First Part

▼ TOC

Q1) Differentiate between Views and ViewGroups?

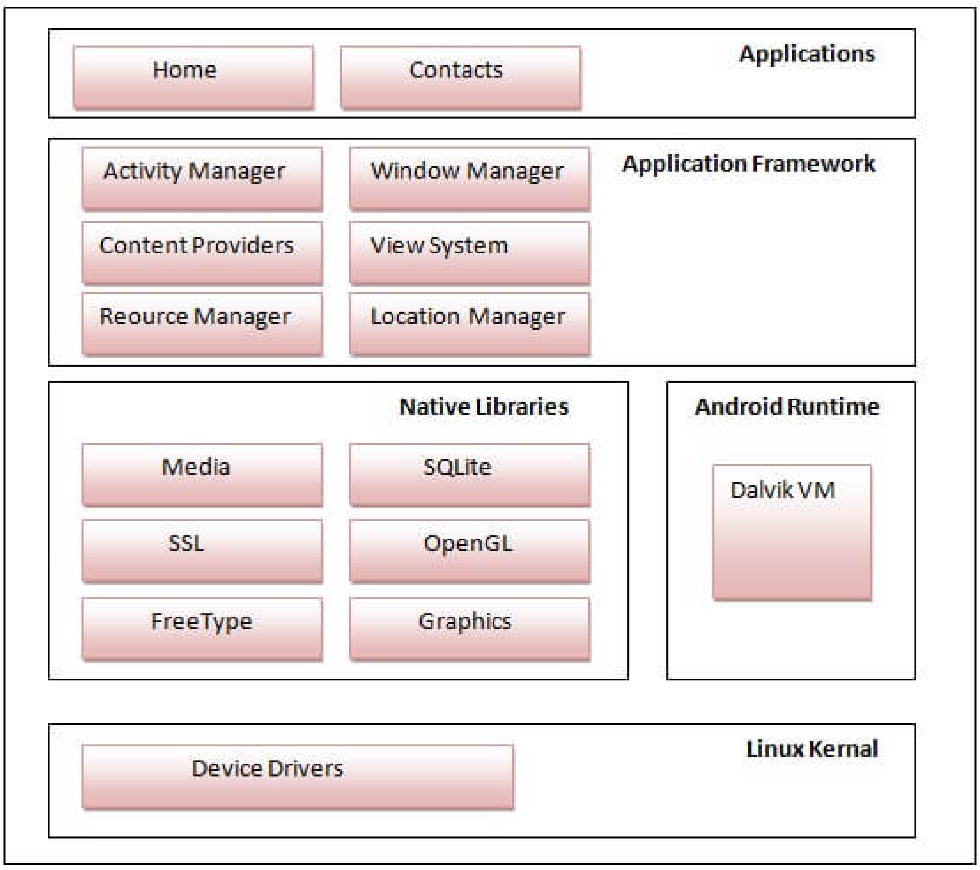
View	ViewGroups
View is a simple rectangle box that responds to the user's actions.	ViewGroup is the invisible container. It holds View and ViewGroup
View is the SuperClass of All component like TextView, EditText, ListView, etc.	ViewGroup is a collection of Views(TextView, EditText, ListView, etc.), somewhat like a container.
View refers to the android.view.View class	ViewGroup refers to the android.view.ViewGroup class
A View object is a component of the UI	It is a layout, that is, a container of other ViewGroup objects
Eg: EditText, Button, CheckBox, etc.	Eg: LinearLayout is the ViewGroup that contains Button(View),etc

Q2) Define Android? Explain android architecture with diagram?

1. Android is a Linux-based operating system and software package for mobile devices, including smartphones and tablet computers. It's developed by Google and the Open Handset Alliance, and the code is written in Java and Kotlin

programming languages. In addition to the operating system, Android includes middleware and various mobile applications.

2. Features of Android:
    a. It is open – source
    b. Anyone can customize the android platform
    c. Inter app integration
    d. Cheaper as compared to ios
3. Architecture of an Android:
    a. Android architecture contains different number of components to support any android device needs



    b. Android architecture or Android software stack is categorized into five parts:
        i. Linux kernel:
            1. It is the heart of android architecture that exists at the root of android architecture.

    2. Linux kernel is responsible for device drivers, power management, memory management, device management and resource access.

ii. Native libraries (Middleware):

    1. On the top of linux kernel, their are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.
    2. The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

iii. Android Runtime:

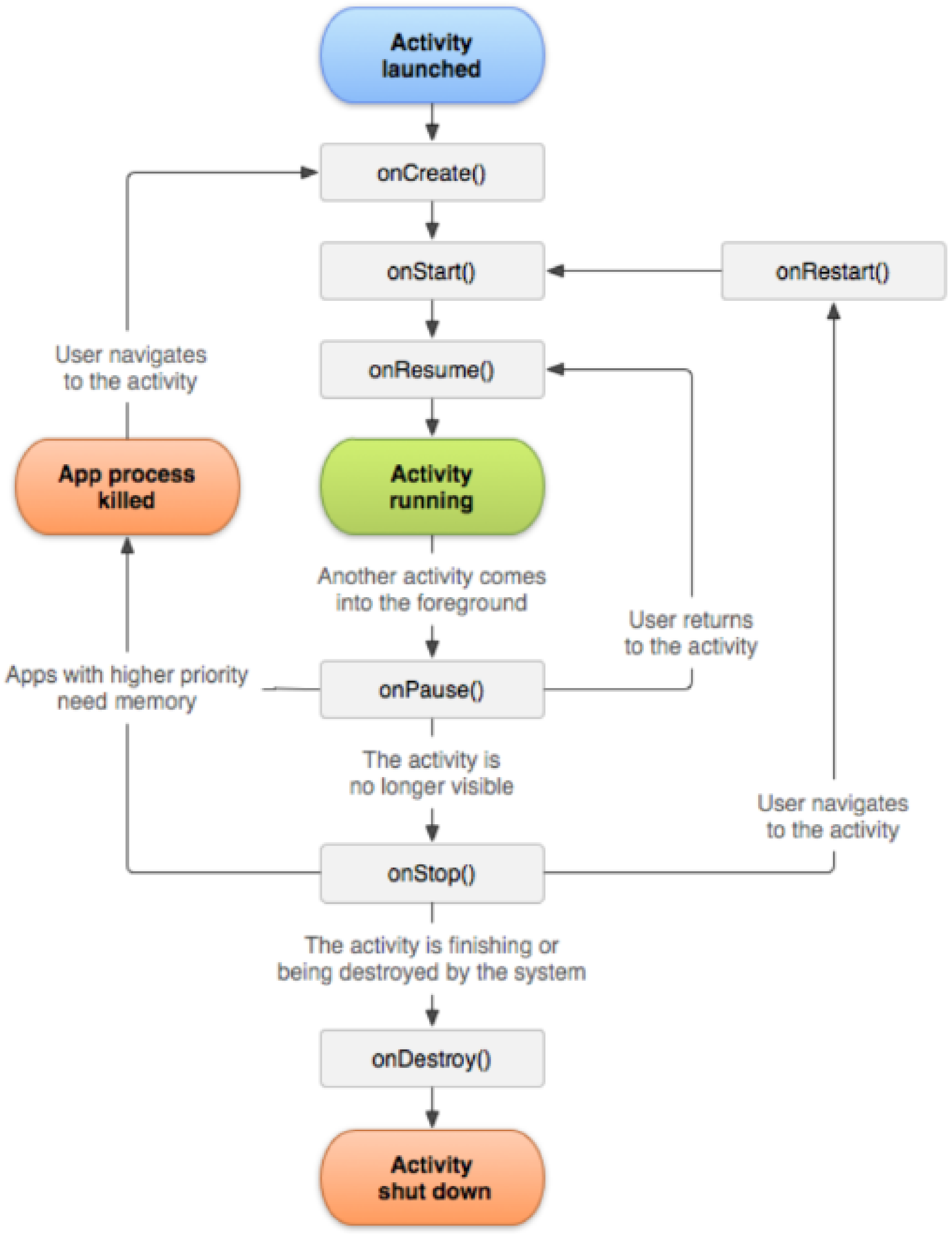
    1. In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application.
    2. DVM is like JVM but it is optimized for mobile devices.

iv. Application Framework:

    1. On the top of Native libraries and android runtime, there is android framework.
    2. Android framework includes Android API's such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers.

v. Applications: All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries.

Q3) Explain Activity life cycle with diagram?


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
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

1. An activity is the entry point for interacting with the user.
2. It represents a single screen with a user interface.
3. Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class.
4. The android Activity is the subclass of ContextThemeWrapper class.

5. The square rectangles represent callback methods which you can implement to perform operations when the Activity moves between states.
6. The ovals are major states the Activity can be in.
7. Seven methods of activity life cycle are:

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.

Q4) Explain radio group with example?

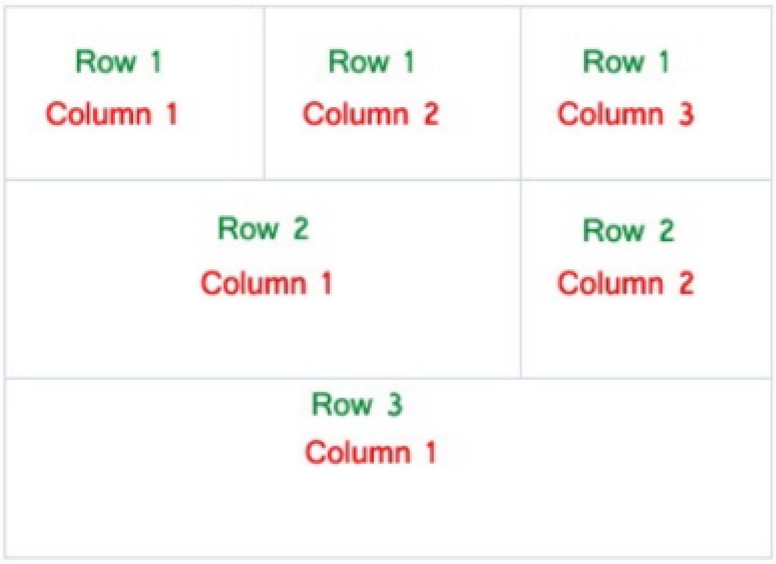
1. A RadioGroup class is used for set of radio buttons.
2. It is a widget in android which is used to handle multiple radio buttons within the android application.
3. We can add multiple radio buttons to our RadioGroup.
4. Eg:

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

<RadioGroup
android:layout_width="match_parent"
android:layout_height="match_parent" >

<RadioButton
android:id="@+id/r2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:checked="true"
android:text="RadioButton" />
```

```
<RadioButton
android:id="@+id/r1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="RadioButton" />

</RadioGroup>
</LinearLayout>
```

Q5) Explain Table layout with example?

Table Layout is a layout manager in Android that allows the grouping of views into rows and columns. It does not display borders for columns, rows, or cells. The number of columns in a table is determined by the row with the most cells, and rows can be built using the <TableRow> element. Table row objects are child views of a table layout, and columns can be stretchable and shrinkable.



Eg:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout 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:layout_gravity="center"
android:stretchColumns="*"
android:collapseColumns="*"
android:shrinkColumns="*"
tools:context=".MainActivity">

<TableRow
android:layout_width="match_parent"
```

```
android:layout_height="match_parent">

<TextView
android:id="@+id/textView"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Student Details" />

</TableRow>

<TableRow
android:layout_width="match_parent"
android:layout_height="match_parent">

<TextView
android:id="@+id/textView7"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />

<TextView
android:id="@+id/textView8"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />

</TableRow>
</TableLayout>
```

Q6) What is intent? Types of intent?

1. Intent is to perform an action.
2. It is mostly used to start activity, send broadcast receiver, start services and send message between two activities.
3. There are two intents available in android :
    a. Implicit Intents :
        i. Implicit Intents are used when we want to perform an action but we don't know which component of the application can handle it. To specify the action, we use the setAction() method in the code. An example of this is when we request another app to show the location details of a user.
        ii. Eg:

```
Intent i = new Intent();
i.setAction(Intent.ACTION_VIEW);
i.setData(Uri.parse("www.google.com"));
startActivity(i);
```

b. Explicit Intents:

   i. By using explicit intents we can send or share data / content from one activity to another activity based on our requirements. To create an Explicit Intent, we need to define the component name for Intent object.

   ii. Eg:

```
Intent send = new Intent(MainActivity.this, SecondActivity.class);
startActivity(send);
```

Q7) What is Service?

1. A service in Android is a component that runs in the background to perform long-running operations or work for remote processes. It serves as a general-purpose entry point for keeping an app running in the background for various reasons. Unlike activities or fragments, a service does not provide a user interface. For instance, a service can play music in the background while the user is interacting with a different app. Other components, such as activities, can start the service and let it run in the background to perform its designated task.

2. Types of Services are:

   a. Foreground services:

      Foreground services are those services that provide ongoing notifications to the user about its ongoing operations. This allows users to interact with the service through notifications provided about the ongoing task. For example, while downloading a file, a user can keep track of the download progress through a notification provided by the foreground service.

   b. Background Services:

      Background services do not require any user intervention. These services do not notify the user about ongoing background tasks and users also cannot access them. Ex. The process like schedule syncing of data

   c. Bound Services:

      Bound services perform their task as long as any application component is bound to it. In order to bind an application component with a service bindService() method is used.

Q8) What is broadcast receiver?

1. A broadcast receiver is an essential component of an Android app that enables the system to deliver events to the app outside of a regular user flow. It allows the app to respond to system-wide broadcast announcements. Unlike other app components, broadcast receivers don't display a user interface. However, they may create a status bar notification to alert the user when a broadcast event occurs. Broadcast events are delivered as an Intent object, which contains information about the event and any data associated with it.
2. There are two types of BR:
    a. Static Broadcast Receivers: These types of Receivers are declared in the manifest file and works even if the app is closed.
    b. Dynamic Broadcast Receivers: These types of receivers work only if the app is active or minimized.
3. Eg:
    a. an app can schedule an alarm to post a notification to tell the user about an upcoming event and by delivering that alarm to a Broadcast Receiver of the app, there is no need for the app to remain running until the alarm goes off.
    b. when incoming calls are received
    c. when a device goes to airplane mode

Q9) What are content providers?

1. A content provider is an essential component of an Android app that allows apps to share data with other apps. It manages a shared set of app data that can be stored in various locations such as the file system, SQLite database, web, or other persistent storage. Other apps can query or modify the data through the content provider if it allows it.

Q10) Explain linear layout?

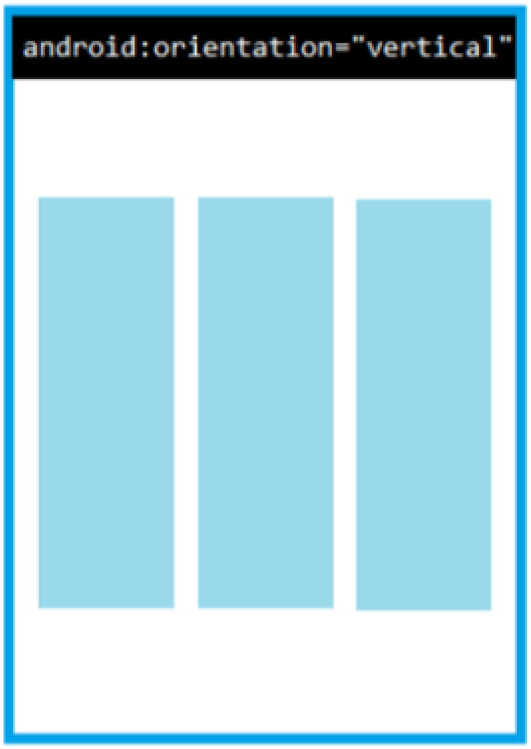
1. A layout that arranges other views either horizontally in a single column or vertically in a single row.
2. Attributes:
    a. Android:orientation:=”vertical/horizontal”:

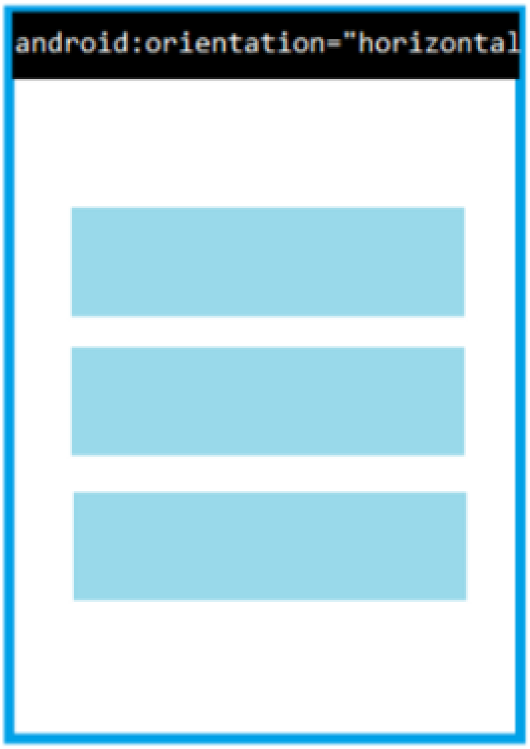
       If one applies android:orientation=”vertical” then elements will be arranged one after another in a vertical manner and If you apply android:orientation=”horizontal” then elements will be arranged one after another in a horizontal manner.

b. Android:gravity: It specifies how an object should position its content on its X and Y axes.

c. Eg:

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

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

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

<Button
android:layout_width="match_parent"
android:layout_margin="10dp"
android:layout_height="wrap_content"/>
</LinearLayout>
```





Q11) Explain Relative layout?

Android RelativeLayout is a powerful tool that allows developers 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. It is a ViewGroup in Android that is used to specify the position of child View instances relative to each other. With the RelativeLayout, you can create complex layouts that adjust to different screen sizes and densities. This makes it an important component for building responsive and adaptive Android applications.

Attributes:

Attribute	Description
layout_alignParentTop	If it specified “true”, the top edge of view will match the top edge of parent.
layout_alignParentBottom	If it specified “true”, the bottom edge of view will match the bottom edge of parent
layout_alignParentLeft	If it specified “true”, the left edge of view will match the left edge of parent.
layout_alignParentRight	If it specified “true”, the right edge of view will match the right edge of parent.

Eg:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="10dp"
android:paddingRight="10dp">

<Button
android:id="@+id/btn1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:text="Button1" />

<Button
android:id="@+id/btn2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentRight="true"
android:layout_centerVertical="true"
android:text="Button2" />

</RelativeLayout>
```

Q12) Explain ScrollView?

A scroll view is a type of view group in Android that enables the view hierarchy placed within it to be scrolled. It allows only one direct child view to be placed within it. However, to add multiple views within the scroll view, you can make the direct child view a view group. Scroll view supports vertical scrolling only, and for horizontal scrolling, HorizontalScrollView can be used instead. With the help of scroll view, users can scroll through a large amount of content that would not fit within the screen's visible area.

Eg:

```
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fillViewport="true">
</ScrollView>
```

Q13) Explain Constraint layout?

Constraint Layout is a ViewGroup that provides a flexible way to create large and complex layouts with a flat view hierarchy. It allows you to position and size widgets in a very flexible way, which helps reduce the nesting of views and improve the performance of layout files. To make a view constrained, you need to make at least two connections of handles with something else. This way, you can create a layout that adapts to different screen sizes and orientations while maintaining a consistent look and feel. With Constraint Layout, you can build more dynamic and responsive user interfaces for your Android app.

Eg:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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="Hello "/>

<TextView
android:id="@+id/textview2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
app:layout_constraintLeft_toRightOf="@id/textview1"
android:text="MITWPU"/>
</android.support.constraint.ConstraintLayout>
```

Q14) Explain Spinner in android?

A Spinner is a type of view in Android that presents items in the form of a dropdown menu that the user can select from. It provides a convenient way to display a list of options and allows the user to choose a single item from the list. When the user taps on the spinner, a dropdown menu appears with a list of options, and the user can select one of the items. The selected item is displayed in the spinner, replacing the previous selection. Overall, a Spinner is a simple and effective way to provide the user with a list of options to choose from.

```
<Spinner
    android:id="@+id/my_spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Q15) Explain android rating bar?

The RatingBar is a user interface control in Android that enables users to provide a rating. It is an extension of SeekBar and ProgressBar, and displays a rating in stars. Users can set the rating value by touching or clicking on the stars. The RatingBar always returns a rating value as a floating point number, such as 1.0, 2.0, 2.5, 3.0, 3.5, and so on. In Android, we can use the RatingBar methods getNumStars() and getRating() to obtain the number of stars and the selected rating value.

Attribute	Description
Android:id	It specifies the unique id
Android:numStars	It defines the number of stars to display
Android:rating	It is used to set the default rating value

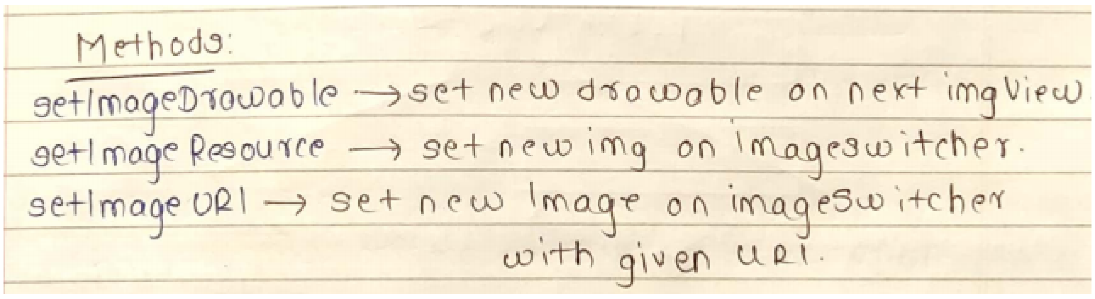
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="example.javatpoint.com.ratingbar.MainActivity">

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="submit"
android:id="@+id/button"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.615" />

<RatingBar
android:id="@+id/ratingBar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="72dp"
android:layout_marginTop="60dp"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

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

```
RatingBar rb;
Button b1;

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

```
b1.setOnClickListener(new View.OnClickListener() {
  @Override public void onClick(View v) {

  int noofstars=rBar.getNumStars();

  float getrating = rBar.getRating();

  tView.setText("Rating: "+getrating+"/"+noofstars);
}
```

Q16) Explain Image Switcher and text switcher?

1. Image Switcher :
    a. An image switcher is a UI component that enables you to add transitions when images are displayed on the screen. To use an image switcher, you need to create an instance of the class and implement the ViewFactory interface using the setFactory() method. This method returns an image view that is used to display the images.



Methods:
setImageDrawable → set new drawable on next imgView
setImageResource → set new img on imageswitcher.
setImageURI → set new Image on imageSwitcher with given URI.

    b. Eg:

```
<ImageSwitcher
android:id="@+id/is1"
android:layout_width="wrap_content"
android:layout_height="wrap_content">

</ImageSwitcher>
```

```
private ImageSwitcher imageSwitcher;
imageSwitcher = (ImageSwitcher)findViewById(R.id.is1);
```

2. Text Switcher:
    a. A TextSwitcher is a transition widget that allows animating a label (i.e. text) on the screen. It is a useful element for adding transitions to the labels.

When the setText(CharSequence) method is called, the TextSwitcher animates the current text out and animates new text in. To create new views, the setFactory() method is used. When settext() is called, the old view is first removed using setOutAnimation() and then the new view is placed using setinAnimation().

b. Eg:

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

    <ImageSwitcher
        android:id="@+id/img"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:layout_marginTop="20dp" />

    <TextSwitcher
        android:id="@+id/ts"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:layout_marginTop="20dp" />

    <LinearLayout
        android:layout_marginTop="20dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/prev"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="90dp"
            android:layout_marginEnd="20dp"
            android:text="Previous" />

        <Button
            android:id="@+id/next"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Next" />

    </LinearLayout>

</LinearLayout>
```

```
ImageSwitcher sw;
TextSwitcher ts;
Button prev, next;
int pos1 = 0;
String[] title = {"MAHI", "MS"};
int[] title2 = {R.drawable.download, R.drawable.images};

ts = (TextSwitcher) findViewById(R.id.ts);
prev = (Button) findViewById(R.id.prev);
next = (Button) findViewById(R.id.next);

ts.setFactory(new ViewSwitcher.ViewFactory() {
    @Override
    public View makeView() {
        TextView t = new TextView(Question2.this);
        t.setTextSize(15);
        t.setGravity(Gravity.CENTER);
        t.setText(title[pos1]);
        return t;
    }
});

sw = (ImageSwitcher) findViewById(R.id.img);
sw.setFactory(new ViewSwitcher.ViewFactory() {
    @Override
    public View makeView() {
        ImageView myView = new ImageView(getApplicationContext());
        myView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        myView.setLayoutParams(new ImageSwitcher.LayoutParams(ViewGroup.Layou
tParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT));
        return myView;
    }
});

prev.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (pos1 <= 0)
            pos1 = title.length - 1;
        else
            pos1--;
        ts.setCurrentText(title[pos1]);
        sw.setImageResource((title2[pos1]));
    }
});

next.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (pos1 == title.length - 1)
            pos1 = 0;
        else
            pos1++;
        ts.setCurrentText(title[pos1]);
        sw.setImageResource((title2[pos1]));
    }
});

ts.setInAnimation(this, android.R.anim.slide_in_left);
ts.setOutAnimation(this, android.R.anim.slide_out_right);

sw.setInAnimation(this, android.R.anim.slide_in_left);
sw.setOutAnimation(this, android.R.anim.slide_out_right);
```