Model Question Paper-2

Mobile Application Development

2Marks

1) Define Dalvik Virtual machine?

A) The Dalvik Virtual Machine (DVM) is a process virtual machine in the Android operating system that executes applications written for Android. It is specifically designed for mobile devices and optimized to ensure that Android applications run efficiently in environments with limited memory and processing power.

2) What is menu?

A) In Android, a menu is a user interface component that provides a list of options or actions for users to interact with. Menus help organize and present functionality in a structured manner within an Android application.

There are several types of menus in Android:

Options Menu: The primary menu for an activity, displayed when the user presses the Menu button (for older devices) or taps the overflow button (three vertical dots) in the action bar.

Context Menu: A floating menu that appears when the user performs a long-click on an element.

3) List the type of mobile technology?

1. Cellular Technology
2. Wifi
3. SMS & MMS
4. Bluetooth
5. SMS
6. MMS
7. 4G (fourth Generation)
8. 5G (fifth generation)
9. GSM
10. CDMA
11. Wi-Fi

4) What are the two attributes of Imageview?

A) **android:src:** This attribute sets the image to be displayed in the ImageView.
android:src="@drawable/example_image"
android:contentDescription: This attribute provides a textual description of the image, which is useful for accessibility purposes (e.g., for screen readers).
android:contentDescription="@string/image_description"

5) What is GPS?

A) **Global Positioning System (GPS)** is a satellite-based navigation system that provides location and time information to a GPS receiver anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The role of GPS in various applications, including Android devices, is crucial for providing accurate and reliable location-based services.

6) List the steps to create database cursors.

A) 1. Initialize Database Helper Class
2. Get a Reference to the Database
3. Execute Query Using query () Method
4. Iterate Through the Cursor
5. Close the Cursor
6. Close the Database Connection

4Marks

7) What are the types of basic views?

Some of the basic views that can be used to design the UI components of the android applications.

1. TextView
2. EditText
3. Button
4. ImageButton
5. CheckBox
6. ToggleButton
7. RadioButton
8. RadioGroup

TextView:

This is a view that displays text. It can be used to show a single line or multi-line text. It's one of the most basic and frequently used views in Android. Android TextView Attributes:

Code:

```
<TextView
   android:id="@+id/textView"
   android:layout_width="fill_content"
   android:layout_height="wrap_content"
   android: text="hello BCA students"/>
```

EditText:

An EditText is a subclass of the TextView that is configured to allow the user to edit the text inside it.

<Edittext

android:id =”@+id/myEdittext”
android:layout_width=”fill_parent”
android:layout_height=”fill_parent”
android:hint=”Enter a Number”
android:singleLine=”true”
android: inputType=”textPassword”/>

Button:

- In android, Button is a user interface control that is used to perform an action whenever the user clicks or tap on it.
- Generally, Buttons in android will contain a text or an icon or both and perform an action when the user touches it.

Code:

<Button

android:id="@+id/button"
android:layout_width="fill_content"
android:layout_height="wrap_content"
android:text="Click Here!"/>

ImageButton:

- In android, Image Button is a user interface control that is used to display a button with an image and to perform an action when a user clicks or taps on it.
- In android, we can add an image to the button by using attribute android: src in XML layout file or by using the setImageResource () method.

Code:

<ImageButton
android:id="@+id/addBtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/add_icon" />

Check Box:

A CheckBox in Android is a type of button that has two states: checked and unchecked. By default, the android checkbox will be in the OFF (unchecked) state.we can change the default state of checkbox by using android: checked attribute.

Code:

```
<CheckBox
android:id="@+id/simpleCheckBox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Simple CheckBox"/>
android: checked=”false”
```

ToggleButton:

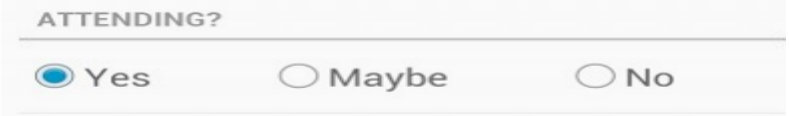
- ToggleButton allow the users to change the setting between two states like turn on/off your wifi, Bluetooth etc from your phone’s setting menu.

Code:

```
<ToggleButton
android:id="@+id/toggle1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:checked="true"
android:textOff="OFF"
android:textOn="ON"/>
```

Radio Button:

- In android, Radio Button is a two-state button that can be either checked or unchecked and it’s the same as CheckBox control, except that it will allow only one option to select from the group of options.



Code:

```
<RadioButton
android:text="Java"
android:checked="true"/>
```

Radio Group:

In android, we use radio buttons with in a RadioGroup to combine multiple radio buttons into one group and it will make sure that users can select only one option from the group of multiple options.

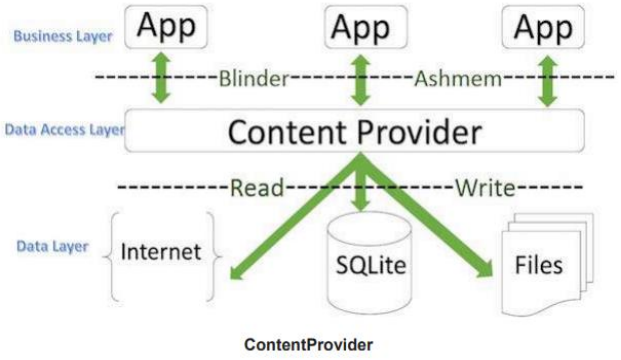
```
<RadioGroup
   android:id="@+id/radioGroup"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="vertical">

   <RadioButton
      android:id="@+id/radioButton1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Option 1" />

   <RadioButton
      android:id="@+id/radioButton2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Option 2" />
</RadioGroup>
```

8) **Explain content provider?**

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentReslover class. It's a way to share data between different applications securely. A content provider can use different ways to store its data and data can be stored in a database, in files, or even over a network.



Here's a breakdown of the storage options:

Database: The content provider can store its data in a database, typically a SQLite database. This is a common method because databases are efficient for handling structured data and complex queries.

Files: Data can also be stored in files. This method might be used for handling unstructured data, such as media files (images, videos) or large datasets.

Network: A content provider can also store data over a network. This could mean the data is fetched from a remote server or cloud storage.

Android ships with many useful content providers, including the following:

➢ Browser—Stores data such as browser bookmarks, browser history, and so on

➢ CallLog—Stores data such as missed calls, call details, and so on

➢ Contacts—Stores contact details

Content URI is the key concept of Content Providers. To access the data from a content provider, Content URI is used as a query string.

Structure of a Content URI is **Content: //authority /optionalPath /optionalID**

Content: It represents that the given URI is content URI.

Authority: Name of the Content provider like contacts, browser, and etc.It must be unique for every content provider.

OptionalPath: Specifies the type of data provided by the Content provider. For example, if you are getting all the contacts from the **Contacts** content provider, then the data path would be people and URI would look like this **content://contacts/people**

OptionalID: It is a numeric Value it specifies the specific record requested. For example, example, if you are looking for a contact number 5 in the Contacts content provider then URI would look like this **content: //contacts/people/5.**

Operations in Content provider: The fundamental operations are possible in content Provider namely Create, Read, update, get type, and delete. These operations are often termed as CRUD operations.

1. **OnCreate ( ):** This method is called when the provider is started.
2. **Query ( ):** This method receives a request from a client. The result is returned as a Cursor object.
3. **insert( ):** This method inserts a new record into the content provider.
4. **delete( ):** This method deletes an existing record from the content provider.
5. **update( ):** This method updates an existing record from the content provider.
6. **getType( ):** This method returns the MIME type of the data at the given URI.

9. Explain fragments and intents?

A) In Android, you navigate between activities through what is known as intent.

- Intent is a messaging object used to request any action from another app component. Intent's most common use is to launch a new activity from the current activity.
- Intent facilitates communication between different components. Intent object is used to call other activities.

The intent is used to launch an activity, start the services, broadcast receivers, display a web page, dial a phone call, send messages from one activity to another activity, and so on.

TYPES OF INTENTS:

EXPLICIT INTENT:

- Explicit Intent is used to invoke a specific target component. It is used to switch from one activity to another activity in the same application. It is also used to pass data by invoking the external class. Explicit Intents specify the target component by providing the exact class name of the component to be invoked.

IMPLICIT INTENT:

- An implicit intent is used when you want to perform an action, but you don't care which component performs it. The system will determine the appropriate component to handle the intent based on the available components that can respond to it.

Android Fragments:

Android Fragment is a Graphical User Interface component of Android. It resides within the Activities of an Android application. It represents a portion of UI that the user sees on the screen.

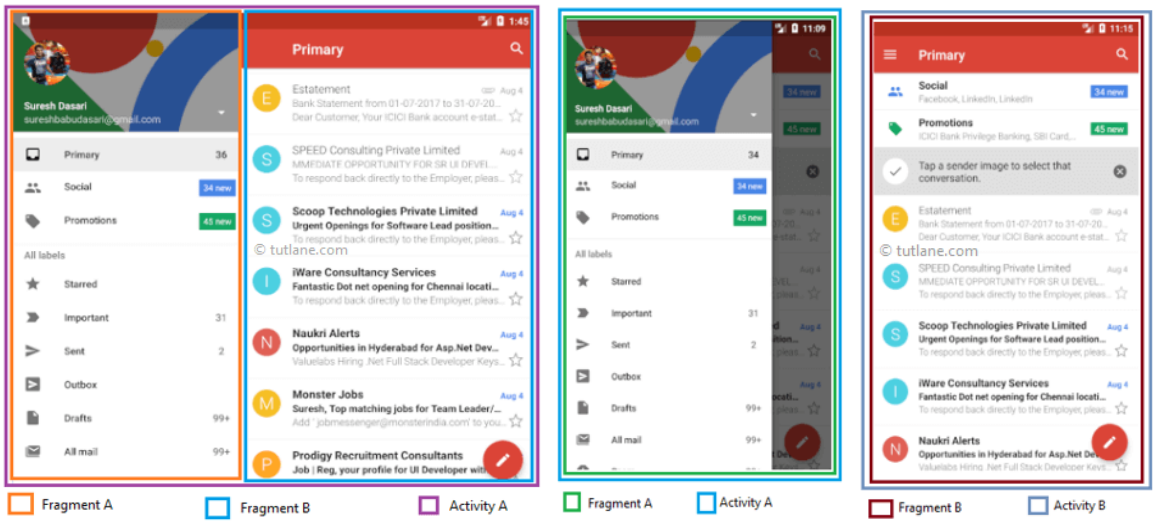
- Android Fragments cannot exist outside an activity. Another name for Fragment can be Sub-Activity as they are part of Activities.

How Fragment Interacts with Activity in Different Devices:

- If you observe above example for **Tablet** we defined an **Activity A** with two fragments such as one is to show the list of items and second one is to show the details of item which we selected in first fragment.
- For **Handset** device, there is no enough space to show both the fragments in single activity, so the **Activity A** includes first fragment to show the list of items and

the **Activity B** which includes another fragment to display the details of an item which is selected in **Activity A**.

- For example, **GMAIL app** is designed with multiple fragments, so the design of GMAIL app will be varied based on the size of device such as tablet or mobile device.



Tablet View **Mobile View**

10. Explain the life cycle of an android activity?

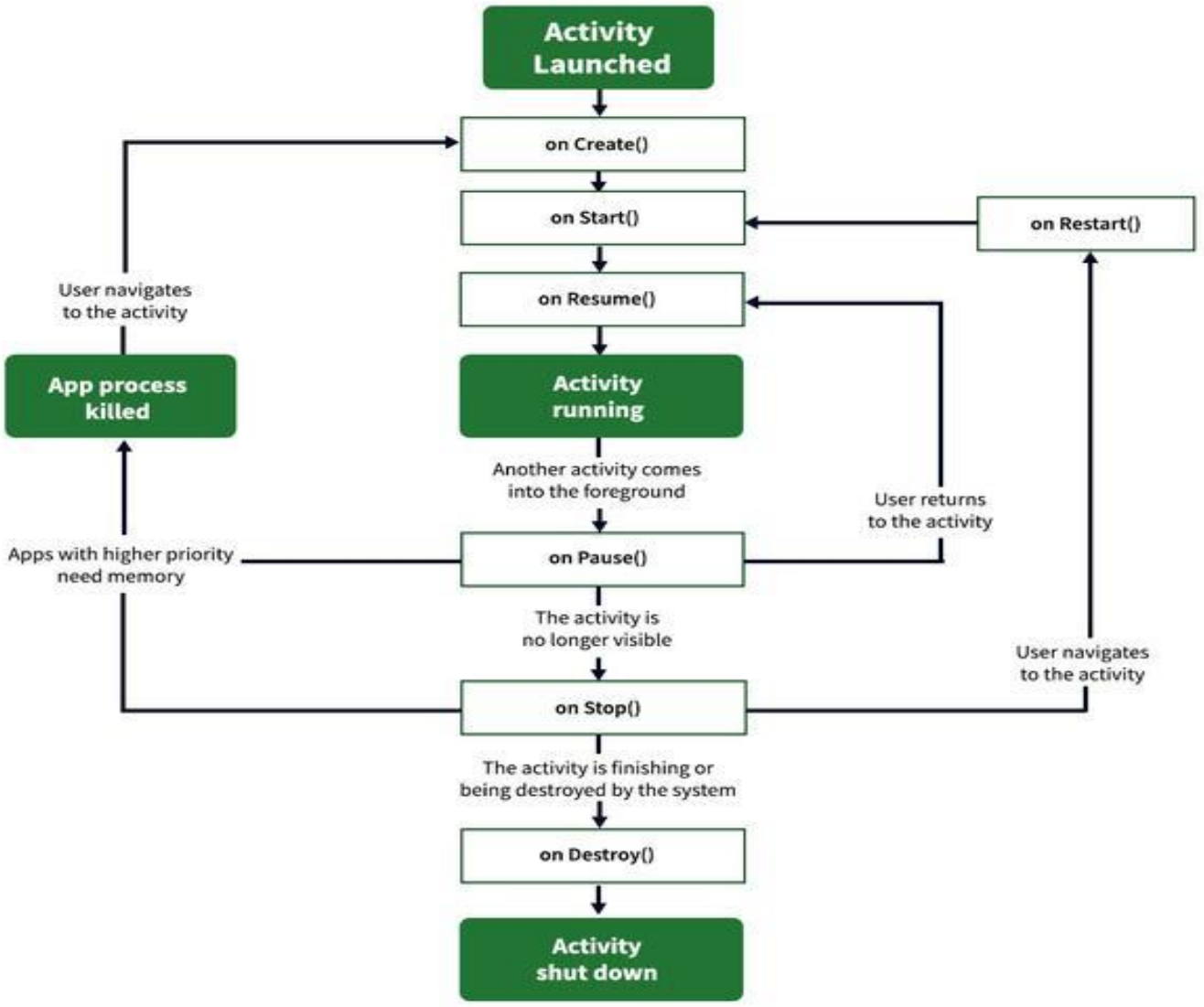
Every activity has different functions throughout its life, onCreate (), onStart (), onResume (), onPause (), onStop (), onRestart (), onDestroy ().

- **The Activity class defines the following Methods:**

- **OnCreate()** — Called when the activity is first created
- **OnStart()** — Called when the activity becomes visible to the user
- **OnResume()** — Called when the activity starts interacting with the user **or** activity is becoming visible to the user and is ready to start receiving user interactions.
- **OnPause()** — Called when the current activity is being paused and the previous activity is being resumed. This is where you typically pause ongoing processes or save transient data.
- **OnStop()** — Called when the activity is no longer visible to the user.
- **OnDestroy()** — Called before the activity is destroyed by the system (either manually or by the system to conserve memory)

- **OnRestart()** — Called when the activity has been stopped and is restarting again.

➢ By default, the activity created for you contains the onCreate() event. Within this event handler is the code that helps to display the UI elements of your screen.
Figure 3-2 shows the life cycle of an activity and the various stages it goes through—from when the activity is started until it ends.



Activity Lifecycle in Android

11. Explain to apply a style as theme?

Apply the Style as a Theme

Applying as Application Theme

To apply AppTheme as the theme for your entire application, modify the AndroidManifest.xml

```
<application
   android:theme="@style/AppTheme"
   ... >
   ...
</application>
```

Applying as Activity Theme

To apply AppTheme as the theme for a specific activity, modify the <activity> tag in AndroidManifest.xml:

```
<activity
   android:name=".MainActivity"
   android:theme="@style/AppTheme"
   ... >
   ...
</activity>
```

Application Theme: Set in the <application> tag of AndroidManifest.xml, applies @style/AppTheme to the entire application.
Activity Theme: Set in the <activity> tag of AndroidManifest.xml, applies @style/AppTheme specifically to the named activity (MainActivity in this case).

12. Explain menus and additional view.

- **Menus** in Android refer to user interface components that provide options for users to interact with the app's functionalities. There are primarily two types of menus:
- **Menus** in Android provide options for user interaction through options menus and contextual action modes.

a. Options Menu

- **Options Menu**: This is a standard menu that appears when the user presses the menu button on their device or when they touch the menu icon in the app bar (if using a toolbar). It typically contains actions related to the current context or screen.

b. Contextual Action Mode

- **Contextual Action Mode**: This is used to provide actions for selected items in a list or grid. It appears when the user performs a long-click on a selectable item (e.g., long-pressing an item in a RecyclerView).

Additional Views

- **Additional Views** refer to UI components beyond basic widgets like buttons and text fields. These views provide specialized functionalities and often enhance user interaction and experience. **Additional Views** such as RecyclerView and ViewPager enhance user interfaces by efficiently displaying large datasets or supporting swipe-based navigation.

a. RecyclerView

- **RecyclerView**: A flexible view for providing a limited window into a large dataset of items. It efficiently handles large lists or grids by recycling item views and supports various layout managers for arranging items.

. ViewPager

- **ViewPager**: Allows users to swipe between different fragments or pages. It's commonly used for creating swipeable tabs, image galleries, or onboarding screens.

8Marks

13.Explain Date picker, Time picker, and List view.

Using picker Views:

- Picker views in Android are specialized UI components that allow users to select a value from a predefined set of values. Selecting a date and time is one of the common tasks you need to perform in a mobile application. Android supports this functionality through the TimePicker and DatePicker views.
  1. Date picker View
  2. Time picker View

Time Picker View:

- In Android, TimePicker is a widget used for selecting the time of the day in either AM/PM mode or 24 hours mode. The displayed time consist of hours, minutes and clock format.
- Generally in android TimePicker available in two modes
  1. Show the time in clock mode
  2. Show the time in Spinner mode

TimePicker with clock mode:

- We can define the Timepicker to show time in clock format by using
- android: timePickerMode attribute.
- android:timePickerMode="clock": This attribute sets the mode of the TimePicker to display a clock-style time picker.
- android:format24Hour="HH:mm": This sets the format of the time displayed. In this case, it's in 24-hour format (e.g., 13:45).

```
<TimePicker
android:id="@+id/simpleTimePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:timePickerMode="Clock"/>
```

TimePicker with Spinner mode:

- The TimePicker displays a standard UI to enable users to set a time. By default, it displays the time in the AM/PM format. If you want to display the time in the 24-hour format, you can use¬ the setIs24HourView () method.
- We can define the Timepicker to show time in Spinner format by using android: timePickerMode attribute

Code:

```
<TimePicker
```

```
android:id="@+id/simpleTimePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:timePickerMode="spinner"/>
```

Date picker:

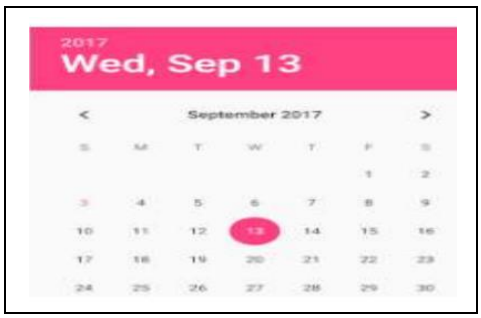
- In android, DatePicker is a control that will allow users to select the date by a day, month and year in our application user interface.
- If we use DatePicker in our application, it will ensure that the users will select a valid date. Following is the pictorial representation of using a datepicker control in android applications.
- Generally, in android DatePicker available in two modes,
  1. Show the complete calendar
  2. Show the dates in spinner view.

Date Picker in Calendar format:

- We can define the Datepicker to show Date in Calender format by using android: DatePickerMode attribute.

Code:

```
<DatePicker
android:id="@+id/simpleDatePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:datePickerMode="Calender"
```



Date Picker in Spinner format:

- If we want to show the DatePicker in spinner format like showing day, month and year separately to select the date, then by using DatePicker **android:datePickerMode** attribute we can achieve this.
- Following is the example of showing the DatePicker in Spinner mode.

Code:

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

- The above code will return the DatePicker like as shown below

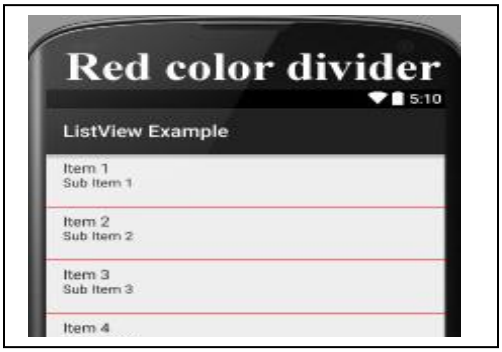
android:id - It is used to uniquely identify the control .
android:datePickerMode - It is used to specify datepicker mode either spinner or calendar.
android:background - It is used to set the background color for the date picker.
android:padding - It is used to set the padding for left, right, top or bottom of the date picker.

ListView :

- Android ListView is a view which Contains several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database. ListView is implemented by importing android.widget.ListView class.
- In Android, there are two types of list views: **ListView and SpinnerView.**
- A very common example of ListView is your phone contact book, where you¬ have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.

Code:

```
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
/>
```



SpinnerView:

In Android development a spinner is a view, A spinner typically appears as a small rectangular box with an arrow icon at the right side. When the user taps on the spinner, a dropdown list of items appears below it, allowing the user to select one.

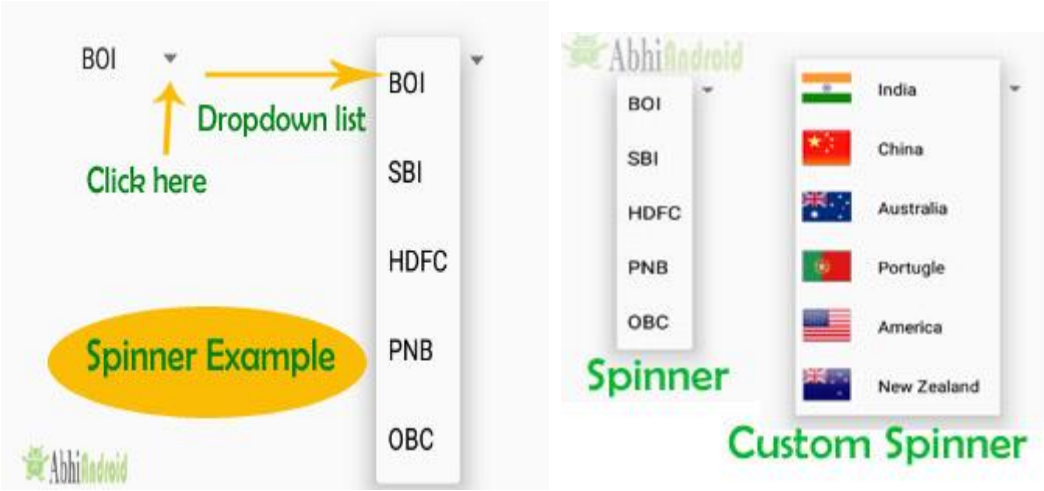
Code:

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

- To fill the data in a spinner we need to implement an adapter class. A spinner is mainly used to display only text field so we can implement Array Adapter

for that. We can also use Base Adapter and other custom adapters to display a spinner with more customize list.

- Suppose if we need to display a textview and a imageview in spinner item list then array adapter is not enough for that. Here we have to implement custom adapter in our class. Below image of Spinner and Custom Spinner will make it more clear.



14. What are Android application components?

Android - Application Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application –

Sl.No	Components & Description
1	**Activities** They dictate the UI and handle the user interaction to the smart phone screen.
2	**Services** They handle background processing associated with an application.
3	**Broadcast Receivers** They handle communication between Android OS and applications.
4	**Content Providers** They handle data and database management issues.

1. **Activities**

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of Activity class as follows –

```
public class MainActivity extends Activity {
}
```

2. Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of Service class as follows –

```
public class MyService extends Service {
}
```

3. Broadcast Receivers

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

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcaster as an Intent object.

```
public class MyReceiver extends BroadcastReceiver {
        public void onReceive (context, intent){}
}
```

4. Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider {
        public void onCreate(){}
}
```

5. Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are –

S.No	Components & Description
1	**Fragments** Represents a portion of user interface in an Activity.
2	**Views** UI elements that are drawn on-screen including buttons, lists forms etc.
3	**Layouts** View hierarchies that control screen format and appearance of the views.
4	**Intents** Messages wiring components together.
5	**Resources** External elements, such as strings, constants and drawable pictures.
6	**Manifest** Configuration file for the application.

15) Explain the creation of Login window.

Create an application to develop Login window using UI controls.

activity_main.xml code:

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">
```

```
    <TextView
        android:id="@+id/textView6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Login"
        android:gravity="center"/>

    <EditText
        android:id="@+id/editTextText8"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="text"
        android:text="Name" />

    <EditText
        android:id="@+id/editTextText9"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="text"
        android:text="password" />

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="login" />

</LinearLayout>
```

MainActivity.Java code:

```
package com.example.jithin33lab;
import android.os.Bundle;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    private EditText a,b;
    private Button c;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        a= findViewById(R.id.editTextText8);
        b= findViewById(R.id.editTextText9);
        c = findViewById(R.id.button3);
        c.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = a.getText().toString().trim();
                String password = b.getText().toString().trim();
                if(username.equals("admin") && password.equals("pass")){
                    Toast.makeText(MainActivity.this, "Login successful",
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(MainActivity.this, "Invalid username or password",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

16. Explain Android studio components?

Project Window:
- Displays your project structure, files, and resources.
- Allows navigation and management of project files.

Editor Window:
- Where you write and edit code, XML layouts, and resources.
- Provides features like syntax highlighting and code completion.

Toolbar:
- Contains buttons for common tasks like running your app, debugging, and accessing SDK tools.

Navigation Bar:
- Helps navigate between files and within files (methods, classes) in the editor.

Component Tree:
- Shows the hierarchy of UI components for editing layouts.
- Helps visualize and navigate the structure of your app's UI.

Build Variants:
- Manages different versions of your app (e.g., debug, release) and flavors (e.g., free, paid).

Gradle Console:
- Displays messages related to the build process handled by Gradle.

Logcat:
- Shows log messages generated by your app during runtime.
- Helps debug and monitor your app's behavior.

Device File Explorer:
- Allows viewing and managing files on your connected Android device or emulator.

AVD Manager:
- Creates and manages Android Virtual Devices (AVDs) for testing your app on different configurations.

Profiler:
- Provides real-time insights into your app's CPU, memory, and network usage.
- Helps optimize app performance and diagnose issues.

Layout Editor:
- Visual editor for designing XML layouts.
- Allows drag-and-drop of UI components and previewing layouts.

Data Binding Viewer:
- Helps visualize and debug data binding layouts in Android projects.

Firebase Integration:
- Integrates Firebase services (analytics, authentication, etc.) directly into your app development process.

17) What is the use of APK file?

An APK (Android Package Kit) file is the package file format used by the Android operating system for distributing and installing mobile apps. It contains all the necessary files for an Android application to be installed on an Android device. Here are some key uses of APK files:

1. **Installation**: APK files are used to install applications on Android devices. Users can download APK files from various sources, including app stores like Google Play or directly from developers' websites.
2. **Distribution**: Developers use APK files to distribute their Android apps. They compile their application code, resources, and manifest file into an APK, which users can then download and install on their devices.
3. **Testing**: During the development process, developers often share APK files with testers or stakeholders for testing purposes. This allows them to get feedback and identify issues before releasing the app to a wider audience.
4. **Offline Installation**: APK files enable users to install apps without needing an internet connection. This is particularly useful in regions with limited internet access or for devices that do not have access to the Google Play Store.
5. **Backup**: Users can save APK files as backups of their installed apps. This can be useful if they need to reinstall an app without downloading it again from the internet.

Overall, APK files are essential for the distribution, installation, and maintenance of Android applications across various devices.

18. Create an application to send SMS and receive SMS.

```
<?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">

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter number"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter message"
        android:inputType="textPersonName" />
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="SEND" />
</LinearLayout>
```

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {
    EditText phonenumber,message;
    Button send;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        send=findViewById(R.id.button);
        phonenumber=findViewById(R.id.editText);
        message=findViewById(R.id.editText2);
        send.setOnClickListener(new View.OnClickListener() {

            public void onClick(View view) {
                String number=phonenumber.getText().toString();
                String msg=message.getText().toString();
                try {
                    SmsManager smsManager=SmsManager.getDefault();
                    smsManager.sendTextMessage (number,null,msg,null,null);
                    Toast.makeText(getApplicationContext(),"Message
Sent",Toast.LENGTH_LONG).show();
                }catch (Exception e)
                {
                    Toast.makeText(getApplicationContext(),"Some fields is
Empty",Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
```