Lecture 2

- Activity
- Fragment
- References to Views
- Event Management
- Workshop
  - Design and implementation of programs
  - Activity to initiate programs and keep Fragment objects
  - UI and event handling in Fragments
  - Controller class with logic

Important UI classes

View

Super class for all UI components, visible (controls / widgets) and invisible as layouts.

View Group

Subclass of View. Can contain multiple Views. One example is the LinearLayout

Fragment

Much like a JPanel in java. Includes a custom UI components and event handling.

Activity

Much like a JFrame in java. Represents a window on the screen.

Activity

- A window in the application
- The application starts in an Activity
- Contains UI often consisting of fragments
- Contains reference to resources
- The activity class must extend Activity

```
public class App extends Activity {

}
```

- An Activity must be recorded in the Manifest

```
:
<application ….>
    :
    <activity …>
        <intent-filter>
        :
        </intent-filter>
    </activity>
    :
</application>
```

Activity - lifecycle

An Activity can be in three different states:

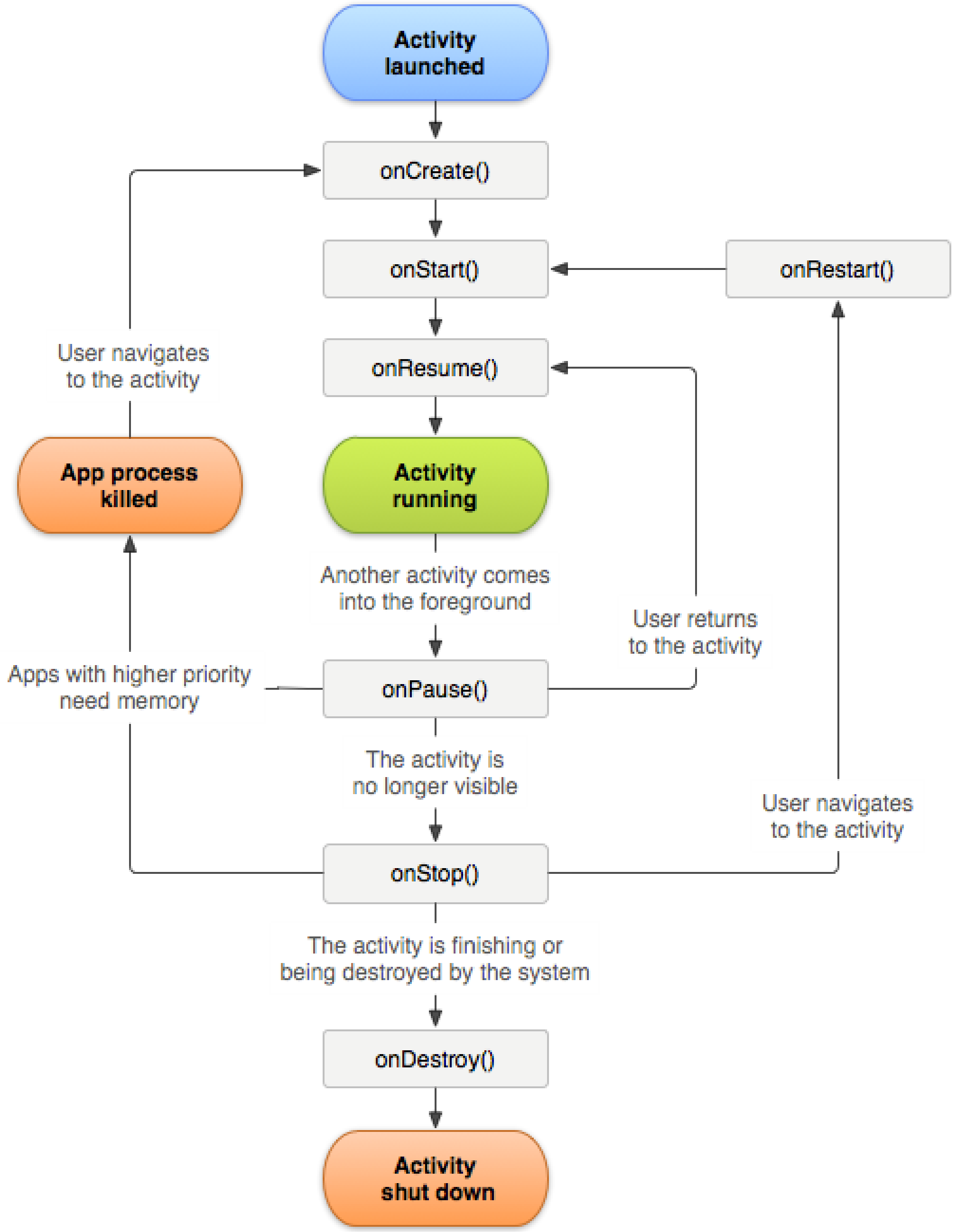
- Active - visible

  onCreate (), onStart () and onResume () is called
- Paused - partially visible

  After Active when onPause () called
- Stopped - invisible

  After paused, then onStop () called

https://developer.android.com/guide/components/activities/activity-lifecycle.html


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity - lifecycle

onCreate(Bundle savedInstance)
Initializes the Activity and creates the UI

Possibly activates resources that are disabled at the interruption by Activity (savedInstance! = Null)

onRestart(), onStart()
Enable resources disabled by onStop ()

onResume()
Enable resources disabled in onPause ()

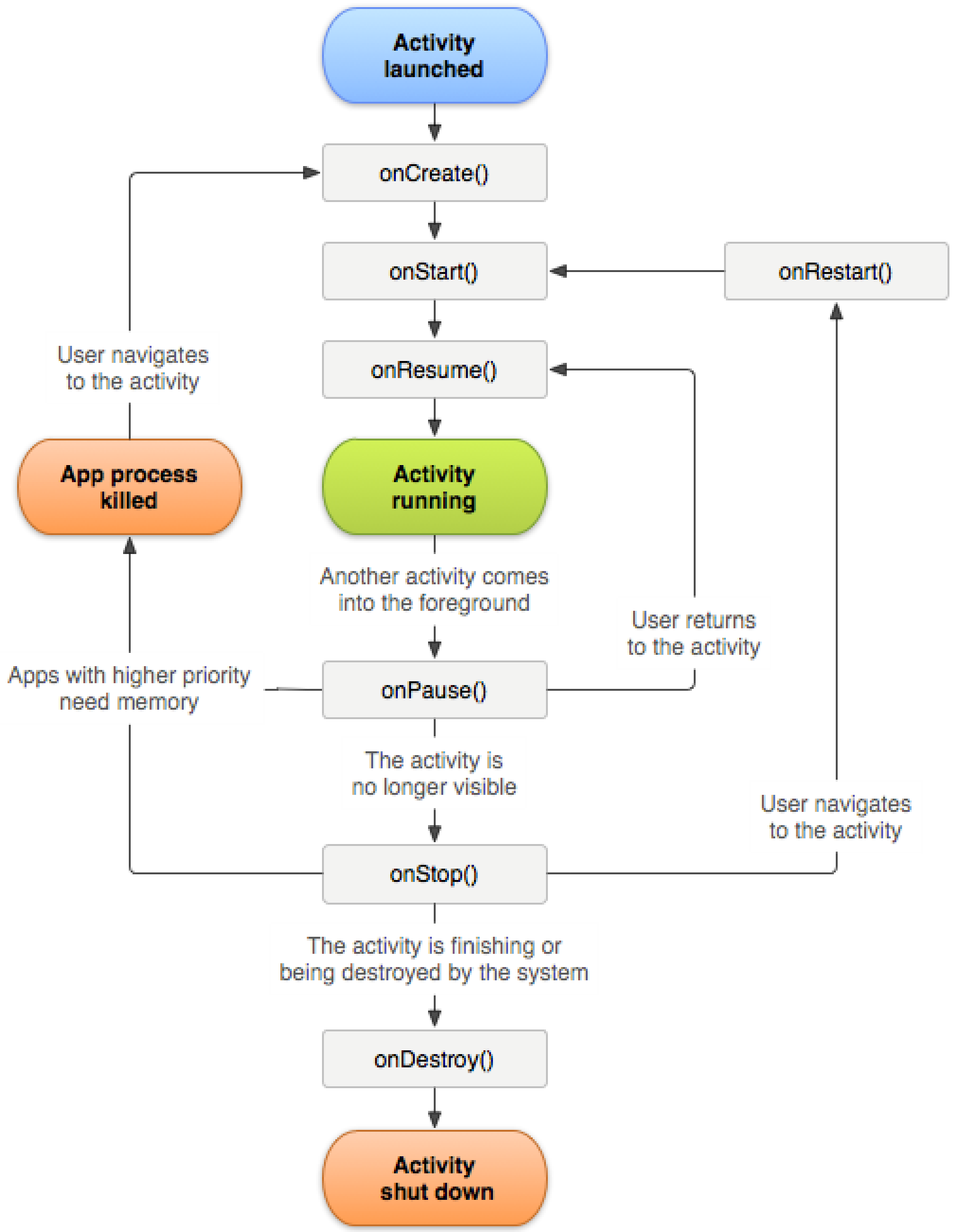
onSaveInstanceState(Bundle savedInstance)
Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state.

onPause()
Disables resources that are not required. Saves the necessary data.

Fast execution is required!

onStop()
Disable resources that are not disabled in onPause()

onDestroy()
final cleaning


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity - UI, Views and Events

- A window in the application
- The application starts in a Activity
- The Activity holds the screen and handles the UI. It’s often composed by fragments.

```
public class AnActivity extends Activity {
    private Button btnHello;
    private TextView tvInfo;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initializeComponents();
        registerListeners();
    }

    private void initializeComponents() {
        tvInfo = (TextView)findViewById(R.id.tvInfo);
        btnHello = (Button)findViewById(R.id.btnHello);
    }
```

- `onCreate, onRestart, onStart and onResume begin with a call to the superclass method.`
- `onSaveInstanceState, onPause, onStop and onDestroy end with a call to the superclass method.`

Activity - UI, Views and Events

- A window in the application
- The application starts in a Activity
- The Activity holds the screen and handles the UI. It’s often composed by fragments.

```
public class AnActivity extends Activity {
    :
    private void registerListeners() {
        btnHello.setOnClickListener(new BL());
    }

    private class BL implements OnClickListener {
        int index=0;
        String[] info = {"Hello","GoodBye"};

        public void onClick(View v) {
            index = (index+1) % 2;  // index = 0,1,0,1,0,1 osv
            tvInfo.setText(info[index]);
        }
    }
   }
```

Fragment

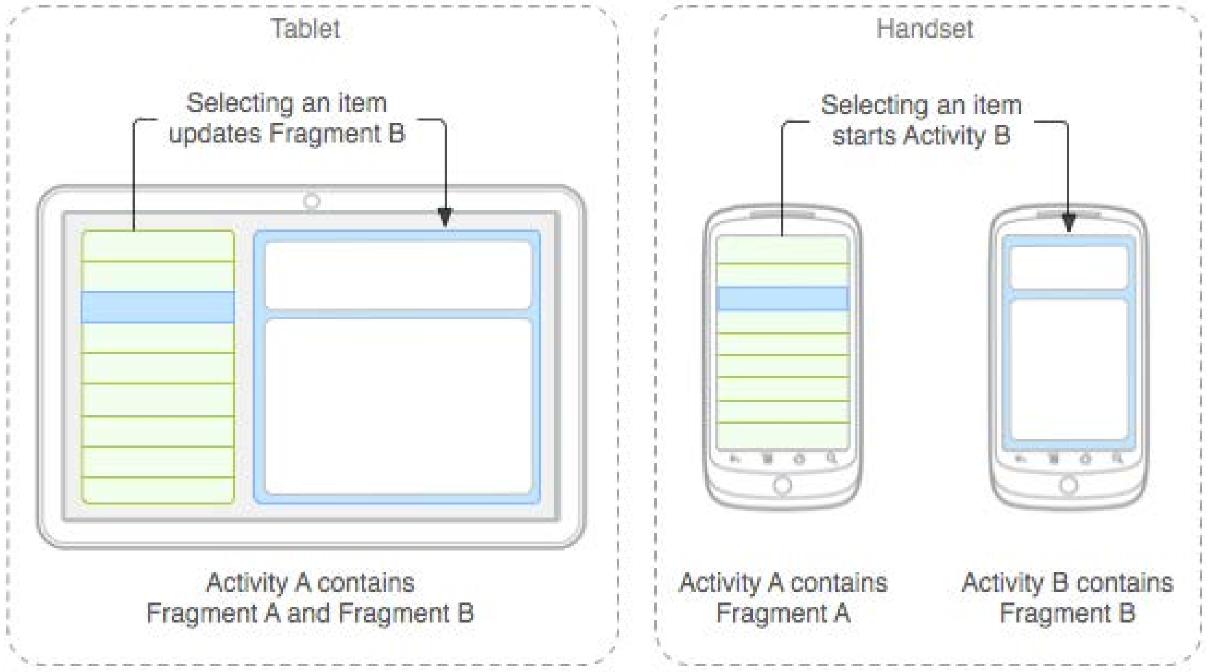
- A Fragment represents a behavior or a portion of user interface in an Activity.
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
- As a panel in Java
- Has its own UI and event management
- Should be well encapsulated
- Should not contain logic
- Can reference to his Activity: method getActivity()
- The fragment class must extend Fragment

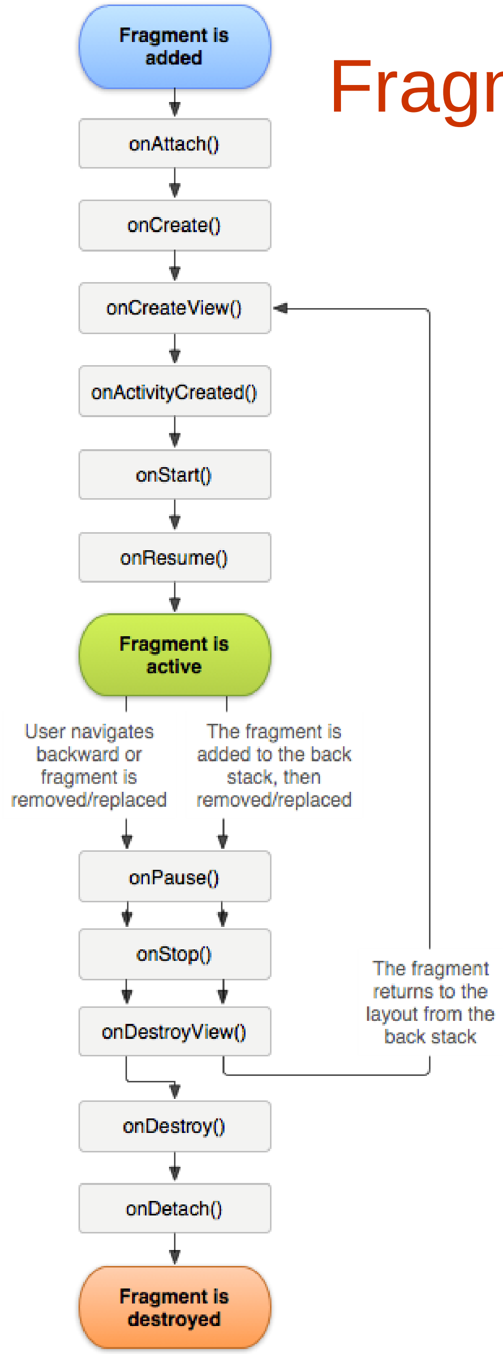
```
public class Frag extends Fragment {

}
```

Fragment

- Fragments can be reused to support tablets and cell phones in different configurations.
- Cell phones might need to separate fragments to provide a single-pane UI when more than one cannot fit within the same activity.


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B



Fragment - lifecycle

A Fragment has methods corresponding to the methods in the Activity class.

onAttach(Activity activity)
called once the fragment is associated with its activity.

onCreate(Bundle savedInstanceState)
initializes the fragment

public View onCreateView(LayoutInflater inflater,
ViewGroup container,
Bundle savedInstanceState)

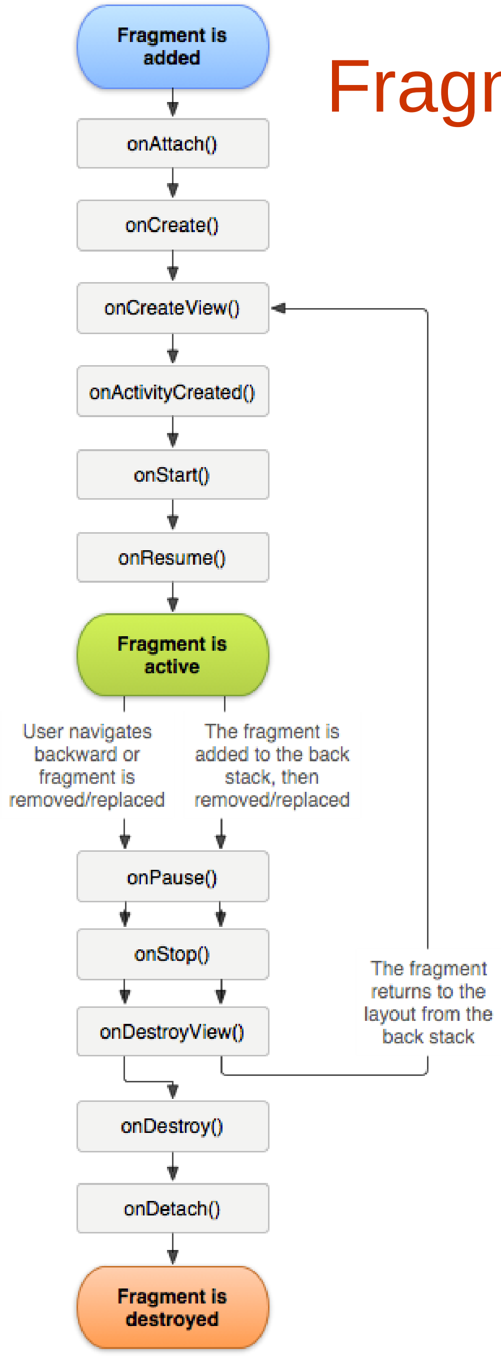
creates and returns the view hierarchy associated with the fragment.

onActivityCreated()

tells the fragment that its activity has completed its own Activity.onCreate().

onStart() and **onResume()**
make the fragment visible and interactable to the user.

Superclass methods are called at the beginning of these methods, except for onCreateView.



Fragment - lifecycle

As a fragment is no longer being used, it goes through a reverse series of callbacks:

onPause()
disables resources

onSaveInstanceState(Bundle savedInstance)
Called to ask the fragment to save its current dynamic state, so it can later be reconstructed in a new instance of its process is restarted.

onStop()
fragment is no longer visible to the user either because its activity is being stopped.

onDestroyView()

allows the fragment to clean up resources associated with its View.

onDestroy()
final clean up.

onDetach()

Deataches the fragment from the activity

Superclass methods called at the end of these methods.

Fragments - in an Activity

Fragments can be placed in a layout and labeled with a tag. E.g.:

```
<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“ >

    <fragment
        android:id="@+id/viewer_fragment"
        android:name="se.mah.tsroax.staticfragment.ViewerFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <fragment
        android:id="@+id/input_fragment"
        android:name="se.mah.tsroax.staticfragment.InputFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Fragments - UI, Views and Events

```
public class AFragment extends Fragment {
    private Button btnHello;
    private TextView tvInfo;

    public View onCreateView(LayoutInflater inflater,
                                ViewGroup container,
                             Bundle savedInstanceState) {
        View view =  inflater.inflate(R.layout.viewer,
                                            container, false);
        initializeComponents(view);
        initializeResources();
        return view;
    }

    private void initializeComponents(View view) {
        tvInfo = (TextView)view.findViewById(R.id.tvInfo);
        btnHello = (Button)view.findViewById(R.id.btnHello);
    }
```

Fragments - UI, Views and Events

```
public class AFragment extends Fragment {
       private Controller controller;
    :
    private void registerListeners() {
        btnHello.setOnClickListener(new BL());
    }

    public void setInfo(String str) {
           tvInfo.setText(str);
    }

    private class BL implements OnClickListener {
        public void onClick(View v) {
            controller.newInfo();
        }
    }
   }
```

FragmentManager

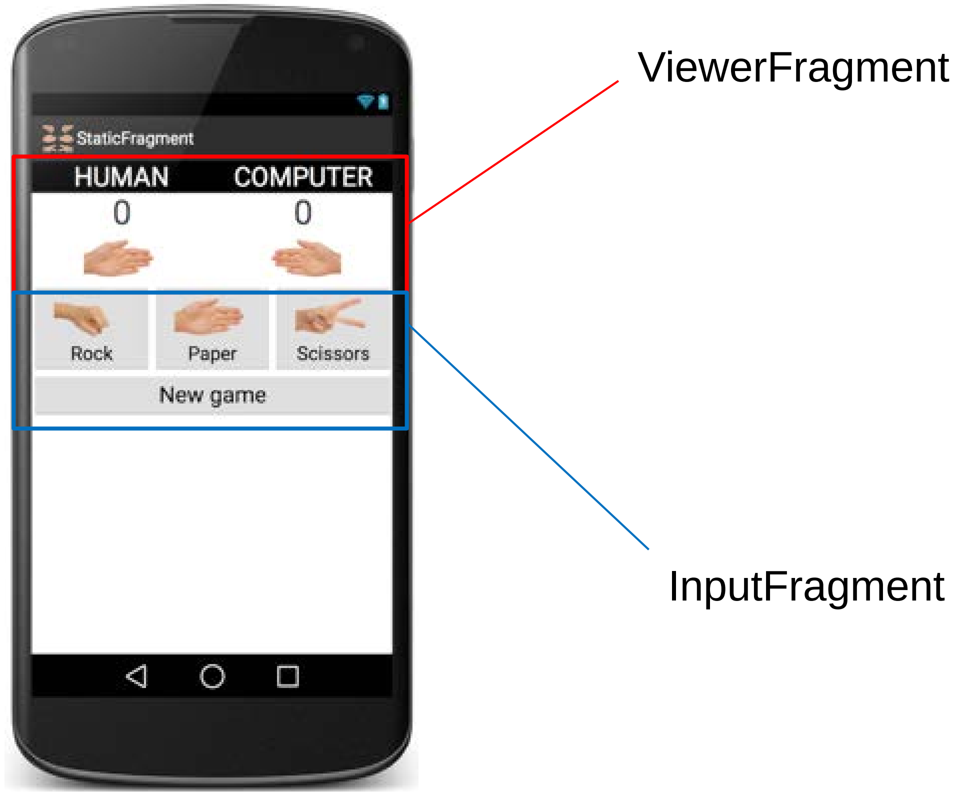
- The Activity class can manage its Fragments with a FragmentManager object.
- FragmentManager is an interface for interacting with Fragment objects inside of an Activity.

```
FragmentManager fm = getFragmentManager();
ViewerFragment viewer =
  (ViewerFragment)fm.findFragmentById(R.id.viewer_fragment);
InputFragment input =
  (InputFragment)fm.findFragmentById(R.id.input_fragment);
```

Workshop

A Rock, Paper, Scissors game is composed by:

- An Activity (Main Activity)
- Two Fragments (Input Fragment and Viewer Fragments)
- A controller class (RPSController)
- A computer player class (RPSPlayer)


ViewerFragment
StaticFragment
HUMAN
COMPUTER
0
0
Rock
Paper
Scissors
New game
InputFragment