Fragments

Mario Mata
mario.mata@gcu.ac.uk
M215a

Glasgow Caledonian University

University for the Common Good

Fragments

- A **Fragment** is a *portion* of a user interface in an activity
  - A Fragment has its own layout and code
    - It can be "self-contained" and handle its own behaviour
  - Fragments can be combined in an activity to build *multi-pane* User interfaces
    - Can prepare one placeholder in the main layout, then inflate one fragment or another inside it as needed
  - Fragments can be easily *reused* in multiple activities

- Fragments are attractive to the developer when:
  - You want to create an independent Android component that can be used by any Activity
  - You want functionality that is easier to reuse within activities and layouts
  - Where different screen sizes are available, Fragments can be used to provide functionality to take this into account

Fragments life cycle

- A **fragment** runs in the context of an activity, but:
  - it has its **own user interface** (defined in its own *.xml* layout file) and Java class (in its own *.java* file)
  - and it has its **own life cycle**

- Accordingly, a fragment can be in any life cycle states:
  - INITIALIZED, CREATED, STARTED, RESUMED, DESTROYED

  https://developer.android.com/guide/fragments/lifecycle

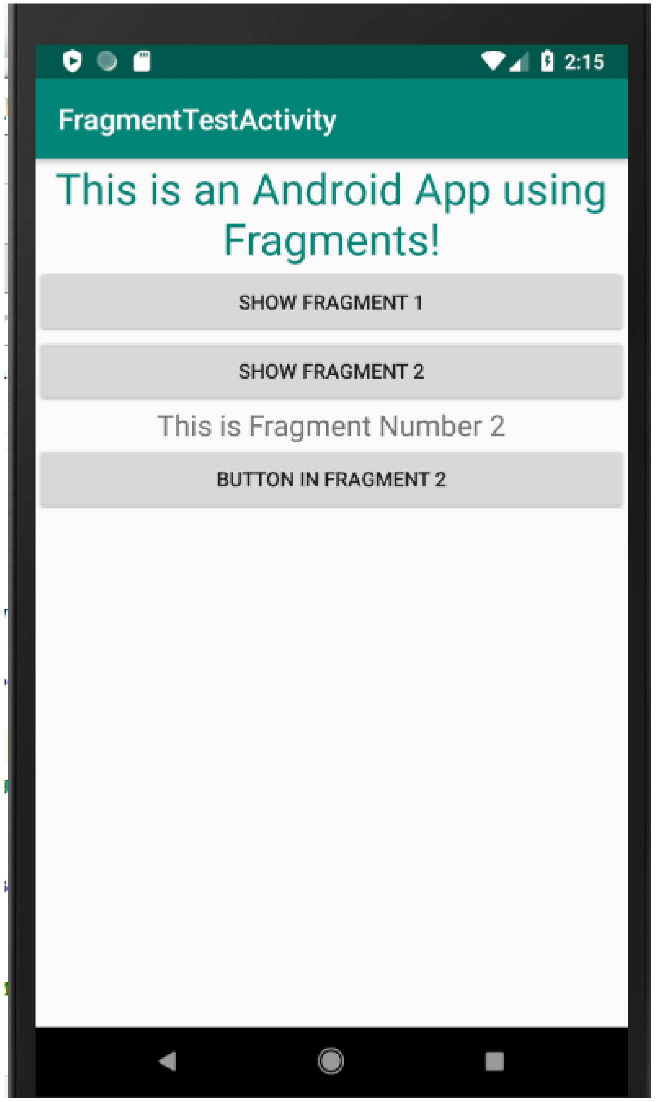
- And it has callback methods that correspond to each of the changes in a fragment's lifecycle:
  - `onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy()`

Fragments for flexible interfaces

- Fragments can be **dynamically** *added* and *removed* from an activity UI with its Java code
  - Much better than using *visibility* properties: invisible widgets still use resources

- This approach allows building flexible interfaces

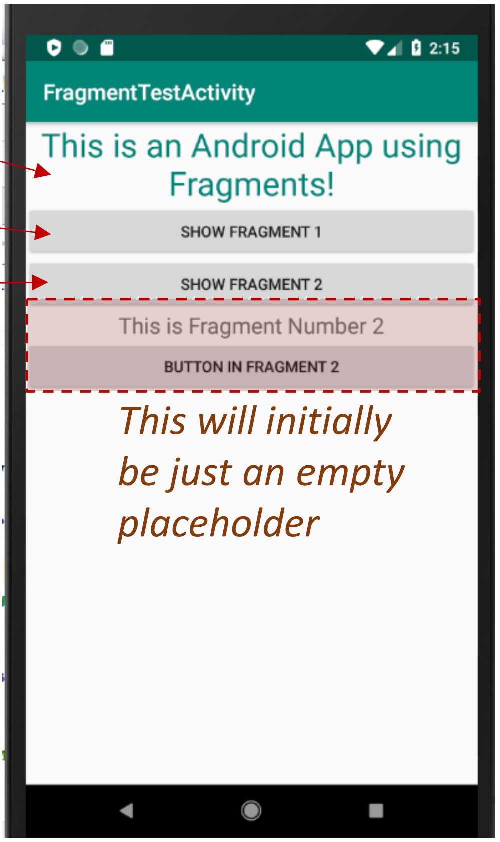
- To utilise fragments in your application, you need to use the `Fragment` class or one of its sub classes (`ListFragment`, etc.)



Fragments example – main layout

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:text="This is an Android App using Fragments!" />

    <Button
        android:id="@+id/frbutton1"
        android:text="Show Fragment 1" />
    <Button
        android:id="@+id/frbutton2"
        android:text="Show Fragment 2" />
    <FrameLayout
        android:id="@+id/fragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```



This is an Android App using Fragments!

This is Fragment Number 2

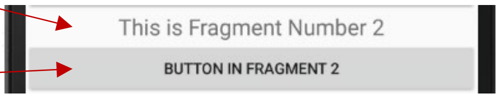
This will initially be just an empty placeholder

Fragments example – fragment **layout**

fragment_fragment_two.xml

```
<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:id="@+id/fragment2"
    tools:context="gcu.mpd.labstuff.FragmentTwo">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal"
        android:textSize="20sp"
        android:text="@string/fragment2Text" />
    <Button
        android:id="@+id/dialogButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button in Fragment 2" />
</LinearLayout>
```

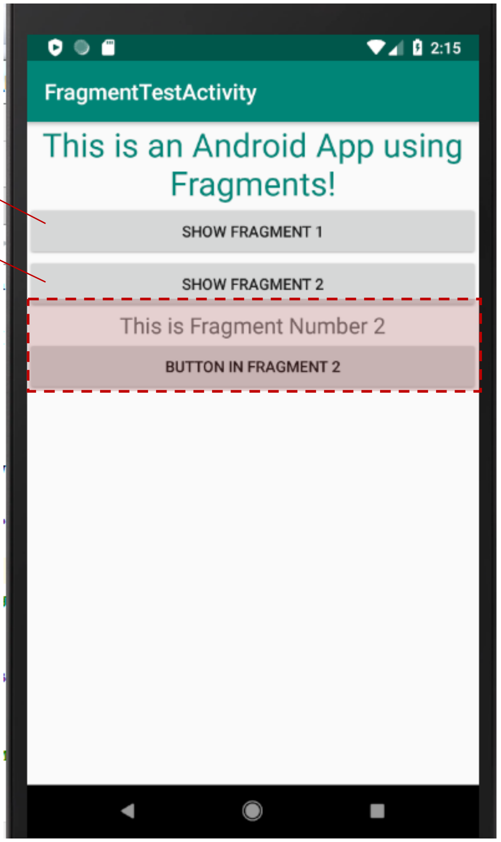
This will get inflated into the placeholder



Fragments ex. – app widget handlers

MainActivity.java

```
public class MainActivity extends AppCompatActivity
implements View.OnClickListener {
   private Button frButton1;
   private Button frButton2;
   private Fragment fr1;
   private Fragment fr2;
   private Fragment fr;

       // Code

}
```



Displaying a fragment (*transaction*)

MainActivity.java

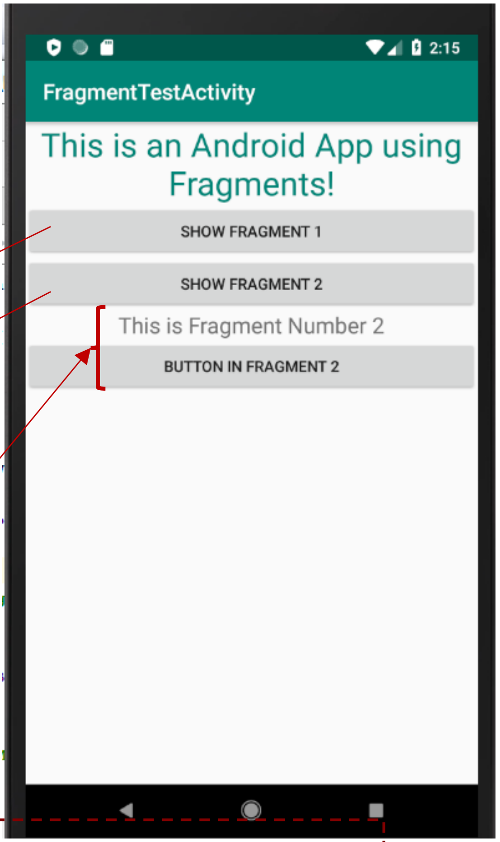
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    frButton1 = findViewById(R.id.frbutton1);
    frButton2 = findViewById(R.id.frbutton2);
    frButton1.setOnClickListener(this);
    frButton2.setOnClickListener(this);
    fr1 = new FragmentOne();
    fr2 = new FragmentTwo();
    fr = fr2; //pick fragment 2
    //Set fr as the current Fragment to show
    FragmentManager manager = getSupportFragmentManager();
    FragmentTransaction transaction = manager.beginTransaction();
    transaction.replace(R.id.fragment,fr); //send fragment fr into Frame
    transaction.commit(); //update the view
}
```

bind

bind

replace **fr** into placeholder

*Prepare fragment **transaction**, then **commit** to apply*



Changing fragment to show

MainActivity.java

```
public void onClick(View v) { //handle buttons onClicks
   if (v == frButton1){
     fr = fr1;
     Log.e("MyTag","Fragment 1 selected");
   }
   else if (v == frButton2){
     fr = fr2;
     Log.e("MyTag","Fragment 2 selected");
   }
   //update fragment view
   FragmentManager manager =
                    getSupportFragmentManager();
   FragmentTransaction transaction =
                    manager.beginTransaction();
   transaction.replace(R.id.fragment,fr);
   transaction.commit();
}
```

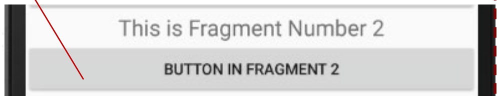
*Prepare fragment **transaction**, then **commit** to apply*

Fragment's own code

FragmentTwo.java

Notice that we must use the fragment's `onCreateView()` to **inflate** the layout into the GUI (this is done automatically for an activity)

```
public class FragmentTwo extends Fragment implements View.OnClickListener {
   private Button dialogCall;
   public FragmentTwo() {} // Required empty public constructor
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
     View v = inflater.inflate(R.layout.fragment_fragment_two, container,
                                  false);
     // handle the fragment's own widgets
     dialogCall = (Button) v.findViewById(R.id.dialogButton);
     dialogCall.setOnClickListener(this);
     return v;
   }
   public void onClick(View v){
       FragmentManager fm = getFragmentManager();
       MyDialogFragment dialogFragment = new MyDialogFragment ();
       dialogFragment.show(fm, "Sample Fragment");
   }
}
```



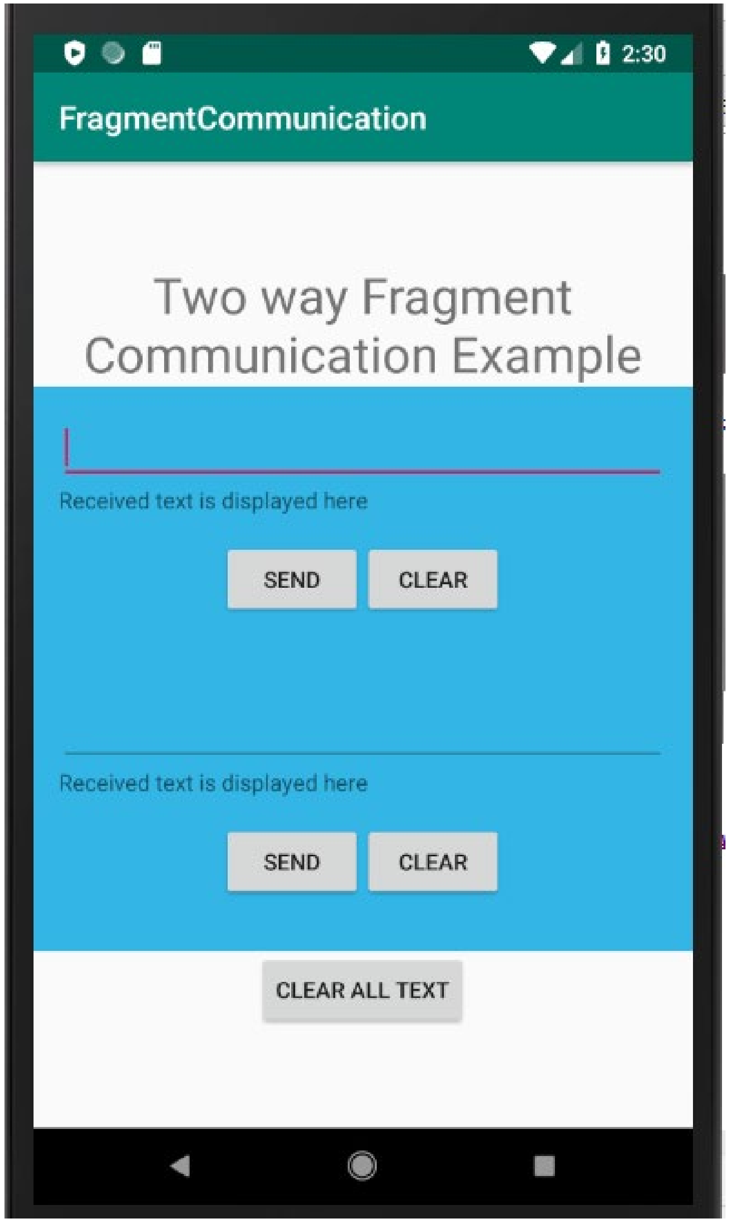
The fragment can handle its own widgets (so it is self-contained)

Fragment communication

- Although fragments are self-contained, **input/result data** can be sent from/to main activity
  - or between fragments, but they shouldn't be coupled together

- Useful if you use a fragment for data entry (producing result data), or for data display (needing input data)

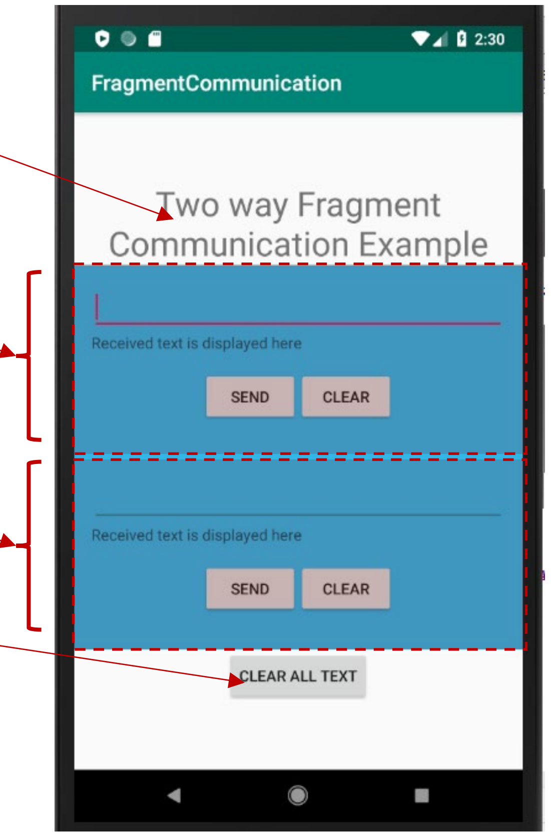
- Example: *FragmentCommunication*


2:30
FragmentCommunication
Two way Fragment Communication Example
Received text is displayed here
SEND
CLEAR
Received text is displayed here
SEND
CLEAR
CLEAR ALL TEXT

Fragment communication – main layout

activity_main.xml

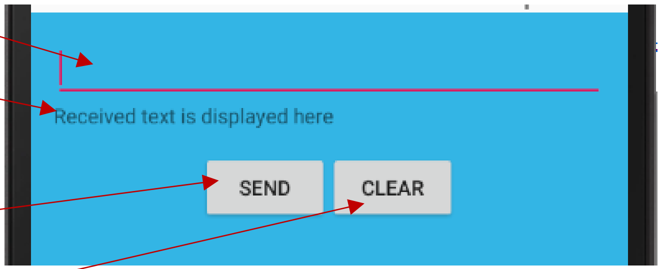
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
       android:orientation="vertical">
<TextView
        android:id="@+id/received_textInA"
        android:text="Two way Fragment Communication Example"

/>

 <FrameLayout
        android:id="@+id/container_a"
 />

 <FrameLayout
        android:id="@+id/container_b"
 />


 <Button
        android:id="@+id/button_clearAll"
        android:text="Clear All Text"
 />
</LinearLayout>
```


2:30
FragmentCommunication
Two way Fragment Communication Example
Received text is displayed here
SEND
CLEAR
Received text is displayed here
SEND
CLEAR
CLEAR ALL TEXT

Fragment A layout

fragment_a.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    ……
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:padding="16dp">
    <EditText
        android:id="@+id/edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/received_textInA"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Received text is displayed here" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_blue_light"
        android:gravity="center_horizontal"
        android:orientation="horizontal"
        android:padding="16dp">
    <Button
        android:id="@+id/button_ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send" />
    <Button
        android:id="@+id/button_clear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clear" />
    </LinearLayout>
</LinearLayout>
```



Fragment communication – main activity

MainActivity.java

```
public class MainActivity extends AppCompatActivity
  implements FragmentA.FragmentAListener,
  FragmentB.FragmentBListener, View.OnClickListener
{
    private FragmentA fragmentA;
    private FragmentB fragmentB;
    private Button clearButton;

    …………
    protected void onCreate(Bundle savedInstanceState){
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     clearButton=
            (Button)findViewById(R.id.button_clearAll);
     clearButton.setOnClickListener(this);

     fragmentA = new FragmentA();
     fragmentB = new FragmentB();

     //Send fragments into Frames
     getSupportFragmentManager().beginTransaction()
             .replace(R.id.container_a, fragmentA)
             .replace(R.id.container_b, fragmentB)
             .commit();

    }
```

*Inline **transaction** and **commit***

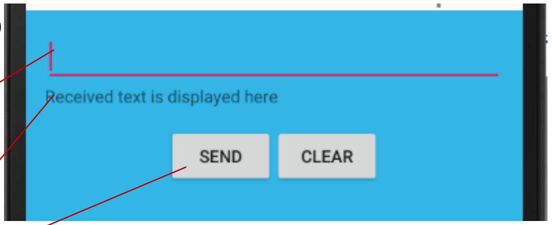
Fragment A – create Listener interface

FragmentA.java

```
public interface FragmentAListener{ //interface definition
   void onInputASent(CharSequence input); //callback method
}
```

*Public **interface** for data output*

```
private FragmentAListener listener;

public View onCreateView(……){
    View v = inflater.inflate(R.layout.fragment_a, co
```



```
    editText = v.findViewById(R.id.edit_text);
    receivedinA = v.findViewById(R.id.received_textInA);
    buttonOk = v.findViewById(R.id.button_ok);
    buttonOk.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            CharSequence input = editText.getText();
            listener.onInputASent(input); //sends via interface's callback
        }
    });
```

```
public void updateEditText(CharSequence newText){
     receivedinA.setText(newText);        //called from main
}
```

Method to get input data

Fragment comm. – main activity (II)

MainActivity.java

```
public class MainActivity extends AppCompatActivity
  implements FragmentA.FragmentAListener,
  FragmentB.FragmentBListener, View.OnClickListener
{
      …………
      //implement fragments A and B interface callbacks:
      @Override
      public void onInputASent(CharSequence input){
              fragmentB.updateEditText(input);
      }
      @Override
      public void onInputBSent(CharSequence input){
              fragmentA.updateEditText(input);
      }


      …………
```

Receives data from fragment via its interface

Sends data to the other fragment via its input method

Use of interfaces to set up callbacks

- In the previous example, notice how the use of **callbacks** is achieved –you can apply it to other situations!

1. Define an interface I in class A (with just a *signature* for method X, that will be the *callback* method)
   - An object A will **call** method X when needed –but this will be *implemented* somewhere else, in the *listener* class
2. In the listener class, *implement* interface I (which will need overriding method X, the *callback* method)
   - When an object A calls method X, this is the method that will then be executed (called back)

- We have been using this mechanism already, using the pre-defined interfaces inside Views (widgets)!