COMP30510 Mobile Application Development

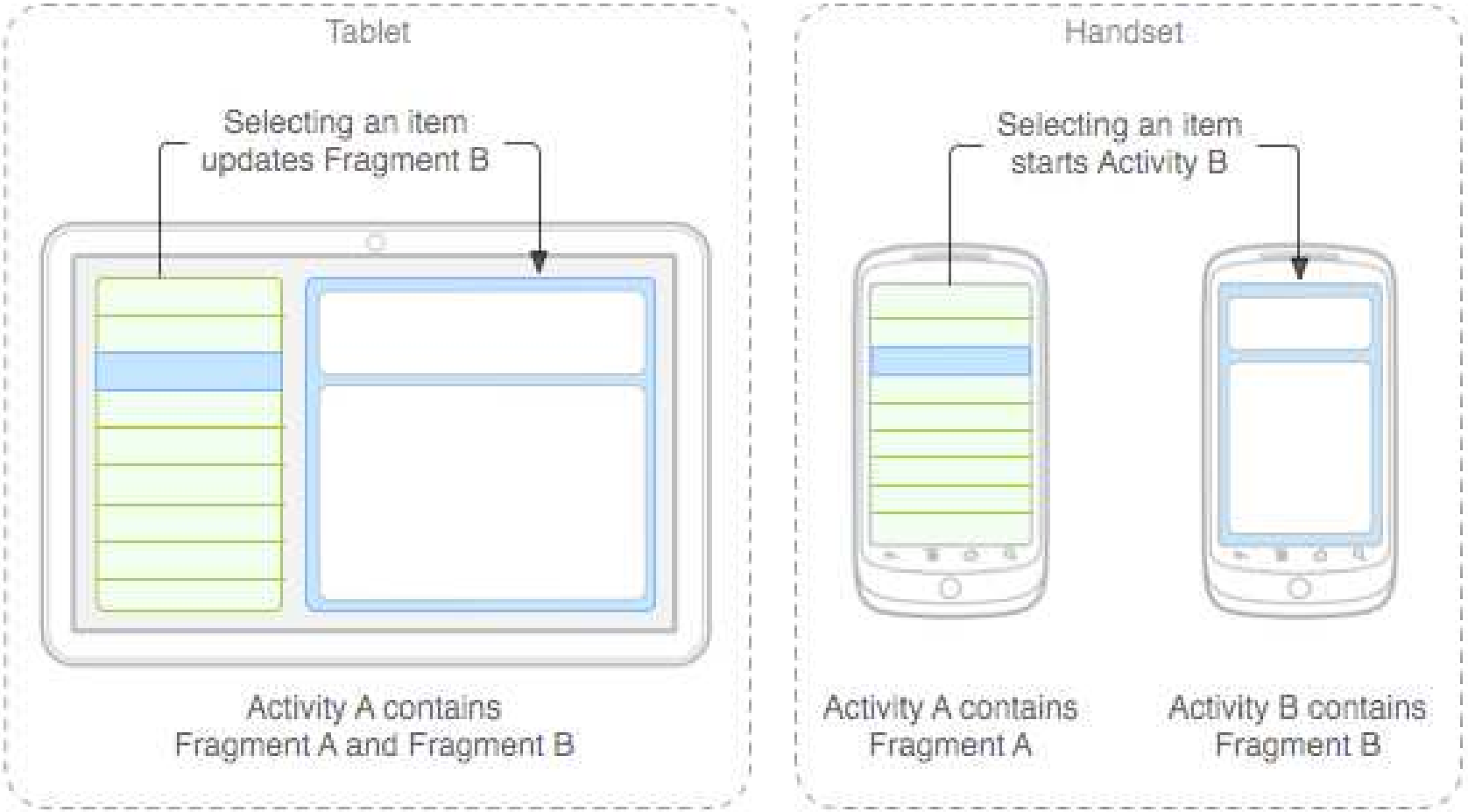
Fragments and Action Bar

Dr. Abraham Campbell
University College Dublin
Abey.campbell@ucd.ie

Fragments

- Introduced in Android 3.0
- Designed to create a more dynamic / flexible design environment for large android devices such as Tablets.
- With the development of larger/ High density Screens for android phones such as the Galaxy Note Series , they have begun to be used for many apps , that want to support smaller and larger screens with content that can adapt automatically to screen size, beyond the standard android resource model.

Fragment- Example



Creating a Fragment

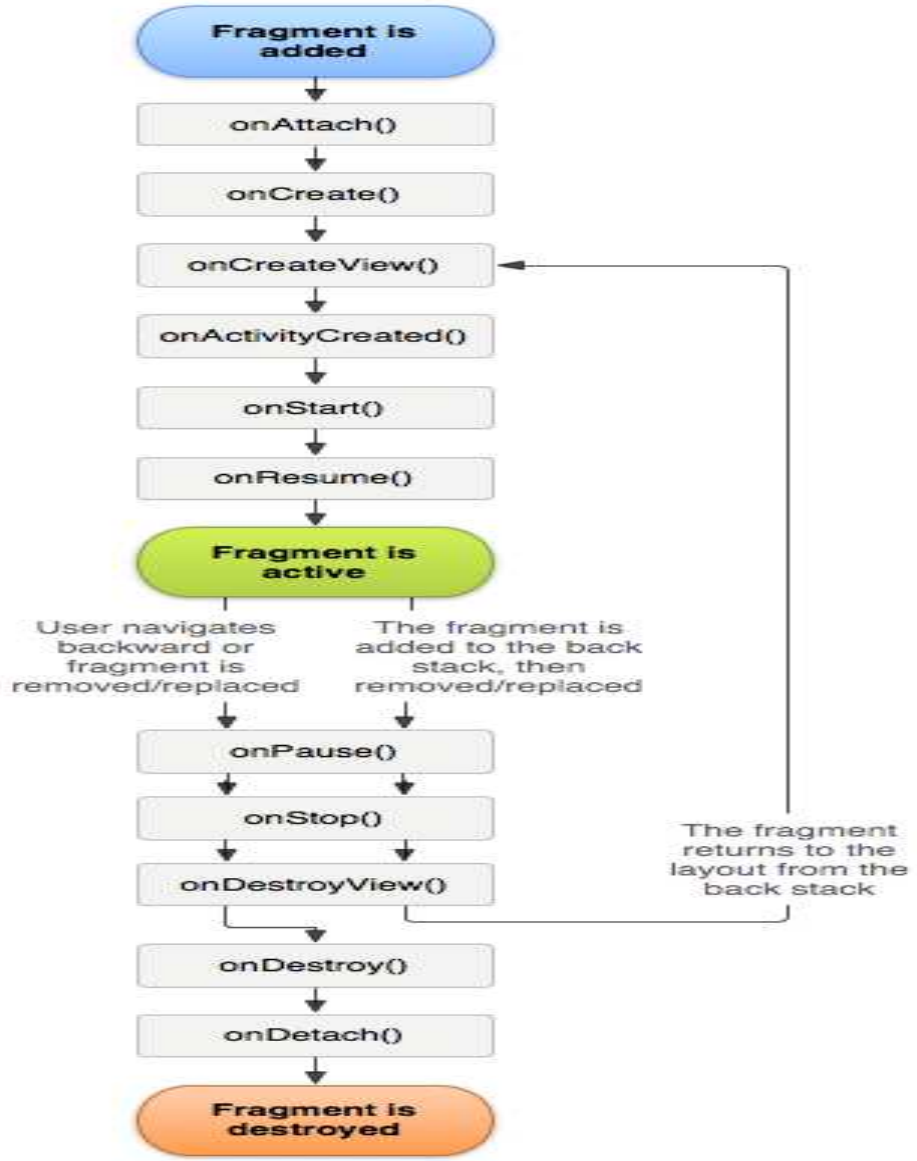
- Similar to creating an Activity , you create a subclass of Fragment
- Its call back methods mirror Activity but you should implement the follow methods
  - onCreate()
    - Initialise the components of the fragment page
  - onCreateView()
    - This is where you create the UI and return it as a view object
    - You can return null , I the fragment does not provide a UI.
  - onPause()
    - Commit any changes
    - Pause/ release any resources that you are using

Fragment lifecycle

- onCreateView() is the restart method if the fragment has been paused or stopped
- In this method you will setup the layout for the fragment and any UI components that are present.
- Below is an example from the Android dev guide , showing the use of a standard XML layout but you can also programmable create its layout just the same as an activity.

```
public static class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container,

                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.example_fragment,
container, false);
    }
}
```



Example of a Fragment setup in a layout xml file

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

Managing Fragments

- To manage what fragments are running you use Fragment Manger
- Calling it by using getFragmentManager()
- You can then find your fragment by tag or ID
  - findFragmentById()
  - findFragmentByTag()
- You can pop fragments directly off the back stack or register a listener to monitor such changes.
  - popBackStack()
  - addOnBackStackChangedListener()

Fragment Transactions

- Fragments are great to setup applications where you may want multiple windows , but if you are on a smaller screen, how do you swap between fragments.
- This is achieved by Fragment Transactions , usually in combination with the action bar which will be discuss later.
- First an instance of FragmentTransaction must be gotten from the Fragment Manager.

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

Fragment Transactions (cont.)

- Once you have the transaction object, you can call
  - add() ,remove() or replace()
- Put you must commit() before anything will happen , for example

```
// Create new fragment and transaction
Fragment newFragment = new ExampleFragment();
FragmentTransaction transaction = getFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

Fragments communicating with its host Activity

- A fragment can get a reference to the activity that is hosting it by call getActivity() method , for example :

```
View listView = getActivity().findViewById(R.id.list);
```

- Alternatively an Activity can gain access to the fragment is running by calling

```
ExampleFragment fragment =
            (ExampleFragment) getFragmentManager().findFragmentById(R.id.example_fragment);
```

Action Bar

- Common Navigational bar for all Android Apps
- Set up in your android Manifest

```
<activity android:theme="@android:style/Theme.Holo">
```

- Created when you setup the Android:Theme unless you pick a theme without an Action Bar in it.

```
<activity android:theme="@android:style/Theme.Holo.NoActionBar">
```

- You can also hide or show the actionbar at any time

```
ActionBar actionBar = getActionBar();
actionBar.hide();
actionBar.show();
```

Adding Action Items

- When you setup a project in eclipse, the following method and xml file will be created automatically
- The method generates the menu and populates it using the res/menu/main.xml

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main_activity, menu);
    return true;
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_save"
          android:icon="@drawable/ic_menu_save"
          android:title="@string/menu_save"
          android:showAsAction="ifRoom|withText" />
</menu>
```

Clicking an Actionbar item

- An example below is to show how clicking on an item could go to another activity .

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            // app icon in action bar clicked; go home
            Intent intent = new Intent(this, HomeActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Using an icon instead of text

- You can improve the look of your actionbar item with an icon. As per this example below

```
?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_search"
          android:title="@string/menu_search"
          android:icon="@drawable/ic_menu_search"
          android:showAsAction="ifRoom|collapseActionView"
          android:actionViewClass="android.widget.SearchView" />
</menu>
```

Nesting Actionbar items to make a menu

- You can nest ActionBar items to aid creating a menu as seen in this example below.

```
<item
    android:id="@+id/action_settings"
    android:orderInCategory="100"

android:title="@string/action_settings"
android:visible="true">

        <menu >

         <item
        android:id="@+id/menu2"
        android:title="@string/menu"
        android:visible="true"
         />
                <item
                android:id="@+id/action_settings3"
                android:title="@string/Menu2"
                android:visible="true" />

                <item
                android:id="@+id/menu23"
                android:title="@string/menu"
android:visible="true"
                />
                </menu>
</item>
</menu>
```

Split actionbar

- You can enable a split actionbar by adding in
- uiOptions="splitActionBarWhenNarrow“ in your activity manifest