## CET3013: - Toolbar, Menu and Choice

**Week 4 Exercises: User Choice and Dialogs**

###### Objectives – What this lesson is trying to achieve.

You are trying to learn

1. Create material toolbar
2. Create option menu
3. Use Spinner and AlertDialog.Builder
4. ViewModel

This week we will be building a simple application with the toolbar, option menus, dialog box and ViewModel..

### New Project

1. Create a new project called Week4\_CET3013\_Discount.
2. Apply the default settings for the project.
3. Create the following items in the string resource files.

<string name="header">Get a free discount code by TODAY!</string>  
<string name="discount\_button">Get Discount</string>  
<string name="discount\_code">Hey, %1$s! Here is your discount code</string>  
<string name="error\_message">Please fill in all form fields</string>  
<string name="hint1">Enter User Name</string>  
<string name="hint2">Enter Email Address</string>  
<string name="menu\_clear">Clear All</string>  
<string name="about">About</string>  
<string name="about\_message">Simple Discount Application Version 1.0</string>  
  
<string-array name="region">  
 <item name="mal">Malaysia</item>  
 <item name="sin">Singapore</item>  
 <item name="chn">China</item>  
 <item name="oth">Others</item>  
</string-array>

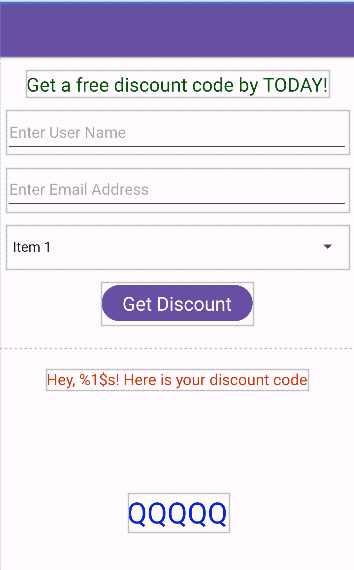
1. Apply view binding in the build.gradle (Module level)

buildFeatures **{** viewBinding = true  
**}**

1. Resync the project file.

### Layout Design

1. Design the following layout for your app.



guideline

android:id="@+id/text\_discount "

android:id="@+id/text\_message"

android:id="@+id/button\_discount "

android:id="@+id/spinner\_region "

android:id="@+id/text\_email"

android:id="@+id/text\_username"

android:id="@+id/text\_header"

1. Run the app and check it works in the emulator or physical mobile phone.

### Configure Toolbar and View Binding

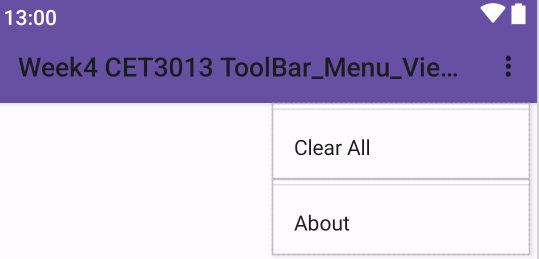
1. Add the material toolbar to the activity\_main.xml layout file.

<com.google.android.material.appbar.MaterialToolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 style="@style/Widget.MaterialComponents.Toolbar.Primary"  
 tools:ignore="MissingConstraints" />

1. Declare a view binding variable called **binding** as **ActivityMainBinding** type. Inflate the binding object using the inflate method of **ActivityMainBinding.**
2. Change the **setContentView** method’s parameter with the root attribute of binding object.
3. Use the **setSupportActionBar** to add in the material toolbar.
4. Run the project to view the toolbar. You may adjust the layout margin so that all the views are under the toolbar.

### Create a menu resource file

1. Create a menu resource folder.
2. Create a menu file inside the menu folder.
3. Design the menu file with two menu items.



1. Add in the **onCreateOptionsMenu**() and **onOptionsItemSelected**() method in the MainActivity file.
2. Inflate the menu using the **menuInflater** under the **onCreateOptionsMenu()** method.
3. Run the app to view the options menus.

### Event handling for the button

1. Implement the **onclick** method using the **setOnClickListener** method for the get discount button.

binding.buttonDiscount.setOnClickListener **{** v **->**

**}**

1. Retrieve the text value for the user name, email address and region

1. Use the **isEmpty** method to validate the user inputs. Display the Toast message if no values were captured in the app.
2. Get the discount code message from the string resource file using the **getString** method.

1. Use the **UUID.randomUUID**() to get a random UUID string value.
2. Extract the first six letters for the UUID string value by using the **substring** method. Update the **text\_discount** message.
3. Run the app and check it works in the emulator or physical mobile phone.

### Save the app states

1. After the discount code is shown, rotate the screen to landscape and back to portrait mode. The discount code is missing.
2. The simplest way to solve it is using the **onSaveInstanceState** method.

Override this method.

1. Create a companion object to represent the static members as defined in Java previously.

companion object {  
 const val DISCOUNT\_MESSAGE = "discount\_message"  
 const val DISCOUNT\_CODE = "discount\_code"  
}

1. Save the discount message and discount code to the bundle object using the **putString** method.

outState.putString(DISCOUNT\_MESSAGE, ...)

1. Update the **onCreate** method to restore the saved values.

if (savedInstanceState !=null) {

*//Get the saved values using the getString method*  
 *//update the text\_message & text\_discount* }

1. Run the app and check it works in the emulator or physical mobile phone.

### Clear settings and display the about dialog

You now need to code this functionality:

1. Override the **onOptionsItemSelected** method as follows:

override fun onOptionsItemSelected(item: MenuItem): Boolean {  
  
 when (item.*itemId*) {

R.id.*menu\_clear*->{

clearAll()

}  
  
R.id.*menu\_about*->{

showDialog()

}

else -> {  
  
}

}

}

1. Implements the **clearAll** function. Reset all text values to an empty string value.
2. Create a class (Kotlin version) and inherit the class with the **DialogFrament** super class.

class AboutDialog: DialogFragment() {}

1. Use the AlertDialog.Builder class to create the dialog object.
2. Set the dialog message with the **about\_message** from the string resource file.
3. Implement the **onclick** method for the positive and negative button.
4. In the main activity, display the dialog using the **show** method. Use the **supportFragmentManager** as default parameter for the show method.

1. Run the app and check it works in the emulator or physical mobile phone.

### ViewModel

ViewModel is under the Jetpack library component. It’s used to persist the data instead of using the **onSaveInstanceState** method in more professional way. We will implement this feature in the coming week.