Unit 5 : UI Fragments, Menus and Dialogs

1. Introduction to Fragments

Android Fragment is the part of the activity, which is also known as sub-activity. There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity.
A fragment is a reusable portion of the user interface in an Android activity. It has its own lifecycle, receives its own input events, and can be added or removed dynamically within an activity.

The Need for UI Flexibility

In Android development, creating dynamic and flexible user interfaces (UIs) that adapt to different screen sizes, orientations, and device types is essential. Activities are often used to manage a single screen in an app, but when you need to divide the UI into more manageable, reusable parts that can be updated independently, **fragments** are the solution. Fragments allow developers to create modular sections of an interface, making it easier to design responsive layouts for both phones and tablets.

Lifecycle of a Fragment

The fragment lifecycle is similar to an activity but with additional stages:

1. **onAttach()**: Called when the fragment is first attached to its context.
2. **onCreate()**: Initializes the fragment.
3. **onCreateView()**: Creates and returns the view hierarchy associated with the fragment.
4. **onViewCreated()**: Called after the fragment's view has been created.
5. **onStart()**: Fragment becomes visible.
6. **onResume()**: Fragment becomes active and ready for user interaction.
7. **onPause()**: Fragment is still visible but not active.
8. **onStop()**: Fragment is no longer visible.
9. **onDestroyView()**: Removes the view hierarchy.
10. **onDestroy()**: Final cleanup before fragment is destroyed.
11. **onDetach()**: Called when the fragment is detached from the activity.

Creating a UI Fragment

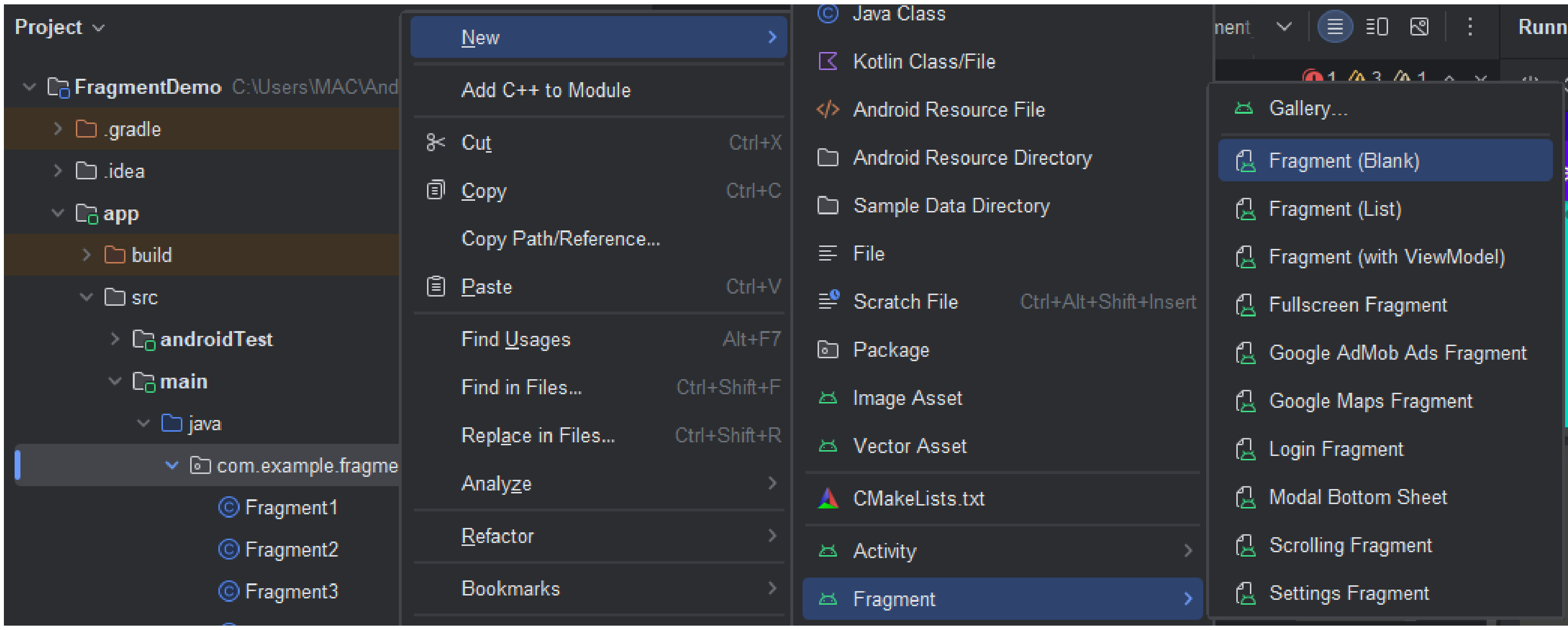
Creating a UI fragment in Android Studio involves a few steps. Here's a detailed guide to help you get started:

Step 1: Create a New Fragment Class

- Right-click on your project folder.
- Navigate to New > Fragment > Fragment (Blank).

Step 2: Define the Fragment Layout

- **Android Studio will automatically generate** a layout XML file for your fragment in the res/layout folder. Open this file (it will be named something like ***fragment_example.xml***).
- **Design your UI** using the XML editor or the Design view.
- Example layout file (res/layout/fragment_example.xml):



Creating a Fragment Class

To create a fragment, extend the Fragment class, then override key lifecycle methods to insert your app logic, similar to the way you would with an Activity class. One difference when creating a Fragment is that you must use the ***onCreateView()*** callback to define the layout. In fact, this is the only callback you need in order to get a fragment running. For example, here's a simple fragment that specifies its own layout:

ExampleFragment.java

```
public class FragmentExample extends Fragment {

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

Example of Fragment

Following example creates two fragments named as ***Fragment1*** and ***Fragment2***. After creating fragments, we will add these two fragments in activity named as ***MainActivity*** and display them. Firstly, we start by creating two fragments,

fragment1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:background="#AEB6BF"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
```

```
<TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="I am inside Fragment 1"
      android:layout_gravity="center"
      android:textSize="20sp"
      android:layout_marginTop="20sp"
      android:textStyle="bold" />
</LinearLayout>
```

<u>fragment2.xml</u>

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:background="#D0ECE7"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="I am inside Fragment 2"
      android:layout_gravity="center"
      android:textSize="20sp"
      android:layout_marginTop="20sp"
      android:textStyle="bold" />
</LinearLayout>
```

<u>fragment1.java</u>

```
public class Fragment1 extends Fragment {

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                      Bundle savedInstanceState) {
      // Inflate the layout for this fragment
      return inflater.inflate(R.layout.fragment_1, container, false);
   }
}
```

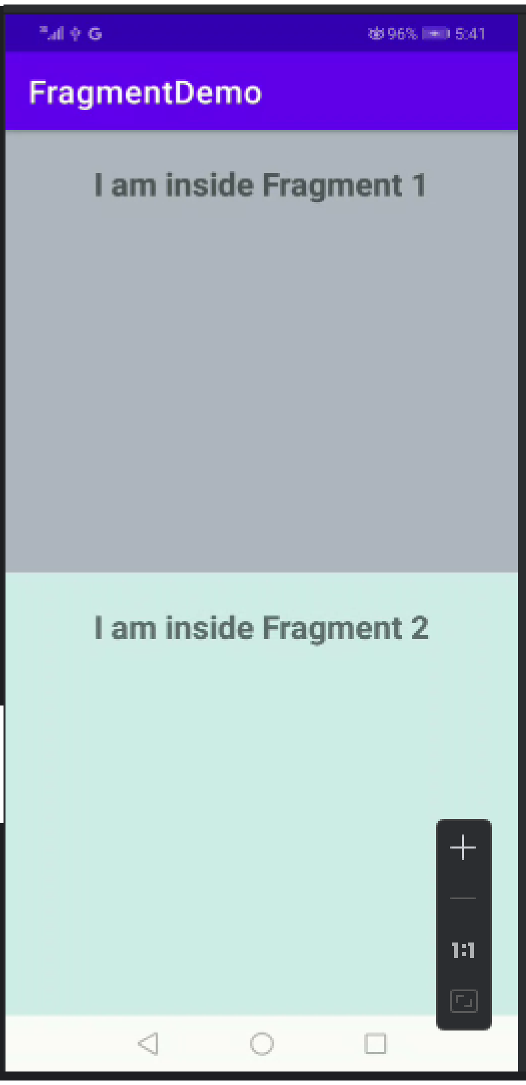
<u>Fragment2.java</u>

```
public class Fragment2 extends Fragment {
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                      Bundle savedInstanceState) {
      // Inflate the layout for this fragment
      return inflater.inflate(R.layout.fragment_2, container, false);
   }
}
```

Now we are going to add these fragments to Activity.

<u>first_activity.xml</u>

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   xmlns:android="http://schemas.android.com/apk/res/android">
   <fragment
      android:name="com.example.fragmentdemo.Fragment1"
      android:id="@+id/fragment1"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="1"/>
   <fragment
      android:name="com.example.fragmentdemo.Fragment2"
      android:id="@+id/fragment2"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="1"/>
</LinearLayout>
```



MainActivity.java

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

Wiring Widgets in Fragment

Wiring widgets in a fragment involves finding and linking the UI elements defined in the fragment's layout XML file to the corresponding views in the fragment's Java class. This allows you to interact with those UI elements, such as setting text, responding to button clicks, and more.

Steps to Wire Widgets in a Fragment

1. **Define the Fragment Layout XML**:
   - Create an XML layout file that defines the UI for the fragment.
2. **Create the Fragment Class**:
   - In the fragment class, inflate the layout in the onCreateView method.
   - Use findViewById to get references to the widgets.
3. **Use the Widgets in Fragment**:
   - Once you have the references, you can set listeners, change properties, and interact with the widgets.

Example Implementation

1. Define the Fragment Layout XML

Let's create a simple layout for a fragment with a `TextView` and a `Button`.

fragment_1.xml

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

    <TextView
        android:id="@+id/textViewFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_marginTop="50dp"
        android:layout_gravity="center"
        android:text="Hello from Fragment!" />

    <Button
        android:id="@+id/buttonFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Click Me" />
</LinearLayout>
```

2. Create the Fragment Class

Now, create a Java class for the fragment and wire the widgets.

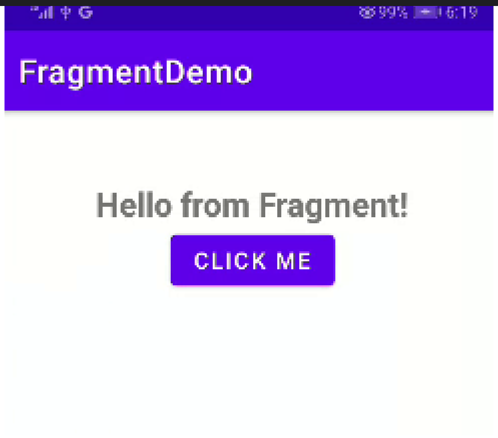
Fragment1.java

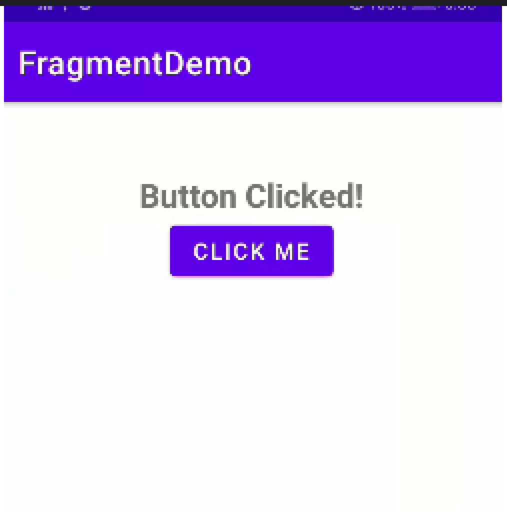
```java
public class Fragment1 extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_1, container, false);

        // Wiring widgets
        TextView textView = view.findViewById(R.id.textViewFragment);
        Button button = view.findViewById(R.id.buttonFragment);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView.setText("Button Clicked!");
            }
        });
        return view;
    }
}
```

Now add above fragment to Activity

activity_main.xml

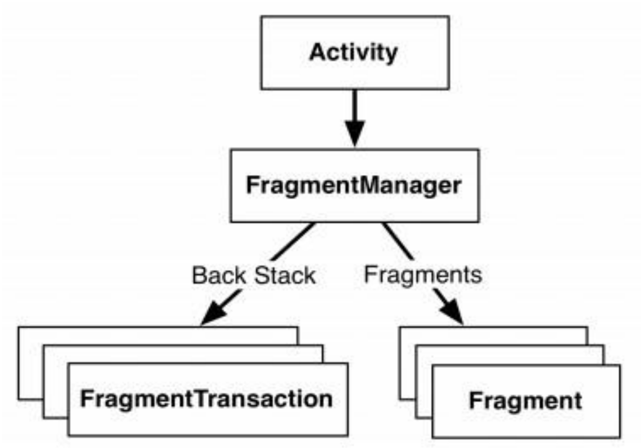
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <fragment
        android:name="com.example.fragmentdemo.Fragment1"
        android:id="@+id/fragment1"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```





Introduction to FragmentManager

FragmentManager is responsible for managing fragments within an activity. It handles the addition, removal, and replacement of fragments in an activity.



Example of Adding a Fragment:

MainActivity.java

```java
Fragment fragment=new Fragment1();
FragmentManager manager = getSupportFragmentManager();
FragmentTransaction transaction = manager.beginTransaction();
transaction.add(R.id.myfragment, fragment);

transaction.commit();
```

Example Implementation

Let’s create two fragments and one activity. Activity contains two buttons for switching fragments i.e. if first button is clicked Fragment1 is displayed and if second button is clicked Fragment2 is displayed in the screen.

fragment_1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#AEB6BF"
    android:orientation="vertical">
    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="I am inside fragment 1"
        android:textSize="20sp" />
</LinearLayout>
```

fragment_2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#D0ECE7"
    android:orientation="vertical">
    <TextView
        android:id="@+id/text2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="I am inside fragment 2"
        android:textSize="20sp" />
</LinearLayout>
```

Fragment1.java

```
public class Fragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_1, container, false);
    }
}
```

Fragment2.java

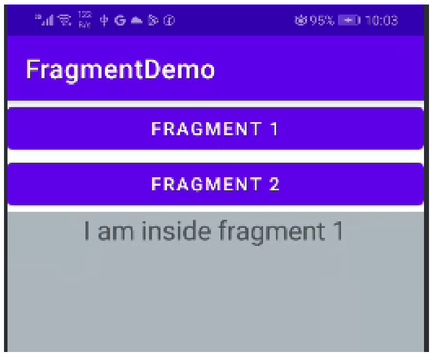
```
public class Fragment2 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_2, container, false);
    }
}
```

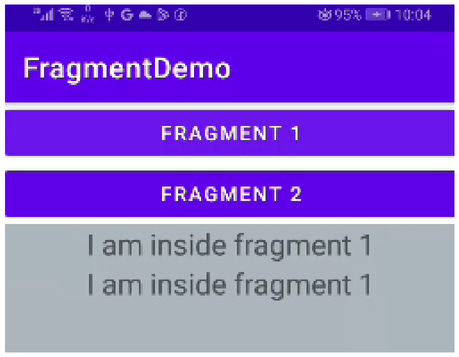
activity_main.xml

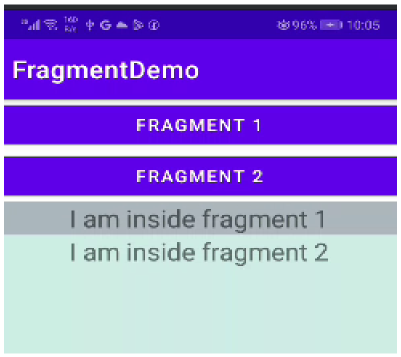
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Fragment 1"
        android:id="@+id/btn1"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Fragment 2"
        android:id="@+id/btn2"/>
    <fragment
        android:name="com.example.fragmentdemo.Fragment1"
        android:id="@+id/myfragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    Button btn1, btn2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn1 = findViewById(R.id.btn1);
        btn2 = findViewById(R.id.btn2);
        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Fragment f = new Fragment1();
                FragmentManager fm = getSupportFragmentManager();
                FragmentTransaction ft = fm.beginTransaction();
                ft.replace(R.id.myfragment, f);
                ft.commit();
            }
        });
        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Fragment f = new Fragment2();
                FragmentManager fm = getSupportFragmentManager();
                FragmentTransaction ft = fm.beginTransaction();
                ft.replace(R.id.myfragment, f);
                ft.commit();
            }
        });
    }
}
```

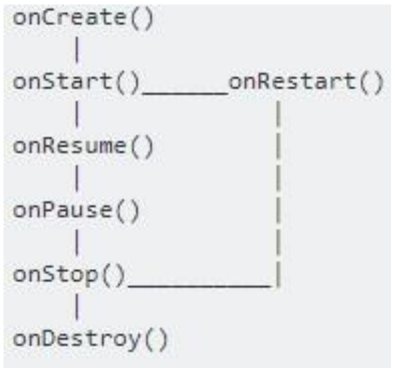
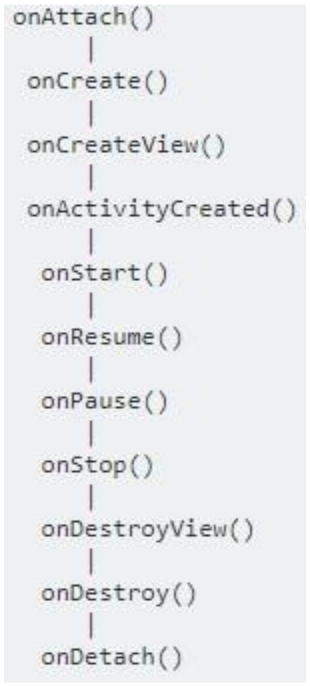






Difference Between Activity and Fragments

S.N.	Activity	Fragment
1)	Activity is an application component that gives a user interface where the user can interact.	Fragment is a part of an activity, which contributes its own UI to that activity.
2)	Activity is not dependent on Fragment.	Fragment is dependent on Activity, it can't exist independently.
3)	Without using fragment in Activity we can't create multi-pane UI.	Using multiple fragments in a single activity we can create multi-pane UI.
4)	For Activity, we just need to mention in Manifest.	For fragment it's not required.
5)	Activity is heavy weight.	Fragment is light weight.
6)	Activity use a lot of memory.	Fragment doesn't use any memory because it resides on Activity.
7)	Activity is the UI of an application through which user can interact.	Fragment is the part of the Activity, it is an sub-Activity inside activity which has its own Life Cycle which runs parallel to the Activities Life Cycle.

8)	Lifecycle of Activity:  onCreate()
	onStart()______onRestart()
	onResume()
	onPause()
	onStop()___________
	onDestroy()
		Lifecycle of Fragment:  onAttach()
		onCreate()
		onCreateView()
		onActivityCreated()
		onStart()
		onResume()
		onPause()
		onStop()
		onDestroyView()
		onDestroy()
		onDetach()

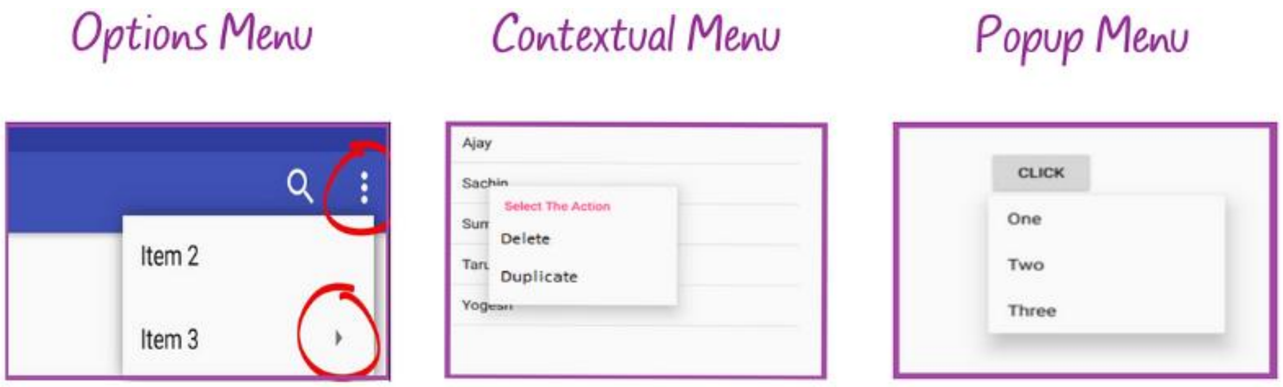
9)	Lifecycle methods of Activity are hosted by OS.	Lifecycle methods of Fragments are hosted by are hosted by hosting activity.
10)	When an activity is placed to the back stack of activities the user can navigate back to the previous activity by just pressing the back button.	When an fragment is placed to the activity we have to request the instance to be saved by calling addToBackstack() during the fragment transaction.

2. Menus

Menus in Android provide a way for users to interact with your app using an action bar or toolbar. Menus can include actions, navigation items, or options.

Types of Menus

- **Options Menu**: Appears when the user presses the menu button or in the app's action bar.
- **Context Menu**: Appears when the user performs a long-click on an element.
- **Popup Menu**: Displays a small list of items anchored to a view.



Implementing a Menu in an Application

For all menu types, Android provides a standard XML format to define menu items. Instead of building a menu in your activity's code, you should define a menu and all its items in an XML menu resource. You can then inflate the menu resource (load it as a Menu object) in your activity or fragment.

To define the menu, create a new directory named ***menu*** under the ***res*** directory. Then create an XML file inside your project's **res/menu/** directory ***(eg.mymenu.xml)*** and build the menu with the following elements:
<menu> : Defines a Menu, which is a container for menu items. A <menu> element must be the root node for the file and can hold one or more <item> and <group> elements.
<item> : Creates a Menu Item, which represents a single item in a menu. This element may contain a nested <menu> element in order to create a submenu.
<group> : An optional, invisible container for <item> elements. It allows you to categorize menu items so they share properties such as active state and visibility.
1. Define the Menu in XML:
```
<!-- res/menu/mymenu.xml -->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_settings"
        android:title="Settings"/>
</menu>
```

2. Inflate the Menu in Activity:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.mymenu, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_settings:
            // Handle settings click
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Example Implementation

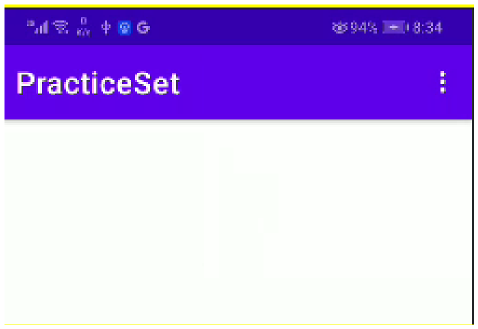
Creating Options Menu

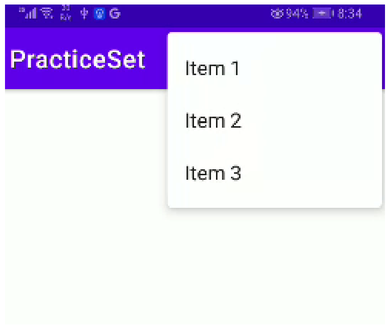
mymenu.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item  android:id="@+id/item1"
        android:title="Item 1"/>

    <item  android:id="@+id/item2"
        android:title="Item 2"/>

    <item  android:id="@+id/item3"
        android:title="Item 3" />
</menu>
```

MainActivity.java

```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
//adding options menu
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.mymenu, menu);
        return true;
    }
//handling clicks
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int itemId = item.getItemId();
        if (itemId == R.id.item1) {
            Toast.makeText(this, "Item 1 selected", Toast.LENGTH_SHORT).show();
            return true;
        } else if (itemId == R.id.item2) {
            Toast.makeText(this, "Item 2 selected", Toast.LENGTH_SHORT).show();
            return true;
        } else if (itemId == R.id.item3) {
            Toast.makeText(this, "Item 3 selected", Toast.LENGTH_SHORT).show();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```





Creating Context Menu

To provide a floating context menu:

1. Register the View to which the context menu should be associated by calling **registerForContextMenu**() and pass it the View.
2. Implement the **onCreateContextMenu**() method in your Activity or Fragment.
3. For event handling on click implement **onContextItemSelected**().

context_menu.xml

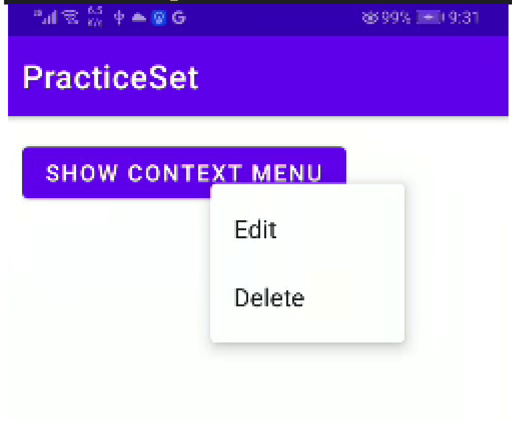
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item  android:id="@+id/item1"
        android:title="Item 1"/>

    <item  android:id="@+id/item2"
        android:title="Item 2"/>

    <item  android:id="@+id/item3"
        android:title="Item 3" />
</menu>
```

MainActivity.java

```

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

        btn = findViewById(R.id.btn);
        // Registering the TextView for the context menu
        registerForContextMenu(btn);
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo){
        super.onCreateContextMenu(menu, v, menuInfo);
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.context_menu, menu);
    }
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        int itemId = item.getItemId();
        if (itemId == R.id.action_edit) {
            Toast.makeText(this, "Edit selected", Toast.LENGTH_SHORT).show();
            return true;
        } else if (itemId == R.id.action_delete) {
            Toast.makeText(this, "Delete selected", Toast.LENGTH_SHORT).show();
            return true;
        }
        return super.onContextItemSelected(item);
    }
}
```

activity_main.xml

```
 <?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
    <Button
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show context menu" />
</LinearLayout>
```



Creating Popup Menu

popup_menu.xml

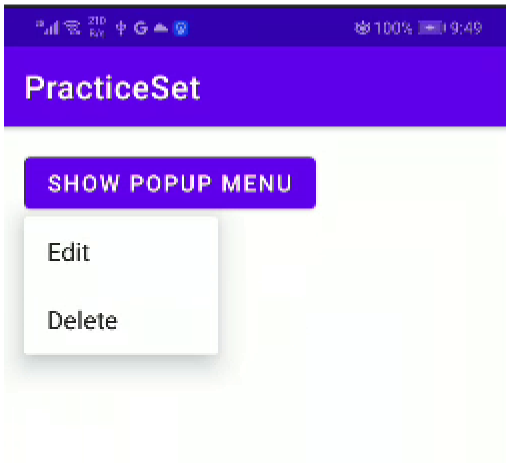
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item  android:id="@+id/item1"
        android:title="Item 1"/>

    <item  android:id="@+id/item2"
        android:title="Item 2"/>

    <item  android:id="@+id/item3"
        android:title="Item 3" />
</menu>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    Button myButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        myButton = findViewById(R.id.my_button);

        myButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Creating a PopupMenu
                PopupMenu popup = new PopupMenu(MainActivity.this, myButton);
                MenuInflater inflater = popup.getMenuInflater();
                inflater.inflate(R.menu.mymenu, popup.getMenu());

                // Handling clicks on the popup menu items
                popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                    @Override
                    public boolean onMenuItemClick(MenuItem item) {
                        int itemId = item.getItemId();
                        if (itemId == R.id.action_edit) {
                            Toast.makeText(MainActivity.this, "Edit selected", Toast.LENGTH_SHORT).show();
                            return true;
                        } else if (itemId == R.id.action_delete) {
                            Toast.makeText(MainActivity.this, "Delete selected", Toast.LENGTH_SHORT).show();
                            return true;
                        }
                        return false;
                    }
                });
                // Show the popup menu
                popup.show();
            }
        });
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
    <Button
        android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Popup Menu" />
</LinearLayout>
```



3. Dialogs

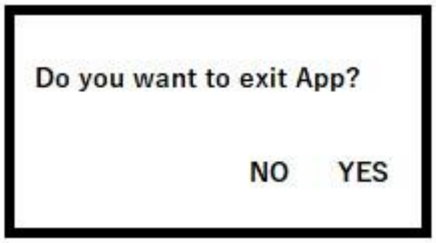
Dialogs in Android are small windows that prompt the user to make a decision or enter additional information.

Creating a Dialog Fragment

A DialogFragment is a fragment that displays a dialog window, floating on top of its activity's content.

```
// DialogFragment.java
      public class MainActivity extends DialogFragment {
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
          AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
          builder.setMessage("Do you want to proceed?")
                  .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                      public void onClick(DialogInterface dialog, int id) {
                          // Proceed with the action
```

```
                }
            })
            .setNegativeButton("No", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    // Cancel the action
                }
            });
                //create the AlertDialog object and return it
        return builder.create();
    }
}
```


Do you want to exit App?

NO YES

Setting a Dialog's Content

You can set the content of the dialog using the `AlertDialog.Builder` methods:

```
builder.setMessage("Message");
builder.setTitle("Title");
builder.setView(R.layout.custom_layout); // To set a custom layout
```

To show the dialog:

```
MyDialogFragment dialogFragment = new MyDialogFragment();
dialogFragment.show(getSupportFragmentManager(), "MyDialogFragment");
```

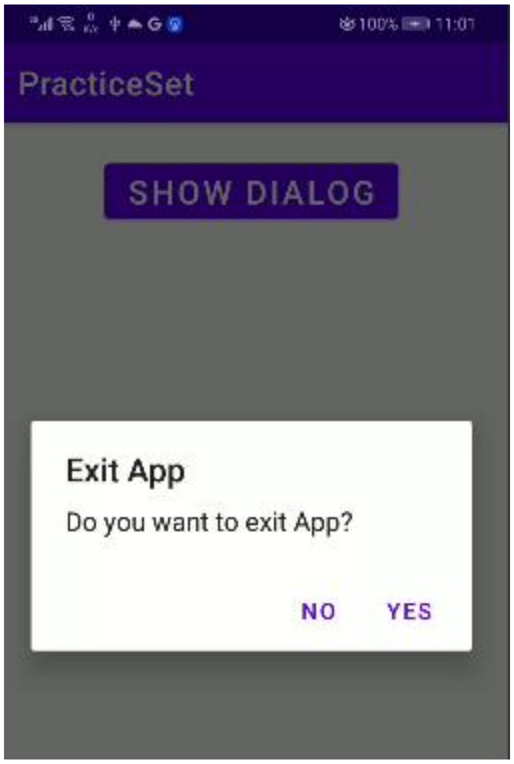
Example Implementation

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
        Button btnClick;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            btnClick=findViewById(R.id.btnClick);
            btnClick.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    showDialog();
                }
            });
        }
        public void showDialog(){
            AlertDialog.Builder builder = new AlertDialog.Builder
                    (MainActivity.this);
            builder.setTitle("Exit App");
            builder.setMessage("Do you want to exit App?");
            builder.setCancelable(true);
            builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            //your stuffs
                        }
                    });
            builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            dialog.cancel();
                        }
                    });
            AlertDialog alert = builder.create();
            alert.show();
        }
    }
```

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Dialog"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20sp"
        android:id="@+id/btnClick"
        android:textSize="20sp" />
</RelativeLayout>
```



Exit App

Do you want to exit App?