor
    gameView = new GameView(this, size.x, size.y);
```

```

        //adding it to contentview
        setContentView(gameView);
    }

```

In the above code we are passing screen size to **GameView** constructor, so we also need to change the **GameView** constructor. So change the **GameView** constructor as follow.

```

public GameView(Context context, int screenX, int screenY) {
    super(context);

    //initializing player object
    //this time also passing screen size to player constructor
    player = new Player(context, screenX, screenY);

    //initializing drawing objects
    surfaceHolder = getHolder();
    paint = new Paint();
}

```

In the above code you can see we are passing the screen size to player constructor. So again we also need to modify the **Player** class constructor.

```

public Player(Context context, int screenX, int screenY) {
    x = 75;
    y = 50;
    speed = 1;
    bitmap = BitmapFactory.decodeResource(context.getResources(), R.drawable.player);
    //calculating maxY
    maxY = screenY - bitmap.getHeight();
    //top edge's y point is 0 so min y will always be zero
    minY = 0;
    //setting the boosting value to false initially
    boosting = false;
}

```

Now to complete adding the boosters come inside **GameView.java** file and modify the **onTouchEvent()** as follows.

```

@Override
public boolean onTouchEvent(MotionEvent motionEvent) {
    switch (motionEvent.getAction() & MotionEvent.ACTION_MASK) {
        case MotionEvent.ACTION_UP:
            //stopping the boosting when screen is released
            player.stopBoosting();
            break;
        case MotionEvent.ACTION_DOWN:
            //boosting the space jet when screen is pressed
            player.setBoosting();
            break;
    }
    return true;
}

```

RUBRICS EVALUATION

Performance Criteria	Scale 1 (0-25%)	Scale 2 (26-50%)	Scale 3 (51-75%)	Scale 4 (76-100%)	Score (Numerical)
Understandability Ability to analyse Problem and Identify solution	Unable to understand the problem.	Able to understand the problem partially and unable to identify the solution	Able to understand the problem completely but unable to identify the solution	Able to understand the problem completely and able to provide alternative solution too.	
Logic Ability to specify Conditions & control flow that are appropriate for the problem domain.	Program logic is incorrect	Program logic is on the right track but has several errors	Program logic is mostly correct, but may contain an occasional boundary error or redundant or contradictory condition.	Program logic is correct, with no known boundary errors, and no redundant or contradictory conditions.	
Debugging Ability to execute /debug	Unable to execute program	Unable to debug several errors.	Able to execute program with several warnings.	Able to execute program completely	
Correctness Ability to code formulae and algorithms that reliably produce correct answers or appropriate results.	Program does not produce correct answers or appropriate results for most inputs.	Program approaches correct answers or appropriate results for most inputs, but can contain miscalculations in some cases.	Program produces correct answers or appropriate results for most inputs.	Program produces correct answers or appropriate results for all inputs tested.	
Completeness Ability to demonstrate and deliver on time.	Unable to explain the code.and the code was overdue.	Unable to explain the code and the code submission was late.	Able to explain code and the program was delivered within the due date.	Able to explain code and the program was delivered on time.	
TOTAL					