Birzeit University
Faculty of Engineering and Technology
Electrical and Computer Engineering Department
Advance Computer Systems Engineering Lab ENCS515

EXP. No. 7. Fragments

1. Pre-Lab: Click on the link and watch the video.

2. Objectives

- Introduce to new concept in Android Applications.
- Simplify code and reduce latency time.

3. Introduction

Tablet have larger display screen than small devices, so they can support multiple UI panes/user behavior at the same time.

A Fragment represents a behavior or a portion of user interface in an Activity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities as shown in Figure 7.1.

Let us take Fragments application as an example application. It uses two activities, the first one shows the titles of plays or characters and allows user to select one title as shown in Figure 7.2. The second one shows a quote from selected title as shown in the Figure 7.3.

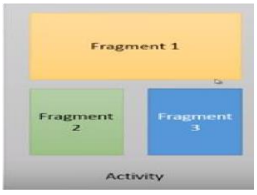

Figure 7.1 Multiple Fragments in the Same Activity

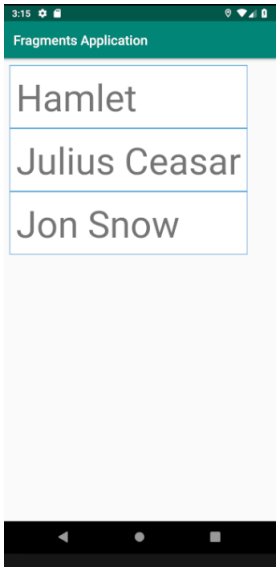

Figure 7.2 First activity shows three titles

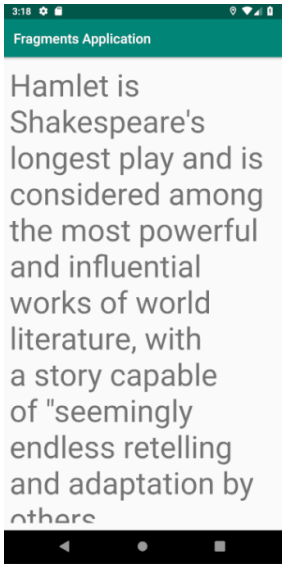

Figure 7.3 Second activity shows selected title

Figure 7.4 Two Fragments in One Activity

This interface is reasonable for a phone or small screen device, but it is inefficient on large device. So, it is better to use two cooperating layout units on one screen as shown in Figure 7.4. In Figure 7.4, the activity contains two fragments: first one shows three titles and second one shows selected title information.

3.1. There are two general ways to add fragments to an activity:

- Declare it statically in the activity layout file

  Attach the fragment inside the activity through XML using <fragment> tag
  Example: to add FirstFragment in the main activity we can use the following XML code in the layout of the main activity.

- Add it programmatically using the fragment manager

  We will add the fragment to main activity using java. Every activity has its own Fragment Manager, it maintains references to all fragments inside the activity. You can get access to Fragment Manager throw getFragmentManager() method . You can use findFragmentById() or findFragmentByTag() methods to get reference to a particular fragment. Changes to UI in terms of adding, removing and replacing fragments are conducted as Fragment Transactions. You must begin a transaction, then add, remove or replace a fragment and finally you have to commit the transaction to see the effect on your activity.

3.2. Inter fragment communication in android:

How to communicate between AFragment and BFragment without making a decencies between the two fragments?

- Define an Interface in the BFragment class and implement it within the Activity.

- Implement the Interface: The activity that hosts AFragment and BFragment must implement the interface defined in the BFragment class.
- The AFragment captures the interface implementation during its onAttach() or onActivityCreated() lifecycle methods and can then call the Interface methods in order to communicate with the Activity.

3.3. Fragment Transactions:

If you want to add, replace, remove, attach or detach a fragment, you have to use the transactions by the following steps:

- get a new object from your fragment.
- beging the transaction by using beginTransaction method in the fragment manager.
- add/remove/replace/attach/detach the fragment.
- commit the transaction

4. Procedure

In this lab you will design two different applications, the first is to add two fragments and make a communication between the fragments, where the second one is to how to add, remove, attach, detach and replace fragments dynamically.

4.1. First Application

In this application we will have two fragments (FirstFragment and SecondFragment) where the FirstFragment will send a string to the SecondFragment containing the number of times the button in the first fragment is clicked and the SecondFragment will display the string in a text view as shown in Figure 7.10. Create new project and change the application name to “Fragments Communication Application”.

- Adding fragments

Add two fragments as shown in Figure 7.5 and call them FirstFragment and SecondFragment as shown in Figure 7.6 and uncheck the *include fragment factory methods?* and *include interface callbacks?* Check boxes then click on finish.

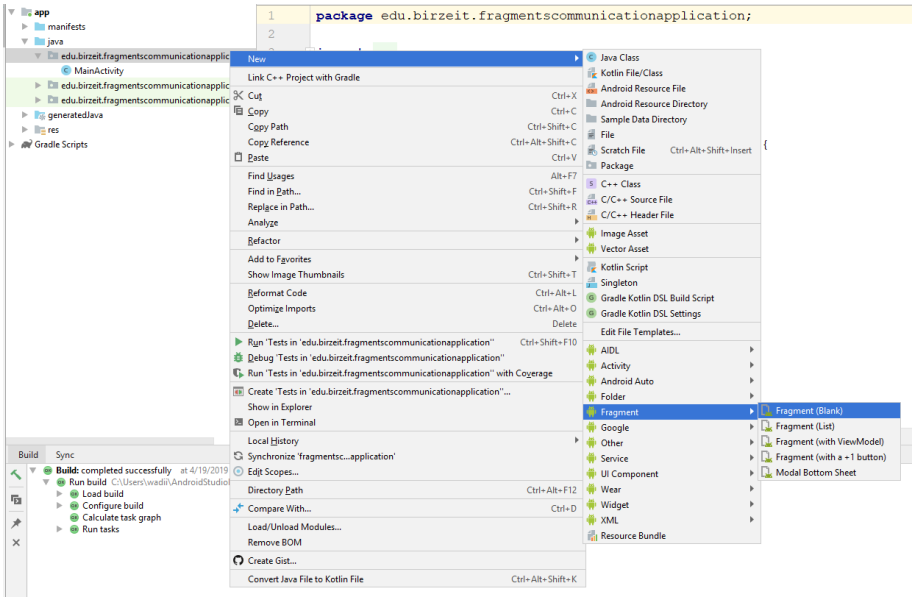

Figure 7.5 Adding Frsgment Screen

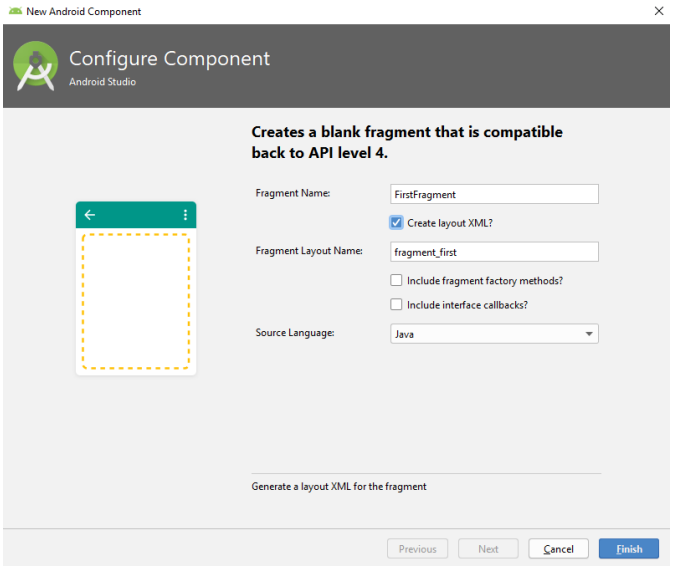

Figure 7.6 Fragment Android Component

When creating fragments, this will create a java class for the fragment and a layout as when creating an activity, after creating the fragments the project folders should look as in Figure 7.7.

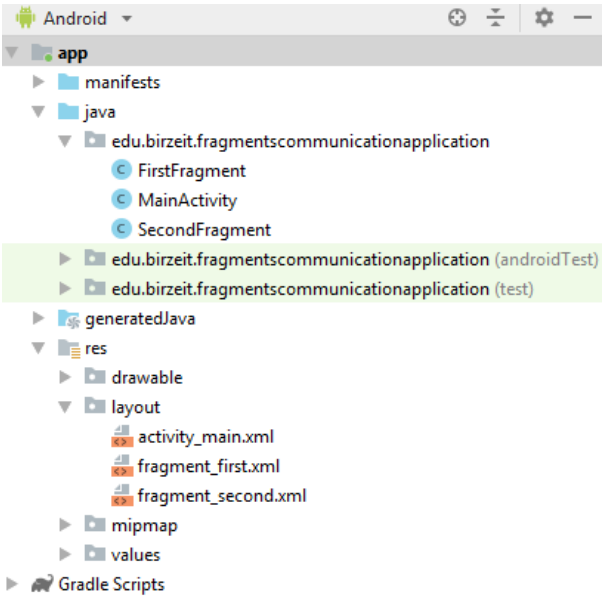

Figure 7.7 project classes and layouts

- Design the fragments layouts

  - In the fragment_first layout start by adding a button and making the root layout height to wrap_content. You also can change the background color of the root layout to dark holo_green_dark by adding int following code to the root layout xml code.

```
android:background="@android:color/holo_green_dark"
```

  - In the fragment_second layout start by adding a textview and making the root layout height to wrap_content. You also can change the background color of the root layout to dark holo_blue_light by adding int following code to the root layout xml code.

```
android:background="@android:color/holo_blue_light"
```

the fragments should look as in Figure 7.8.a for the first fragment and Figure 7.8 for the second fragment.

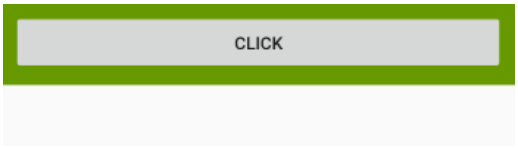

Figure 7.8.a First Fragment layout with button

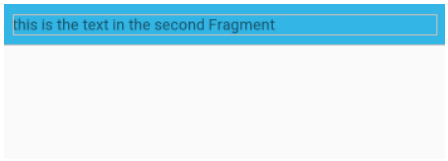

Figure 7.8.b Second Fragment layout with textView

Figure 7.8 fragments layouts

- Adding fragments to the Main Activity

Convert the constraint layout into linear vertical layout and then add the following:

- Text view and change the text to “Main Activity”
- From the palette on the left side hand of the window choose <fragment> (drag and drop the fragment to the activity_main layout). A fragment popup screen will show up, choose the FirstFragment as shown in Figure 7.9. this will connect the added fragment to the FirstFragment. Change the id to fragment1.
- Repeat the previous step for the SecondFragment. Change the id to fragment2

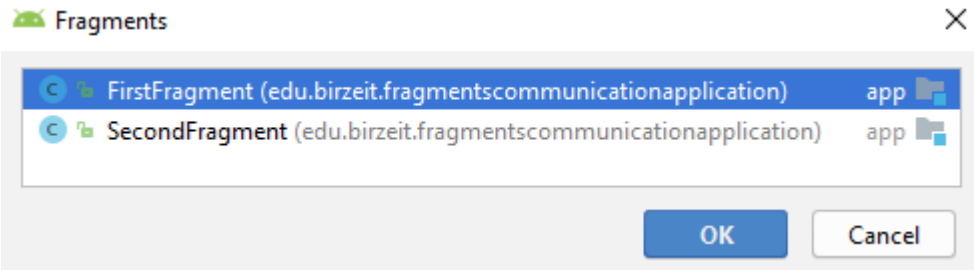

Figure 7.9 adding Fragment screen

❖ Making the communication between the fragments

➢ Define an Interface in the SecondFragment class and implement it within the Activity.

- In the second fragment java class make an interface and call it communicator.
- Add an abstract method in the communicator and call it respond. This method will be implemented in the Main activity, this method will take a string to be displayed in the text view. As shown in the following code.
- Inside SecondFragment class, define a method to display the data on a Text View. As shown in the following code.

```
interface communicator {
    public void respond(String data);
}

public void changeData(String data){
    TextView textView = (TextView) getActivity().findViewById(R.id.textView);
    textView.setText(data);
}
```

➢ Implement the Interface: The activity that hosts FristFragment and SecondFragment must implement the interface defined in the SecondFragment class.

- Implement the SecondFragment.communicator in the main activity.
- Override the respond method in the Main activity.
- Get a reference to the second fragment from the layout.
- Call the change data method in the second fragment.
- The following code shows the override respond method in the main activity.

```
@Override
public void respond(String data) {
    SecondFragment secondFragment =
(SecondFragment)getSupportFragmentManager().findFragmentById(R.id.fragment2);
    secondFragment.changeData(data);
}
```

➢ Calling the respond method from the FirstFragment

- In the first fragment override the onActivityCreated method
- Make an instance of the Second Fragment communicator
- Add a click Listener to the button
- And send the text containg the number of times the button has been clicked to be displayed in the SecondFragment
- The following code shows the First Fragment onActivityCreated method.

```
@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    final SecondFragment.communicator communicator =
(SecondFragment.communicator)getActivity();
    Button button = (Button) getActivity().findViewById(R.id.button);
    final int[] i = {0};
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            i[0]++;
            communicator.respond("the button is clicked "+ i[0] +" times");
        }
    });
}
```

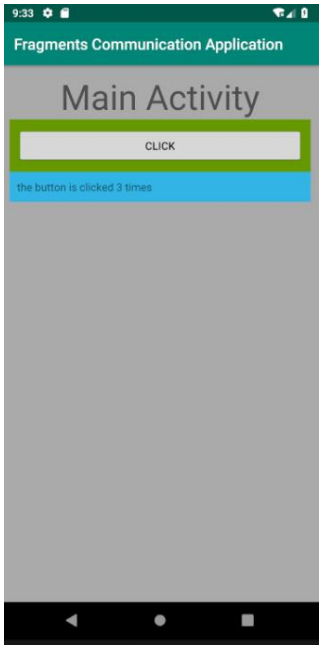

Figure 7.10 Fragment communication Application

4.2. Second Application

You will design an application that can add, remove, attach, detach and replace fragments using fragment transaction. These will be done dynamically in the java code of the main activity. Figure 7.11 shows the design of the application before and after adding the fragments.

Start by creating a new project and change the name of the project to "Fragments Transaction Application". Add two Fragments (FirstFragment and Second Fragment) as we did in the previous section and add a text view in each fragment and change the text of each one to indicate the fragment name (e.g. text view in the First fragment change the text to "This Is The First Fragment"). Also change of the root layout for each fragment layout. The code below shows the FirstFragment Layout.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".FirstFragment"
    android:background="@android:color/holo_green_dark">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="This Is The First Fragment"

android:textAppearance="@style/TextAppearance.AppCompat.Display2" />

</FrameLayout>
```

- We will show how to make a transaction for adding a fragment and you should do the remove, attach, detach and replace.
- To add a fragment dynamically you will use fragment manager to make a fragment transaction to add a new fragment and after you finish you should commit the transaction you made. The code below shows how to add a the FirstFragment when clicking on a button.

```
Button buttonAddF = findViewById(R.id.add_f);
final FirstFragment firstFragment = new FirstFragment();
final FragmentManager fragmentManager = getSupportFragmentManager();

buttonAddF.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.add(R.id.root_layout, firstFragment, "FristFrag");
        fragmentTransaction.commit();
    }
});
```

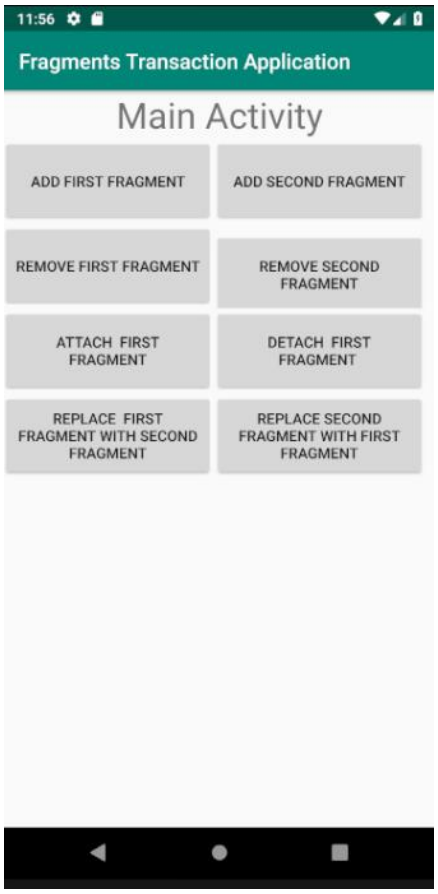

Figure 7.11 before adding the fragments

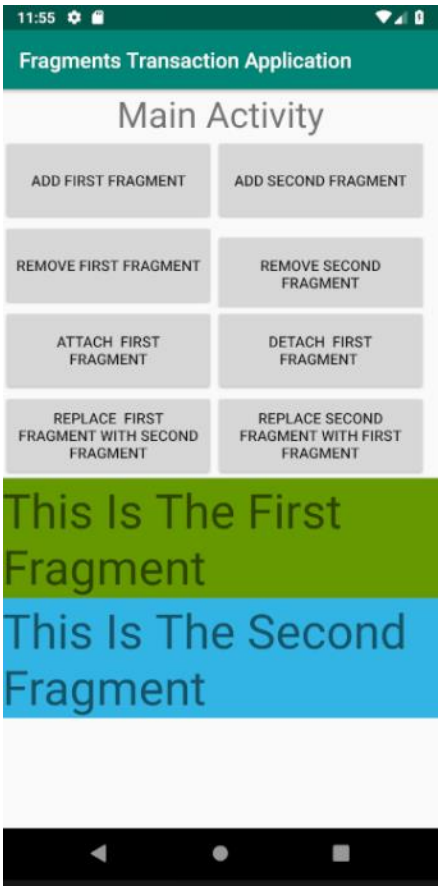

Figure 7.11 after adding the fragments

Figure 7.11 Second application Layout

5. Todo

This part will be given to you by the teacher assistant in the lab time.