Mobile Computing Lab
Module 1, 2: Introduction to Android and its components, Basic Controls and UI Components

Assignment 1:

1. Write a program to demonstrate the activity lifecycle.

Aim: To demonstrate the activity lifecycle in an Android application using Java

Objective:

- To understand the different states of an Android activity.
- To observe how the activity lifecycle methods are invoked during the activity's lifetime.

Theory:

In Android, an activity is a single screen with a user interface. The lifecycle of an activity represents the different states it goes through, from its creation to its destruction. Understanding these states is crucial for managing resources, saving data, and ensuring a smooth user experience.

The key lifecycle methods are:

1. **onCreate()**: This method is called when the activity is first created. It is used to initialize the activity, including setting up the UI and initializing variables.
2. **onStart()**: This method is called when the activity is becoming visible to the user. The activity is now visible but not interactive.
3. **onResume()**: This method is called when the activity starts interacting with the user. At this point, the activity is at the top of the activity stack, and the user can interact with it.
4. **onPause()**: This method is called when the system is about to start another activity. The current activity is still visible but partially obscured.
5. **onStop()**: This method is called when the activity is no longer visible to the user. It can happen because the activity is being destroyed or another activity is taking its place.
6. **onDestroy()**: This method is called before the activity is destroyed. It is the final call that the activity receives.
7. **onRestart()**: This method is called when the activity is being restarted after it was stopped. This can occur if the activity is being resumed from the background.

These methods allow developers to manage the activity's state, save data, and release resources appropriately. Understanding and handling these lifecycle methods correctly ensures that the application behaves predictably in different scenarios, such as screen rotations, incoming calls, or when the app is sent to the background.

Code:

> activity_main.xml

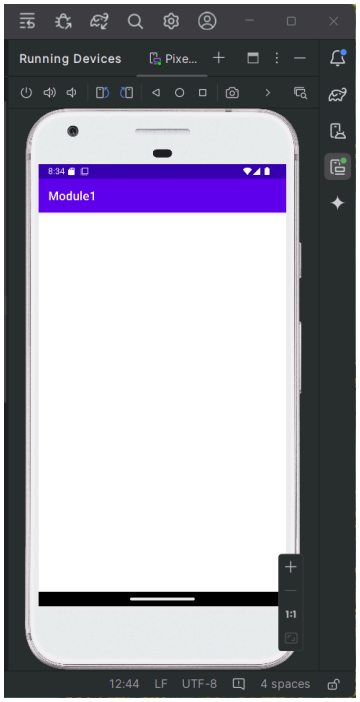
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

</LinearLayout>
```

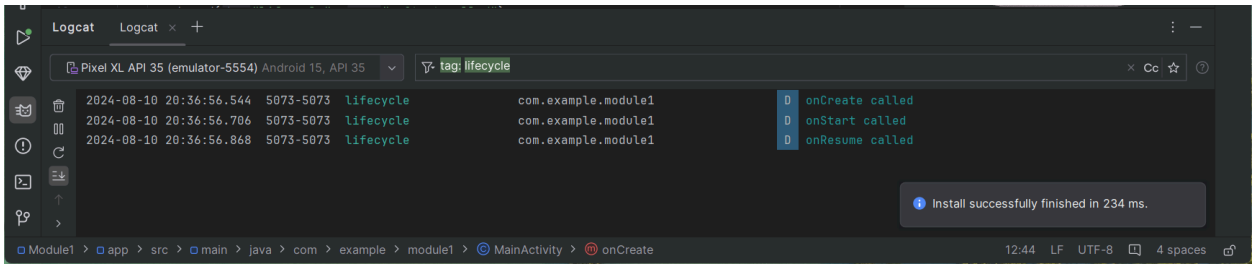
> MainActivity.java

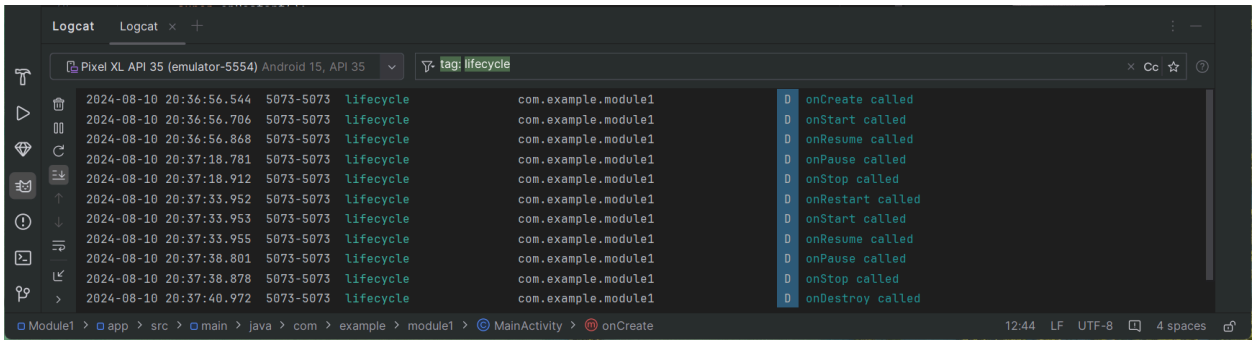
```
package com.example.module1;

import android.os.Bundle;
import android.util.Log;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("lifecycle", "onCreate called");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d("lifecycle", "onStart called");
    }

    @Override
    protected void onResume() {
```

```
        super.onResume();
        Log.d("lifecycle", "onResume called");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d("lifecycle", "onPause called");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d("lifecycle", "onStop called");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("lifecycle", "onDestroy called");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d("lifecycle", "onRestart called");
    }
}
```

Output:


Running Devices
Pixe...
Module1
12:44 LF UTF-8 4 spaces


Logcat
Logcat
Pixel XL API 35 (emulator-5554) Android 15, API 35
tag: lifecycle
2024-08-10 20:36:56.544 5073-5073 lifecycle com.example.module1 D onCreate called
2024-08-10 20:36:56.706 5073-5073 lifecycle com.example.module1 D onStart called
2024-08-10 20:36:56.868 5073-5073 lifecycle com.example.module1 D onResume called
Install successfully finished in 234 ms.
Module1 › app › src › main › java › com › example › module1 › MainActivity › onCreate
12:44 LF UTF-8 4 spaces


Logcat
Logcat
Pixel XL API 35 (emulator-5554) Android 15, API 35
tag: lifecycle
2024-08-10 20:36:56.544 5073-5073 lifecycle com.example.module1 D onCreate called
2024-08-10 20:36:56.706 5073-5073 lifecycle com.example.module1 D onStart called
2024-08-10 20:36:56.868 5073-5073 lifecycle com.example.module1 D onResume called
2024-08-10 20:37:18.781 5073-5073 lifecycle com.example.module1 D onPause called
2024-08-10 20:37:18.912 5073-5073 lifecycle com.example.module1 D onStop called
2024-08-10 20:37:33.952 5073-5073 lifecycle com.example.module1 D onRestart called
2024-08-10 20:37:33.953 5073-5073 lifecycle com.example.module1 D onStart called
2024-08-10 20:37:33.955 5073-5073 lifecycle com.example.module1 D onResume called
2024-08-10 20:37:38.801 5073-5073 lifecycle com.example.module1 D onPause called
2024-08-10 20:37:38.878 5073-5073 lifecycle com.example.module1 D onStop called
2024-08-10 20:37:40.972 5073-5073 lifecycle com.example.module1 D onDestroy called
Module1 › app › src › main › java › com › example › module1 › MainActivity › onCreate
12:44 LF UTF-8 4 spaces

2. Design the following User Interface using Linear Layout and Explicit Intent.

Aim: To design a User Interface (UI) using Linear Layout and implement navigation between activities using Explicit Intent in Android Studio

Objective:

- Create a UI using a Linear Layout.
- Arrange UI elements like TextView, EditText, Button, etc., in a vertical or horizontal order.
- Implement navigation between activities using Explicit Intent.
- Test the app to ensure that the UI is responsive and the navigation works correctly.

Theory:

In Android development, Linear Layout is a ViewGroup that aligns all children in a single direction, either vertically or horizontally. It's one of the most basic and commonly used layouts for arranging UI elements in a straightforward manner.

Explicit Intent is used in Android to navigate from one activity to another within the same application. It is called "explicit" because it explicitly defines the target component to handle the intent, such as specifying the activity class to be started.

By combining Linear Layout with Explicit Intent, developers can create intuitive and navigable UIs in Android applications. Linear Layout helps in aligning elements neatly, while Explicit Intent facilitates smooth transitions between different screens or activities.

Code:

> MainActivity2.java

```
package com.example.module1;

import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity2 extends AppCompatActivity {
```

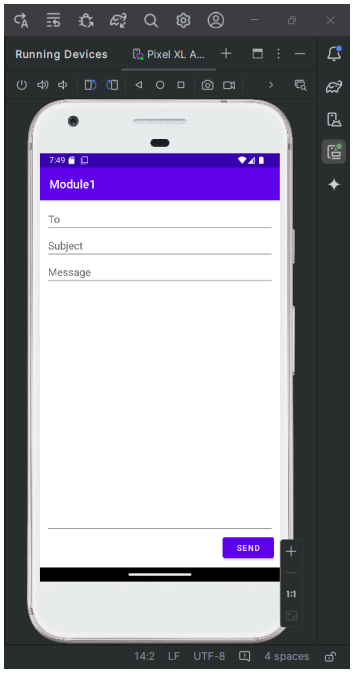
```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
    }
}
```

> activity_main2.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity2">

    <EditText
        android:id="@+id/to"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="To" />

    <EditText
        android:id="@+id/subject"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Subject" />

    <EditText
        android:id="@+id/message"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Message" />

    <EditText
        android:id="@+id/data"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:layout_weight="1" />
```

```
    <Button
        android:id="@+id/send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Send" />
</LinearLayout>
```

Output:



Code:

> MainActivity2.java

```
package com.example.module1;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity2 extends AppCompatActivity {

    EditText Username;
    Button Submit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        Username = findViewById(R.id.username);
        Submit = findViewById(R.id.submit);

        Submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String username = Username.getText().toString();
                Intent intent = new Intent(MainActivity2.this, ExplicitIntentActivity.class);
                intent.putExtra("user_name", username);
                startActivity(intent);
            }
        });
    }
}
```

> ExplicitIntentActivity.java

```
package com.example.module1;

import android.os.Bundle;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class ExplicitIntentActivity extends AppCompatActivity {

    TextView textView;

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

```
        textView = findViewById(R.id.welcome);

        String username = getIntent().getStringExtra("user_name");
        textView.setText("Welcome, " + username + "!");
    }
}
```

> activity_main2.xml

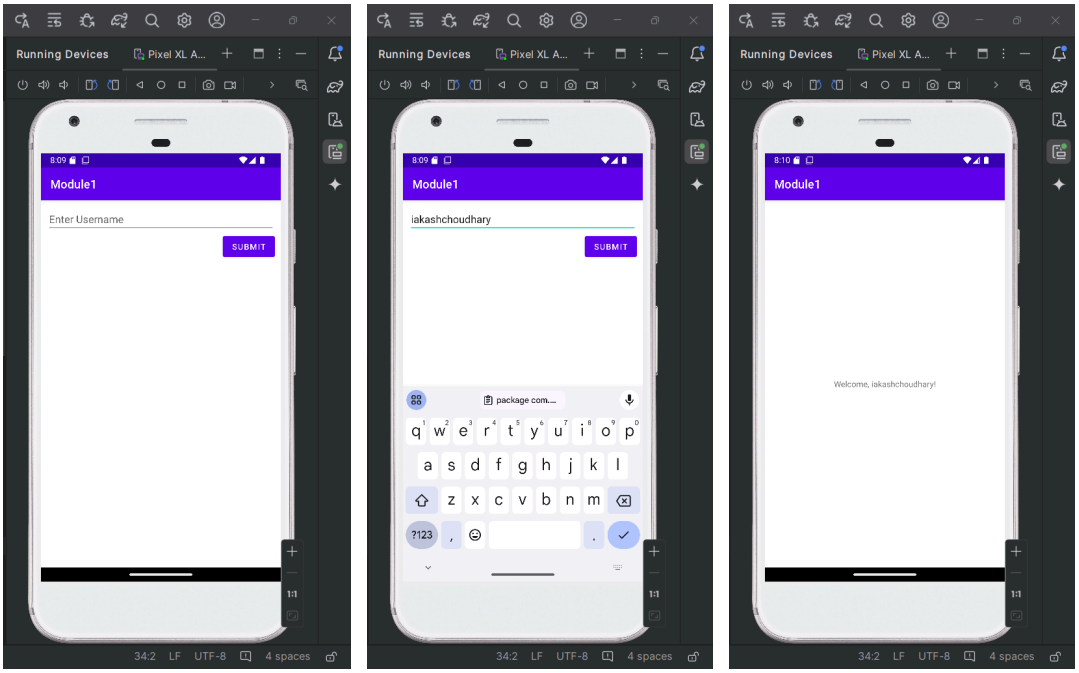
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity2">

    <EditText
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Enter Username" />

    <Button
        android:id="@+id/submit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Submit" />
</LinearLayout>
```

> activity_explicit_intent.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="10dp"
```

```
    tools:context=".ExplicitIntentActivity">

    <TextView
        android:id="@+id/welcome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome!" />
</LinearLayout>
```

Output:



3. Design the following User Interface using Relative Layout.

Aim: To design a User Interface (UI) for an Android application using a Relative Layout in Android Studio with Java

Objective:

- Align UI elements relative to each other and their parent.
- Create a dynamic and flexible layout that adjusts to different screen sizes and orientations.
- Implement a user-friendly interface using Relative Layout's features like alignment, margins, and padding.

Theory:

Relative Layout in Android Studio allows developers to position UI elements relative to one another or the parent container. This layout provides flexibility, making it possible to create complex and responsive designs. By defining rules such as alignParentTop, alignParentLeft, or below, developers can control the exact placement of elements on the screen. This is particularly useful for creating interfaces that need to adapt to different screen sizes and orientations without hard-coding absolute positions. In Java, the layout parameters can be set programmatically to further customize the UI based on specific conditions or user interactions.

Code:

> MainActivity3.java

```
package com.example.module1;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity3 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
    }
}
```

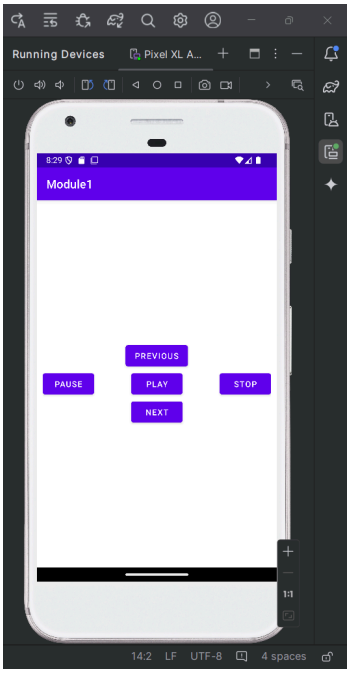
> activity_main3.xml

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp"
    tools:context=".MainActivity3">

    <Button
        android:id="@+id/play"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Play" />

    <Button
        android:id="@+id/pause"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:text="Pause" />

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

    <Button
        android:id="@+id/previous"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_above="@+id/play"
        android:text="Previous" />

    <Button
        android:id="@+id/next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/play"
```

```
        android:text="Next" />
</RelativeLayout>
```

Output:



4. Design the following User Interface using TableLayout.

Aim: Design a user interface using TableLayout to organize content in rows and columns effectively

Objective:

- Utilize TableLayout to create a structured layout.
- Arrange UI components in a tabular format.
- Enhance UI design by aligning elements in an organized manner.

Theory:

The TableLayout in Android is used to create a grid layout with rows and columns, similar to a table in HTML. Each child of a TableLayout is a TableRow that contains views like TextView, Button, or EditText. The TableRow objects are arranged sequentially to form rows, and the views within each TableRow are aligned into columns. This layout is especially useful when you need to display data in a tabular form, as it allows for a clean and organized arrangement of UI components.

Code:

> MainActivity4.java

```
package com.example.module1;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity4 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main4);
    }
}
```

> activity_main4.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout 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"
```

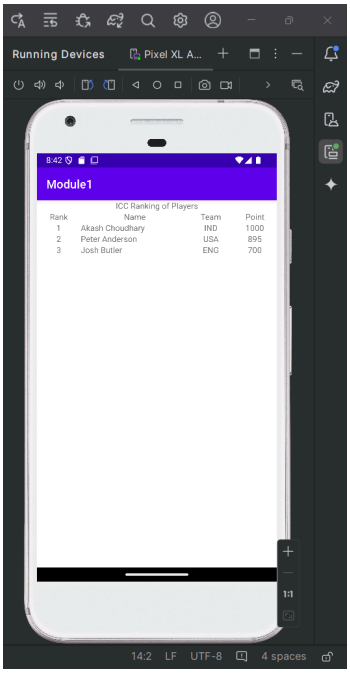
```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity4">

    <!-- Row 1 -->
    <TableRow
        android:orientation="vertical" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="ICC Ranking of Players"/>
    </TableRow>

    <!-- Row 2 -->
    <TableRow
        android:orientation="horizontal" >

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="Rank"/>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="Name"/>

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="Team"/>

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
```

```
        android:text="Point"/>
    </TableRow>

    <!-- Row 3 -->
    <TableRow
        android:orientation="horizontal" >

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="1"/>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="left"
            android:text="Akash Choudhary"/>

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="IND"/>

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="1000"/>
    </TableRow>

    <!-- Row 4 -->
    <TableRow
        android:orientation="horizontal" >

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="2"/>
```

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="left"
        android:text="Peter Anderson"/>

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:text="USA"/>

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:text="895"/>
</TableRow>

<!-- Row 5 -->
<TableRow
    android:orientation="horizontal" >

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:text="3"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="left"
        android:text="Josh Butler"/>

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
```

```
        android:text="ENG"/>

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:text="700"/>
  </TableRow>
</TableLayout>
```

Output:



5. Write a program to take user input from one activity and display it in another activity.

Aim: To create an Android application that takes user input in one activity and displays it in another activity using Java in Android Studio

Objective:

To learn how to:

- Create multiple activities in an Android application.
- Pass data between activities using Intent.
- Retrieve and display user input in the second activity.

Theory:

In Android development, an activity represents a single screen with a user interface. Often, we need to pass data between different activities. This can be achieved using Intents, which are messaging objects used to request an action from another app component.

To pass data from one activity to another, we use Intent.putExtra() method, which allows us to attach additional information to the Intent. The receiving activity can then retrieve this data using the getIntent() method along with Intent.getExtras().

This concept is fundamental in creating applications where user interaction and data flow between different screens are required.

Code:

> MainActivity5.java

```
package com.example.module1;

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

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity5 extends AppCompatActivity {
```

```
    EditText Name, Email, PhoneNo;
    Button Save;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main5);

        Name = findViewById(R.id.name);
        Email = findViewById(R.id.email);
        PhoneNo = findViewById(R.id.phoneno);

        Save = findViewById(R.id.save);
        Save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String name = Name.getText().toString();
                String email = Email.getText().toString();
                String phone = PhoneNo.getText().toString();
                Intent intent = new Intent(MainActivity5.this, SecondActivity5.class);
                intent.putExtra("name", name);
                intent.putExtra("email", email);
                intent.putExtra("phone", phone);
                startActivity(intent);
            }
        });
    }
}
```

> activity_main5.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity5">

    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:inputType="text"
        android:hint="Name" />

    <EditText
        android:id="@+id/email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="E-mail Address" />

    <EditText
        android:id="@+id/phoneno"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Phone Number" />

    <Button
        android:id="@+id/save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Save" />
</LinearLayout>
```

> SecondActivity5.java

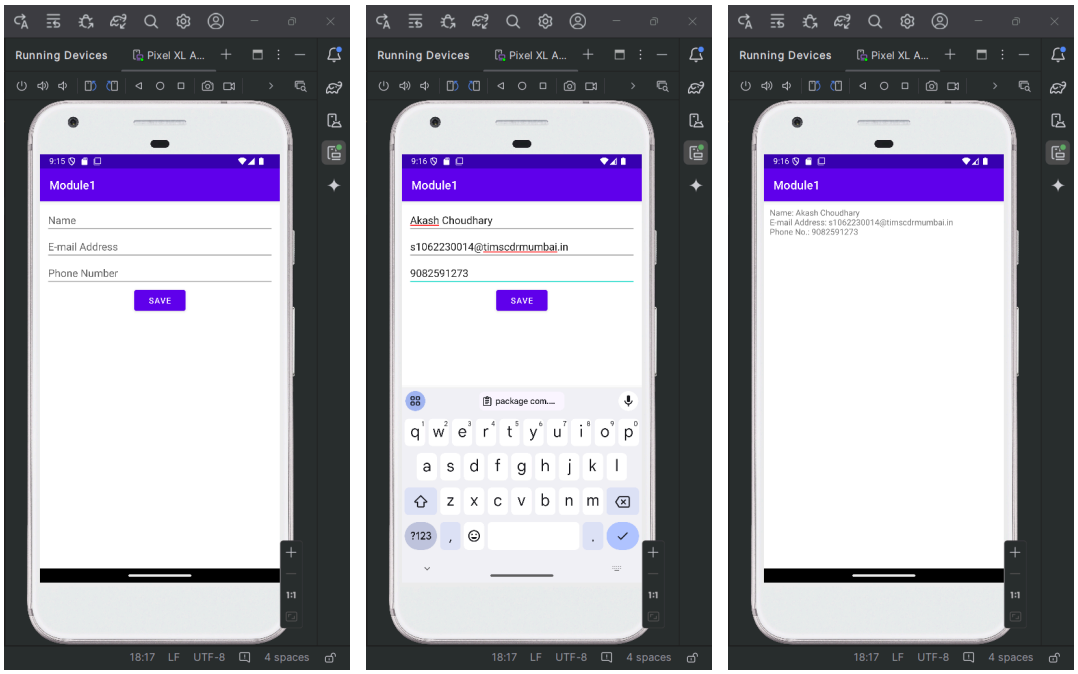
```
package com.example.module1;

import android.os.Bundle;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity5 extends AppCompatActivity {

    TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second5);

        textView = findViewById(R.id.textView);

        String name = getIntent().getStringExtra("name");
        String email = getIntent().getStringExtra("email");
```

```
        String phone = getIntent().getStringExtra("phone");

            textView.setText("Name: " + name + "\nE-mail Address: " + email + "\nPhone No.: " +
phone);
    }
}
```

> activity_second5.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".SecondActivity5">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Display text" />
</LinearLayout>
```

Output:




Run
app
2024-08-13 21:15:15: Launching app on 'Pixel XL API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.module1/.MainActivity5 }
Open logcat panel for emulator Pixel_XL_API_35
Connected to process 7299 on device 'Pixel_XL_API_35 [emulator-5554]'.
Install successfully finished in 1 s 375 ms.
Module1 > app > src > main > java > com > example > module1 > MainActivity5 > Save
18:17 LF UTF-8 4 spaces

6. Write a program to demonstrate implicit intent.

Aim: To demonstrate the use of implicit intent in an Android application

Objective:

- To understand the concept of implicit intents and how they can be used to trigger actions in other applications.
- To learn how to create and handle implicit intents to perform tasks such as opening a web page, sending an email, or viewing a location on a map.

Theory:

Implicit intents are a powerful feature in Android that allow applications to communicate with each other without specifying the exact component to handle the action. Unlike explicit intents, where the target component is specified by the developer, implicit intents let the system decide which component can best handle the requested action based on the intent data and action provided.

For example, an implicit intent can be used to open a web page by specifying the action ACTION_VIEW and passing a URI of the web page. The system will then determine which application can handle this action (such as a web browser) and launch it to fulfill the intent. Implicit intents provide flexibility and promote better app interoperability by allowing different apps to handle common actions without requiring tight coupling between them.

In Android development, understanding how to create and use implicit intents is crucial for enabling seamless user experiences across different apps and services.

Code:

> MainActivity6.java

```
package com.example.module1;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity6 extends AppCompatActivity {
```

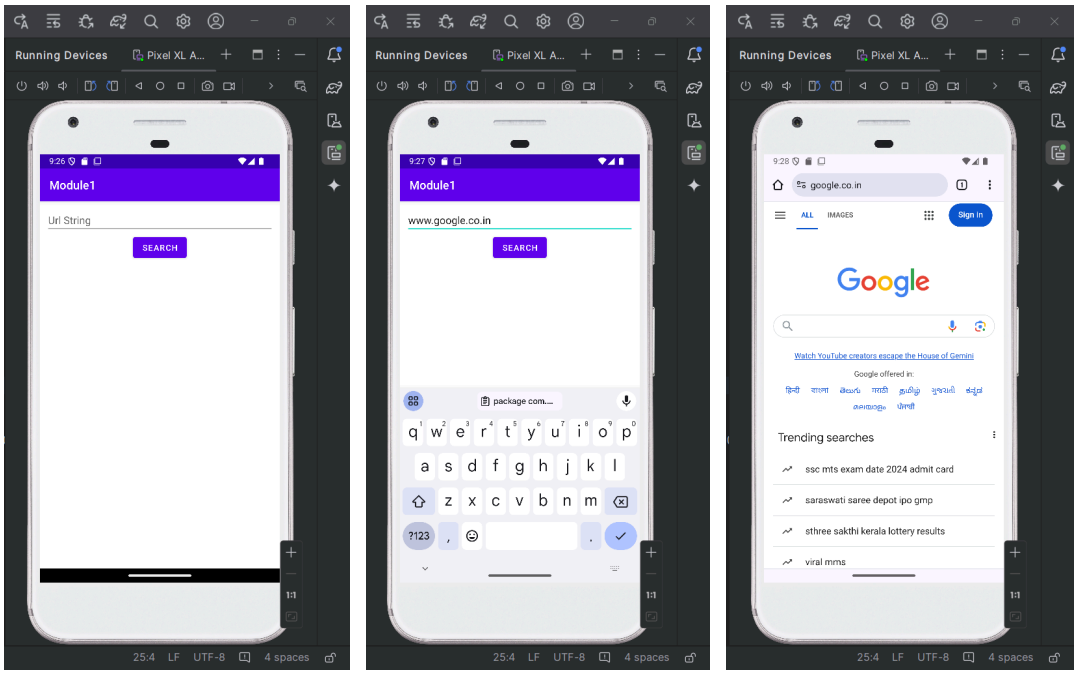
```
    EditText UrlString;
    Button Search;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main6);

        UrlString = findViewById(R.id.urlstring);

        Search = findViewById(R.id.search);
        Search.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://" +
UrlString.getText().toString()));
                startActivity(intent);
            }
        });
    }
}
```

> activity_main6.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity6">

    <EditText
        android:id="@+id/urlstring"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Url String" />

    <Button
        android:id="@+id/search"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:layout_gravity="center"
    android:text="Search" />
</LinearLayout>
```

Output:



7. Write a program to add two numbers and display the result in a toast message.

Aim: To develop an Android application that adds two numbers and displays the result using a toast message

Objective:

- To create a simple Android application using Java in Android Studio.
- To understand the basics of user input handling and toast messages in Android.

Theory:

In Android development, a toast is a small, non-interactive message that appears on the screen and automatically disappears after a short period. Toasts are typically used to provide feedback to the user. In this program, we create a user interface that allows the user to input two numbers. After processing the input, the application calculates the sum of the two numbers and displays the result using a toast message.

The process involves setting up an Android Studio project, defining the user interface in XML, handling user input, performing the addition operation in the Java activity file, and then using the Toast class to display the result. This simple application demonstrates fundamental Android development concepts such as UI design, event handling, and output display.

Code:

> MainActivity7.java

```
package com.example.module1;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity7 extends AppCompatActivity {

    EditText Num1, Num2;
    Button Add;

    @Override
```

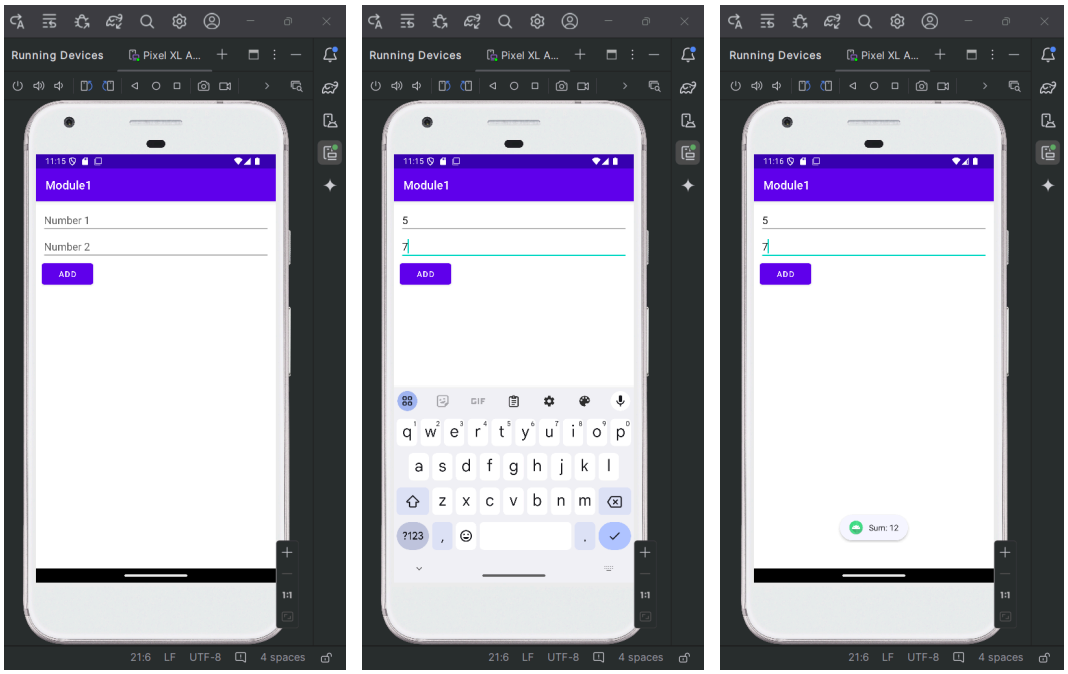
```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main7);

        Num1 = findViewById(R.id.number1);
        Num2 = findViewById(R.id.number2);

        Add = findViewById(R.id.add);
        Add.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String num1 = Num1.getText().toString();
                String num2 = Num2.getText().toString();
                int sum = Integer.parseInt(num1) + Integer.parseInt(num2);
                Toast.makeText(MainActivity7.this, "Sum: " + sum, Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

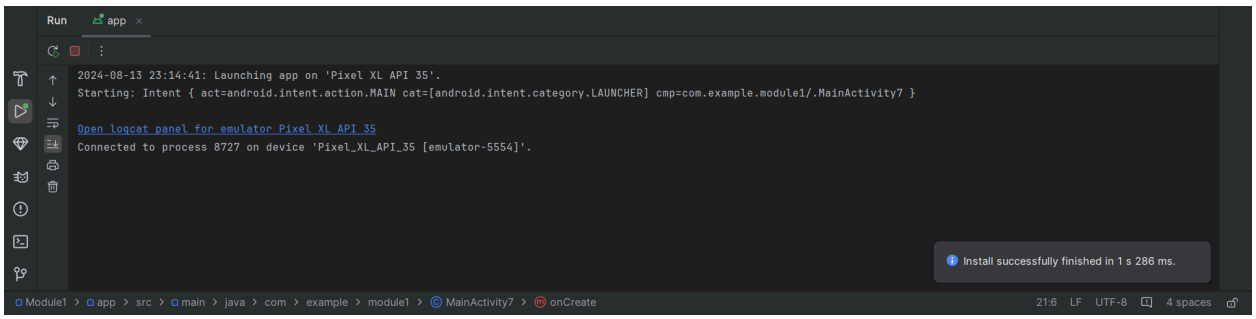
> activity_main7.xml

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity7">

    <EditText
        android:id="@+id/number1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Number 1" />

    <EditText
        android:id="@+id/number2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Number 2" />
```

```
    <Button
        android:id="@+id/add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add" />
</LinearLayout>
```

Output:





8. Design a screen that displays a frame image and write a quote on it.

Aim: To design a user interface in Android Studio using Java that displays an image within a frame and overlays a quote on it

Objective:

- To create a UI layout that incorporates a frame for an image.
- To position a text view on top of the image to display a quote.
- To ensure that the design is visually appealing and responsive across different screen sizes.

Theory:

In Android development, the user interface (UI) is designed using XML layouts and managed through Java code. The FrameLayout is a versatile layout manager in Android that allows stacking of views, where child views can be placed on top of each other. This is particularly useful when you want to display an image and overlay text on top of it.

The ImageView component is used to display an image in the layout, while the TextView component is used to display text. By using FrameLayout, the ImageView and TextView can be layered to create the desired effect. The layout can be further styled using attributes like padding, margins, text alignment, and image scaling to achieve the desired visual aesthetics.

In Android Studio, this UI design can be previewed in the design editor, ensuring it looks consistent across various devices and screen resolutions. The design can be enhanced by applying custom styles, fonts, and colors to match the app's theme.

Code:

> MainActivity8.java

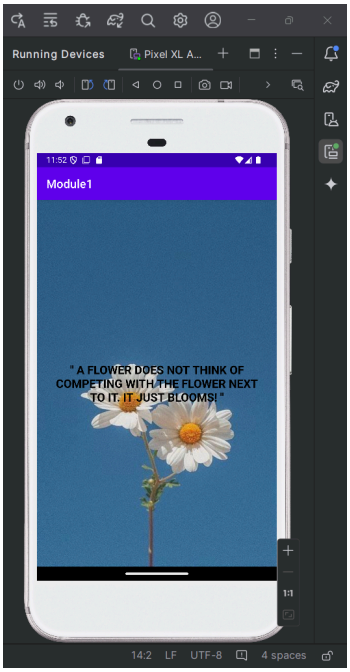
```
package com.example.module1;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity8 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main8);
```

```
    }
}
```

> activity_main8.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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=".MainActivity8">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="fitXY"
        app:srcCompat="@drawable/img" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="360dp"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:gravity="center"
        android:textSize="20sp"
        android:layout_gravity="center"
        android:textAllCaps="true"
            android:text="" A flower does not think of competing with the flower next to it. It just blooms! "" />
</FrameLayout>
```

Output:


Running Devices
Pixel XL A...
11:52
Module1
" A FLOWER DOES NOT THINK OF COMPETING WITH THE FLOWER NEXT TO IT. IT JUST BLOOMS! "
1:1
14:2 LF UTF-8 4 spaces

9. Write a program to demonstrate radio buttons and checkboxes.

Aim: To demonstrate the use of radio buttons and checkboxes in an Android application using Java

Objective:

- To understand the implementation of radio buttons for single-choice selection.
- To explore the use of checkboxes for multiple-choice selection.
- To learn how to handle user input using radio buttons and checkboxes in Android Studio.

Theory:

In Android development, radio buttons and checkboxes are essential UI components used to capture user preferences. A radio button allows users to select one option from a set, ensuring only one option can be chosen at a time. They are usually grouped in a RadioGroup to manage selection behavior. On the other hand, checkboxes enable users to select multiple options independently, meaning they can check or uncheck several options. The Android SDK provides built-in classes and methods to implement and manage these components. Using Java in Android Studio, developers can set up these UI elements in an XML layout and handle user interactions in the corresponding Java activity, ensuring dynamic and responsive applications.

Code:

> MainActivity9.java

```
package com.example.module1;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity9 extends AppCompatActivity {

    RadioGroup GenderGroup;
    CheckBox Anime, Games, Fashion;
    Button Display;
    TextView Data;
```

```
    RadioButton Gender;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main9);

        GenderGroup = findViewById(R.id.genderGroup);
        Anime = findViewById(R.id.watchAnime);
        Games = findViewById(R.id.playGames);
        Fashion = findViewById(R.id.fashion);
        Display = findViewById(R.id.display);
        Data = findViewById(R.id.data);

        Display.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String result = "\nSelected Values are:\n";
                Gender = findViewById(GenderGroup.getCheckedRadioButtonId());
                if (Anime.isChecked()) {
                    result += "Watching Anime, Movies & Web Series";
                }
                if (Fashion.isChecked()) {
                    result += "Fashion & Grooming";
                }
                if (Games.isChecked()) {
                    result += "Playing Games";
                }
                Data.setText("Selected gender is " + Gender.getText().toString() + result);
            }
        });
    }
}
```

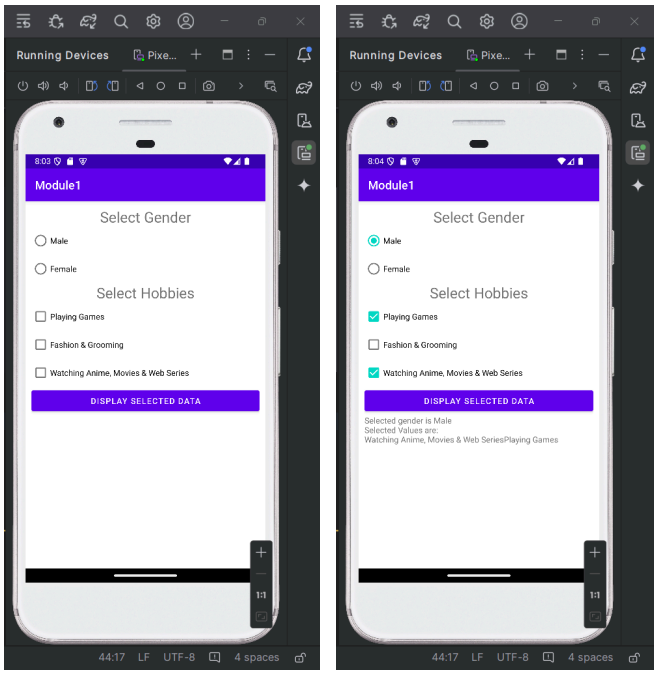
> activity_main9.xml

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity9">
```

```
<TextView
   android:id="@+id/gender"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:gravity="center"
   android:textSize="25dp"
   android:text="Select Gender" />

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

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

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

<TextView
   android:id="@+id/hobbies"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:gravity="center"
   android:textSize="25dp"
   android:text="Select Hobbies" />

<CheckBox
   android:id="@+id/playGames"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="Playing Games" />

<CheckBox
   android:id="@+id/fashion"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
```

```
        android:text="Fashion &amp;amp; Grooming" />

    <CheckBox
        android:id="@+id/watchAnime"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Watching Anime, Movies &amp;amp; Web Series" />

    <Button
        android:id="@+id/display"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Display Selected Data" />

    <TextView
        android:id="@+id/data"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" />
</LinearLayout>
```

Output:




Run
app
2024-08-14 20:03:30: Launching app on 'Pixel XL API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.module1/.MainActivity9 }
Open logcat panel for emulator Pixel XL API 35
Connected to process 4343 on device 'Pixel_XL_API_35 [emulator-5554]'.
Install successfully finished in 79 ms.
Module1 › app › src › main › java › com › example › module1 › MainActivity9 › onCreate › anonymous OnClickListener › onClick
44:17 LF UTF-8 4 spaces

10. Design an options menu (use WhatsApp's options menu as a reference).

Aim: Create a user-friendly options menu in an Android application, similar to WhatsApp's, to enhance user experience and provide easy access to various functionalities

Objective:

- Implement a contextual options menu that allows users to perform actions relevant to the current screen or selection.
- Include options like "Download," "Settings," and "Help" to facilitate common user tasks.
- Ensure menu items are easily accessible and logically organized for intuitive navigation.

Theory:

In Android Studio using Java, the options menu is a feature that provides users with a set of actions or choices that they can select from. It is typically accessed via the app's toolbar or an action button and is used to present options related to the current context or functionality of the app.

1. Menu Definition: Define the menu items in an XML file located in the res/menu directory. Each item should be given an ID, title, and optionally, an icon.
2. Menu Inflation: Inflate the menu XML in the onCreateOptionsMenu() method of your activity. This method is called by the system to display the menu when the user interacts with the menu button or toolbar.
3. Menu Item Handling: Handle menu item selections in the onOptionsItemSelected() method. This method is invoked when a user selects an item from the options menu. You can perform different actions based on the selected item's ID.

This approach ensures that the options menu enhances the usability of your application while following best practices in Android development.

Code:

> menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/download"
      android:title="Download" />
   <item android:id="@+id/search"
      android:title="Search" />
   <item android:id="@+id/help"
      android:title="Help" />
```

```
    <item android:id="@+id/settings"
        android:title="Settings" />
</menu>
```

> MainActivity10.java

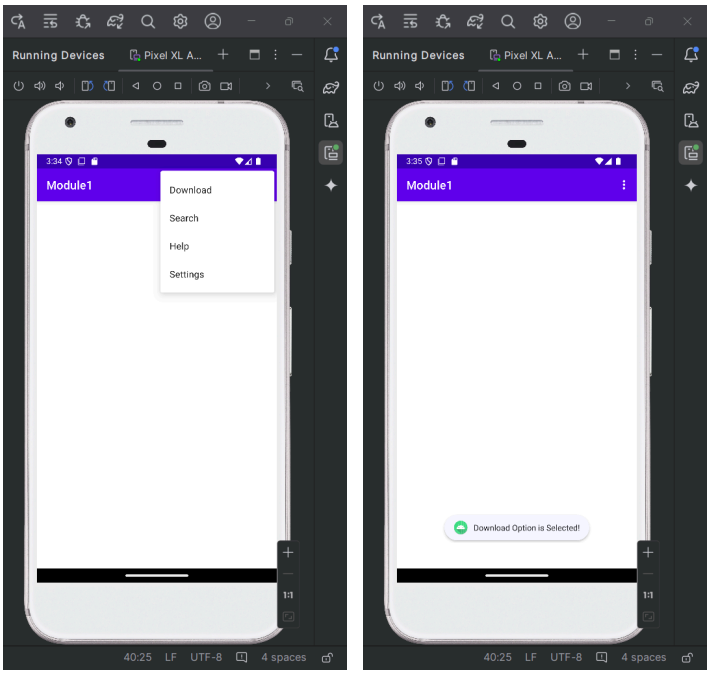
```
package com.example.module1;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity10 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main10);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.menu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.download) {
                            Toast.makeText(MainActivity10.this, "Download Option is Selected!",
Toast.LENGTH_SHORT).show();
            return true;
        }
        else if (id == R.id.search) {
                              Toast.makeText(MainActivity10.this, "Search Option is Selected!",
Toast.LENGTH_SHORT).show();
            return true;
        }
        else if (id == R.id.help) {
```

```
                Toast.makeText(MainActivity10.this, "Help Option is Selected!", Toast.LENGTH_SHORT).show();
            return true;
        }
        else if (id == R.id.settings) {
                Toast.makeText(MainActivity10.this, "Settings Option is Selected!", Toast.LENGTH_SHORT).show();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

> activity_main10.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity10">

</LinearLayout>
```

Output:




Run
app
2024-08-14 19:38:41: Launching app on 'Pixel XL API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.module1/.MainActivity10 }
Open logcat panel for emulator Pixel XL API 35
Connected to process 2155 on device 'Pixel_XL_API_35 [emulator-5554]'.
Install successfully finished in 9 s 528 ms.
Module1 › app › src › main › java › com › example › module1 › MainActivity10
48:2 LF UTF-8 4 spaces

11. Design an application with an image that displays a context menu when clicked, and create and redirect to different activities.

Aim: Design an Android application that displays a context menu when an image is clicked, allowing users to navigate to different activities

Objective:

- Implement an image view that triggers a context menu upon clicking.
- Create multiple activities and enable navigation between them through context menu options.

Theory:

In Android Studio using Java, the options menu is a feature that provides users with a set of actions or choices that they can select from. It is typically accessed via the app's toolbar or an action button and is used to present options related to the current context or functionality of the app.

- Contextual Menu: For more complex use cases, like WhatsApp's options menu, you might use context-specific menus (like long-press menus) that provide different actions based on the context of the user's interaction.

This approach ensures that the options menu enhances the usability of your application while following best practices in Android development.

In Android Studio using Java, a context menu is a floating menu that appears when users perform a long-click on an item (or click on an item in this case). This menu provides additional options related to the item. To implement this, you will need to use the registerForContextMenu method to register the view (image) for the context menu, override the onCreateContextMenu method to inflate the menu layout, and handle item selection in the onContextItemSelected method to navigate to different activities. This approach allows for a responsive and user-friendly interface where users can interact with the image to access various functionalities.

Code:

> menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/download"
      android:title="Download" />
```

```
    <item android:id="@+id/search"
        android:title="Search" />
    <item android:id="@+id/help"
        android:title="Help" />
    <item android:id="@+id/settings"
        android:title="Settings" />
</menu>
```

> MainActivity11.java

```
package com.example.module1;

import android.content.Intent;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity11 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main11);

        ImageView imageView = findViewById(R.id.imageView2);
        registerForContextMenu(imageView);
    }

    @Override
                public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.menu.menu, menu);
        super.onCreateContextMenu(menu, v, menuInfo);
    }

    @Override
    public boolean onContextItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.download) {
            Intent intent = new Intent(MainActivity11.this, DownloadActivity.class);
```

```
            startActivity(intent);
            return true;
        } else if (id == R.id.search) {
            Intent intent = new Intent(MainActivity11.this, SearchActivity.class);
            startActivity(intent);
            return true;
        } else if (id == R.id.help) {
            Intent intent = new Intent(MainActivity11.this, HelpActivity.class);
            startActivity(intent);
            return true;
        } else if (id == R.id.settings) {
            Intent intent = new Intent(MainActivity11.this, SettingsActivity.class);
            startActivity(intent);
            return true;
        }
        return super.onContextItemSelected(item);
    }
}
```

> activity_main11.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity11">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:srcCompat="@drawable/astronaut" />
</LinearLayout>
```

> DownloadActivity.java

```
package com.example.module1;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class DownloadActivity extends AppCompatActivity {
```

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

> activity_download.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".DownloadActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="You're redirected to download page!" />
</LinearLayout>
```

> SearchActivity.java

```
package com.example.module1;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class SearchActivity extends AppCompatActivity {

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

> activity_search.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".SearchActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="You're redirected to search page!" />
</LinearLayout>
```

> HelpActivity.java

```
package com.example.module1;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class HelpActivity extends AppCompatActivity {

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

> activity_help.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".HelpActivity">
```

```
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="You're redirected to help page!" />
</LinearLayout>
```

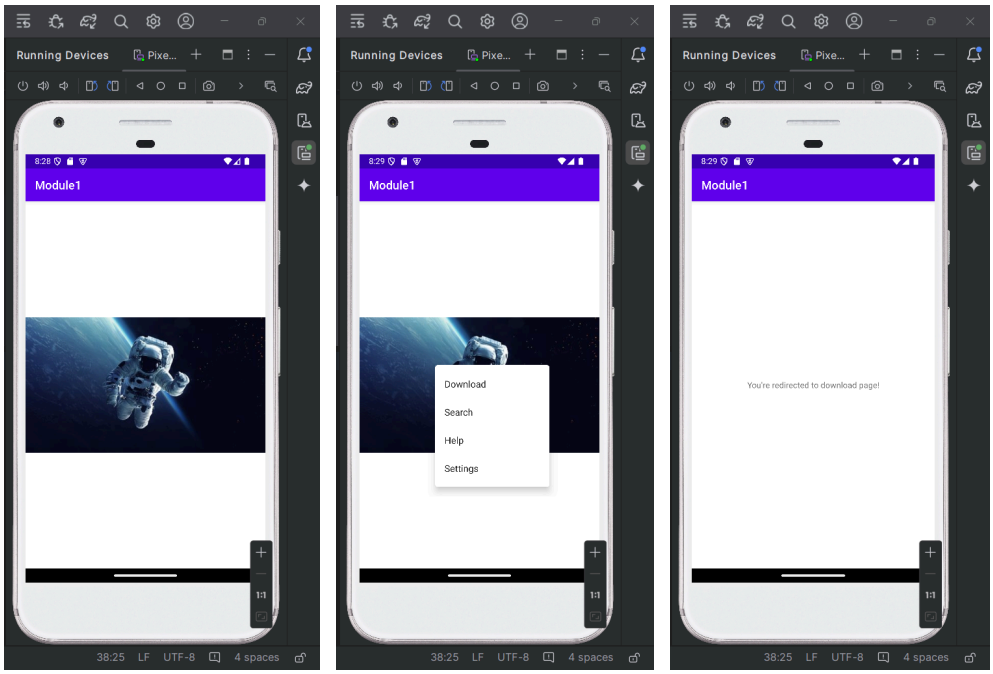
> SettingsActivity.java

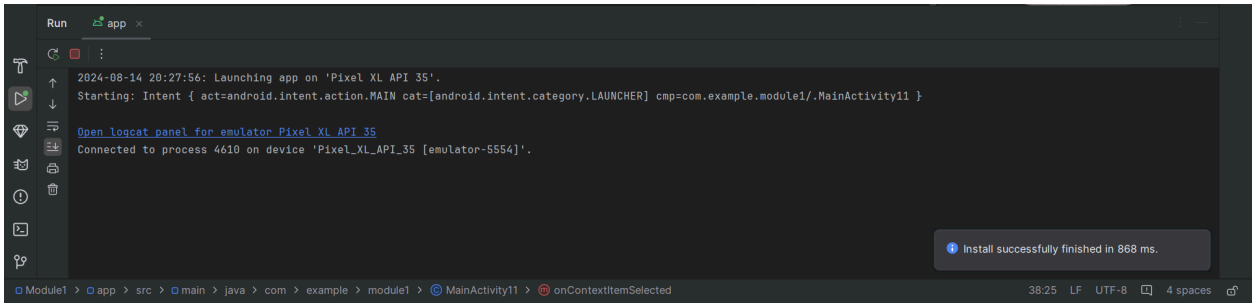
```
package com.example.module1;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class SettingsActivity extends AppCompatActivity {

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

> activity_settings.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".SettingsActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="You're redirected to settings page!" />
</LinearLayout>
```

Output:


Running Devices
Module1
Download
Search
Help
Settings
You're redirected to download page!


Run
app
2024-08-14 20:27:56: Launching app on 'Pixel XL API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.module1/.MainActivity11 }
Open logcat panel for emulator Pixel XL API 35
Connected to process 4610 on device 'Pixel_XL_API_35 [emulator-5554]'.
Install successfully finished in 868 ms.

12. Demonstrate different shapes of controls.

Aim: To showcase how to create and use various shapes for UI controls in Android Studio using Java

Objective:

- Understand how to define and apply different shapes (rectangles, ovals, and custom shapes) to UI controls.
- Learn to modify the appearance of UI controls using shape drawables.

Theory:

In Android development, UI controls can be visually customized using shape drawables. These drawables are XML files that define shapes, colors, and borders. They can be used to style buttons, text fields, and other UI elements. Shapes can be simple, like rectangles and ovals, or more complex, involving gradients and rounded corners. Understanding how to use these shapes allows for better control over the app's visual aesthetics and user experience.

Code:

> MainActivity12.java

```
package com.example.module1;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity12 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main12);
    }
}
```

> activity_main12.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity12">

    <TextView
        android:id="@+id/cylindrical"
        android:background="@drawable/cylindrical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Cylindrical" />

    <TextView
        android:id="@+id/oval"
        android:background="@drawable/oval"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:gravity="center"
        android:text="Oval" />

    <TextView
        android:id="@+id/rectangle"
        android:background="@drawable/rectangle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:gravity="center"
        android:text="Rectangle" />

    <TextView
        android:id="@+id/ring"
        android:background="@drawable/ring"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:gravity="center"
        android:text="Ring" />
</LinearLayout>
```

> cylindrical.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape                                  xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
    <stroke android:color="@color/black"
        android:width="2dp" />
```

```
    <corners android:radius="50dp" />
    <solid android:color="#EB455F" />
    <size android:width="120dp" android:height="40dp" />
</shape>
```

> oval.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="oval">
    <stroke android:color="@color/black"
        android:width="2dp" />
    <solid android:color="#52DE97" />
    <size android:width="75dp" android:height="30dp" />
</shape>
```

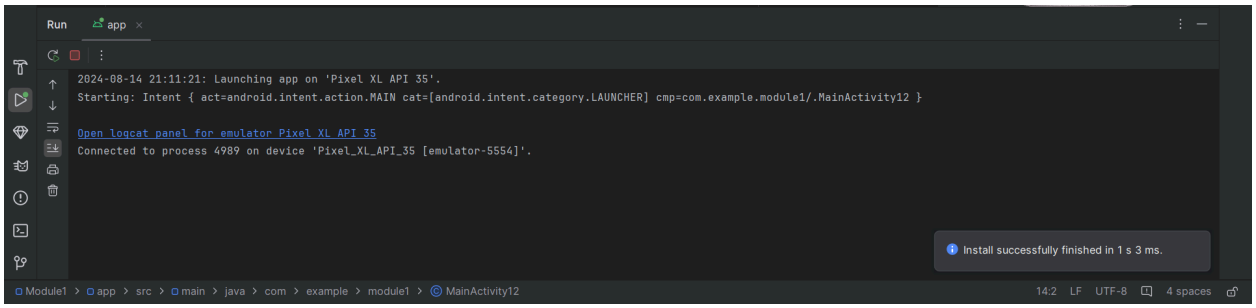
> rectangle.xml

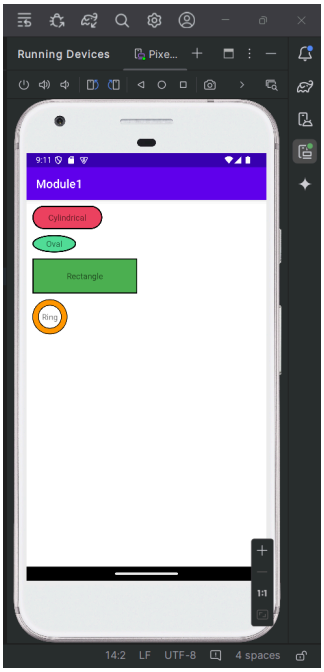
```
<?xml version="1.0" encoding="utf-8"?>
<shape                                    xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
    <stroke android:color="@color/black"
        android:width="2dp" />
    <solid android:color="#4CAF50" />
    <size android:width="180dp" android:height="60dp" />
</shape>
```

> ring.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="ring"
    android:thicknessRatio="6"
    android:useLevel="false">
    <stroke android:width="1dp" android:color="@color/black"/>
    <size android:width="60dp" android:height="60dp" />
    <solid android:color="#FF9800" />
</shape>
```

Output:




Running Devices
Pixe...
9:11
Module1
Cylindrical
Oval
Rectangle
Ring
1:1
14:2
LF
UTF-8
4 spaces

13. Write an application to increase font size using a SeekBar.

Aim: To create an Android application that allows users to adjust the font size of text dynamically using a SeekBar

Objective:

- Implement a SeekBar to control the font size of a TextView.
- Update the TextView's font size in real-time as the SeekBar is adjusted.

Theory:

In Android development, the SeekBar widget provides a way for users to select a value from a range by sliding a thumb along a horizontal bar. By integrating the SeekBar with a TextView, you can dynamically alter the font size of the text displayed to users. This involves setting up event listeners to respond to changes in the SeekBar's progress and adjusting the TextView's text size accordingly. The SeekBar's progress is typically mapped to a range of font sizes, allowing for flexible and interactive font size adjustments.

Code:

> MainActivity13.java

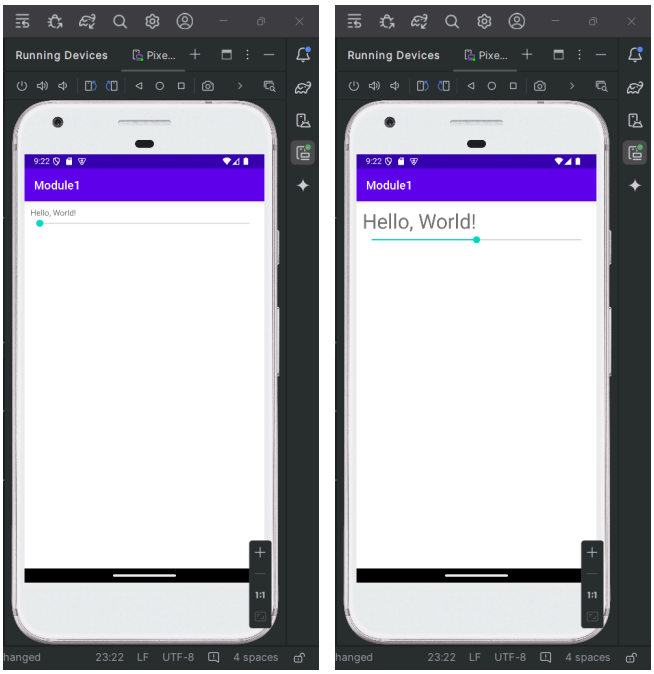
```
package com.example.module1;

import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity13 extends AppCompatActivity {

    TextView Data;
    SeekBar SizeRange;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main13);

        Data = findViewById(R.id.textView);
        SizeRange = findViewById(R.id.seekBar);

        SizeRange.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
```

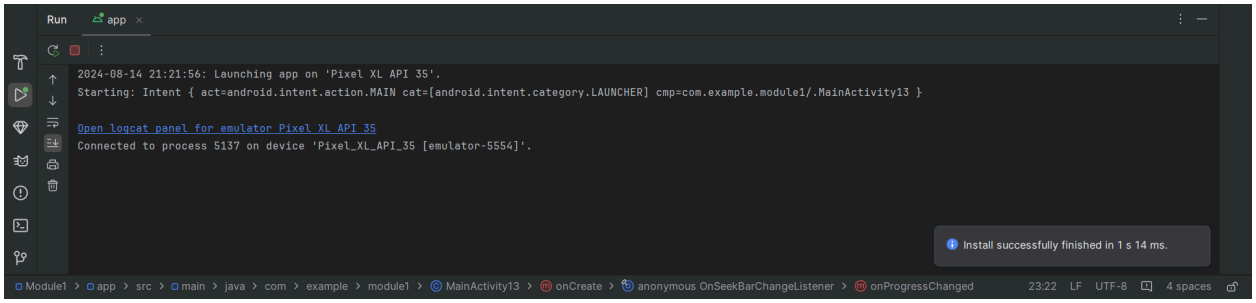
```
        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
            Data.setTextSize(i);
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });
  }
}
```

> activity_main13.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity13">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Hello, World!" />

    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="50"
        android:min="20" />
</LinearLayout>
```

Output:


Running Devices
Module1
Hello, World!
Running Devices
Module1
Hello, World!


Run
app
2024-08-14 21:21:56: Launching app on 'Pixel XL API 35'.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.module1/.MainActivity13 }
Open logcat panel for emulator Pixel XL API 35
Connected to process 5137 on device 'Pixel_XL_API_35 [emulator-5554]'.
Install successfully finished in 1 s 14 ms.

14. Write a program to demonstrate a background service in an Android application.

Aim: To create a background service in an Android application that performs tasks while the app is running in the background

Objective:

- Understand how to define and implement a background service in Android.
- Learn how to interact with a background service and ensure it runs even when the application is not in the foreground.

Theory:

In Android, a background service is a component that runs in the background to perform long-running operations or to perform work for other applications. Services can continue to run even when the app is not visible.

Key Concepts:

1. Service: A service is an application component that can perform long-running operations in the background and does not provide a user interface.
2. Lifecycle: The lifecycle of a service is different from that of an activity. A service can be started and stopped by other components and it will continue to run until explicitly stopped.
3. Binding: Services can be bound to other components, allowing them to interact with the service and retrieve data.

Implementation Steps:

- Define the Service: Create a service class by extending Service or IntentService.
- Register the Service: Declare the service in the AndroidManifest.xml.
- Start the Service: Use startService() or bindService() to initiate the service.
- Service Operations: Implement the onStartCommand() or onBind() methods to perform tasks in the background.
- Stop the Service: Ensure the service is properly stopped using stopSelf() or stopService().

Understanding these concepts helps in efficiently managing background tasks and ensures smooth operation and user experience in Android applications.

Code:

> MainActivity14.java

```
package com.example.module1;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity14 extends AppCompatActivity {

    Button Start, Stop;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main14);

        Start = findViewById(R.id.start);
        Stop = findViewById(R.id.stop);
    }

    public void startMusic(View view) {
        Intent intent = new Intent(MainActivity14.this, MusicService.class);
        startService(intent);
    }

    public void stopMusic(View view) {
        Intent intent = new Intent(MainActivity14.this, MusicService.class);
        stopService(intent);
    }
}
```

> activity_main14.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:gravity="center"
```

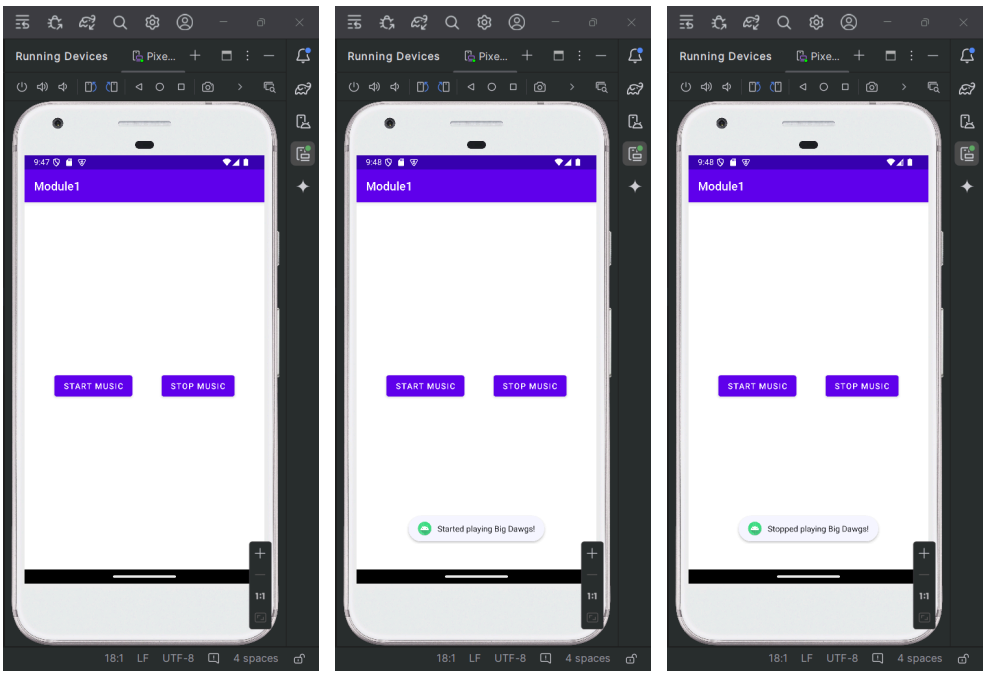
```
    android:padding="10dp"
    tools:context=".MainActivity14">

    <Button
        android:id="@+id/start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="startMusic"
        android:text="Start Music" />

    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="stopMusic"
        android:layout_marginStart="50dp"
        android:text="Stop Music" />
</LinearLayout>
```

> MusicService.java

```
package com.example.module1;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.widget.Toast;

public class MusicService extends Service {

    MediaPlayer mediaPlayer;

    public int onStartCommand(Intent intent, int flags, int startId) {
        mediaPlayer = MediaPlayer.create(MusicService.this, R.raw.music);
        mediaPlayer.setLooping(true);
        mediaPlayer.start();
                        Toast.makeText(MusicService.this, "Started playing Big Dawgs!",
Toast.LENGTH_SHORT).show();
        return START_STICKY;
    }

    public void onDestroy() {
        super.onDestroy();
        mediaPlayer.stop();
                        Toast.makeText(MusicService.this, "Stopped playing Big Dawgs!",
Toast.LENGTH_SHORT).show();
```

```
    }

    public MusicService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

Output:



15. Write a program to demonstrate a fragment in Android.

Aim: To demonstrate the use of fragments in an Android application

Objective:

- To understand the concept of fragments.
- To create a simple Android app that uses fragments to display different sections of the UI.

Theory:

In Android development, a fragment represents a portion of a user interface or behavior in an activity. Fragments are modular sections of an activity that can be combined to create a more flexible and reusable UI. Each fragment has its own layout and lifecycle, and it can be dynamically added, removed, or replaced within an activity. Using fragments helps in designing adaptive and responsive UIs that work well on various device configurations and screen sizes. Fragments are managed through the FragmentManager, which provides methods to perform transactions such as adding, removing, or replacing fragments in the activity's view hierarchy.

Code:

> MainActivity15.java

```
package com.example.module1;

import android.os.Bundle;
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentTransaction;

public class MainActivity15 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main15);
    }

    public void fragmentOne(View view) {
        FragmentOne fragmentOne = new FragmentOne();
                                                FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();
        fragmentTransaction.replace(R.id.frameSwitch, fragmentOne);
        fragmentTransaction.commit();
    }
```

```
    public void fragmentTwo(View view) {
        FragmentTwo fragmentTwo = new FragmentTwo();
                                                FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();
        fragmentTransaction.replace(R.id.frameSwitch, fragmentTwo);
        fragmentTransaction.commit();
    }
}
```

> activity_main15.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity15">

    <FrameLayout
        android:id="@+id/frameSwitch"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

    </FrameLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        android:gravity="center_horizontal"
        android:paddingTop="10dp">

        <Button
            android:id="@+id/fragmentOne"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="fragmentOne"
            android:text="Fragment One" />

        <Button
            android:id="@+id/fragmentTwo"
            android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:onClick="fragmentTwo"
        android:text="Fragment Two" />
  </LinearLayout>
</LinearLayout>
```

> FragmentOne.java

```
package com.example.module1;

import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link FragmentOne#newInstance} factory method to
 * create an instance of this fragment.
 */
public class FragmentOne extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public FragmentOne() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment FragmentOne.
     */
```

```
    // TODO: Rename and change types and number of parameters
    public static FragmentOne newInstance(String param1, String param2) {
        FragmentOne fragment = new FragmentOne();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

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

> fragment_one.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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"
    tools:context=".FragmentOne">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal"
        android:text="Hello, fragment one!" />

</FrameLayout>
```

> FragmentTwo.java

```
package com.example.module1;
```

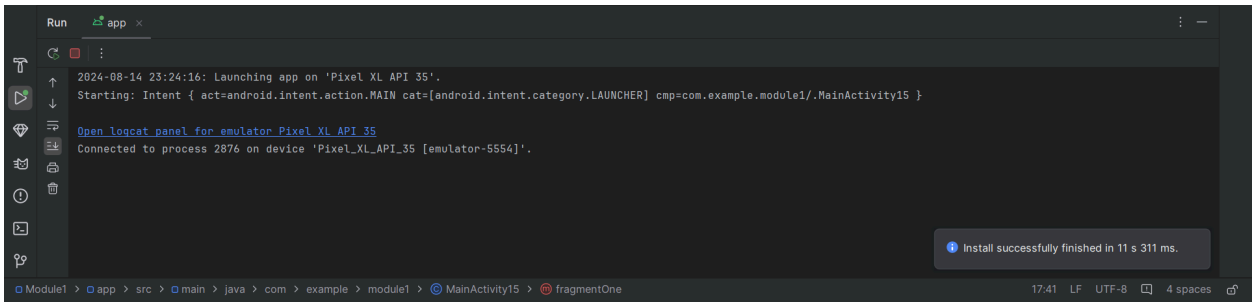
```
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link FragmentTwo#newInstance} factory method to
 * create an instance of this fragment.
 */
public class FragmentTwo extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public FragmentTwo() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment FragmentTwo.
     */
    // TODO: Rename and change types and number of parameters
    public static FragmentTwo newInstance(String param1, String param2) {
        FragmentTwo fragment = new FragmentTwo();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }
```

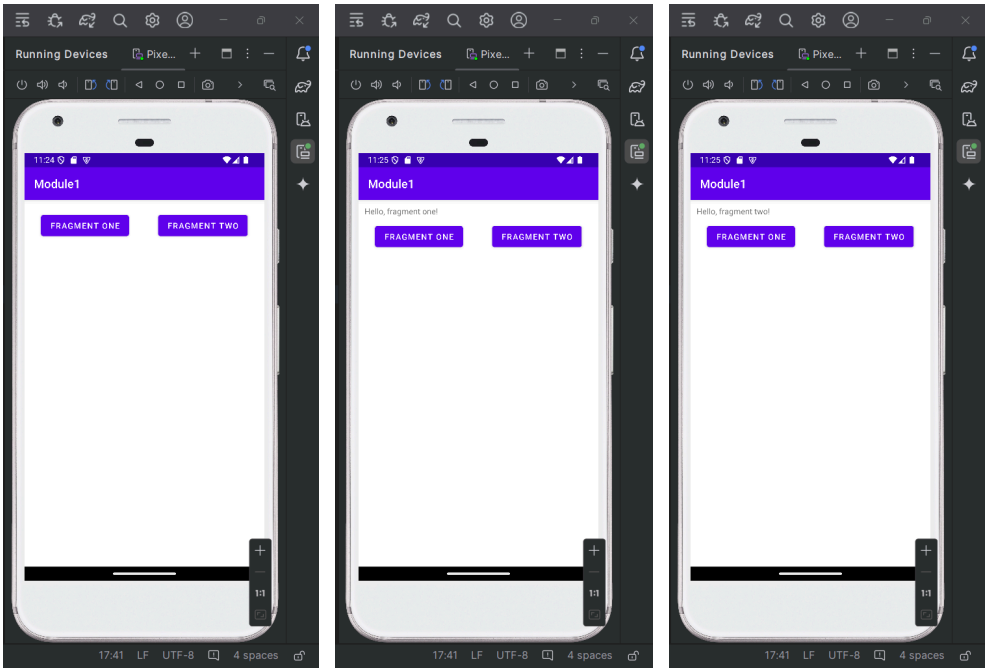
```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

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

> fragment_two.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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"
    tools:context=".FragmentTwo">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal"
        android:text="Hello, fragment two!" />

</FrameLayout>
```

Output:




Running Devices
Module1
FRAGMENT ONE
FRAGMENT TWO
Hello, fragment one!
Hello, fragment two!
17:41 LF UTF-8 4 spaces

16. Create a mobile application for a currency converter. Use a spinner for selecting the currency (minimum of 5 currencies).

Aim: To develop a mobile application that allows users to convert between multiple currencies using a simple and interactive interface

Objective:

- To implement a spinner for selecting from a minimum of five different currencies.
- To provide real-time currency conversion based on user input.

Theory:

In Android Studio using Java, the currency converter app leverages Spinner widgets to allow users to select from multiple currencies. The conversion logic utilizes APIs to fetch real-time exchange rates. This involves:

- Using ArrayAdapter to populate the spinner with currency options.
- Integrating a network library (such as Retrofit) to fetch live exchange rates.
- Performing currency conversion calculations based on user-selected currencies and displaying results in a user-friendly manner.

Code:

> MainActivity16.java

```
package com.example.module1;

import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity16 extends AppCompatActivity {

    Spinner To, From;
    EditText Amount;
    Button Convert;
    TextView Display;
```

```
    private String[] currencies = {"USD", "EUR", "INR", "GBP", "JPY"};
    private double[][] rates = {
            {1.0, 0.85, 74.34, 0.75, 110.62},
            {1.18, 1.0, 87.43, 0.88, 130.06},
            {0.013, 0.011, 1.0, 0.010, 1.49},
            {1.33, 1.14, 100.84, 1.0, 148.36},
            {0.009, 0.0077, 0.67, 0.0067, 1.0}
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main16);

        To = findViewById(R.id.to);
        From = findViewById(R.id.from);
        Amount = findViewById(R.id.amount);
        Convert = findViewById(R.id.convert);
        Display = findViewById(R.id.display);

                ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(MainActivity16.this,
android.R.layout.simple_spinner_item, currencies);

arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        To.setAdapter(arrayAdapter);
        From.setAdapter(arrayAdapter);

        Convert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                convertCurrency();
            }
        });
    }

    private void convertCurrency() {
        int from = From.getSelectedItemPosition();
        int to = To.getSelectedItemPosition();
        double dataAmount = Double.parseDouble(Amount.getText().toString());
        double result = dataAmount * rates[from][to];

        String fromCurrency = currencies[from];
        String toCurrency = currencies[to];

            String displayText = String.format("%.2f %s = %.2f %s", dataAmount, fromCurrency,
result, toCurrency);
```

```
        Display.setText(displayText);
    }
}
```

> activity_main16.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp"
    tools:context=".MainActivity16">

    <Spinner
        android:id="@+id/to"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"/>


    <Spinner
        android:id="@+id/from"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"/>


    <EditText
        android:id="@+id/amount"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal"
        android:layout_marginTop="20dp"
        android:layout_below="@+id/to"
        android:hint="Enter amount" />


    <Button
        android:id="@+id/convert"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/amount"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
```

```
        android:text="Convert" />


    <TextView
        android:id="@+id/display"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/convert"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:text="" />
</RelativeLayout>
```

Output: