Fragments

https://developer.android.com/guide/components/fragments

Cont ...

- A Fragment represents a behavior or a portion of user interface in a FragmentActivity.
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
- You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

Cont ...

- A fragment must always be hosted in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle.
- For example, when the activity is paused, so are all fragments in it, and when the activity is destroyed, so are all fragments.
- However, while an activity is running (it is in the resumed lifecycle state), you can manipulate each fragment independently, such as add or remove them.

Cont ...

- When you perform such a fragment transaction, you can also add it to a back stack that's managed by the activity—each back stack entry in the activity is a record of the fragment transaction that occurred.
- The back stack allows the user to reverse a fragment transaction (navigate backwards), by pressing the Back button.

Cont ...

- When you add a fragment as a part of your activity layout, it lives in a ViewGroup inside the activity's view hierarchy and the fragment defines its own view layout.
- You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element, or from your application code by adding it to an existing ViewGroup.

Design Philosophy

- Android introduced fragments in Android 3.0 (API level 11), primarily to support more dynamic and flexible UI designs on large screens, such as tablets.
- Because a tablet's screen is much larger than that of a handset, there's more room to combine and interchange UI components.
- Fragments allow such designs without the need for you to manage complex changes to the view hierarchy.

Cont ...

- By dividing the layout of an activity into fragments, you become able to modify the activity's appearance at runtime and preserve those changes in a back stack that's managed by the activity.
- They are now widely available through the fragment support library.

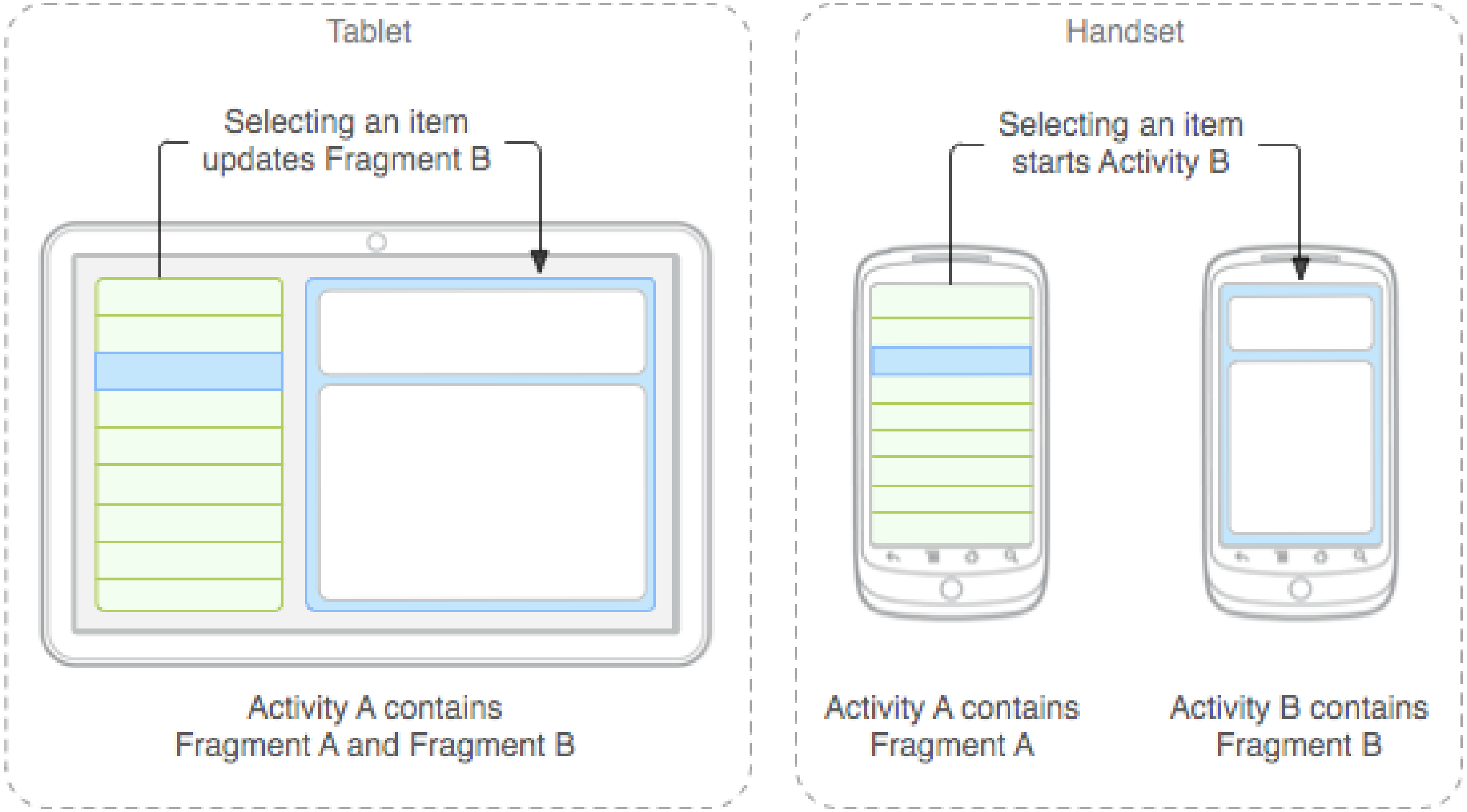
Cont ...

- You should design each fragment as a modular and reusable activity component.
- That is, because each fragment defines its own layout and its own behavior with its own lifecycle callbacks, you can include one fragment in multiple activities, so you should design for reuse and avoid directly manipulating one fragment from another fragment.
- This is especially important because a modular fragment allows you to change your fragment combinations for different screen sizes.

Cont ...

- When designing your application to support both tablets and handsets, you can reuse your fragments in different layout configurations to optimize the user experience based on the available screen space.

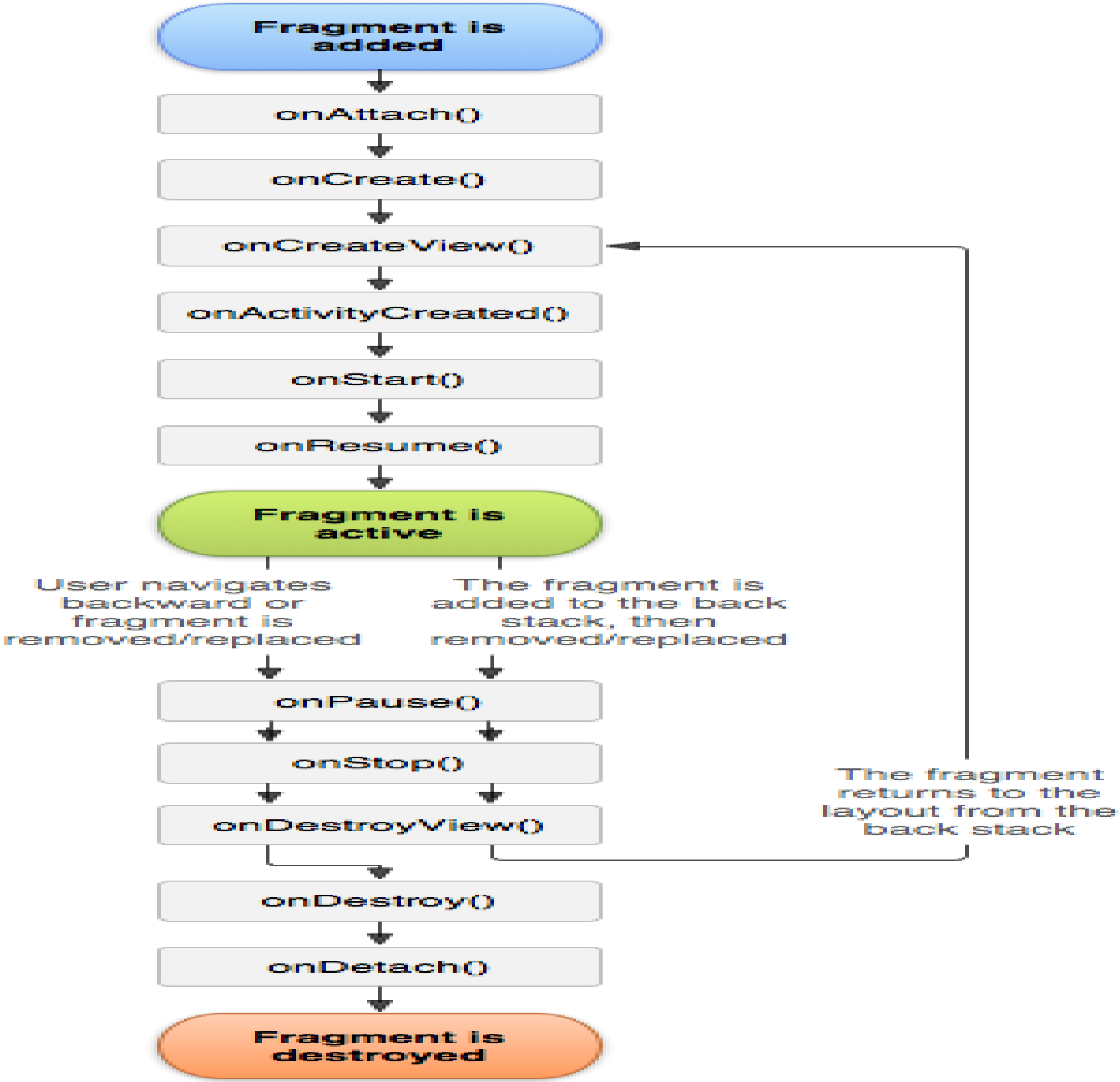
Cont ...


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

Creating a Fragment

- Usually, you should implement at least the following lifecycle methods:
  - OnCreate()
    - The system calls this when creating the fragment.
    - Within your implementation, you should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.
  - OnCreateView()
    - The system calls this when it's time for the fragment to draw its user interface for the first time.
  - OnPause()
    - The system calls this method as the first indication that the user is leaving the fragment

Cont ...



Fragment is added

onAttach()

onCreate()

onCreateView()

onActivityCreated()

onStart()

onResume()

Fragment is active

User navigates backward or fragment is removed/replaced

The fragment is added to the back stack, then removed/replaced

onPause()

onStop()

onDestroyView()

The fragment returns to the layout from the back stack

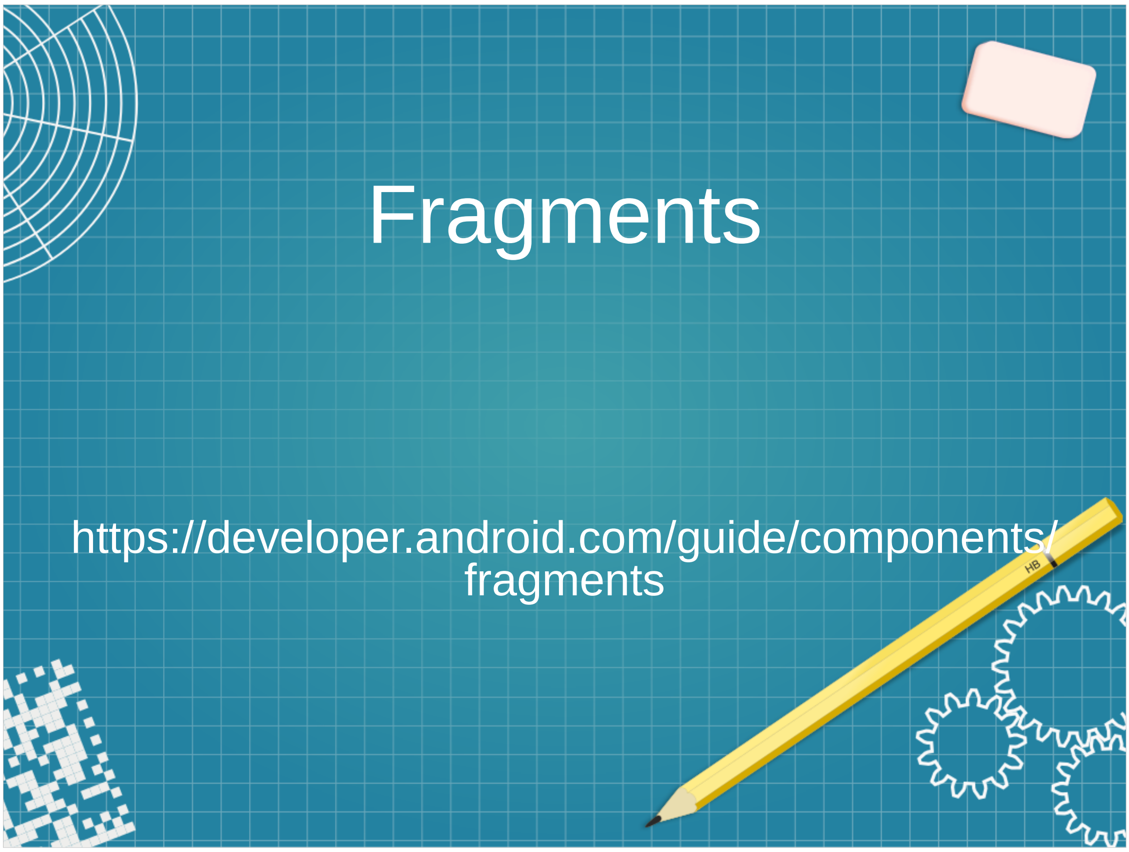
onDestroy()

onDetach()

Fragment is destroyed

Cont ...

- There are also a few subclasses that you might want to extend, instead of the base Fragment class:
  - DialogFragment
    - Displays a floating dialog.
  - ListFragment
    - Displays a list of items that are managed by an adapter (such as a SimpleCursorAdapter), similar to ListActivity.
  - PreferenceFragmentCompat
    - Displays a hierarchy of Preference objects as a list. This is used to create a settings screen for your application.



Fragments

https://developer.android.com/guide/components/fragments

Cont ...

- A Fragment represents a behavior or a portion of user interface in a FragmentActivity.
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
- You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

Cont ...

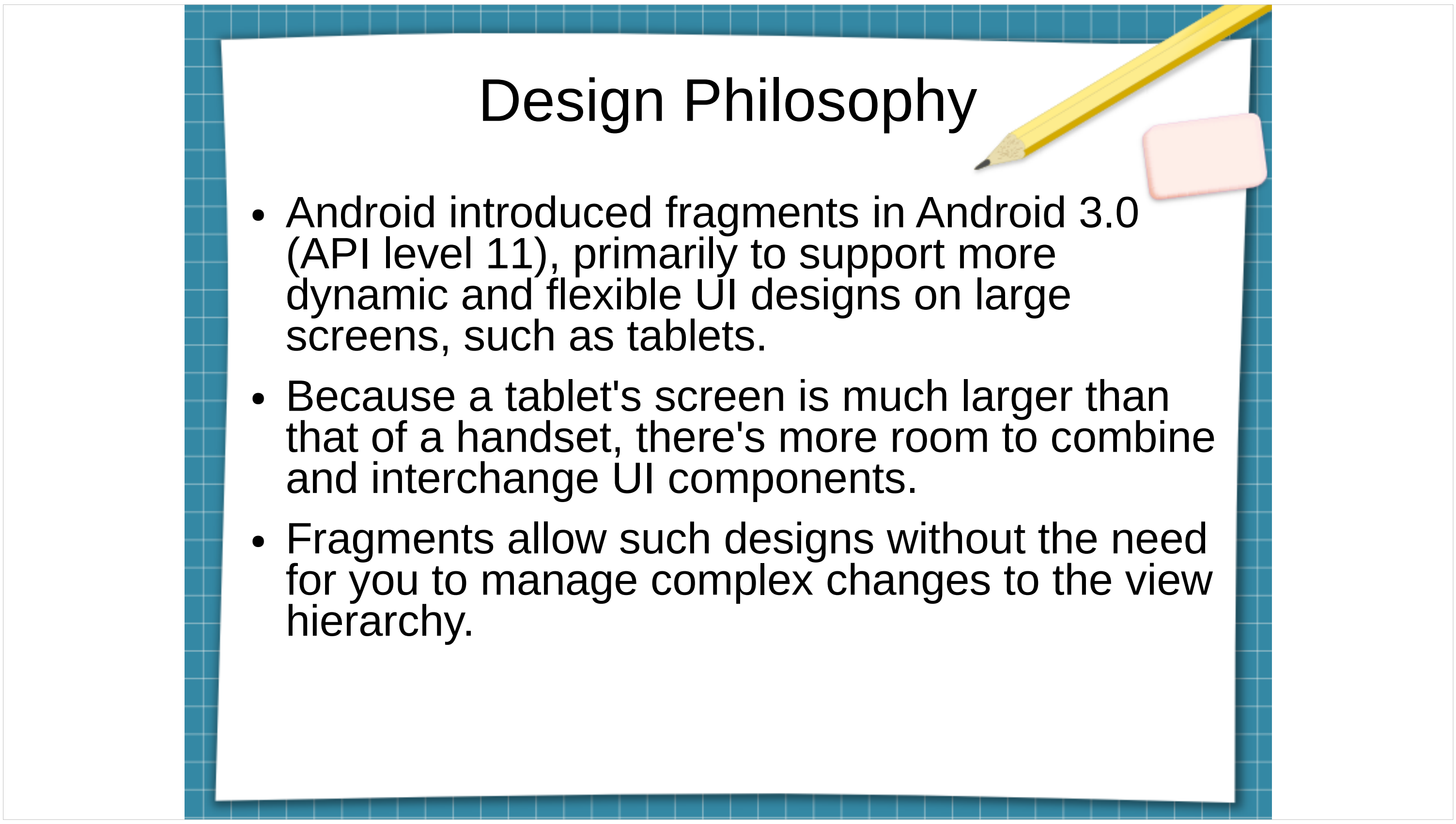
- A fragment must always be hosted in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle.
- For example, when the activity is paused, so are all fragments in it, and when the activity is destroyed, so are all fragments.
- However, while an activity is running (it is in the resumed lifecycle state), you can manipulate each fragment independently, such as add or remove them.

Cont ...

- When you perform such a fragment transaction, you can also add it to a back stack that's managed by the activity—each back stack entry in the activity is a record of the fragment transaction that occurred.
- The back stack allows the user to reverse a fragment transaction (navigate backwards), by pressing the Back button.

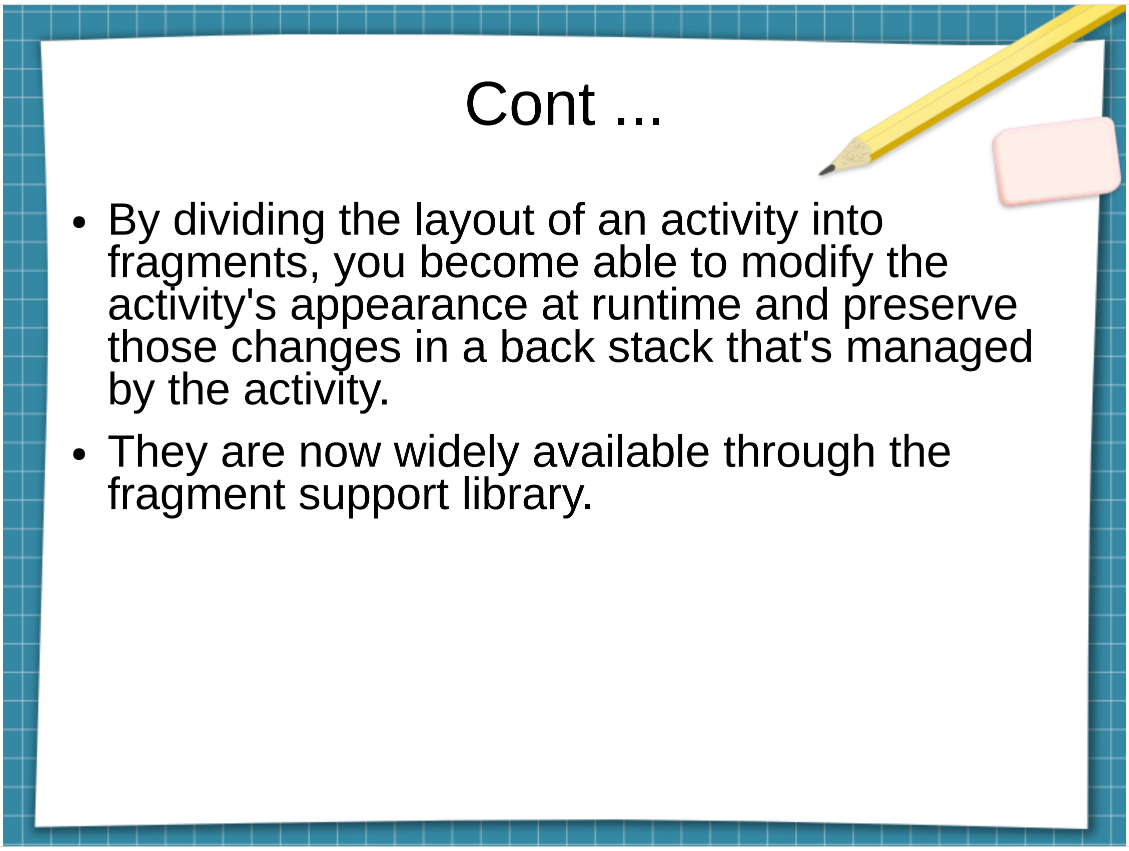
Cont ...

- When you add a fragment as a part of your activity layout, it lives in a ViewGroup inside the activity's view hierarchy and the fragment defines its own view layout.
- You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element, or from your application code by adding it to an existing ViewGroup.



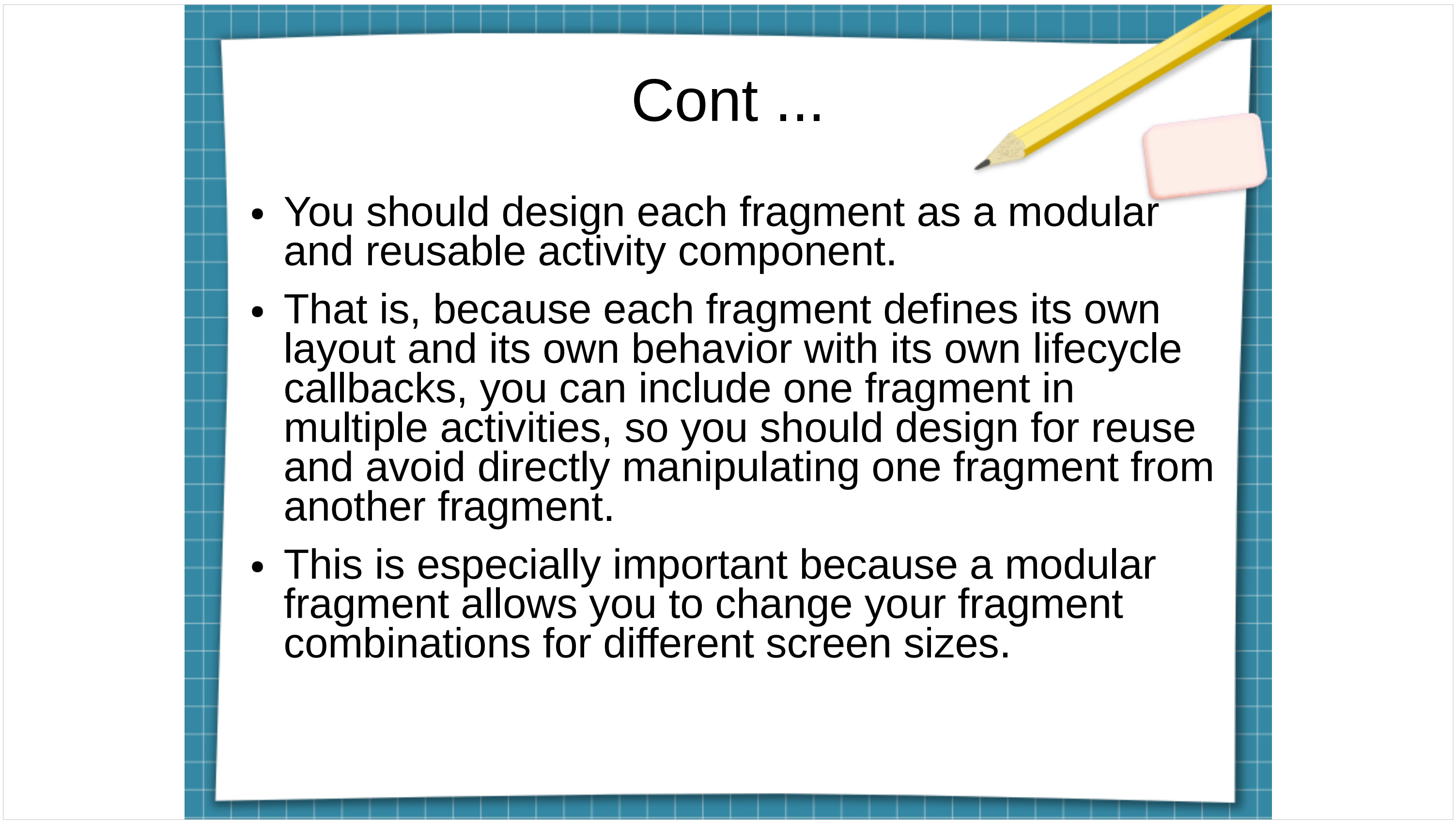
Design Philosophy

- Android introduced fragments in Android 3.0 (API level 11), primarily to support more dynamic and flexible UI designs on large screens, such as tablets.
- Because a tablet's screen is much larger than that of a handset, there's more room to combine and interchange UI components.
- Fragments allow such designs without the need for you to manage complex changes to the view hierarchy.



Cont ...

- By dividing the layout of an activity into fragments, you become able to modify the activity's appearance at runtime and preserve those changes in a back stack that's managed by the activity.
- They are now widely available through the fragment support library.



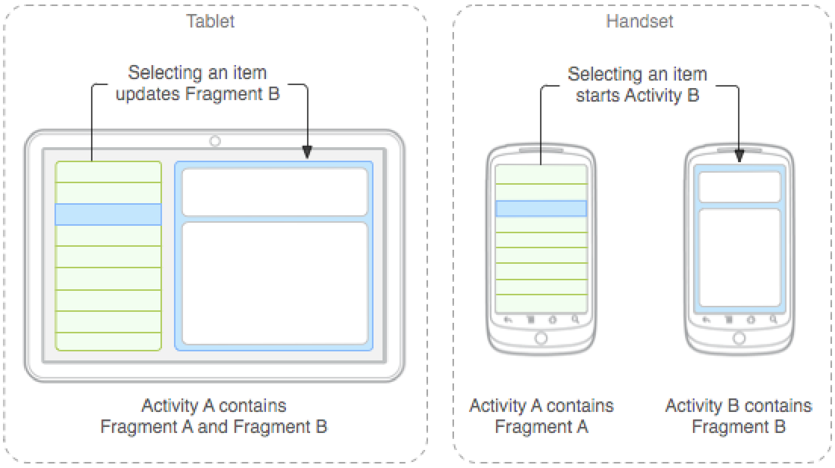
Cont ...

- You should design each fragment as a modular and reusable activity component.
- That is, because each fragment defines its own layout and its own behavior with its own lifecycle callbacks, you can include one fragment in multiple activities, so you should design for reuse and avoid directly manipulating one fragment from another fragment.
- This is especially important because a modular fragment allows you to change your fragment combinations for different screen sizes.

Cont ...

- When designing your application to support both tablets and handsets, you can reuse your fragments in different layout configurations to optimize the user experience based on the available screen space.

Cont ...



Tablet

Selecting an item updates Fragment B

Activity A contains Fragment A and Fragment B

Handset

Selecting an item starts Activity B

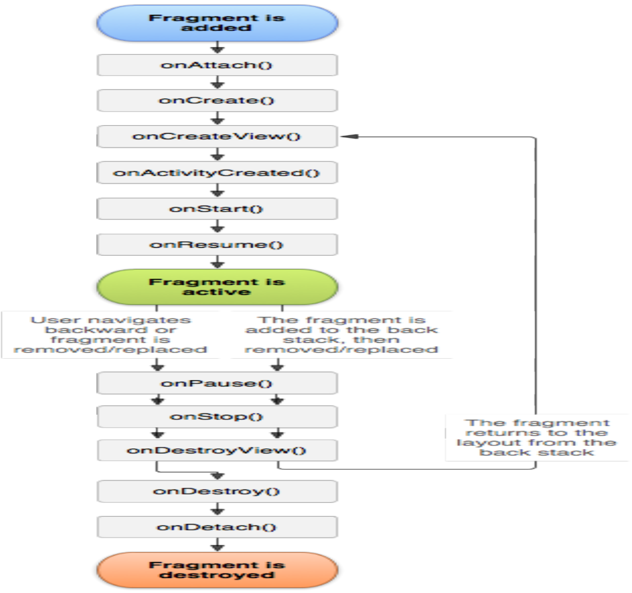
Activity A contains Fragment A

Activity B contains Fragment B

Creating a Fragment

- Usually, you should implement at least the following lifecycle methods:
  - OnCreate()
    - The system calls this when creating the fragment.
    - Within your implementation, you should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.
  - OnCreateView()
    - The system calls this when it's time for the fragment to draw its user interface for the first time.
  - OnPause()
    - The system calls this method as the first indication that the user is leaving the fragment

Cont ...



Fragment is added

onAttach()

onCreate()

onCreateView()

onActivityCreated()

onStart()

onResume()

Fragment is active

User navigates backward or fragment is removed/replaced

The fragment is added to the back stack, then removed/replaced

onPause()

onStop()

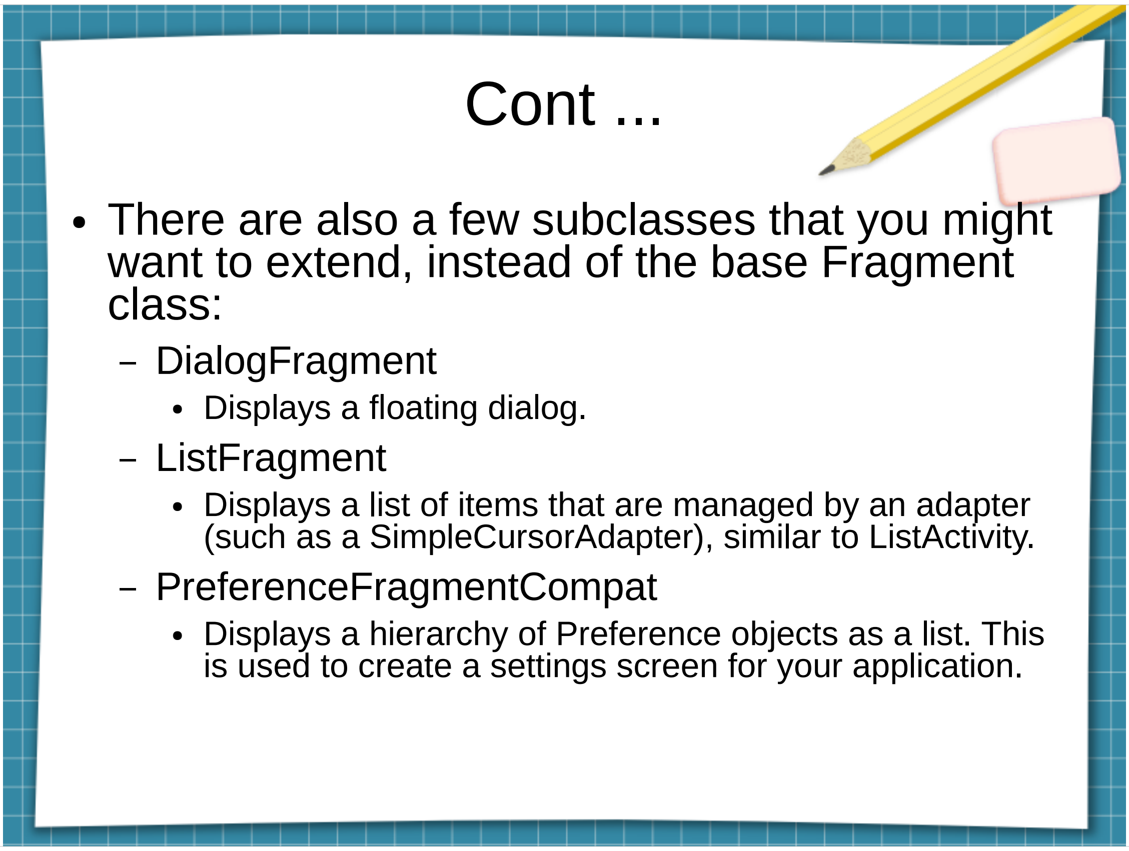
onDestroyView()

The fragment returns to the layout from the back stack

onDestroy()

onDetach()

Fragment is destroyed



Cont ...

- There are also a few subclasses that you might want to extend, instead of the base Fragment class:
  - DialogFragment
    - Displays a floating dialog.
  - ListFragment
    - Displays a list of items that are managed by an adapter (such as a SimpleCursorAdapter), similar to ListActivity.
  - PreferenceFragmentCompat
    - Displays a hierarchy of Preference objects as a list. This is used to create a settings screen for your application.