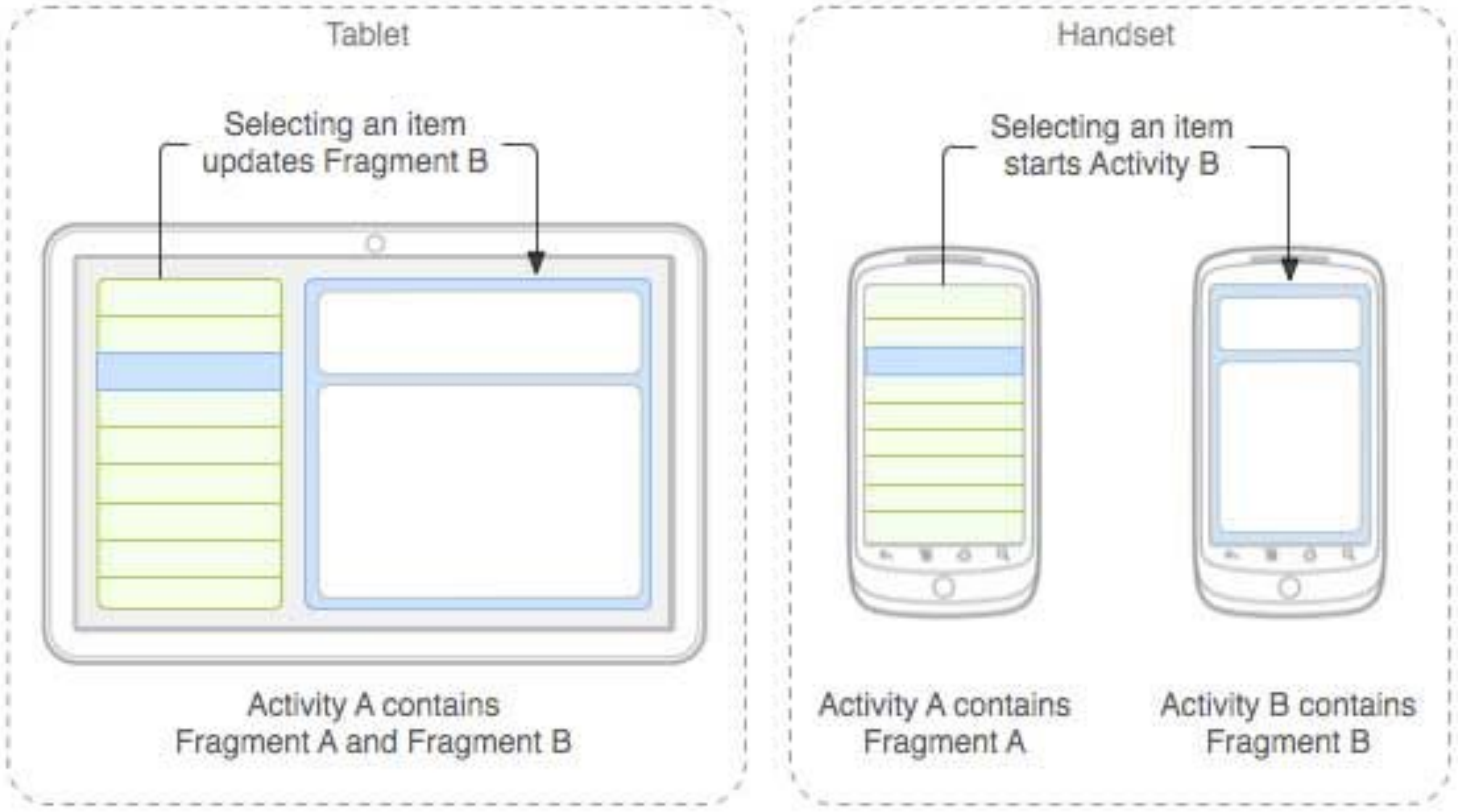
# Fragments in android

By Michael Sadgun Rao Kona,

# Fragment

- A `Fragment` represents a reusable portion of  app's UI.
- A fragment defines and manages its own layout.
- Fragment has its own lifecycle
- Fragment can handle its own input events.
- Fragments cannot live on their own--they must be *hosted* by an activity or another fragment.
- The fragment's view hierarchy becomes part of, or *attaches to*, the host's view hierarchy.


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

Source:- https://www.tutorialspoint.com/android/android_fragments.htm

## modularity

- Divide the UI into discrete chunks ( FRAGMENTS)
- Dividing UI into fragments makes it easier to modify your activity's appearance at runtime.
- Fragments can be activated and deactivated during runtime.
- Fragments are introduced in HoneyComb API 11.
- Activity - ideal place to put global elements around app's UI.
    - Ex:-navigation drawer.
- Fragments - better suited to define and manage the UI of a single screen or portion of a screen.

- While the activity is in the `STARTED` lifecycle state or higher, fragments can be added, replaced, or removed.
- Keep a record of these changes in a back stack (that is managed by the activity), allowing the changes to be reversed.
- Multiple instances of the same fragment class can be used
  - within the same activity,
  - in multiple activities,
  - or even as a child of another fragment.
- Note:-
  - Provide a fragment with the logic necessary to manage its own UI.
  - Avoid depending on or manipulating one fragment from another.

# Example - app that responds to various screen sizes.

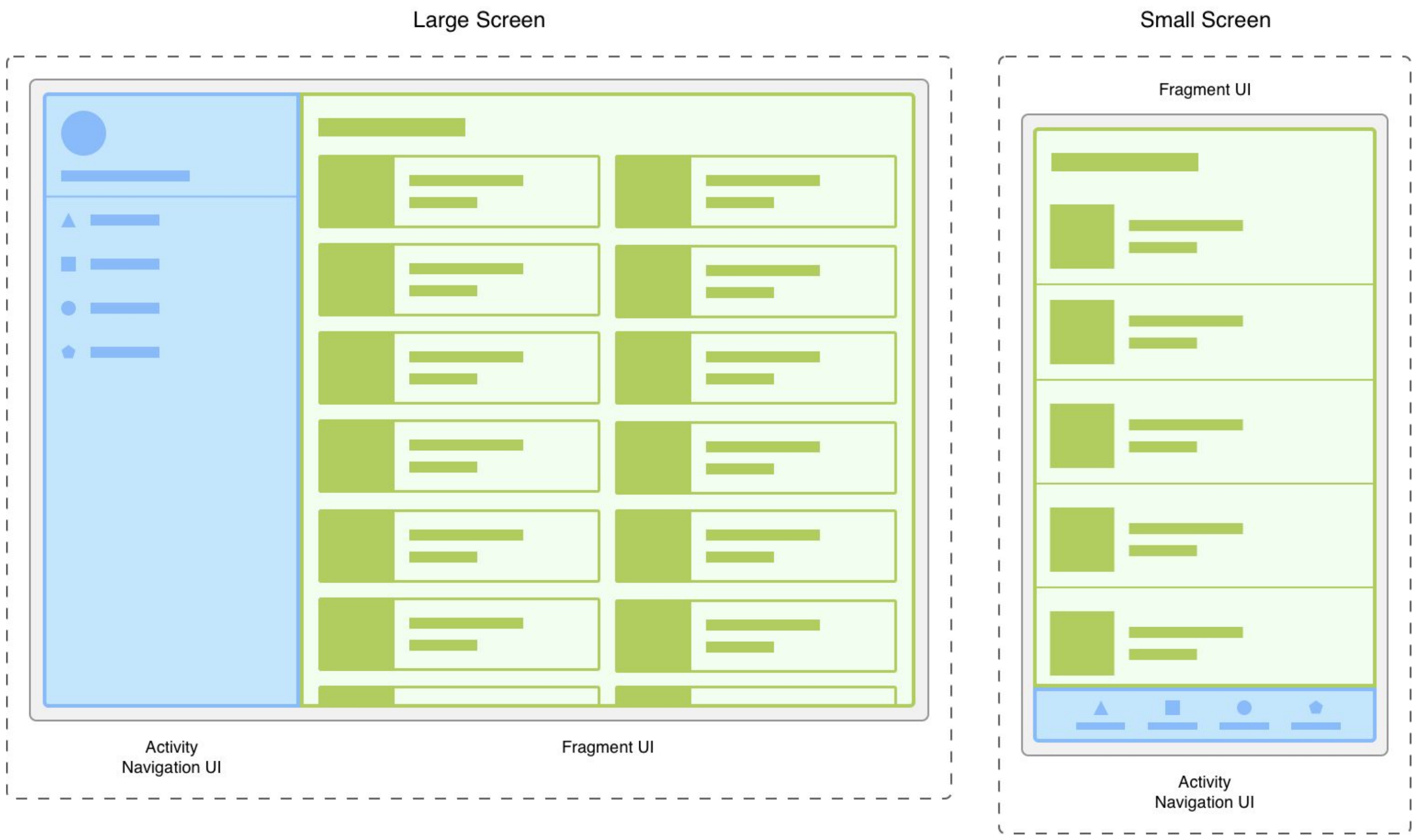
**Requirement:**

On larger screens, the app should display a static navigation drawer and a list in a grid layout.

On smaller screens, the app should display a bottom navigation bar and a list in a linear layout.

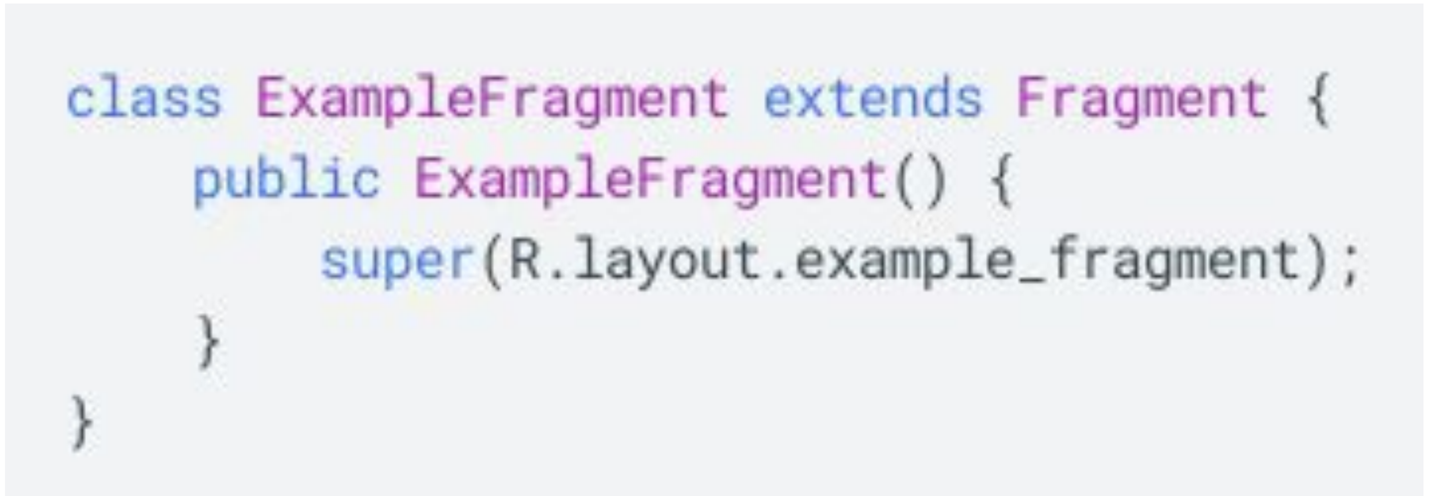
**Solution:**

The activity is then responsible for displaying the correct navigation UI while the fragment displays the list with the proper layout.



# CREATE FRAGMENT

- To create a fragment,
  - extend the AndroidX `Fragment` class, and
  - override its methods to insert your app logic
- To create a minimal fragment that defines its own layout, provide your fragment's layout resource to the base constructor



```
class ExampleFragment extends Fragment {
    public ExampleFragment() {
        super(R.layout.example_fragment);
    }
}
```

# Different fragment classes available

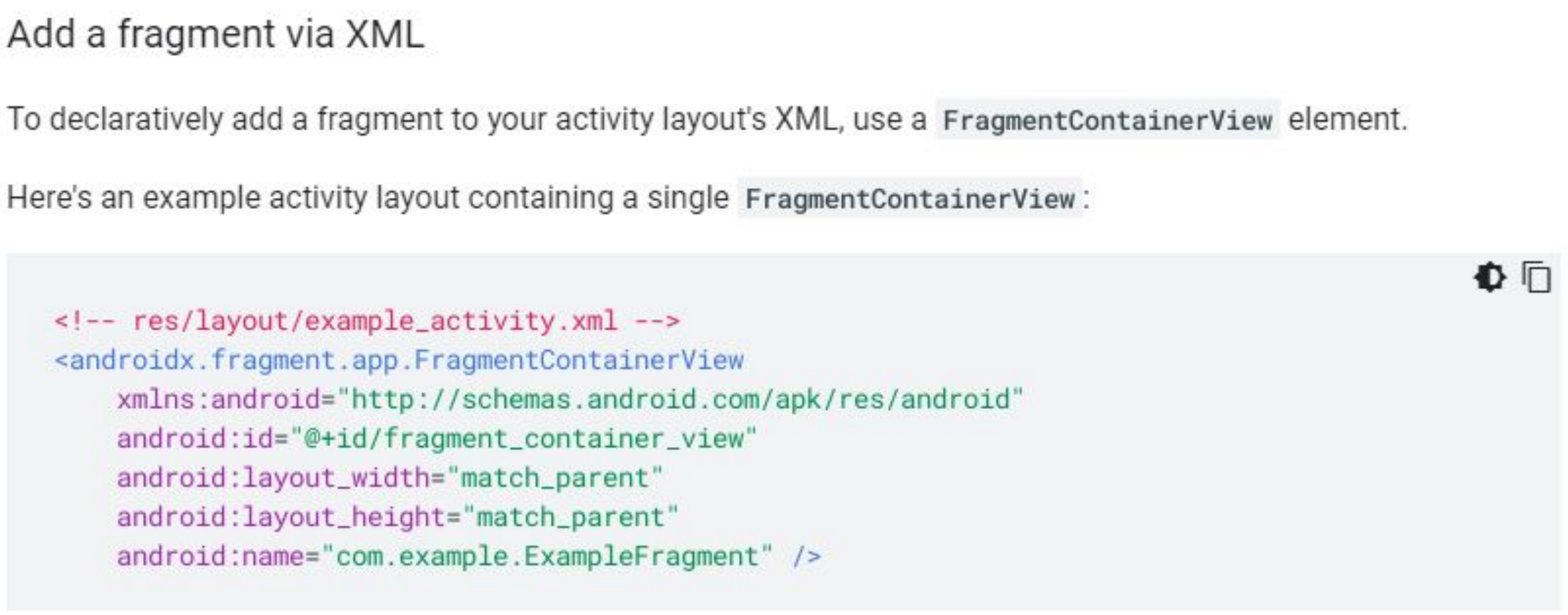
- `DialogFragment:` Displays a floating dialog
- `ListFragment`: Displays a list of items that are managed by an adapter.
- `PreferenceFragment`: Displays a hierarchy of `Preference` objects as a list, useful when creating a "settings" `Activity` for the app.

# Using a fragment

- The general steps to use a `Fragment`:
  - Create a subclass of [`Fragment`](#).
  - Create a layout for the `Fragment`.
  - Add the `Fragment` to a host `Activity`
    - either statically (XML)

      or
    - Dynamically (Java)

# ADD a fragment via xml



## Add a fragment via XML

To declaratively add a fragment to your activity layout's XML, use a `FragmentContainerView` element.

Here's an example activity layout containing a single `FragmentContainerView`:

```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.example.ExampleFragment" />
```

Source:-https://developer.android.com/guide/fragments/create#java

# ADD a fragment via xml

- The `android:name` attribute - class name of the `Fragment`
- When the activity's layout is inflated, the specified fragment is instantiated
- `onInflate()` is called on the newly instantiated fragment,
- `FragmentTransaction` is created to add the fragment to the `FragmentManager`.

Source:-https://developer.android.com/guide/fragments/create#java

## Add a fragment programmatically

- no specific fragment is automatically instantiated.
- Instead, a `FragmentTransaction` is used to instantiate a fragment and add it to the activity's layout.
- While the activity is running, fragment transactions such as adding, removing, or replacing a fragment can be performed.

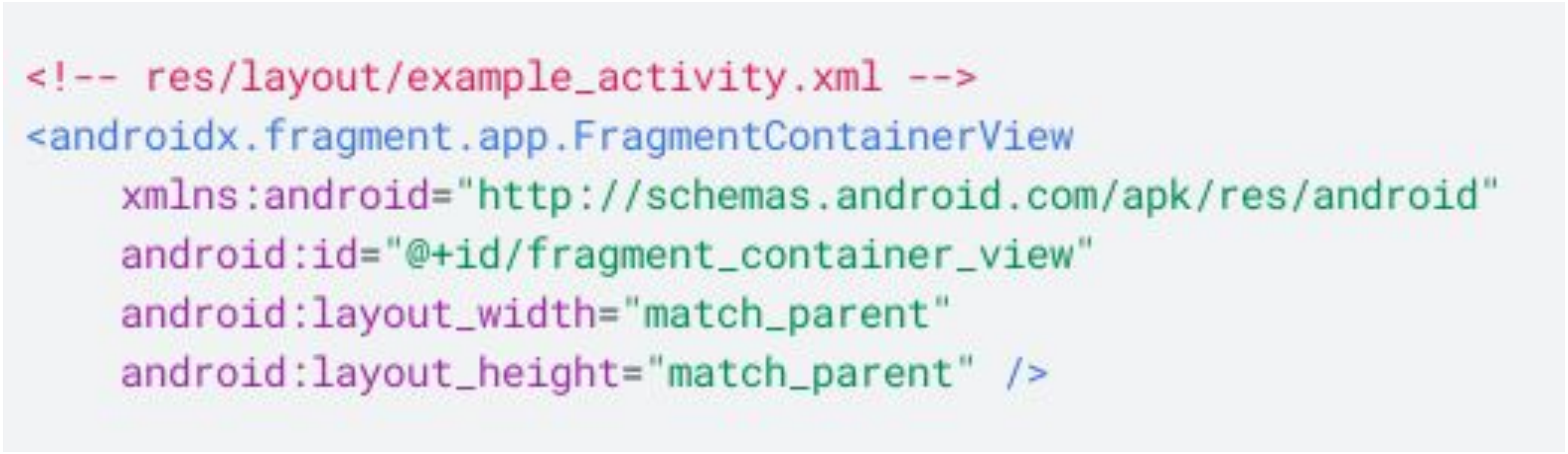
Source:-https://developer.android.com/guide/fragments/create#java

# Add a fragment programmatically

- In `FragmentActivity`, *get an instance* of the `FragmentManager`, which can be used to create a `FragmentTransaction`.
- Then *instantiate fragment* within activity's **`onCreate()`** method using `FragmentTransaction.add()`
  - Arguments are
    - `ViewGroup` ID of the container in layout and
    - fragment class
  - then commit the transaction,

Source:-https://developer.android.com/guide/fragments/create#java

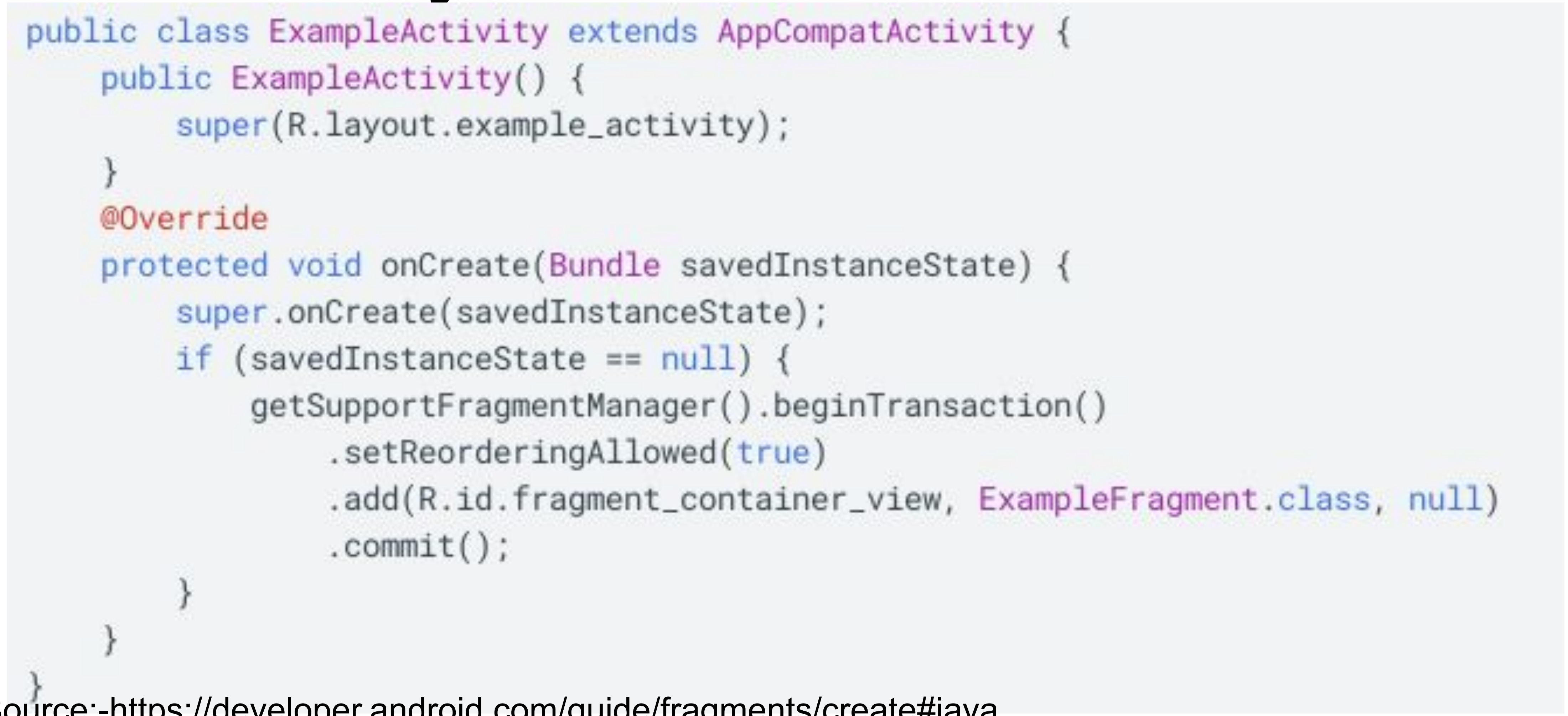
## XML Code ( with no "name" attribute)



```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Source:-https://developer.android.com/guide/fragments/create#java

## Java code to add fragment into an activity



```java
public class ExampleActivity extends AppCompatActivity {
    public ExampleActivity() {
        super(R.layout.example_activity);
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                .setReorderingAllowed(true)
                .add(R.id.fragment_container_view, ExampleFragment.class, null)
                .commit();
        }
    }
}
```

Source:-https://developer.android.com/guide/fragments/create#java

# Fragment transactions

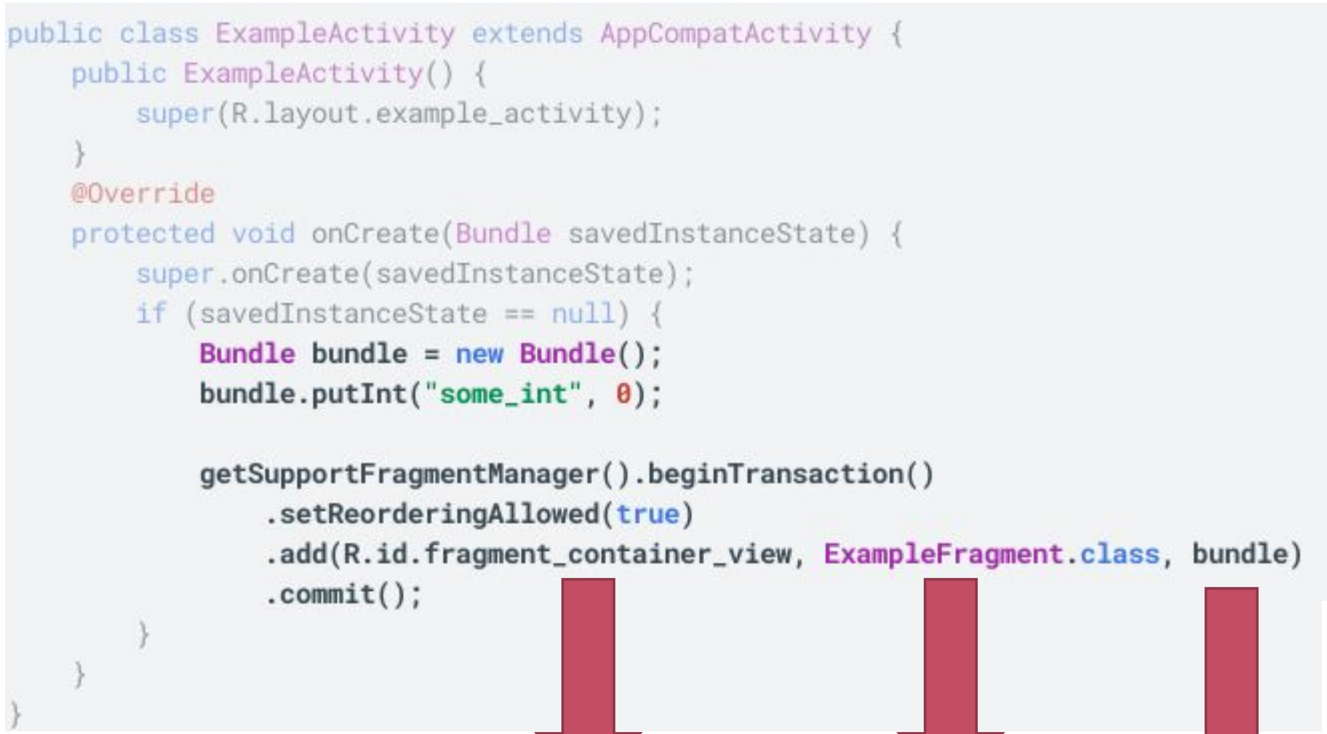
- Possible transactions with a fragment are:
  - Add a `Fragment` using `add()`.
  - Remove a `Fragment` using `remove()`.
  - Replace a `Fragment` with another `Fragment` using `replace()`.
  - Hide and show a `Fragment` using `hide()` and `show()`.

Source:-
https://google-developer-training.github.io/android-developer-advanced-course-concepts/unit-1-expand-the-user-experience/lesson-1-fragments/1-1-c-fragments/1-1-c-fragments.html#creating

## `if (savedInstanceState == null)`

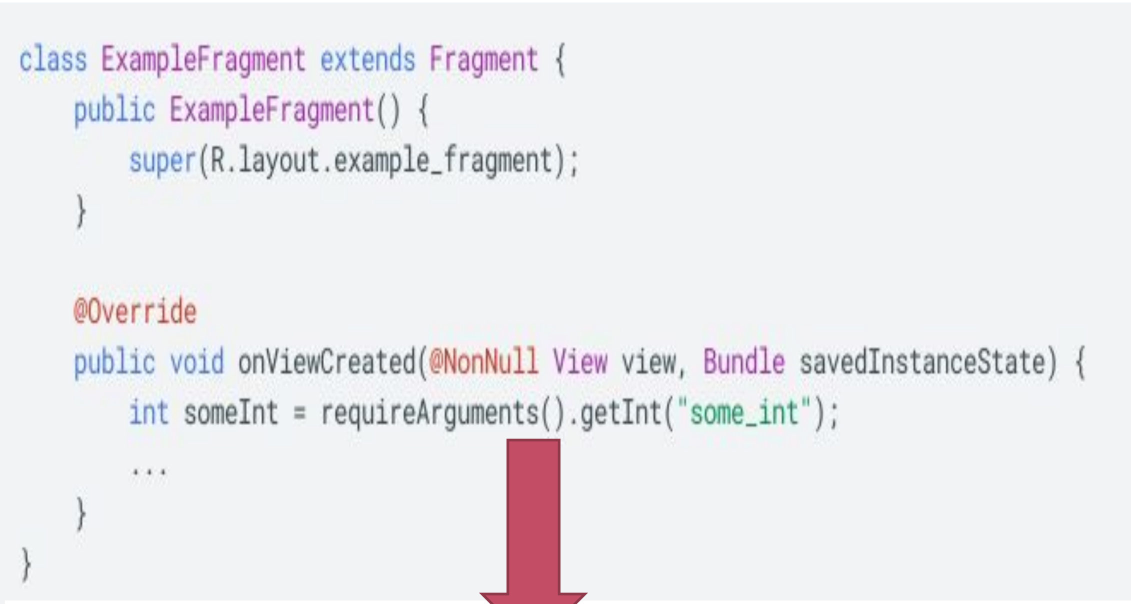
- To ensure that the fragment is added only once, when the activity is first created.
- When a configuration change occurs and the activity is recreated, `savedInstanceState` is no longer `null`, and the fragment does not need to be added a second time.
- fragment is automatically restored from the `savedInstanceState`.

## Passing arguments to fragment through `Bundle`



```
public class ExampleActivity extends AppCompatActivity {
    public ExampleActivity() {
        super(R.layout.example_activity);
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (savedInstanceState == null) {
            Bundle bundle = new Bundle();
            bundle.putInt("some_int", 0);

            getSupportFragmentManager().beginTransaction()
                .setReorderingAllowed(true)
                .add(R.id.fragment_container_view, ExampleFragment.class, bundle)
                .commit();
        }
    }
}
```



```
class ExampleFragment extends Fragment {
    public ExampleFragment() {
        super(R.layout.example_fragment);
    }

    @Override
    public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
        int someInt = requireArguments().getInt("some_int");
        ...
    }
}
```

To read arguments

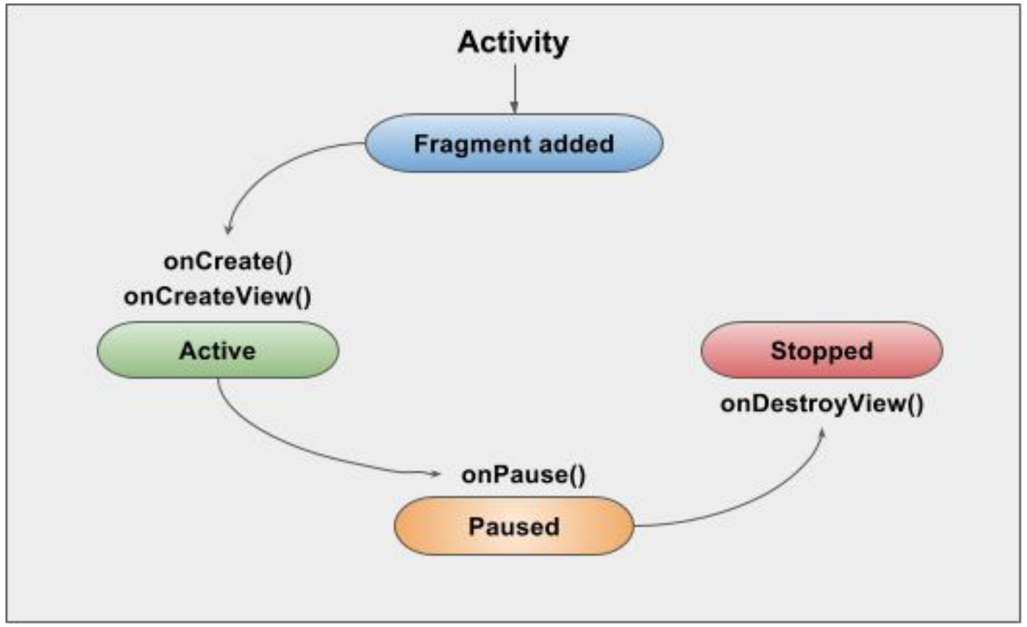
ViewGroup ID of the fragment

Name of the Fragment class

To pass arguments or input to fragment

# Fragment life cycle

- The `Fragment` is added by an `Activity` (which acts as the host of the `Fragment`).
- Once added, the `Fragment` goes through three states:
  - Active (or resumed)
  - Paused
  - Stopped



# Fragment life cycle states

- `Fragment` is always hosted by an `Activity`
- `Fragment` lifecycle is directly affected by the host `Activity` lifecycle.
- For example,
  - when the `Activity` is paused, so are all `Fragments` in it, and
  - when the `Activity` is destroyed, so are all `Fragments`.

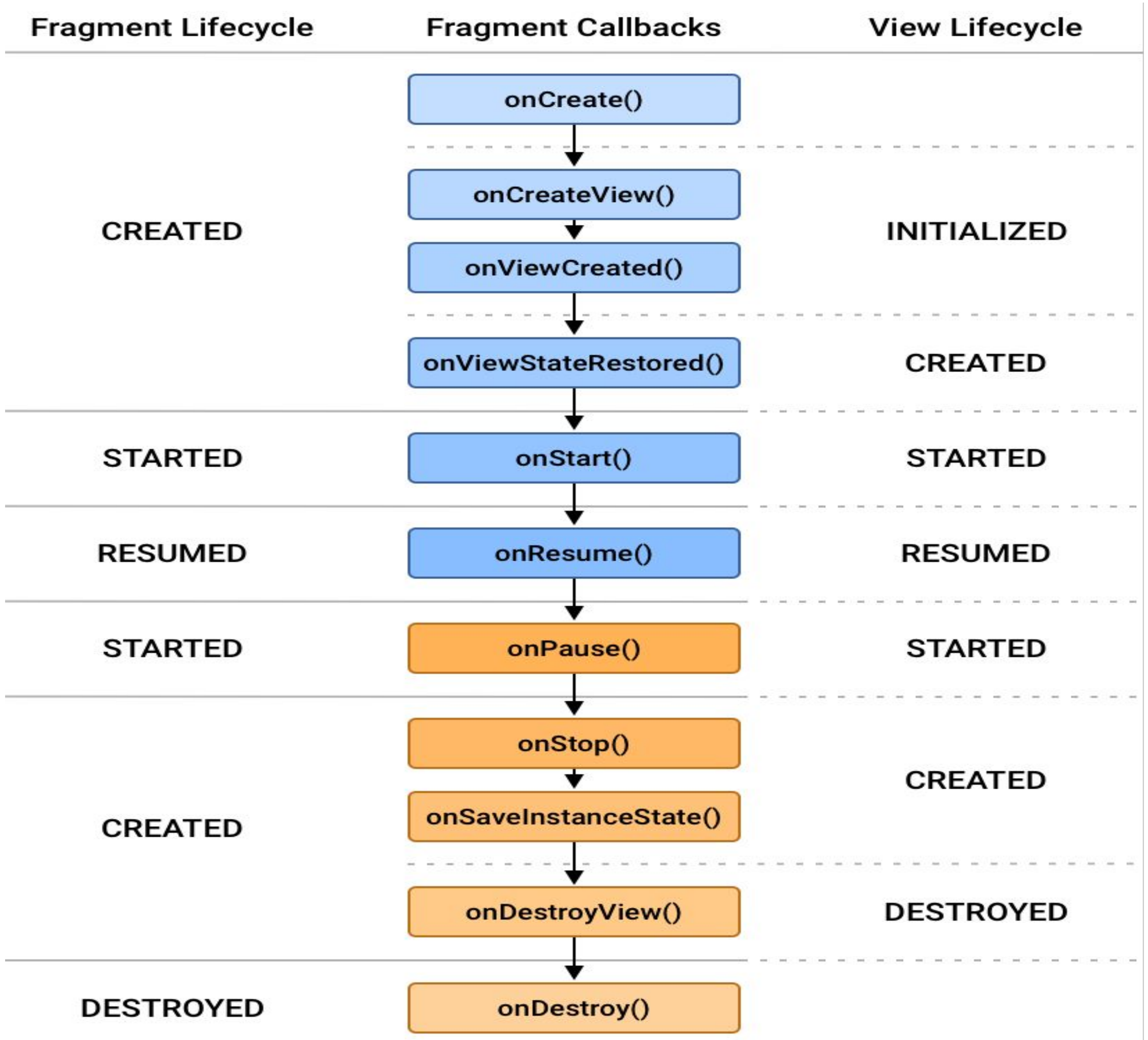
# Fragment life cycle states

- Each lifecycle callback for the `Activity` results in a similar callback for each `Fragment`, as shown in the following table.

| Activity State | Fragment Callbacks Triggered | Fragment Lifecycle |
|---|---|---|
| Created | onAttach(), onCreate(), onCreateView(), onActivityCreated() | Fragment is added and its layout is inflated. |
| Started | onStart() | Fragment is active and visible. |
| Resumed | onResume() | Fragment is active and ready for user interaction. |
| Paused | onPause() | Fragment is paused because the Activity is paused. |
| Stopped | onStop() | Fragment is stopped and no longer visible. |
| Destroyed | onDestroyView(), onDestroy(), onDetach() | Fragment is destroyed. |

# Fragment lifecycle callbacks

Fragment Lifecycle states and their relation both the fragment's lifecycle callbacks and the fragment's view Lifecycle.



# Fragment call backs

- `onCreate()`:
  - Initialize essential components and variables of the `Fragment`
  - Anything initialized in `onCreate()` is preserved if the `Fragment` is paused and resumed.
- `onCreateView()`:
  - Inflate the XML layout for the `Fragment` in this callback.
  - The system calls this method to draw the `Fragment` UI for the first time.
  - As a result, the `Fragment` is visible in the `Activity`.
  - To draw a UI for your `Fragment`, you must return the root `View` of your `Fragment` layout. Return `null` if the `Fragment` does not have a UI.

# Fragment call backs

- `onPause()`:
  - Save any data and states that need to survive beyond the destruction of the `Fragment`.
  - The system calls this method if any of the following occurs:
    - The user navigates backward.
    - The `Fragment` is replaced or removed, or another operation is modifying the `Fragment`.
    - The host `Activity` is paused.

# Fragment call backs

- `onResume()`: Called by the `Activity` to resume a `Fragment` that is visible to the user and actively running.
- `onAttach()`: Called when a `Fragment` is first attached to a host `Activity`.
- `onDestroyView()`: Called when the `View` previously created by `onCreateView()` has been detached from the `Fragment`.