Lab 3: Advanced GUI Components

(Due: Mon, Feb 17, 2020 @ 3.25pm)

Introduction

In this lab will show you how to work with Advanced GUI components. You will learn how to create Dialogs, Menus and Fragments. First, we will see how to create dialogs and display information in dialogs. Then we will learn how to create menus and add items and subitems to the menu. In Milestone 2, we will learn how to create Fragments and switch between Fragments.

Directions

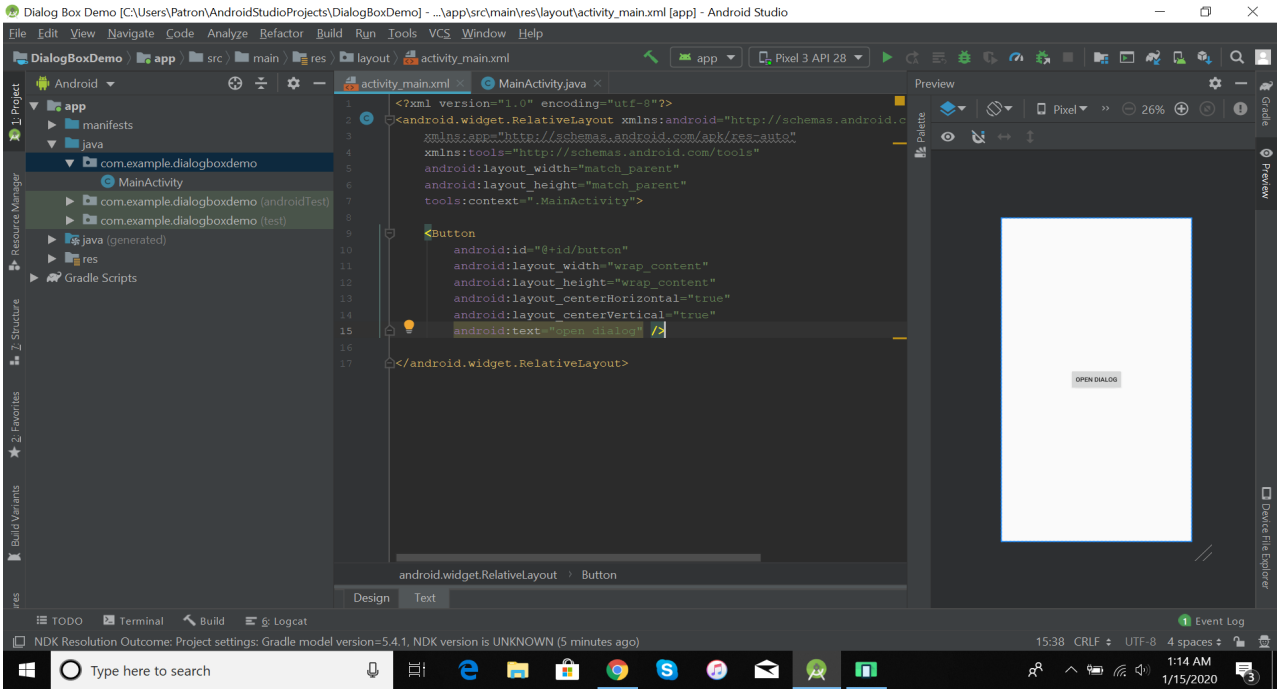
Milestone 1 - Dialogs and Menus

In this Milestone, we will be looking at creating Dialogs and Menus. We will create a dialog and add some information in it. Later, we will also create menus and add items and subitems to it.

1. First, we are going to see how to use dialogs in Android

2. As you did in Lab 1, clone the Lab 3 Milestone 1 repository by selecting File > New > Project from Version Control > Git
3. Then click the following link and accept the invitation: https://classroom.github.com/a/jW5xrVcS
4. You'll get a link in the form https://github.com/CS-407-Spring2020/lab-3-milestone-1-yourgithubnamehere

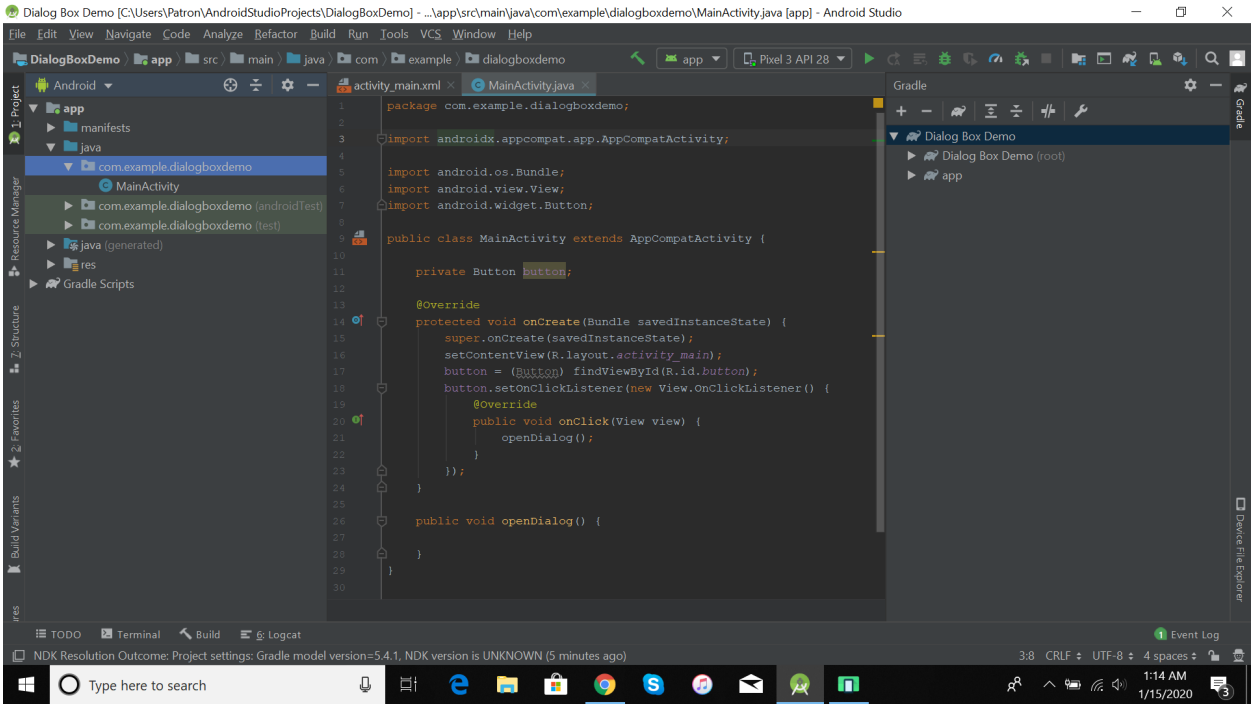
5. Set up a button at the center of the app as shown below. On pressing the button, we'll display a dialog box



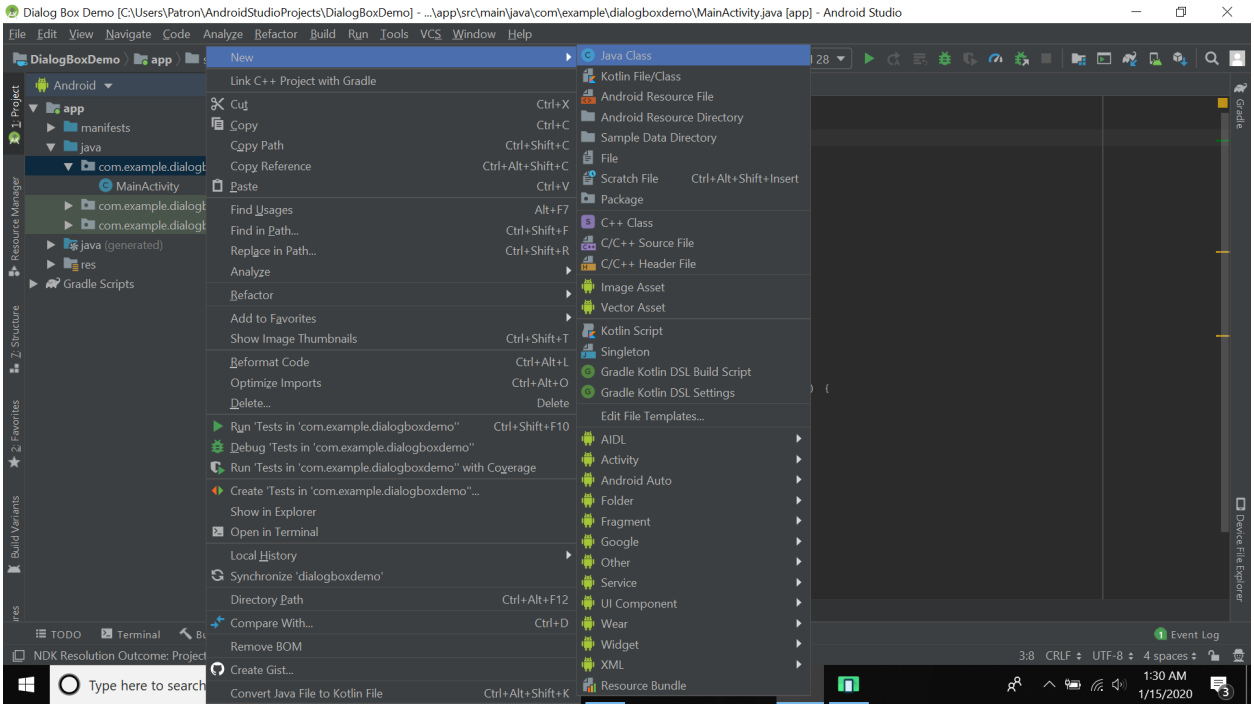
6. Next, switch to MainActivity.java. We will create an onClickListener() for the button in the onCreate() method.

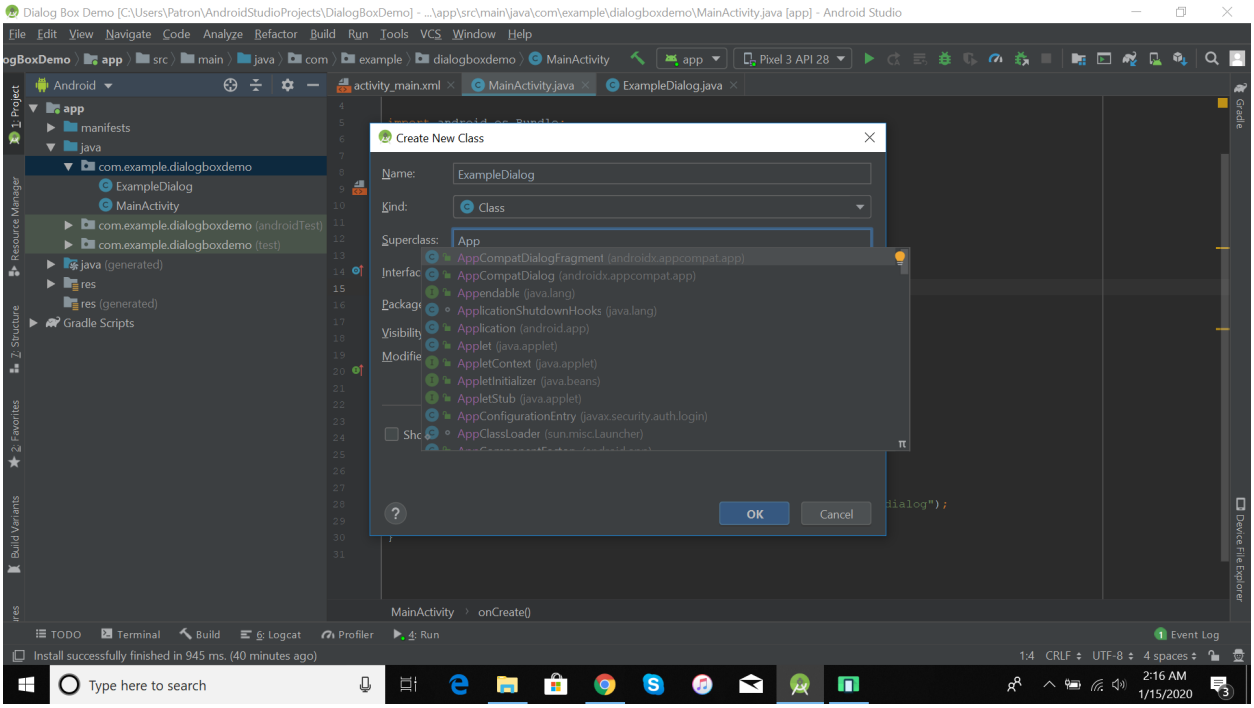
   When a user presses the button, the button object receives an "on-click" event. In order to respond to this event, in the the "MainActivity.java" file, we create an event handler called onClickListener() for the button. This is added in the onCreate() method.
   You will notice that the onClickListener() contains an onClick() method. Recall from Lab-2 that the "onClick" method is used to specify the action that needs to be taken when the button is pressed.

7. Create a new method openDialog() which will create the dialog. We will define the body of this method in a moment. Call this openDialog() method within the onClick method we created in the previous step. Your code after step-3 and step-4 should look something like this:

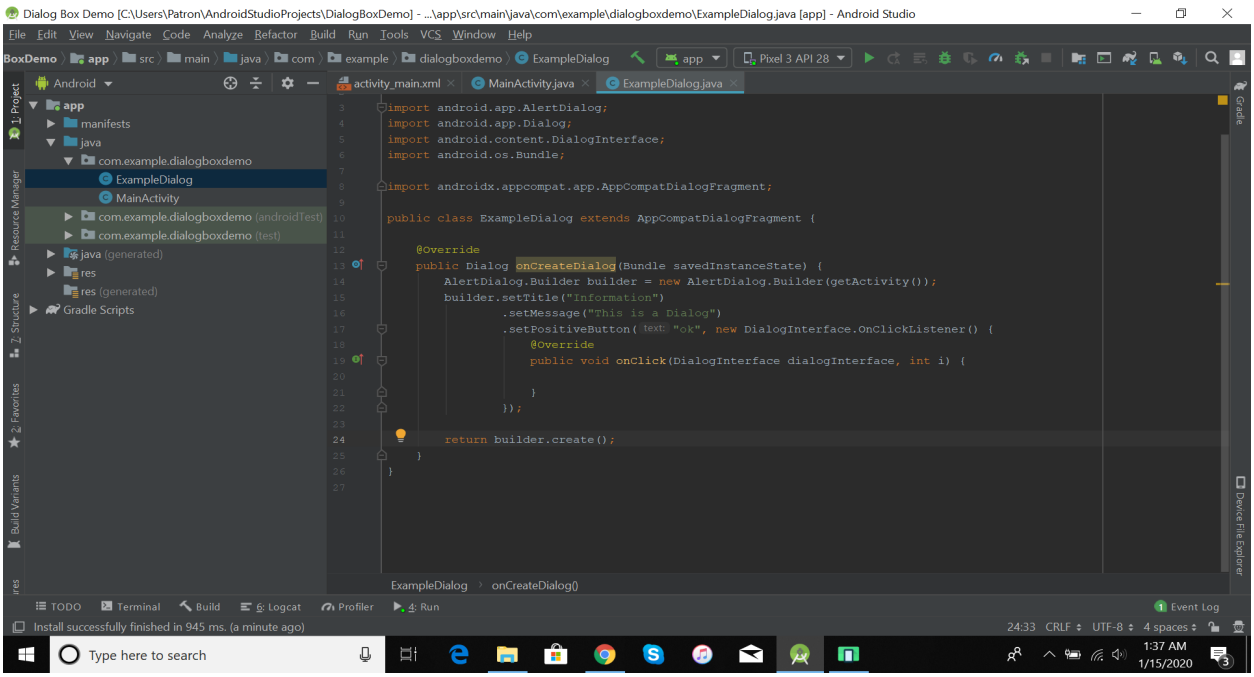


8. Now, we will create a new class that will extend the AppCompatDialogFragment.
   To create a new class, right-click on com.example.you-project-name in the “Project” tab on the left side of Android studio. Select New → Java Class. Name the class and under superclass search AppCompatDialogFragment. This is shown in the following two snapshots:

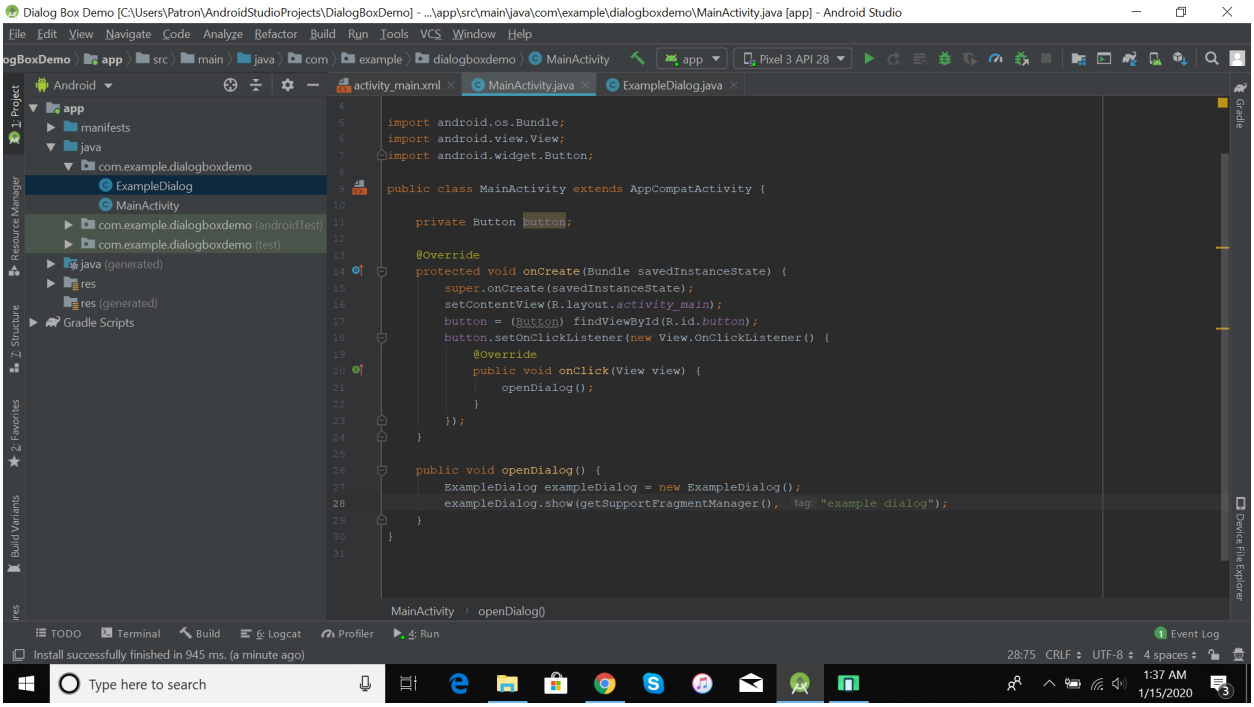




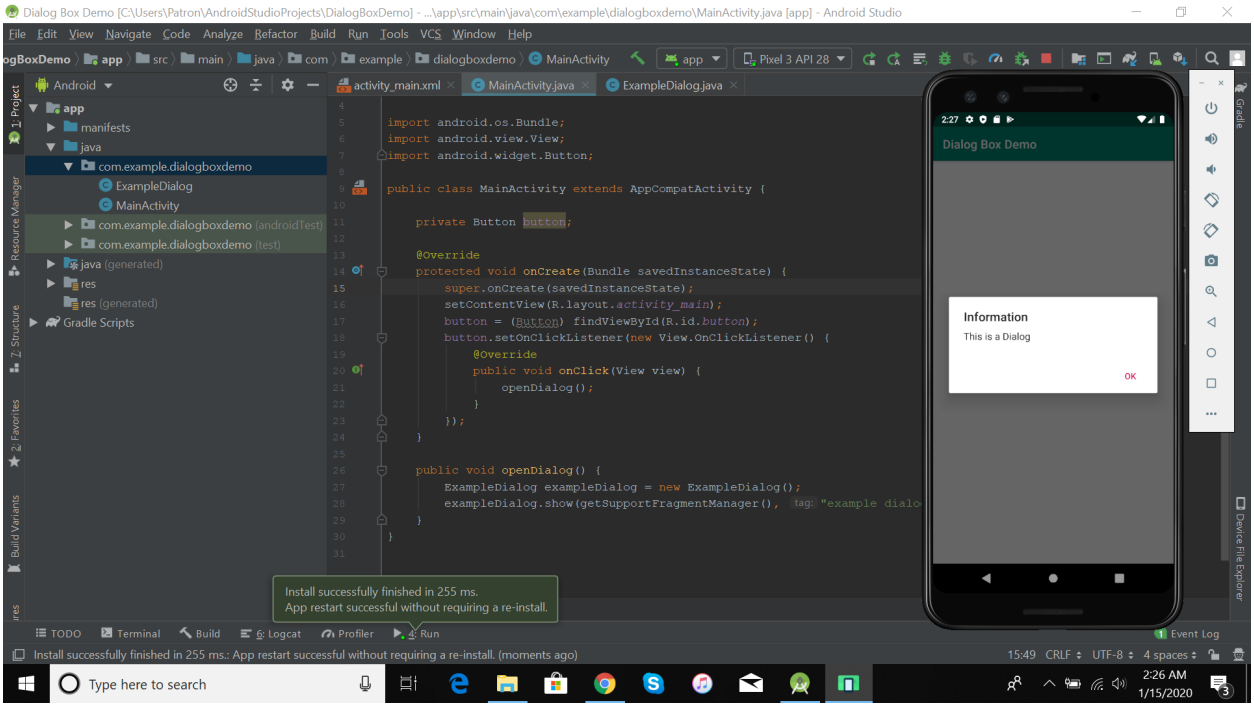
9. Now, we override the onCreateDialog() method within this new class. Create a new AlertDialog instance within this onCreateDialog() method. We set a title and a message for the dialog box. We also set a button that the user can click to make the dialog box disappear. For this example, we leave the onClick() method for this button empty. Here is a snapshot of how the code should look:
Note: Make sure that you have the import statements added correctly.



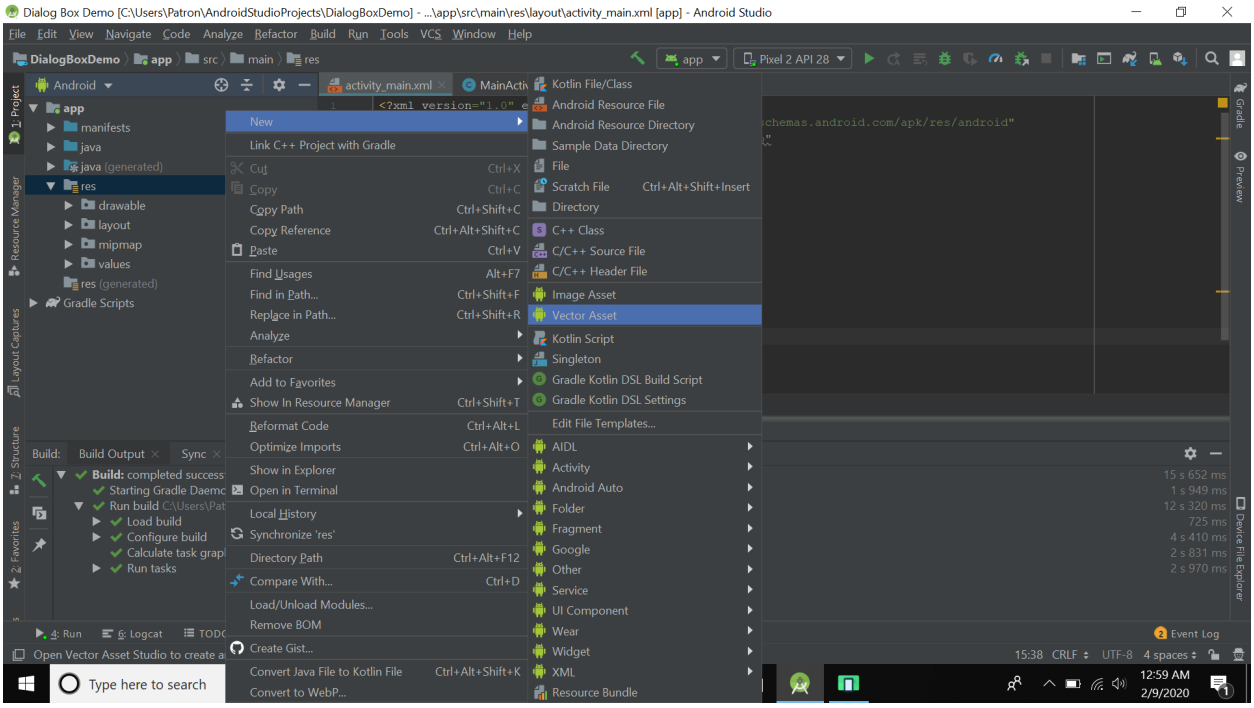
10. Switch to MainActivity.java and in the openDialog() method we created earlier, create an instance of our ExampleDialog class. Now show the dialog.



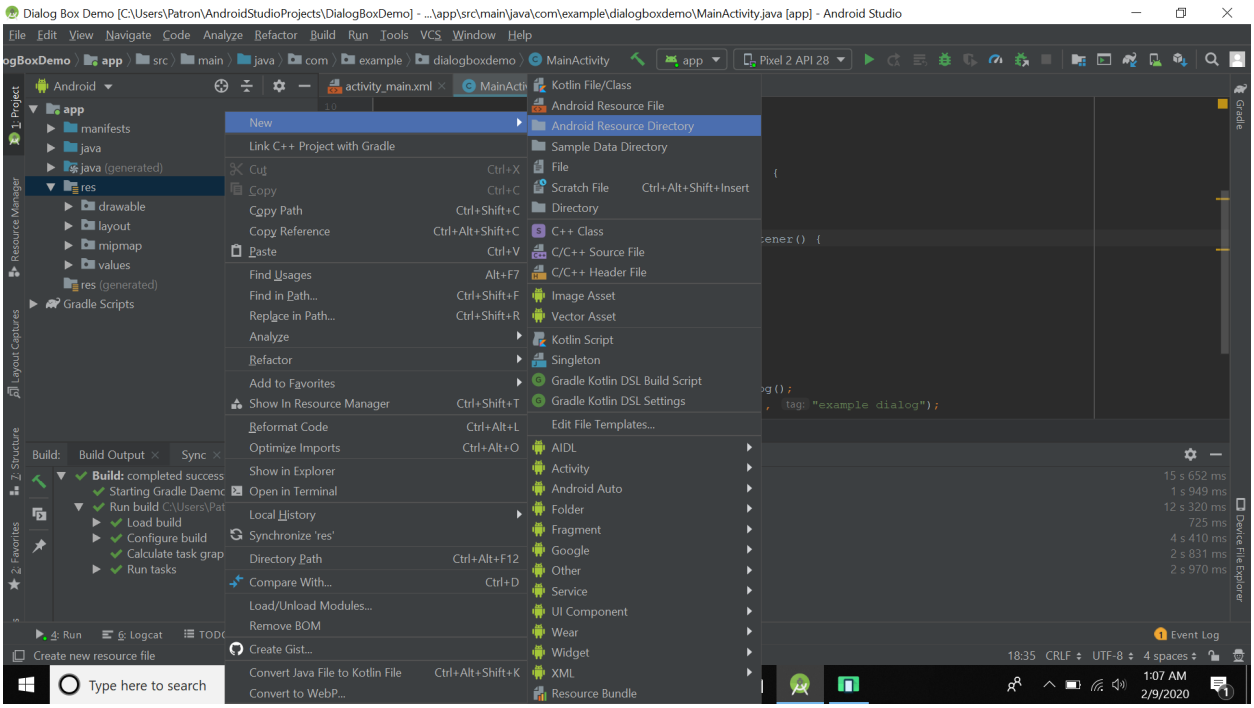
11. Your app should look like this:

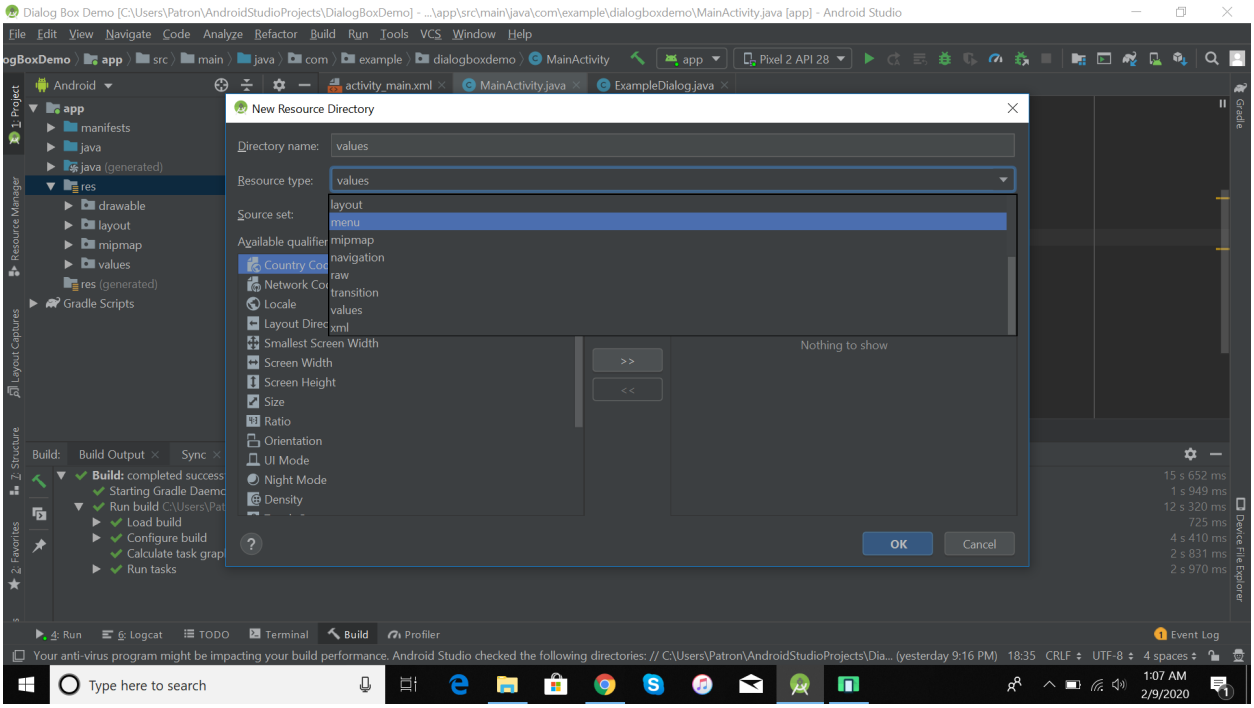


12. Now we will see how to create menus for your app.
13. First, we will add a vector asset to the app. Right click res -> New -> Vector Asset.

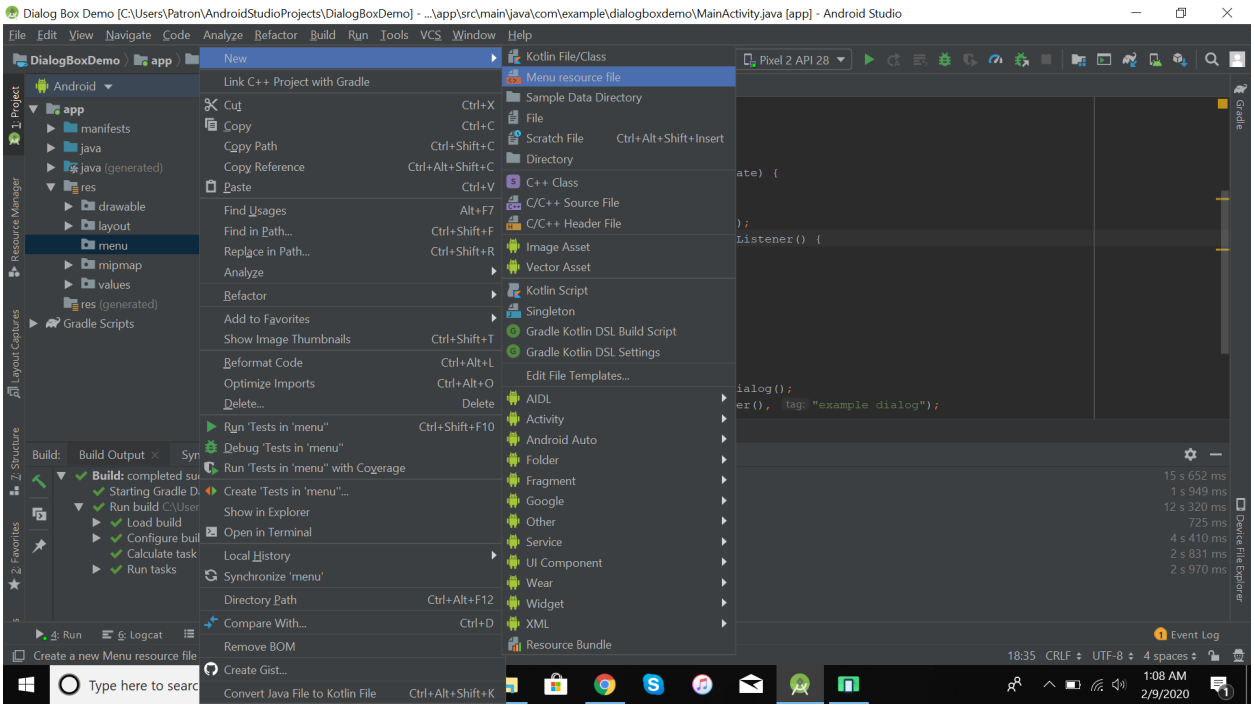


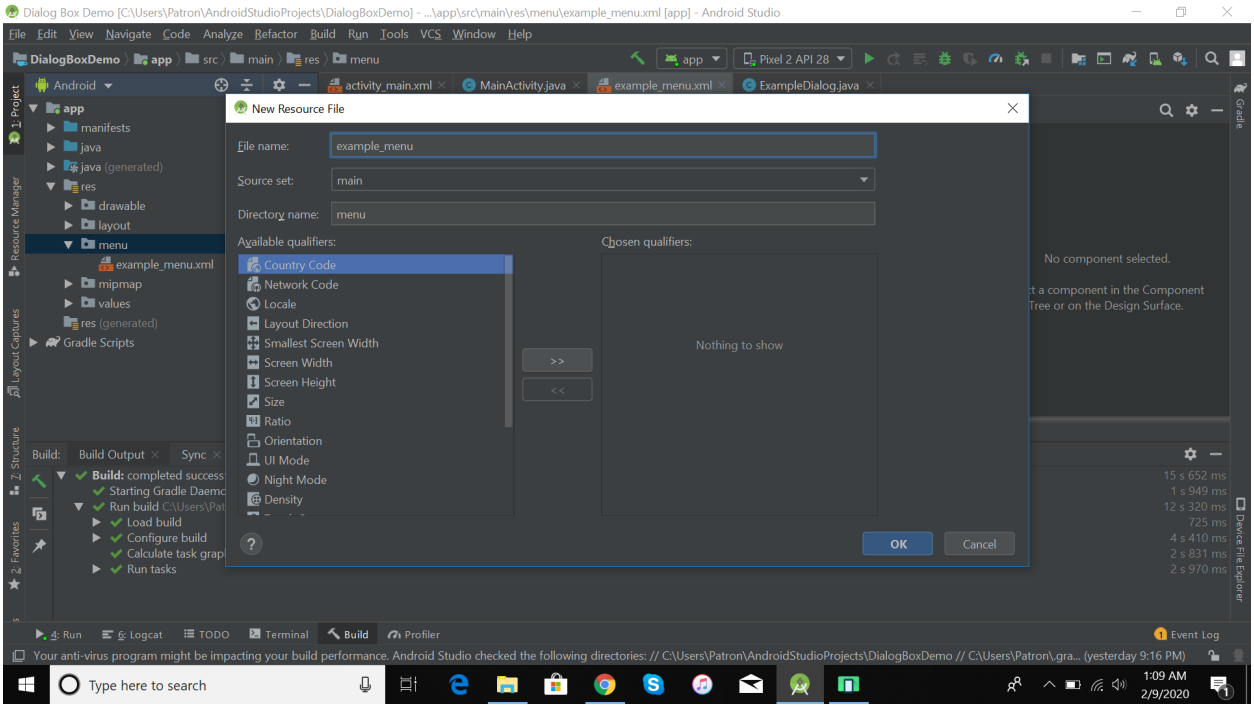
14. Select ‘Clip Art’ and choose any icon you wish. Change the name to ‘ic_icon’ and click next.
15. Now, we will create an Android Resource Directory for menus. Right Click res -> New -> Android Resource Directory. Under the ‘Resource type’ select menu.



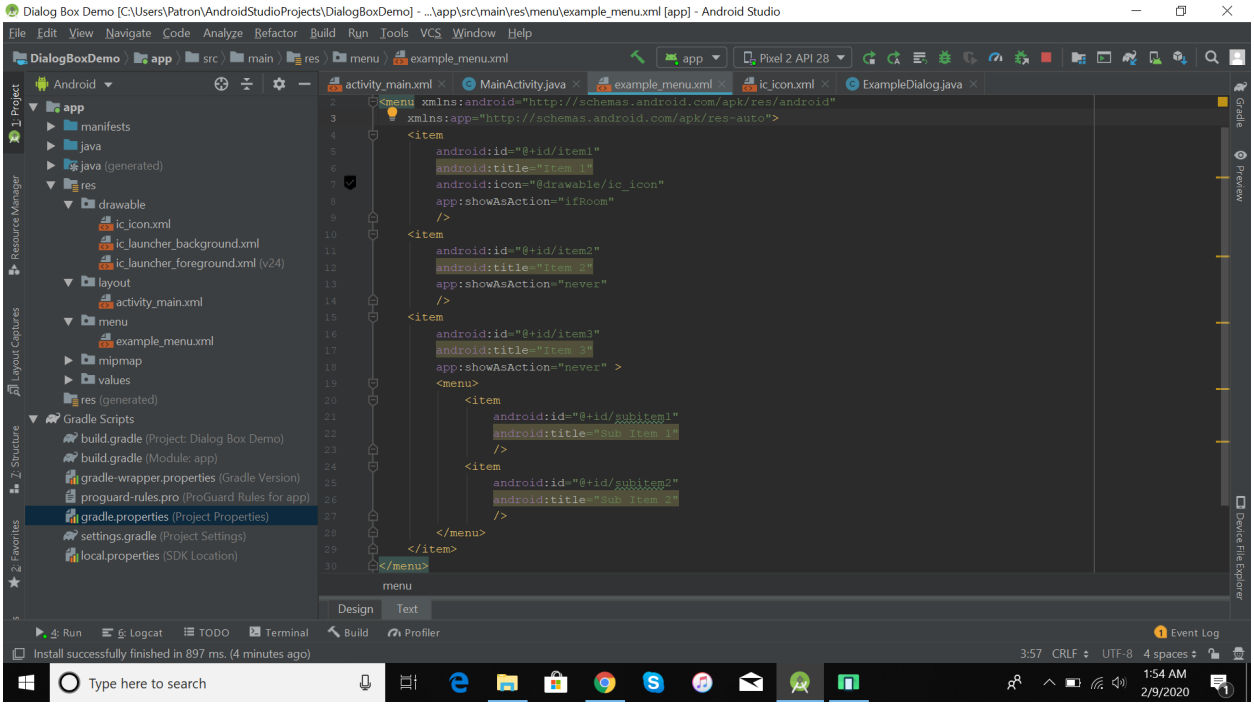


16. Now right click on ‘menu’ -> New -> Menu Resource File. Give it a suitable name.





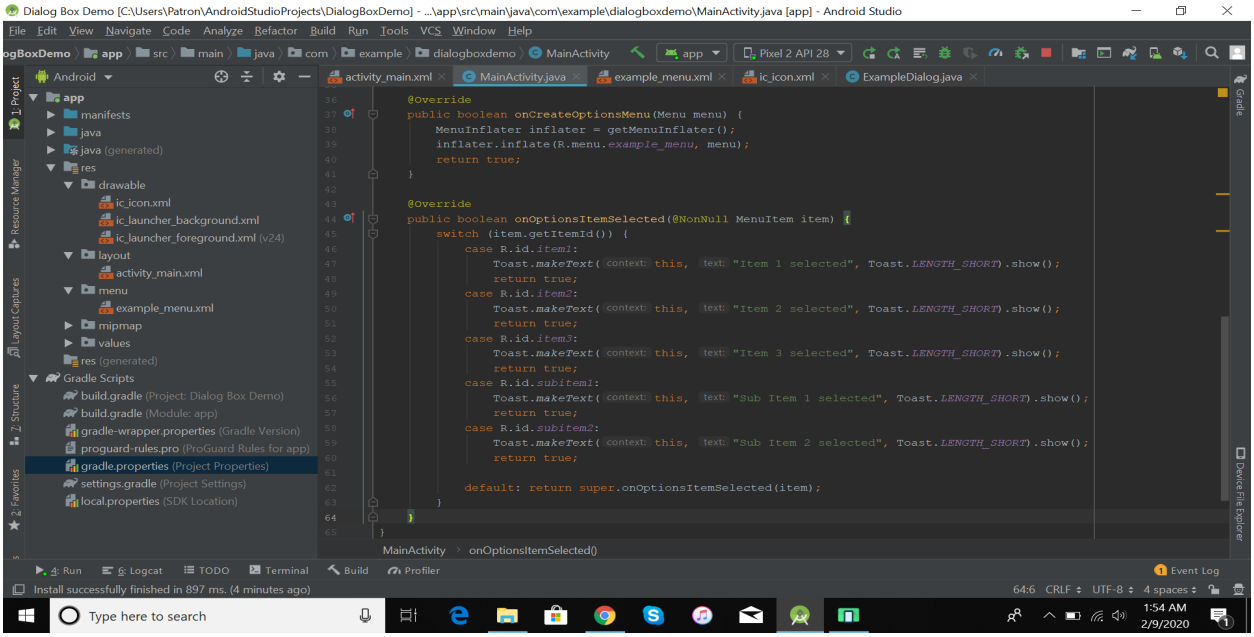
17. Now in the example_menu.xml file we just created, we will add items to our menu. We can assign the ic_icon we created in step 11 to an item. Assign it to item 1 using the ‘android:icon’ attribute. You can assign as many items and sub-items to the menu as you like. Check out the code below:



18. Now, we will assign actions for each menu item in MainActivity.java.

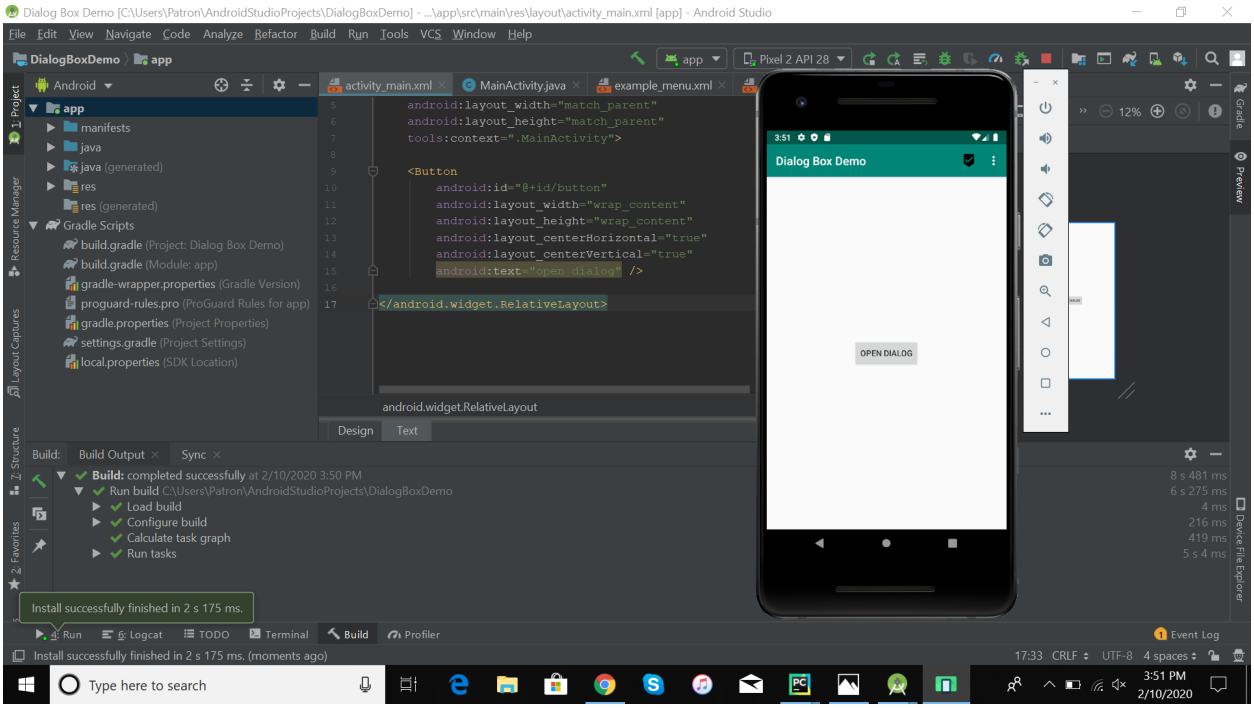
19. First, override the function onCreateOptionsMenu. In this function, we will create a MenuInflater to get our menu. You can refer to the code in the screenshot below.

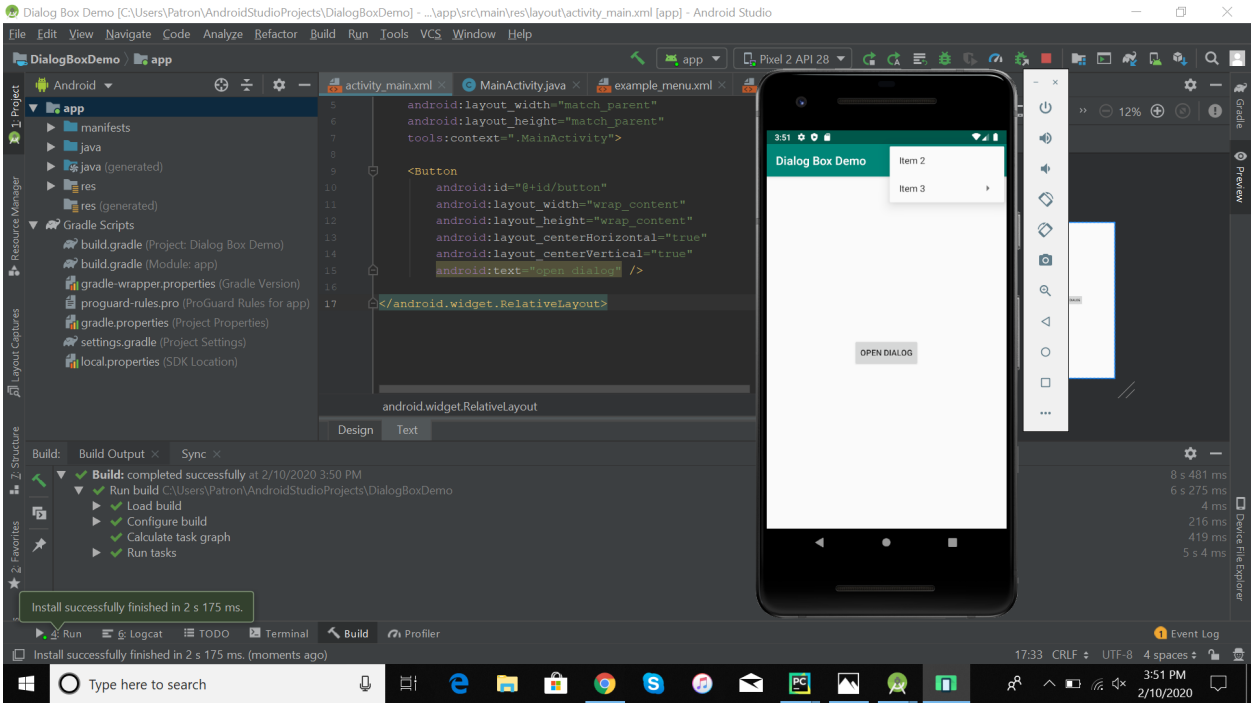
20. Next, override the function onOptionsItemSelected. In this function we will create a switch() to see which menu item was pressed. We will display a different toast for each menu item so we can check that the app is working properly. Refer to the following code:

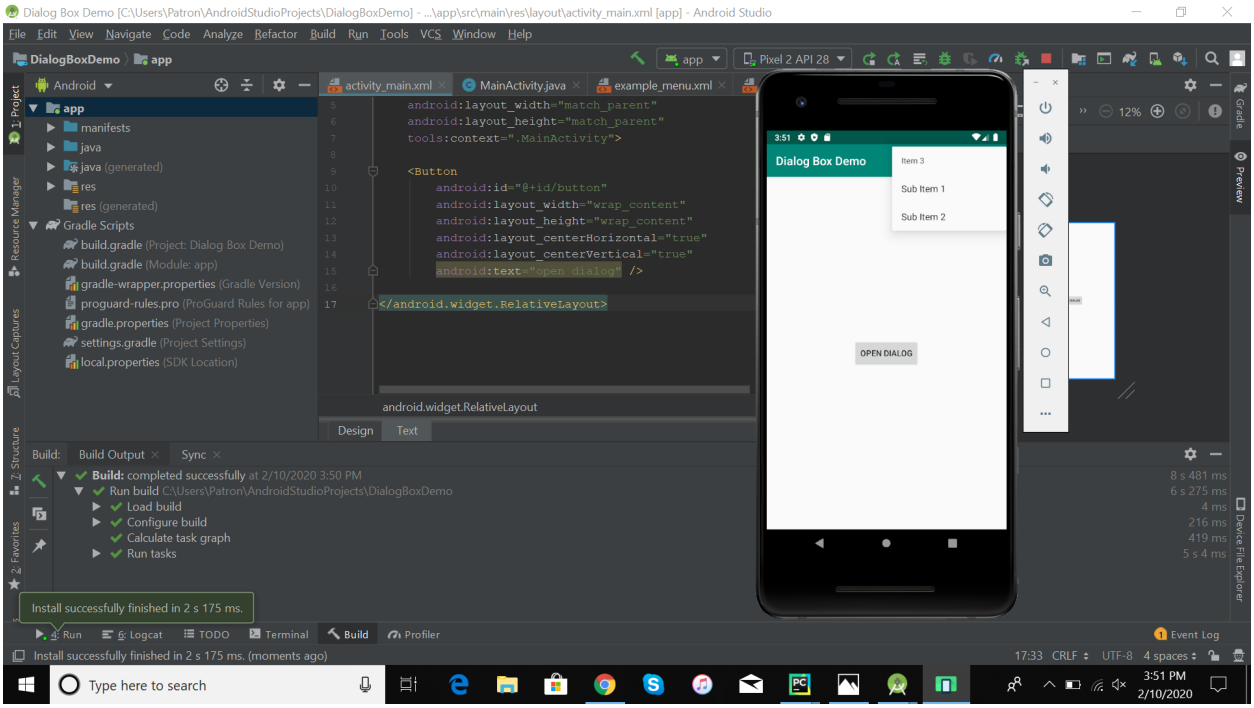


DELIVERABLES:

- Dialog box should be functioning correctly.
- The menu items are displaying the corresponding toasts correctly.
- Commit and push changes to the github classroom.


Dialog Box Demo [C:\Users\Patron\AndroidStudioProjects\DialogBoxDemo] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
DialogBoxDemo app
app
Pixel 2 API 28
activity_main.xml
MainActivity.java
example_menu.xml
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="open dialog" />
</android.widget.RelativeLayout>
android.widget.RelativeLayout
Design
Text
Dialog Box Demo
OPEN DIALOG
Build: completed successfully at 2/10/2020 3:50 PM
Run build C:\Users\Patron\AndroidStudioProjects\DialogBoxDemo
Load build
Configure build
Calculate task graph
Run tasks
Install successfully finished in 2 s 175 ms.
Install successfully finished in 2 s 175 ms. (moments ago)
Type here to search
3:51 PM
2/10/2020


Dialog Box Demo [C:\Users\Patron\AndroidStudioProjects\DialogBoxDemo] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
DialogBoxDemo app
app
Pixel 2 API 28
activity_main.xml
MainActivity.java
example_menu.xml
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="open dialog" />
</android.widget.RelativeLayout>
android.widget.RelativeLayout
Design
Text
Dialog Box Demo
Item 2
Item 3
OPEN DIALOG
Build: completed successfully at 2/10/2020 3:50 PM
Run build C:\Users\Patron\AndroidStudioProjects\DialogBoxDemo
Load build
Configure build
Calculate task graph
Run tasks
Install successfully finished in 2 s 175 ms.
Install successfully finished in 2 s 175 ms. (moments ago)
Type here to search
3:51 PM
2/10/2020


Dialog Box Demo [C:\Users\Patron\AndroidStudioProjects\DialogBoxDemo] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
DialogBoxDemo app
app
Pixel 2 API 28
Android
app
manifests
java
java (generated)
res
res (generated)
Gradle Scripts
build.gradle (Project: Dialog Box Demo)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
activity_main.xml
MainActivity.java
example_menu.xml
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="open dialog" />
</android.widget.RelativeLayout>
android.widget.RelativeLayout
Design
Text
Dialog Box Demo
Item 3
Sub Item 1
Sub Item 2
OPEN DIALOG
Build:
Build Output
Sync
Build: completed successfully at 2/10/2020 3:50 PM
Run build C:\Users\Patron\AndroidStudioProjects\DialogBoxDemo
Load build
Configure build
Calculate task graph
Run tasks
Install successfully finished in 2 s 175 ms.
Run
Logcat
TODO
Terminal
Build
Profiler
Event Log
Install successfully finished in 2 s 175 ms. (moments ago)
17:33 CRLF UTF-8 4 spaces
Type here to search
3:51 PM
2/10/2020

Milestone 2:

Below is an example of Fragment's. In this example we create two Fragments and load them on the click of buttons. We display two buttons and a FrameLayout in our Activity and perform setOnClickListener event on both Buttons. On the click of First Button we replace the First Fragment and on click of Second Button we replace the Second Fragment with the layout(FrameLayout). In both Fragments we display a TextView and a Button and onclick of Button we display the name of the Fragment with the help of Toast.

Step 1: Create a new project. As you did in lab 1, clone the Lab 3 Milestone 2 repository by selecting File > New > Project from Version Control > Git. Then click the following link and accept the invitation: https://classroom.github.com/a/uafJw7Hr

You'll get the following form: https://github.com/CS-407-Spring2020/lab-3-milestone-2-yourgithubnamehere

Step 2: Open res -> layout ->activity_main.xml (or) main.xml and add following code:

```
<LinearLayout 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"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity">
<!-- display two Buttons and a FrameLayout to replace the Fragment's  -->
<Button
android:id="@+id/firstFragment"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/button_background_color"
android:text="First Fragment"
android:textColor="@color/white"
android:textSize="20sp" />

<Button
android:id="@+id/secondFragment"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:background="@color/button_background_color"
android:text="Second Fragment"
android:textColor="@color/white"
android:textSize="20sp" />

<FrameLayout
android:id="@+id/frameLayout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_marginTop="10dp" />
</LinearLayout>
```

Step 3: Open src -> package -> MainActivity.java

In this step we open MainActivity and add the code to initiate the Buttons. After that we perform a setOnClickListener event on both buttons. On the click of First Button we replace the First Fragment and on click of Second Button we replace the Second Fragment with the layout(FrameLayout). For replacing a Fragment with FrameLayout firstly we create a Fragment Manager and then begin the transaction using Fragment Transaction and finally replace the Fragment with the layout, i.e., FrameLayout.

```
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

Button firstFragment, secondFragment;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
// get the reference of Button's
firstFragment = (Button) findViewById(R.id.firstFragment);
secondFragment = (Button) findViewById(R.id.secondFragment);

// perform setOnClickListener event on First Button
firstFragment.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// load First Fragment
loadFragment(new FirstFragment());
}
});
// perform setOnClickListener event on Second Button
secondFragment.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// load Second Fragment
loadFragment(new SecondFragment());
}
});

}

private void loadFragment(Fragment fragment) {
// create a FragmentManager
FragmentManager fm = getFragmentManager();
// create a FragmentTransaction to begin the transaction and replace the
Fragment
FragmentTransaction fragmentTransaction = fm.beginTransaction();
// replace the FrameLayout with new Fragment
fragmentTransaction.replace(R.id.frameLayout, fragment);
fragmentTransaction.commit(); // save the changes
}
}
```

Step 4: Now we need 2 fragments and 2 xml layouts. So create two fragments by right click on your package folder and create classes and name them as FirstFragment and SecondFragment and add the following code respectively.

FirstFragment.class

In this Fragment firstly we inflate the layout and get the reference of Button. After that we perform a setOnClickListener event on Button so whenever a user clicks on the button a message “First Fragment“ is displayed on the screen by using a Toast.

```
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

public class FirstFragment extends Fragment {


View view;
Button firstButton;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
// Inflate the layout for this fragment
view = inflater.inflate(R.layout.fragment_first, container, false);
// get the reference of Button
firstButton = (Button) view.findViewById(R.id.firstButton);
// perform setOnClickListener on first Button
firstButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// display a message by using a Toast
Toast.makeText(getActivity(), "First Fragment", Toast.LENGTH_LONG).show();
}
});
return view;
}
}
```

SecondFragment.class

In this Fragment firstly we inflate the layout and get the reference of Button. After that we perform a setOnClickListener event on Button so whenever a user clicks on the button a message “Second Fragment“ is displayed on the screen by using a Toast.

```
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;


public class SecondFragment extends Fragment {

View view;
Button secondButton;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
// Inflate the layout for this fragment
view = inflater.inflate(R.layout.fragment_second, container, false);
// get the reference of Button
secondButton = (Button) view.findViewById(R.id.secondButton);
// perform setOnClickListener on second Button
secondButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// display a message by using a Toast
Toast.makeText(getActivity(), "Second Fragment", Toast.LENGTH_LONG).show();
}
});
return view;
}
}
```

Step 5: Now create 2 xml layouts by right clicking on res/layout -> New -> Layout Resource File and name them as fragment_first and fragment_second and add the following code in respective files.

Note: In some Android versions, the xml layouts are created automatically when the fragment files are added in Step 4.

Here we will design the basic simple UI by using TextView and Button in both xml's.

fragment_first.xml

```
<RelativeLayout 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="com.abhiandroid.fragmentexample.FirstFragment">
```

```
<!--TextView and Button displayed in First Fragment -->
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="100dp"
android:text="This is First Fragment"
android:textColor="@color/black"
android:textSize="25sp" />

<Button
android:id="@+id/firstButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:layout_marginLeft="20dp"
android:layout_marginRight="20dp"
android:background="@color/green"
android:text="First Fragment"
android:textColor="@color/white"
android:textSize="20sp"
android:textStyle="bold" />
</RelativeLayout>
```

fragment_second.xml

```
<RelativeLayout 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="com.abhiandroid.fragmentexample.SecondFragment">

<!--TextView and Button displayed in Second Fragment -->
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="100dp"
android:text="This is Second Fragment"
android:textColor="@color/black"
android:textSize="25sp" />

<Button
android:id="@+id/secondButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:layout_marginLeft="20dp"
android:layout_marginRight="20dp"
```

```
android:background="@color/green"
android:text="Second Fragment"
android:textColor="@color/white"
android:textSize="20sp"
android:textStyle="bold" />

</RelativeLayout>
```

Step 6: Open res ->values ->colors.xml

In this step we define the color's that are used in our xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- colors used in our project -->
<color name="black">#000</color>
<color name="green">#0f0</color>
<color name="white">#fff</color>
<color name="button_background_color">#925</color>
</resources>
```

Step 7: Open res → values → styles.xml

A style resource defines the format and looks for a UI.

Add the following changes to this file:

```
<resources>

    <!-- Base application theme. -->

    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

        <!-- Customize your theme here. -->

        <item name="colorPrimary">@color/black</item>

        <item name="colorPrimaryDark">@color/green</item>

        <item name="colorAccent">@color/white</item>

    </style>

</resources>
```

Step 8: dimens.xml File

Some project templates may not include a dimens.xml file which lets you define a custom definition for the dimension values that you define in your project. For ex: you can define a vertical margin to be of 8dp.

If this file is not available in the res → values folder, do the following:

- Right click res → values →New → Values Resources File
- Name the file as: dimens.xml and click Ok

- Edit the dimens.xml file as follows:

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

<resources>

    <dimen name="activity_vertical_margin">8dp</dimen>

    <dimen name="activity_horizontal_margin">8dp</dimen>

</resources>
```

Step 7: Open AndroidManifest.xml

In this step we show the Android Manifest file in which do nothing because we need only one Activity i.e MainActivity which is already defined in it. In our project we create two Fragment's but we don't need to define the Fragment in manifest because Fragment is a part of an Activity.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.abhiandroid.fragmentexample" >

<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".MainActivity"
android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

Step 8: Now run the App and you will have two buttons. Clicking on the first button shows First Fragment and on click of Second Button shows the Second Fragment which is actually replacing layout(FrameLayout).

Common Issues:

Some of the common issues while implementing the Milestones have been added here. Please look through these if needed.

1. In ***activity_main.xml***, Make sure the line: `tools:context=".MainActivity">`
   Reflects your app package. For example, you may have to change it to:
   `tools:context="com.example.`**your-project-name**`.MainActivity">`

2. In ***MainActivity.java***, if the following line:
   `import android.support.v7.app.AppCompatActivity;`
   gives you an error, look into your ***gradle.properties file*** *in the* Gradle Scripts folder, and check for the following two lines:
   ```
   android.useAndroidX=true
   android.enableJetifier=true
   ```
   If you see the above two lines, then in your ***MainActivity.java*** file,
   Instead of:
   `import android.support.v7.app.AppCompatActivity;`
   Use this:
   `import androidx.appcompat.app.AppCompatActivity;`

3. Ensure that ***MainActivity.java, FirstFragment.java*** and ***SecondFragment.java*** files, import your app package at the very beginning. For example, it may need to be changed to:
   `package com.example.`**your-project-name**`;`

4. In the **activity_main.xml, fragment_first.xml and fragment_second.xml,** ensure that you are using the right app package the following line may not reflect the right package in your case:
   Instead of:
   `tools:context="com.abhiandroid.fragmentexample.SecondFragment">`
   Use this:
   `tools:context="com.`**your-project-name**`.SecondFragment">`

DELIVERABLES:

- Show the correct functioning of the fragments to a peer mentor or TA
- The app should look presentable
- Switching between fragments should work correctly
- Commit and push the project to github classroom

Conclusion:

Great job on getting through this lab. There's a lot more things you can do using these components in Android Studio. Explore these components as much as you can because it will be a huge help for your projects.

References:

https://abhiandroid.com/ui/fragment