Recording in Progress

This class is being recorded
Please turn off your video and/or video if you do not wish to be recorded
CMSC436: Programming Handheld Systems
The Fragment Class
Tablet UIs

Tablets grew in popularity after Android’s original design.

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.
FragmentActivitiesLayout

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
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.
FragmentQuoteViewerWithActivity UI

This layout is reasonable on a phone
But unnecessary 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
<table>
<thead>
<tr>
<th>The Tragedy of Hamlet, Prince of Denmark</th>
</tr>
</thead>
<tbody>
<tr>
<td>King Lear</td>
</tr>
<tr>
<td>Julius Caesar</td>
</tr>
</tbody>
</table>

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

Typically serves as a container of a portion of the app’s 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

Fragments are hosted by Activities

A Fragment’s lifecycle is coordinated with the lifecycle of its 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 returns a View that will contain its UI
onViewCreated()

Fragment sets up its UI
onStart()

Activity is started
Hosting Activity about to become visible
onResume()

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

onDestroyView()

onDestroy()
onDetach()

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

Two general ways to add a Fragment to an Activity’s layout

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

Layout can be inflated in onCreateView()
onCreateView() must return the View at the root of the Fragment’s layout
This View is added to the containing Activity
Best practice is that UI is created in onCreateView() and finalized in onViewCreated()
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>Title</th>
<th>Select Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Tragedy of Hamlet, Prince of Denmark</td>
<td>Please select a Title</td>
</tr>
<tr>
<td>King Lear</td>
<td></td>
</tr>
<tr>
<td>Julius Caesar</td>
<td></td>
</tr>
</tbody>
</table>

FragmentStaticLayout
Design Philosophy

Fragments should be reusable across Activities

Avoid coupling Fragments

i.e., If app contains two Fragments, Frag1 and Frag2, then Frag1 should not directly interact with Frag2

Coupling should be handled by separate components, such as ViewModels (preferred) or callbacks to hosting Activity
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 FragmentContainerView elements)

Fragments are programmatically added to UI at runtime
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
FragmentDynamicLayout

The Tragedy of Hamlet, Prince of Denmark
King Lear
Julius Caesar
FragmentDynamicLayout

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.
Navigation

Android provides support for structured navigation between app components

See:
https://developer.android.com/guide/navigation
Principles of Navigation

Every app you build has a fixed start destination
Actions take you to a new destination
Navigation state is a stack of destinations
Up and Back actions supported
  Up doesn’t exit the app; back does
SafeArgs (gradle plugin) ensures type safety in argument passing
Navigation Structure

Designed for apps with one Activity and multiple Fragment destinations

Each Activity has a navigation graph – XML resource that defines navigation paths through an app (destinations and actions)

NavHostFragment: An empty container that displays destinations from your navigation graph

NavController: An object that manages app navigation within a NavHostFragment
SafeArgs

Using SafeArgs is recommended best practice

Once enabled, it generates code for each navigation action

- A class for each originating destination, named according to the originating destination class name, and the word "Directions"

- A static method for each action defined in the originating destination, that takes any defined action parameters and returns a NavDirections object that can be passed to navigate()
HelloAndroidWithLoginFragments

Username
aporter@umd.edu

Password

LOGIN

HelloAndroidWithLoginFragments

Hello aporter!
Next

LifeCycleAware Components
Example Applications

FragmentQuoteViewerWithActivity
FragmentStaticLayout
FragmentProgrammaticLayout
FragmentDynamicLayout
HelloAndroidWithFragments