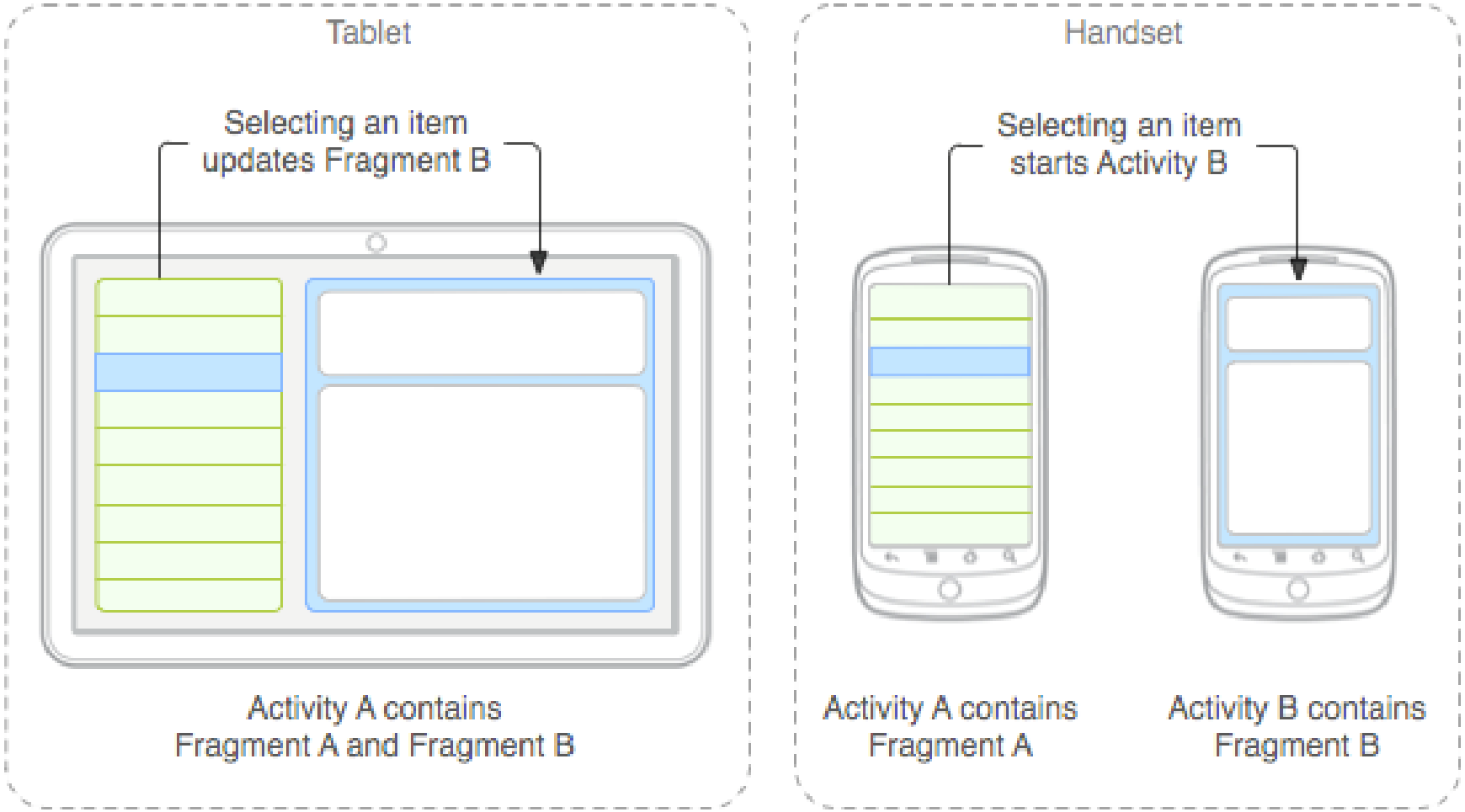
Fragments

Lecturer: Yehuda Rozilyo

http://developer.android.com/guide/components/fragments.html


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

- A Fragment represents a behavior or a portion of user interface in an Activity.
- You can combine multiple fragments in a single activity.
- fragment can be reused in multiple activities.
- A fragment must always be embedded in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle.
- Activity manages fragments using back stack
- each back stack entry in the activity is a record of the fragment transaction that occurred.

Fragment

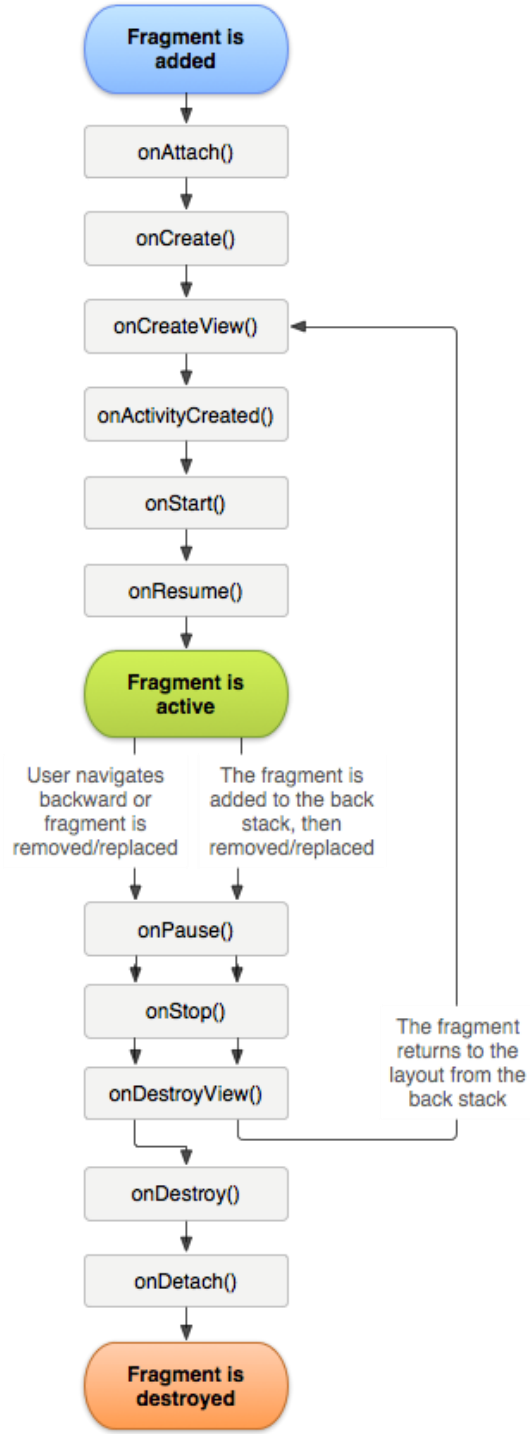
You can think of a fragment as:

- a modular section of an activity
- which has its own lifecycle
- receives its own input events
- which you can add or remove while the activity is running

Creating Fragment

- To create a fragment, you must create a subclass of Fragment (or an existing subclass of it).

- Fragment contains callback methods similar to an activity, such as onCreate(), onStart(), onPause(), and onStop().


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

Fragment UI

- A fragment is usually used as part of an activity's user interface and contributes its own layout to the activity.
- To provide a layout for a fragment, you must implement the onCreateView() callback method
- Android system calls this method when it's time for the fragment to draw its layout.
- onCreateView() must return a View that is the root of the fragment's layout.

Fragment UI

```
class ExampleFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.example_fragment,
container, false)
    }
}
```

Adding Fragment to Activity

- Declare the fragment inside the activity's layout file
- Programmatically

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

Add a Fragment in Layout

- The android:name attribute in the <fragment> specifies the Fragment class to instantiate in the layout.
- When the system creates this activity layout, it instantiates each fragment specified in the layout and calls the onCreateView() method for each one, to retrieve each fragment's layout.
- The system inserts the View returned by the fragment directly in place of the <fragment> element.

Getting the Fragment

To get the fragment within the Activity:

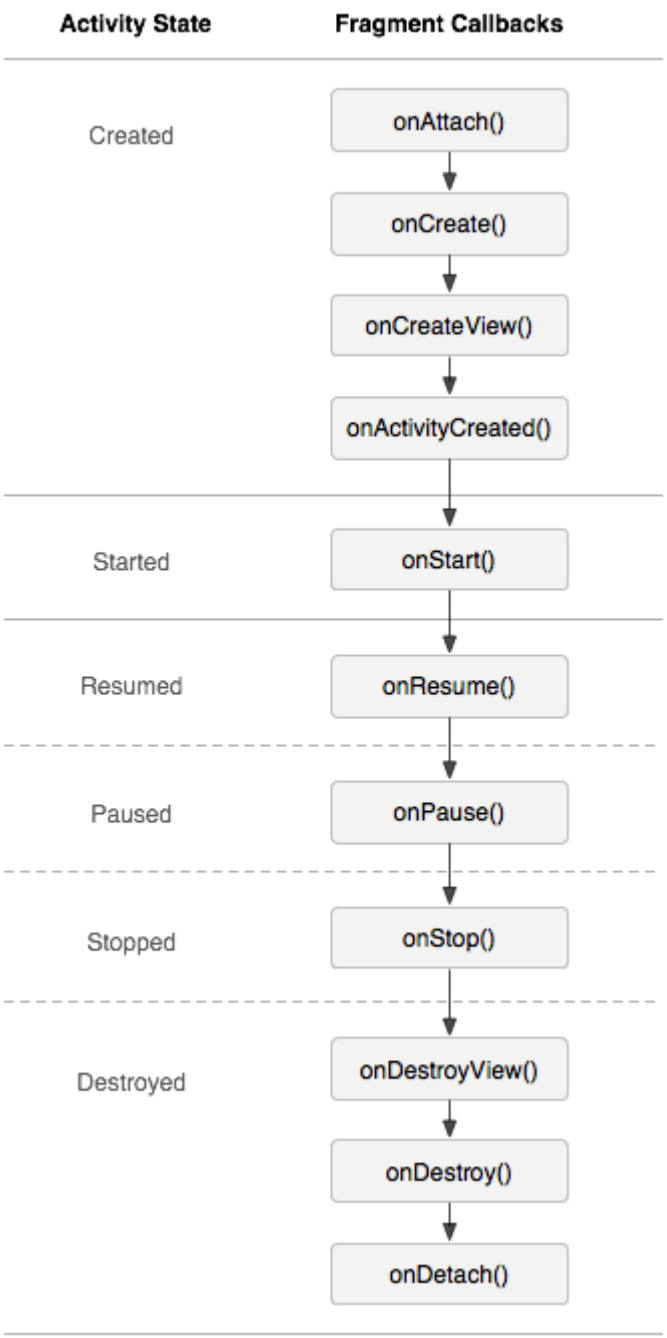
```
val frag =
supportFragmentManager.findFragmentById(R.id.my_f
ragment) as MyFragment?
```

Fragment Lifecycle

- Like an activity, a fragment can exist in three states:
  - ***Resumed:*** The fragment is visible in the running activity.
  - ***Paused:*** Another activity is in the foreground and has focus, but the activity in which this fragment lives is still visible (the foreground activity is partially transparent or doesn't cover the entire screen).
  - ***Stopped:*** The fragment is not visible. Either the host activity has been stopped or the fragment has been removed from the activity but added to the back stack.

Managing Fragment Lifecycle

- onAttach(): Called when the fragment has been associated with the activity (the Activity is passed in here).
- onCreateView(): Called to create the view hierarchy associated with the fragment.
- onActivityCreated(): Called when the activity's onCreate() method has returned.
- onDestroyView(): Called when the view hierarchy associated with the fragment is being removed.
- onDetach(): Called when the fragment is being disassociated from the activity.


Activity State
Fragment Callbacks
Created
onAttach()
onCreate()
onCreateView()
onActivityCreated()
Started
onStart()
Resumed
onResume()
Paused
onPause()
Stopped
onStop()
Destroyed
onDestroyView()
onDestroy()
onDetach()

Fragment Example

Add Fragment Programmatically

- At any time while your activity is running, you can add fragments to your activity layout with this steps:
  - specify a ViewGroup in which to place the fragment.
  - Make fragment transactions in your activity (such as add, remove, or replace a fragment)

The Code

```
val fragmentTransaction = supportFragmentManager.beginTransaction()
val fragment = ExampleFragment()
fragmentTransaction.add(R.id.fragment_container, fragment)
fragmentTransaction.commit()
```

Fragment and Activity

The most significant difference in lifecycle between an activity and a fragment is how one is stored in its respective back stack.

- An activity is placed into a back stack of activities
- A fragment is placed into a back stack managed by the host activity only when you explicitly request that the instance be saved by calling addToBackStack() during a transaction that adds the fragment.

Back Button

- If you add multiple changes to the transaction (such as another add() or remove()) and call addToBackStack(), then all changes applied before you call commit() are added to the back stack as a single transaction and the *Back* button will reverse them all together.

- Otherwise the Back button will remove the hosting Activity from the Activity stack

- to programmatically remove the last item from backstack:

  ```
  getFragmentManager().popBackStack();
  ```

Home Button

```
 supportActionBar?.setDisplayHomeAsUpEnabled(true)

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        android.R.id.home -> {
            switchToListFragment()
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}
```

Tabs

Action Bar Tabs

- Action bar tabs offer users a familiar interface for navigating between and identifying sibling screens in your app.
- To create tabs using ActionBar, you need:
  - enable NAVIGATION_MODE_TABS
  - create several instances of ActionBar.Tab
  - supply an implementation of the ActionBar.TabListener interface for each one.

Tabs...

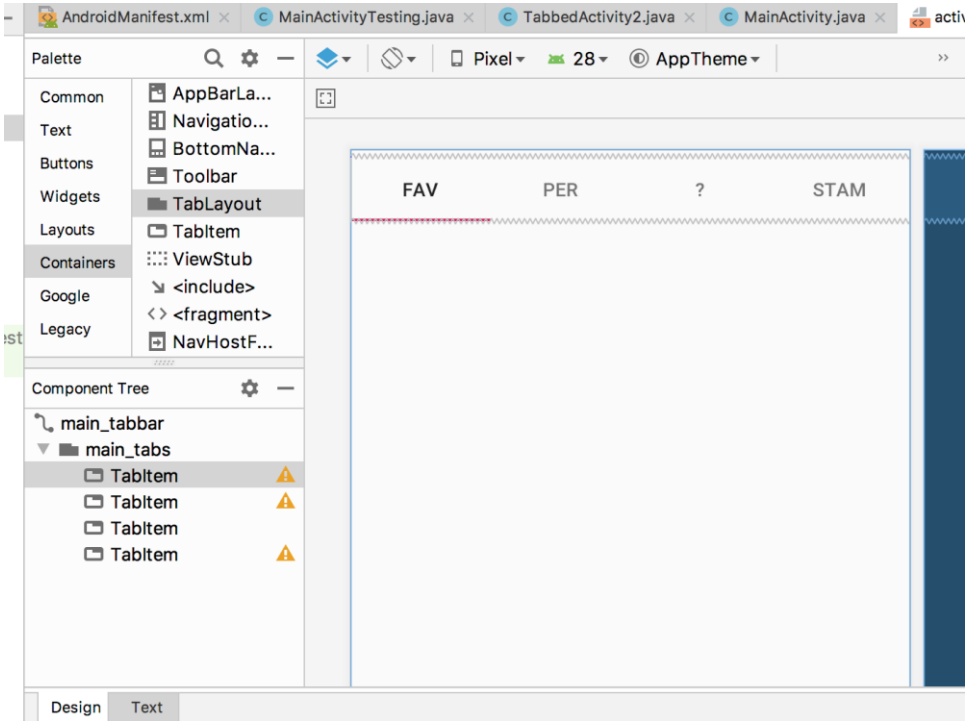
```
val actionBar = supportActionBar
// Specify that tabs should be displayed in the action bar.
actionBar?.navigationMode = ActionBar.NAVIGATION_MODE_TABS

// Add tabs, specifying the tab's text and TabListener
val tab = actionBar?.newTab()?.setText("List")?.setTabListener(object :
ActionBar.TabListener {
    override fun onTabUnselected(tab: ActionBar.Tab?, ft: FragmentTransaction?) {
        ft?.hide(myListFragment)
    }

    override fun onTabSelected(tab: ActionBar.Tab?, ft: FragmentTransaction?) {
        ft?.show(myListFragment)
    }

    override fun onTabReselected(tab: ActionBar.Tab?, ft: FragmentTransaction?) {
        // Handle tab reselection if
```

ViewPager & Tabs

View Pager Tabs

Another way to implement tabs is using the view pager

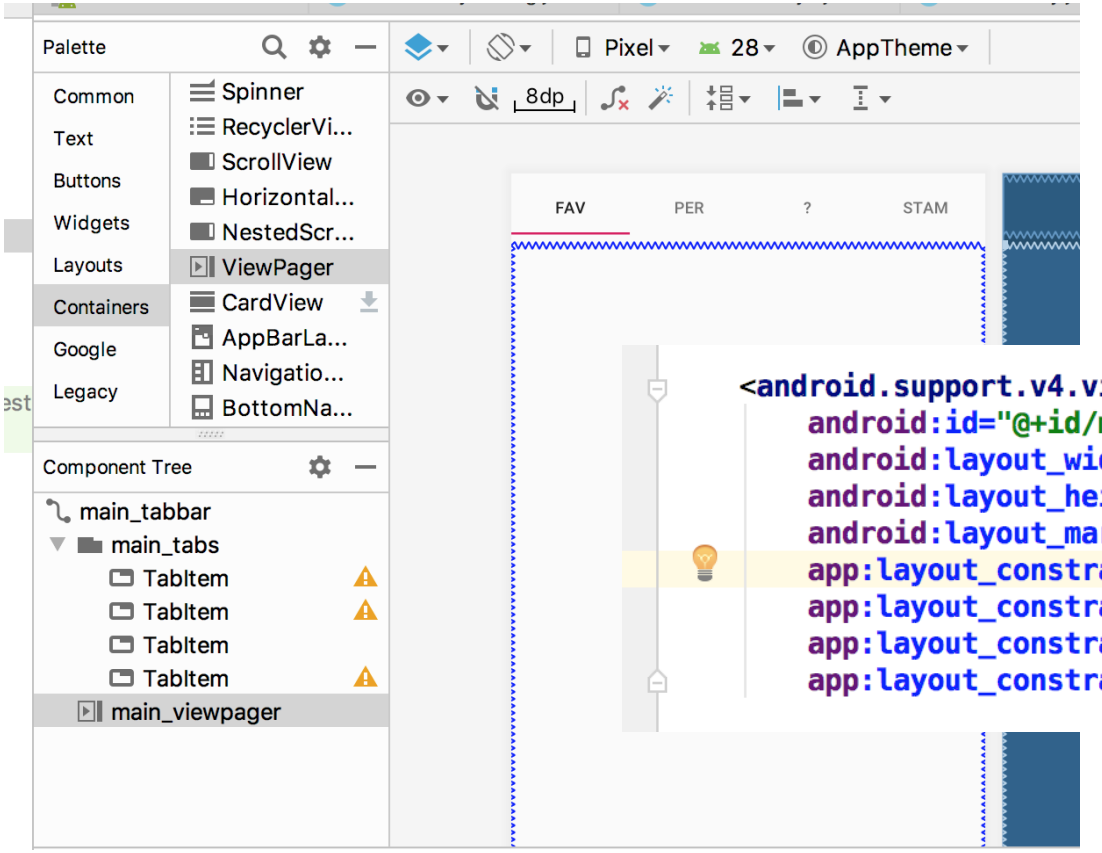
first add the tabs layout to the page




```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns
    xmlns:app="http://schemas.android.com/apk/res-
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main_tabbar"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivityTesting">

    <android.support.design.widget.TabLayout
        android:id="@+id/main_tabs"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="pare
        app:layout_constraintTop_toTopOf="parent">

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="FAV" />

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="PER" />

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="?" />

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="STAM" />
    </android.support.design.widget.TabLayout>

</android.support.constraint.ConstraintLayout>
```

ViewPager

Add ViewPager to the page




```
<android.support.v4.view.ViewPager
    android:id="@+id/main_viewpager"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginTop="9dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/main_tabs" />
```

ViewPager Adapter

```
class ViewPagerAdapter(fm: FragmentManager) :
FragmentPagerAdapter(fm,
BEHAVIOR_RESUME_ONLY_CURRENT_FRAGMENT) {

    override fun getItem(position: Int): Fragment {
        // getItem is called to instantiate the fragment
for the given page.
        return PlaceholderFragment.newInstance(position +
1)
    }

    override fun getCount(): Int {
        // Show 4 total pages.
        return 4
    }
}
```

Connect Viewpager and Adapter

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main_testing)
        val tabs: TabLayout = findViewById(R.id.main_tabs)
        val viewPager: ViewPager = findViewById(R.id.main_viewpager)

        val adapter = ViewPagerAdapter(supportFragmentManager)
        viewPager.adapter = adapter
        viewPager.addOnPageChangeListener(TabLayout.TabLayoutOnPageChangeListener(tabs))
        tabs.addOnTabSelectedListener(TabLayout.ViewPagerOnTabSelectedListener(viewPager))
    }
}
```

Saving State

Saving Fragment State

- By default Android will save the activity and Fragments current display state and will restore it on restart of the activity
- To save additional items use the callback:

```
override fun onViewStateRestored(savedInstanceState: Bundle?) {
  super.onViewStateRestored(savedInstanceState)
}
```

- Restore state:

```
override fun onSaveInstanceState(outState: Bundle) {
  super.onSaveInstanceState(outState)
}
```