Unit : 2 Android Application Design Essesntials

22 12 22

S.Q. 1. What is the use of onPause() method?

→ OnPause() gets called when your activity is visible but another activity has the focus. onPause() called whe the activity is leaving the foreground and back to the foreground, can be used to release resources or initializ states.

Q.2 What is an R file?

→ Android R.java is an auto-generated file by aapt (Android Asset Packaging Tool) that contains resource IDs for all the resources of res/ directory.

Q.3 What is the use of the AndroidManifest.xml file?

→ The AndroidManifest.xml file provides essential information about your app required for it to run on the Android operating System.

Q.4 List the methods of the android life cycle.

→ An Android activity goes through six major lifecycle stages or callbacks. These are:

1. onCreate()	4. OnPause()
2. OnStart()	5. onStop()
3. OnResume()	6. onDestroy()

22 12 22

Q.5 What is the work of the content Provider and ressource managen?

- Content provider : A content provider manages access to a central repository of data. A provider is part of an Android application, which often provides its own UI for working with the data.

- Ressource manager : The job of a resource manager is, quite simply, to manage all available resources that your company has, especially employees. One of the many responsibilities of a resource manager (more commonly known as a human resource manager, or HR manager) is to assign the right people to a job.

Q.6 What is the difference between onResume() and onRestart() activity?

onResume()	onRestart()
It is called just before the user starts interacting with the application	It is called when the activity in the stopped state is about to start again.

Q.7 What is the use of String.xml file in Android?

A String resource provides text strings for your application with optional text styling and formatting.

There are three types of resources that can provide your application with strings: string XML resource that provides a single string.

Q8 Write the use of intent filter

→ An intent filter declares the capabilities of its ~~parent~~ parent component - What an activity or service can do and what types of broadcasts a receiver can handle. It opens the component to receiving intents of the advertised type, while filtering out those that are not meaningful for the component.

Q9 Drawable folder

→ A drawable resource is a general concept for a graphic that can be drawn to the screen and which you can retrieve with APIs such as getDrawable(int) or apply to another XML resource with attributes such as android: drawable and android: icon. There are several different types of drawables: Bitmap File.

Q10 Use of Spinner

→ Spinners provide a quick way to select one value from a set. In the default state, a spinner shows its currently selected value. Touching the spinner displays a dropdown menu with all other available values, from which the user can select a new one

Q.11 List types of files in android

(1) MP3
(2) Opus
(3) PCM / WAVE
(4) Vorbis
(5) MIDI

- 3GPP (.3gp) , ogg (.ogg)
- MPEG-4 (.mp4, .m4a) , WAVE (.WAV)
- ADTS raw AAC (.aac) , Matroska (.mkv)
- MPEG-TS (.ts)
- ~~3GPP (.3~~ FLAC (.flac) , .AMR (.amr)

Q.12 Significance of /res and /src folders

→ Src is where your source files go.
res is where your resource files go
(icons, images, externalised strings for internationalisation and so forth).
The res/values folder is used to store the values for the resources that are used in many Android projects to include features of color, styles, dimensions etc.

Q.13 what do you mean by activity

→ An activity provides the window in which the app draws its UI. This window typically fills the screen, but may be smaller than the screen and float on top of other windows. Generally, one activity implements one screen in an app.

Q.14 Various dialog boxes with their use in android

(1) AlertDialog (2) ProgressDialog
(3) DatePicker Dialog (4) TimePickerDialog

(1) AlertDialog:
An alert dialog box supports 0 to 3 buttons and a list of selectable elements, including check boxes and radio buttons.

(2) ProgressDialog:
This dialog box displays a progress wheel or a progress bar. It is an extension of AlertDialog and supports adding buttons.

(3) DatePickerDialog:
This dialog box is used for selecting a date by the user.

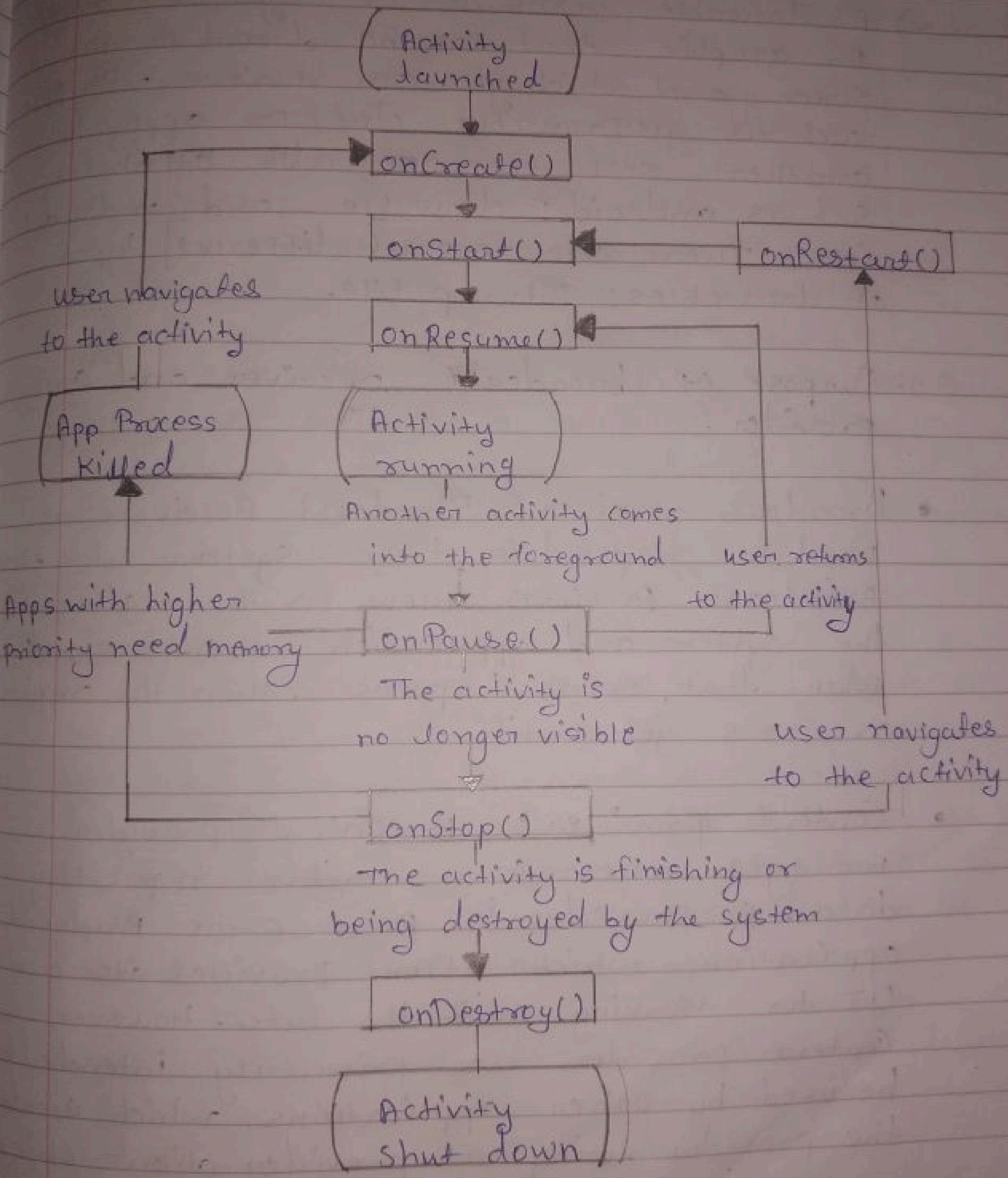
(4) TimePickerDialog:
This dialog box is used for selecting time by user.

~~Q.15 Diagram of Android~~ life Cycle

Q.16 Define 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. A View usually draws something the user can see and interact with. Android layout is used to define the user interface that holds the UI controls or widgets. ~~that~~

S.15 Diagram of Android life cycle


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

24 | 12 | 22

Q.17 use of a drawable folder

→ A drawable resource is a general concept for graphic that can be drawn to the screen and which you can retrieve with APIs such as getDrawable (int) or apply to another XML resource with attributes such as android: drawable and android: icon. There are several different types of drawables: Bitmap File.

Q.18 Purpose of broadcast receiver and content provider

- Broadcas 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.

- Contend provider :- A content provider manages access to a central repository of data. A provider is pard of an Android application, which often provides its own UI for working with the data. However, content providers are primarily intended to be used by other applications, which access the provider using a provider client object.

10

Q.1 Explain about main component of the Android application
-> There are the following main components of an android app:

1. Activities :-
activities are said to be the presentation layer of our applications. The UI of our application is built around one or more extensions of the Activity class. By using Fragments and Views, activities set the layout and display the output and also respond to the user's actions. An activity is implemented as a Subclass of class Activity.

2. Services :-
Services are like invisible workers of our app. These components run at the backend, updating your data Source and Activities, triggering Notifications, and also broadcast Intents. They also perform Some tasks when applications are not active. A service can be used as a subclass of ~~&~~ class Service:

```
public class ServiceName extends Service {
}
```

3. Content Providers :

It is used to manage and persist the application data also typically interacts with the SQL database. They are also responsible for sharing the data beyond the application boundaries. The content providers of a particular application can be configured to allow access from other applications, and the content providers exposed by other applications can also be configured.
A content provider should be a sub-class of the class ContentProvider:

```
public class contentProviderName extends
 ContentProvider {
          public void onCreate () {.}
 }
```

4. Broadcast Receivers:

They are known to be intent listeners as they enable your application to listen to the Intents that ~~st~~ satisfy the matching criteria specified by us. Broadcast Receivers make our application reach to any received Intent thereby making them perfect for creating event-driven applications.

5. Intents :-

It is a powerful inter-application message-passing framework. They are extensively used throughout Android. Intents can be used to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service or Broadcast Receiver or to request action be performed on a particular piece of data.

6. <u>Widgets</u> :
These are the small visual application components that you can find on the home screen of the devices. They are a special Variation of Broadcast Receivers that allow us to create dynamic, interactive application components for users to embed on their Home Screen.

7. <u>Notifications</u> :-
Notifications are the application alerts that are used to draw the user's attention to some particular app event without Stealing focus or interrupting the Current activity of the user. They are generally used to grab user's attention when the application is not visible or active, particularly from within a Service or Broadcast Receiver.
Examples: E-mail popups, Messenger popups, etc.

Q.2 Explain Android Resource Management

→ In Android applications, Java code calls internal Project elements such as XML files, strings, numbers, images, and more. The best way to

24 12 22

Keep all these "Values" available to the application is to place them in the project folder called res and manage them using the appropriate resource machanism. All images and sounds of our video game will be put in an assests folder and loaded into memory using the Android Assest Management machanisms.

Resources are compiled in a binary format and indexed using a Unique ID. These IDs are stored in a Java class, named R, auto-generated at each modification and visible in the folder of the Android project.

The official documentation lists all the resources type that can be used. There is also a resource "raw" type that can be placed in the res/raw folder. You can place there everything that does not fit in other folders.

Q.3 Explain different Android layouts and their attributes.

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

2. Relative Layout: RelativeLayout is a view Group that displays child views in relative Positions.

3. Table Layout :- TableLayout is a view that groups views into rows and columns.

4. Absolute Layout :- Absolutelayout enables you to specify the exact location of its children.

5. Frame Layout :- The FrameLayout is a placeholder on screen that you can use to display a single view.

6. List View :- ListView is view group that displays a list of scrallable items.

7. Grid view :- GridViews is a ViewGroup that displays items in a two-dimensional, scrollable grid.

<u>Layout Attributes:-</u>

1. android: id
   This is the ID which uniquely identifies the view.
2. android: layout_width
   This is the width of the layout
3. android: layout_height
   This is the height of the layout
4. android: layout_margin Top

This is the extra space on the top side of the layout.

5. android: layout_marginBottom
   This is the extra space on the bottom side of the layout

6. android: layout_marginLeft
   This is the extra space on the left side of the layout.

7. android: layout_marginRight
   This is the extra space on the right side of the layout.

8. android: layout_gravity
   This specifies how child Views and positioned

9. android: layout_weight
   This specifies how much of the extra space in the layout should be allocated to the View.

10. android: layout_x
    This specifies the x- coordinate of the layout.

11. android: layout_y
    This specifies the y- coordinate of the layout

12. android: layout_width
    This is the width of the layout.

13. android: paddingLeft
This is the left padding filled for the layout.

14. android: paddingRight.
This is the right padding filled for the layout

15. android: Padding Top
This is the top padding filled for the layout.

16. android: paddingBottom
This is the bottom padding filled for the layout.

Q4 Explain Android Activity life Cycle.

Android provides us with a set of 7 method.

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 on Resume() 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 major 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). The activity, under the active activity, will not be created until the active activity's onPause() returns, so it is recommended that heavy processing should not be done in this part.

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 ~~thin~~ finished, and nothing when the activity remains on the background only. Note that this method may never be called, in low memory situations where the system

does not have enough memory to keep the activity's process running after its onPause() method is called.

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 scenarious, check it with isFinishing() method.

Q5 Explain the Android manifest file and its basic settings in Android.

→ The Android manifest file helps to declare the permissions that an app must have to access data from other apps. The Android manifest file also specifies the app's package name that helps the Android SDK while building the app. The Android manifest file provides information such as activities, services, broadcast receivers, and content providers of an android application.

AndroidManifest.xml file

- Supported screen sizes
- Supported SDK versions : minimum, target, and maximum
- Ability to send Push Notifications
- Various permissions for the application.

1. From the project Explorer, click project Settings. The Project Settings window appears.

2. click the Native tab.
3. click the Android sub-tab, and then scroll down to the manifest Properties & Gradle Entries section
4. Configure the Permissions, Tags, and Deeplink URL scheme tabs.

following Permissions are set to true and added by default:
- ACCESS_NETWORK_STATE
- INTERNET
- READ_PHONE_STATE

Q.6 Android Intent. What is the use of it? Explain by giving an example / How do activities.

→ Android Intent is the message that is passed between componets such as activities, content Providers, broadcast receivers, services etc.

- It is generally used with startActivity() method to invoke activity, broadcast receivers etc. The dictionary meaning of intent is intention or purpose. so, it can be described as the intention to do action.

21 12 22

- The ~~Label~~ ~~e~~ Intent is the subclass of android.content.Intent class.

Android intents are mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

- Intents are used to signal to the Android System that a certain event has occurred ~~Inter~~ Intents often describe the action which ~~we~~ ~~was~~ should be performed and provide data upon which such as action should be done. For example, your application can start a browser component for a certain URL via a intent.

Q.7 Communicate with each other? example giving an example

→ At the simplest level, there are two different ways for apps to interact on Android: via intents, passing data from one application to another; and through services, where one application provides functionality for others to use.

- If your devices are very close to one another (up to about 10 meters), you can communicate using Bluetooth, ~~as~~ D~~[illegible]~~

24 12 22

- If your devices are somewhat further away but within WiFi range of each other (up to about 100 meters), then they can communicate with each other using the Peer-to Peer WiFi API. This does not require a WiFi router to be present, and the devices will find each other and communicate directly.

- The Android Wireless API will also work if your devices are on the same local network. even if they are not themselves within range of each other.

- If none of of these options are viable / guaranteed then the easiest way would be to use ServerSocket and Socket to create a server /client interface through the Internet.

Q.8 Broadcast Receiver

→ Broadcast in android is the system-wide events that can occur when the device starts, when a message is received on the device or when incoming calls are received, or when a device goes to airplane mode, etc. Broadcast Receivers are used to respond to these system-wide event. 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 mainly two types of Broadcast Receivers:

- Static Broadcast Receivers : These types of Receivers are declared in manifest file and works even if the app is closed.

- Dynamic Broadcast Receivers : These types of receivers work only if the app is active or minimized.

Q.9 Various files and folders in android

We will explore all the folders and files in the android app.

1. Manifests Folder
2. Java Folder
3. res (Reasources) Folders
   - Drawable Folder
   - Layout Folder
   - Minimap Folder
   - Values Folder
4. Gradle Scripts

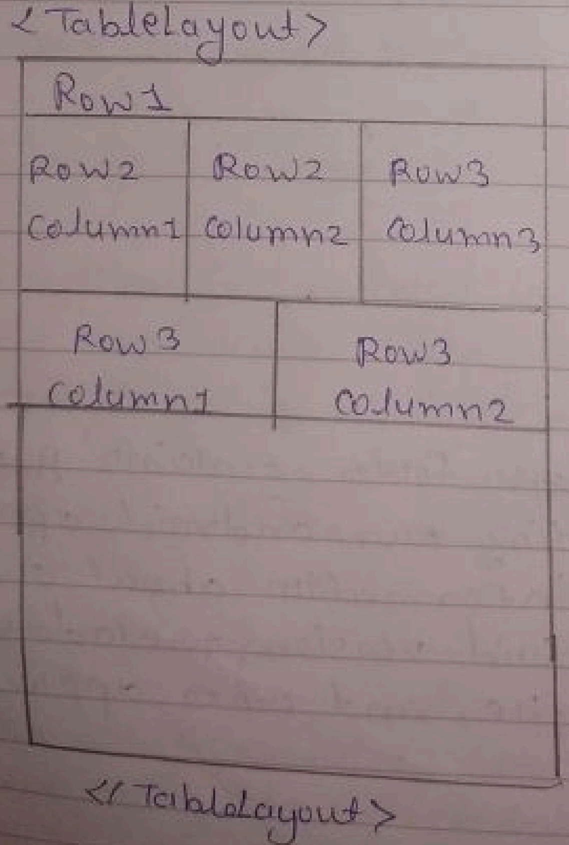
Manifests Folder :-

Manifests folder contains AndroidManifest.xml for creating our android application. This file contains information about our application such as the Android version, metadata, states package for kotlin file, and other application components.

- MainActivity.kt file ⟶ Java Folder
- activity_main.xml file ⟶ res/layout Folder
- launcher.xml file -> res/mipmap Folder
- strings.xml file ⟶ res/values Folder

Q.10 Table layout with example

→ Android TabelLayout going to be arranged group of views into rows and columns. You will use the <TabelRow> element to build a row in the table. Each row has zero or more calls, each cell can hold one view object. Tablelayout containers do not display border lines for their rows, columns, or cells



Example

src/com.example.demo/MainActivity.java

```
Package com.example.demo;

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

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

res/layout/activity_main.xml file

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

 <TableRow
    android:layout_width="fill_parent"
    android:layout height="fill_parent">

  <TextView
     android:text="Time"
     android:layout_width ="wrap_content"
```

24.12.22

```
e
rc/com.example.demo/MainActivity.java

ckage com.example.demo;

nport android.os.Bundle;
nport android.app.Activity;
nport android.view.menu;

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

res/layout/activity_main.xml file

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

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

 <TextView
```

Example

21.12.22

src/com.example.demo/MainActivity.java

```
package com.example.demo;

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

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

res/layout/activity_main.xml file

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

 <TableRow
    android:layout_width="fill_parent"
    android:layout height="fill_parent">

  <TextView
     android:text="Time"
     android:layout_width ="wrap content"
```

```
android:layout_height="wrap_content"
android:layout_column="1" />

<TextClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textClock"
    android:layout_column="2" />

</TableRow>
<TableRow>

 <Textview
    android:text="First Name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="1" />

<EditText
    android:width="200px"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

</TableRow>

<TableRow>

<Textview
    android:text="Last Name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content'
```

24 12 22

```
android: layout_ column ="1" />

<Edit Text
    android: layout _width = "fill_parent"
    android: layout_height = "fill_ parent" >

 < RatingBar
    android: layout_ width ="wrap_ content"
    android: layout_height = "wrap_content"
    android: id =" @ + id /ratingBar"
    android: layout_column ="2" />

</Table Row>
<Table Row
    android: layout_width = "fill_parent"
    android: layout_height ="fill_ parent" />

< Table Row
    android layout_width ="fill_parent"
    android layout_height= "fill_parent">

 < Button
    android: layout_width ="wrap_content"
    android: layout_height= "wrap_content"
    android : text = "submit"
    android: id ="@ + id / button"
    android: layout _column = "2" />
 </ TableRow>
</ TableLayout>
```

```
<?xml version ="1.0" encoding ="Utf-8"?>
<resources>
    <String name ="app_name"> HelloWorld </String>
  <String name=" action_settings"> Settings </string>
</resources>
```

Q.11 Discuss Android Screen Orientation

Screen Orientation, also known as screen rotation, is the attribute of activity element in android. When screen orientation change from one state to other, it is also known as configuration change. There are various possible screen orientation states for any android application, such as: ActivityInfo.

The screenOrientation is the attribute of activity element. The orientation of android activity can be portrait, landscape, sensor, Unspecified etc. You need to define it in the AndroidManifest xml file.

Syntax:

```
<activity android= "package_name. Your_ActivityName"
  android : screenOrientation =" orientation_type">
</activity>
```

Example :

```
<activity android:name="example.javatpoint.com.
  screenorientation.MainActivity" android: screenOrient
ation=" portrait">
</activity>

<activity android: name=".SecondActivity"
   android:screenOrientation= "landscape">
</activity>
```

ScreenOrientation attribute are as follows:

- Unspecified : It is the default value. In such case, system choose the Orientation.

- Portrait:- taller not wider

- landscape:- wide not taller

- sensor:- Orientation is determined by the device orientation Sensor.