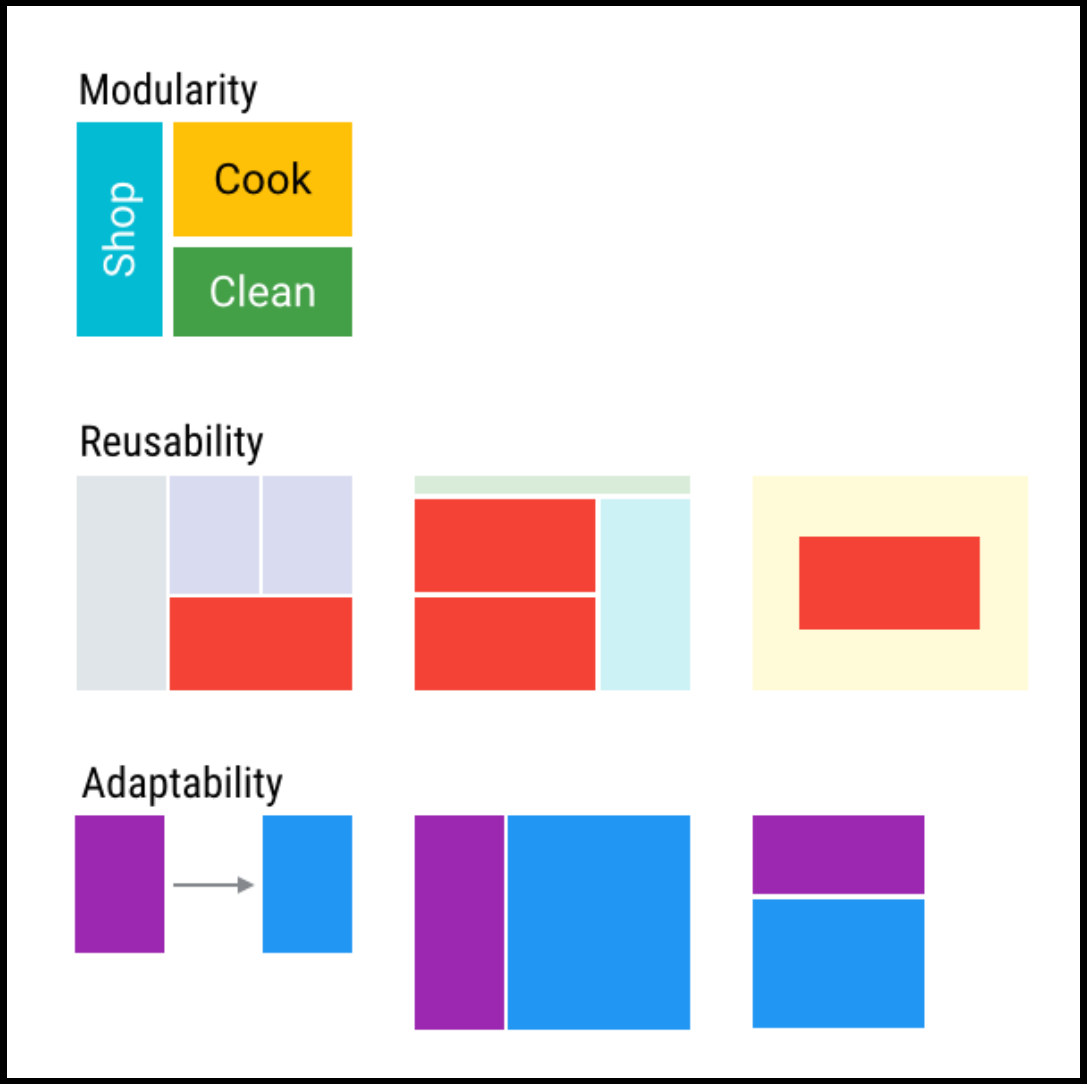
Mobile Application Development
Aileen Pierce

FRAGMENTS

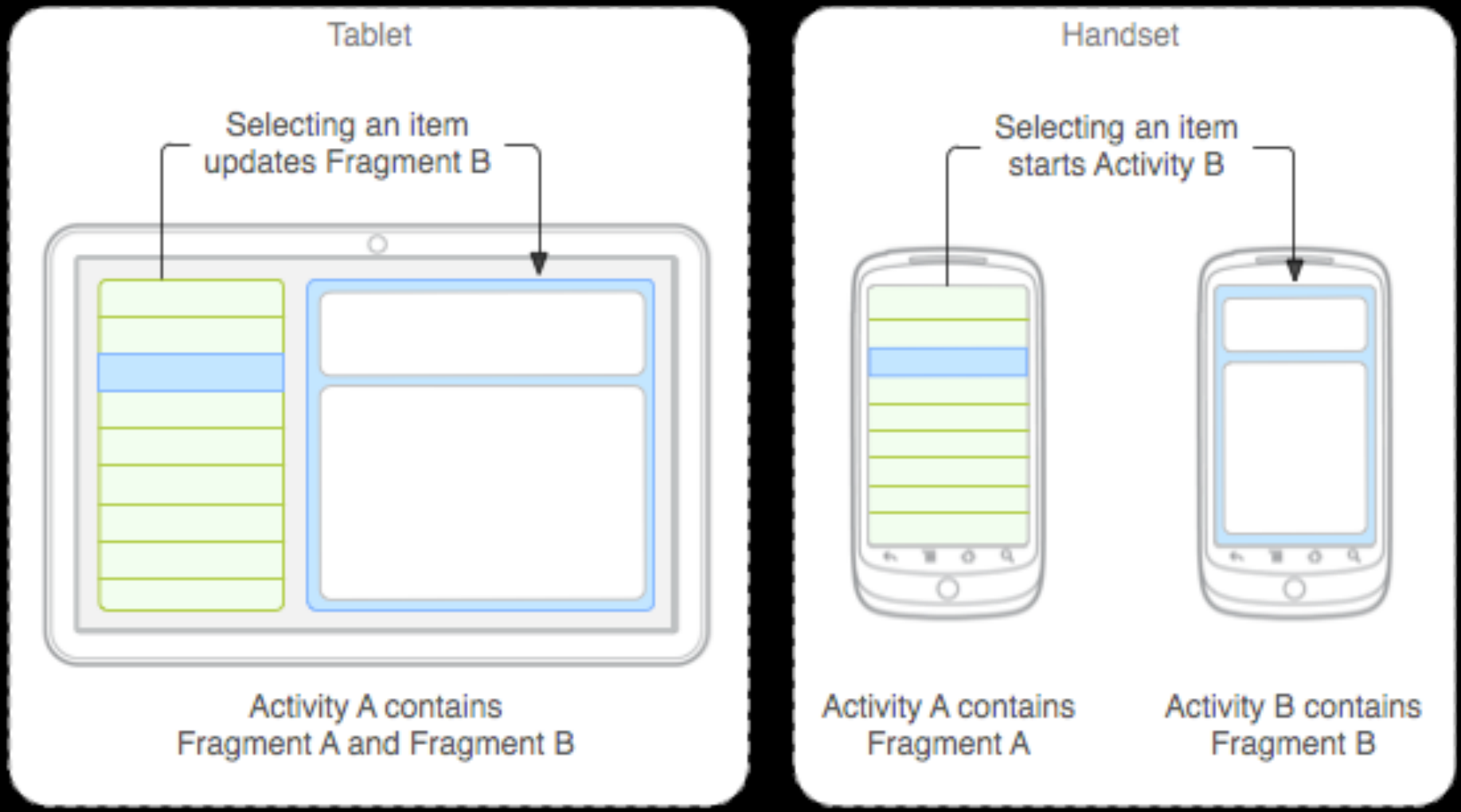
Fragments

- We've been using Activities in our apps but sometimes you want more flexibility
- Fragments allow you to break your activities up into smaller modular components
- Fragments can easily be reused and adapted for different device sizes, orientation, or other criteria
- You can one or more fragments embedded in an activity

Fragments



Fragments


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

Fragments

- The **`Fragment`** class has a lot of the same methods as the **`Activity`** class.
- It also has a **`ListFragment`** subclass which displays a list of items using an adapter just like activities
  - **`onListItemClick()`** handles click events

Fragment Layout

- Use the **`<fragment>`** element to add a fragment to an activity's layout
  - Specify which fragment using the **`class`** attribute

`class="com.example.aileen.superheroes.UniverseListFragment"`

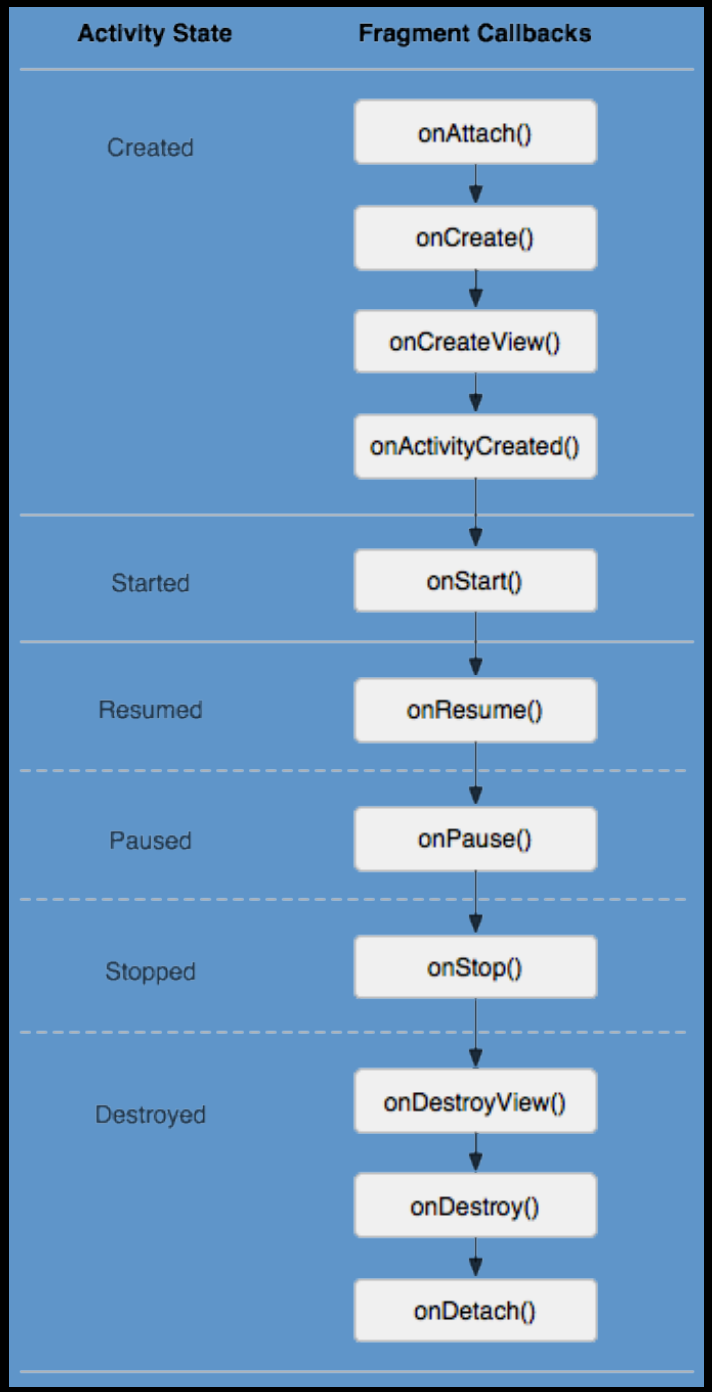
- **`<FrameLayout>`** is a view group that let's you specify an area in your layout where you can add a fragment later programmatically

Fragment Layout

- Use the **`FragmentTransaction`** class to programmatically interact with fragments
- The **`FragmentManager`** class manages an activity's fragments
  - **`getFragmentManager()`** gets reference to a fragment
  - **`beginTransaction()`** starts a fragment transaction
  - **`replace(id, fragment)`** replaces a fragment
  - **`commit()`** commits the changes

Back Stack

- With activities the back button goes to the previous activity
- The back button should do the same with fragments, show the previous fragment
- But if we just update the fragment with different data we won't have the previous fragment on the back stack
- Instead we'll create a new fragment and replace it each time



Fragment Lifecycle

Interface

- Because we want fragments to be reusable we want them to be as independent from their activity as possible
- We'll use an interface to decouple the fragment and the activity
- An interface defines the minimum requirements for one object to usefully talk to another

Interface

- Define the interface in fragment A (master)
- Create an instance of the interface and attach the activity to it in the `onAttach()` method
- The activity that hosts fragment A implements the listener and overrides the method that will handle passing data from fragment A to fragment B(detail)

Rotation

- Remember what happens in the Android lifecycle when the device is rotated?
- If the activity uses a fragment the fragment gets destroyed along with the activity so after rotation the fragment will default back to the first item
- Override `onSaveInstanceState()` to save the current state of the fragment
- In `onCreateView()` restore the saved state if there is one

Multiple Screen Support

- You can customize your apps appearance and flow based on different specifications
  - Screen size
  - Screen density
  - Orientation
  - Resolution
  - Density independent pixel(dp)
- Create layout specific folders for the configuration you want to target and provide modified layout files in those folders