



Savitribai Phule Pune University

(From Academic Year 2021)

Section-I

T. Y. B. C. A. (Science)

SEMESTER VI BCA – 366

Lab Course – I

DSE IV Laboratory
(Android Programming)

Name: _____

Roll No: _____ Seat No _____

Academic Year: 20__ - 20__

From the Chairman's Desk

It gives me a great pleasure to present this workbook prepared by the Board of studies in Computer Applications.

The workbook has been prepared with the objectives of bringing uniformity in implementation of lab assignments across all affiliated colleges, act as a ready reference for both fast and slow learners and facilitate continuous assessment using clearly defined rubrics.

The workbook provides, for each of the assignments, the aims, pre-requisites, related theoretical concepts with suitable examples wherever necessary, guidelines for the faculty/lab administrator, instructions for the students to perform assignments and a set of exercises divided into three sets.

I am thankful to the Chief Editor and the entire team of editors appointed. I am also thankful to the members of BOS. I thank everyone who have contributed directly or indirectly for the preparation of the workbook.

Constructive criticism is welcome and to be communicated to the Chief Editor. Affiliated colleges are requested to collect feedbacks from the students for the further improvements.

I am thankful to Hon. Vice Chancellor of Savitribai Phule Pune University Prof. Dr. Nitin Karmalkar and the Dean of Faculty of Science and Technology Prof. Dr. M G Chaskar for their support and guidance.

Prof. Dr. S. S. Sane

Chairman, BOS in Computer Applications

SPPU, Pune

Chairperson and Editor:

**Mr. Kamil Khan Ajmal Khan Abeda Inamdar Senior College of Arts, Science
and Commerce, Pune**

Prepared and Compiled by:

Sr. No.	Assignment Name	Teacher Name	College
1	Introduction to Android	Mr. Kamil Khan	Abeda Inamdar Senior College, Pune
2	Activities, Fragments and Intents	Mr. Deepak Kumbhar	PES Modern College, Pune
3	Android User Interface	Ms. Rajshree Nehe	
4	Designing User Interface with Views	Mr. Mohsin Tamboli	Abeda Inamdar Senior College, Pune
5	Databases-SQLite, Messaging and E-mail	Mr. Mohsin Tamboli	Abeda Inamdar Senior College, Pune
6	Location-Based Services and Google Map	Mr. Kamil Khan	Abeda Inamdar Senior College, Pune

Reviewed By:**1. Dr. Madhukar Shelar**

KTHM College, Nashik 02

2. Dr. Pallawi Bulakh

PES Modern College, of Arts, Science and Commerce, Ganeshkhind Pune 16

3. Dr. Sayyad Razak

Ahmednagar College, Ahmednagar

Introduction

1. About the Workbook:

This workbook is intended to be used by TYBCA (Science) students for the Android Programming Assignments in Semester–VI. This workbook is designed by considering all the practical concepts / topics mentioned in syllabus.

2. The objectives of this Workbook are:

- 1) Defining the scope of the course.
- 2) To bring the uniformity in the practical conduction and implementation in all colleges affiliated to SPPU.
- 3) To have continuous assessment of the course and students.
- 4) Providing ready reference for the students during practical implementation.
- 5) Provide more options to students so that they can have good practice before facing the examination.
- 6) Catering to the demand of slow and fast learners and accordingly providing the practice assignments to them.

3. How to use this Workbook:

The workbook is divided into two sections. Section-I is related to DS assignments

The Section-I (Android Programming) is divided into six assignments. Each Android Programming assignment has several SET. It is mandatory for students to complete all the SET in given slot.

4. Instructions to the students:

Please read the following instructions carefully and follow them.

- Students are expected to carry this workbook every time they come to the lab for practical.
- Students should prepare for the assignment by reading the relevant material which is mentioned in ready reference.
- Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However, student should spend additional hours in Lab and at home to cover all workbook assignments if needed.
- Students will be assessed for each assignment on a scale from 0 to 5

Not done	0
Incomplete	1

Late Complete	2
Needs improvement	3
Complete	4
Well Done	5

5. Instruction to the Instructors:

- Make sure that students should follow above instructions.
- Explain the assignment and related concepts using white board if required or by demonstrating the software.
- Give specific input to fill the blanks in queries which can vary from student to student.
- Evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
- The value should also be entered on assignment completion page of the respective Lab course.

6. Instructions to the Lab administrator:

You have to ensure appropriate hardware and software is made available to each student. The operating system and software requirements on server side and also client side areas given below:

- Server and Client Side-(Operating System) Linux/Windows
- RAM : Minimum 8 GB
- JDK
- Android Studio

Table of Contents

Assign. No	Topics for the Assignments	Page No.
1	Introduction to Android	
2	Activities, Fragments and Intents	
3	Android User Interface	
4	Designing User Interface with Views	
5	Databases-SQLite, Messaging and E-mail	
6	Location-Based Services and Google Map	

Assignment Completion Sheet

Assign. No.	Topics for the Assignments	Marks (5)	Teacher Signature
1	Introduction to Android		
2	Activities, Fragments and Intents		
3	Android User Interface		
4	Designing User Interface with Views, Pictures & Menus		
5	Databases-SQLite, Messaging and E-mail		
6	Location-Based Services and Google Map		
Total Marks (Out of 30)			
Total Marks (Out of 15)			



CERTIFICATE

This is to certify that

Mr. / Ms. _____

has successfully completed the course work for BCA-366: DSE

IV Laboratory Android Programming in the year _____

and his/her Seat Number is _____.

Instructor

H.O.D / Coordinator

Internal Examiner

External Examiner

Objectives

- Study Android Studio installation.
- Create basic android application.

Reading

- You should read the following topics before starting this exercise:
 1. Java Installation
 2. Android Studio
 3. Android Directory Structure

Ready Reference

- Android Studio uses the Java tool chain to build, so you need to make sure that you have the Java Development Kit (JDK) installed on your computer before you start using Android Studio.
- It's quite possible that you already have the JDK installed on your computer, particularly if you're a seasoned Android or Java developer.
- If you already have the JDK installed on your computer, and you're running JDK version 1.6 or higher, then you can skip this section. However, you may want to download, install, and configure the latest JDK anyway.
- You can download the JDK from the following Oracle site:

www.oracle.com/technetwork/java/javase/downloads/index.html



Figure 1.1 : Installation Wizard for the JDK on Windows

Make a note of where you are installing your JDK. Follow the prompts until the installation is complete. If prompted to install the Java Runtime Edition (JRE), choose the same directory where you installed the JDK.



Figure 1.2: Select the JDK installation directory

Configuring Environmental Variables on Windows

This section shows you how to configure Windows so that the JDK is found by Android Studio. On a computer running Windows, hold down the Windows key and press the Pause key to open the System window. Click the Advanced System Settings option, shown in Figure -.

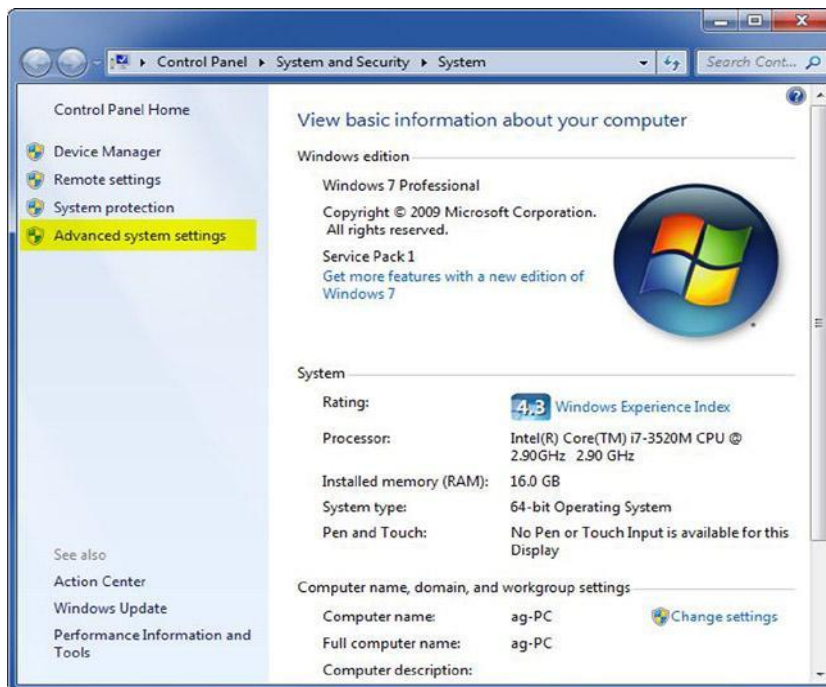


Figure 1.3: Windows System window

- Click the Environmental Variables button, In the System Variables list along the bottom, shown in Figure 1-7, navigate to the JAVA_HOME item. If the JAVA_HOME item does not exist, click New to create it. Otherwise, click Edit.
- Clicking either New or Edit displays a dialog box. Be sure to type JAVA_HOME in the Variable Name field. In the Variable Value field, type the location where you installed the JDK earlier (less any trailing slashes). Now click OK.

Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Downloading Android Studio is straightforward. Point your browser to this site:

<https://developer.android.com/studio>

Now click the large green Download Android Studio for your OS button,

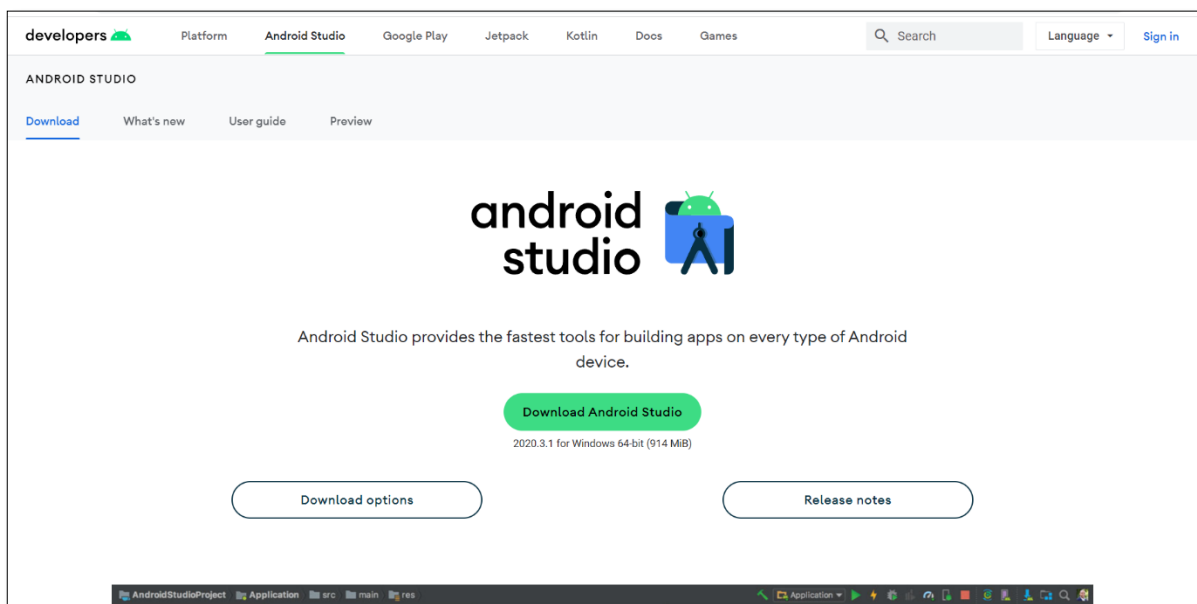


Figure 1.4: Download Android Studio

After the Installation Wizard begins, move through its screens by clicking the Next buttons until you reach the Choose Components screen. There, select all the component check boxes, shown in Figure. Then click Next. Agree to the terms and conditions once again. When you reach the Configuration Settings: Install Locations screen, shown in Figure 1, select the locations for Android Studio and the Android SDK. To be consistent, we chose to install Android Studio in C:\Java\astudio\ and the Android SDK in C:\Java\asdk\.

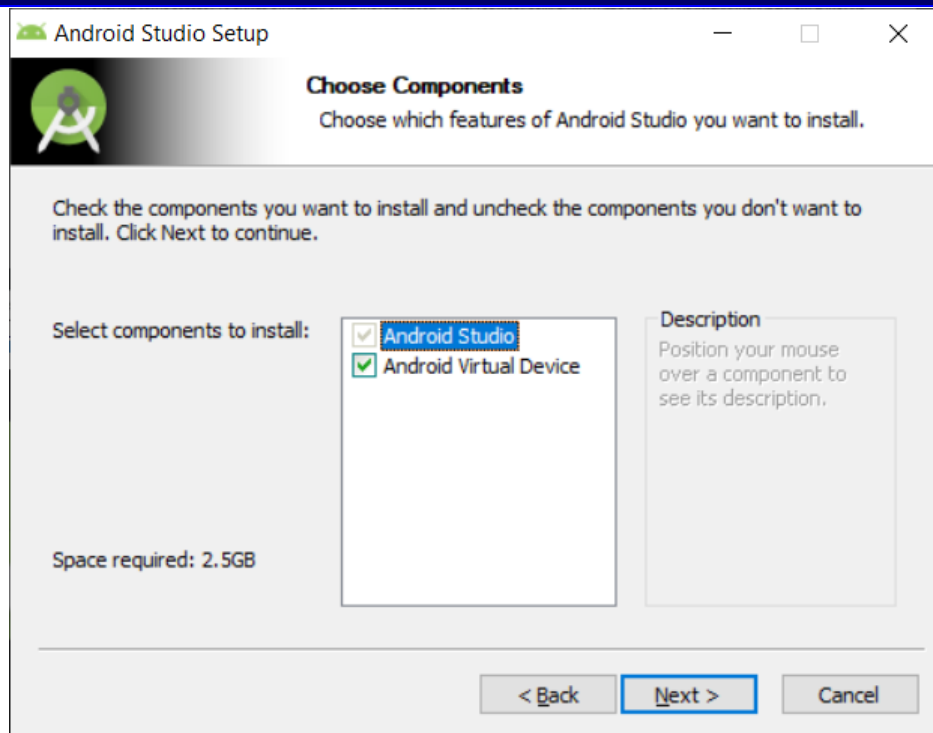


Figure 1.5 : Choose components

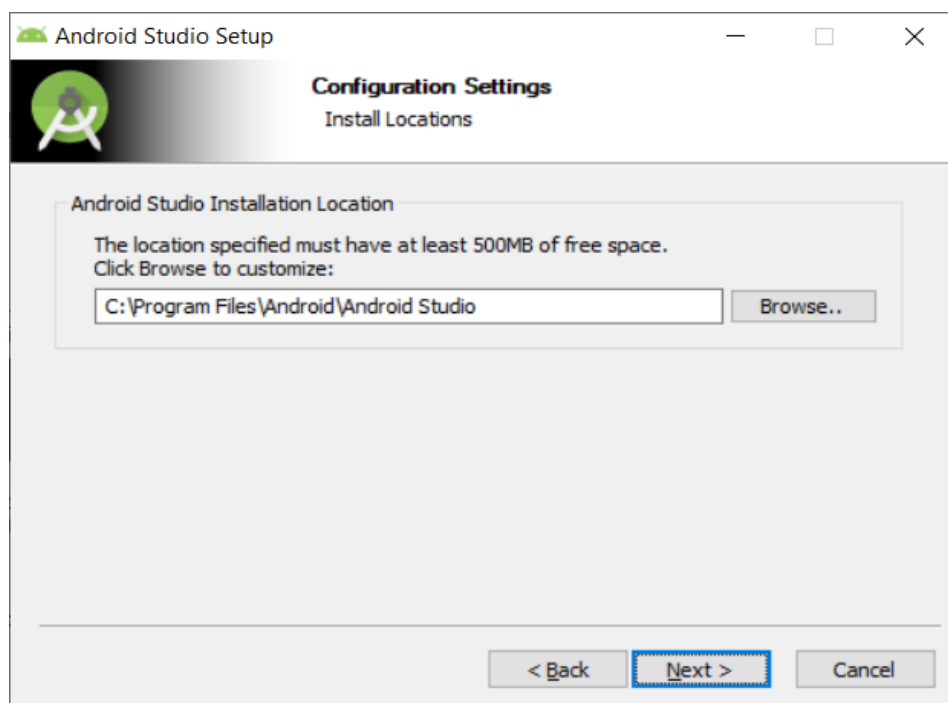


Figure 1.6: Select locations for Android Studio and the SDK

Click through several Next buttons as you install both Android Studio and the Android SDK. You should eventually arrive at the Completing the Android Studio Setup screen, shown in Figure. The Start Android Studio check box enables Android Studio to launch after you click Finish. Make sure the check box is selected, and then go ahead and click Finish, and Android Studio will launch. Please note that from here on out, you will need to navigate to either the desktop icon or the Start menu to launch Android Studio.

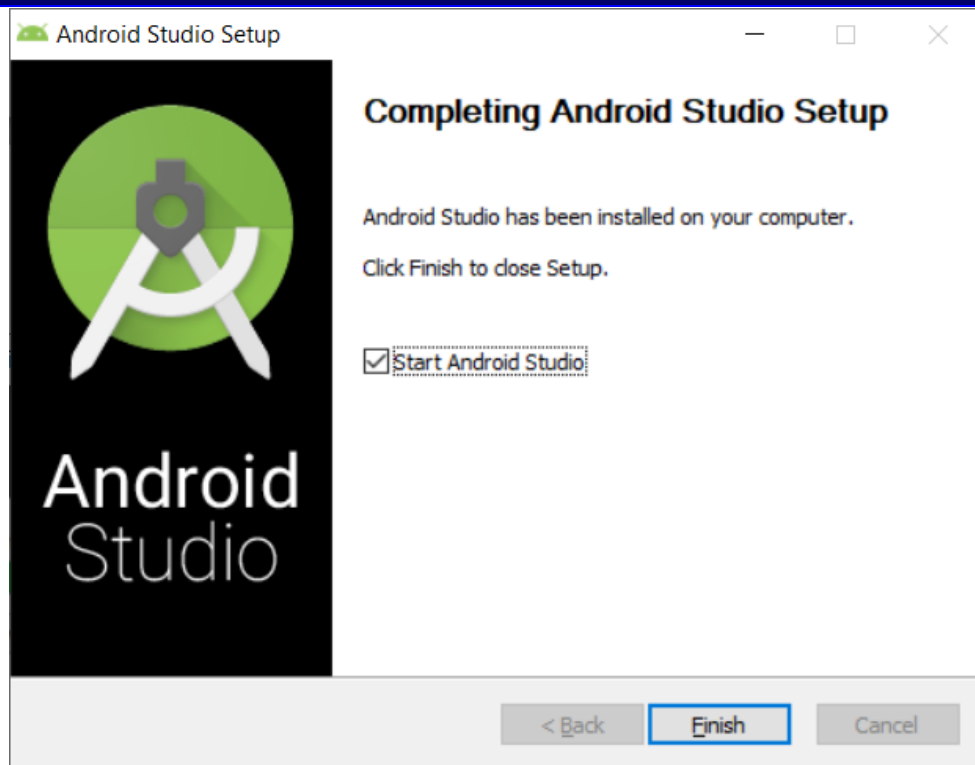
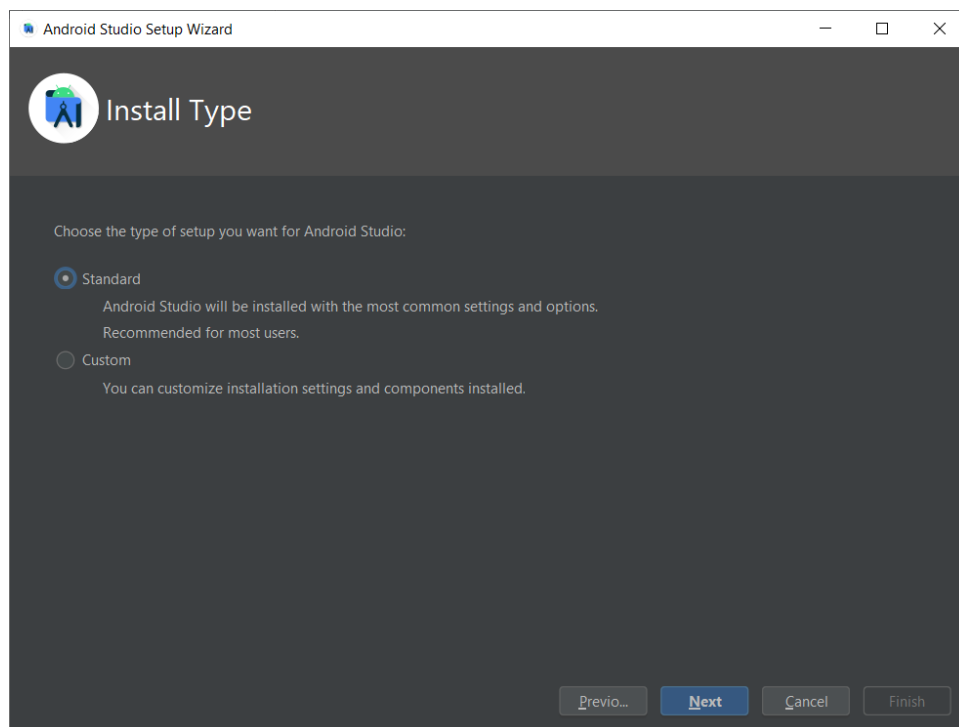


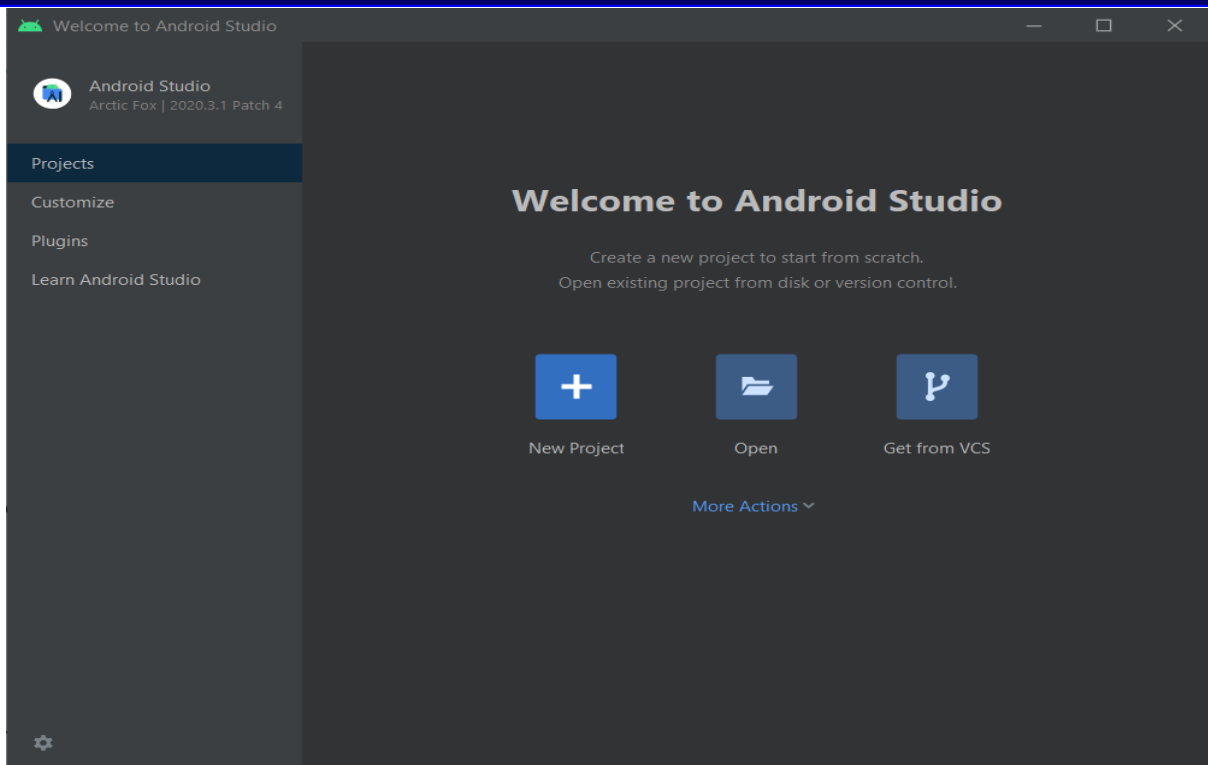
Figure 1.7: Completing the Android Studio setup

Now that Android Studio 2 is installed, you need to adjust the settings and options using the following steps:

Click Continue at the Welcome screen and choose Standard from the Install Type selection screen shown in Figure. Click Next to continue.



Click Finish on the Verify Settings screen, and Android Studio 2 finalizes the setup process. You know the process is complete when you are greeted with the Welcome to Android Studio screen (see Figure).

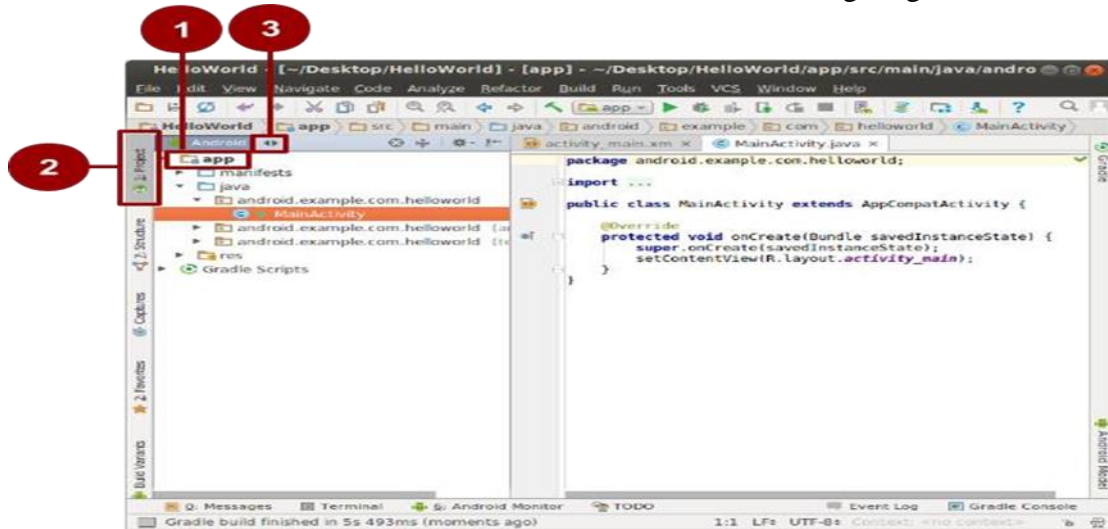


Now that Android Studio is set up, it's time to install the latest and greatest Android SDK.

2. Create "Hello World" application

1. Launch Android Studio if it is not already opened.
2. In the main Welcome to Android Studio window, click "Start a new Android Studio project".
3. In the New Project window, give your application an Application Name, such as "Hello World".
4. Verify the Project location, or choose a different directory for storing your project.
5. Choose a unique Company Domain.
 - Apps published to the Google Play Store must have a unique package name. Since domains are unique, prepending your app's name with your or your company's domain name is going to result in a unique package name.
 - If you are not planning to publish your app, you can accept the default example domain. Be aware that changing the package name of your app later is extra work.
6. Verify that the default Project location is where you want to store your Hello World app and other Android Studio projects, or change it to your preferred directory. Click Next.
7. On the Target Android Devices screen, "Phone and Tablet" should be selected. And you should ensure that API 15: Android 4.0.3 IceCreamSandwich is set as the Minimum SDK. (Fix this if necessary.)
 - At the writing of this book, choosing this API level makes your "Hello World" app compatible with 97% of Android devices active on the Google Play Store.
 - These are the settings used by the examples in this book.
8. Click Next.
9. If your project requires additional components for your chosen target SDK, Android Studio will install them automatically. Click Next.
10. Customize the Activity window. Every app needs at least one activity. An activity represents a single screen with a user interface and Android Studio provides templates to help you get started. For the Hello World project, choose the simplest template (as of this writing, the "Empty Activity" project template is the simplest template) available.

11. It is a common practice to call your main activity MainActivity. This is not a requirement.
12. Make sure the Generate Layout file box is checked (if visible).
13. Make sure the Backwards Compatibility (App Compat) box is checked.
14. Leave the Layout Name as activity_main. It is customary to name layouts after the activity they belong to. Accept the defaults and click Finish.
15. Builds your project with Gradle (this may take a few moments). Android Studio uses Gradle as its build system. See the Configure your build developer page for more information.
16. Opens the code editor with your project. And it displays a tip of the day.
- Android Studio offers many keyboard shortcuts, and reading the tips is a great way to learn them over time.
17. The Android Studio window should look similar to the following diagram:



18. You can look at the hierarchy of the files for your app in multiple ways.
 - Click on the Hello World folder to expand the hierarchy of files (1),
 - Click on Project (2).
 - Click on the Android menu (3).
 - Explore the different view options for your project.

3.Explore the project structure

3.1 Explore the project structure and layout

In the **Project > Android** view of your previous task, there are three top-level folders below your **app** folder: **manifests**, **java**, and **res**.

1. Expand the **manifests** folder.

This folder contains **AndroidManifest.xml**. This file describes all of the components of your Android app and is read by the Android run-time system when your program is executed.

2. Expand the **java** folder. All your Java language files are organized in this folder. The **java** folder contains three subfolders:

- **com.example.hello.helloworld (or the domain name you have specified):** All the files for a package are in a folder named after the package. For your Hello World application, there is one package and it only contains MainActivity.java (the file extension may be omitted in the Project view).
- **com.example.hello.helloworld(androidTest):** This folder is for your instrumented

tests, and starts out with a skeleton test file.

- **com.example.hello.helloworld(test):** This folder is for your unit tests and starts out with an automatically created skeleton unit test file.
3. Expand the **res** folder. This folder contains all the resources for your app, including images, layout files, strings, icons, and styling. It includes these subfolders:
- **drawable:** Store all your app's images in this folder.
 - **layout:** Every activity has at least one layout file that describes the UI in XML. For Hello World, this folder contains `activity_main.xml`.
 - **mipmap:** Store your launcher icons in this folder. There is a sub-folder for each supported screen density. Android uses the screen density, that is, the number of pixels per inch to determine the required image resolution. Android groups all actual screen densities into generalized densities, such as medium (mdpi), high (hdpi), or extra-extra-extra-high (xxxhdpi). The `ic_launcher.png` folder contains the default launcher icons for all the densities supported by your app.
 - **values:** Instead of hardcoding values like strings, dimensions, and colors in your XML and Java files, it is best practice to define them in their respective values file. This makes it easier to change and be consistent across your app.
4. Expand the **values** subfolder within the res folder. It includes these subfolders:
- **colors.xml:** Shows the default colors for your chosen theme, and you can add your own colors or change them based on your app's requirements.
 - **dimens.xml:** Store the sizes of views and objects for different resolutions.
 - **strings.xml:** Create resources for all your strings. This makes it easy to translate them to other languages.
 - **styles.xml:** All the styles for your app and theme go here. Styles help give your app a consistent look for all UI elements.

3.1. The Gradle build system

Android Studio uses Gradle as its build system. As you progress through these practicals, you will learn more about gradle and what you need to build and run your apps.

1. Expand the **Gradle Scripts** folder. This folder contains all the files needed by the build system.
2. Look for the **build.gradle(Module:app)** file. When you are adding app-specific dependencies, such as using additional libraries, they go into this file.

4. Create a virtual device (emulator)

In this task, you will use the Android Virtual Device (AVD) manager to create a virtual device or emulator that simulates the configuration for a particular type of Android device.


Using the AVD Manager, you define the hardware characteristics of a device and its API level, and save it as a virtual device configuration.

When you start the Android emulator, it reads a specified configuration and creates an emulated device that behaves exactly like a physical version of that device, but it resides on your computer.

Why: With virtual devices, you can test your apps on different devices (tablets, phones) with different API levels to make sure it looks good and works for most users. You do not need to depend on having a physical device available for app development.

4.1 Create a virtual device

In order to run an emulator on your computer, you have to create a configuration that describes the virtual device.

1. In Android Studio, select Tools > Android > AVD Manager, or click the AVD Manager icon  in the toolbar.
2. Click the **+Create Virtual Device....** (If you have created a virtual device before, the window shows all of your existing devices and the button is at the bottom.)

The Select Hardware screen appears showing a list of preconfigured hardware devices. For each device, the table shows its diagonal display size (Size), screen resolution in pixels (Resolution), and pixel density (Density).

For the Nexus 5 device, the pixel density is xxhdpi, which means your app uses the launcher icons in the xxhdpi folder of the mipmap folder. Likewise, your app will use layouts and drawables from folders defined for that density as well.

3. Choose the Nexus 5 hardware device and click **Next**.
4. On the **System Image** screen, from the **Recommended** tab, choose which version of the Android system to run on the virtual device. You can select the latest system image.


There are many more versions available than shown in the Recommended tab. Look at the **x86 Images** and **Other Images** tabs to see them.

5. If a **Download** link is visible next to a system image version, it is not installed yet, and you need to download it. If necessary, click the link to start the download, and click **Finish** when it's done.
6. On **System Image** screen, choose a system image and click **Next**. Verify your configuration, and click **Finish**. (If the **Your Android Devices** AVD Manager window stays open, you can go ahead and close it.)

5. Run your app on an emulator

In this task, you will finally run your Hello World app.

5.1 Run your app on an emulator

1. In Android Studio, select **Run > Run app** or click the **Run icon**  in the toolbar.
2. In the **Select Deployment Target** window, under **Available Emulators**, select **Nexus 5 API 23** and click **OK**.

You should see the Hello World app as shown in the following screenshot.



Figure: Display Output on Emulator

Set A

1. Create android application to change Font Size, Color and Font Family of String.
2. Setup Android virtual device on your machine.
3. Execute the Hello World Application on Physical Device.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well done []

Objectives -:

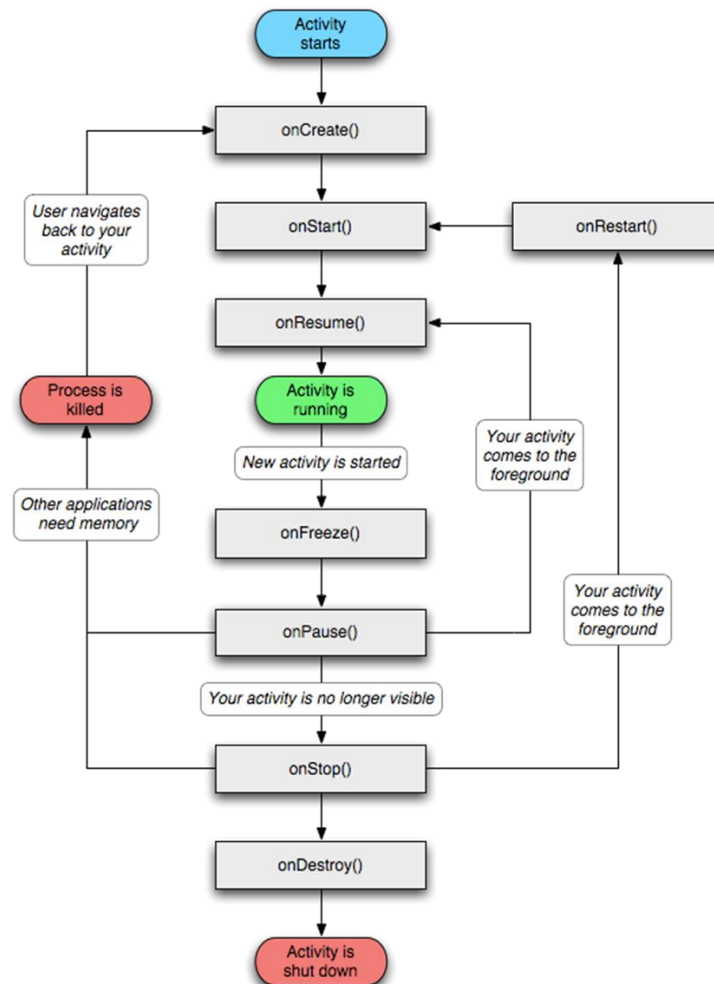
- To study how to use Activities, Fragments and Intents in your application.
- To study how to link Activities and interaction between Fragments.

Reading-:

- You should read the following topics before starting this exercise:
 1. Activity, Activity Lifecycle
 2. Linking Activities using Intent.
 3. Fragment and Fragment Lifecycle.

Ready Reference-:

1. **Activity-:** An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View).
2. **Activity Lifecycle-:** Activities in the system are managed as an activity stack. When a new activity is started, it is placed on the top of the stack and becomes the running activity - the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.



The lifecycle is shown diagrammatically in Figure 2-1.

Your activity monitors and reacts to these events by instantiating methods that override the Activity class methods for each event:

onCreate

Called when your activity is first created. This is the place you normally create your views, open any persistent datafiles your activity needs to use, and in general initialize your activity. When calling onCreate, the Android framework is passed a Bundle object that contains any activity state saved from when the activity ran before.

onStart

Called just before your activity becomes visible on the screen. Once onStart completes, if your activity can become the foreground activity on the screen, control will transfer to onResume. If the activity cannot become the foreground activity for some reason, control transfers to the onStop method.

onResume

Called right after onStart if your activity is the foreground activity on the screen. At this point your activity is running and interacting with the user. You are receiving keyboard and touch inputs, and the screen is displaying your user interface. onResume is also called if your activity loses the foreground to another activity, and that activity eventually exits, popping your activity back to the foreground. This is where your activity would start (or resume) doing things that are needed to update the user interface (receiving location updates or running an animation, for example).

onPause

Called when Android is just about to resume a different activity, giving that activity the foreground. At this point your activity will no longer have access to the screen, so you should stop doing things that consume battery and CPU cycles unnecessarily. If you are running an animation, no one is going to be able to see it, so you might as well suspend it until you get the screen back. Your activity needs to take advantage of this method to store any state that you will need in case your activity gains the foreground again—and it is not guaranteed that your activity will resume.

onStop

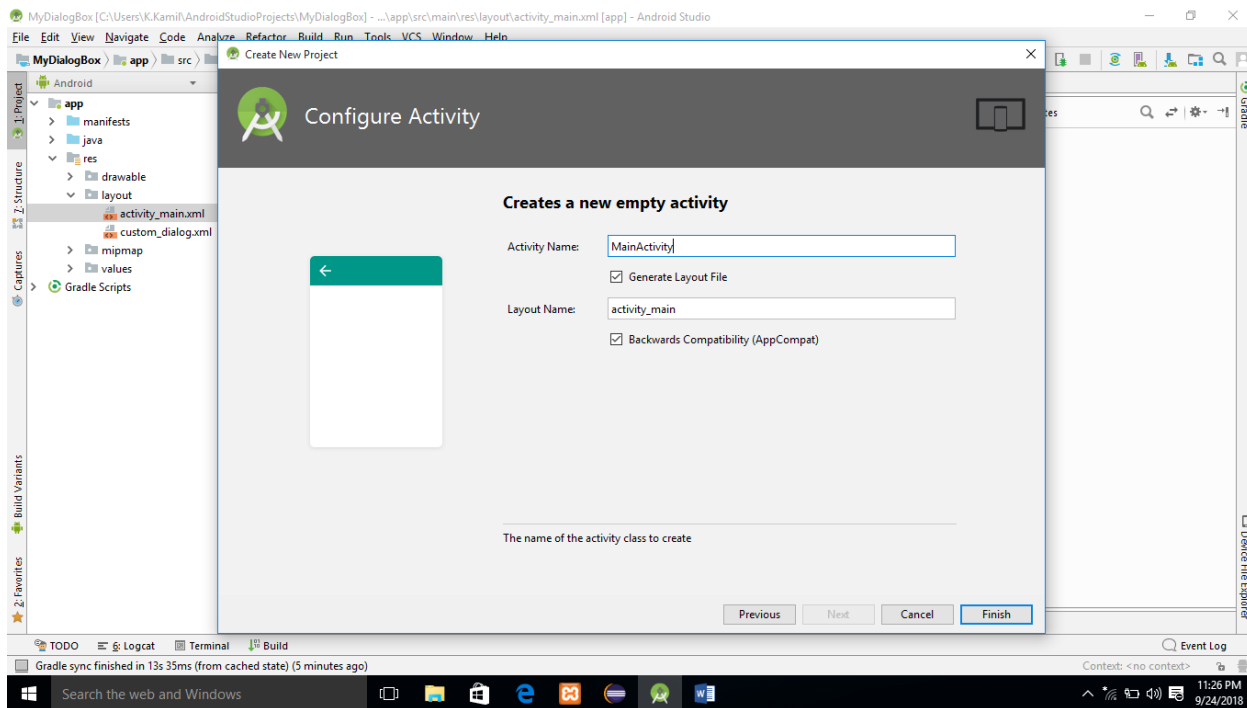
Called when your activity is no longer visible, either because another activity has taken the foreground or because your activity is being destroyed.

onDestroy

The last chance for your activity to do any processing before it is destroyed. Normally you'd get to this point because the activity is done and the framework called its finish method. But as mentioned earlier, the method might be called because Android has decided it needs the resources your activity is consuming.

The best way to understand the various stages experienced by an activity is to in android studio select File->New->create a new project, implement the various events, and then subject the activity to various user interactions.

Step 1:Using Android Studio, create a new Android project and name it as shown in Figure 2.



Step 2: In the MainActivity.java file, add the following statements in bold

```
package com.example.kkamil.learnfromkamil;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
  
public class MainActivity extends AppCompatActivity {  
    String tag="Events";  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Log.d(tag,"In on Create Event");  
    }  
  
    public void onStart()  
    {  
        super.onStart();  
        Log.d(tag, "In the onStart() event");  
    }  
  
    public void onRestart()  
    {
```

```
super.onRestart();
Log.d(tag, "In the onRestart() event");
}
```

```
public void onResume()
{
super.onResume();
Log.d(tag, "In the onResume() event");
}
```

```
public void onPause()
{
super.onPause();
Log.d(tag, "In the onPause() event");
}
```

```
public void onStop()
{
super.onStop();
Log.d(tag, "In the onStop() event");
}
```

```
public void onDestroy()
{
super.onDestroy();
Log.d(tag, "In the onDestroy() event");
}
}
```

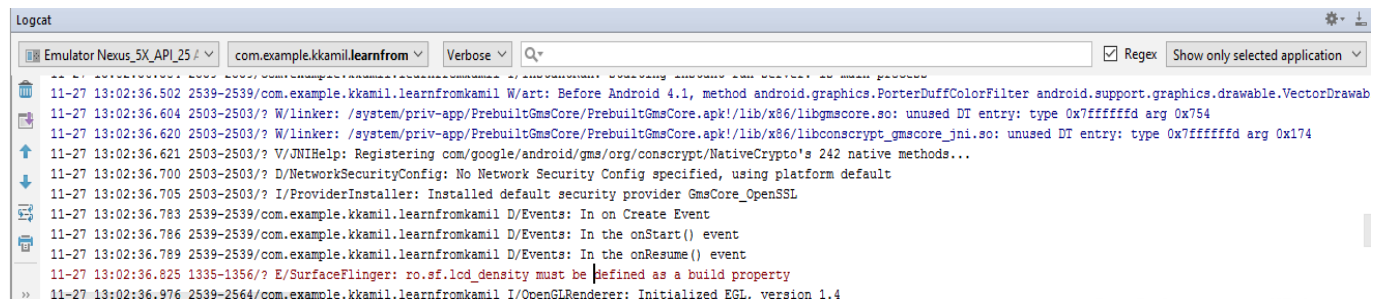
Step 3: Press F11 to debug the application on the Android Emulator

Step 4: When the activity is first loaded, you should see the following in the LogCat window (click on the Debug perspective; see also Figure):

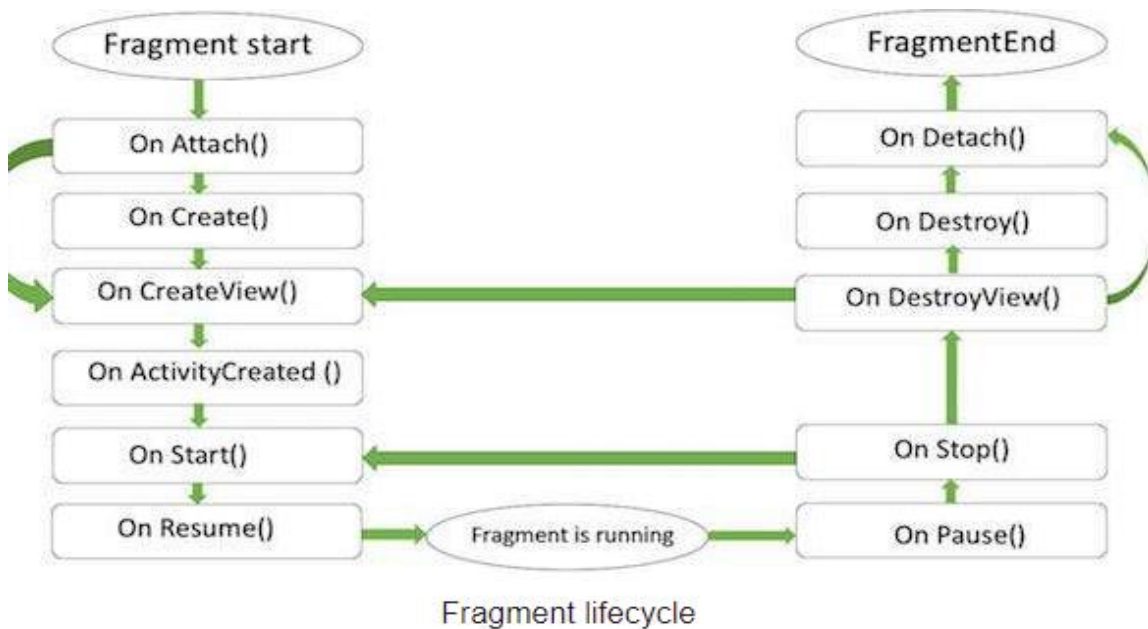
11-27 13:02:36.783 2539-2539/com.example.kkamil.learnfromkamil D/Events: In onCreate() event

11-27 13:02:36.786 2539-2539/com.example.kkamil.learnfromkamil D/Events: In the onStart()event

11-27 13:02:36.789 2539-2539/com.example.kkamil.learnfromkamil D/Events: In the onResume()

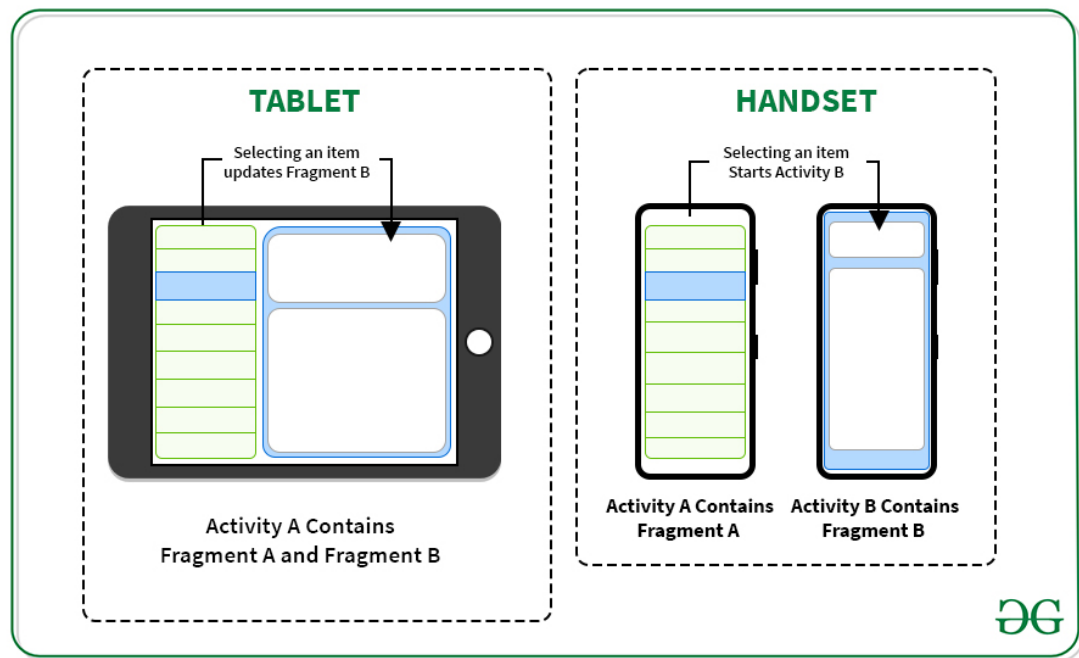


2.Fragments :



Fragment is a piece of an activity which enable more modular activity design. It will not be wrong if we say, a fragment is a kind of **sub-activity**.

Public Methods	Description
void on Attach (Activity activity)	This method is used get in this method a reference to the activity which uses the fragment for further initialization work.
void onCreate (Bundle savedInstanceState)	The system calls this method when creating the fragment.
View onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)	You should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.
void onActivityCreated (Bundle savedInstanceState)	The system calls this callback when it's time for the fragment to draw its user interface for the first time.
void onStart()	This method is called when the host activity is created. In this method you can instantiate objects which require Context object.
void onResume()	This method is called once the fragment gets visible.
void onPause()	Fragment becomes active.
void onStop()	The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session.
void onDestroyView()	Fragment going to be stopped by calling onStop()
void onDestroy()	Fragment view will destroy after call this method
	This method is called to do final cleanup of the fragment's state but Not guaranteed to be called by the Android platform.



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

Method	Description
Context.startActivity()	The Intent object is passed to this method to launch a new activity or get an existing activity to do something new
Context.startService()	The Intent object is passed to this method to initiate a service or deliver new instructions to an ongoing service.
Context.sendBroadcast()	The Intent object is passed to this method to deliver the message to all interested broadcast receivers.

Example:

This example shows how you can open a URL programmatically in the built-in web browser rather than within your application. This allows your app to open up a webpage without the need to include the INTERNET permission in your manifest file.

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Uri uri = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    // Verify that the intent will resolve to an activity
    if (intent.resolveActivity(getPackageManager()) != null) {
        // Here we use an intent without a Chooser unlike the next example
        startActivity(intent);
    }
}
```

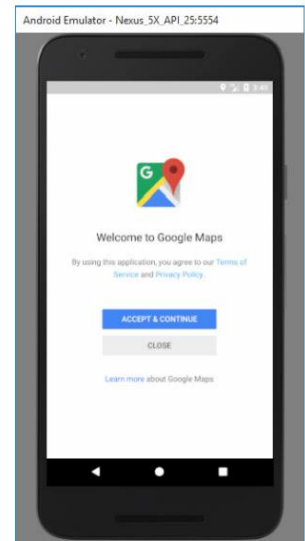

- **Types Of Intent :-**

- 1) **Implicit Intent :** 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 application.

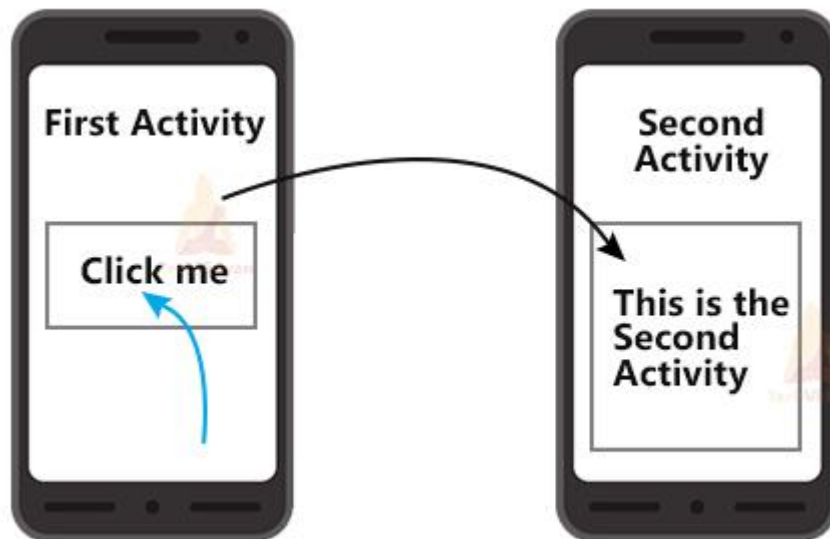
Example: Open Google MAP using Implicit Intent

```
b3 = (Button) findViewById(R.id.button3);
b3.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View arg0){
        Intent i = new
        Intent(Intent.ACTION_VIEW, Uri.parse("geo:37.827500,-
        122.481670"));
        startActivity(i); }

});
```



- 2) **Explicit Intent:** 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



Above code will give result as shown below:

Example:

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

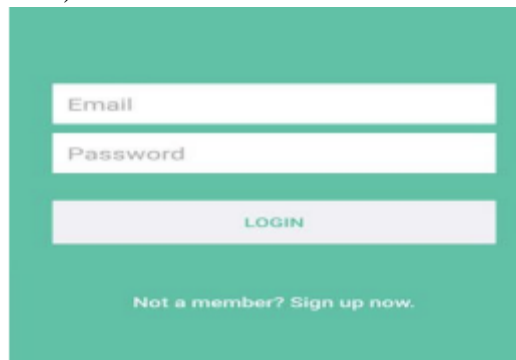
Lab Assignments: -

SET A

1. Create a Simple Application Which Shows Life Cycle of Activity.
2. Create a Simple Application Which Send —Hello! message from one activity to another with help of Button (Use Intent).
3. Create Simple application to display details of selected list item on Second Activity (use Fragmentation).

SET B

1. Create simple application with Login Screen. On successful login, gives message go to next Activity (Without Using Database).

A login screen with a teal background. It features three white input fields: the first is labeled 'Email', the second is labeled 'Password', and the third is a button labeled 'LOGIN'. Below the button, there is a link that says 'Not a member? Sign up now.'

2. Create First Activity to accept information like Student First Name, Middle Name, Last Name, Date of birth, Address, Email ID and display all information on Second Activity when user click on Submit button.

SET C

1. Design Following Screens Using Intents. On second activity take Button. On clicking it, it should Show Information of profile on Third activity. (Without Using Database)



Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Objective

- Study how to interact with mobile hardware.
- Study how to create user interface in Android.

Reading

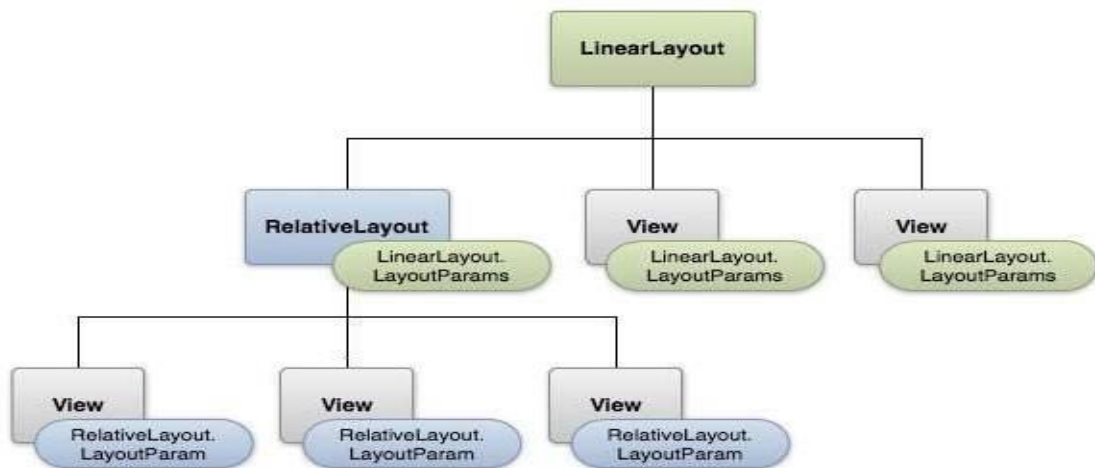
- You should read the following topics before starting this exercise:
 1. Views and ViewGroups.
 2. Orientation of Screen.
 3. Different layouts available in android.
 4. Understanding concept of screen

Ready Reference**Views and ViewGroup**

Views - View is a basic building block of UI (User Interface) in android. A view is a small rectangular box that responds to user inputs. Eg: EditText, Button, CheckBox, etc.

Views Groups - ViewGroup is an invisible container of other views (child views) and other ViewGroup. Eg: LinearLayout is a ViewGroup that can contain other views in it

At third level we have different layouts which are subclasses of View Group class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using View/View Group objects or you can declare your layout using simple XML file main_layout.xml which is located in the res/layout folder of your project.



Types of Layouts

- **Types of layouts**

1. **Linear Layout:** Linear Layout is a view group that aligns all children in a single direction, vertically or horizontally.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is a TextView"/>
</LinearLayout>
```

2. **Absolute Layout:** Absolute Layout enables you to specify the exact location of its children.

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="OK"
        android:layout_x="50px"
        android:layout_y="361px"/>
</AbsoluteLayout>
```

2. Table Layout: Table Layout is a view that groups views into rows and columns.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TableRow android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TextView android:text="First Name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"/>
    </TableRow>
    <TableRow android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Submit"
            android:id="@+id/button"
            android:layout_column="2"/>
    </TableRow>
</TableLayout>
```

3. RelativeLayout: Relative Layout is a view group that displays child views in relative positions.

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res
    /android" android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp">
```

```

<EditText android:id="@+id/name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/reminder"/>

<LinearLayout android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/name">

<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button"/>

<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button2"/>

</LinearLayout>

</RelativeLayout>

```

4. FragmentLayout: FrameLayout is a placeholder on screen that you can use to display a single view

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

<TextView android:text="Frame Demo"
    android:textSize="30px"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:gravity="center"/>

</FrameLayout>

```

5. ScrollLayouts: When an app has layout content that might be longer than the height of the device and that content should be vertically scrollable, then we need to use a Scroll View.

```
<ScrollView

    xmlns:android="http://schemas.android.com/apk/res/andro

    id"android:layout_width="fill_parent"

    android:layout_height="wrap_content"

    android:orientation="vertical"

    android:padding="10dp"

    android:fillViewport="false">

    <LinearLayout

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:orientation="vertical">

        <Button android:id="@+id/button"

            android:layout_width="match_parent"

            android:layout_height="wrap_content"

            android:text="KNOW MORE"/>

        <TextView android:id="@+id/textView"

            android:layout_width="wrap_content"

            android:layout_height="wrap_content"

            android:text="@string/title"

            android:textAppearance="?android:attr/textAppearanceLarge"/>

        <TextView android:id="@+id/textView2"

            android:textAppearance="?android:attr/textAppearanceSmall"/>

    </LinearLayout>

</ScrollView>
```

- 6. ScrollView:** **ScrollView** is a special kind of layout, designed to hold view larger than its actual size. When the Views size goes beyond the ScrollView size, it automatically adds scroll bars and can be scrolled vertically.

```
<ScrollView android:layout_width="match_parent"
            android:layout_height="match_parent">
    <LinearLayout android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical">
        <TextView
            android:id="@+id/tv_long" android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:text="@string/really_long_string">
        </TextView>
        <Button android:id="@+id/btn_act"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="View">
        </Button>
    </LinearLayout>
</ScrollView>
```

- **Adopting to Display Orientation**

- **Anchoring View**

Anchoring – This is easiest way to anchor views to the four edges of the screen. When the screen orientation changes, the views can anchor neatly to the edges.

Attribute Name	Description
android:layout_alignParentLeft	If set true, component place at left corner
android:layout_alignParentRight	If set true, component place at right corner
android:layout_alignParentTop	If set true, component place at top.

android:layout_alignParentBottom	If set true, component place at bottom.
android:layout_centerVertical	If set true, component place at vertically center position.
android:layout_centerHorizontal	If set true, component place at horizontally center position.

Below is the example to display button at bottom center position.

```
<Button android:layout_width="120px"
        android:layout_height="wrap_content"
        android:text="Position - Bottom Middle"
        android:id="@+id/buttondisplay7"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true" />
```

- **Managing changes to screen orientation**

Some device configurations can change during runtime (such as screen orientation, keyboard availability, and language). When such a change occurs, Android restarts the running Activity.

To properly handle a restart, it is important that your activity restores its previous state through the normal Activity lifecycle, in which Android calls `onSaveInstanceState()` before it destroys your activity so that you can save data about the application state.

Details on how you can use `onSaveInstanceState()` can be found at [Saving and restoring activity state](#).

- Retain an object during a configuration change**

Allow your activity to restart when a configuration changes, but carry a stateful object to the new instance of your activity.

- Handle the configuration change yourself**

Prevent the system from restarting your activity during certain configuration changes, but receive a callback when the configurations do change, so that you can manually update your activity as necessary.

c. Retaining an Object during a Configuration Change

When the Android system shuts down your activity due to a configuration change, the fragments of your activity that you have marked to retain are not destroyed.

You can add such fragments to your activity to preserve stateful objects.

- i) Extend the Fragment class and declare references to your stateful objects.
- ii) Call `setRetainInstance(boolean)` when the fragment is created.
- iii) Add the fragment to your activity.
- iv) Use `FragmentManager` to retrieve the fragment when the activity is restarted.

```
public class RetainedFragment extends Fragment{
    // data object we want to retain
    private MyDataObject data;
    // this method is only called once for this fragment
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        // retain this fragment
        setRetainInstance(true);
    }
    public void setData(MyDataObject data){
        this.data= data;
    }
    public MyDataObject getData(){
        return data;
    }
}
```

While `onCreate()` is called only once when the retained fragment is first created you can use `onAttach()` or `onActivityCreated()` to know when the holding activity is ready to interact with this fragment. And in your activity you can use this fragment to preserve states across configuration change restarts. Then use `FragmentManager` to add the fragment to the activity. You can obtain the data object from the fragment when the activity starts again during runtime configuration changes.

For example, define your activity as follows:

```
public class MyActivity extends Activity{

    private static final String TAG_RETAINED_FRAGMENT = "RetainedFragment";

    private RetainedFragment mRetainedFragment;

    @Override

    public void onCreate(Bundle savedInstanceState){

        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        // find the retained fragment on activity restarts

        FragmentManager fm = getFragmentManager();

        mRetainedFragment =

        (RetainedFragment)fm.findFragmentByTag(TAG_RETAINED_FRAGMENT);

        // create the fragment and data the first time

        if(mRetainedFragment == null){

            // add the fragment

            mRetainedFragment = new RetainedFragment();

            fm.beginTransaction().add(mRetainedFragment, TAG_RETAINED_FRAGMENT).commit();

            // load data from a data source or perform any calculation

            mRetainedFragment.setData(loadMyData());

        }

        // the data is available in mRetainedFragment.getData() even after

        // subsequent configuration change restarts.

        ...

    }

}
```

In order to proactively remove the retained fragment when you no longer need it, you may check for `isFinishing()` in `onPause()` in the activity.

```
@Override
public void onPause(){
    // perform other onPause related actions
    ...
    // this means that this activity will not be recreated now, user is leaving it
    // or the activity is otherwise finishing
    if(isFinishing()){
        FragmentManager fm = getFragmentManager();
        // we will not need this fragment anymore, this may also be a good place to signal
        // to the retained fragment object to perform its own cleanup.
        fm.beginTransaction().remove(mDataFragment).commit();
    }
}
```

Split Screen / Multi-Screen Activities

Split-screen mode allows two activities to be visible on screen at the same time.

Layout

To support split-screen usage, viewable content should be scaled to an appropriate size and density. Primary controls should be adapted for split-screen mode. For example, navigation tabs may be collapsed into a menu.

Responsive UI

- Apps in split-screen mode should elegantly adjust across device sizes and orientations. The screen is split along the x-axis in portrait, and along the y-axis...
- Apps in split-screen mode should elegantly adjust across device sizes and orientations.
- The screen is split along the x-axis in portrait, and along the y-axis in landscape. Changing a device's orientation should not cause the UI to change unexpectedly. For example, a video displayed in one side of a split screen shouldn't switch to full-screen if the device is rotated from portrait to landscape orientation.
- Apps may use the same or different layouts for mobile and tablet:
- Apps with similar layouts for mobile and tablet may switch between the tablet and mobile UIs when the app is resized, as the transition will not be jarring.
- Apps with completely different layouts for mobile and tablet should avoid using the mobile UI on tablet in split-screen mode. Instead, the existing tablet UI should be adapted to fit the smaller size to ensure that users have a consistent experience on both devices.

After you can use the following function to launch different activities in split screen programmatically:

```
private void goToSplitMode() {
    Intent i = new Intent(this, SplitMainActivity.class);
    PackageManager manager = getPackageManager();
    i = manager.getLaunchIntentForPackage("com.google.android.apps.maps");

    i.addFlags(Intent.FLAG_ACTIVITY_LAUNCH_ADJACENT |
        Intent.FLAG_ACTIVITY_NEW_TASK |
        Intent.FLAG_ACTIVITY_MULTIPLE_TASK);

    DisplayMetrics displayMetrics = new DisplayMetrics();
    Rect mBounds = new Rect(300, 0, getScreenWidth(this), getScreenHeight(this));
    mOptions = getActivityOptions(MainActivity.this);
    mOptions = mOptions.setLaunchBounds(mBounds);

    startActivity(i, mOptions.toBundle());

    i = new Intent(this, SplitMainActivity.class);
    mBounds = new Rect(0, 0, 300, getScreenHeight(this));
    mOptions = getActivityOptions(MainActivity.this);
    mOptions = mOptions.setLaunchBounds(mBounds);

    i.addFlags(Intent.FLAG_ACTIVITY_LAUNCH_ADJACENT |
        Intent.FLAG_ACTIVITY_NEW_TASK |
        Intent.FLAG_ACTIVITY_MULTIPLE_TASK);

    startActivity(i, mOptions.toBundle());
}

public static ActivityOptions getActivityOptions(Context context) {
    ActivityOptions options = ActivityOptions.makeBasic();
    int freeform_stackId = 5;
    try {
        Method method = ActivityOptions.class.getMethod("setLaunchWindowingMode", int.class);
        method.invoke(options, freeform_stackId);
    } catch (Exception e) { /* Gracefully fail */
    }

    return options;
}

public static int getScreenWidth(@NonNull Activity activity) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
        WindowMetrics windowMetrics = activity.getWindowManager().getCurrentWindowMetrics();
        Insets insets = windowMetrics.getWindowInsets()
            .getInsetsIgnoringVisibility(WindowInsets.Type.systemBars());
        return windowMetrics.getBounds().width() - insets.left - insets.right;
    } else {
        DisplayMetrics displayMetrics = new DisplayMetrics();

```

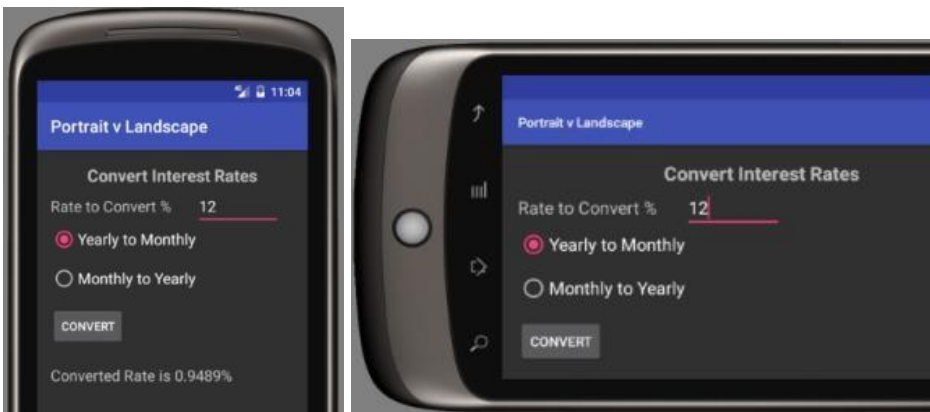
Lab Assignment

SET A

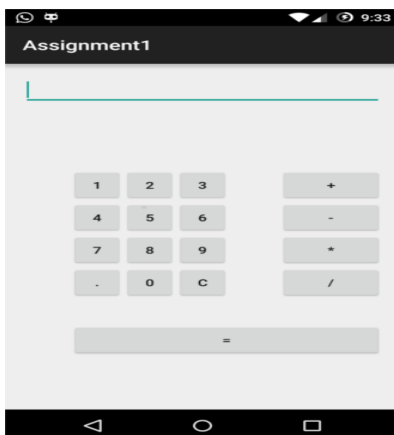
1. Design following-add a border to an Android Layout



2. Design an Android Portrait and Landscape Screen Layout Example

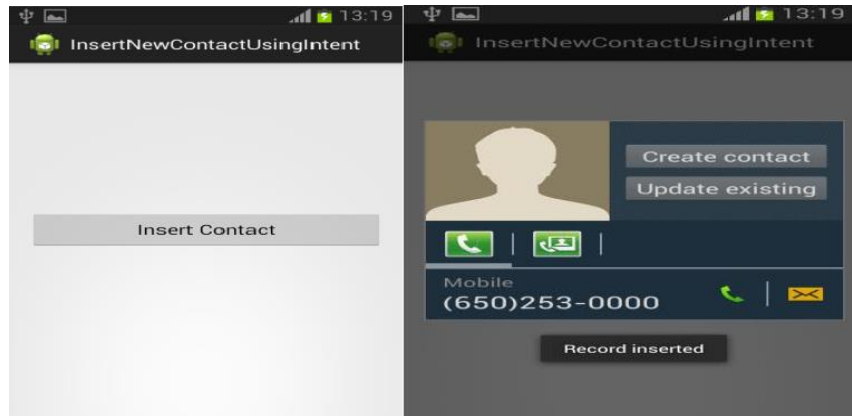


3. Create the simple calculator shown below also perform appropriate operation

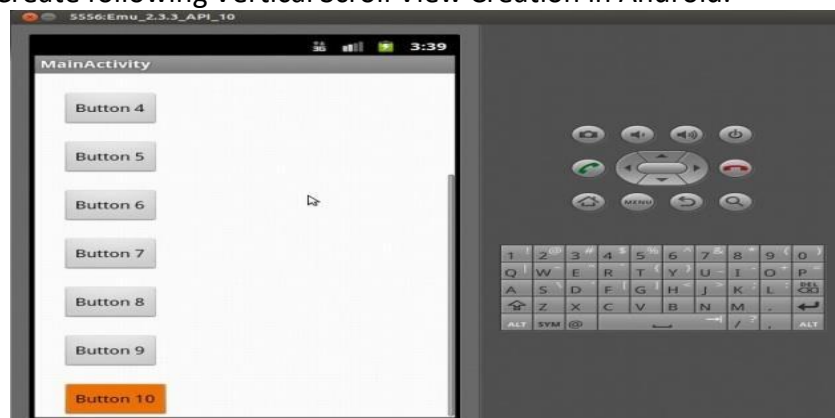


SET B

1. Create new contact for designing following layout

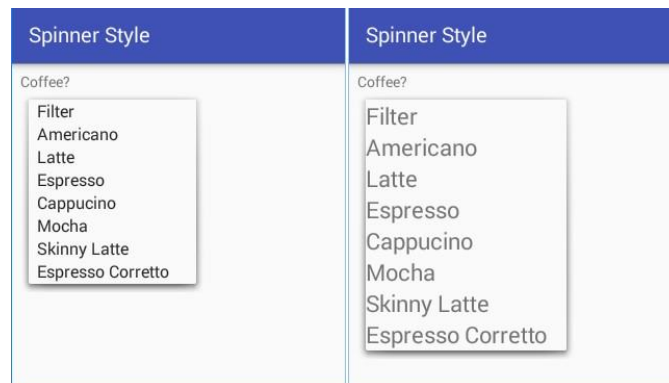


2. Create following Vertical Scroll View Creation in Android.



SET C

1. Create following layout which is changing android spinner text size with styles.



Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Objectives

- How to create and handle the views in your application.
- How to use fragments.
- How to display Pictures and menus in android

Reading

You should read the following topics before starting this exercise:

1. UI components available in android.
2. Use of List and Spinner view
3. Use of Fragments and types of fragments.
4. Image view and Image Switcher
5. Views available to create menus

Ready Reference**Views**

View is the basic building block of UI(User Interface) in android. View refers to the android.view.View class, which is the super class for all the GUI components like Text View, ImageView, and Button etc. View class extends Object class and implements Drawable.Callback, KeyEvent.Callback and Accessibility Event Source.

View can be considered as a rectangle on the screen that shows some type of content. It can be an image, a piece of text, a button or anything that an android application can display. The rectangle here is actually invisible, but every view occupies a rectangle shape.

Types of View

TextView: TextView used to create text field. A TextView displays text to the user and optionally allows them to edit it.

Attribute Name	Description
android:id	This is the ID which uniquely identifies the control.
android:editable	If set to true, specifies that this TextView has an input method.
android:inputType	The type of data being placed in a text field. Phone, Date, Time, Number, Password etc.
android:text	Text to display.
android:textAllCaps	If set to true, display the text in ALL CAPS.

To create TextView make following changes in **res/layout/activity_main.xml** file

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

    android:layout_width="300dp"

    android:layout_height="200dp"

    android:capitalize="characters"

    android:text="hello_world"/>
```

To access the value of TextView make following changes in **MainActivity.java** file

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

EditText: An EditText is an overlay over TextView that configures itself to be editable. It is the predefined subclass of TextView that includes rich editing capabilities.

Attribute Name	Description
android:autoText	If set to true, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
android:drawableBottom	This is the drawable to be drawn below the text.

To create EditText make following changes in **res/layout/activity_main.xml** file

```
<EditText android:id="@+id/edittext"

    android:layout_width="fill_parent"

    android:layout_height="wrap_content"

    android:text="@string/enter_text"

    android:inputType="text" />
```

To access the value of EditText make following changes in **MainActivity.java** file

```
EditText eText = (EditText) findViewById(R.id.edittext);
```

Button: A Button is a Push-button which can be pressed, or clicked, by the user to perform an action.

To create Button make following changes in **res/layout/activity_main.xml** file

```
<Button android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="Button"

        android:id="@+id/button" />
```

To access the value of Button make following changes in **MainActivity.java** file

```
Button b1 = (Button)findViewById(R.id.button);
```

ImageButton: An ImageButton is an AbsoluteLayout which enables you to specify the exact location of its children. This shows a button with an image.

Attribute Name	Description
android:adjustViewBounds	Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
android:baseline	This is the offset of the baseline within this view.
android:baselineAlignBottom	If true, the image view will be baseline aligned with based on its bottom edge.
android:cropToPadding	If true, the image will be cropped to fit within its padding.
android:src	This sets a drawable as the content of this ImageView.

To create **ImageButton** make following changes in **res/layout/activity_main.xml** file

```
<ImageButton android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:id="@+id/imageButton"

        android:src="@drawable/abc"/>
```

To access the value of **ImageButton** make following changes in *MainActivity.java* file

```
ImageButton imgButton = (ImageButton)findViewById(R.id.imageButton);
```

ToggleButton:

A toggle button allows the user to change a setting between two states.

Attribute Name	Description
android:disabledAlpha	This is the alpha to apply to the indicator when disabled.
android:textOff	This is the text for the button when it is not checked.
android:textOn	This is the text for the button when it is checked.

To create **ToggleButton** make following changes in *res/layout/activity_main.xml* file

```
<ToggleButton android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="On"
    android:id="@+id/toggleButton"
    android:checked="true" />
```

To access the value of **ToggleButton** make following changes in *MainActivity.java* file

```
ToggleButton tg1 = (ToggleButton)findViewById(R.id.toggleButton);
```

RadioButton: A RadioButton has two states: either checked or unchecked. This allows the user to select one option from a set.

To create **RadioButton** make following changes in *res/layout/activity_main.xml* file

```
<RadioButton android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ANDROID"
    android:id="@+id/radioButton2"
    android:checked="false" />
```

RadioGroup: A RadioGroup class is used for set of radio buttons. If we check one radio button that belongs to a radio group, it automatically unchecks any previously checked radio button within the same group.

Attribute Name	Description
android:checkedButton	This is the id of child radio button that should be checked by default within this radio group.

To create **RadioGroup** make following changes in **res/layout/activity_main.xml** file

```
<RadioGroup android:layout_width="fill_parent"
    android:layout_height="90dp"
    android:layout_marginTop="58dp"
    android:id="@+id/radioGroup" >
    <RadioButton android:layout_width="wrap_content"
        android:layout_height="55dp"
        android:text="Male"
        android:id="@+id/radioButton"
        android:checked="false" />
    <RadioButton android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Female"
        android:id="@+id/radioButton2"
        android:checked="false"
        android:layout_weight="0.13" />
</RadioGroup>
```

To access the value of **RadioGroup** make following changes in **MainActivity.java** file

```
RadioGroup radioGroup = (RadioGroup)findViewById(R.id.radioGroup);  
  
int selectedId=radioGroup.getCheckedRadioButtonId();  
  
RadioButton radioButton = (RadioButton)findViewById(selectedId);
```

CheckBox: A CheckBox is an on/off switch that can be toggled by the user. You should use check-boxes when presenting users with a group of selectable options that are not mutually exclusive.

To create **CheckBox** make following changes in **res/layout/activity_main.xml** file

```
<CheckBox android:id="@+id/checkBox2"  
  
    android:layout_width="wrap_content"  
  
    android:layout_height="wrap_content"  
  
    android:text="Do you like android "  
  
    android:checked="false" />
```

To access the value of **CheckBox** make following changes in **MainActivity.java** file

```
CheckBox ch1 = (CheckBox)findViewById(R.id.checkBox1);
```

ProgressBar: Allows you to create progress bar. Progress bars are used to show progress of a task.

Public Methods	Description
int getMax()	This method returns the maximum value of the progress.
void incrementProgressBy(int diff)	This method increment the progress bar by the difference of value passed as a parameter.
void setIndeterminate(boolean indeterminate)	This method sets the progress indicator as determinate or indeterminate.
void setMax(int max)	This method sets the maximum value of the progress dialog.
void setProgress(int value)	This method is used to update the progress dialog with some specific value.
void show(Context context, CharSequence title, CharSequence message)	This is a static method, used to display progress dialog.

To create **ProgressBar** make following changes in **res/layout/activity_main.xml** file

```
<ProgressBar android:id="@+id/progressBar"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:progress="25" />
```

To access the value of **ProgressBar** make following changes in **MainActivity.java** file

```
progressBar = (ProgressBar) findViewById(R.id.progressBar);
new Thread(new Runnable() {
    public void run() {
        while (progressStatus < 100) {
            progressStatus += 1;
            // Update the progress bar and display the current value in the text view
            handler.post(new Runnable() {
                public void run() {
                    progressBar.setProgress(progressStatus);
                    textView.setText(progressStatus + "/" + progressBar.getMax());
                }
            });
            try {
                Thread.sleep(200); // Sleep for 200 milliseconds.
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}).start();
```

AutoCompleteTextView: An AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.

Attribute Name	Description
android:completionHint	Used to define the hint displayed in the drop down menu.
android:completionThreshold	Used to define the number of characters that the user must type before completion suggestions are displayed in a drop down.
android:dropDownAnchor	This is the View to anchor the auto-complete dropdown to.
android:dropDownHeight	Used to specify the basic height of the dropdown.
android:dropDownSelector	This is the selector in a drop down list.
android:dropDownWidth	This specifies the basic width of the dropdown.

To create **AutoCompleteTextView** make following changes in **res/layout/activity_main.xml** file

```
<AutoCompleteTextView android:id="@+id/autoCompleteTextView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10" />
```

To access the value of **AutoCompleteTextView** make following changes in **MainActivity.java**

```
AutoCompleteTextView autocomplete = (AutoCompleteTextView) findViewById(
R.id.autoCompleteTextView1);

String[] arr = { "India", "United States", "Brazil", "Italy" };

ArrayAdapter<String> adapter = new ArrayAdapter<String> (this,
android.R.layout.select_dialog_item, arr);

autocomplete.setThreshold(2);

autocomplete.setAdapter(adapter);
```

TimePicker: Android Time Picker allows you to select the time of day in either 24 hour or AM/PM mode. The time consists of hours, minutes and clock format.

Public Methods	Description
boolean is24HourView()	Returns true if this is in 24 hour view
boolean isEnabled()	Returns the enabled status for this view
void setCurrentHour(Integer currentHour)	This method sets the current hour
void setCurrentMinute(Integer currentMinute)	This method sets the current minute
void setIs24HourView(Boolean is24HourView)	This method set whether in 24 hour or AM/PM mode
void setOnTimeChangedListener (TimePicker.OnTimeChangedListener onTimeChangedListener)	This method Set the callback that indicates the time has been adjusted by the user

To create **TimePicker** make following changes in **res/layout/activity_main.xml** file

```
<TimePicker android:id="@+id/timePicker1"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

To access the value of **TimePicker** make following changes in **MainActivity.java** file

```
import android.widget.TimePicker;

private TimePicker timePicker1;

timePicker1 = (TimePicker) findViewById(R.id.timePicker1);

int hour = timePicker1.getCurrentHour();

int min = timePicker1.getCurrentMinute();
```


DatePicker: Android Date Picker allows you to select the date consisting of day, month and year in your custom user interface.

Public Methods	Description
int getDayOfMonth()	Returns the selected day of month
int getMonth()	Returns the selected month
int getYear()	Returns the selected year
void setMaxDate(long maxDate)	This method sets the maximal date supported by this DatePicker in milliseconds since January 1, 1970 00:00:00 in getDefault() time zone
void updateDate(int year, int month, int dayOfMonth)	This method updates the current date
CalendarView getCalendarView()	Returns calendar view
int getFirstDayOfWeek()	Returns first day of the week

To create **DatePicker** make following changes in **res/layout/activity_main.xml** file

```
<DatePicker android:id="@+id/datePicker1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
```

To access the value of **DatePicker** make following changes in **MainActivity.java** file

```
DatePicker picker = (DatePicker)findViewById(R.id.datePicker1);
```

ListView: ListView is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.

To create **ListView** make following changes in **res/layout/activity_main.xml** file

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

</ListView>
```

To access the value of **ListView** make following changes in **MainActivity.java** file

```
// Array of strings...

String[] mobileArray = {"Android","IPhone","WindowsMobile","Blackberry"};

ArrayAdapter adapter = new ArrayAdapter<String>(this, R.layout.activity_listview,
mobileArray);

ListView listView = (ListView) findViewById(R.id.mobile_list);

listView.setAdapter(adapter);
```

Spinner: Spinner allows you to select an item from a drop down menu To create **Spinner** make following changes in **res/layout/activity_main.xml** file

```
<Spinner android:id="@+id/spinner"

    android:layout_width="fill_parent"

    android:layout_height="wrap_content"

    android:prompt="@string/spinner_title"/>
```

To access the value of **Spinner** make following changes in **MainActivity.java** file

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Spinner spinner = (Spinner) findViewById(R.id.spinner);
    spinner.setOnItemClickListener(this); // Spinner click listener

    List<String> categories = new ArrayList<String>();
    categories.add("Automobile");
    categories.add("Computers");
    categories.add("Education");
    categories.add("Personal");

    // Creating adapter for spinner
    ArrayAdapter<String> dataAdapter = new ArrayAdapter<String> (this,
    android.R.layout.simple_spinner_item, categories);

    // Drop down layout style - list view with radio button
    dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
    em);
    spinner.setAdapter(dataAdapter);    // attaching data adapter to spinner }
```

```

@Override

public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
{

    // On selecting a spinner item

    String item = parent.getItemAtPosition(position).toString();

    // Showing selected spinner item

    Toast.makeText(parent.getContext(), "Selected: " + item,
    Toast.LENGTH_LONG).show();
}

public void onNothingSelected(AdapterView<?> arg0) {

    // TODO Auto-generated method stub

}

```

Understanding Specialized Fragments

ListFragment: A list fragment is a fragment that contains a ListView, displaying a list of items from a data source such as an array or a Cursor.

Steps to create ListFragment

Step 1: Create fragment in res/layout/activity_main.xml file

```

<fragment

    android:id="@+id/fragment1"

    android:name="com.example.admin.ListFragmentExample.MainActivity"

    android:layout_width="match_parent"

    android:layout_height="match_parent" />

```

Step 2: Create fragment1.xml in res/layout folder and add below code in fragment1.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:gravity="center"

    android:padding="10dp"

    android:orientation="vertical">

```

```

<ListView android:id="@id/android:list"

        android:layout_width="match_parent"

        android:layout_height="match_parent"

        android:layout_weight="1"

        android:drawSelectorOnTop="false" />

</LinearLayout>

```

```

package com.example.admin.ListFragmentExample;
import android.app.ListFragment;
import android.os.Bundle;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
public class MainActivity extends ListFragment {

        String[ ] CountryNames = { -India, -Nepal, -South Africa, -Shri Lanka }; public

        View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {

                return inflater.inflate(R.layout.fragment1, container, false);
        }
        public void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);

                setListAdapter(new ArrayAdapter<String>(getActivity(),
android.R.layout.simple_list_item_1, CountryNames) );
        }
}

```

Step 3: Create a Java Class file and name it MainActivity1.

```

        public void onListItemClick(ListView parent, View v, int position, long id) {

                Toast.makeText(getActivity(), -You have selected || + CountryNames[position],
Toast.LENGTH_SHORT).show( );

        } }

```

DialogFragment: A DialogFragment is a fragment that displays a modal window, floating on top of the current activity window. Android provides several standard dialog implementation AlertDialog, ProgressDialog, DatePickerDialog or TimePickerDialog

Steps to create dialog fragment

Step 1: Create a new class which extends DialogFragment and override the onCreateView() method to create dialog fragment

```
public class MyDialogFragment extends DialogFragment {

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_sample_dialog, container, false);

        getDialog().setTitle("Test Dialog");

        return view;

    }

}
```

Step 2: Define DialogFragment Layout as given below:

```
<?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:gravity="center"
    android:padding="10dp"
    android:orientation="vertical">

    <TextView android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Test" android:textSize="20dp" />

    <Button android:id="@+id/dismiss"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Dismiss" />

</LinearLayout>
```

Step 3: To display the DialogFragment, Add below code into *MyDialogFragment.java* onCreateView() method.

```
FragmentManager fm = getFragmentManager();

MyDialogFragment dialogFragment = new MyDialogFragment ();

dialogFragment.show(fm, "Sample Fragment");
```

Step 4: To dismiss DialogFragment, Add below code into *MyDialogFragment.java* onCreateView() method.

```
Button dismiss = (Button) rootView.findViewById(R.id.dismiss);

dismiss.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) { dismiss();

    }

});
```

PreferenceFragment: Allow users to personalize the application for their own use.

Steps to create PreferenceFragment

Step 1: Create a new xml folder under res folder and then add a new Android XML file to it. Name the xml file preferences.xml

Step 2: Populate the preferences.xml file as follows:

```
<?xml version="1.0" encoding="utf-8"?>

<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">

    <PreferenceCategory android:title="Category 1">

        <CheckBoxPreference android:title="CheckBox"

            android:defaultValue="false"

            android:summary="True or False"

            android:key="checkboxPref" />
```

```

</PreferenceCategory>

<PreferenceCategory android:title="Category 2">

    <EditTextPreference android:name="EditText"

        android:title="Edit Text"

        android:key="editTextPref" />

    <RingtonePreference android:name="Ringtone Preference"

        android:summary="Select a ringtone"

        android:title="Ringtones"

        android:key="ringtonePref" />
    <PreferenceScreen android:title="Second Preference Screen"

        android:summary="Click here to go to the second Preference Screen"

        android:key="secondPreferencePref" />

    <EditTextPreference android:name="EditText"

        android:title="Edit Text (Second Screen)"

        android:key="secondEditTextPref" />

</PreferenceScreen>

</PreferenceCategory>

</PreferenceScreen>

```

Step 3: Create a Java Class file and name it Fragment1 and add below code in Fragment1.java

```

package com.example.admin.PreferenceFragmentExample;
import android.os.Bundle;
import android.preference.PreferenceFragment;
public class Fragment1 extends PreferenceFragment {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}

```

Step 4: Modify the PreferenceFragmentExampleActivity.java file add below code in onCreate() method.

```
import android.app.FragmentManager;

import android.app.FragmentTransaction;

FragmentManager fragmentManager = getFragmentManager();

FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

Fragment1 fragment1 = new Fragment1();

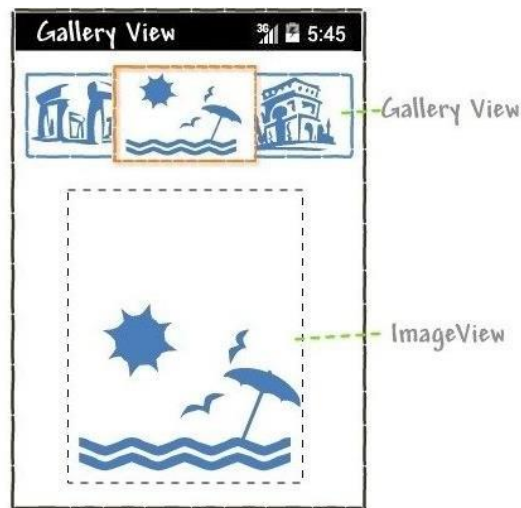
fragmentTransaction.replace(android.R.id.content, fragment1);

fragmentTransaction.addToBackStack(null);

fragmentTransaction.commit ();
```

Android Gallery and ImageView to View:

In your application if you want display a series of images to the user, you can make use of the Gallery. The Gallery is a view that shows items (such as images) in a center-locked, horizontal scrolling list.



How to Create Android Gallery and ImageView:

1. Gallery is used to show Views in a horizontal list, and user will select a view , Userselected view will be shown in center of the Horizontal list.
2. The items of Gallery are getting from an Adapter, just like ListView, in which ListViewitems are getting from an Adapter.
3. We need to make an Adapter class that extends BaseAdapter class and override getView() method.

Example:

1. Using the Gallery View

1. For Android Gallery View Example, assume you have some images stored in the res/drawable-mdpi folder of your project.
2. Create an XML file named attrs.xml and store it in the res/values folder.
3. Add this content to the attrs.xml file:

```
<resources>

    <declare-styleable name="MyGallery">

        <attr name="android:galleryItemBackground" />

    </declare-styleable>

</resources>
```

4. Open **-res/layout/activity_main.xml** file and put this xml code it into file:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context="{relativePackage}.$ {activityClass}"

    android:orientation="vertical" >

    <Gallery android:id="@+id/gallery1"
```

```
        android:layout_width="fill_parent"

        android:layout_height="wrap_content" />

<ImageView android:id="@+id/image1"

        android:layout_width="215dp"

        android:layout_height="315dp"

        android:layout_gravity="center_horizontal"

        android:background="#cfcfcf"

        android:paddingTop="5dp"

        android:paddingBottom="5dp"

        android:paddingLeft="10dp"

        android:paddingRight="10dp"

        android:src="@drawable/pic1" />

</LinearLayout>
```

5. Open **-src/package-name/MainActivity.java** file and add following JAVA code.

```
package androidinterview.com.androidgalleryview;

import android.app.Activity;
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;
```

```

public class MainActivity extends Activity {

    //the images to display
    Integer[] imageIDs = { R.drawable.pic1, R.drawable.pic2, R.drawable.pic3,
                           R.drawable.pic4, R.drawable.pic5, R.drawable.pic6,
                           R.drawable.pic7

    };

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

        // Note that Gallery view is deprecated in Android 4.1---
        Gallery gallery = (Gallery) findViewById(R.id.gallery1);
        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View v, int position, long id)
            {
                Toast.makeText(getApplicationContext(), "pic" + (position + 1) + " selected",
Toast.LENGTH_SHORT).show();

                // display the images selected
                ImageView imageView = (ImageView) findViewById(R.id.image1);
                imageView.setImageResource(imageIDs[position]);
            }
        });
    }

    public class ImageAdapter extends BaseAdapter {
        private Context context;
        private int itemBackground;
        public ImageAdapter(Context c)
        {

```

```

        context = c;
        // sets a grey background; wraps around the images
        TypedArray a = obtainStyledAttributes(R.styleable.MyGallery);
        itemBackground =
a.getResourceId(R.styleable.MyGallery_android_galleryItemBackground, 0);
        a.recycle();
    }
    public int getCount() {
        return imageIDs.length; // returns the number of images
    }
    public Object getItem(int position) {
        return position;          // returns the ID of an item
    }
    public long getItemId(int position) {
        return position;
    }
    // returns an ImageView view
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView = new ImageView(context);
        imageView.setImageResource(imageIDs[position]);
        imageView.setLayoutParams(new Gallery.LayoutParams(100, 100));
        imageView.setBackgroundResource(itemBackground);
        return imageView;
    }
}

```

2. The following Try It Out shows you how to use the Gallery view to display a set of images.

- Using Android Studio, create a new Android project and name it **Gallery**.
- Add the following code in the *main.xml* file.

```

<Gallery android:id="@+id/gallery1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
<ImageView android:id="@+id/image1"
        android:layout_width="320dp"
        android:layout_height="250dp"
        android:scaleType="fitXY" />

```

- Right-click on the *res/values* folder and select **New File**. Name the file as **attrs.xml**.
- Populate the **attrs.xml** file as follows:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Gallery1">
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>

```

- Prepare a series of images and name them **pic1.png**, **pic2.png**, and so on for each subsequent image. Drag and drop all the images into the *res/drawable-mdpi* folder. When a dialog is displayed, check the **Copy files** option and click **OK**.
- Add the following statements into the **GalleryActivity.java** file:

```

package net.learn2develop.Gallery;
import android.app.Activity;
import android.content.Context;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;

```

```

import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;

public class GalleryActivity extends Activity{

    ///---theimagestodisplay---
    Integer[] imageIDs ={ R.drawable.pic1, R.drawable.pic2, R.drawable.pic3,
                           R.drawable.pic4, R.drawable.pic5, R.drawable.pic6,
                           R.drawable.pic7 };

    /**Calledwhentheactivityisfirstcreated.*/
    @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            Gallery gallery = (Gallery) findViewById(R.id.gallery1);
            gallery.setAdapter(new ImageAdapter(this));
            gallery.setOnItemClickListener(new OnItemClickListener()
            {
                public void onItemClick(AdapterView parent, View v, int position, longid)
                {
                    Toast.makeText(getBaseContext(), -pic|| +(position+1)+
-selected||, Toast.LENGTH_SHORT).show();
                }
            });
        }

    public class ImageAdapter extendsBaseAdapter {
        Context context;
        int itemBackground;
        public ImageAdapter(Context c) {

```

```

        context = c;
        //---setting the style---
        TypedArray a = obtainStyledAttributes( R.styleable.Gallery1);
        itemBackground = a.getResourceId(
            R.styleable.Gallery1_android_galleryItemBackground, 0);
        a.recycle();
    }
    public int getCount(){
        return imageIDs.length;    // returnsthenumberofimages
    }
    public Object getItem(int position){
        return position;    // returns the item
    }
    public long getItemId(int position){
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if (convertView == null) {
            imageView = new ImageView(context);
            imageView.setImageResource(imageIDs[position]);
            imageView.setScaleType(ImageView.ScaleType.FIT_XY);
            imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
        } else {
            imageView = (ImageView) convertView;
        }
        imageView.setBackgroundResource(itemBackground);
        return imageView;
    }
}

```

- Press F11 to debug the application on the Android emulator.
- To display the selected image in the ImageView, add the following statements in bold to the ***GalleryActivity.java*** file:

```
//---display the imagesselected---
```

```
ImageView imageView = (ImageView) findViewById(R.id.image1);
```

```
imageView.setImageResource(imageId[position]);
```

- **Image Switcher**

An image switcher allows you to add some transitions on the images through the way they appear on screen. In order to use image Switcher, you need to define its XML component first. Its syntax is given below:

```
<ImageSwitcher android:id="@+id/imageSwitcher1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true" >
</ImageSwitcher>
```

Now we create an instance of ImageSwitcher in java file and get a reference of this XML component. Its syntax is given below:

```
private ImageSwitcher imageSwitcher;
imageSwitcher = (ImageSwitcher)findViewById(R.id.imageSwitcher1);
```

The next thing we need to do is implement the ViewFactory interface and implement unimplemented method that returns an imageView. Its syntax is below:

```
imageSwitcher.setImageResource(R.drawable.ic_launcher);
imageSwitcher.setFactory(new ViewFactory() {
    public View makeView() {
        ImageView myView = new ImageView(getApplicationContext());
```



```
        return myView;
    }
}
```

The last thing you need to do is to add Animation to the ImageSwitcher. You need to define an object of Animation class through AnimationUtilities class by calling a static method loadAnimation. Its syntax is given below:

```
Animation in = AnimationUtils.loadAnimation(this,android.R.anim.slide_in_left);
imageSwitcher.setInAnimation(in);
imageSwitcher.setOutAnimation(out);
```

The method setInAnimaton sets the animation of the appearance of the object on the screen whereas setOutAnimation does the opposite. The method loadAnimation() creates an animation object. Apart from these methods, there are other methods defined in the ImageSwitcher class. They are defined below:

Public Methods	Description
setImageDrawable(Drawable drawable)	Sets an image with image switcher. The image is passed in the form of bitmap.
setImageResource(int resid)	Sets an image with image switcher. The image is passed in the form of integer id.
setImageURI(Uri uri)	Sets an image with image switcher. The image is passed in the form of URI.
ImageSwitcher(Context context, AttributeSet attrs)	Returns an image switcher object with already setting some attributes passed in the method.

Example:

1. Using the ImageSwitcher View

We want to apply some animation to the image when it transitions from one image to another. In this case, you need to use the ImageSwitchertogether with the Galleryview.

The following Try It Out shows you how.

- 1) Using Andriod, create a new Android project and name it **ImageSwitcher**
- 2) Modify the main.xml file by adding the following statements:

```
<Gallery android:id="@+id/gallery1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<ImageSwitcher android:id="@+id/switcher1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true" />
```

- 3) Right-click on the **res/values** folder and select New, File. Name the file **attrs.xml**.
- 4) Populate the attrs.xml file as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Gallery1">
        <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>
```

- 5) Drag and drop a series of images into the res/drawable-mdpi folder. When a dialog is displayed, check the Copy files option and click OK.
- 6) Add the following statements to the ImageSwitcherActivity.java file:

```
import android.content.Context;
import android.content.res.TypedArray;
import android.view.View; import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
```

```

import android.view.animation.AnimationUtils;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.BaseAdapter;

import android.widget.Gallery;

import android.widget.ImageSwitcher;

import android.widget.ImageView;

import android.widget.ViewSwitcher.ViewFactory;

public class ImageSwitcherActivity extends Activity implements ViewFactory {

    //---the image to display---

    Integer[] imageIDs = { R.drawable.pic1, R.drawable.pic2,

                           R.drawable.pic3, R.drawable.pic4,

                           R.drawable.pic5, R.drawable.pic6, R.drawable.pic7 };

    private ImageSwitcher imageSwitcher;

    public void onCreate(Bundle savedInstanceState) {

        imageSwitcher = (ImageSwitcher) findViewById(R.id.switcher1);

        imageSwitcher.setFactory(this);

        imageSwitcher.setInAnimation(AnimationUtils.loadAnimation
(this, android.R.anim.fade_in));

        imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation
(this, android.R.anim.fade_out));

        Gallery gallery = (Gallery) findViewById(R.id.gallery1);

        gallery.setAdapter(new ImageAdapter(this));

        gallery.setOnItemClickListener(new OnItemClickListener() {

            public void onItemClick(AdapterView<?> parent, View v, int position,
long id) {

```

```

        imageSwitcher.setImageResource(imageIDs[position]);

    }

});

}

public View makeView(){

    ImageView imageView = new ImageView(this);

    imageView.setBackgroundColor(0xFF000000);

    imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);

    imageView.setLayoutParams(new ImageSwitcher.LayoutParams(

        LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));

    return imageView;

}

public class ImageAdapter extends BaseAdapter{

    private Context context;

    private int itemBackground;

    public ImageAdapter(Context c){

        context = c;

        TypedArray a = obtainStyledAttributes(R.styleable.Gallery1);

        itemBackground = a.getResourceId(

R.styleable.Gallery1_android_galleryItemBackground, 0);

        a.recycle();

    }

    public int getCount(){

        return imageIDs.length;

    }

```

```

        public Object getItem(int position){

            return position;

        }

        public long getItemId(intposition){

            return position;

        }

        public View getView(int position, View convertView, ViewGroup parent){

            ImageViewimageView=newImageView(context);

            imageView.setImageResource(imageIDs[position]);

            imageView.setScaleType(ImageView.ScaleType.FIT_XY);

            imageView.setLayoutParams(newGallery.LayoutParams(150,120));

            imageView.setBackgroundResource(itemBackground);

            return imageView;

        }

    }
}

```

- **ImageView**

Like the Gallery example in the previous section, you also implemented an ImageAdapter class so that it can bind to the Gallery view with a series of ImageViews.

@Override

```

public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    imageSwitcher = (ImageSwitcher) findViewById(R.id.switcher1);
}

```

```

        imageSwitcher.setFactory(this);

        imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,android.R.anim.fade_in));

        imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
        android.R.anim.fade_out));

        Gallery gallery = (Gallery) findViewById(R.id.gallery1);

        gallery.setAdapter(new ImageAdapter(this));

        gallery.setOnItemClickListener(new OnItemClickListener() {

            public void onItemClick(AdapterView<?> parent, View v, int position, long id) {

                imageSwitcher.setImageResource(imageIDs[position]);

            }

        });

    }
}

```

- **Grid View**

Android **GridView** shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a **ListAdapter**.

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter can be used to supply the data to like spinner, list view, grid view etc.

The **ListView** and **GridView** are subclasses of **AdapterView** and they can be populated by binding them to an **Adapter**, which retrieves data from an external source and creates a View that represents each data entry.

GridView Attributes

Following are the important attributes specific to GridView –

Attribute Name	Description
android:id	This is the ID which uniquely identifies the layout.

android:columnWidth	This specifies the fixed width for each column. This could be in px, dp, sp, in, or mm.
android:gravity	Specifies the gravity within each cell. Possible values are top, bottom, left, right, center etc.
android:horizontalSpacing	Defines the default horizontal spacing between columns. This could be in px, dp, sp, in, or mm.
android:numColumns	Defines how many columns to show. May be an integer value, such as "100" or auto_fit which means display as many columns as possible to fill the available space.
android:verticalSpacing	Defines the default vertical spacing between rows. This could be in px, dp, sp, in, or mm.

Example

- Following is the content of **src/com.example.helloworld/MainActivity.java**.

```
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.GridView;

public class MainActivity extends Activity {

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

        GridView gridview = (GridView) findViewById(R.id.gridview);
        gridview.setAdapter(new ImageAdapter(this));
    }
}
```

- Following will be the content of **res/layout/activity_main.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center" />
```

- 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">HelloWorld</string>
    <string name="action_settings">Settings</string>
</resources>
```

- Following will be the content of **src/com.example.helloworld/ImageAdapter.java** file.

```
package com.example.helloworld;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
```



```
public class ImageAdapter extends BaseAdapter {  
    private Context mContext;  
    public ImageAdapter(Context c) {  
        mContext = c;  
    }  
    public int getCount() {  
        return mThumbIds.length;  
    }  
    public Object getItem(int position) {  
        return null;  
    }  
    public long getItemId(int position) {  
        return 0;  
    }  
    // create a new ImageView for each item referenced by the Adapter  
    public View getView(int position, View convertView, ViewGroup parent) {  
        ImageView imageView;  
        if (convertView == null) {  
            imageView = new ImageView(mContext);  
            imageView.setLayoutParams(new GridView.LayoutParams(85, 85));  
            imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);  
            imageView.setPadding(8, 8, 8, 8);  
        }  
        else {  
            imageView = (ImageView) convertView;  
        }  
        imageView.setImageResource(mThumbIds[position]);  
        return imageView;  
    }  
    public Integer[] mThumbIds = { R.drawable.sample_2, R.drawable.sample_3,  
                                   R.drawable.sample_4, R.drawable.sample_5,
```

```
R.drawable.sample_6, R.drawable.sample_7,  
R.drawable.sample_0, R.drawable.sample_1,  
R.drawable.sample_2, R.drawable.sample_3,  
R.drawable.sample_4, R.drawable.sample_5,  
R.drawable.sample_6, R.drawable.sample_7 };  
}
```

- Let's try to run our modified **Hello World!** Application we just modified.

- **Using Menus with Views**

Menus are useful for displaying additional options that are not directly visible on the main UI of an application. There are two main types of menus in Android:

- **Options menu** — Displays information related to the current activity. In Android, you activate the options menu by pressing the MENU key.
- **Context menu** — Displays information related to a particular view on an activity. In Android, to activate a context menu you tap and hold on to it.

- **Creating the Helper Methods**

Before you go ahead and create your options and context menus, you need to create two helper methods. One creates a list of items to show inside a menu, while the other handles the event that is fired when the user selects an item inside the menu.

Example:

1. Creating the Menu Helper Methods:--

- a.** Add the following statements in the MainActivity.java file

```
import android.view.Menu;  
import android.view.MenuItem;  
import android.widget.Button;  
import android.widget.Toast;  
  
private void CreateMenu(Menu menu) {  
    MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");  
    {
```

```

        mnu1.setAlphabeticShortcut('_a');
        mnu1.setIcon(R.drawable.icon);
    }
    MenuItem mnu2 = menu.add(0, 1, 1, -Item 2);
    {
        mnu2.setAlphabeticShortcut('_b');
        mnu2.setIcon(R.drawable.icon);
    }
    menu.add(0, 2, 2, -Item 3);
}
private boolean MenuChoice(MenuItem item)
{
    switch (item.getItemId()) {
        case 0: Toast.makeText(this, -You clicked on Item 1,
            Toast.LENGTH_LONG).show();
            return true;
        case 1: Toast.makeText(this, -You clicked on Item 2,
            Toast.LENGTH_LONG).show();
            return true;
        case 2: Toast.makeText(this, -You clicked on Item 3,
            Toast.LENGTH_LONG).show();
            return true;
    }
    return false;
}
}

```

- **Options Menu**

You are now ready to modify the application to display the options menu when the user presses the MENU button on the Android device.

Example:

- **Displaying an Options Menu**

1. Using the same project created in the previous section, add the following statements into the MainActivity.java file:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    CreateMenu(menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    return MenuChoice(item);
}
```

2. Press F11 to debug the application on the Android Emulator.

- **Context Menu**

A context menu is usually associated with a view on an activity, and it is displayed when the user long clicks an item. For example, if the user taps on a Button view and hold it for a few seconds, a context menu can be displayed.

Exmple:

- **Displaying a Context Menu**

1. using the same project created in the previous section, add the following statements in the MainActivity.java file:

```
import android.view.View;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
public void onCreate(Bundle savedInstanceState) {
    Button btn = (Button) findViewById(R.id.btn1);
    btn.setOnCreateContextMenuListener(this);
}
```

@Override

```
public void onCreateContextMenu(ContextMenu menu, View view,  
ContextMenuInfo menuInfo) {  
    super.onCreateContextMenu(menu, view, menuInfo);  
    CreateMenu(menu);  
}
```

@Override

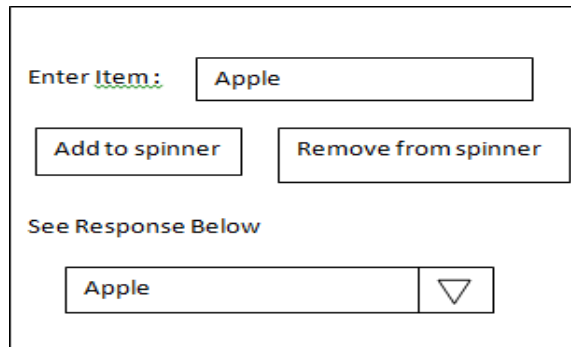
```
public boolean onOptionsItemSelected(MenuItem item) {  
    return MenuChoice(item);  
}
```

2. Press F11 to debug the application on the Android Emulator.

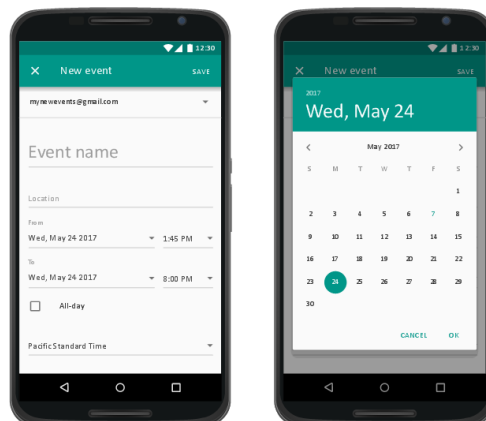
Lab Assignments

SET A

1. Create a custom "Contact" layout to hold multiple pieces of information, including: Photo, Name, Contact Number, E-mail id.
2. By using Spinner, Buttons. Write a program to draw following GUI.



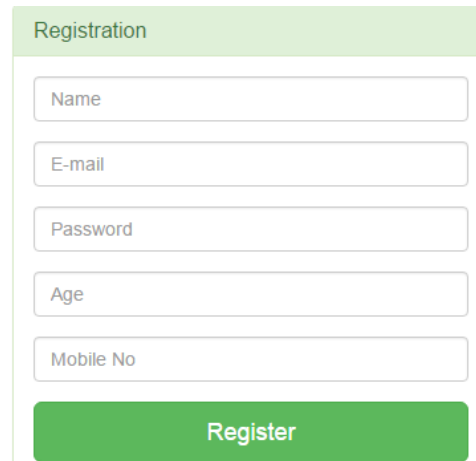
3. Create application to demonstrate date and time picker.



4. Construct an app that toggles a light bulb on and off when the user clicks on toggle button.
5. Construct an app to display the image on date wise.
6. Create a custom launcher icon.

SET B

1. Create application to demonstrate file explorer (Use ListView).
2. Create registration form given below. Also perform appropriate validation and display the message using dialog fragment.

A registration form with a light green header bar containing the title "Registration". Below the header, there are five text input fields stacked vertically, each with a placeholder label: "Name", "E-mail", "Password", "Age", and "Mobile No". At the bottom of the form is a green rectangular button with the text "Register" in white.

3. Construct image switcher using setFactory().
4. Construct a bank app to display different menu like windrow, deposit etc.

SET C

1. Create application to demonstrate alarm.
2. Construct App which shows Slides of multiple images after click on Slide menu.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Objectives

- How to use database in an application.
- How to create and execute database queries.
- How to send and receive messages.
- How to send E-mail from an application.

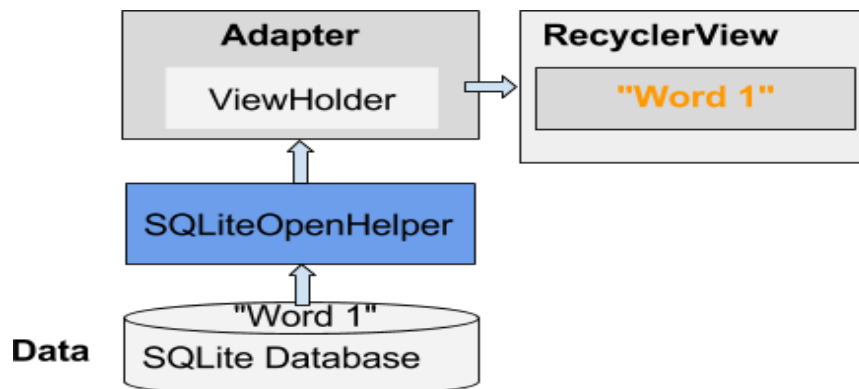
Reading

You should read the following topics before starting this exercise:

1. SQLite database
2. Classes used for SQLite database
3. Use of SmsManager and Intent
4. Concept of broadcast receiver.

Ready Reference

SQLite is an open source 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 ODBC.



- **SQLiteOpenHelper**

Android database `sqlite.SQLiteOpenHelper` manages database creation, up gradation, down gradation, version management and opening it. We need to create sub class of `SQLiteOpenHelper` and override `onCreate` and `onUpgrade` and optionally `onOpen`. If database is

not created, onCreate is called where we write script for database creation. If already created, then onOpen is called which opens database.

Constructors	Description
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	Create a helper object to create, open, and/or manage a database
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	Create a helper object to create, open, and/or manage a database.

Public Methods	Description
void close()	Close any open database object.
String getDatabaseName()	Return the name of the SQLite database being opened, as given to the constructor.
SQLiteDatabase getReadableDatabase()	Create and/or open a database.
SQLiteDatabase getWritableDatabase()	Create and/or open a database that will be used for reading and writing.
void onConfigure(SQLiteDatabase db)	Called when the database connection is being configured, to enable features such as write-ahead logging or foreign key support.
abstract void onCreate(SQLiteDatabase db)	Called when the database is created for the first time.
void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	Called when the database needs to be downgraded.
void onOpen(SQLiteDatabase db)	Called when the database has been opened.
void setIdleConnectionTimeout(long idleConnectionTimeoutMs)	Sets the maximum number of milliseconds that SQLite connection is allowed to be idle before it is closed and removed from the pool.
public synchronized void close ()	Closes the database object.

Example:

```
public class DBHelper extends SQLiteOpenHelper{

    public DBHelper(){
        super(context,DATABASE_NAME,null,1);
    }
    public void onCreate(SQLiteDatabase db){
        // Creates new database
        // Create the tables
        // execute query with the help of execSQL();

        // Add initial data
        .....
    }
    public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion){
    }
}
```

- **SQLite Database:** The main package is android.database.sqlite that contains the classes to manage your own databases:

1. 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. Its syntax is given below:

```
SQLiteDatabase mydatabase = openOrCreateDatabase("Database name",MODE_PRIVATE,null);
```

Public Methods	Description
<code>openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)</code>	This method only opens the existing database with the appropriate flag mode.
<code>openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)</code>	It is similar to the above method but it does not define any handler to handle the errors of databases
<code>openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)</code>	It not only opens but creates the database if it not exists. This method is equivalent to <code>openDatabase</code> method.

openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)	This method is similar to above method but it takes the File object as a path rather than a string. It is equivalent to file.getPath()
--	--

2. Database – Insertion

We can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

Example:

```
mydb.execSQL(—CREATE TABLE IF NOT EXISTS Login(Username varchar>Password
Varchar);l);
mydb.execSQL(—INSERT INTO Login VALUES(_admin‘,‘admin‘);l);
```

Another method that also does the same job but take some additional parameter is given below.

execSQL(String sql, Object[] bindArgs): This method not only inserts data, but also used to update or modify already existing data in database using bind arguments.

3. 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.

Example:

```
Cursor resultSet=mydb.rawQuery(—Select * from Loginl,null);
resultSet.moveToFirst();
String username= resultSet.getString(0);
String password= resultSet.getString(1);
```

Public Methods	Description
int getColumnCount()	Returns the total number of columns of the table.
int getColumnIndex(String columnName)	Returns the index number of a column by specifying the name of the column.

String getColumnName(int columnIndex)	Returns the name of the column by specifying the index of the column.
String[] getColumnNames()	Returns the array of all the column names of the table.
int getCount()	Returns the total number of rows in the cursor.
int getPosition()	Returns the current position of the cursor in the table.
boolean isClosed()	Returns true if the cursor is closed.

Insert Format: long insert (String table, String nullColumnHack, ContentValues values)

- First argument is the table name.
- Second argument is a String nullColumnHack
 - Workaround that allows you to insert empty rows
 - Use null
- Third argument must be a ContentValues with values for the row.
- Returns the id of the newly inserted item.

Example:

```
newId = mWritableDatabase.insert( TABLE_NAME, null, values);
```

Delete Format:

```
int delete (String table, String whereClause, String[] whereArgs)
```

- First argument is table name
- Second argument is WHERE clause
- Third argument are arguments to WHERE clause

Example:

```
deleted = mWritableDatabase.delete( WORD_LIST_TABLE, KEY_ID + "=?", new String[] {
String.valueOf(id) } );
```

Update Format:

```
int update(String table, ContentValues values, String whereClause, String[] whereArgs)
```

- First argument is table name
- Second argument must be ContentValues with new values for the row
- Third argument is WHERE clause
- Fourth argument are the arguments to the WHERE clause

Example:

```
ContentValues values = new ContentValues();

values.put(KEY_WORD, word);

mNumberOfRowsUpdated = mWritableDatabase.update( WORD_LIST_TABLE, values, // new values to insert
KEY_ID + " = ?", new String[]{String.valueOf(id)});
```

- **Sending SMS messages**

SMS messages can be send using SmsManager Class or Built-in SMS application

1. **Using SmsManager class:** Using the SmsManager class, you can send SMS messages from within your application without the need to involve the built-in Messaging application.

Steps to send SMS

1. Create a new Android project and name it SMS
2. Replace the TextView with the following statements in the **main.xml** file:

```
<Button android:id="@+id/btnSendSMS"

    android:layout_width="fill_parent"

    android:layout_height="wrap_content"

    android:text="Send SMS"

    android:onClick="onClick" />
```

3. In the AndroidManifest.xml file, add the following statements:

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

4. Add the following statements to the **SMSActivity.java** file:

```

import android.telephony.SmsManager;

import android.view.View;

public void onClick(View v) {

    sendSMS(-5556ll, -Hello my friends!!);

}
//---sends an SMS message to another device--

private void sendSMS(String phoneNumber, String message) {

    SmsManager sms = SmsManager.getDefault();

    sms.sendTextMessage(phoneNumber, null, message, null, null);

}

```

5. Debug the application on Android Emulator.
6. Click Send SMS button on first Android Emulator and see the message on second Emulator (5556).

Methods of SmsManager Class

Public Methods	Description
ArrayList<String> divideMessage(String text)	This method divides a message text into several fragments, none bigger than the maximum SMS message size.
static SmsManager getDefault()	This method is used to get the default instance of the SmsManager
void sendDataMessage (String destAddress , String scAddress, short destinationPort, byte[]	This method is used to send a data based SMS to a specific application port.
data, PendingIntent sentIntent, PendingIntent deliveryIntent)	

void sendMultipartTextMessage (String destAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList<PendingIntent> deliveryIntents)	Send a multi-part text based SMS.
void sendTextMessage(String destinationAddress, String scAddress, Stringtext, PendingIntent sentIntent, PendingIntent deliveryIntent)	Send a text based SMS.

2. Sending SMS messages using Intent: We can also use built-in messaging application to send the message.

Steps to send SMS

- 1) Create an Intent Object (Action to send SMS)

ACTION_VIEW action is used to launch an SMS client installed on your Android device.

```
Intent smsIntent = new Intent(Intent.ACTION_VIEW);
```

- 2) Using Intent Object set Data/Type to send SMS

smsto: is used as a URI to send message using setData() method.

data type is set to vnd.android-dir/mms-sms using setType() method.

```
smsIntent.setData(Uri.parse("smsto:"));
smsIntent.setType("vnd.android-dir/mms-sms");
```

- 3) Intent Object used to set a message for one or more phone number with putExtra() method and a message can be send to multiple receivers separated by _:'.

```
smsIntent.putExtra(-address||,new String(-0123456789;3398765587||)); OR
smsIntent.putExtra(-address||,||5556 ; 5553 ; 5566||);
```

3. Getting Feedback after Sending a Message: To check the status of the SMS message-sending process, two objects of PendingIntent are created. These objects are passed as last two arguments of the sendTextMessage() method

//---sends an SMS message to another device---

```
private void sendSMS(String phoneNumber, String message)
{
    String SENT = -SMS_SENT||;

    String DELIVERED = -SMS_DELIVERED||;

    PendingIntent sentPI = PendingIntent.getBroadcast(this, 0, new Intent(SENT), 0);
    PendingIntent deliveredPI = PendingIntent.getBroadcast(this, 0, new
Intent(DELIVERED), 0);

    //---when the SMS has been sent---

    registerReceiver(new BroadcastReceiver(){

        @Override

        public void onReceive(Context arg0, Intent arg1)
        {
            switch (getResultCode())
            {
                case Activity.RESULT_OK : Toast.makeText(getBaseContext(),
-SMSsent||, Toast.LENGTH_SHORT).show();

                                                                    break;
                case SmsManager.RESULT_ERROR_GENERIC_FAILURE
: Toast.makeText(getBaseContext(), -Generic failure||, Toast.LENGTH_SHORT).show();
                                                                    break;
                case SmsManager.RESULT_ERROR_NO_SERVICE:
Toast.makeText(getBaseContext(), -No service||, Toast.LENGTH_SHORT).show();

                                                                    break;
                case SmsManager.RESULT_ERROR_NULL_PDU:
Toast.makeText(getBaseContext(), -Null PDU||, Toast.LENGTH_SHORT).show();

                                                                    break;
            }
        }
    });
}
```

```

        break;
    }
}

}, new IntentFilter(SENT));

//---when the SMS has been delivered---
registerReceiver(new BroadcastReceiver()
{
    @Override
    public void onReceive(Context arg0, Intent arg1)
    {
        switch (getResultCode())
        {
            case Activity.RESULT_OK: Toast.makeText(getBaseContext(),
-SMS delivered, Toast.LENGTH_SHORT).show();
                                break;
            case Activity.RESULT_CANCELED:
Toast.makeText(getBaseContext(), -SMS not delivered, Toast.LENGTH_SHORT).show();
                                break;
        }
    }
}, new IntentFilter(DELIVERED));
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber, null, message, sentPI, deliveredPI);

```

4. Receiving SMS Messages: Using a BroadcastReceiver object, we can receive the SMS messages within the application.

Steps to receive SMS

1) Add the following statements in the xml file.

```

<receiver android:name = -.SMSReceiver>
    <intent-filter>
        <action android:name = -android.provider.Telephony.SMS_RECEIVED/>
    </intent-filter>
</receiver>

```


AND

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

2) Create SMSReceiver.java file in **src** folder of the project.

3) Create a java file SMSReceiver.java

```
package net.learn2develop.SMS;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;
public class SMSReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent) {
        //---get the SMS message passed in---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "";
        if (bundle != null) {
            //---retrieve the SMS message received---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i <msgs.length; i++)
            {
                if(i==0) {
                    //-- get the sender address/phone number --
                    str+=msgs[i].getOriginatingAddress();
                    str+=": -";
                }
            }
        }
    }
}
```

```

        //--get the message body--
        str += msg[i].getMessageBody().toString();
    }
    //--display the new SMS message--
    Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
    Log.d("SMSReceiver",str);
}
}
}

```

4) Debug the application

5) Using DDMS, send the message and application should receive it.

5. Sending E-mail: We can send Email by using Email/Gmail application available on Android. An Email account is configured by using POP3 or IMAP.

Steps to send Email

- 1) Create Android project and name it as Emails
- 2) Add the following statements in main.xml file:

```

<Button android:id="@+id/btnSendEmail" android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Send Email"
        android:onClick="onClick" />

```

- 3) Add the following statements in **MainActivity.java** file:

```

package net.learn2develop.Email;
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.net.Uri;
import android.view.View;
import android.widget.Button;

```

```

public class MainActivity extends Activity {
    Button btnSendEmail;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnSendEmail = (Button) findViewById(R.id.btnSendEmail);
        btnSendEmail.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String[] to = {-weimenglee@learn2develop.net,
- weimenglee@gmail.com};
                String[] cc = {- };
                sendEmail(to, cc, -Hello, -Hello my friends!);
            }
        });
    }
    //---sends an SMS message to another device---
    private void sendEmail(String[] emailAddresses, String[] carbonCopies, String subject,
String message)
    {
        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setData(Uri.parse(-mailto:));
        String[] to = emailAddresses;
        String[] cc = carbonCopies;
        emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
        emailIntent.putExtra(Intent.EXTRA_CC, cc);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
        emailIntent.putExtra(Intent.EXTRA_TEXT, message);
        emailIntent.setType(-message/rfc822);
    }
}

```

```
        startActivity(Intent.createChooser(emailIntent, -Email));  
    }  
}
```

- 4) Test the application

Lab Assignments

SET A

1. Create table Company (id, name, address, phno). Create Application for Performing the following operation on the table.

- i) Insert New Company Details.
- ii) Show All the Company Details.

2. Create table Student (sno , s_name,s_class,s_addr)
Teacher (tno, t_name, qualification, experience)

Student-Teacher has many to many relationship.

Using above database Write Application to accept a teacher name from user and display the names of students along with subjects to whom teacher is teaching.

3. Create Following Table:

Emp (emp_no,emp_name,address,phone,salary)

Dept (dept_no,dept_name,location)

Emp-Dept is related with one-many relationship.

Create application for performing the following Operation on the table

- 1) Add Records into Emp and Dept table.
- 2) Accept Department name from User and delete employee information which belongs to that department.

4. Create application to send and receive messages using SMSManager.
5. Create application to send email.
6. Create application to send message. After sending message display delivery report of message.

SET B

1. Create sample application with login module (Check username and password). On successful login, pass username to next screen And on failing login, alert user using Toast (Hint :Use Login(username, password) Table.)

2. Create Table project (pno, p_name, ptype, duration) and employee (id, e_name, qualification, joindate)
Project – employee have many to many relationship.
Using database perform following operation.
 - 1) Add new record into table.
3. Accept a project name from user and display information of employees working on the project.
4. Create application to design login form, validate it. Write and send email with appropriate message.
5. Create application to send SMS with image and contact as attachment.

SET C

- 1) Create simple application shown below. Create table Student(Sid, Sname, phno). Use autoincrement for Sid and Perform following Operation.
 - a. Add Student and display its information.
 - b. Delete Student.

- 2) Create application to send email with attachment.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Objectives

- Study the location based services in android
- Use various operations of Google Map

Reading

You should read the following topics before starting this exercise:

1. Concept of Location Based Services and Google Map.
2. Use of Google API.
3. Concept of Geocoding and Monitoring a Location.

Ready Reference

Location based services: –Location-based services is used to create an application which uses current location, location updates, and location information. The two main LBS elements are:

- **Location Manager** — these services allow applications to obtain periodic updates of the device's geographical location.
- **Location Providers** — provides periodic reports on the geographical location of the device.

Google Map: **Google Maps** is web based service developed by Google. It provides many facilities such as Satellite view, Street view, real time traffic condition, and navigations for travelling by foot, car, bicycle and public transportation. To use Google Maps in your Android applications programmatically.

Google API (Application Programming Interface)

Google APIs is a set of application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Your application needs an API key to access the Google Maps servers. The key is free. You can use it with any of your applications that call the Google Maps Android API.

Steps to create Google Map Application

Step 1: Start Android Studio

Step 2: Install the Google Play Services SDK.

- Click **Tools** in the Android Studio menu bar, then **Android, SDK Manager**.
- Click **SDK Tools**, and if Google Play Services is not installed, install Google Play Services.

Step 3: Create a Google Maps Project

1. Create a new project as follows:
 - a. If you see the **Welcome to Android Studio** dialog, choose **Start a new Android Studio project**, available under 'Quick Start' on the right of the dialog.
 - b. Otherwise, click **File** in the Android Studio menu bar, then **New, New Project**.
2. Enter your app name, company domain, and project location, as prompted. Then click **Next**.
3. Select the form factors you need for your app. If you're not sure what you need, just select **Phone and Tablet**. Then click **Next**.
4. Select **Google Maps Activity** in the 'Add an activity to Mobile' dialog. Then click **Next**.
5. Enter the activity name, layout name and title as prompted. The default values are fine. Then click **Finish**.

Step 4: Get Google Maps API Key

- Use the link provided in the *google_maps_api.xml* file.
- Copy the link provided in the *google_maps_api.xml* file and paste it into your browser. The link takes you to the Google API Console and supplies the required information to the Google API Console via URL parameters.
- Follow the instructions to create a new project on the Google API Console or select an existing project.
- Create an Android-restricted API key for your project.
- Copy the resulting API key, go back to Android Studio, and paste the API key into the `<string>` element in the *google_maps_api.xml* file.

Step 5: Examine the code supplied by the template.

Step 6: Build an APK file - Click **Build** in the Android Studio menu bar, then **Build APK**.

Step 7: install APK file on real device and open the application.

- **Displaying the Map**
- **Classes and Interface used for displaying Map are given below:**

- **Interfaces:**

- **Package Name: com.google.android.gms.maps**

OnMapReadyCallback – This interface is used to execute method when Map is ready.

Public Method	Description
abstract void onMapReady (GoogleMap googleMap)	This method execute automatically when map is ready

Classes:

Package Name: android.app

FragmentManager: FragmentManager is a class used to create transactions for adding, removing or replacing fragments. It used to interact with Fragment inside the activity.

Public Methods	Description
Fragment findFragmentById(int id)	Finds a fragment that was identified by the given id.
Fragment findFragmentByTag(String tag)	Finds a fragment that was identified by the given tag.

GoogleMap: This is the main class of the Google Maps Android API. You cannot instantiate a GoogleMap object directly, rather, you must obtain one from the getMapAsync() method on a MapFragment. This class provides methods to set type of map, add markers, add polyline or move camera etc.

Constants	Description
MAP_TYPE_HYBRID	Satellite maps with a transparent layer of major streets.
MAP_TYPE_NONE	No base map tiles.
MAP_TYPE_NORMAL	Basic maps.

MAP_TYPE_SATELLITE	Satellite maps with no labels.
MAP_TYPE_TERRAIN	Terrain maps (Topographic data).

Public Method	Description
Circle addCircle(CircleOptions options)	Used to add a circle on the map.
Marker addMarker(MarkerOptions options)	Used to add a marker on the map.
Polyline addPolyline (PolylineOptions options)	Used to add a polyline on the map.
void animateCamera(CameraUpdate update)	Animates the movement of the camera from the current position to the position defined in the update.
void clear()	Removes all circles, markers, polylines, polygons, overlays from the map.
CameraPosition getCameraPosition()	Returns the current position of the camera on the map.
int getMapType()	Returns the type of map.
UiSettings getUiSettings()	Returns the user interface settings for the map.
void moveCamera(CameraUpdate update)	Used to move the camera to the position specified by update.
void setMapType(int type)	Sets the type of map.
void setMyLocationEnabled (boolean enabled)	Enables or disables the my-location layer.

SupportMapFragment: SupportMapFragement is used to create a map fragment.

Public Methods	Description
void getMapAsync (OnMapReadyCallback callback)	This method called when map is ready.
static SupportMapFragment newInstance()	Creates a map fragment, using default options.

- By default, the XML file that defines the app's layout is at *res/layout/activity_maps.xml*. It contains the following code:

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.admin.googlemapdemo.MapsActivity"
    />
```

The MapsActivity Java file

```
Package com.example.admin.googlemapdemo;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment = (SupportMapFragment)
            getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
    public void onMapReady(GoogleMap googleMap) {

        mMap = googleMap;
    }
}
```

- **Changing Views:** Google Maps is displayed in map view, which basically drawings of streets and places of interest.

To display the satellite view add following line in `onMapReady()` method in the *MapsActivity.java* file

```
mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
```

- **Getting Location Data:** To know the information of a location we need to find latitude and longitude values corresponding to the position. Converting location name to respective latitude and longitude values is called as geocoding. To do the reverse geocoding we need to use Geocoder class.

Package Name: android

Manifest.permission: It is static nested class of Manifest. It contains the constants to add user permission or security settings in android application.

Constants	Description
String ACCESS_FINE_LOCATION	Used to access precise location.
String ACCESS_COARSE_LOCATION	Used to access approximate location.
String ACCESS_NETWORK_STATE	Used to access the state of the network.
String ACCESS_WIFI_STATE	Used to access the state of Wi-Fi.
String CHANGE_NETWORK_STATE	Used to change network connectivity state.
String INTERNET	Used to open network sockets.
String WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.

Add below line in the manifests/AndroidManifest.xml file to add permission

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

Package Name: com.google.android.gms.maps

CameraUpdateFactory: This class is used to create CameraUpdate objects that change or move a map's camera. It also provides the functionality such as Zoom In, Zoom Out and Zoom By specific amount.

Public Methods	Description
static CameraUpdate newCameraPosition (CameraPosition cameraPosition)	Returns a CameraUpdate that moves the camera to a specified CameraPosition.
static CameraUpdate newLatLng(LatLng latLng)	Returns a CameraUpdate that moves the center of the screen to a latitude and longitude specified by a LatLng object.
static CameraUpdate newLatLngZoom(LatLng latLng, float zoom)	Returns a CameraUpdate that moves the center of the screen to a latitude and longitude specified by a LatLng object, and moves to the given zoom level.
static CameraUpdate zoomIn()	Returns a CameraUpdate that zooms in on the map.
static CameraUpdate zoomOut()	Returns a CameraUpdate that zooms out on the map.

Package Name: com.google.android.gms.maps.model

LatLng – This class represents a pair of latitude and longitude coordinates.

Constructor	Description
LatLng(double latitude, double longitude)	Constructs a LatLng with the given latitude and longitude, measured in degrees.

Fields	Description
public final double latitude	Latitude, in degrees.
public final double longitude	Longitude, in degrees.

Marker: This class is used to mark particular point on Map. This class contains methods to set opacity, title, rotation, position, or visibility of marker.

Properties	Description
Alpha	Sets the opacity of the marker. Defaults to 1.0.
Position	The LatLng value for the marker's position on the map.
Title	Sets the title and display when the user taps the marker.

Icon	An image or logo that's displayed for the marker.
Visibility	Set visibility of marker. By default, the marker is visible.
Rotation	The rotation of the marker in degrees clockwise about the marker's anchor point.
zIndex	The draw order for the marker. The default value is 0.

Methods	Description
float getAlpha()	Returns the alpha of the marker.
String getId()	Returns the marker's id.
LatLng getPosition()	Returns the position of the marker.
float getRotation()	Return the rotation of the marker.
String getTitle()	Return the title of the marker.
float getZIndex()	Returns the zIndex of the marker.
void setPosition(LatLng latlng)	Sets the location of the marker.
void remove()	Removes this marker from the map.
void setAlpha(float alpha)	Sets the alpha (opacity) of the marker.
void setTitle(String title)	Sets the title of the marker.
void setIcon(BitmapDescriptor iconDescriptor)	Sets the icon for the marker.
void setRotation(float rotation)	Sets the rotation of the marker in degrees clockwise about the marker's anchor point.

MarkerOptions: This class defines different options available for a marker.

Constructor	Description
MarkerOptions()	Creates a new set of marker options.

Public Methods	Description
MarkerOptions icon(BitmapDescriptor iconDescriptor)	Sets the icon for the marker.
MarkerOptions position(LatLng latlng)	Sets the location for the marker.
MarkerOptions rotation(float rotation)	Sets the rotation of the marker in degrees clockwise about the marker's anchor point.

MarkerOptions title(String title)	Sets the title for the marker.
MarkerOptions visible(boolean visible)	Sets the visibility for the marker.

Package Name: android.location

Criteria: A class indicating the application criteria for selecting a location provider. Providers maybe ordered according to accuracy, power usage, ability to report altitude, speed, and bearing, and monetary cost.

Constructors	Description
Criteria()	Constructs a new Criteria object.
Criteria(Criteria criteria)	Constructs a new Criteria object that is a copy of the given criteria.

Public Methods	Description
int getAccuracy()	Returns accuracy of location.
int getSpeedAccuracy()	Returns a constant indicating the desired speed accuracy.
void setAccuracy(int accuracy)	Sets accuracy for latitude and longitude.
void setSpeedAccuracy(int accuracy)	Sets speed accuracy.

Address: A class representing an Address. An address contains methods to find the information such as Country name, Postal Code, locality etc.

Constructor	Description
Address(Locale locale)	Constructs a new Address object set to the given Locale.

Public Methods	Description
String getAddressLine(int index)	Returns a line of the address numbered by the given index, or null if no such line is present.
String getCountryCode()	Returns the country code of the address, or null if it is unknown.
String getCountryName()	Returns the localized country name of the address, or null if it is unknown.
double getLatitude()	Returns the latitude of the address.

double getLongitude()	Returns the longitude of the address.
void setAddressLine(int index, String line)	Sets the line of the address numbered by index (starting at 0) to the given String.
void setCountryName(String countryName)	Sets the country name of the address to the given String, which may be null.
void setLatitude(double latitude)	Sets the latitude associated with this address.
void setLongitude(double longitude)	Sets the longitude associated with this address.

Geocoder: A class for handling geocoding and reverse geocoding. Geocoding is the process of transforming an address of a location into a (latitude, longitude) coordinates. Reverse geocoding is the process of transforming a (latitude, longitude) coordinate into an address.

Constructors	Description
Geocoder (Context context)	Constructs a Geocoder whose responses will be localized for the default system Locale.
Geocoder (Context context, Locale locale)	Constructs a Geocoder whose responses will be localized for the given Locale.

Public Methods	Description
List<Address> getFromLocation(double latitude, double longitude, int maxResults)	Returns an array of Addresses that are known to describe the area the by given latitude and longitude.
List<Address> getFromLocationName(String locationName, int maxResults)	Returns an array of Addresses that are known to describe the named location.

LocationManager: This class provides access to the system location services. These services allow applications to obtain periodic updates of the device's geographical location.

Constants	Description
String GPS_PROVIDER	Name of the GPS location provider.
String NETWORK_PROVIDER	Name of the network location provider.

Public Methods	Description
boolean isProviderEnabled(String provider)	Returns the current enabled status of the given provider.
void requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener listener)	Register for location updates using the named provider, and a pending intent.
String getBestProvider(Criteria criteria, boolean enabledOnly)	Returns the name of the provider that best meets the given criteria.
Location getLastKnownLocation(String provider)	Returns a Location indicating the data from the last known location obtained from the given provider.

Example:

1. Program to display current location of android device on the Map

- a. Add below line in the *manifests/AndroidManifest.xml* file

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

- b. To display the current location add following line in onMapReady() method in the *MapsActivity.java* file

```

LatLng pune = new LatLng(18.531387, 73.849467);
mMap.addMarker(new MarkerOptions().position(pune).title("Pune"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(pune));

if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION ) ==
PackageManager.PERMISSION_GRANTED || ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION ) ==
PackageManager.PERMISSION_GRANTED) {

    mMap.setMyLocationEnabled(true);
}

```


2. Navigating to specific location – You can set Google Maps to display a particular location by getting the latitude and longitude values.

To display the specific location add following line in `onMapReady()` method in the *MapsActivity.java* file

```
LatLng pune = new LatLng(18.531387, 73.849467);  
mMap.addMarker(new MarkerOptions().position(pune).title("Pune"));  
mMap.moveCamera(CameraUpdateFactory.newLatLng(pune));
```

3. To display the information about the location using geocoding add following line in `onMapReady()` method in the *MapsActivity.java* file

```
List<Address> addresses = null;  
String locationName = "Shaniwar Wada Fort"; Geocoder geocoder =  
new Geocoder(this);  
try {  
    addresses = geocoder.getFromLocationName(locationName,1);  
} catch (IOException e)  
    {e.printStackTrace()  
    }  
  
Address address = addresses.get(0);  
LatLng shaniwar_wada= new LatLng(address.getLatitude(), address.getLongitude());  
String addressline = addresses.get(0).getAddressLine(0);  
  
String city = addresses.get(0).getLocality();  
String state =  
addresses.get(0).getAdminArea();  
  
String postalCode = addresses.get(0).getPostalCode();  
String title = addressline + "\n" + city + "\n" + state + "\n" + postalCode;  
mMap.addMarker(new  
MarkerOptions().position(shaniwar_wada).title(title));  
mMap.moveCamera(CameraUpdateFactory.newLatLng(shaniwar_wada));  
Toast.makeText(getBaseContext(),title,Toast.LENGTH_LONG).show();
```

Lab Assignments

SET A

1. Write a program to find the specific location of an Android device and display details of the place like Address line, city with Geocoding.
2. Write a program to search a specific location on Google Map.
3. Write a program to perform Zoom In, Zoom Out operation and display Satellite view, Terrain view of current location on Google Map.

SET B

1. Write a program to calculate distance between two locations on Google Map.
2. Write a program to modify the above program to draw the path along a route on Google Map.

SET C

1. Write a program to track an android device using mobile number and display the location on Google Map.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []