**LAB # 13**

**Android MENU**

In Android development, menus play a vital role in providing user interaction options within an application. They allow users to access various actions and settings within the app. This tutorial will guide you through the process of creating menus in Android using Java.

Step 1: Set up the Project

Before starting, ensure that you have the Android development environment set up. Create a new Android project in your preferred IDE, such as Android Studio. Choose an appropriate project name, package name, and minimum SDK version.

Step 2: Define Menu XML Resource

In Android, menus are defined using XML resources. To create a new menu resource file, follow these steps:

1. Right-click on the `res` directory in your project structure.

2. Choose "New" -> "Android Resource File."

3. Set the file name as "menu\_main.xml" (you can use any desired name).

4. Set the Resource type as "Menu."

5. Click on "OK" to create the file.

Step 3: Design the Menu

Open the `menu\_main.xml` file and define the menu items and their properties. Here's an example of a simple menu:

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item

android:id="@+id/menu\_item1"

android:title="Menu Item 1"/>

<item

android:id="@+id/menu\_item2"

android:title="Menu Item 2"/>

<item

android:id="@+id/menu\_item3"

android:title="Menu Item 3"/>

</menu>

In this example, we have defined three menu items with their respective IDs and titles.

Step 4: Inflate the Menu

To display the menu in your activity, you need to inflate it in the `onCreateOptionsMenu` method. Open your activity class and override the method as follows:

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.menu\_main, menu);

return true;

}

This code inflates the menu XML resource and adds it to the options menu.

Step 5: Handle Menu Item Selection

To perform actions when a menu item is selected, override the `onOptionsItemSelected` method. Add the following code snippet to your activity:

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

switch (id) {

case R.id.menu\_item1:

// Handle menu item 1 selection

return true;

case R.id.menu\_item2:

// Handle menu item 2 selection

return true;

case R.id.menu\_item3:

// Handle menu item 3 selection

return true;

default:

return super.onOptionsItemSelected(item);

}

}

In this example, we use a switch statement to handle different menu item selections based on their IDs. You can add your desired functionality within each case block.

Step 6: Test the Menu

Build and run your application on an Android emulator or device. When you open the activity, you should see the menu items displayed in the options menu. Upon selecting a menu item, the corresponding code in the `onOptionsItemSelected` method will execute..

Remember to clean up any resources and ensure proper memory management within your application.

Conclusion:

Menus in Android provide users with convenient access to various actions and settings. This tutorial has guided you through the process of creating menus in Android using Java. By following these steps, you can incorporate menus into your Android applications and enhance user interaction and functionality.

**Tasks:**

**Task 1**: **Create a Simple Menu**

Implement a menu in your Android application with three menu items: "Home", "Settings", and "About". Display the menu in the options menu of your activity. Handle the selection of each menu item by displaying a Toast message indicating which item was selected.

**Task 2: Add Icons to Menu Items**

Enhance the menu in your Android application by adding icons to the menu items. Choose appropriate icons from the material design icon library or create custom icons. Display the icons alongside the menu item text in the options menu. Update the **onOptionsItemSelected** method to handle the selection of each menu item and display a Toast message with the item's name and icon.

**Task 3: Create a Submenu**

Extend your existing menu by adding a submenu with two grouped items: "Share" and "Rate App". The submenu should appear when the user selects the "Options" menu item. Implement functionality to handle the selection of each submenu item. For example, when the user selects "Share," display a dialog box allowing them to share the app's link via various sharing options.