

CS475/575

Handling Different Screen Sizes

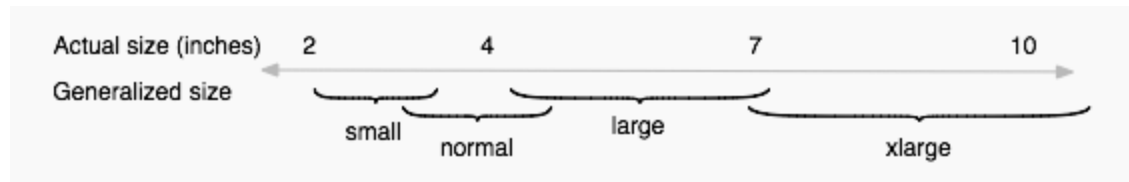
Fragments

# Handling different screen sizes

- Must design to 3.7" 5" 7" 10" etc.devices
- Use one layout?. Android will scale it but does not look “custom”
- Better – Develop layouts specific to particular screen size and density so:
  - Looks good on small screens
  - Takes advantage of tablets extra real estate
  - Is optimized for both Landscape and Portrait
- Android will pick the best one depending on size of target screen
- How?

# Handling different screen sizes

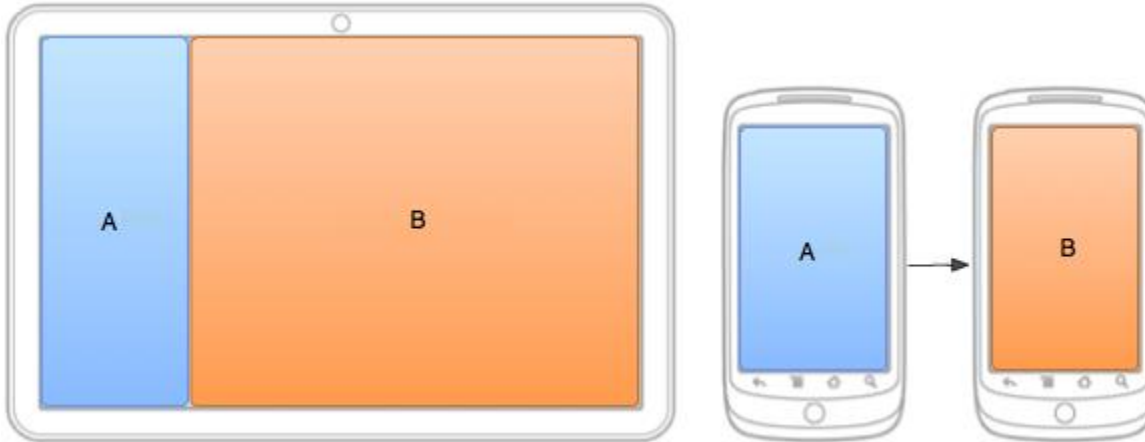
- You must create layouts specific to size and density targets (or specify width or height required)



- Android will choose from your layouts based on size and density of screen

+ layout	-- default
+ layout-port	-- portrait for any screen size
+ layout-xlarge	-- any orientation on xlarge screens
+ layout-xlarge-land	-- landscape on xlarge screens

# Now have layouts customized to Screen parameters



- Tablets have more real estate than handsets
- The layouts differ, but Left activity is functionally equivalent to right 2 activities **(3 activities total)**
- The java and XML will have a lot of common code, want to componentize it
- use Fragments

# Fragments

- Benefits
- Lifecycle
- Examples
- FragmentManager
- PreferenceFragment
- Summary

# What are Fragments

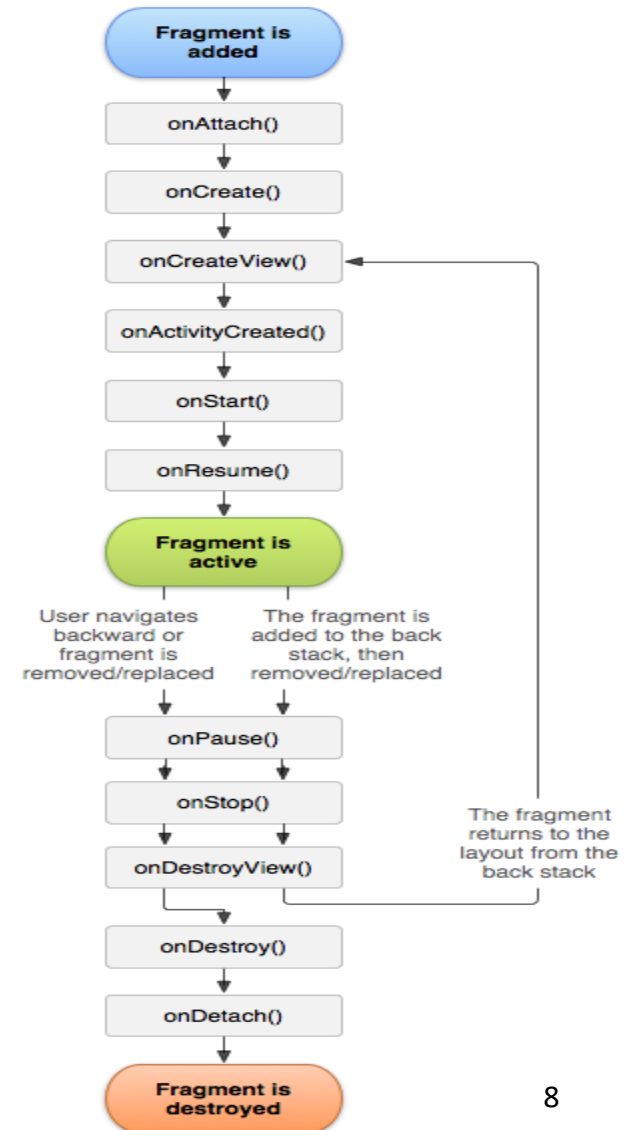
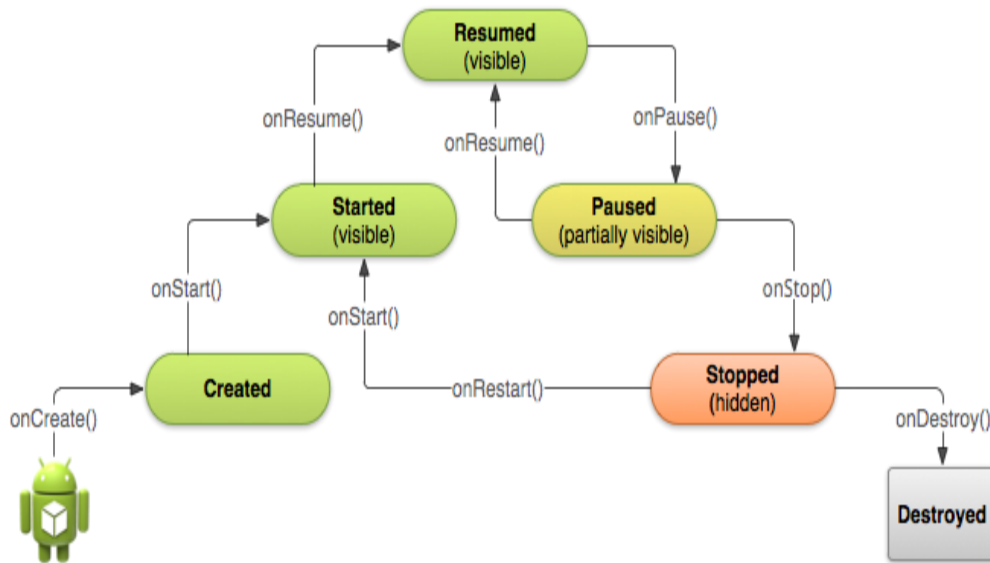
- Reusable components
- You still have to provide custom layouts (tablets show more than handsets, portrait verses landscape)
- Java code is where most reuse occurs
- Cost:
  - additional complexity
- Used in most professional apps because they must be optimized for multiple screen sizes

# Simple Example

## Fragments in XML

- Fragment defined in XML
- Populate fragment as needed
- Cannot remove fragment
- Show 5\_Fragment\_Static

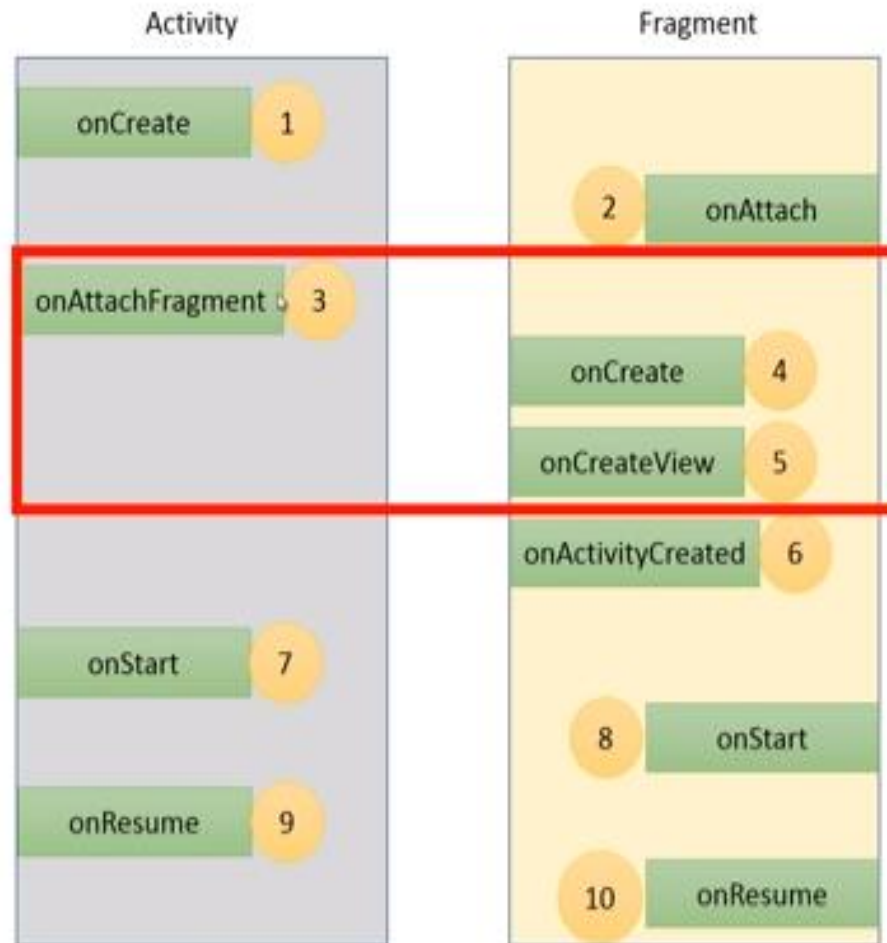
# Activity and Fragment Lifecycle



Fragment lives within the context of Host Activity.  
How do you reconcile these 2 lifecycles?



# Activity and Fragment Lifecycle Creation



**onAttach** is called after Fragment is associated with its Activity  
Gets a reference to the Activity object which can be used as Context

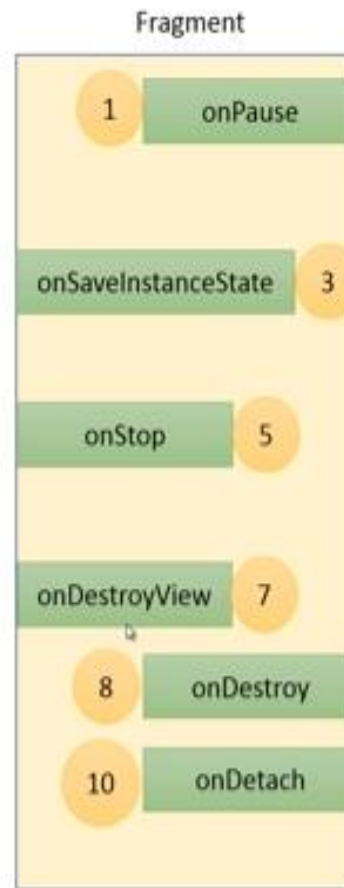
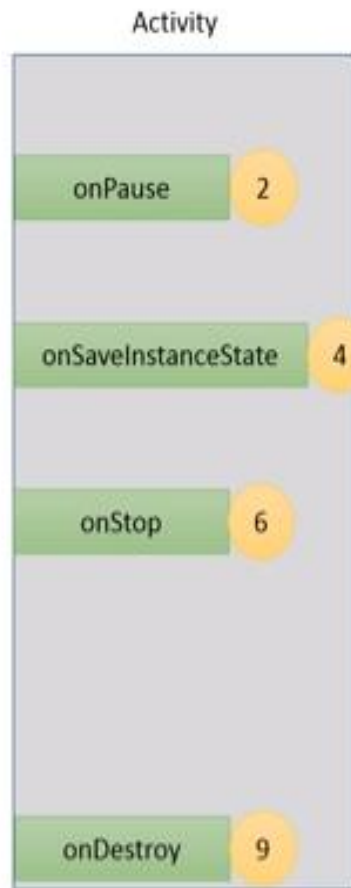
**onCreate** Don't use onCreate to access View hierarchy because Activity's onCreate may/may not be finished. Create background threads here for long running operations

**onCreateView** You are expected to return a View Hierarchy for your fragment

**onActivityCreated** Called after Activity onCreate has completed execution  
Use this method to access/modify UI elements



# Activity and Fragment Lifecycle Destruction



**onSaveInstanceState** Use this to save information inside a Bundle object

**onDestroyView** Called after the Fragment View Hierarchy is no longer accessible

**onDestroy** Called after fragment is not used. It still exists as a Java object attached to the Activity



**onDetach** Fragment is not tied to the Activity and does not have a View hierarchy



# Complex Example

## Dynamic Fragments using Java

- Usually more than 1 fragment
- FragmentManager
- Want all fragments or none (ACID)
- Do not comingle fragment libraries!

 android.support.v4.app.Fragment  
 android.app.Fragment

- Populate your fragments view properly

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
    //KP DO NOT FORGET THE FALSE  
    return inflater.inflate(R.layout.myfrag1, container, false);  
}
```

– See inflater.inflate documentation

- Show Fragment\_Dynamic

# PreferenceFragment

- Used in project2
- Supported from API 11 on. (Pre 11 use Preference Activity).
- Ensures standard look and feel for preference screens
- Also you get a framework that does much of the work
  - PrefActivity
    - onCreate PrefFragment and FragmentManager
  - PrefFragment loads the xml preferences defined in
  - Preference1.xml for UI (in res/xml)
- Easy notifications for rest of code via PreferenceChangeListener (interface and anonymous listener)
- Demo PrefFragment

# Why fragments are not emphasized in this class

- They really complicate development
- CPSC 475 develops to one size (600dp in portrait)
- HOWEVER – Fragments are essential when developing for multiple screen sizes
- AND they make the UI modular and portable

# Summary

- Fragments are VERY IMPORTANT
  - just not while learning a huge API
- Fragments Promote reusability
- Fragments - you will use them for commercial apps
- PreferenceFragment – Ties simple XML UI to shared preferences to give you professional look