(3170726)MAD **SUMMER 2022**

[Q.1]

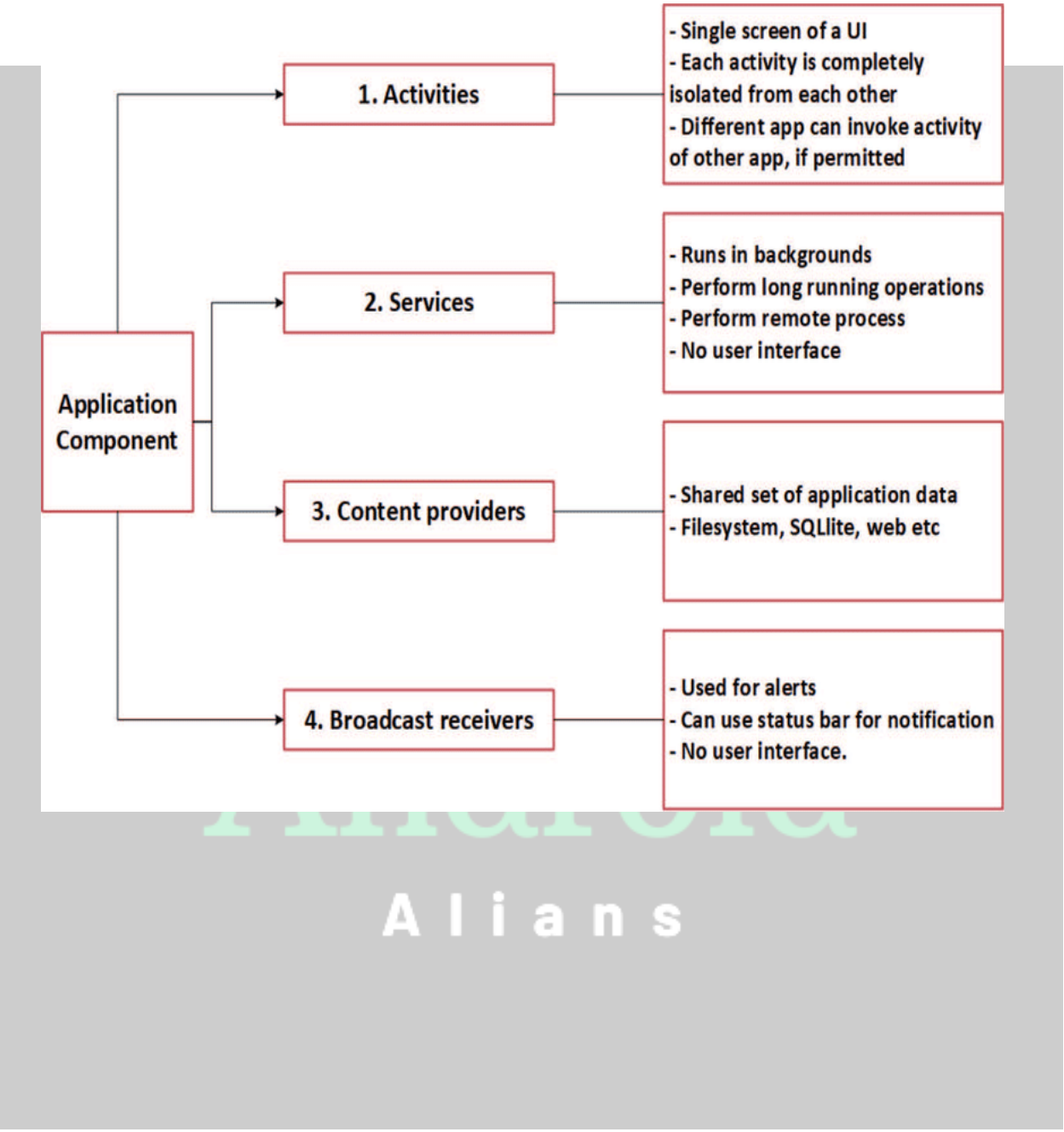
(a) Differentiate between JVM and DVM.

➔

DVM (Dalvik Virtual Machine)	JVM (Java Virtual Machine)
It is Register based which is designed to run on low memory.	It is Stack based.
DVM uses its own byte code and runs the ".Dex" file. From Android 2.2 SDK Dalvik has got a Just in Time compiler	JVM uses java byte code and runs ".class" file having JIT (Just In Time).
DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance.	A single instance of JVM is shared with multiple applications.
DVM supports the Android operating system only.	JVM supports multiple operating systems.
For DVM very few Re-tools are available	For JVM many Re-tools are available.
There is a constant pool for every application.	It has a constant pool for every class.
Here the executable is APK.	Here the executable is JAR.

(b) Enlist and define the components of android application.

➔



Q-6 Explain basic building blocks/ components of Android Application.

→ An android component is a simply a piece of code that has a well defined life cycle eg. Activity, Receiver, service etc.

→ The core building blocks or fundamental components of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

→ Activity:

→ An activity is a class that represent a single screen.

→ It is like a Frame in AWT.

→ View =

→ A view is the UI element such as button, label, text field etc.

→ Anything that you see is a view.

→ Intent:

→ Intent is used to invoke components.

It is mainly used to:

→ Start the service

→ Lunch an activity

→ Display a web Page

→ Display a list of contacts
→ Broadcast a message
→ Dial a phone call etc.

→ for ex, you may write the following code to view the webpage.

```
Intent intent = new Intent (Intent.ACTION_
                            VIEW);
intent.setData(Uri.Parse("http://www.
                youtube [redacted].com"));
StentActivity(intent);
```

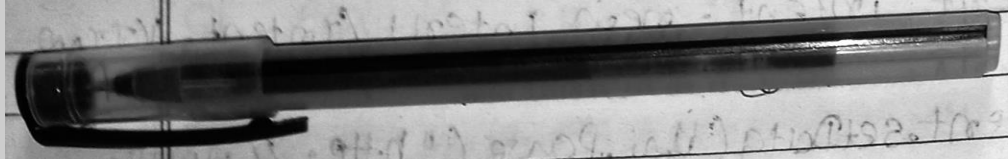
⇒ Service :
→ Service is a background process that can run for a long time.
→ There are two types of services - local & - Remote
→ local service is accessed from within the application.
→ remote Service is accessed remotely from other applications running on the same device

⇒ Content Providers :
→ content providers are used to share data between the applications.

⇒ Fragment :
→ Fragments are like parts of Activity.
→ An Activity can display one or more fragments on the screen at the same time

Teacher's Signature.....

→ Androidmanifest.xml
→ It contains informations about activities, content providers, permissions etc.
→ It is like the web.xml file in java EE.



(7) Broadcast Receiver:
→ Broadcast Receivers are used to respond to these system-wide events.
→ Broadcast receivers allow us to register for the system and application events, and when that event happens, then the register receivers get notified.
→ There are 2 types of broadcast receivers:
  → Static Broadcast receivers
  → Dynamic Broadcast receivers

(c) Describe the Android architecture with neat diagram in detail.

Q-1 Explain Android Architecture with diagram.

→ Android Architecture is a stack of s/w components.

→ It is in the form of a software Application, Operating System, a run-time environment, middleware, native libraries and services.

→ It is categorized into five parts as below:

1) Linux kernel
2) Native Libraries
3) Android runtime
4) Application Framework
5) Applications

→ 1) Linux Kernel :-

→ Linux is the heart of Android Architecture.

→ It Provides a level of abstraction between the hardware devices and the upper layers of the Android s/w stack.

→ The Android Operating System is based on the Linux Kernel.

→ The Linux Kernel is responsible for various device drivers such as camera driver, Display driver, Bluetooth driver, Keypad driver, memory management, Process management, Power management etc.

→ 2) Native Libraries :-

→ The native libraries such as Media, Webkit, SQLite, OpenGL, FreeType, C Runtime library (libc) etc. are situated on the top of a Linux Kernel.

→ Media library is responsible for playing and recording audio and video formats, FreeType is for font support, Webkit is for browser support, SQLite is for database, SSL is for Internet security etc.

→ 3) Android Runtime :-

→ Android Runtime environment is one of the most important part of Android.

→ Android runtime includes core libraries and Dalvik virtual machine (DVM) which is responsible to run android application.

→ DVM is like JVM in Java, but DVM is optimized for mobile devices.

Date : / /

→ DVM makes use of the Linux core features like memory management and multi-threading, which are essential in the Java language.

→ DVM provides fast performance and consumes less memory.

4) Application Framework :-

→ Application framework provides several important classes which are used to create an Android application.

→ It includes Android API's such as Activity manager, window manager, content provider, Telephony manager etc.

5) Applications :-

→ Applications are situated on the top of the Application framework.

→ The applications such as home, contact, Alarm, Calender, Camera, Browser etc. use the Android framework which uses Android runtime and libraries.

→ Android runtime & native libraries use Linux Kernel.

→ The user can write his/her application to be installed on this layer only.

Teacher's Signature.....

[Android Architecture]

[Q.2]

(a) What is Toast? Explain How to customize it?

➔ Toast is a small popup notification that is used to display information about the operation which we performed in our app.
➔The Toast will show the message for a small period of time and it will disappear automatically after a timeout.
➔Generally, the size of Toast will be adjusted based on the space required for the message and it will be displayed on the top of the main content of activity for a short period of time.

➔You are able to create custom toast in android. So, you can display some images like congratulations or loss on the toast. It means you are able to customize the toast now.

➔Custom Toast in Android:

In Android, Sometimes simple Toast may not be satisfactory, and then we can go for customizing a Toast. For creating a custom layout, define a View layout, in XML and pass the root View object to the setView(View) method.

➔Steps for Implementation of Custom Toast In Android:

Step 1: Firstly Retrieve the Layout Inflater with getLayoutInflater() (or getSystemService()) and then inflate the layout from XML using inflate(int, ViewGroup). In inflate method first parameter is the layout resource ID and the second is the root View.

Step 2: Create a new Toast with Toast(Context) and set some properties of the Toast, such as the duration and gravity.

Step 3: Call setView(View) and pass the inflated layout in this method.

Step 4: Display the Toast on the screen using show() method of Toast.

- **Only for understanding**…….

➔activity_main.xml file

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

        <Button
                android:id="@+id/button1"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Show Custom Toast"
        />
```

</ RelativeLayout >

➔Custom_toast.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/custom_toast_layout"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:background="#80CC28">

    <TextView android:id="@+id/txtvw"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="13dp"
        android:textColor="#FFF"
        android:textStyle="bold"
        android:textSize="15dp" />
</LinearLayout>
```

➔MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn = (Button)findViewById(R.id.btnShow);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                LayoutInflater inflater = getLayoutInflater();
                View layout = inflater.inflate(R.layout.custom_toast, (ViewGroup)
      findViewById(R.id.custom_toast_layout));
                TextView tv = (TextView) layout.findViewById(R.id.txtvw);
                tv.setText("Custom Toast Notification");
                Toast toast = new Toast(getApplicationContext());
                toast.setGravity(Gravity.CENTER_VERTICAL, 0, 100);
                toast.setDuration(Toast.LENGTH_LONG);
                toast.setView(layout);
                toast.show();
```

```
            }
        });
    }
}
```

(b) What is AndroidManifest.xml? Write its usages with example.

➔ The **AndroidManifest.xml file** *contains information of your package*, including components of the application such as activities, services, broadcast receivers, content providers etc.

➔**Element Tags of Manifest File**

Following are the essential element tags of Manifest.xml files:

1. <manifest>

It is the root element of this element. It consists of a package attribute package that tells the activity's package name.

2. <application>

It is the subelement of the manifest file that includes the declaration of the namespace. It contains certain attributes. These attributes declare the application components, and these attributes include:

icon
allowBackup
label
theme

3. <activity>

Activity is a subelement of application. It has the declaration of the activity that must be there in the manifest file. It also has certain attributes like name, label, theme, etc.

4. <intent-filter>

It is an element in the activity, it describes the type of intent in which the Android components can respond.

5. <action>

This element provides an action for the intent filter. Each intent filter must have at least one action element in it.

6. <category>

This element adds the category name in an intent-filter.

7. <service>

This element contains the operations that are provided by libraries or APIs.

⇨ **A simple AndroidManifest.xml file looks like this:**

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

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

(c) Explain any four UI Components of Android application.

➔There are many UI Components of Android Application.

- TextView
- EditText
- Button
- ImageButton
- ToggleButton
- RadioButton
- RadioGroup
- CheckBox
- AutoCompleteTextView
- ProgressBar
- Spinner
- TimePicker
- DatePicker
- SeekBar
- AlertDialog
- Switch
- RatingBar

➔1. TextView

TextView is a UI Component that displays the text to the user on their Display Screen.
We can create it in two ways:

➔XML file:

```
<Linear Layout xmls:android= "http://schemas.android.com/apk/res/android"
      <TextView
      //attributes to describe it
      />
</LinearLayout>
```

➔**Activity file:**

In this, we declare it using the setText() method as follows:

```
setContentView(R.layout.activity_main);

LinearLayout linearlayout_name = (LinearLayout)findViewById(R.id.LinearLayout);

TextView textview_name = new TextView(this);

textveiw_name.setText("Hello I am Text View");

linearLayout.addView(textView);
```

⇨ Here are various attributes to describe the TextView some of them are named below:

Android: id – it is a unique id for the control.

Android: width – It displays the exact width of the TextView.

Android: height – It displays the exact height of the TextView.

Android:textColor – It set the color of the text.

Android: gravity – It is to align the TextView.

2. EditText

➔EditText is a user interface control that allows the users to enter some text.

We can create it in two ways:

➔**XML file:**

```
<Linear Layout xmls:android= "http://schemas.android.com/apk/res/android">
      <EditText
      //attributes
      >
</LinearLayout>
```

➔**Activity file:**

⇨ In activity, we declare it using the getText() method as follows:

```
setContentView(R.layout.activity_main);

LinearLayout linearlayout_name = (LinearLayout) findViewById (R.id.LinearLayout) ;

EditText edittext_name = new EditText(this);

edittext_name.setHint("Hello I am EditText");
```

linearLayout.addView(edittext_name);

3. Button

➔This is a UI that is used to perform some action as soon as the user clicks on it.
We can create it in two ways:

➔**XML file:**

```
<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>
    <Button
    //attributes
    />
</LinearLayout>
```

➔**Activity file:**

```
setContentView(R.layout.activity_main);
LinearLayout linearlayout_name = (LinearLayout)findViewById(R.id.LinearLayout);
Button btn_name = new Button(this);
btn_name.setText(“Hello I am Button”);
linearLayout.addView(btn_name);
```

4. ImageButton

It is the same as a Button but it’s used to display an image on the button to perform an Action. In this, we need to give the source of the image so that the system can load it.
We can create it in two ways:

➔**XML file:**

```
<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>

<ImageButton
//other attributes...
android:src= “@drawable/add_icon”/>

</LinearLayout>
```

➔**Activity file:**

```
setContentView(R.layout.activity_main);
LinearLayout linearlayout_name = (LinearLayout)findViewById(R.id.LinearLayout);
ImageButton btn_name = new Button(this);
btn_name.setImageResource(R.drawable.add_icon);
linearLayout.addView(btn_name);
```

OR

(c) Develop a Registration form using android UI components, which take details from user like Name, Email ID, Password, Conform Password, Mobile Number, gender etc... On click of register button all details should show on another activity with a welcome message.

➔Activity_main.xml

```
<?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:layout_gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

        <EditText
            android:id="@+id/et_Name"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Name"
            android:inputType="text"/>

        <EditText
            android:id="@+id/et_Mail"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Email Id"
            android:inputType="textEmailAddress">

         <EditText
            android:id="@+id/et_Pass"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Password"
            android:textColor="@color/black"
            android:inputType="textPassword"/>
```

```
        <EditText
            android:id="@+id/et_Cpass"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Confirm Password"
            android:inputType="textPassword"
            />
        <EditText
            android:id="@+id/Mobile"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Enter Mobile"
            android:inputType="number"
            android:maxLength="10”/>


    <RadioGroup
        android:id="@+id/Sex"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_gravity="center">

        <RadioButton
            android:id="@+id/Male"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male"/>

        <RadioButton
            android:id="@+id/Female"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female"/>
    </RadioGroup>

    <Button
        android:id="@+id/btn1"
        android:layout_width="190sp"
        android:layout_height="wrap_content"
        android:text="Register"/>

</LinearLayout>
```

➔**MainActivity.java**

```
public class MainActivity extends AppCompatActivity {
    Button btn1;
    EditText et_name, et_mail, et_pass, et_cpass, et_mobile;
    RadioButton rb1, rb2;



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



        et_name = findViewById(R.id.et_Name);
        et_mail = findViewById(R.id.et_Mail);
        et_pass = findViewById(R.id.et_Pass);
        et_cpass = findViewById(R.id.et_Cpass);
        et_mobile = findViewById(R.id.Mobile);
        rb1 = findViewById(R.id.Male);
        rb2 = findViewById(R.id.Female);
        btn1 = findViewById(R.id.btn1);


        btn1.setOnClickListener(view -> {

            String name = et_name.getText().toString();
            String mail = et_mail.getText().toString();
            String pass = et_pass.getText().toString();
            String mobile = et_mobile.getText().toString();

Intent intent = new Intent(MainActivity.this, ValidReg.class);
intent.putExtra("name", name);
intent.putExtra("mail", mail);
intent.putExtra("pass", pass);
intent.putExtra("mobile", mobile);


    startActivity(intent);
    Toast.makeText(this, "Successfully Register!!!",
                                    Toast.LENGTH_SHORT).show();
            }
```

```
        });


        Intent i= getIntent();
        String name=i.getStringExtra("name");
        et_name.setText(name);

        String mail=i.getStringExtra("mail");
        et_mail.setText(mail);

        String pass=i.getStringExtra("pass");
        et_pass.setText(pass);

        String mobile=i.getStringExtra("mobile");
        et_mobile.setText(mobile);


    }

}
```

➔validreg.java

```
package com.example.registration;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;

public class ValidReg extends AppCompatActivity {

    EditText et_name, et_mail, et_pass, et_mobile;
    String name,mail,pass,mobile;
    ImageView iv1;



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

        et_name = findViewById(R.id.et_Name);
```

```
        et_mail = findViewById(R.id.et_Mail);
        et_pass = findViewById(R.id.et_Pass);
        et_mobile = findViewById(R.id.Mobile);
        iv1=findViewById(R.id.left_icon);


        iv1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                name= et_name.getText().toString().trim();
                mail= et_mail.getText().toString().trim();
                pass= et_pass.getText().toString().trim();
                mobile= et_mobile.getText().toString().trim();

                Intent i= new
Intent(ValidReg.this,MainActivity.class);
                i.putExtra("name",name);
                i.putExtra("mail",mail);
                i.putExtra("pass",pass);
                i.putExtra("mobile",mobile);

                startActivity(i);

            }
        });


       name=getIntent().getStringExtra("name");
        mail=getIntent().getStringExtra("mail");
        pass=getIntent().getStringExtra("pass");
        mobile= getIntent().getStringExtra("mobile");

            et_name.setText(name);
            et_mail.setText(mail);
            et_pass.setText(pass);
            et_mobile.setText(mobile);
       }


}
```

➔**validreg.xml**

```
<?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:layout_gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity"
    android:background="@drawable/img">

<EditText
            android:id="@+id/et_Name"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Name"
            android:inputType="text"
            />

        <EditText
            android:id="@+id/et_Mail"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Email Id"
            android:inputType="textEmailAddress"
            >

         <EditText
            android:id="@+id/et_Pass"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Password"
            android:textColor="@color/black"
            android:inputType="textPassword"
            />

        <EditText
            android:id="@+id/et_Cpass"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
```

```
            android:hint="Confirm Password"
            android:inputType="textPassword"
            />
        <EditText
            android:id="@+id/Mobile"
            android:layout_width="350sp"
            android:layout_gravity="center"
            android:layout_height="wrap_content"
            android:hint="Enter Mobile"
            android:inputType="number"
            android:maxLength="10"
            />
</LinearLayout>
```

[Q.3]

(a) What is Fragment? Differentiate between Activity and fragment.

➔

(Refer 2019 paper OR Q-3-C).

Activity	Fragment
Activity is an application component that gives a user interface where the user can interact.	The fragment is only part of an activity, it basically contributes its UI to that activity.
Activity is not dependent on fragment	Fragment is dependent on activity. It can't exist independently.
we need to mention all activity it in the manifest.xml file	Fragment is not required to mention in the manifest file
We can't create multi-screen UI without using fragment in an activity,	After using multiple fragments in a single activity, we can create a multi-screen UI.
Activity can exist without a Fragment	Fragment cannot be used without an Activity.
Creating a project using only Activity then it's difficult to manage	While Using fragments in the project, the project structure will be good and we can handle it easily.
Lifecycle methods are hosted by OS. The activity has its own life cycle.	Lifecycle methods in fragments are hosted by hosting the activity.
Activity is not lite weight.	The fragment is the lite weight.

(c) What is an Activity? Explain the activity life cycle with all events in detail.

➔Android Activity Lifecycle

➔**Android Activity Lifecycle** is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

➔An activity is the single screen in android. It is like window or frame of Java.

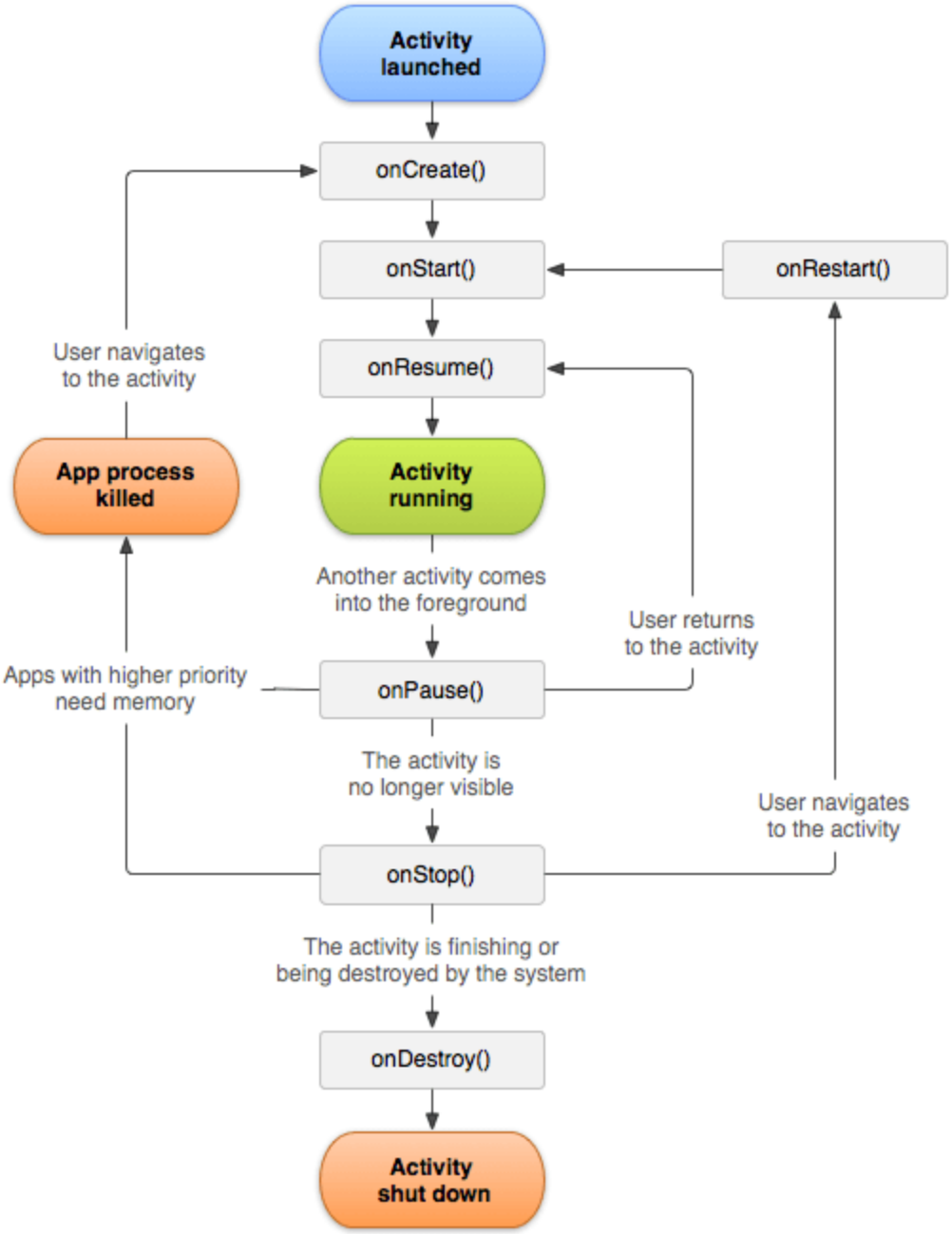
➔By the help of activity, you can place all your UI components or widgets in a single screen.

The 7 lifecycle method of Activity describes how activity will behave at different states.

➔Android Activity Lifecycle methods

⇨ **Let's see the 7 lifecycle methods of android activity.**

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

OR

[Q.3]

(a) Enlist and define types of Menus in android

➔we have three types of Menus available to define a set of options and actions in our android applications. The Menus in android applications are the following:

- ⇨ Android Options Menu
- ⇨ Android Context Menu
- ⇨ Android Popup Menu

- ⇨ **Android Options Menu** is a primary collection of menu items in an android application and is useful for actions that have a global impact on the searching application.
- ⇨ **Android Context Menu** is a floating menu that only appears when the user clicks for a long time on an element and is useful for elements that affect the selected content or context frame.
- ⇨ **Android Popup Menu** displays a list of items in a vertical list which presents the view that invoked the menu and is useful to provide an overflow of actions related to specific content.

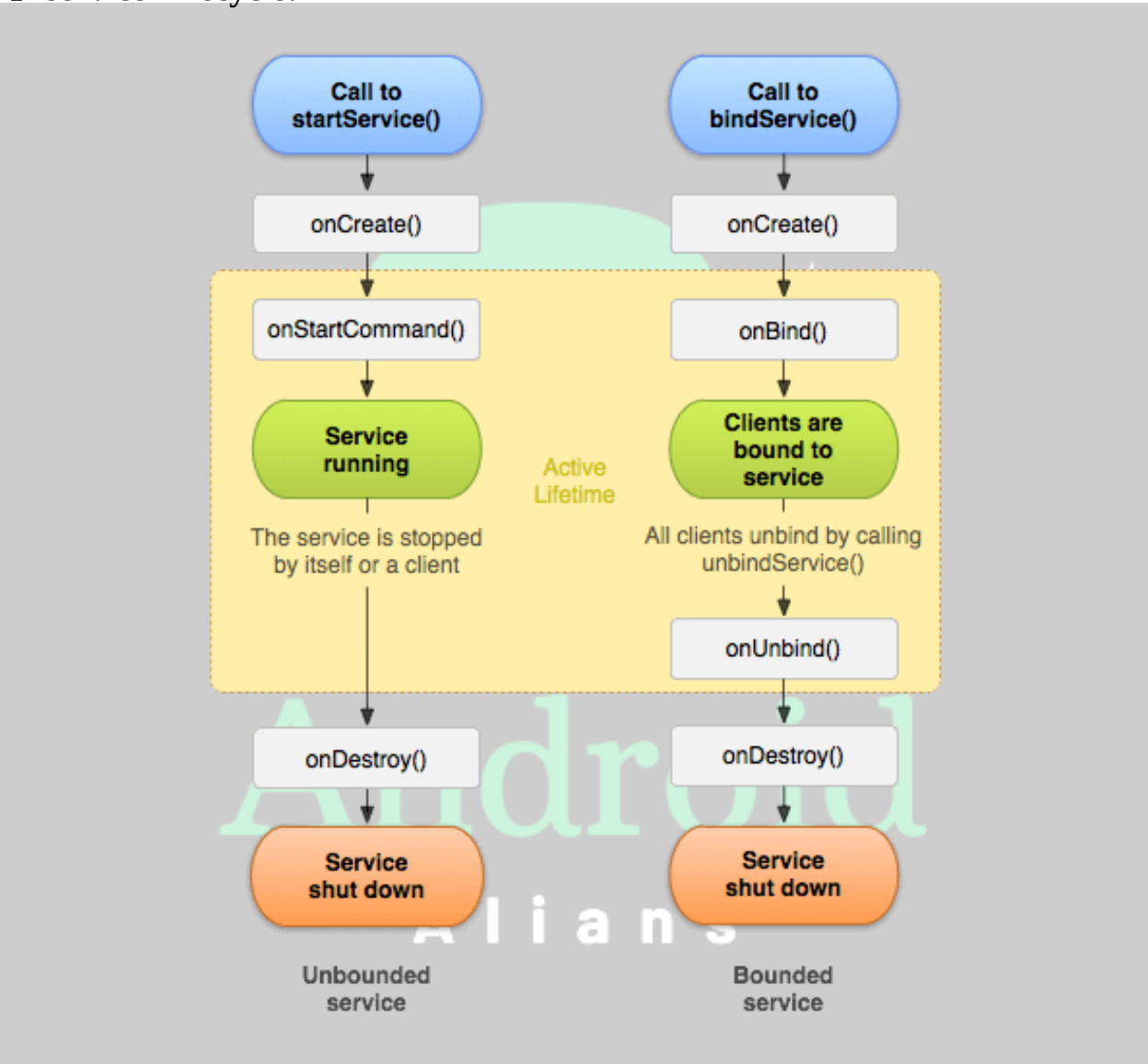
(b) Explain the concept of Recycler view. Write down the steps to implement recycler view in android application.

- ⇨ X

(c) Define services in Android operating system. Explain service Lifecycle with neat diagram.

(Define services ➔Refer 2019 paper OR Q-5-A)

➔ service Lifecycle:-



There can be two forms of a service.The lifecycle of service can follow two different paths: started or bound.

1. Started
2. Bound

1.Started

⇨ A service is **started** when an application component, such as an activity, starts it by calling *startService()*. Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.

2. Bound

A service is **bound** when an application component binds to it by calling *bindService()*. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC)

➔Methods:

1. onStartCommand()

➔The system calls this method whenever a component, say an **activity** requests 'start' to a service, using **startService()**.

Once we use this method it's our duty to stop the service using **stopService()** or **stopSelf()**.

2. onBind()

➔This is invoked when a component wants to bind with the service by calling **bindService()**. In this, we must provide an interface for clients to communicate with the service. For interprocess communication, we use the IBinder object**.**

➔It is a must to implement this method. If in case binding is not required, we should return **null** as implementation is mandatory.

3. onUnbind()

➔The system invokes this when all the clients disconnect from the interface published by the service.

4. onRebind()

➔The system calls this method when new clients connect to the service. The system calls it after the **onBind()** method.

5. onCreate()

➔This is the first callback method that the system calls when a new component starts the service. We need this method for a one-time set-up.

6. onDestroy()

➔This method is the final clean up call for the system. The system invokes it just before the service destroys. It cleans up resources like threads, receivers, registered listeners, etc.

[Q.4]

(a) what do you mean by NoSQL? How it is differ from relational database?

➔NoSQL, also referred to as "not only SQL", "non-SQL", is an approach to database design that enables the storage and querying of data outside the traditional structures found in relational databases.

CECHA	SQL DATABASE	NOSQL DATABASE
Type	Relational base	Non-relational base
Structure	In the SQL database, data is stored in tables with a certain number of columns	Non-relational databases can have the following structure: document, graph, key-value store
Type of data structure	Fixed data structure	Variable data structures
Scalability	Vertical scalability	Horizontal scalability
Queries	Usually, it is SQL	How to query a NoSQL database depends mainly on the specific database – there is no declarative query language
Key features	Data consistency, integrity and stability	Scalability, flexibility, and quick queries

SQL	NOSQL
Relational Database management system	Distributed Database management system
Vertically Scalable	Horizontally Scalable
Fixed or predifined Schema	Dynamic Schema
Not suitable for hierarchical data storage	Best suitable for hierarchical data storage
Can be used for complex queries	Not good for complex queries

(b) What do you mean by Async task? Explain with example.

➔ AsyncTask going to do background operation on background thread and update on main thread. In android we cant directly touch background thread to main thread in android development. asynctask help us to make communication between background thread to main thread.

➔activity_main.xml

```
<RelativeLayout
xmlns:android="https://schemas.android.com/apk/res/android"

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

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity" >



    <TextView

        android:id="@+id/tv_time"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:textSize="10pt"

        android:text="Sleep time in Seconds:"/>

    <EditText

        android:id="@+id/in_time"

        android:layout_width="150dip"

        android:layout_height="wrap_content"

         android:background="@android:drawable/editbox_backg"

        android:layout_toRightOf="@id/tv_time"
```

```
        android:layout_alignTop="@id/tv_time"
        android:inputType="number"/>
    <Button
        android:id="@+id/btn_run"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Run Async task”/>
    <TextView
        android:id="@+id/tv_result"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="7pt"/>
</RelativeLayout>
```

⇨ **MainActivity.java**

```
public class MainActivity extends AppCompatActivity {
    private Button button;
    private EditText time;
    private TextView finalResult;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        time = (EditText) findViewById(R.id.in_time);
        button = (Button) findViewById(R.id.btn_run);
        finalResult = (TextView) findViewById(R.id.tv_result);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AsyncTaskRunner runner = new AsyncTaskRunner();
```

```
                String sleepTime = time.getText().toString();
                runner.execute(sleepTime);
            }
        });
    }
    private class AsyncTaskRunner extends AsyncTask<String,
String, String> {
        private String resp;
        ProgressDialog progressDialog;
        @Override
        protected String doInBackground(String... params) {
            publishProgress("Sleeping..."); // Calls
onProgressUpdate()
            try {
                int time = Integer.parseInt(params[0])*1000;
                Thread.sleep(time);
                resp = "Slept for " + params[0] + " seconds";
            } catch (InterruptedException e) {
                e.printStackTrace();
                resp = e.getMessage();
            } catch (Exception e) {
                e.printStackTrace();
                resp = e.getMessage();
            }
            return resp;
        }
        @Override
        protected void onPostExecute(String result) {
            // execution of result of Long time consuming
operation
            progressDialog.dismiss();
            finalResult.setText(result);
        }
        @Override
        protected void onPreExecute() {
 progressDialog = ProgressDialog.show(MainActivity.this,
                    "ProgressDialog", "Wait for
                    "+time.getText().toString()+ " seconds");
        }
        @Override
        protected void onProgressUpdate(String... text) {
            finalResult.setText(text[0]);
        }
    }
}
```

OR

[Q-4]

(a)Describe the task of Notification Manager in Android.

➔Android allows to put notification into the titlebar of your application. The user can expand the notification bar and by selecting the notification the user can trigger another activity.

(b) Can we use the mobile screen as a Canvas to draw any shapes on it? Write the steps for using Graphics objects in an android application.

➔Refer (https://www.codingconnect.net/android-application-basic-graphical-primitives/)

(c) List different types of data storage available in Android. Describe the significance of SQLite database in Android.

➔

There are 4 type of storage that android provide.

- **Internal file storage:** Store app-private files on the device file system.
- **External file storage:** Store files on the shared external file system. This is usually for shared user files, such as photos.
- **Shared preferences:** Store private primitive data in key-value pairs.
- **Databases:** Store structured data in a private database**.**

➔**Internal file storage**:

⇨ By default, files saved to the internal storage are private to your app, and other apps cannot access them (nor can the user, unless they have root access). This makes internal storage a good place for internal app data that the user doesn't need to directly access. The system provides a private directory on the file system for each app where you can organize any files your app needs.

➔**External storage**

⇨ Every Android device supports a shared “external storage” space that you can use to save files. This space is called external because it’s not a guaranteed to be accessible — it is a storage space that users can mount to a computer as an external storage device, and it might even be physically removable (such as an SD card). Files saved to the external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.

⇨ So before you attempt to access a file in external storage in your app, you should check for the availability of the external storage directories as well as the files you are trying to access.

➔**Shared preferences**

⇨ If you don’t need to store a lot of data and it doesn’t require structure, you should use SharedPreferences. TheSharedPreferences APIs allow you to read and write persistent key-value pairs of primitive data types: booleans, floats, ints, longs, and strings.

⇨ The key-value pairs are written to XML files that persist across user sessions, even if your app is killed. You can manually specify a name for the file or use per-activity files to save your data.

➔Databases

⇨ Android provides full support for SQLite databases. Any database you create is accessible only by your app. However, instead of using SQLite APIs directly, we recommend that you create and interact with your databases with the Room persistence library.

⇨ The Room library provides an object-mapping abstraction layer that allows fluent database access while harnessing the full power of SQLite.

[Q-5]

(a)Describe Tween Animation in android.

➔Tween Animation takes some parameters such as start value , end value, size , time duration , rotation angle e.t.c and perform the required animation on that object. It can be applied to any type of object. So in order to use this , android has provided us a class called Animation.

➔In order to perform animation in android , we are going to call a static function loadAnimation() of the class AnimationUtils. We are going to receive the result in an instance of Animation Object. Its syntax is as follows −

```
Animation animation = AnimationUtils.loadAnimation(getApplicationContext(),
  R.anim.myanimation);
```

➔Note the second parameter. It is the name of the our animation xml file. You have to create a new folder called **anim** under res directory and make an xml file under anim folder.

Sr.No	Method & Description
1	**start()** This method starts the animation.
2	**setDuration(long duration)** This method sets the duration of an animation.
3	**getDuration()** This method gets the duration which is set by above method
4	**end()** This method ends the animation.
5	**cancel()** This method cancels the animation.

(b) Write a program to locate user's current location. (Write ONLY .java and manifest file)

➔Large program :

Refer link : **(https://www.javatpoint.com/android-google-map-displaying-current-location)**

(c) What is Media Player in android? Explain how to play audio using Media Player.

➔ Android provides many ways to control playback of audio/video files and streams.

➔One of this way is through a class called **MediaPlayer**.

➔Android is providing MediaPlayer class to access built-in mediaplayer services like playing audio,videoe.t.c.

➔In order to use MediaPlayer, we have to call a static Method **create()** of this class. This method returns an instance of MediaPlayer class.

- ⇨ Its syntax is as follows –
  MediaPlayermediaPlayer = MediaPlayer.create(this, R.raw.song);

➔ The second parameter is the name of the song that you want to play. You have to make a new folder under your project with name **raw** and place the music file into it.

➔Once you have created the Mediaplayer object you can call some methods to start or stop the music.

- ⇨ These methods are listed below.

```
mediaPlayer.start();
mediaPlayer.pause();
```

➔ On call to **start()** method, the music will start playing from the beginning. If this method is called again after the **pause()** method, the music would start playing from where it is left and not from the beginning.

➔In order to start music from the beginning, you have to call **reset()** method.

- ⇨ Its syntax is given below.
  mediaPlayer.reset();

➔Example : (Ref : https://www.tutorialspoint.com)

OR

[Q.5]

(a) List sensors in Android and explain any one in detail.

➔**Types of Sensors**

1. **Motion Sensors:**
2. **Position Sensors:**
3. **Environmental Sensors:**

⇨ **Position Sensors:** These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

⇨ **Example:-**

⇨ ***File: activity_main.xml***

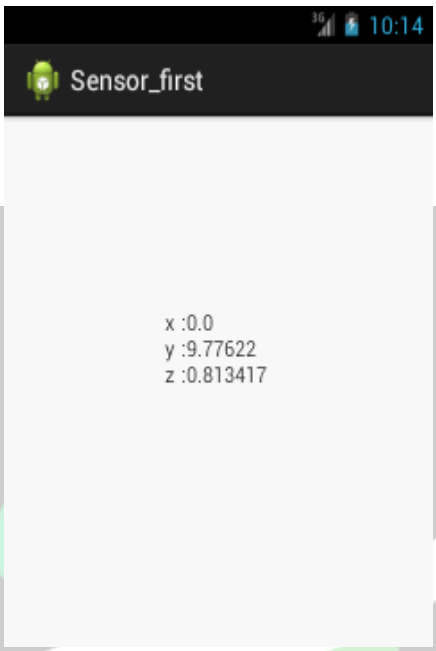
```
<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/
android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity" >

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="TextView" />

</RelativeLayout>
```

⇨ *File: MainActivity.java*

```
public class MainActivity extends Activity {
   SensorManager sm = null;
   TextView textView1 = null;
   List list;

   SensorEventListener sel = new SensorEventListener(){
      public void onAccuracyChanged(Sensor sensor, int accuracy) {}
```

```
    public void onSensorChanged(SensorEvent event) {
      float[] values = event.values;
      textView1.setText("x: "+values[0]+"\ny: "+values[1]+"\nz: "+values[2]);
    }
  };

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

    sm = (SensorManager)getSystemService(SENSOR_SERVICE);

    textView1 = (TextView)findViewById(R.id.textView1);

    list = sm.getSensorList(Sensor.TYPE_ACCELEROMETER);
    if(list.size()>0){
      sm.registerListener(sel, (Sensor) list.get(0), SensorManager.SENSOR_DELA
                                                                  Y_NORMAL)
                                                                  ;
    }else{
      Toast.makeText(getBaseContext(), "Error: No Accelerometer.", Toast.LENG
                                                                  TH_LONG).show();
    }
  }

  @Override
  protected void onStop() {
    if(list.size()>0){
     sm.unregisterListener(sel);
    }
    super.onStop();
  }
}
```

⇨ **OutPut:-**



(b) Write steps to publishing android applications.

➔

1	**Regression Testing** Before you publish your application, you need to make sure that its meeting the basic quality expectations for all Android apps, on all of the devices that you are targeting. So perform all the required testing on different devices including phone and tablets.
2	**Application Rating** When you will publish your application at Google Play, you will have to specify a content rating for your app, which informs Google Play users of its maturity level. Currently available ratings are (a) Everyone (b) Low maturity (c) Medium maturity (d) High maturity.
3	**Targeted Regions** Google Play lets you control what countries and territories where your application will be sold. Accordingly you must take care of setting up time zone, localization or any other specific requirement as per the targeted region.
4	**Application Size** Currently, the maximum size for an APK published on Google Play is 50 MB. If your app exceeds that size, or if you want to offer a secondary download, you can use APK Expansion Files, which Google Play will host for free on its server infrastructure and automatically handle the download to devices.

5	**SDK and Screen Compatibility** It is important to make sure that your app is designed to run properly on the Android platform versions and device screen sizes that you want to target.
6	**Application Pricing** Deciding whether you app will be free or paid is important because, on Google Play, free app's must remain free. If you want to sell your application then you will have to specify its price in different currencies.
7	**Promotional Content** It is a good marketing practice to supply a variety of high-quality graphic assets to showcase your app or brand. After you publish, these appear on your product details page, in store listings and search results, and elsewhere.
8	**Build and Upload release-ready APK** The release-ready APK is what you you will upload to the Developer Console and distribute to users. You can check complete detail on how to create a release-ready version of your app: Preparing for Release.
9	**Finalize Application Detail** Google Play gives you a variety of ways to promote your app and engage with users on your product details page, from colourful graphics, screen shots, and videos to localized descriptions, release details, and links to your other apps. So you can decorate your application page and provide as much as clear crisp detail you can provide.

(c) Write the steps to use firebase database in an android application for CRUID operation.

➔Refer (https://www.simplifiedcoding.net/firebase-realtime-database-crud/)

(3170726) **WINTER 2019** **GTU-2019**

[Q.1]

(a) What is AVD? Explain the process of creating AVD in Android application development.

➔ An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator.

➔ The Device Manager is an interface you can launch from Android Studio that helps you create and manage AVDs.

➔ Process of creating AVD :-

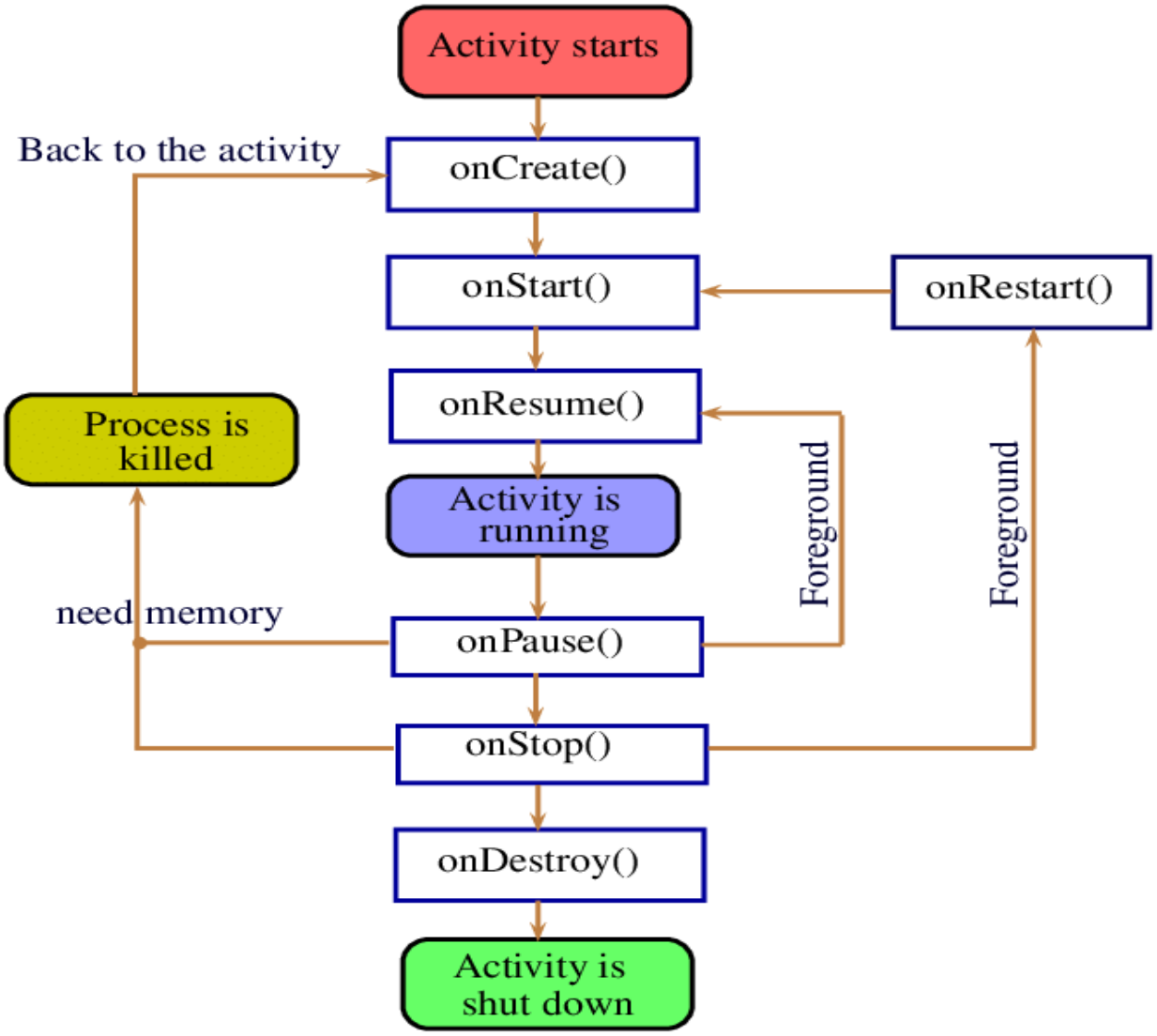
- ⇨ **Step 1**: Go to Tools > AVD Manager.
- ⇨ **Step 2**: Now click on Create Virtual Device.
- ⇨ **Step 3**: A pop-up window will be there and here we select the category Phone because we are creating android app for mobile and select the model of mobile phone we want to install.
- ⇨ **Step 4**: Here we select the android version to download like Q, Pie, Oreo etc and click Next button.
- ⇨ **Step 5**: Click the finish button to complete the installation.
- ⇨ **Step 6**: Now we can select the virtual device we want to run as emulator can click on the run icon.
- ⇨ **Step 7**: Finally our virtual device is ready to run our android app.

(b) What are the orientation modes of available in Android? Explain with

➔ **x**

(c) Draw and explain Activity Life Cycle in detail.

➔



⇨ **1. onCreate():-**

It is called when the activity is first created. This is where all the static work is done like creating views, binding data to lists, etc. This method also provides a Bundle containing its previous frozen state, if there was one.

⇨ **2. onStart():-**

It is invoked when the activity is visible to the user. It is followed by onResume() if the activity is invoked from the background. It is also invoked after onCreate() when the activity is first started.

⇨ **3. onRestart():-**

It is invoked after the activity has been stopped and prior to its starting stage and thus is always followed by onStart() when any activity is revived from background to on-screen.

⇨ **4. onResume():-**

It is invoked when the activity starts interacting with the user. At this point, the activity is at the top of the activity stack, with a user interacting with it. Always followed by onPause() when the activity goes into the background or is closed by the user.

⇨ **5. onPause():-**

It is invoked when an activity is going into the background but has not yet been killed. It is a counterpart to onResume(). When an activity is launched in front of another activity, this callback will be invoked on the top activity (currently on screen).

⇨ **6. onStop():-**

It is invoked when the activity is not visible to the user. It is followed by onRestart() when the activity is revoked from the background, followed by onDestroy() when the activity is closed or finished, and nothing when the activity remains on the background only.

⇨ **7. onDestroy():-**

The final call received before the activity is destroyed. This can happen either because the activity is finishing (when finish() is invoked) or because the system is temporarily destroying this instance of the activity to save space. To distinguish between these scenarios, check it with **is** Finishing() method.

[Q.2]

(a) How to use Navigation drawer in Android App?

➔ The navigation drawer is the most common feature offered by android and the navigation drawer is a UI panel that shows your app's main navigation menu.
➔It is also one of the important UI elements, which provides actions preferable to the users, for example changing user profile, changing settings of the application, etc.
➔In this article, it has been discussed step by step to implement the navigation drawer in android.
➔ The user can view the navigation drawer when the user swipes a finger from the left edge of the activity.
➔They can also find it from the home activity by tapping the app icon in the action bar.
➔The drawer icon is displayed on all top-level destinations that use a Drawer Layout.

➔ Steps to Implement Navigation Drawer in Android

- ⇨ **Step 1: Create a New Android Studio Project**
  - Create an empty activity android studio project. Refer to Android | How to Create/Start a New Project in Android Studio? on how to create an empty activity android studio project.
- ⇨ **Step 2: Adding a dependency to the project**

  In this discussion, we are going to use the Material Design Navigation drawer. So add the following Material design dependency to the app-level Gradle file.

implementation 'com.google.android.material:material:1.3.0-alpha03'

- ⇨ **Step 3: Creating a menu in the menu folder**

  Create the menu folder under the res folder. To implement the menu. Refer to the following video to create the layout to implement the menu.
- ⇨ **Step 4: Working with the activity_main.xml File**

  Invoke the following code in the **activity_main.xml** to set up the basic things required for the Navigation Drawer.
- ⇨ **Step 5: Include the Open Close strings in the string.xml**

  Invoke the following code in the app/res/values/strings.xml file.
- ⇨ **Step 6: Working with the MainActivity File**

  Invoke the following code in the **MainActivity** file to show the menu icon on the action bar and implement the open-close functionality of the navigation drawer.

(b) Write a code to add ProgressDialog in Android App.

➔

⇨ **XML :-**

```
<RelativeLayout xmlns:android="https://schemas.android.com/apk/res/android"
        xmlns:tools="https://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity" >
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Start ProgressDialog"
        android:layout_marginTop="57dp" />
</RelativeLayout>
```

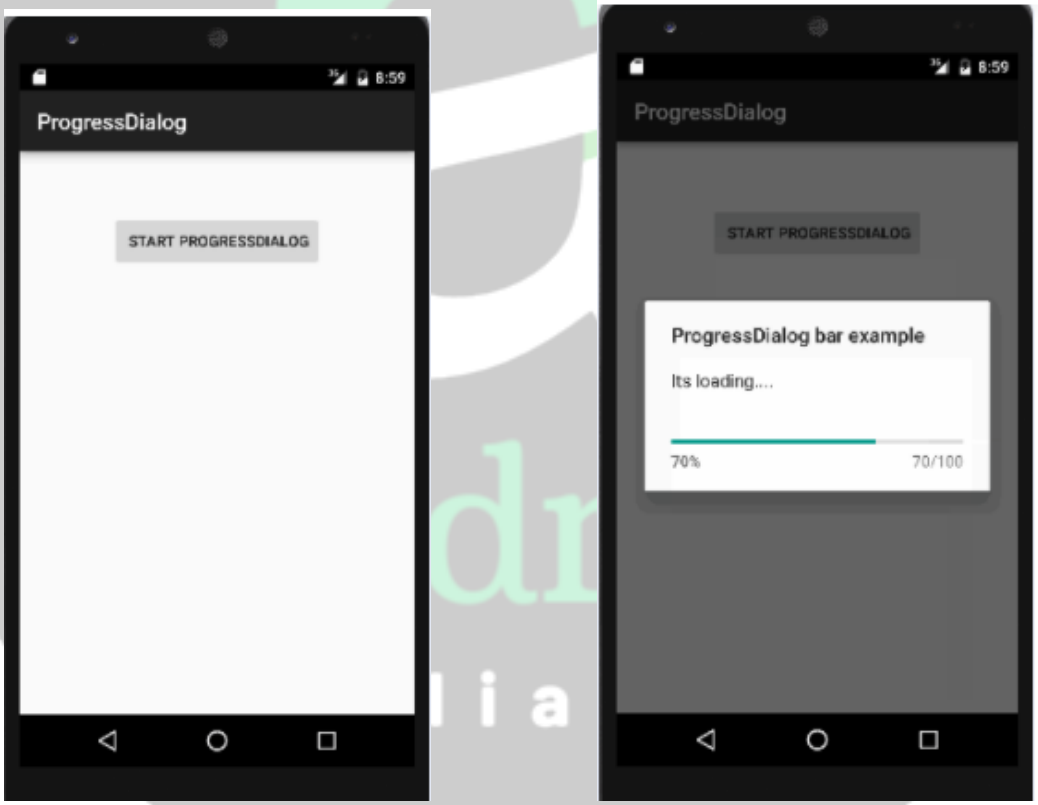
⇨ **JAVA:-**

```
public class MainActivity extends AppCompatActivity {
    Button button;
    ProgressDialog progressDoalog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
            button = (Button) findViewById(R.id.button);
            button.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View v) {
```

```
                progressDoalog = new ProgressDialog(MainActivity.this);
                progressDoalog.setMax(100);
                progressDoalog.setMessage("Its loading....");
                progressDoalog.setTitle("ProgressDialog bar example");
                progressDoalog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                progressDoalog.show();
    new Thread(new Runnable() {
      @Override
       public void run() {
          try {
             while (progressDoalog.getProgress() <= progressDoalog
                   .getMax()) {
               Thread.sleep(200);
               handle.sendMessage(handle.obtainMessage());
               if (progressDoalog.getProgress() == progressDoalog
                     .getMax()) {
                  progressDoalog.dismiss();
               }
             }
          } catch (Exception e) {
             e.printStackTrace();
          }
       }
    }).start();
}
Handler handle = new Handler() {
    @Override
```

```
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                progressDoalog.incrementProgressBy(1);
            }
        };
    });
  }
}
```

⇨ **Output:**

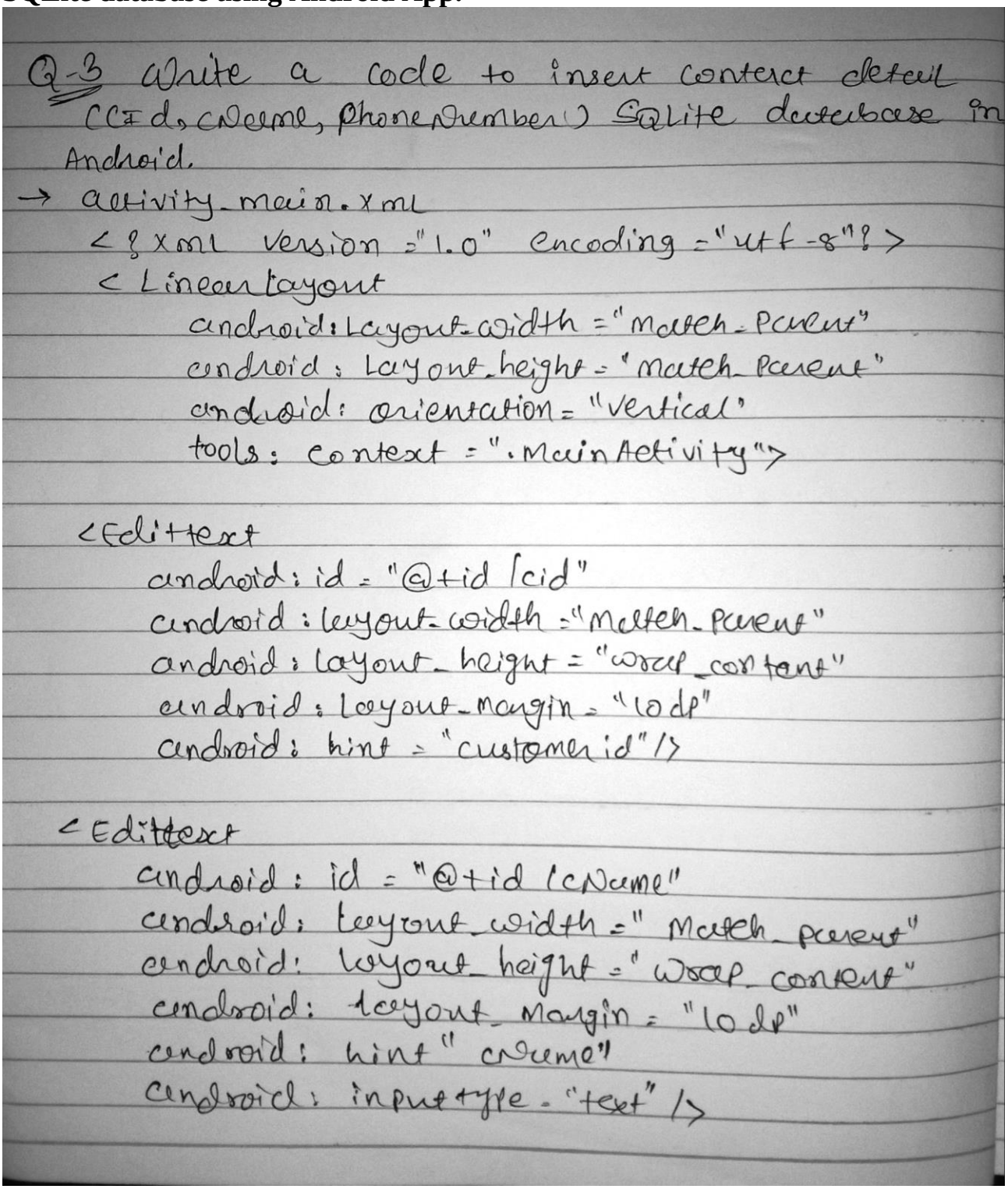


(c) Create an android activity that will display record of a customer fetched from a SQLIteDatabase.

➔ X

OR

(c) Write a code to insert studentDetails (sID, SName, sEnrollmentNo) in SQLite database using Android App.



Q-3 Write a code to insert contact detail (cId, cName, PhoneNumber) SQLite database in Android.

→ activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

<EditText
    android:id="@+id/cid"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:hint="customer id" />

<EditText
    android:id="@+id/cName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:hint="cName"
    android:inputType="text" />
```

```
<EditText
    android:id = "@+id/cphone"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:layout_margin = "10dp"
    android:hint = "PhoneNumber"
    android:inputType = "Number" />

<Button
    android:id = "@+id/save"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:layout_margin = "10dp"
    android:text = "Save"
    android:textAllCaps = "false" />
</LinearLayout>
```

⇒ MainActivity.java

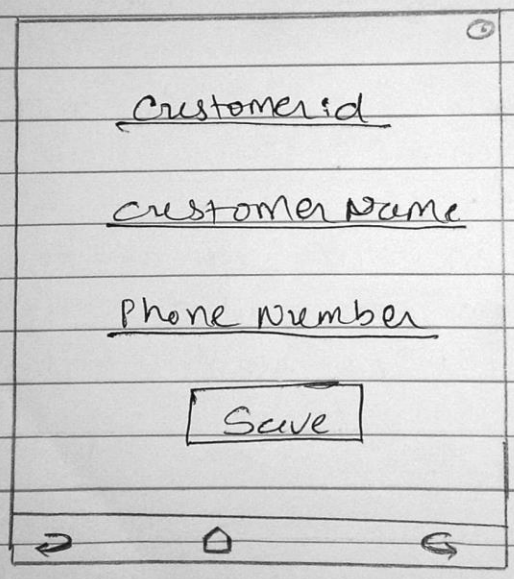
```
import androidx.appcompat.app.AppCompatActivity;
Public class MainActivity extends AppCompatActivity{

    Private EditText cid, cName, cPhone;
    Private Button save;
    Private DBhandler dbHandler;

    @Override
    Protected void onCreate(Bundle savedInstanceState) {
        Super.onCreate(SavedInstanceState);
        setContentView(R.layout.activity_main);
```

```
cid = findViewById(R.id.cid);
cName = findViewById(R.id.cName);
cPhone = findViewById(R.id.cPhone);
save = findViewById(R.id.save);

dbHandler = new DBHandler(MainActivity.this);

save.setOnClickListener(new View.OnClickListener() {

    @Override

    Public void onClick(View v) {

String cid = cid.getText().toString();
String cName = cName.getText().toString();
String cPhone = cPhone.getText().toString();

if(cid.isEmpty() || cName.isEmpty() && cPhone.isEmpty
{
  Toast.makeText(MainActivity.this, "Please enter all the
    data", Toast.LENGTH_SHORT).show();
    return;
}
dbHandler.addNewCourse(cid, cName, cPhone);

Toast.makeText(MainActivity.this, "course has
    been added", Toast.LENGTH_SHORT).show();
cid.setText("");
cName.setText("");
cPhone.setText("");
  }});
  }}
```

> DBHandler.java

```
Public class DBHandler extends SQLiteOpenHelper
{
    Private Static final String DB_Name = "coursedb";
    Private Static final int DB_VERSION = 1;
    Private Static final String TABLE_NAME = "mycourse";
    Private Static final String CID_COL = "cid";
    Private static final String CNAME_COL = "cname";
    Private static final String cphonenumber_col =
                                "cphoneNumber";
Public DBHandler (Context Context) {
    Super(Context, DB_Name, null, DB_VERSION);
}

@Override
Public void onCreate (SQLiteDatabase db) {
    String query = "CREATE TABLE " + TABLE_NAME +
    + ID_COL + "INTEGER PRIMARY KEY AUI
    + NAME_COL + "TEXT"
    + PhoneNumber_col + " NUMBER)";

    db.execSql(query);
}
Public void addNewCourse (String cid, String
cName, String cPhoneNumber) {
SQLiteDatabase db = this.getWritableDatabase();
ContentValue values = new ContentValues();
    Value.Put(CID_COL, cid);
    Value.Put(CNAME_COL, cName);
    value.Put(PhoneNumber_COL, cPhoneNumber);
```

```
db.insert(TABLE_NAME, null, values);
db.close();
}

@Override

public void onUpgrade(SQLiteDatabase db,
int oldVersion, int newVersion)
{
    db.execSQL("DROP TABLE IF EXISTS" +
    TABLE_NAME);
    onCreate(db);
  }
}
```



Database: Course

Cid	cName	phonenumber
1	Dhanu	9065778910
2	Niyati	76280 11012
3	Reshmi	84629 25624

[Q.3]

(a) What is material design in Android?

➔ **The topics covered in these articles are**

1. Color and theming
2. Typography (Choosing the right font)
3. Material design components
4. Shaping the material design components

⇨ **1. Colors and Theming**

⇨ By choosing the right kind of color combination reflects the application's brand and style. For example, the application's main or primary color is the Green, then in the whole application, the green color will be frequently shown. Choosing the color for the application, there are three types of colors to be chosen for developing the android application.

⇨ **Primary Color:**

⇨ **Secondary Color:**

⇨ **Light and Dark variants:**

⇨ **2. Typography (Choosing the Right Font)**

⇨ In android, however, the ***Roboto*** font meets all the requirements. But too, if the developer wants to customize the application more with the font, the font needs to be chosen where it has all its variants. The variants are the light face, regular face, medium face, and sometimes the dark face.

⇨ Choosing the font from Google Font is recommended. As it offers a variety of font families, and almost all the fonts have contained all the variants.

⇨ There are some guidelines that need to be followed by having the font chosen. In this case, the ***Roboto*** is chosen for demonstration purposes only. Looking at the following image which is the type scale chart for applying the styles of the fonts for various contexts.

⇨ The various contexts include captions, Body, Subtitles, Button, captions, etc.

⇨ **3. Material Design Components**

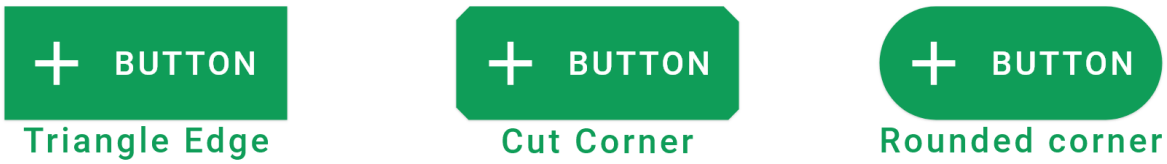
⇨ Material design components are the components that allow a lot of features for the users and easy to implement for the developers.

⇨ Have a look at the following image on how material design components stand out in terms of the customization, styling, and look, from the normal UI components.

⇨ **4. Shaping the Components**

⇨ In material design, there are three types of shaping methods.
1. Cut corner

2. Rounded corner
3. Triangle edge



- ⇨ These methods are also can be applied for the material design buttons, text fields, chips, floating action buttons, cards, navigation bars, bottom sheets, etc.
- ⇨ These features are available with the Material design components out of the proverbial box. Just need to add the dependency of the material design components and start implementing the styling for material design components.
- ⇨ For more of the hard coding, customization refers to this.

(b) List out various layouts available in Android. Explain any one in detail.

➔ **Types of Android Layout**

- ⇨ **Android Linear Layout:**
- ⇨ **Android Relative Layout:**
- ⇨ **Android Constraint Layout:**
- ⇨ **Android Frame Layout:**
- ⇨ **Android Table Layout:**
- ⇨ **Android Web View:**
- ⇨ **Android List View:**
- ⇨ **Android Grid View:**

- ⇨ **Explain any one in detail.**

- **Android Relative Layout:**

- Android Relative Layout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.

- **Relative Layout Attributes**

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.

- **Example: - XML file :**

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

 <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_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>
```

➪ **Java file :**

```
package com.example.demo;

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

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

}
```

➪ **Output:**

(c) Explain the concept of Async Task with an example.

➔ Android AsyncTask going to do background operation on background thread and update on main thread.
➔ In android we cant directly touch background thread to main thread in android development.
➔asynctask help us to make communication between background thread to main thread.

➔ **Methods of AsyncTask**

- ⇨ **onPreExecute()** − Before doing background operation we should show something on screen like progressbar or any animation to user. we can directly comminicate background operation using on doInBackground() but for the best practice, we should call all asyncTask methods .

- ⇨ **doInBackground(Params)** − In this method we have to do background operation on background thread. Operations in this method should not touch on any mainthread activities or fragments.

- ⇨ **onProgressUpdate(Progress…)** − While doing background operation, if you want to update some information on UI, we can use this method.

- ⇨ **onPostExecute(Result)** − In this method we can update ui of background operation result.

OR

[Q.3]

(a) What is the significance of Android Manifest file?

➔ The **AndroidManifest.xml file** *contains information of your package*, including components of the application such as activities, services, broadcast receivers, content providers etc.

➔ It performs some other tasks also:

- It is **responsible to protect the application** to access any protected parts by providing the permissions.
- It also **declares the android api** that the application is going to use.
- It **lists the instrumentation classes**. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.

➔This is the required xml file for all the android application and located inside the root directory.

➔**AndroidManifest.xml file**

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

  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="15" />

  <application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name=".MainActivity"
      android:label="@string/title_activity_main" >
```

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

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

(b) How to use spinner in Android App? Explain with an example.

➔ we are going to display the country list. You need to use **ArrayAdapter** class to store the country list.

➔Let's see the simple example of spinner in android.

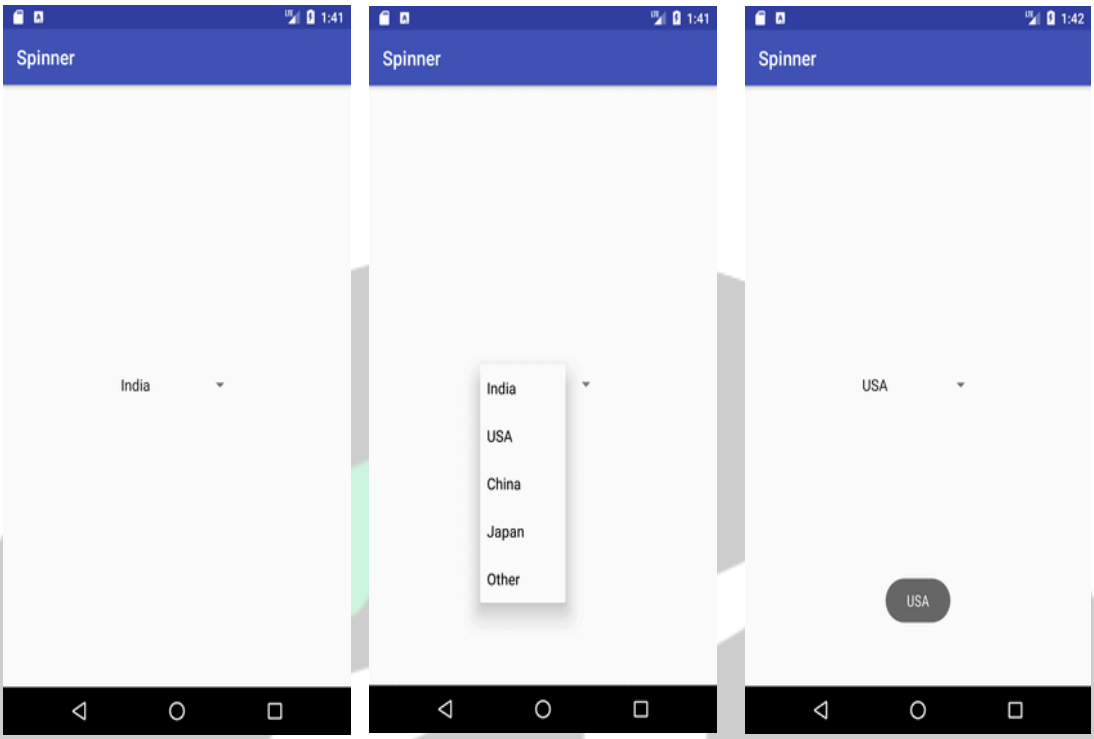
⇨ **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.spinner.MainActivity">

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="149dp"
        android:layout_height="40dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"  />

</RelativeLayout>
```

⇨ **MainActivity.java**

```
public class MainActivity extends AppCompatActivity implements
        AdapterView.OnItemSelectedListener {
    String[] country = { "India", "USA", "China", "Japan", "Other"};

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

        Spinner spin = (Spinner) findViewById(R.id.spinner);
        spin.setOnItemSelectedListener(this);


        ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,co
                                                                                untry);

        aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_iem);

        spin.setAdapter(aa);

    }

    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id) {

     Toast.makeText(getApplicationContext(),country[position] , Toast.LENGTH_LONG)
                                                                        .show();
    }
}
```

⇨ **Output:**



(c) What you mean by fragment in Android? Explain fragment with an example.

→ Android fragment is the part of activity, it is also known as sub-activity.

→ There can be more than one fragment in an activity.

→ Fragment represent multiple screen inside one activity.

→ Android fragment life cycle is affected by activity life cycle because fragments are included in activity.

→ Each fragment has its own life cycle methods that is affected by activity life cycle because fragments are embedded in activity.

→ The FragmentManager class is responsible to make interaction between fragment objects.

→ The lifecycle of android fragment is like the activity lifecycle.

→ There are 12 lifecycle methods for fragment.

⇨ **Example :**

⇨ **activity_main.xml**

```
<?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="fill_parent"
   android:layout_height="fill_parent"  >

   <fragment
      android:id="@+id/fragment1"
      android:name="example.javatpoint.com.fragmentexample.Fragment1"
      android:layout_width="0px"
      android:layout_height="match_parent"
      android:layout_weight="1"
      />

   <fragment
      android:id="@+id/fragment2"
      android:name="example.javatpoint.com.fragmentexample.Fragment2"
      android:layout_width="0px"
      android:layout_height="match_parent"
      android:layout_weight="1"
      />
</LinearLayout>
```

⇨ **File: fragment_fragment1.xml**

```
<FrameLayout 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:background="#F5F5DC"
   tools:context="example.javatpoint.com.fragmentexample.Fragment1">

   <TextView
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/hello_blank_fragment" />

</FrameLayout>
```

⇨ **File: fragment_fragment2.xml**

```
<FrameLayout 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:background="#F0FFFF"
    tools:context="example.javatpoint.com.fragmentexample.Fragment2">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/hello_blank_fragment" />

</FrameLayout>
```

⇨ **File: MainActivity.java**

```
public class MainActivity extends AppCompatActivity {

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

⇨ **File: Fragment1.java**

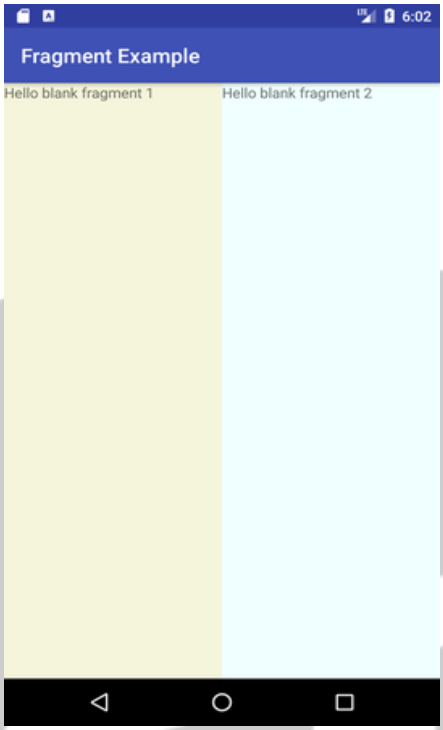
```
public class Fragment1 extends Fragment {
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override     public View onCreateView(LayoutInflater inflater, ViewGroup
container,                     Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_fragment1, container, false);
    }
}
```

⇨ **File: Fragment2.java**

```
public class Fragment2 extends Fragment {

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

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

        return inflater.inflate(R.layout.fragment_fragment2, container, false);
    }
}
```

⇨ **Output:**



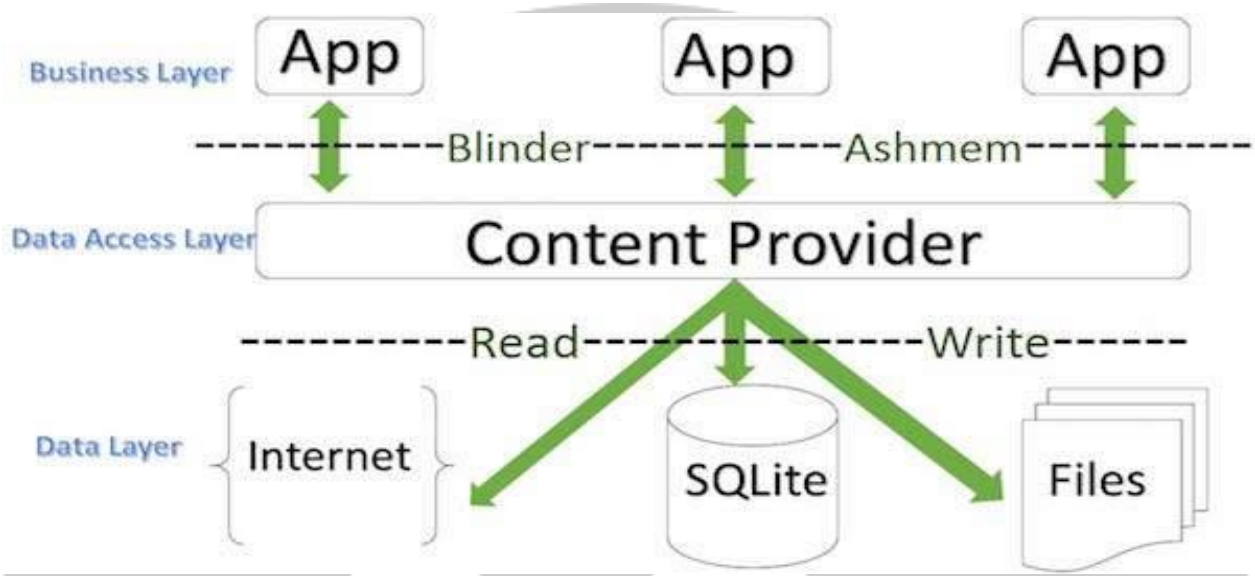
[Q.4]

(a) What is content resolver? Discuss in brief.

➔ A content provider component supplies data from one application to others on request.

➔Such requests are handled by the methods of the ContentResolver class.

➔A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network.



- ⇨ Content providers let you centralize content in one place and have many different applications access it as needed.
- ⇨ A content provider behaves very much like a database where you can query it, edit its content, as well as add or delete content using insert(), update(), delete(), and query() methods.
- ⇨ In most cases this data is stored in an **SQlite** database.

(b) What is the use of Broadcast Receiver? How to add it in Android App?

➔ **Broadcast Receivers** simply respond to broadcast messages from other applications or from the system itself.

➔These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

➔ There are following two important steps to make BroadcastReceiver works for the system broadcasted intents −

- Creating the Broadcast Receiver.
- Registering Broadcast Receiver

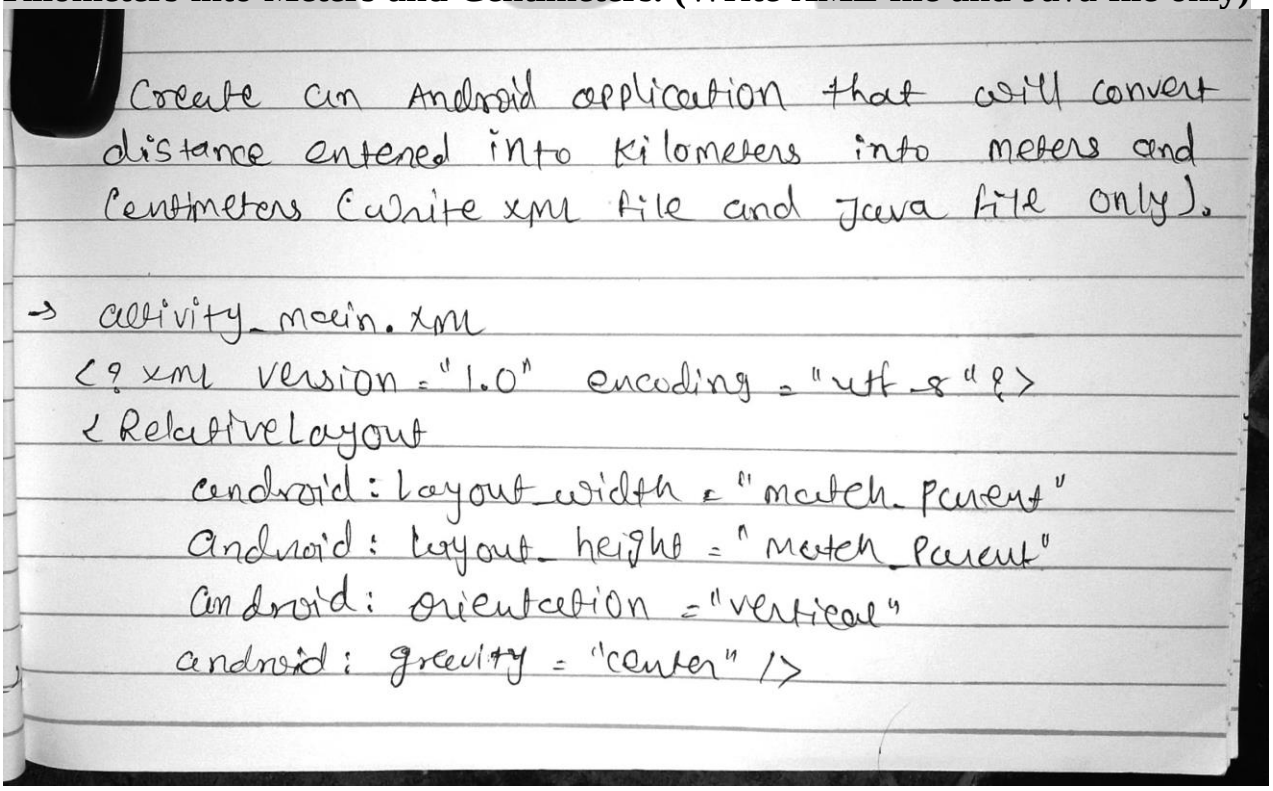
➔ There is one additional steps in case you are going to implement your custom intents then you will have to create and broadcast those intents.

➔Creating the Broadcast Receiver

⇨ A broadcast receiver is implemented as a subclass of **Broadcast Receiver** class and overriding the onReceive() method where each message is received as a **Intent** object parameter.

```
public class MyReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
  }
}
```

(c) Create an android application that will convert distance entered into Kilometers into Meters and Centimeters. (Write XML file and Java file only)



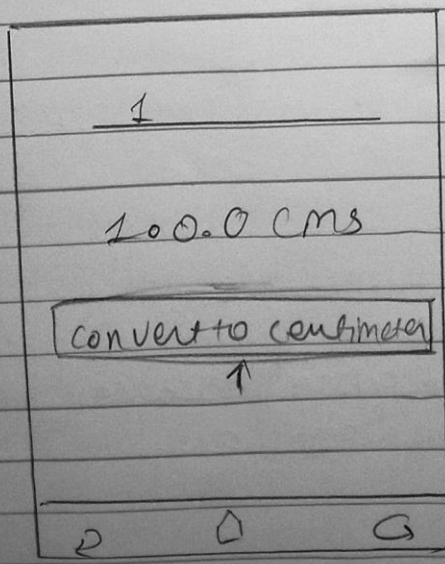
```
<EditText android:id = "MET"
    android : layout_width = " match_parent"
    android : layout_height = " wrap_content"
    android : inputType = "numberDecimal"
    android: hint = "Enter the length in meters"
    android : ems -"10" />

<TextView .
        android : id = " MET"
        android :layout_width = " match_parent"
        android : layout_height = "wrap_content"
        android : text = "centimetere'
        android : textsize = "24sp"
        android : gravity = "center"/>

< Button android :id = "@+id / mbutton"
    android : text = " convert to Centimetre"
    android : layout_width = "wrap_content"
    android : layout_height = "wrap_content'
    android : textStyle = "Bold" />
```

=> MainActivity.java

```
public class mainActivity extends AppCompat
Activity {
        @Override
    Protected void onCreate (Bundle SavedInstance-
        State) {
            super.onCreate (savedInstanceState);
            setContentView (R.layout.activity_main);
```

```
Button Mbutton;
EditText MET;
TextView MTV;

Mbutton = (MButton) findViewById(R.id.button);
MET = (MET) findViewById(R.id.MET);
MTV = (MTV) findViewById(R.id.MTV);

mbutton.setonClickListener(new view.onClickListener
(){
        @override
    Public void onClick(View v){
    Double convert = Double.ParseDouble(MET.
      getText().toString());
    MTV.setText(String.valueof(convert*100));
    MTV.setTextColor(Color.RED);
    }
});
}
}
```

O/P:-



OR

[Q.4]

(a) Explain the use of RadioButton and RadioGroup with an example.

➔

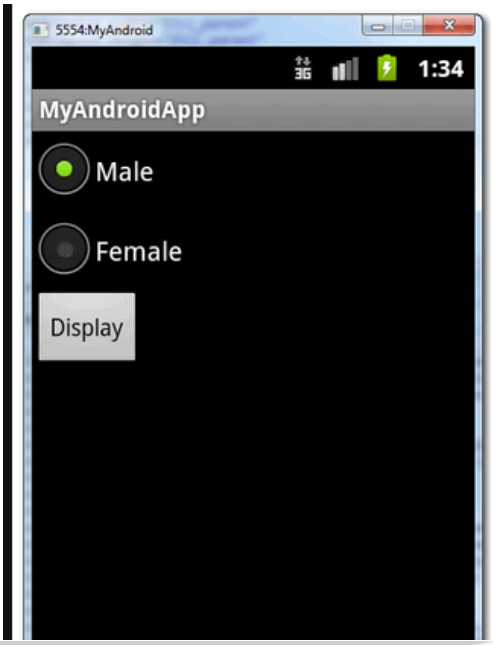
- **RadioButton** is a two states button which is either checked or unchecked. If a single radio button is unchecked, we can click it to make checked radio button. Once a radio button is checked, it cannot be marked as unchecked by user.
- RadioButton is generally used with ***RadioGroup***. RadioGroup contains several radio buttons, marking one radio button as checked makes all other radio buttons as unchecked.
- **File : res/layout/main.xml**

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

   <RadioGroup
      android:id="@+id/radioSex"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" >

         <RadioButton
         android:id="@+id/radioMale"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/radio_male"
         android:checked="true" />

         <RadioButton
          android:id="@+id/radioFemale"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/radio_female" />

</RadioGroup>

          <Button
           android:id="@+id/btnDisplay"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@string/btn_display" />

  </LinearLayout>
```

⇨ **File : MyAndroidAppActivity.java**

```
public class MyAndroidAppActivity extends Activity {

  private RadioGroup radioSexGroup;
  private RadioButton radioSexButton;
  private Button btnDisplay;

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

        addListenerOnButton();

  }

  public void addListenerOnButton() {

        radioSexGroup = (RadioGroup) findViewById(R.id.radioSex);
        btnDisplay = (Button) findViewById(R.id.btnDisplay);

        btnDisplay.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {

                        // get selected radio button from radioGroup
                        int selectedId = radioSexGroup.getCheckedRadioButtonId();

                        // find the radiobutton by returned id
                        radioSexButton = (RadioButton) findViewById(selectedId);

                        Toast.makeText(MyAndroidAppActivity.this,
                                radioSexButton.getText(), Toast.LENGTH_SHORT).show();

                }

        });

  }
}
```

⇨ **Output:-**



(b) What are the monetary advantages which you may get after publishing your application?

➔ x

(c) Write all necessary code to print all the files stored in DCIM folder of SD card.

➔ *File: Activity_Manifest.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="example.javatpoint.com.externalstorage">
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity android:name=".MainActivity">
         <intent-filter>
```

```
            <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

➔ **File: activity_main.xml**

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

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10" />

    <TextView
```

```
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="File Name:" />

     <TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Data:" />

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="save" />

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

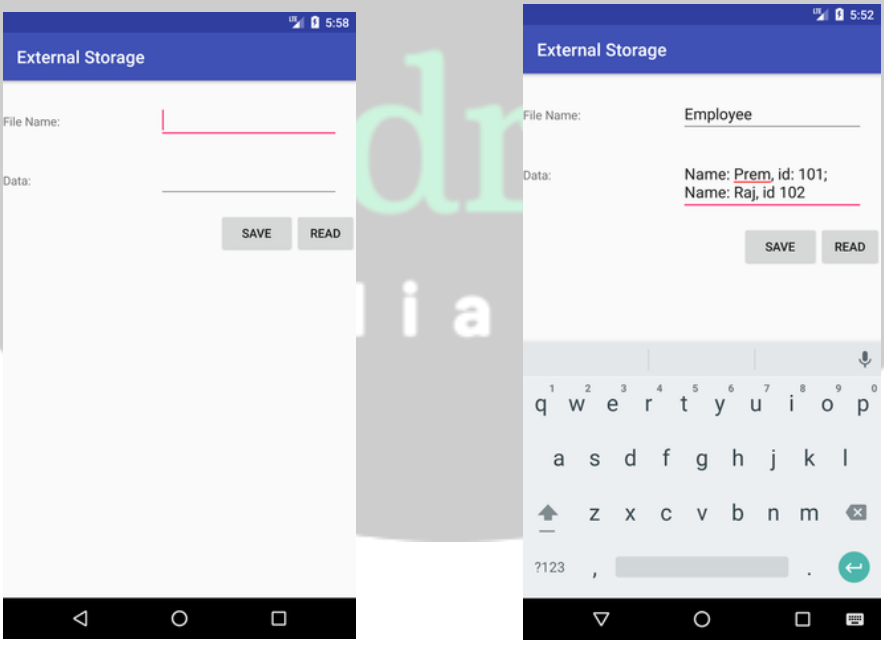
➔ **File: MainActivity.java**

```
  public class MainActivity extends AppCompatActivity {
    EditText editTextFileName,editTextData;
    Button saveButton,readButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   editTextFileName=findViewById(R.id.editText1);
```

```
editTextData=findViewById(R.id.editText2);
saveButton=findViewById(R.id.button1);
readButton=findViewById(R.id.button2);

saveButton.setOnClickListener(new View.OnClickListener(){

   @Override
   public void onClick(View arg0) {
      String filename=editTextFileName.getText().toString();
      String data=editTextData.getText().toString();

      FileOutputStream fos;
      try {
         File myFile = new File("/sdcard/"+filename);
         myFile.createNewFile();
         FileOutputStream fOut = new FileOutputStream(myFile);
         OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
         myOutWriter.append(data);
         myOutWriter.close();
         fOut.close();
Toast.makeText(getApplicationContext(),filename + "saved",Toast.LENGTH_ONG).show();
      } catch (FileNotFoundException e) {e.printStackTrace();}
      catch (IOException e) {e.printStackTrace();}
   }
});


readButton.setOnClickListener(new View.OnClickListener(){
   @Override
   public void onClick(View arg0) {
      String filename=editTextFileName.getText().toString();
      StringBuffer stringBuffer = new StringBuffer();
      String aDataRow = "";
      String aBuffer = "";
```

```
            try {
                File myFile = new File("/sdcard/"+filename);
                FileInputStream fIn = new FileInputStream(myFile);
                BufferedReader myReader = new BufferedReader(
                        new InputStreamReader(fIn));
                while ((aDataRow = myReader.readLine()) != null) {
                    aBuffer += aDataRow + "\n";
                }
                myReader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
            Toast.makeText(getApplicationContext(),aBuffer,Toast.LENGTH_LONG).show();
        }
    });
  }
}
```

➔ **Output :-**



[Q.5]

(a) What is widget?

➔ There are given a lot of **android widgets** with simplified examples such as Button, EditText, AutoCompleteTextView, ToggleButton, DatePicker, TimePicker, ProgressBar etc.

➔Android widgets are easy to learn. The widely used android widgets with examples are given below:

➔Android Button

- ⇨ Let's learn how to perform event handling on button click.

➔Android Toast

- ⇨ Displays information for the short duration of time.

➔Custom Toast

- ⇨ We are able to customize the toast, such as we can display image on the toast

➔ToggleButton

- ⇨ It has two states ON/OFF.

➔CheckBox

- ⇨ Let's see the application of simple food ordering.

➔AlertDialog

- ⇨ AlertDialog displays a alert dialog containing the message with OK and Cancel buttons.

➔Spinner

- ⇨ Spinner displays the multiple options, but only one can be selected at a time.

➔AutoCompleteTextView

- ⇨ Let's see the simple example of AutoCompleteTextView.

➔RatingBar

- ⇨ RatingBar displays the rating bar.

➔DatePicker

- ⇨ Datepicker displays the datepicker dialog that can be used to pick the date.

➔TimePicker

- ⇨ TimePicker displays the timepicker dialog that can be used to pick the time.

➔ProgressBar

- ⇨ ProgressBar displays progress task.

(b) What is parsing? Discuss how you can perform parsing using JSON in Android application.

➔ **JSON stands for JavaScript Object Notation**.

➔ JSON is used for data interchange (posting and retrieving) from the server.

➔Hence knowing the syntax and it's usability is important. JSON is the best alternative for XML and its more readable by human.

➔JSON is **language independent**. Because of language in-dependency we can program JSON in any language (Java/C/C++).

➔A JSON response from the server consists of many fields. An example JSON response/data is given below.

➔We'll use it as a reference and implement it in our application.

```
{
  "title":"JSONParserTutorial",
  "array":[
    {
    "company":"Google"
    },
    {
      "company":"Facebook"
    },
    {
    "company":"LinkedIn"
    },
    {
      "company" : "Microsoft"
    },
    {
      "company": "Apple"
    }
    ],
    "nested":{
    "flag": true,
    "random_number":1
    }
}
```

➔ We've create a random JSON data string from this page. It's handy for editing JSON data. A JSON data consists of 4 major components that are listed below:

1. **Array**: A **JSONArray** is enclosed in square brackets ([). It contains a set of objects
2. **Object**: Data enclosed in curly brackets ({) is a single JSONObject. Nested JSONObjects are possible and are very commonly used
3. **Keys**: Every JSONObject has a key string that's contains certain value
4. **Value**: Every key has a single value that can be of any type string, double, integer, boolean etc.

(c) Write a code to send e-mail from Android App using the concept of explicit intent.

➔ ***File: activity_main.xml***

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

   <EditText
      android:id="@+id/editText1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
     android:ems="10" />

   <EditText
      android:id="@+id/editText2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:ems="10" >

      <requestFocus />
   </EditText>

   <EditText
      android:id="@+id/editText3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:ems="10" />

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
```

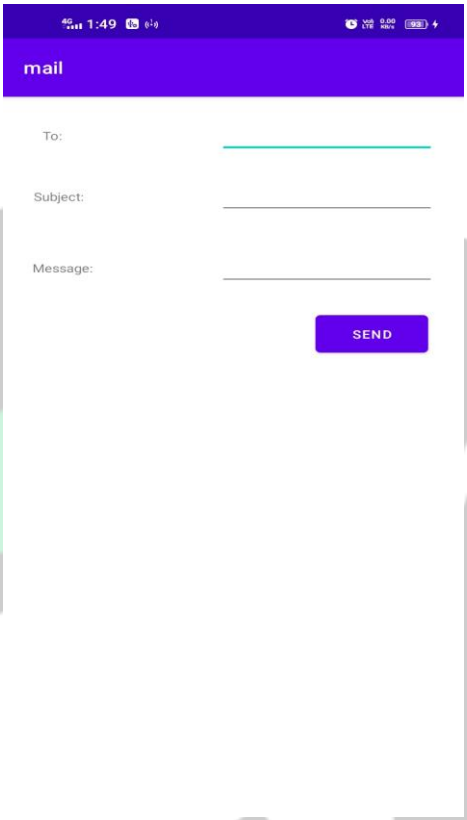
```
        android:layout_height="wrap_content"
        android:text="To:" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Subject:" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Message:" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send" />

</RelativeLayout>
```

➔ **File: MainActivity.java**

```
public class MainActivity extends Activity {
    EditText editTextTo,editTextSubject,editTextMessage;
    Button send;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        editTextTo=(EditText)findViewById(R.id.editText1);
        editTextSubject=(EditText)findViewById(R.id.editText2);
        editTextMessage=(EditText)findViewById(R.id.editText3);

        send=(Button)findViewById(R.id.button1);

        send.setOnClickListener(new OnClickListener(){

            @Override
            public void onClick(View arg0) {
                String to=editTextTo.getText().toString();
                String subject=editTextSubject.getText().toString();
                String message=editTextMessage.getText().toString();


                Intent email = new Intent(Intent.ACTION_SEND);
                 email.putExtra(Intent.EXTRA_EMAIL, new String[]{ to});
                 email.putExtra(Intent.EXTRA_SUBJECT, subject);
                 email.putExtra(Intent.EXTRA_TEXT, message);

                 email.setType("message/rfc822");

                 startActivity(Intent.createChooser(email, "Choose an Email client :"));

            }
        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

➔ **Output:-**



OR

[Q.5]

(a) What is Service? Differentiate between Activity and Service.

➔ What is Service?

- **Android service** is a component that is *used to perform operations on the background* such as playing music, handle network transactions, interacting content providers etc. It doesn't has any UI (user interface).
- The service runs in the background indefinitely even if application is destroyed.
- Moreover, service can be bounded by a component to perform interactivity and inter process communication (IPC).
- The android.app.Service is subclass of ContextWrapper class.

➔ Differentiate between Activity and Service.

Activity	Services
Activity is the foreground task that runs right in front of the user. All activities interact with the user. It creates a window by loading a UI layout where you can place your UI.	Service is an application component that allows it to run in the background. It carries out long-running operations without interacting with the user.
Android does not kill activity separately. It kills the whole app process with all activities.	It is not killed when an application is killed.
It is used with both small and long tasks.	It is used with smaller tasks with no UI.

(b) Discuss AsyncTaskLoader in detail.

➔link more info (https://o7planning.org/12753/android-asynctaskloader#:~:text=AsyncTaskLoader%20is%20used%20to%20perform,be%20updated%20to%20the%20interface.)

➔ AsyncTaskLoader is used to perform an asynchronous task in the background of the application, so the user can also interact with the application during that process. As soon as the task is completed, the result will be updated to the interface.

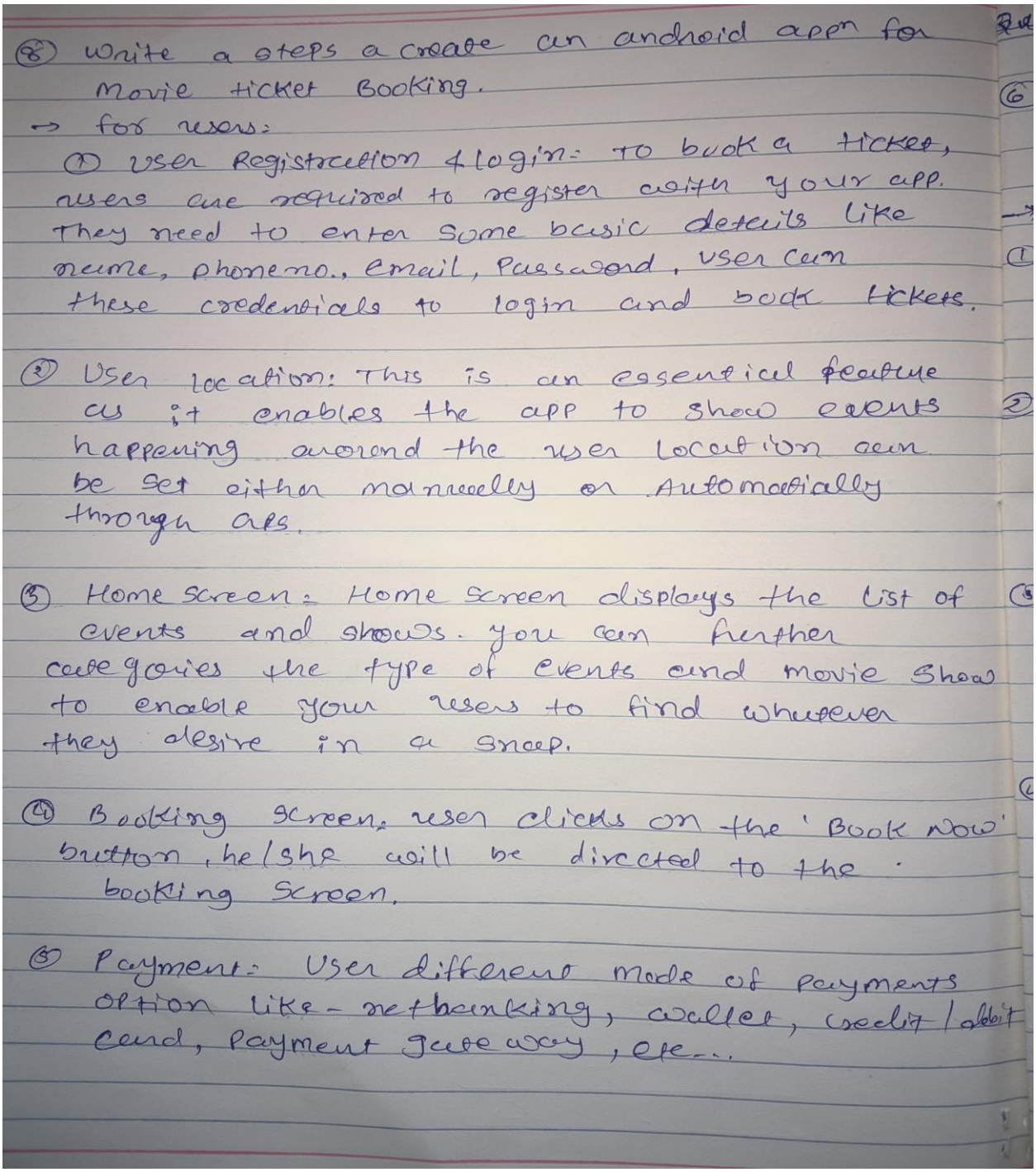
➔AsyncTaskLoader performs the same functions as AsyncTask; however, AsyncTaskLoader is thought to be better, for the following reasons:

➔AsyncTask and AsyncTaskLoader have different behaviors when the configuration of the device changes, for example, if the user rotates the device screen, Activity can be destroyed and re-created. In that case:

- ⇨ AsyncTask will be re-executed and a new Thread will be created, whereas the old Thread will become separate and uncontrolled.
- ⇨ AsyncTaskLoader will be re-used based on the Loader ID that has been registered with the LoaderManager before. As a result, it prevents the duplication of background tasks, and avoids the creation of useless tasks.

(c) Write a steps to create an android application for Movie Ticket Booking.

➔



(8) Write a steps a create an android appn for Movie ticket Booking.

→ for users:

(1) User Registration & login: To book a ticket, users are required to register with your app. They need to enter some basic details like name, phone no., email, Password, user can these credentials to login and book tickets.

(2) User location: This is an essential feature as it enables the app to show events happening around the user location can be set either manually or Automatically through GPS.

(3) Home Screen: Home Screen displays the list of events and shows. You can further categories the type of events and movie shows to enable your users to find whatever they desire in a snap.

(4) Booking Screen: user clicks on the 'Book Now' button, he/she will be directed to the booking Screen.

(5) Payment: User different mode of payments option like - netbanking, wallet, credit / debit card, Payment gateway, etc..

***

[Q.1]

(a) What is the use of Android Manifest file?

➔ Refer **(**SUMMER 2022 **(Q:2) B**)

(b) Define: Class, Object, Encapsulation, Inheritance

➔

- ⇨ **Class:**
  - A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.
- ⇨ **Object:**
  - Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.
- ⇨ **Encapsulation:**
  - Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.
- ⇨ **Inheritance:**
  - When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
- ⇨ **Polymorphism :**
  - If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.
- ⇨ **Abstraction**
  - Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

(c) Explain Android architecture with proper diagram.

➔ Refer **(**SUMMER 2022 **(Q:1) C**)

[Q.2]

(a) What is AVD? Explain the process of creating AVD in Android application development.

➔ The **Android emulator** is an **Android Virtual Device (AVD),** which represents a specific Android device.

➔We can use the Android emulator as a target device to execute and test our Android application on our PC.

➔The Android emulator provides almost all the functionality of a real device. We can get the incoming phone calls and text messages.

➔It also gives the location of the device and simulates different network speeds. Android emulator simulates rotation and other hardware sensors.

➔ Testing Android applications on emulator are sometimes faster and easier than doing on a real device. For example, we can transfer data faster to the emulator than to a real device connected through USB.

➔The Android emulator comes with predefined configurations for several Android phones, Wear OS, tablet, Android TV devices.

➔ **Explain the process of creating AVD in Android application development.**

- ⇨ **Step 1:** Go to **Tools** > **AVD Manager**.
- ⇨ **Step 2:** Now click on **Create Virtual Device**.
- ⇨ **Step 3:** A pop-up window will be there and here we select the category Phone because we are creating android app for mobile and select the model of mobile phone we want to install.
- ⇨ **Step4:** Here we select the android version to download like **Q**, **Pie**, **Oreo** etc and click **Next** button.
- ⇨ **Step 5:** Click the **finish** button to complete the installation.
- ⇨ **Step 6:** Now we can select the virtual device we want to run as emulator can click on the **run** icon.
- ⇨ **Step 7:** Finally our virtual device is ready to run our android app.

(b) Explain basic building blocks/components of Android Application.

➔ A piece of code with a well-defined life cycle e.g. Activity, Receiver, Service, etc, is what an android component can be simply understood as. Activities, views, intents, services, content providers, fragments, and AndroidManifest.xml are the core building blocks or fundamental components of android.

- ⇨ Activity:
  - Being similar to a Frame in AWT, an activity as a class represents a single screen. It is a core building block, i.e., the fundamental component of android.
- ⇨ View:
  - The UI element including button, label, text field, etc, and anything that one can see is a view.
- ⇨ Intent:
  - Along with invoking the components, the Intent is used for many purposes including:
    - Start the service
    - Launch an activity
    - Display a web page
    - Display a list of contacts
    - Broadcast a message
    - Dial a phone call etc.
- ⇨ Service:
  - Being a background process the service can run for a long time.
  - The local and remote are the two types of services. Within the application, the local service is accessed. While from other applications running on the same device the remote service can be accessed remotely.
- ⇨ Content Provider:
  - To share the data between the applications, the Content Providers are used.
- ⇨ Fragment:
  - Being a part of an activity, one or more fragments can be displayed on the screen at the same time by the activity.

- ⇨ AndroidManifest.xml:
  - Information about activities, content providers, permissions, etc is in the AndroidManifest.xml which is like the web.xml file in Java EE.

- ⇨ Android Virtual Device (AVD):
  - To test an android application without using a mobile or a tablet, the Android Virtual Device or AVD is used. To emulate different types of real devices, an AVD can be created in different configurations.

(c) What is Activity? Draw and explain Activity lifecycle in detail.

➔ Refer **(**SUMMER 2022 **(Q:3) C**)

OR

(c) What is Fragment? Draw and explain the lifecycle of a fragment.

➔ **Android Fragment** is the part of activity, it is also known as sub-activity. There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity.

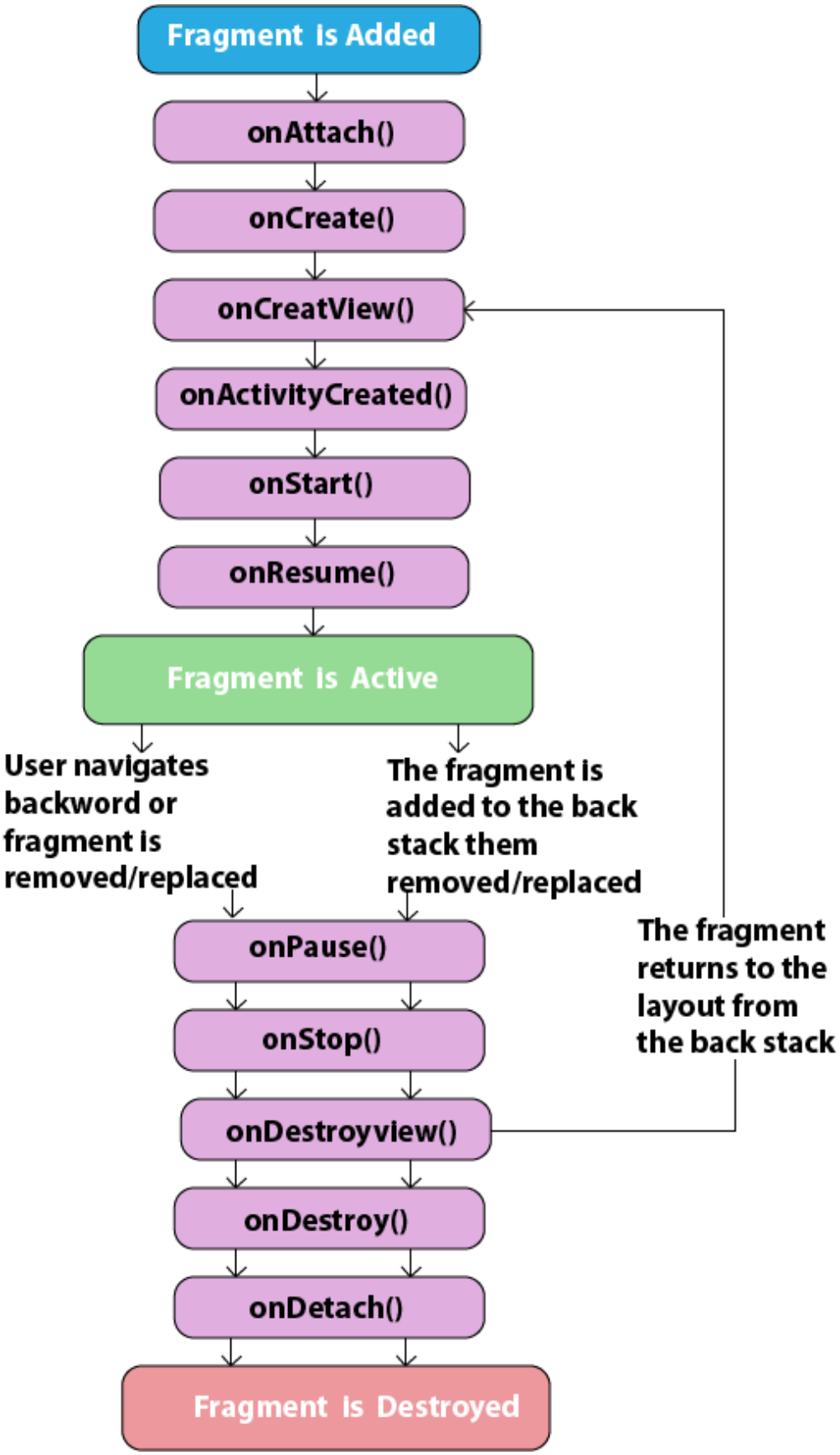
➔Android fragment lifecycle is affected by activity lifecycle because fragments are included in activity.

➔Each fragment has its own life cycle methods that is affected by activity life cycle because fragments are embedded in activity.

➔The **Fragment Manager** class is responsible to make interaction between fragment objects.

➔**Android Fragment Lifecycle**

➔The lifecycle of android fragment is like the activity lifecycle. There are 12 lifecycle methods for fragment.



No.	Method	Description
1)	onAttach(Activity)	it is called only once when it is attached with activity.
2)	onCreate(Bundle)	It is used to initialize the fragment.
3)	onCreateView(LayoutInflater, ViewGroup, Bundle)	creates and returns view hierarchy.
4)	onActivityCreated(Bundle)	It is invoked after the completion of onCreate() method.
5)	onViewStateRestored(Bundle)	It provides information to the fragment that all the saved state of fragment view hierarchy has been restored.
6)	onStart()	makes the fragment visible.
7)	onResume()	makes the fragment interactive.
8)	onPause()	is called when fragment is no longer interactive.
9)	onStop()	is called when fragment is no longer visible.
10)	onDestroyView()	allows the fragment to clean up resources.
11)	onDestroy()	allows the fragment to do final clean up of fragment state.
12)	onDetach()	It is called immediately prior to the fragment no longer being associated with its activity.

[Q.3]

(a) What is Layout? What are the advantages of setting a Layout?

➔ A layout defines the structure for a user interface in your app, such as in an activity. All elements in the layout are built using a hierarchy of View and ViewGroup objects.

⇨ X

(b) List out various layouts available in Android. Explain any one in detail.

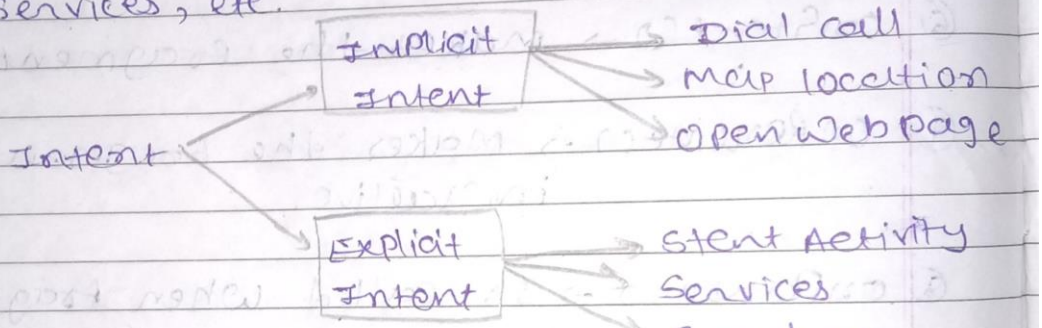
➔ Refer **(**WINTER 2019 **(Q:3) B**)

(c) What is an Intent? Explain types of Intent in Android with example.

Q-4 Implicit and Explicit Intent.

→ The intent is the main component of Android app development.

→ The intent is the medium to pass between components such as activities, Content Providers, broadcast receivers, Services, etc.



Teacher's Signature.....

⇒ Implicit Intent :-

→ Implicit intent doesn't specify the component in the app.

→ In such a case, intent provides information on available components provided by the system that is to be invoked.

⇒ Implicit intent example :-

→ A button on click of which you will redirect to a web page.

→ If your device has multiple browsers then the options popup will open and show them all.

example :-

⇒ activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/
apk/res/android"
xmlns:app="http://schemas.android.com/
apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="....MainActivity">
```

Teacher's Signature.....

```
<EditText

android:id = "@+id/edittext"
android:layout_width = "wrap_content"
android:layout_height = "wrap_content" />

<Button
    android:id = "@+id/button"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "Visit" />
</android.support.constraint.ConstraintLayout>
```

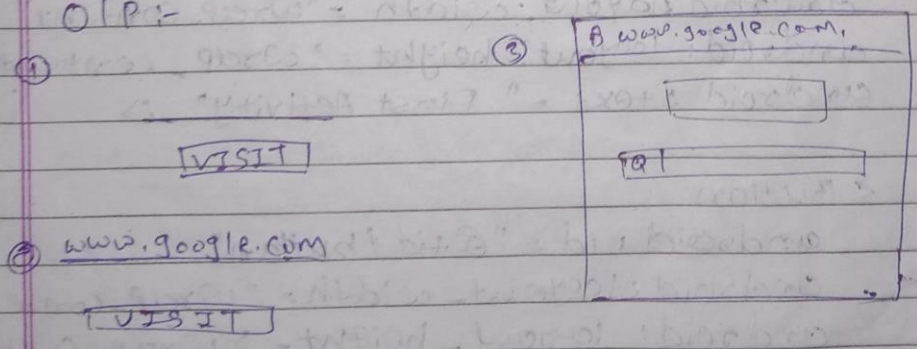
⇒ File : MainActivity.java

```
Package example.abc.com.implicitintent;

import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

Public class MainActivity extends
    AppCompatActivity {
        Button button;
        EditText editText;
```

Teacher's Signature.....

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

    button = findViewById(R.id.button);
    editText = findViewById(R.id.editText);

button.setOnClickListener(new View.OnClickListener
                                    () {
    @Override
    Public void onClick(View view) {
    String url = editText.getText().toString();
    Intent intent = new Intent(Intent.
            ACTION_VIEW, Uri.parse(url));
    startActivity(intent);
      }
  });
 }
}
```

O/P :-



Teacher's Signature.....

Date : / /

② Explicit Intent :

→ An explicit intent is used for starting an activity or service within the same application package.

⇒ activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?
<RelativeLayout
xmlns:android="http://schemas.andro
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:context=".example.abc.com.explicit
.FirstActivity">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="First Activity" />

<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="call second activity"
android:onClick="callSecondActivity" />
</RelativeLayout>
```

Teacher's Signature.....

→ file : MainActivityOne.java

```
Package example.abc.com.explicitintent;
import android.content.Intent;
import android.support.v7.app.AppCompat
Activity;
import android.os.Bundle;
import android.view.View;

Public class FirstActivity extends
    AppCompatActivity {
      @Override
Protected void onCreate(Bundle SavedInstance
                        State) {
  super.onCreate(SavedInstanceState);
  setContentView(R.layout.activity_first);
}
Public void callSecondActivity(View view)
 {
     Intent i = new Intent(getApplication-
                context(), secondActivity.class);
     startActivity(i);
   }
}
```

⇒ activitytwo_main.xml

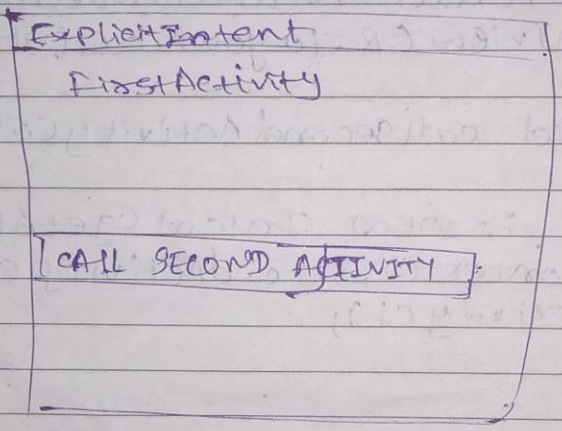
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:"http://schemas.android.com/
  apk/res/android"
```

Teacher's Signature.....

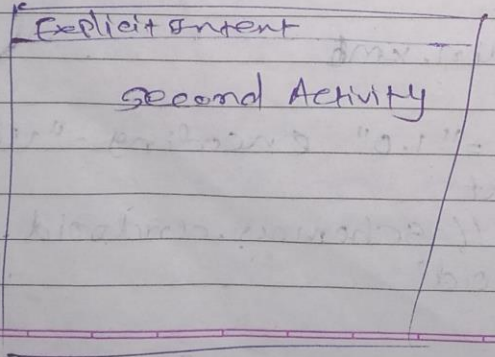
```
xmlns:tools = "http://schemas.android.
              com/tools"
android:layout_width = "match_parent"
android:layout_height = "match_parent"
tools:context = "example.abc.com.explicitintent.
                SecondActivity">

  <TextView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "Second Activity" />

</RelativeLayout>
```

→ O/P:

①



②



Teacher's Signature.....

OR

[Q.3]

(a) Explain Checkbox in Android with example.

➔ Android CheckBox is a type of two state button either checked or unchecked.

➔There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

➔Android CheckBox class is the subclass of CompoundButton class.

- ⇨ **Refer link (Example :- javapoint.com)**

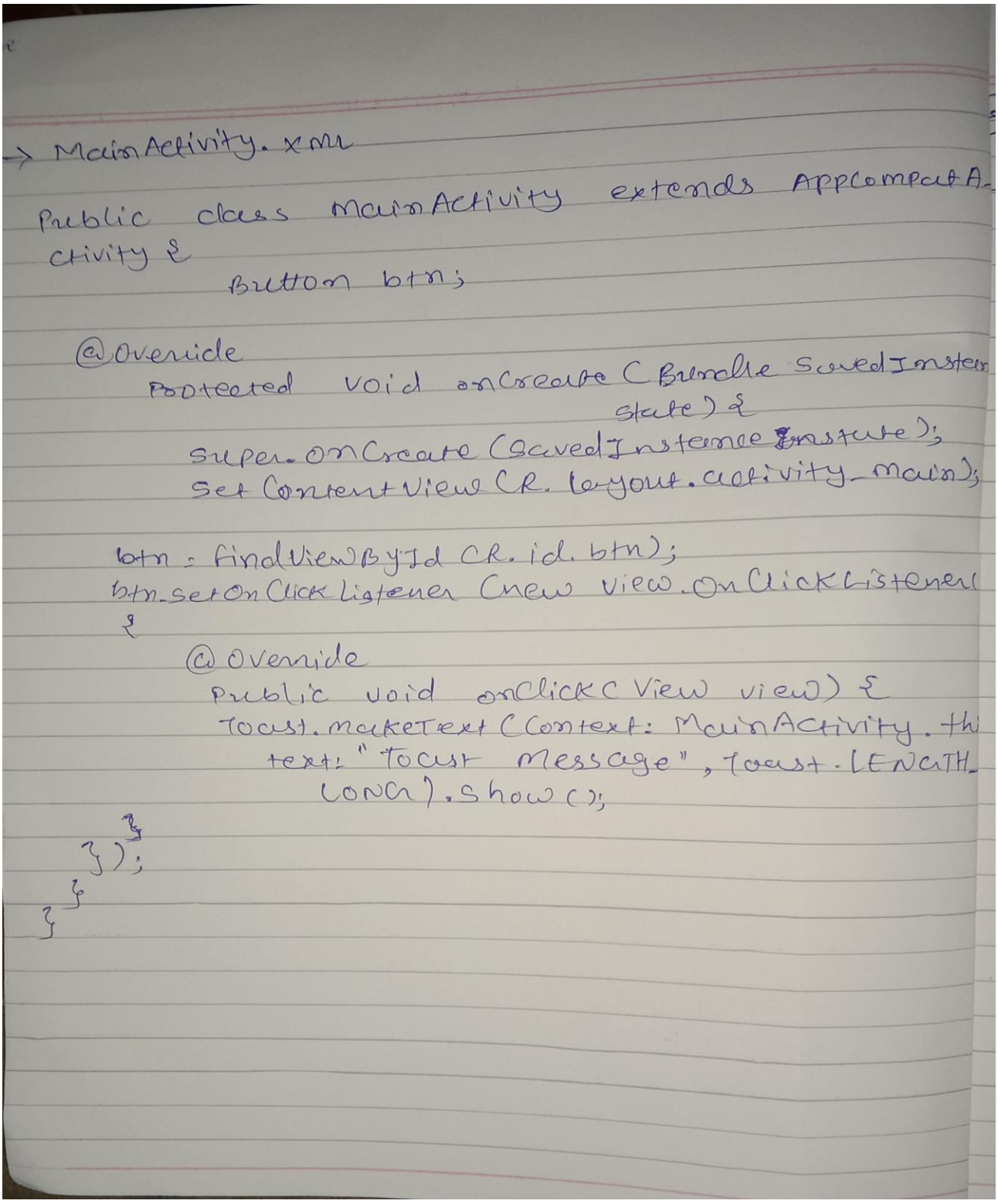
(b) Write code to display Toast Message on click of Button.

➔

Q-1 Write code to display Toast message onClick button.

→ activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_gravity="center"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn"
        android:layout_width="185sp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="720sp"
        android:layout_marginBottom="20sp"
        android:gravity="center"
        android:text="Toast Button Show"
        android:textsize="20sp" />

</RelativeLayout>
```



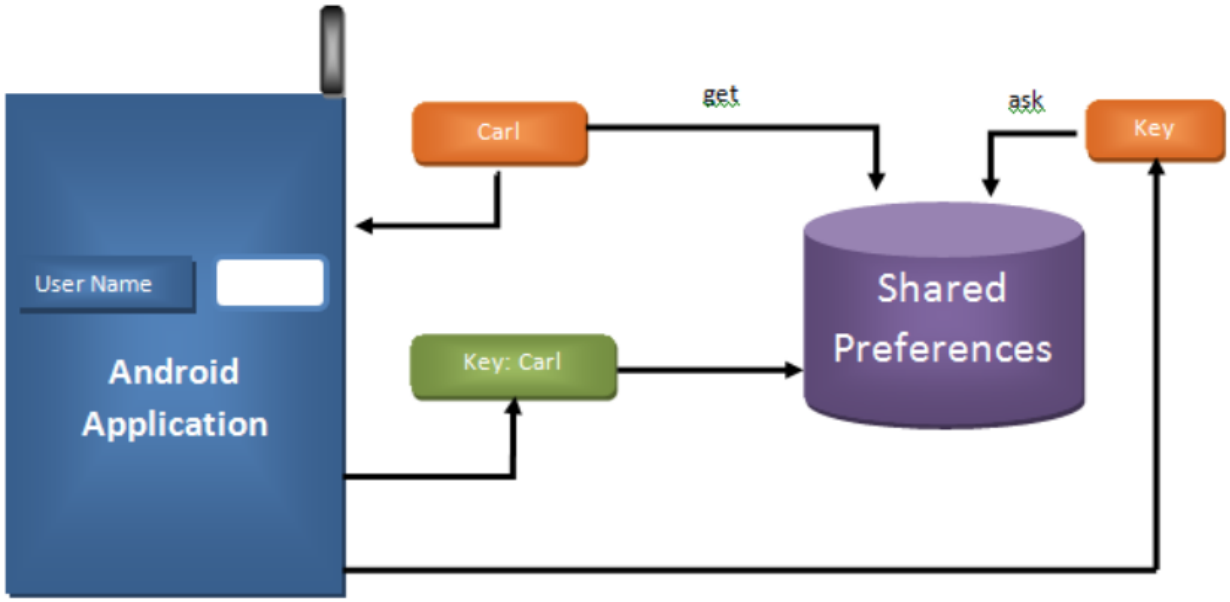
(c) Explain different types of menus in android with example.

➔ Refer **(**SUMMER 2022 OR **(Q:3) A**)

[Q-4]

(a)Explain the use of Shared Preferences.

➔ Android provides many ways of storing app data such as an SQLiteDatabase, saving a data text file, etc. One of the ways is called SharedPreference. SharedPreferences allows you to save and require data in the form of keys and values and provides a simple method to read and write them.



➔The primary purpose of SharedPreference is to store user-specified configuration details, such as settings, and to keep the user logged in to the app. We can make use of SharedPreference in situations where we don't need to store a lot of data and we don't require any specific structure.

➔SharedPreference is used only when you need to save a small amount of data as a key, and value pair. Do not use SharedPreference to manage a larger amount of app data using other methods like an SQLiteDatabase.

➔SharedPreference uses two ways to save data. The first is with activity-based preferences and the second is creating custom preferences. In activity preferences, the users have to call function **getPreference()** for the activity class. In custom preferences, the users have to call the function **getSharedPreference(). **

➔HOW TO USE SHAREDPREFERENCE

➔ In Android, we can create a new SharedPreference file or access an existing one by calling one of these methods:

- **getSharedPreferences() -** This method is used if you need multiple SharedPreference files identified by name, which you specify with the first parameter. This method calls from any context in your app.
- **getPreferences() -** This method is used only if you need one preference file for the activity. In this method, we don't require a name as it will be the only preference file for this activity.
- **Write to SharedPreferences**

  To write to a SharedPreference file we can create a **SharedPreferences.Editor** by calling **edit()** on your sharedpreference.
- **putInt()** and **putString() -** These methods pass the keys and values.
- **apply()** and **commit() -** These methods save the changes. Use **apply()** method to write the updates to disk asynchronously and use **commit()** method to write the data to disk synchronously.
- **Read from SharedPreferences**

  To retrieve values from a SharedPreference file call these methods:

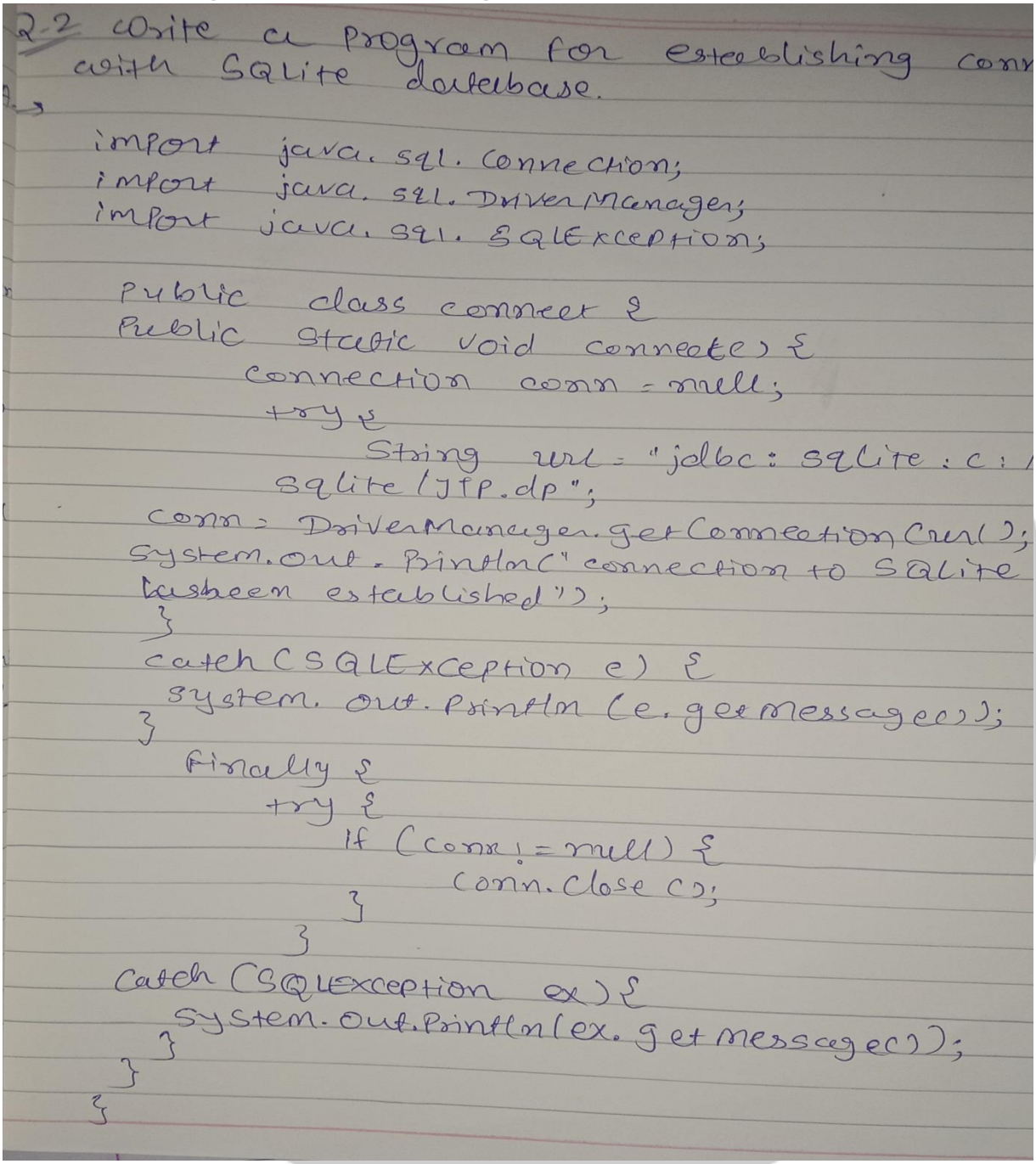
- **getInt()** and **getString() -** These methods provide the key for the value you want.

(b)Explain Internal vs. External storage.

Internal Memory	External Memory
Volatile Memory.	Non- Volatile.
Internal Memory is also called **"Primary Memory"** or **"Main Memory"** or **"Semiconductor Memory".**	External or secondary memory is also called **"AUXILLARY MEMORY"** and **"PERMANENT MEMORY."**
They are installed internally.	They are attached using cables and wires.
Stores small amount of data and information.	Capable of storing huge amount of data and informaiton.
They are faster compared to external memory.	They are slower than internal memory.
Examples of internal memory ***RAM (RANDOM ACCESS MEMORY)*** ***ROM (READ-ONLY MEMORY)***	Examples of Secondary storage ***Computer Hard Disk Drive.*** ***Pen Drives.*** ***SSD {Solid State Drives}.*** ***Optical Disks.*** ***Cloud Storage***
They are very important memory computer system can not boot if absent.	The are installed additionallly for better storage capacity.
They directly interact with central processing unit.	They do not directly interact with CPU.
They are expensive.	They are cheaper.

The internal memory is faster, smaller, and lighter, consuming less power	They require more power than internal memory.
Every computer needs a primary memory to work correctly	Computer can function even if secondary memory is absent.
Their size is measured in **MB** and **GB**.	Their capacity is measured in **GB** and **TB**

(c) Write a Program for establishing connection with SQLite database.



2-2 Write a program for establishing connection with SQLite database.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

Public class connect {
Public static void connect() {
        Connection conn = null;
        try {
            String url = "jdbc:sqlite:c:/
        sqlite/JTP.db";
    conn = DriverManager.getConnection(url);
    System.out.println("connection to SQLite
    has been established");
    }
    catch(SQLException e) {
     System.out.println(e.getMessage());
    }
      Finally {
          try {
              if (conn != null) {
                  conn.close();
              }
          }
    Catch(SQLException ex){
      System.out.println(ex.getMessage());
      }
    }
}
```

Public static void main(String args[])

{

connect( );

} }

OR

[Q-4]

(a) What is parsing? Discuss how can you perform parsing using JSON in Android application.

➔Refer (WINTER 2019 **(Q-5)B)**

(b) Discuss AsyncTask in detail

➔Refer (SUMMER 2022 **(Q-4) B)**

(c) Write Code to insert Contact Details (cID, cName, cPhoneNumber) in SQLite Database in Android.

➔**Refer (**WINTER 2019 **OR (Q-2) C)**

[Q-5]

(a) How to add notification in Android App? Explain with an example.

➔Android Notification provides short, timely information about the action happened in the application, even it is not running. The notification displays the icon, title and some amount of the content text.

➔Android Notification Example

- ⇨ In this example, we will create a notification message which will launch another activity after clicking on it.

➔activity_main.xml

- ⇨ Add the following code in an activity_main.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="example.javatpoint.com.androidnotification.MainActivity">

  <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/button"
      android:layout_marginBottom="112dp"
      android:layout_marginEnd="8dp"
      android:layout_marginStart="8dp"
      android:text="Notify"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent" />

</android.support.constraint.ConstraintLayout>
```

➔**activity_notification_view.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/a
pk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="example.javatpoint.com.androidnotification.NotificationView">

<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
 android:textAppearance="@style/Base.TextAppearance.AppCompat.Medium"
</android.support.constraint.ConstraintLayout>
```

➔ **MainActivity.java**

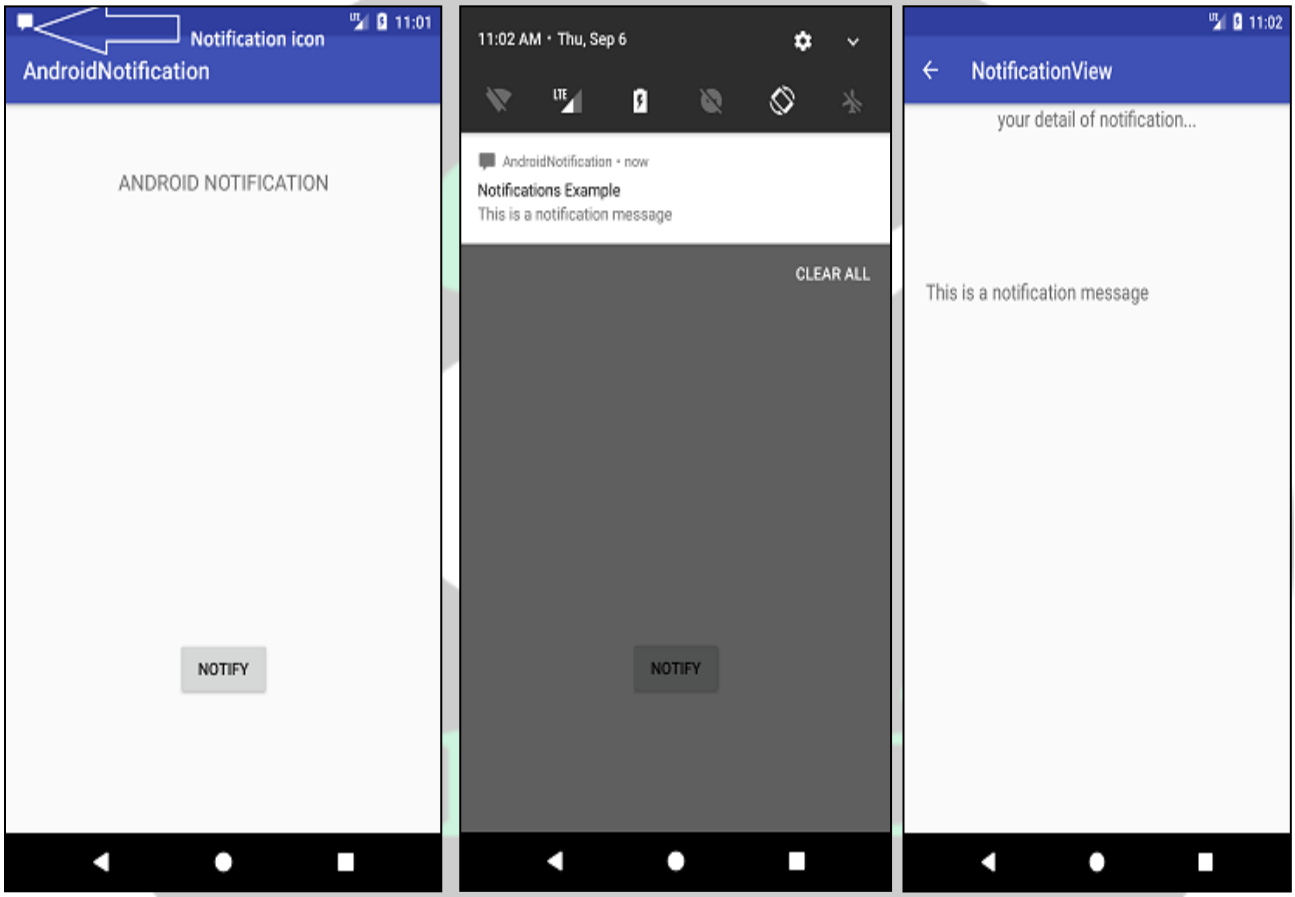
```
public class MainActivity extends AppCompatActivity {
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addNotification();
            }
        });
    }

    private void addNotification() {
        NotificationCompat.Builder builder =
                new NotificationCompat.Builder(this)
                        .setSmallIcon(R.drawable.messageicon) //set icon for notification
                        .setContentTitle("Notifications Example") //set title of notification
```

```
                .setContentText("This is a notification message")//this is notification message
                .setAutoCancel(true) // makes auto cancel of notification
                .setPriority(NotificationCompat.PRIORITY_DEFAULT); //set priority of notification


        Intent notificationIntent = new Intent(this, NotificationView.class);
        notificationIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        //notification message will get at NotificationView
        notificationIntent.putExtra("message", "This is a notification message");

        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, notificationIntent,
                PendingIntent.FLAG_UPDATE_CURRENT);
        builder.setContentIntent(pendingIntent);

        // Add as notification
        NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICA
TION_SERVICE);
        manager.notify(0, builder.build());
    }
}
```

➔ **NotificationView.java**

```
package example.javatpoint.com.androidnotification;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;

public class NotificationView extends AppCompatActivity {
    TextView textView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_notification_view);
```

```
        textView = findViewById(R.id.textView);
        //getting the notification message
        String message=getIntent().getStringExtra("message");
        textView.setText(message);
    }
}
```



(b) What is Geocoding and Reverse Geocoding? Explain it with example

➔Searching location in Google Map API is done through **Geocoder** class. Geocoder class is used to handle *geocoding* and *reverse geocoding*.

➔Geocoding is a process in which street address is converted into a coordinate (latitude,longitude). Reverse geocoding is a process in which a coordinate (latitude,longitude) is converted into an address.

➔Methods of Geocoder class

1. **List<Address> getFromLocation(double latitude, double longitude, int maxResults):** This method returns an array of Address which specifies the surrounding latitude and longitude.
2. **List<Address> getFromLocationName(String location, int results, double leftLatitude, double leftLongitude, double rightLatitude, double rightLongitude):** This method returns an array of Address which describes the given location such as place, an address, etc.
3. **List<Address> getFromLocationName(String location, int results):** This method returns an array of Address which describes te given location such as place, an address, etc.
4. **static boolean isPresent():** This method returns true if the methods getFromLocation() and getFromLocationName() are implemented.

For Example Refer- https://www.javatpoint.com/android-google-map-search-location-using-geocodr

(c) Which are the types of animations supported in Android? Explain any one in detail.

➔The animations are basically of three types as follows:

1. Property Animation
2. View Animation
3. Drawable Animation

1. Property Animation

➔Property Animation is one of the robust frameworks which allows animating almost everything. This is one of the powerful and flexible animations which was introduced in Android 3.0. Property animation can be used to add any animation in the CheckBox, RadioButtons, and widgets other than any view.

2. Drawable Animation

➔Drawable Animation is used if you want to animate one image over another. The simple way to understand is to animate drawable is to load the series of drawable one after another to create an animation. A simple example of drawable animation can be seen in many apps Splash screen on apps logo animation.

3.View Animation:

➔This is the simplest animation used in Android. It define the properties of our Views that should be animated using a technique called Tween Animation.It take the following parameters i.e. size, time duration , rotation angle, start value , end value, and perform the required animation on that object.You can execute the animation by specifying transformations on your View. This can be done in XML resource files or programmatically.

➔Android View animation can make animation on any View objects, such as ImageView, TextView or Button objects. View animation can only animate simple properties like position, size, rotation, and the alpha property that allows you animate the transparency of a View.

➔The drawback of this mechanism is that it can only be applied to Views.

OR

[Q-5]

(a)Explain the use of Broadcast Receiver in Android.

➔Refer (WINTER 2019 **(Q-4) B)**

(b) What is Service? Discuss various Service life cycle methods in detail.

➔Refer (SUMMER 2022 **OR (Q-3) C)**

(c) How can you publish your application in Google Play Store? Explain the entire process.

➔Refer (SUMMER 2022 **OR (Q-5) B)**