## CET3013: - SharedPreferences

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

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

You are trying to learn

1. Create the SharedPreferences object
2. Create an empty activity
3. Passing and returning values from other activity
4. Change the button colour

This week we will be building a simple application with the sharedpreference and activity interaction.

### New Project

1. Create a new project called Week5\_CET3013\_ColorPicker.
2. Apply the default settings for the project.
3. Create the following items in the string resource files.

<string name="header\_text\_main">Please click the button below to choose your favorite color of the rainbow!</string>  
<string name="header\_text\_picker">Rainbow Colors</string>  
<string name="footer\_text\_picker">Click the button above which is your favorite color of the rainbow.</string>  
<string name="color\_chosen\_message">%1$s is your favorite color!</string>  
<string name="submit\_button\_text">CHOOSE COLOR</string>  
<string name="save">Save Color</string>  
<string name="about">About</string>  
  
<string name="red">RED</string>  
<string name="orange">ORANGE</string>  
<string name="yellow">YELLOW</string>  
<string name="green">GREEN</string>  
<string name="blue">BLUE</string>  
<string name="indigo">INDIGO</string>  
<string name="violet">VIOLET</string>

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

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

1. Resync the project file.

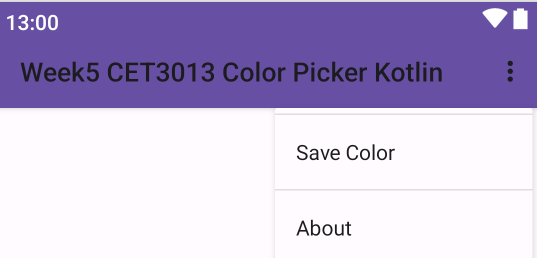
1. Add the following colors to the **colors.xml** file.

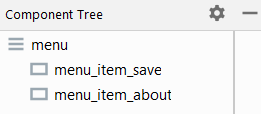
*<!--Colors of the Rainbow -->*<color name="red">#FF0000</color>  
<color name="orange">#FF7F00</color>  
<color name="yellow">#FFFF00</color>  
<color name="green">#00FF00</color>  
<color name="blue">#0000FF</color>  
<color name="indigo">#4B0082</color>  
<color name="violet">#9400D3</color>

1. Add the following styles to the **themes.xml** file.

*<!-- Style for page header on launch screen -->*<style name="header" parent="TextAppearance.AppCompat.Title">  
 <item name="android:gravity">center</item>  
 <item name="android:layout\_marginStart">24dp</item>  
 <item name="android:layout\_marginEnd">24dp</item>  
 <item name="android:layout\_marginLeft">24dp</item>  
 <item name="android:layout\_marginRight">24dp</item>  
 <item name="android:textSize">20sp</item>  
</style>  
  
*<!-- Style for page header on rainbow color selection screen -->*<style name="header.rainbows" parent="header">  
 <item name="android:textSize">22sp</item>  
 <item name="android:textAllCaps">true</item>  
</style>  
  
<style name="footer.rainbows" parent="header">  
 <item name="android:textSize">16sp</item>  
</style>  
  
*<!-- Instruction text style for choosing rainbow color -->*<style name="body" parent="TextAppearance.AppCompat.Body1">  
 <item name="android:layout\_marginTop">16dp</item>  
 <item name="android:layout\_marginBottom">8dp</item>  
 <item name="android:gravity">center</item>  
 <item name="android:layout\_gravity">center</item>  
 <item name="android:singleLine">false</item>  
 <item name="android:textSize">18sp</item>  
</style>  
  
*<!-- Style for displaying chosen rainbow color -->*<style name="color\_block" parent="TextAppearance.AppCompat.Body1">  
 <item name="android:layout\_marginTop">16dp</item>  
 <item name="android:gravity">center</item>  
 <item name="android:textSize">20sp</item>  
 <item name="android:inputType">text</item>  
 <item name="android:padding">16dp</item>  
 <item name="android:layout\_height">60dp</item>  
 <item name="android:layout\_width">340dp</item>  
</style>  
  
*<!-- Adds style to all submit buttons -->*<style name="button" parent="TextAppearance.AppCompat.Button">  
 <item name="android:minWidth">160dp</item>  
 <item name="android:gravity">center</item>  
 <item name="android:textSize">20sp</item>  
 <item name="android:layout\_margin">8dp</item>  
 <item name="android:padding">16dp</item>  
 <item name="android:textColor">@color/white</item>  
</style>  
  
*<!-- General page formatting style -->*<style name="page">  
 <item name="android:layout\_margin">8dp</item>  
 <item name="android:paddingTop">8dp</item>  
</style>

1. Create a menu folder with a menu file. Design the following menu:





### Configure Toolbar and View Binding

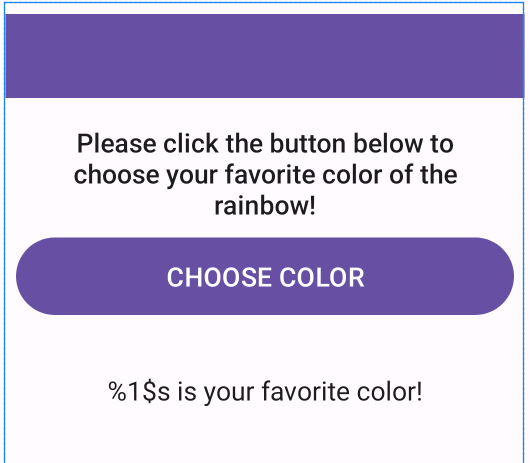
1. Add the material toolbar to the activity\_main.xml layout file.

<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.

### MainActivity Layout Design

1. Design the following layout for your app. You may use the LinearLayout or ConstraintLayout.



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

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

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

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

### Create Second Activity (ColorPickerActivity)

1. Create a new activity. **File->New->Activity->Empty Views Activity**.
2. Name the activity as **ColorPickerActivity**.
3. Use the following layout file for this activity layout.

*<?*xml version="1.0" encoding="utf-8"*?>*<ScrollView 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/ScrollView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".ColorPickerActivity"  
 tools:ignore="SpeakableTextPresentCheck">  
  
 <LinearLayout  
 style="@style/page"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <TextView  
 android:id="@+id/text\_picker\_header"  
 style="@style/header.rainbows"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/header\_text\_picker" />  
  
 <Button  
 android:id="@+id/button\_red"  
 style="@style/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/red"  
 android:text="@string/red" />  
  
 <Button  
 android:id="@+id/button\_orange"  
 style="@style/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/orange"  
 android:text="@string/orange" />  
  
 <Button  
 android:id="@+id/button\_yellow"  
 style="@style/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/yellow"  
 android:text="@string/yellow" />  
  
 <Button  
 android:id="@+id/button\_green"  
 style="@style/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/green"  
 android:text="@string/green" />  
  
 <Button  
 android:id="@+id/button\_blue"  
 style="@style/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/blue"  
 android:text="@string/blue" />  
  
 <Button  
 android:id="@+id/button\_indigo"  
 style="@style/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/indigo"  
 android:text="@string/indigo" />  
  
 <Button  
 android:id="@+id/button\_violet"  
 style="@style/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/violet"  
 android:text="@string/violet" />  
  
 <TextView  
 android:id="@+id/text\_footer"  
 style="@style/footer.rainbows"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/footer\_text\_picker" />  
  
 </LinearLayout>  
</ScrollView>



### Coding in MainActivity

1. Declare the following variables in the MainActivity.

private lateinit var binding:ActivityMainBinding  
private lateinit var startForColorResult: ActivityResultLauncher<Intent>  
private lateinit var pref: SharedPreferences  
private lateinit var colorName:String  
private lateinit var message:String  
var color = 0  
val FILE = "mypref"

1. Create the **SharedPreferences** object right after the **setSupportActionBar** method. For example:

*//create sharedprerences object*pref = getSharedPreferences(FILE, *MODE\_PRIVATE*)

1. Register the **startForColorResult** object for the activity result.

startForColorResult = registerForActivityResult<Intent, ActivityResult>(  
 ActivityResultContracts.StartActivityForResult()  
) **{** result: ActivityResult **->**

**//TODO CODE**

**}**

1. Create a companion object in the **MainActivity** class as follows:

companion object {  
 const val RAINBOW\_COLOR\_NAME = "COLOR\_NAME"  
 const val RAINBOW\_COLOR = "COLOR"  
}

1. Implement the **onclick** method for the button. Use the intent object to invoke the **ColorPickerActivity** with the **startForColorResult** object.

1. Add in the **onCreateOptionsMenu**() and **onOptionsItemSelected**() method in the MainActivity file.
2. Inflate the menu using the **menuInflater** under the **onCreateOptionsMenu()** method.
3. Provide the if condition in the **onOptionsItemSelected** method**.**

if (item.*itemId* == R.id.*menu\_item\_save*) {

**//TODO CODE**

}

1. Create the **SharedPreferences** editor called **editor**. Save the colour name and colour code to the **Sharedpreferences** editor object by using the **companion** object attribute. For example:

editor.putString(RAINBOW\_COLOR\_NAME, colorName)

Save the setting using the **commit** and **apply** methods. Display an appropriate toast message.

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

### Coding in ColorPicker Activity

1. Implement the **Views.OnClickListerner** interface in the **ColorPickerActivity**.

1. Override the **onClick** method.

override fun onClick(v: View?) {  
 **//TODO CODE**

}

1. Create the view-binding variable in this activity.
2. Implement the **setOnClickListener** method for each button (seven buttons) in the **onCreate** method.
3. Create the following customised function to change and return back the color details.

private fun changeColor(colorName: String, color: Int) {  
 *//Return back the user selection to the MainActivity* val resultIntent = Intent()  
 resultIntent.putExtra(MainActivity.RAINBOW\_COLOR\_NAME, colorName)  
 resultIntent.putExtra(MainActivity.RAINBOW\_COLOR, color)

*//set the intent and pass back to MainActivity* setResult(*RESULT\_OK*, resultIntent)

finish() *//go back to MainActivity*}

1. Referring back part 2, provide a **when** condition block in the **onclick** method.

when (v!!.*id*) {

}

Invoke the **changeColor** function created in part 5 for each triggered button and pass necessary parameters to the function accordingly.

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

### Return result to MainActivity

1. Back to the MainActivity and locate the following code for result activity object.

startForColorResult = registerForActivityResult<Intent, ActivityResult>(  
 ActivityResultContracts.StartActivityForResult()  
) **{** result: ActivityResult **->**

**//TODO CODE**

**}**

1. Under the **TODO Code** section, write the following code:

if (result.*resultCode* == *RESULT\_OK*) {  
 val intent = result.*data  
  
 //Recover the returned data* colorName = intent!!.getStringExtra(RAINBOW\_COLOR\_NAME)!!  
 color = intent.getIntExtra(RAINBOW\_COLOR, R.color.*white*)  
  
 *//form the favorite color message* message = getString(R.string.*color\_chosen\_message*, colorName)  
  
 *//display the message to the* binding.textResult.*text* = message  
 binding.textResult.setBackgroundColor(ContextCompat.getColor(this, color)  
 )  
}

1. Create a customised function called **reloadSetting**.

private fun reloadSetting() {

**//TODO CODE**

}

1. Inside the **reloadSetting** function, try to recover the **colour name** and **colour code** from the **SharedPreferences** object. Update the text result value and reapply the saved colour.
2. Called the **reloadSetting** function right above the following code:

***//recover the SharedPreferences object values*reloadSetting()**

startForColorResult = registerForActivityResult<Intent, ActivityResult>(  
 ActivityResultContracts.StartActivityForResult()  
) **{** result: ActivityResult **->**

**}**

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

### About Dialog implementation

1. Create a Kotlin class called **AboutDialog.** Refer to the previous project and create a customised dialog with the **DialogFragment**.

1. Set the dialog title and message as follows:

Title: Color Picker App

Message: Color Picker Version 1

Programmer: Your name

1. Set no implementation for the negative and positive buttons.
2. Call the dialog under the **onOptionsItemSelected** method.

if (item.getItemId() == R.id.*menu\_item\_about*) {

}

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