CMSC436: Programming Handheld Systems
The Fragment Class
Tablet UIs

Tablets have larger displays than phones do. They can support multiple UI panes / user behaviors at the same time.

The “1 activity – 1 thing the user can do” heuristic may not make sense for larger devices.
FragmentQuoteViewerWithActivity

Application uses two Activities

One shows titles of Shakespeare plays & allows user to select one title

The other shows a quote from the selected play
FragmentQuoteViewerWithActivity

The Tragedy of Hamlet, Prince of Denmark

King Lear

Julius Caesar

Now cracks a noble heart. Goodnight, sweet prince; And flights of angels sing thee to thy rest.
FragmentQuoteViewerWithActivity UI

This layout is reasonable on a phone
But inefficient on a larger device
The Tragedy of Hamlet, Prince of Denmark

King Lear

Julius Caesar
Now cracks a noble heart. Good-night, sweet prince; And flights of angels sing thee to thy rest.
Better Layout

Use two cooperating layout units on one screen
### Plays

<table>
<thead>
<tr>
<th>Play</th>
<th>Quote</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>The Tragedy of Hamlet, Prince of Denmark</em></td>
<td>Now cracks a noble heart. Good-night, sweet prince; And flights of angels sing thee to thy rest.</td>
</tr>
<tr>
<td><em>King Lear</em></td>
<td></td>
</tr>
<tr>
<td><em>Julius Caesar</em></td>
<td></td>
</tr>
</tbody>
</table>
The Fragment Class

Typically represents a behavior / portion of UI
Multiple Fragments can be embedded in an Activity to create a multi-pane UI
A single Fragment can be reused across multiple Activities
Fragment Lifecycle

Fragment lifecycle is coordinated with the lifecycle of its containing/hosting Activity. Fragments have their own lifecycles and receive their own callbacks.
Fragment Lifecycle States

Resumed
  Fragment is visible in the hosting Activity

Paused
  Another Activity is in the foreground and has focus, this Fragment’s hosting Activity is still visible

Stopped
  The Fragment is not visible
Lifecycle Callback Methods
onAttach()

Activity is created
Fragment is first attached to its Activity
onCreate()

Initialize the Fragment

Note: The hosting Activity may not be fully created at this point
onCreateView()

Fragment sets up & returns its user interface View
onActivityCreated()

Containing Activity has completed onCreate() and the Fragment has been installed
Can now safely access hosting Activity
onStart()

Activity is started

Hosting Activity about
to become visible
Activity is resumed
Hosting Activity is about to become visible and ready for user interaction
onPause()

Activity is paused

Hosting Activity is visible, but does not have focus
onStop()

Activity is stopped
Hosting Activity is no longer visible
onDestroyView()

Activity is destroyed

View previously created in onCreateView() has been detached from the Activity

Clean up view resources
onDestroy()

Fragment is no longer in use
Clean up Fragment resources
onDetach()

Fragment no longer attached to its activity
Null out references to hosting Activity
Adding Fragments to Activities

Two general ways to add Fragments to an Activity’s layout

Declare it statically in the Activity’s layout file
Add it programmatically using the FragmentManager
Fragment Layout

Layout can be inflated/implemented in `onCreateView()`

`onCreateView()` must return the View at the root of the Fragment’s layout

This View is added to the containing Activity
FragmentStaticLayout

Display titles and quotes in two Fragments, side-by-side

Fragments are statically added to UI based on a layout file
<table>
<thead>
<tr>
<th>The Tragedy of Hamlet, Prince of Denmark</th>
<th>Please select a Title.</th>
</tr>
</thead>
<tbody>
<tr>
<td>King Lear</td>
<td></td>
</tr>
<tr>
<td>Julius Caesar</td>
<td></td>
</tr>
</tbody>
</table>
class QuoteViewerActivity : FragmentActivity(),
    ListSelectionListener {

    ...  
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        ...  
        setContentView(R.layout.quote_activity)
    }
quote_activity.xml

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
...

<fragment
  android:id="@+id/titles"
  class="course.examples.fragments.staticlayout.TitlesFragment"
  android:layout_width="0px"
  android:layout_height="match_parent"
  android:layout_weight="1" />

<fragment
  android:id="@+id/details"
  class="course.examples.fragments.staticlayout. QuotesFragment"
  android:layout_width="0px"
  android:layout_height="match_parent"
  android:layout_weight="2" />

</LinearLayout>
```
Design Philosophy

Fragments should be reusable across Activities

Avoid coupling Fragments

i.e., Frag1 should not directly interact with Frag2

Coupling should be handled by callbacks to hosting Activity
// Callback interface that defines how a TitlesFragment notifies
// the QuoteViewerActivity when user clicks on a List Item in the
// TitlesFragment

internal interface ListSelectionListener {
    fun onListSelection(index: Int)
}
TitlesFragment.kt

... // Called when the user selects an item from the List
override fun onListItemClick(l: ListView?, v: View?,
                           pos: Int, id: Long) {
  ...
  // Inform the QuoteViewerActivity that item at position pos was selected
  mListener.onListSelection(pos)
  ...
} override fun onAttach(context: Context) {
  ...
  // ListSelectionListener for communicating with QuoteViewerActivity
  mListener = context as ListSelectionListener
  ...

```kotlin
class QuoteViewerActivity : FragmentActivity(), ListSelectionListener {

    // Called by TitlesFragment when the user selects an item
    override fun onListSelection(index: Int) {
        // Tell the QuoteFragment to show the quote string at position index
        mQuotesFragment.showQuoteAtIndex(index)
    }
}
```
Adding Fragments Programmatically

While an Activity is running you can add and remove Fragments from its layout

Four-step process

1. Get reference to the FragmentManager
2. Begin a FragmentTransaction
3. Add the Fragment
4. Commit the FragmentTransaction
FragmentProgrammaticLayout

Displays titles and quotes side-by-side in two Fragments.

Layout file reserves space for Fragments (using FrameLayout elements).

Fragments are programmatically added to UI at runtime.
The Tragedy of Hamlet, Prince of Denmark

Now cracks a noble heart. Good-night, sweet prince; And flights of angels sing thee to thy rest.

King Lear

Julius Caesar
override fun onCreate(savedInstanceState: Bundle?) {
    ...
    val fragmentManager = supportFragmentManager
    if (null == fragmentManager.findFragmentById(R.id.title_frame)) {
        val fragmentTransaction = fragmentManager.beginTransaction()
        fragmentTransaction.add(R.id.title_frame, TitlesFragment())
        fragmentTransaction.add(R.id.quote_frame, mQuoteFragment)
        fragmentTransaction.commit()
    } else {
        mQuoteFragment = fragmentManager.findFragmentByIndex(R.id.quote_frame) as QuotesFragment
    }
}
quote_activity.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activityFrame"
    ...>

    <FrameLayout
        android:id="@+id/title_frame"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <FrameLayout
        android:id="@+id/quote_frame"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2" />

</LinearLayout>
Dynamic Layout

Fragment transactions allow you to dynamically change your app’s user interface.

Can make the interface more fluid & take better advantage of available screen space.
FragmentDynamicLayout

Starts with a single Fragment
Changes to two-Fragment layout when user selects a title
<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Tragedy of Hamlet, Prince of Denmark</td>
</tr>
<tr>
<td>King Lear</td>
</tr>
<tr>
<td>Julius Caesar</td>
</tr>
</tbody>
</table>
The Tragedy of Hamlet, Prince of Denmark

Now cracks a noble heart. Good-night, sweet prince; And flights of angels sing thee to thy rest.

King Lear

Julius Caesar
// Get a reference to the FragmentManager
mFragmentManager = supportFragmentManager

mQuoteFragment = mFragmentManager.
    findFragmentById(R.id.quote_fragment_container) as QuotesFragment?
mTitleFragment = mFragmentManager.
    findFragmentById(R.id.title_fragment_container) as TitlesFragment?
if (null == mFragmentManager.
       findFragmentById(R.id.title_fragment_container)) {
    mTitleFragment = TitlesFragment()
    mTitleFragment?.let { mTitleFragment ->
        // Start a new FragmentTransaction
        val fragmentTransaction = mFragmentManager.beginTransaction()

        // Add the TitleFragment to the layout
        fragmentTransaction.add(R.id.title_fragment_container,
                                mTitleFragment)

        // Commit the FragmentTransaction
        fragmentTransaction.commit()
    }
}
// Called when the user selects an item in the TitlesFragment
override fun onListSelection(index: Int) {
    // If the QuoteFragment has not been created, create and add it now
    if (null == mFragmentManager.findFragmentById(R.id.quote_fragment_container)) {
        mQuoteFragment = QuotesFragment()
        mQuoteFragment?.let {mQuoteFragment ->
            // Start a new FragmentTransaction
            val fragmentTransaction = mFragmentManager.beginTransaction()
            // Add the QuoteFragment to the layout
            fragmentTransaction.add(R.id.quote_fragment_container, mQuoteFragment)
// Add this FragmentTransaction to the backstack
fragmentTransaction.addToBackStack(null)

// Commit the FragmentTransaction
fragmentTransaction.commit()

// Force Android to execute the committed FragmentTransaction
mFragmentManager.executePendingTransactions()

// Tell the QuoteFragment to show the quote string at position index
mQuoteFragment?.showQuoteAtIndex(index)
Configuration Changes

If you call setRetainInstance(true) on a Fragment, Android won’t destroy that Fragment on configuration changes.
Configuration Changes

Results in some changes to lifecycle callback sequence

onDestroy() will not be called
onCreate() will not be called
FragmentStaticConfigLayout

Essentially the same as FragmentStaticLayout
Focus here is on how Fragments are saved and restored on configuration changes
FragmentStaticConfigLayout

In landscape mode
Both Fragments use a large font
TitleFragment takes more horizontal space & allows long titles to span multiple lines
In portrait mode
Both Fragments use a smaller font
TitleFragment will use less space and will ellipsize long titles, limiting them to a single line
The Tragedy of Hamlet, Prince of Denmark

Now cracks a noble heart. Good-night, sweet prince; And flights of angels sing thee to thy rest.

King Lear

Julius Caesar
class QuotesFragment : Fragment() {
    ...
    override fun onCreate(savedInstanceState: Bundle?) {
        ...
        // Retain this Fragment across Activity reconfigurations
        retainInstance = true
    }
}
QuotesFragment.kt

// Set up some information about the mQuoteView TextView
override fun onActivityCreated(savedInstanceState: Bundle?) {
    ...
    mQuoteView = activity!!.findViewById(R.id.quoteView)
mQuoteArrayLen = QuoteViewerActivity.mQuoteArray.size

    showQuoteAtIndex(mCurrIdx)
}
Next

User Interface classes
Example Applications

FragmentQuoteViewerWithActivity
FragmentStaticLayout
FragmentProgrammaticLayout
FragmentDynamicLayout
FragmentStaticConfigLayout