CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test I – August 2023

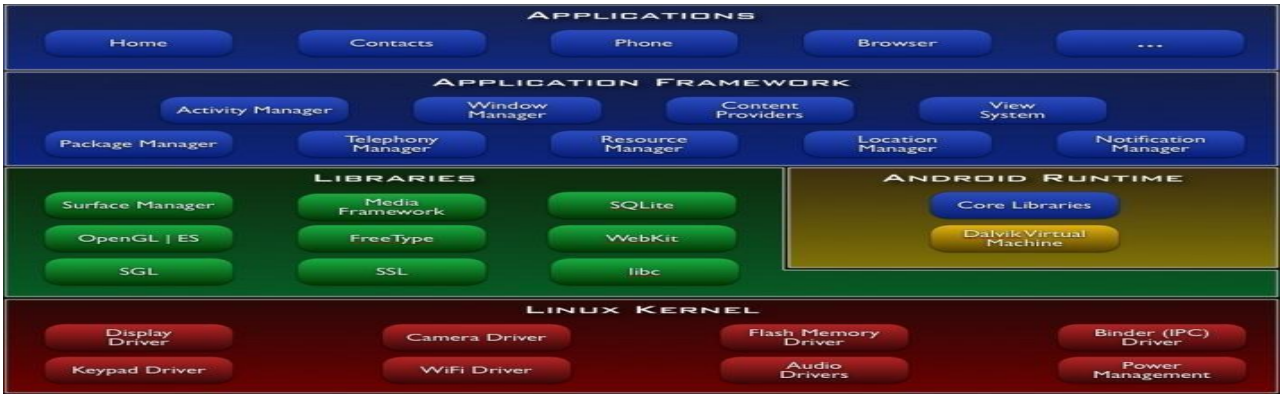
Sub:	Mobile Applications							Sub Code:	22MCA263
Date:	3/8/2023	Duration:	90 min's	Max Marks:	50	Sem:	II	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE	
			CO	RBT
1	Explain the Android Software Stack in detail with a neat diagram. **OR**	[10]	CO1	L2
2	Devise a mobile app to create a login form and display the message accordingly(if user name and password is "CMR" it should display "Login successful" otherwise "invalid login"	[10]	CO3,CO4, CO5	L4
	PART II			
3	What is Android? Describe the features of Android 4.1 Jelly Bean? **OR**	[10]	CO1	L2
4	Develop a mobile app to display 2 images on screen and toggle each image if they click.	[10]	CO3,CO4, CO5	L4
	PART III			
5	Describe the anatomy of mobile app? **OR**	[10]	CO1	L2
6	Develop a UI in mobile app to demonstrate the attributes of Linear Layout.	[10]	CO3, CO4, CO5	L3
	PART IV			
7	What are the different attributes of relative layout? Explain with an example. **OR**	[10]	CO2	L2
8	Develop a Mobile App to create a Text View and change the attributes from XML and Java File	[10]	CO3, CO4, CO5	L3
	PARTV			
9	With an example program explain the differences between Table Layout and Grid Layout. **OR**	[10]	CO1, CO2	L2
10	How do you create the First Android Project? Illustrate the steps with an example?	[10]	CO1, CO3	L2

1) Explain Android Software Stack OR Explain Android architecture

The Android software stack consists of a Linux kernel and a collection of C/C++ libraries that are exposed through an application framework for application development.
The Android software stack consists of four main layers as shown in diagram.



The following list gives a brief description of each layers in the software stack:

→ Linux kernel : The kernel on which Android is based contains device drivers for various hardware components of an Android device, including Display, Camera, keypad, wi-fi, Flash Memory and Audio.

→ Libraries : The next layer on top of the Linux kernel is the libraries that implement different Android features. A few of these libraries are listed here:

- Webkit library - Responsible for browser support
- Freetype library - Responsible for font support
- SQLite library - Provides database support
- Media library - Responsible for recording and playback of audio and video formats.
- Surface Manager library - Provides graphics libraries that include SGL and OpenGL for 2D and 3D graphics support.

→ Android runtime: The engine in the same layer as the libraries. It provides a set of core Android libraries and a Dalvik virtual machine that enable developers to write Android applications using java. The core Android libraries provide most of the functionality available in the core Java libraries, as well as the Android-specific libraries. Dalvik VM is explained in detail later in this chapter.

→ Application framework - Provides the classes that enable application developers to develop Android applications. It manages the user interface, application resources, and abstraction for hardware access

→ Application layer - Displays the application developed and downloaded by users, along with the built-in applications provided with the Android device itself.

2) Devise a mobile app to create a login form and display the message accordingly(if user name and password is "CMR" it should display "Login successful" otherwise "invalid login"

XML File

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.relativelayout2.MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="84dp"
        android:layout_marginTop="14dp"
        android:text="Sign In" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
```

```
    android:layout_below="@+id/textView1"
    android:layout_marginLeft="15dp"
    android:layout_marginTop="48dp"
    android:text="UserId" />

<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/textView2"
    android:layout_alignBottom="@+id/textView2"
    android:layout_toRightOf="@+id/textView1"
    android:ems="10" >

    <requestFocus />
</EditText>

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/textView2"
    android:layout_below="@+id/editText1"
    android:layout_marginTop="39dp"
    android:text="Password" />

<EditText
    android:id="@+id/editText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/textView3"
    android:layout_alignBottom="@+id/textView3"
    android:layout_toRightOf="@+id/textView1"
    android:ems="10" />

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="Sign In" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button1"
    android:layout_marginTop="52dp"
    android:layout_toRightOf="@+id/textView3"
    android:text="TextView" />
```

```
</RelativeLayout>
```

Java Code

```
package com.example.relativelayout2;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;


public class MainActivity extends Activity implements OnClickListener{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button b=(Button)findViewById(R.id.button1);
        b.setOnClickListener(this);
    }
    public void onClick(View v){
            EditText userid=(EditText)findViewById(R.id.editText1);
            EditText password=(EditText)findViewById(R.id.editText2);
            TextView resp=(TextView)findViewById(R.id.textView4);
            String usr=userid.getText().toString();
            String pwd=password.getText().toString();
            if(usr.trim().length()==0 ||pwd.trim().length()==0 ){
                    String str="left blank";
                    resp.setText(str);
            }
            else{
                    if(usr.equals("cmr")&&pwd.equals("cmr"))
                            resp.setText("welcome");
                    else
                            resp.setText("invalid login");
            }
    }


}
```

Q3) What is Android? Discuss features of Android 4.1 Jelly Bean SDK

Android is Google's open source and free Java-based platform for mobile development. It enables developers to build real-world mobile applications using the Android software development kit (SDK) & publish them to the Android market.
Android comes with several application programming Interfaces (APIs) that make the task of developing full-featured applications easy. You can even use a camera, accelerometer, or GPS in an Android application.
Android is cross-compatible - it can run on Android phone/tablet/devices of different screen sizes and resolutions. Using Android, you can develop applications for a wide variety of devices, including phones, e-book readers, netbooks and GPS units.
Android was initially developed by Android, Inc., a small Palo Alto, California, company. Google bought this company in July 2005.

→ Project Butter: Makes the Jelly Bean UI faster & more responsive. Also CPU Touch Responsiveness is added, which increases CPU performance whenever the screen is touched. It uses the finger's speed and direction to predict where it will be located after some milliseconds, hence making the navigation faster.

→ Faster speech recognition: Speech recognition is now faster and doesn't require any network to convert voice into text. That is, users can dictate to the device without an internet connection.

→ Improved notification system :- Besides text, the notifications include pictures and lists too. Notifications can be expanded or collapsed through a variety of gestures, and users can block notifications if desired. The notifications also include action buttons that enable users to call directly from the notification menu rather replying to email.

→ Supports new languages :- Jelly Bean includes support for several languages including Arabic, Hebrew, Hindi & Thai. It also supports bidirectional text.

→ Predictive keyboard - On the basis of the current context, the next word of the message is automatically predicted.

→ Auto-arranging Home screen - Icons and widgets automatically resize and realign as per the existing space.

→ Helpful for visually impaired users - The Gesture mode combined with voice helps visually impaired users to easily navigate the user interface.

→ Improved camera app - The Jelly Bean Camera app includes a new review mode of the captured photos. Users can swipe in from the right of the screen to quickly view the captured photos. Also, users can pinch to switch to a new film strip view, where they can swipe to delete photos.

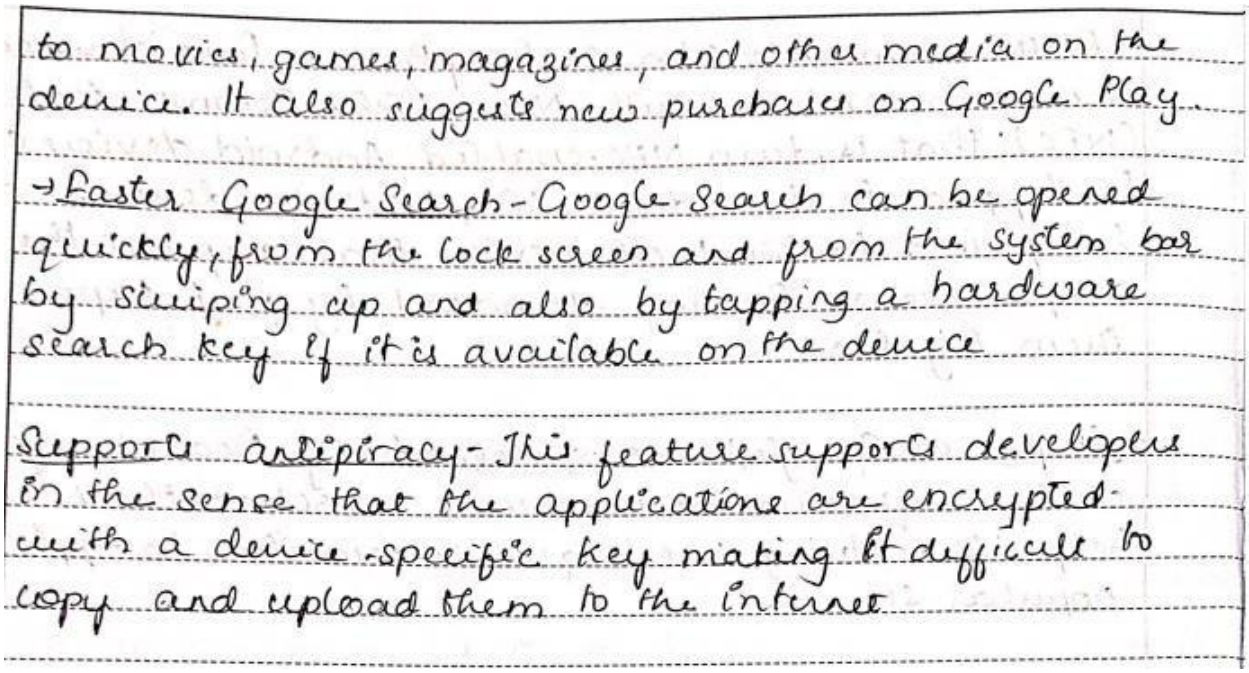
→ Better communication in Jelly Bean - Two devices can communicate with Near Field Communication (NFC); that is, two NFC-enabled Android device can be tapped to share data. Also, Android devices can be paired to Bluetooth devices that support the simple Secure Pairing standard by just tapping them together.

→ Improved Google Voice search - Jelly Bean is equipped with a question and answer search method that helps in solving user's queries similar to Apple's popular Siri.

→ Face Unlock - Unlocks the device when the user looks at it. It also prevents the screen from blacking out. Optionally "blink" can be used to confirm that a live person is unlocking the device instead of a photo.

→ Google Now - Provides users "just the right information at just the right time". It displays cards to show desired information automatically. For example, the Places card displays nearby restaurants and shops while moving; the Transit card displays information on the next train or bus when the user is near a bus stop or railway station; the Sports card displays live scores or upcoming game events; the Weather card displays the weather conditions at a user's current location and so on.

→ Google Play widgets - Provides quick and easy access


to movies, games, magazines, and other media on the device. It also suggests new purchases on Google Play.

→ Faster Google Search - Google Search can be opened quickly, from the lock screen and from the system bar by swiping up and also by tapping a hardware search key if it is available on the device

Supports antipiracy - This feature supports developers in the sense that the applications are encrypted with a device-specific key making it difficult to copy and upload them to the Internet

4) Develop a mobile app to display 2 images on screen and toggle each image if they click.

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

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="456dp"
        android:src="@drawable/logo" />

</FrameLayout>
```

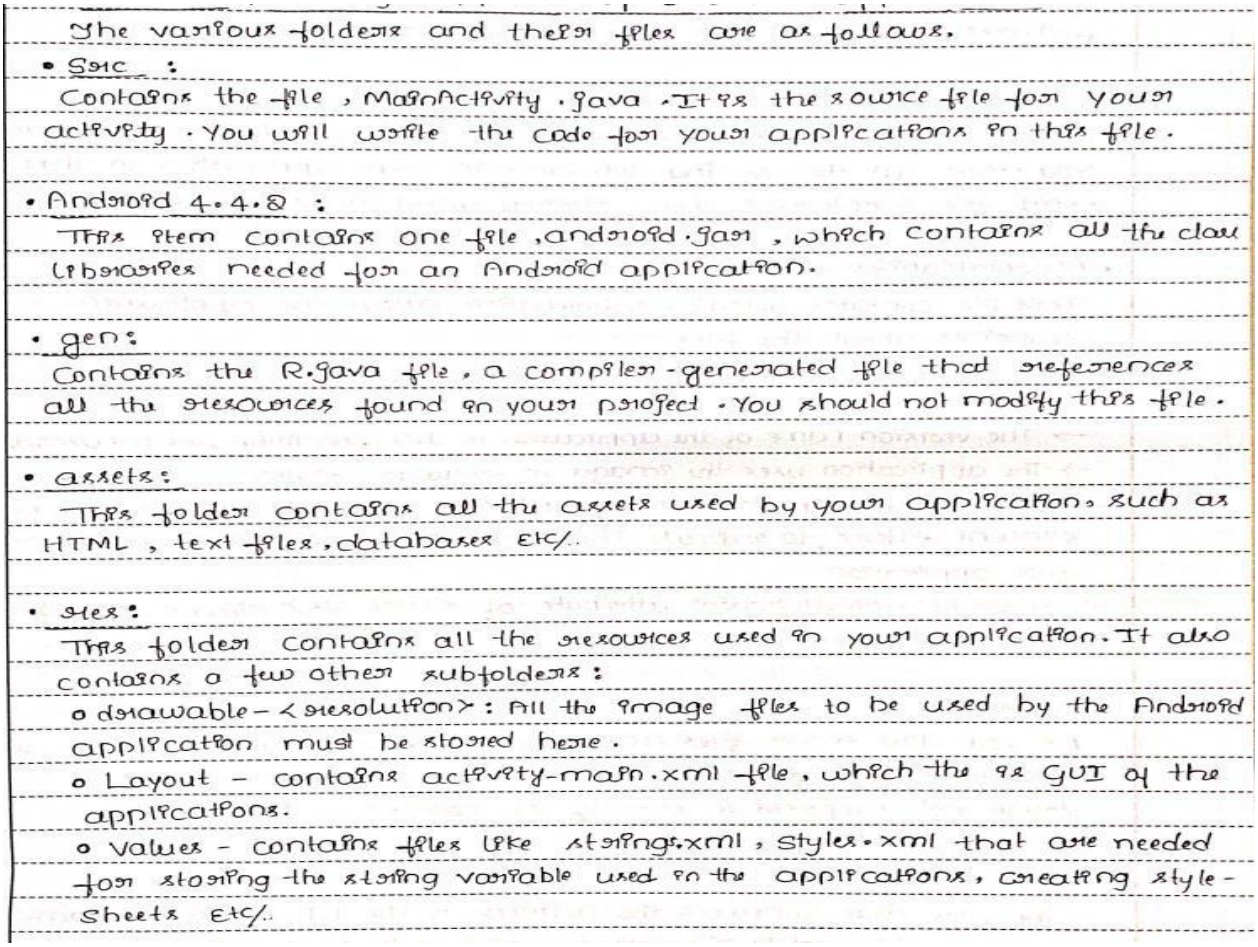
Java Code

```
package com.example.framelayoutnew;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;
```

```
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageView;


public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final ImageView i1=(ImageView)findViewById(R.id.imageView1);
        final ImageView i2=(ImageView)findViewById(R.id.imageView2);
        i1.setOnClickListener(new OnClickListener(){
            public void onClick(View view){
                i2.setVisibility(View.VISIBLE);
                view.setVisibility(View.GONE);
            }
        });
            i2.setOnClickListener(new OnClickListener(){
                public void onClick(View view){
                i1.setVisibility(View.VISIBLE);
```

```
        view.setVisibility(View.GONE);
        }
    });

}
}
```

5. Describe the anatomy of mobile app?



The various folders and their files are as follows.

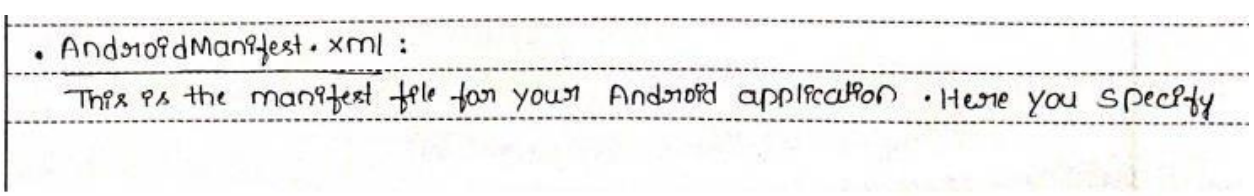
- **Src :**
  Contains the file, MainActivity.java. It is the source file for your activity. You will write the code for your applications in this file.

- **Android 4.4.2 :**
  This item contains one file, android.jar, which contains all the class libraries needed for an Android application.

- **gen:**
  Contains the R.java file, a compiler-generated file that references all the resources found in your project. You should not modify this file.

- **assets:**
  This folder contains all the assets used by your application, such as HTML, text files, databases etc.

- **res:**
  This folder contains all the resources used in your application. It also contains a few other subfolders:
  - drawable-<resolution>: All the image files to be used by the Android application must be stored here.
  - Layout - contains activity-main.xml file, which is the GUI of the applications.
  - values - contains files like strings.xml, styles.xml that are needed for storing the string variable used in the applications, creating style-sheets etc.



- **AndroidManifest.xml :**
  This is the manifest file for your Android application. Here you specify

the permissions needed by your application, as well as other features (such as intent-filters, receivers, etc.)

- Strings.xml file:

The activity_main.xml file defines the user interface for your activity. You store all the string constants in your application in this strings.xml file & reference these strings using @string identifier.

- AndroidManifest.xml file:

This file contains detailed information about the application,

Keypoints about the file:

→ It defines the package name of application

→ The version code of application is 1.

→ The version name of the application is 1.0, mainly used for user display

→ The application uses the images in drawable folder.

→ Within the definition for the activity, there is an Element named <intent-filter> to indicate this activity serves as the entry point for application.

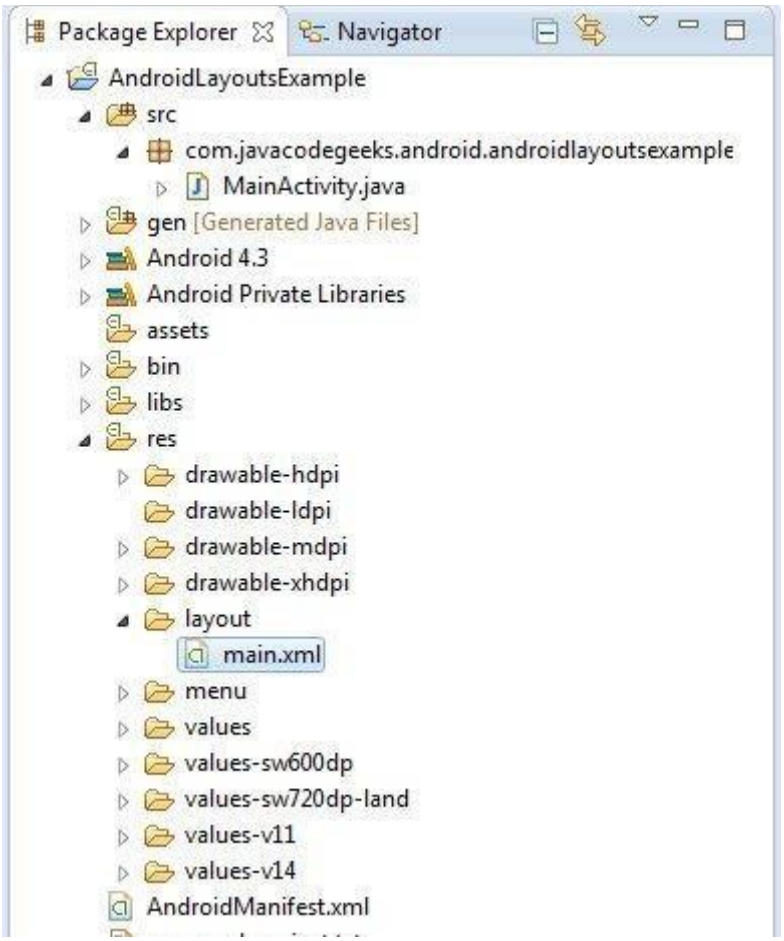
→ android:minSdkVersion attribute of <uses-sdk> Element specifies the minimum version of OS.

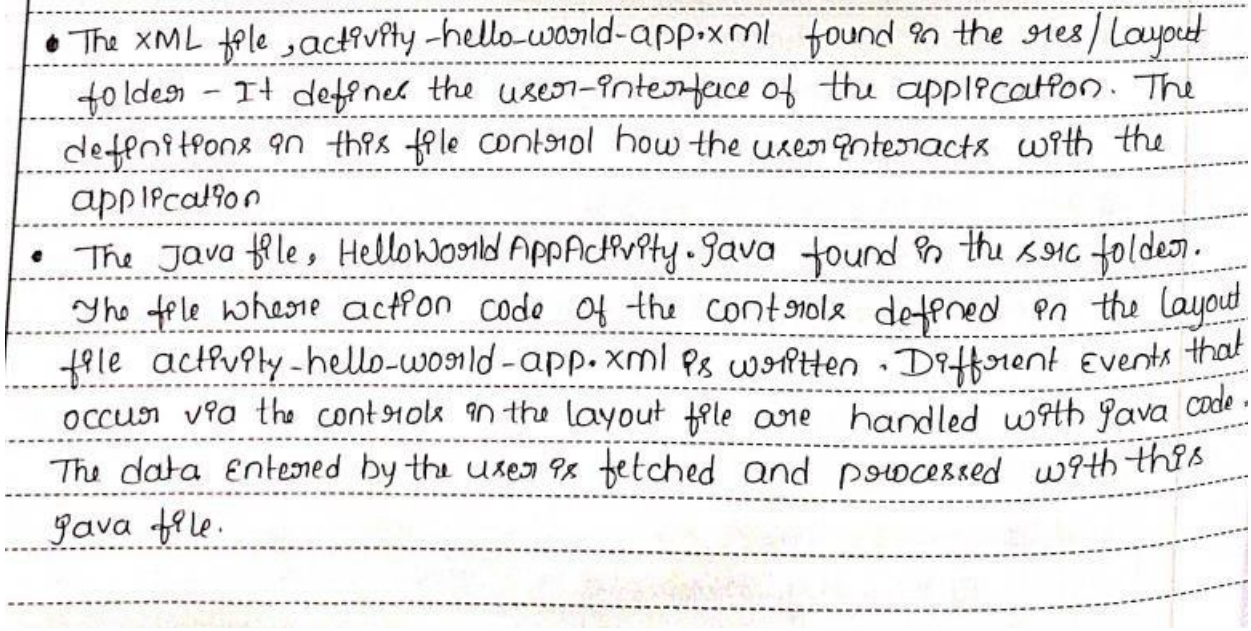
- R.java File:

As you add more files and folders to your project, Eclipse will automatically generate the contents of R.java.
You're not supposed to modify the content of R.java file.

- MainActivity.java File:

The code that connects the activity to the UI (activity_main.xml) is the setContentView() method, which is in MainActivity.java.


Package Explorer
Navigator
AndroidLayoutsExample
src
com.javacodegeeks.android.androidlayoutsexample
MainActivity.java
gen [Generated Java Files]
Android 4.3
Android Private Libraries
assets
bin
libs
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
drawable-xhdpi
layout
main.xml
menu
values
values-sw600dp
values-sw720dp-land
values-v11
values-v14
AndroidManifest.xml



6. Develop a UI in Mobile App to demonstrate the attributes of Linear Layout.

```
?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <!-- Add vertical in the android:orientation-->

    <!-- Add Button-->
    <Button
        android:layout_width="match_parent"
        android:layout_margin="10dp"
        android:layout_height="wrap_content"/>

    <!-- Add Button-->
    <Button
        android:layout_width="match_parent"
        android:layout_margin="10dp"
        android:layout_height="wrap_content"/>

    <!-- Add Button-->
    <Button
        android:layout_width="match_parent"
        android:layout_margin="10dp"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

7. What are the different attributes of relative layout? Explain with an example.

Following are the important attributes specific to RelativeLayout −

Sr.No.	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the layout.

2	**android:gravity** This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
3	**android:ignoreGravity** This indicates what view should not be affected by gravity.

Sr.No.	Attribute & Description
1	**android:layout_above** Positions the bottom edge of this view above the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name"
2	**android:layout_alignBottom** Makes the bottom edge of this view match the bottom edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
3	**android:layout_alignLeft** Makes the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
4	**android:layout_alignParentBottom** If true, makes the bottom edge of this view match the bottom edge of the parent. Must be a boolean value, either "true" or "false".
5	**android:layout_alignParentEnd** If true, makes the end edge of this view match the end edge of the parent. Must be a boolean value, either "true" or "false".
6	**android:layout_alignParentLeft** If true, makes the left edge of this view match the left edge of the parent. Must be a boolean value, either "true" or "false".
7	**android:layout_alignParentRight** If true, makes the right edge of this view match the right edge of the parent. Must be a boolean value, either "true" or "false".

8	**android:layout_alignParentStart** If true, makes the start edge of this view match the start edge of the parent. Must be a boolean value, either "true" or "false".
9	**android:layout_alignParentTop** If true, makes the top edge of this view match the top edge of the parent. Must be a boolean value, either "true" or "false".
10	**android:layout_alignRight** Makes the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
11	**android:layout_alignStart** Makes the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
12	**android:layout_alignTop** Makes the top edge of this view match the top edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
13	**android:layout_below** Positions the top edge of this view below the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
14	**android:layout_centerHorizontal** If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".
15	**android:layout_centerInParent** If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".
16	**android:layout_centerVertical** If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".

17	**android:layout_toEndOf** Positions the start edge of this view to the end of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
18	**android:layout_toLeftOf** Positions the right edge of this view to the left of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
19	**android:layout_toRightOf** Positions the left edge of this view to the right of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
20	**android:layout_toStartOf** Positions the end edge of this view to the start of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="Button1" />
    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:text="Button2" />
    <Button
        android:id="@+id/btn3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:text="Button3" />

    <Button
        android:id="@+id/btn4"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Button4" />
    <Button
        android:id="@+id/btn5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btn2"
        android:layout_centerHorizontal="true"
        android:text="Button5" />
    <Button
        android:id="@+id/btn6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btn4"
        android:layout_centerHorizontal="true"
        android:text="Button6" />
    <Button
        android:id="@+id/btn7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/btn1"
        android:layout_toRightOf="@+id/btn1"
        android:layout_alignParentRight="true"
        android:text="Button7" />
</RelativeLayout>
```

8. Develop a Mobile App to create a Text View and change the attribute s from XML and Java File?

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.demo2.MainActivity" >

    <TextView
        android:id="@+id/i1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/ic_launcher"
        android:ellipsize="start"
        android:height="100dp"
        android:singleLine="True"
        android:text="XML Properties XML PropertiesXML Properties"
        android:textColor="#FF0000"
        android:textSize="20sp"
        android:textStyle="bold"
        android:typeface="monospace"
        android:width="150dp" />

</RelativeLayout>
```

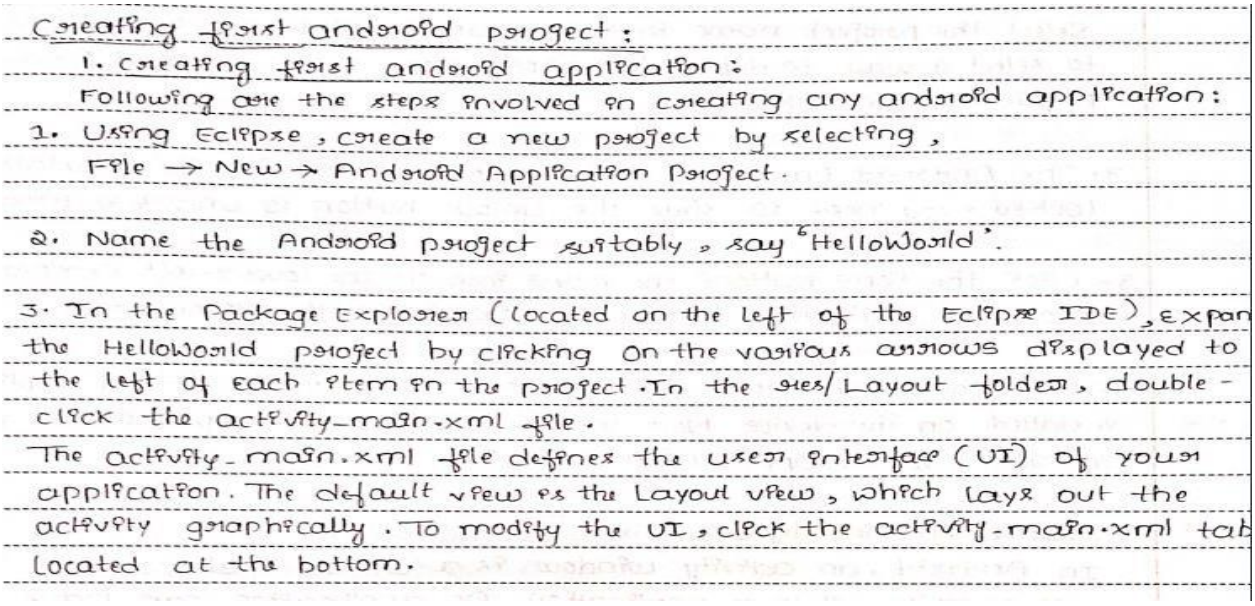
```
import android.graphics.Color;
import android.graphics.Typeface;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.Menu;
import android.widget.TextView;


public class MainActivity extends Activity {

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

       TextView t2=(TextView) findViewById(R.id.i2);

       t2.setWidth(200);
       t2.setHeight(200);
       t2.setText("JavaProperties");
       t2.setTextColor(Color.RED);
       t2.setTextSize(20);
       t2.setSingleLine(true);
       t2.setEllipsize(TextUtils.TruncateAt.MIDDLE);
       t2.setBackgroundResource(R.drawable.ic_launcher);
       t2.setTypeface(Typeface.BOLD);

    }
}
```

9. With an example program explain the differences between Table Layout and Grid Layout.

Table Layout	Grid Layout
Table Layout is used to arrange the group of views	GridView is basically like a ListView, whose items are arranged in a strict grid.

Table Layout	Grid Layout
into rows and columns.	
Table Layout containers do not display a border line for their columns, rows or cells.	Table Layout containers display a border line for their columns, rows or cells.
Need not specify row and column count	Need to specify row and column count
All elements may be of different size.	All elements in the grid must be of same size.
Syntax: <TableLayout android:layout_width="match_parent" android:layout_height="wrap_content" android:id="@+id/table01" android:shrinkColumns="1" android:collapseColumns="2" android:stretchColumns="3"> <TableRow> </TableRow> </TableLayout>	**Syntax:** <GridLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="match_parent" android:layout_height="match_parent" android:rowCount="1" android:columnCount="1" android:id="@+id/root"> </GridLayout>
Table Layout is slower compared to Grid Layout	GridLayout will generally be faster than a TableLayout.
Table Layout takes more memory than Grid Layout	GridLayout takes less memory than a TableLayout.
Needs to add views by using	Flexibly place the views randomly by

Table Layout	Grid Layout
<tableRow>	specifying column and row positions.



Creating first android project :

1. Creating first android application :

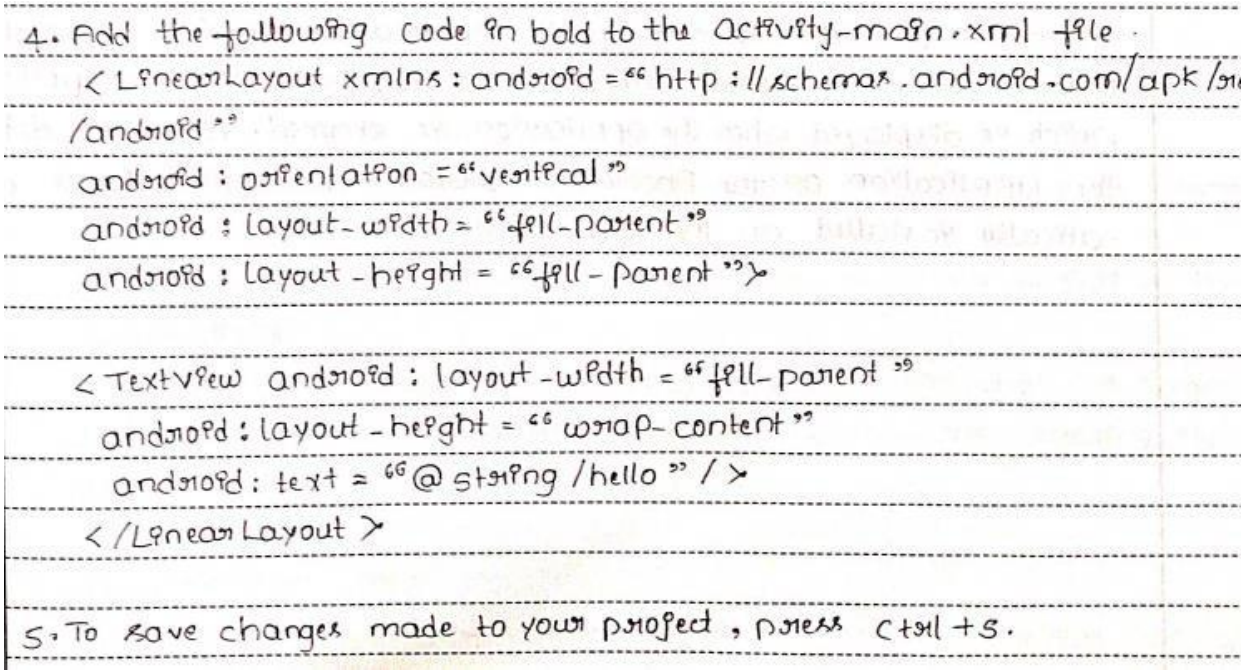
Following are the steps involved in creating any android application :

1. Using Eclipse, create a new project by selecting,
   File → New → Android Application Project.

2. Name the Android project suitably, say "HelloWorld".

3. In the Package Explorer (located on the left of the Eclipse IDE), expan
the HelloWorld project by clicking on the various arrows displayed to
the left of each item in the project. In the res/Layout folder, double-
click the activity_main.xml file.
The activity_main.xml file defines the user interface (UI) of your
application. The default view is the Layout view, which lays out the
activity graphically. To modify the UI, click the activity_main.xml tab
located at the bottom.

10. How do you Create the first android project? Illustrate the steps with an example.



4. Add the following code in bold to the activity_main.xml file.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">

<TextView android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello" />
</LinearLayout>
```

5. To save changes made to your project, press Ctrl+S.

6. You are now ready to test your application on the Android Emulator. Select the project name in Eclipse and press F11. You will be asked to select a way to debug the application. Select required Android Application and click OK.

7. The Android Emulator will now be started (if the Emulator is locked, you need to slide the unlock button to unlock it first).

8. Click the Home button (the house icon in the lower-left corner above the keyboard) so that it now shows the Home Screen.

9. Click the application Launcher icon to display the list of applications installed on the device. Note that the HelloWorld application is now installed in the application Launcher.

Creating AVD in step 6:

In Android, an activity window is a window that contains the user interface of your application. An application can have zero or more activities. In this Eg, the application contains one activity: MainActivity. This MainActivity is the Entry point of the application which is displayed when the application is started. When you debug the application on the Android Emulator, the application is automatically installed on the Emulator.