SESSION 04

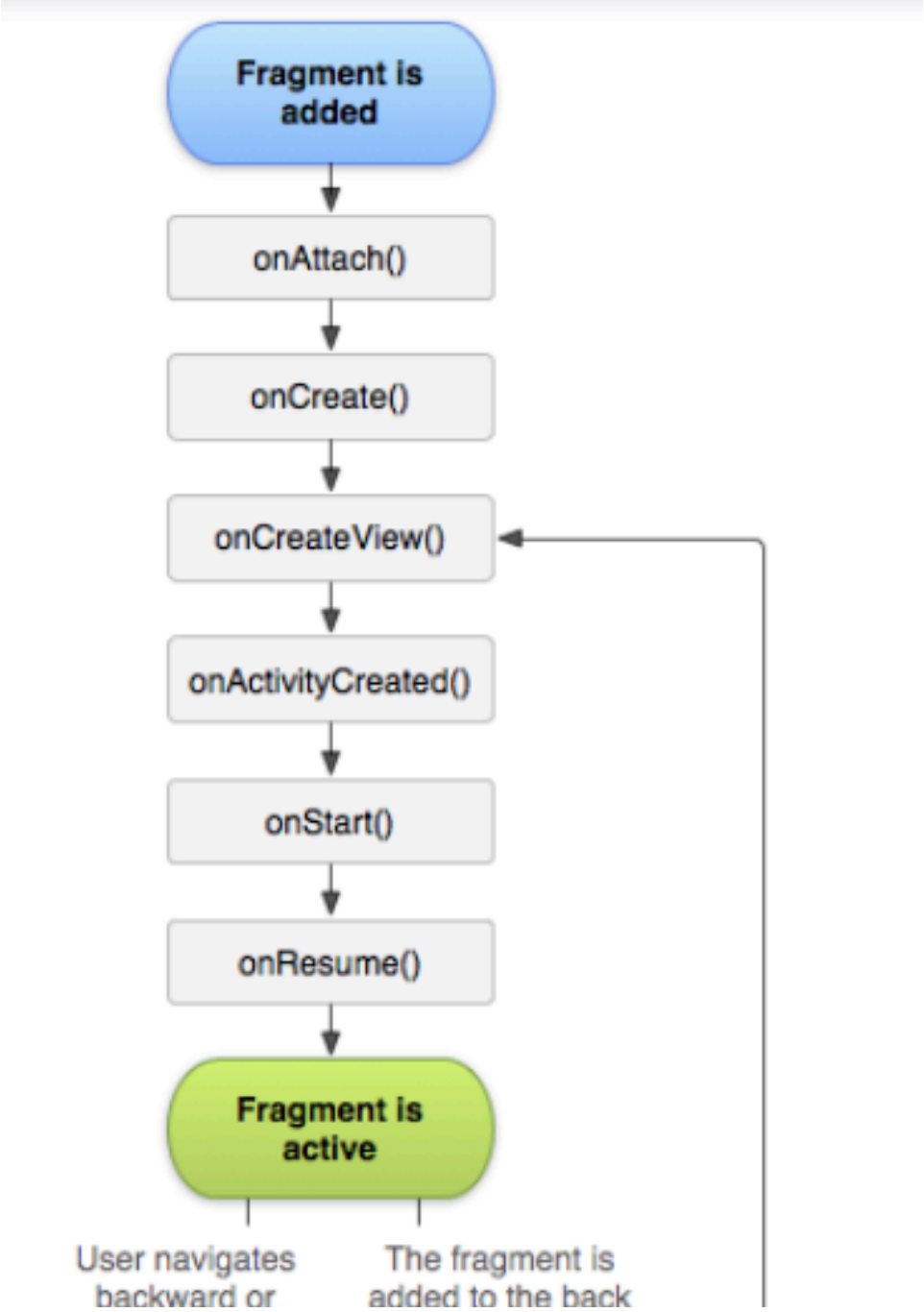
export PATH=$PATH:/Users/sajib/Library/Android/sdk/platform-tools/

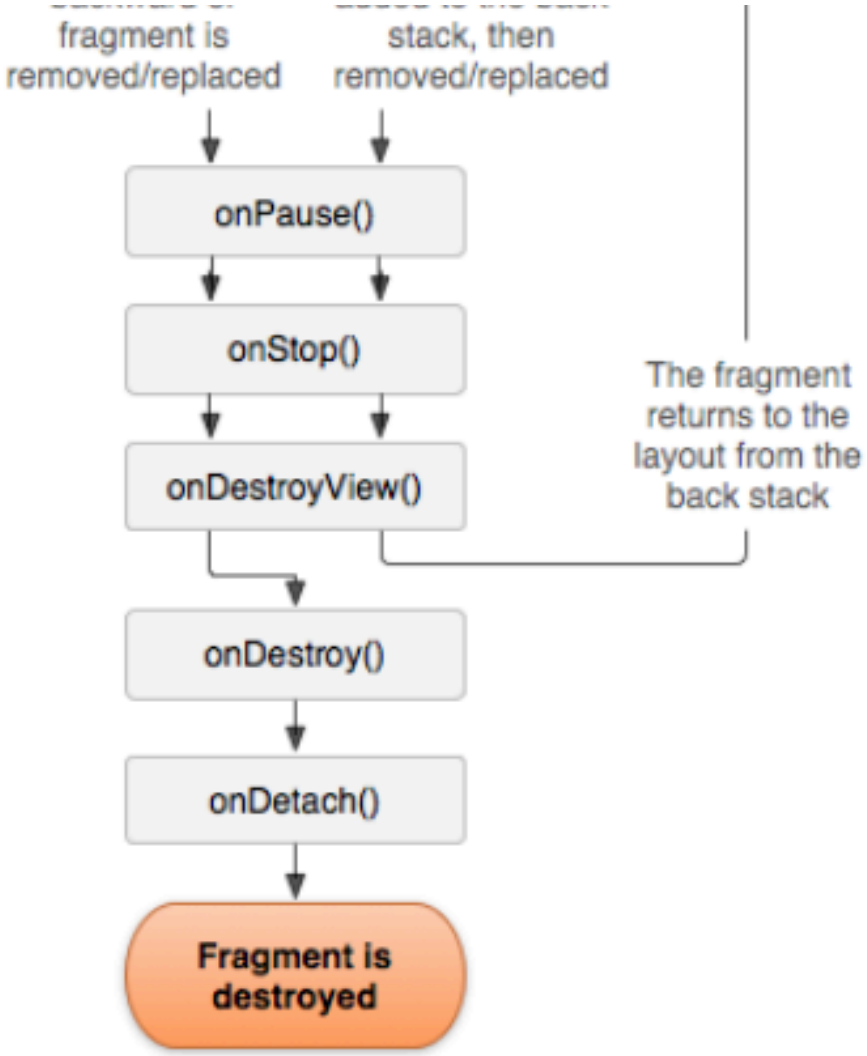
 # Fragments

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).

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. 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.

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.





Fragment Lifecycle

- onAttach(Context) - [https://developer.android.com/reference/android/app/Fragment#onAttach(android.app.Activity)] - Called when a fragment is first attached to its context. onCreate(android.os.Bundle) will be called after this. If you override this method you *must* call through to the superclass implementation.
- onCreate(Bundle) - [https://developer.android.com/reference/android/app/Fragment#onCreate(android.os.Bundle)] - Called to do initial creation of a fragment. This is called

after onAttach(android.app.Activity) and before onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle), but is not called if the fragment instance is retained across Activity re-creation (see setRetainInstance(boolean)). Note that this can be called while the fragment's activity is still in the process of being created. As such, you can not rely on things like the activity's content view hierarchy being initialized at this point. If you want to do work once the activity itself is created, see onActivityCreated(android.os.Bundle). If your app's targetSdkVersion is Build.VERSION_CODES.M or lower, child fragments being restored from the savedInstanceState are restored after onCreate returns. When targeting Build.VERSION_CODES.N or above and running on an N or newer platform version they are restored by Fragment.onCreate.

- onCreateView(LayoutInflater, ViewGroup, Bundle) - [https://developer.android.com/reference/android/app/Fragment#onCreateView(android.view.LayoutInflater,%20android.view.ViewGroup,%20android.os.Bundle)] - Called to have the fragment instantiate its user interface view. This is optional, and non-graphical fragments can return null (which is the default implementation). This will be called between onCreate(android.os.Bundle) and onActivityCreated(android.os.Bundle). If you return a View from here, you will later be called in onDestroyView() when the view is being released.
- onActivityCreated(Bundle) - [https://

developer.android.com/reference/android/app/Fragment#onActivityCreated(android.os.Bundle)] - Called when the fragment's activity has been created and this fragment's view hierarchy instantiated. It can be used to do final initialization once these pieces are in place, such as retrieving views or restoring state. It is also useful for fragments that use setRetainInstance(boolean) to retain their instance, as this callback tells the fragment when it is fully associated with the new activity instance. This is called after onCreateView(LayoutInflater, ViewGroup, Bundle) and before onViewStateRestored(android.os.Bundle). If you override this method you *must* call through to the superclass implementation.

- onStart() - [https://developer.android.com/reference/android/app/Fragment#onStart()] - Called when the Fragment is visible to the user. This is generally tied to Activity#onStart() of the containing Activity's lifecycle. If you override this method you *must* call through to the superclass implementation.
- onResume() - [https://developer.android.com/reference/android/app/Fragment#onResume()] - Called when the fragment is visible to the user and actively running. This is generally tied to Activity#onResume() of the containing Activity's lifecycle. If you override this method you *must* call through to the superclass implementation.
- onPause() - [https://developer.android.com/reference/android/app/Fragment#onPause()] - Called when the Fragment is no longer resumed. This is generally tied to Activity#onPause() of the

containing Activity's lifecycle. If you override this method you *must* call through to the superclass implementation.

- onStop - [https://developer.android.com/reference/android/app/Fragment#onStop()] - Called when the Fragment is no longer started. This is generally tied to Activity#onStop() of the containing Activity's lifecycle. If you override this method you *must* call through to the superclass implementation.
- onDestroyView() - [https://developer.android.com/reference/android/app/Fragment#onDestroyView()] - Called when the view previously created by onCreateView(LayoutInflater, ViewGroup, Bundle) has been detached from the fragment. The next time the fragment needs to be displayed, a new view will be created. This is called after onStop() and before onDestroy(). It is called *regardless* of whether onCreateView(LayoutInflater, ViewGroup, Bundle) returned a non-null view. Internally it is called after the view's state has been saved but before it has been removed from its parent. If you override this method you *must* call through to the superclass implementation.
- onDestroy - [https://developer.android.com/reference/android/app/Fragment#onDestroy()] - Called when the fragment is no longer in use. This is called after onStop() and before onDetach(). If you override this method you *must* call through to the superclass implementation.
- onDetach() - [https://developer.android.com/reference/android/app/Fragment#onDetach()] - Called when the fragment is no longer attached to its activity. This is called after onDestroy(), except in the cases where the fragment instance is retained

across Activity re-creation (see setRetainInstance(boolean)), in which case it is called after onStop(). If you override this method you *must* call through to the superclass implementation.
- onViewStateRestored(Bundle) - [https://developer.android.com/reference/android/app/Fragment#onViewStateRestored(android.os.Bundle)] - Called when all saved state has been restored into the view hierarchy of the fragment. This can be used to do initialization based on saved state that you are letting the view hierarchy track itself, such as whether check box widgets are currently checked. This is called after onActivityCreated(android.os.Bundle) and before onStart(). If you override this method you *must* call through to the superclass implementation.

 # Fragment vs FragmentActivity

[https://stackoverflow.com/questions/10609268/what-is-the-difference-between-fragment-and-fragmentactivity]

- Fragments don't subclass the Context class. Therefore you have to use the getActivity() method to get the parent activity.
- It is possible to define in the layout file of an activity that it contains fragments (static definition). You can also modify the fragments of an activity at runtime (dynamic definition).

- To define a new fragment you either extend the android.app.Fragment class or one of its subclasses. Subclasses are for example, ListFragment, DialogFragment, Preference Fragment or WebViewFragment.
- To increase reuse of fragments, they should not directly communicate with each other. Every communication of the fragments should be done via the host activity. For this purpose a fragment should define an interface as an inner type. The fragment requires that the activity, which uses it, must implement this interface. This way you avoid that the fragment has any knowledge about the activity which uses it. In its onAttach() method it can check if the *activity* correctly implements this interface.
-

```
<fragment
    android:id="@+id/fragment_one"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    class="com.debacharya.androidbasics.session04.FragmentOne"
    tools:layout="@layout/fragment_one">

</fragment>
```

```java
@Override
protected void onStart() {
    super.onStart();

    FragmentTwo fragmentTwo = new FragmentTwo();
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

    fragmentTransaction.add(R.id.ll_main, fragmentTwo);
    fragmentTransaction.commit();
}
```

```java
@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater,
                         @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_one, container, attachToRoot: false);
}

@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
}

@Override
public void onStart() {
    super.onStart();

    TextView textView = (TextView) getView().findViewById(R.id.tv_one);

    textView.setText("Fragment One");
}
```

```java
@Override
public void onStart() {
    super.onStart();

    Button button = (Button) getView().findViewById(R.id.fragmenttwo_button);

    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(getContext(), text: "Clicked!", Toast.LENGTH_SHORT).show();
        }
    });
}
```

```java
@Override
public void listen(String value) {
    Toast.makeText(getApplicationContext(), value, Toast.LENGTH_SHORT).show();

    FragmentThree newFragmentThree = new FragmentThree();
    Bundle bundle = new Bundle();

    bundle.putString(Keystore.KEY_ONE, value);
    newFragmentThree.setArguments(bundle);

    getSupportFragmentManager() FragmentManager
        .beginTransaction() FragmentTransaction
        .replace(R.id.fl_fragmentthree, newFragmentThree)
        .commit();
}
```