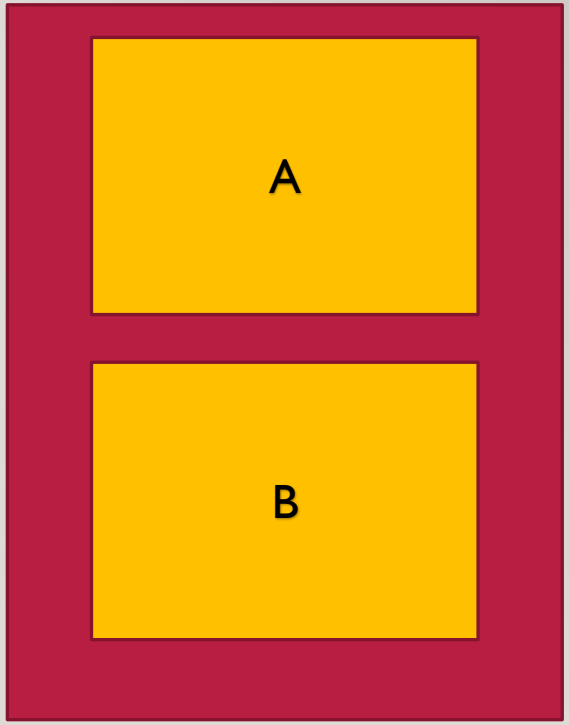
INTRODUCTION TO FRAGMENTS

OUTLINE

- Intro Fragments in androids
- Fragment Life cycle
- Adding Fragment
- Activity – Fragment communication
- Examples

INTRODUCTION

- Activity contains >> view and View Group
  - views .e.g. Text View , Image View ..
  - View Group .e.g. Linear Layout, Relative Layout ..



INTRO TO FRAGMENTS

- Represents a behavior or a portion of user interface in an Activity
- must always be embedded in an activity
- We can add or remove while the activity is running
- An Activity may contain more than one fragment

FRAGMENTS

- Modular section of an activity
- flexible UI designs
- Has its own lifecycle
- Directly affected by host activity's lifecycle
- Receives its own input events
- Can be added or removed while the activity is running
- Can be declared in layout XML using <fragment> element

FRAGMENTS :- FLEXIBLE UI DESIGNS

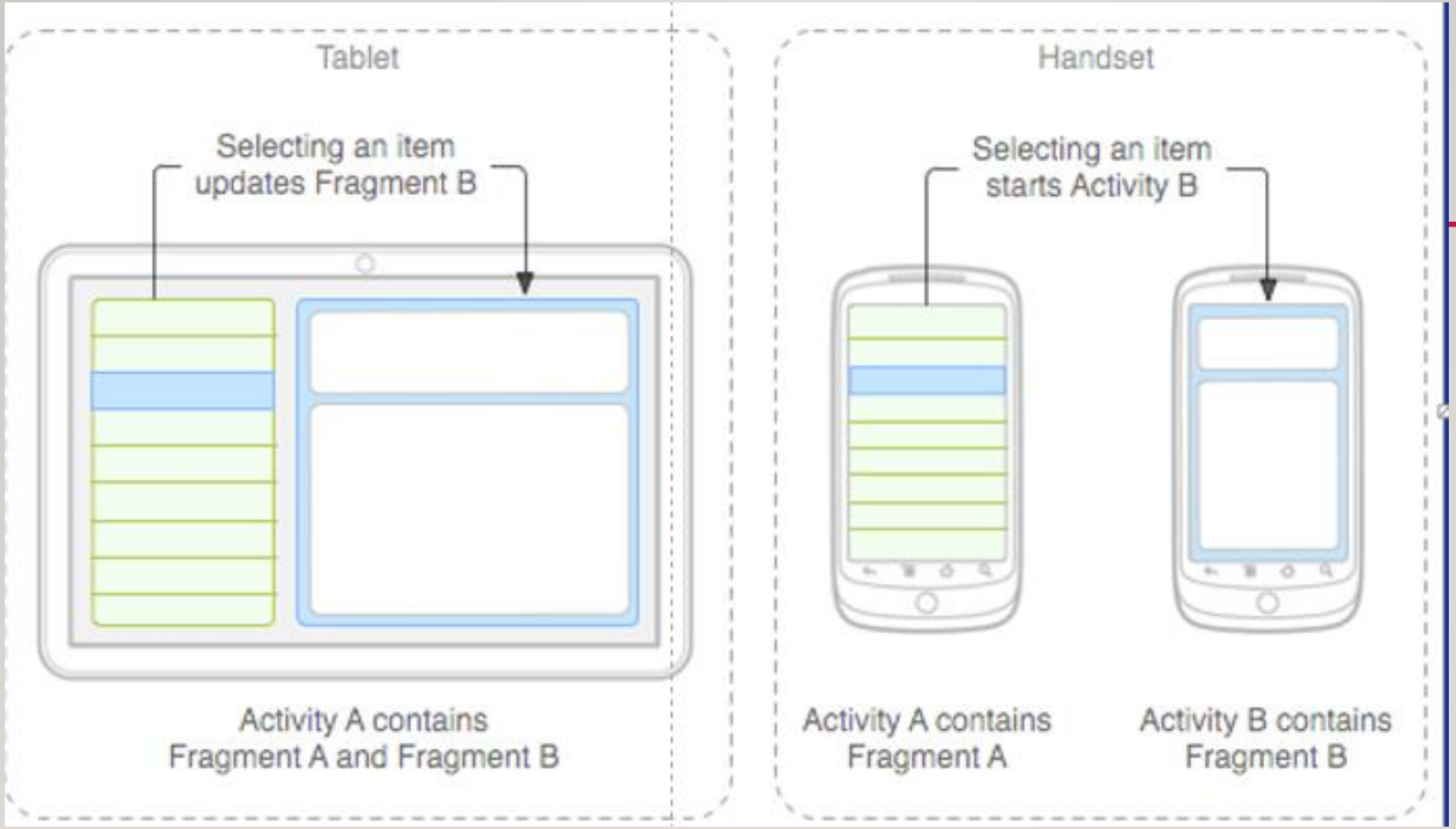


Image src: https://developer.android.com/guide/components/fragments.html

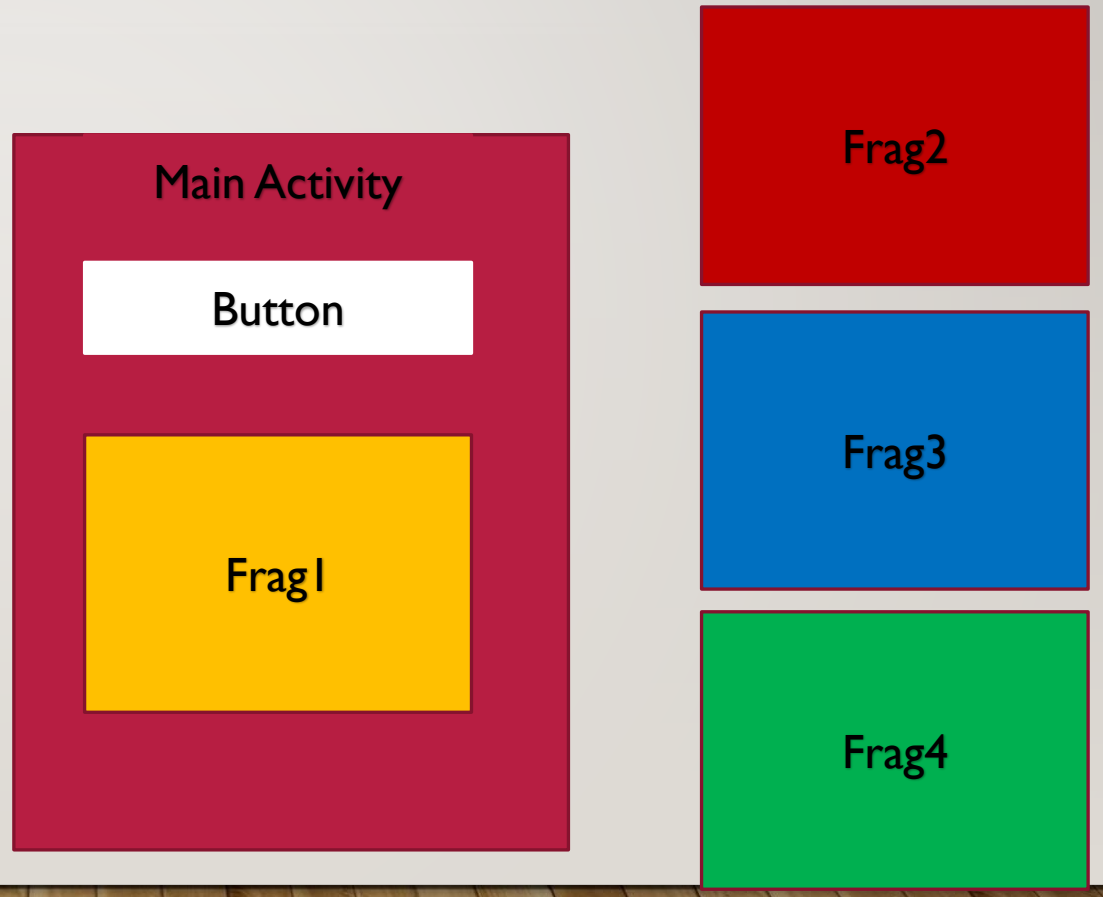
INTRODUCTION TO FRAGMENTS

Fragments

Allow us to add /replace

and remove “activity’s sub views”

dynamically



FRAGMENT LIFE CYCLE

- Fragment has its own life cycle
- Fragment life cycle is affected by the hosting Activity life cycle

THE LIFECYCLE OF A FRAGMENT

Fragment lifecycle callback methods

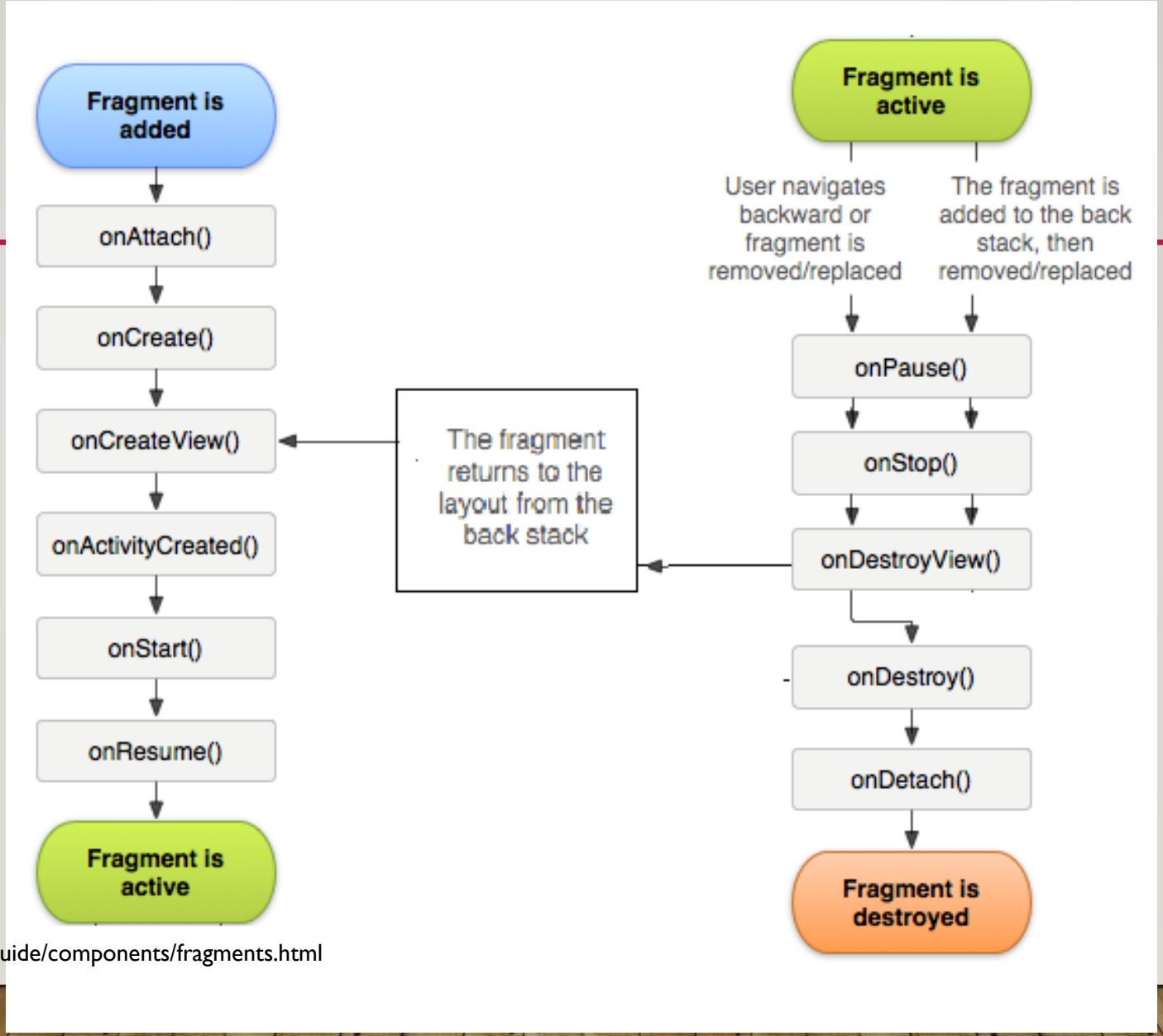


Figure source: http://developer.android.com/guide/components/fragments.html

LIFECYCLE OF FRAGMENT

- It is recommended that the following 3 callback should be implemented:
  - onCreate()
  - onCreateView()
  - onPause()

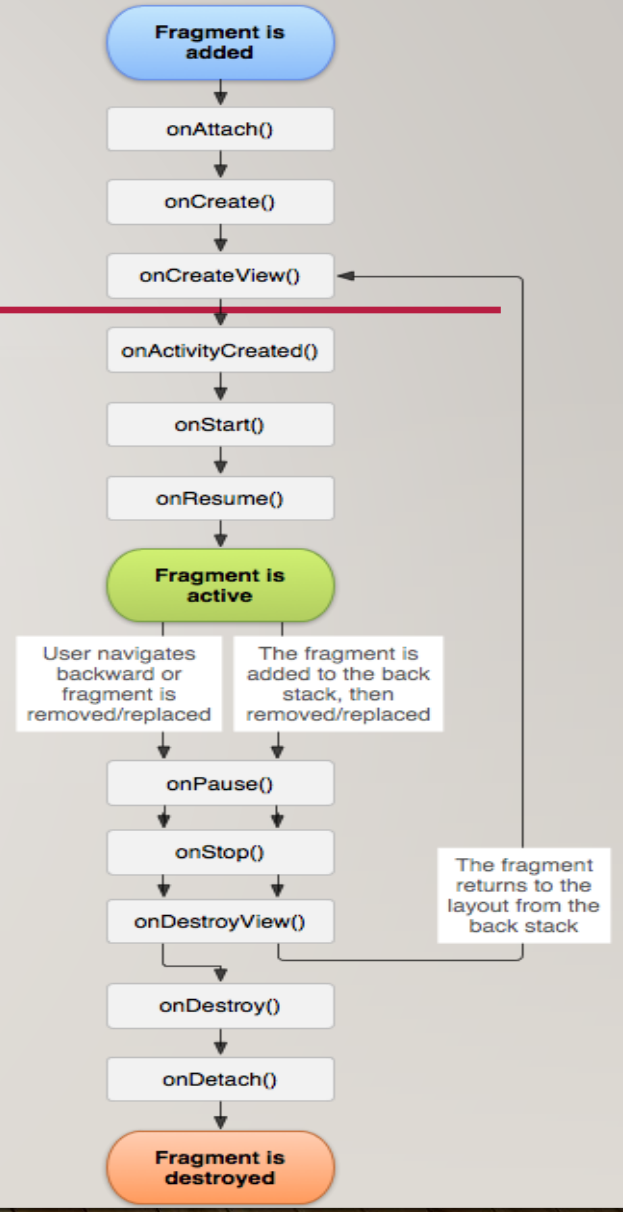


Image source: https://developer.android.com/

FRAGMENT LIFECYCLE CALLBACKS

onCreate()

- Called when creating the fragment.
- Initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed

onCreateView()

- Called when it's time for the fragment to draw its user interface for the first time.
- Must return a View from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.

onPause()

- Called as the first indication that the user is leaving the fragment
- At this point, you should commit any changes that should be persisted beyond the current user session

CREATING FRAGMENT

- Steps required to create fragment
- 1- create subclass that extends Fragment class

  class MyFragment extends Fragment{ … }

  2- create fragment layout XML File

```
public class Fragment2 extends Fragment {

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

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {

        return inflater.inflate(R.layout.fragment1_layout, container ,
false);
    }
}
```

ADDING FRAGMENT TAG TO THE ACTIVITY XML

```
<fragment
    android:name=“packgeName.Fragment1”
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/fragment"
/>
```

ADDING FRAGMENT PROGRAMMATICALLY

```
private void addFragment(){

 FragmentManager  manager = getSupportFragmentManager();
 FragmentTransaction transaction = manager.beginTransaction();
    Fragment fragment2 = Fragment2();
    transaction.add(R.id.fragment, fragment2);
    transaction.commit();
}
```

ADDING FRAGMENT PROGRAMMATICALLY

- To make fragment transactions in your activity (such as add, remove, or replace a fragment), you must use APIs from FragmentTransaction.
- Then, you can use the add() method to add a fragment.
- The first argument passed to add() is the viewGroup id in which the fragment should be placed, and the second parameter is the fragment to add.
- You must call commit() with FragmentTransaction for the changes to take effect.

FRAGMENT MANAGER

- Use `supportFragmentManager`

❖ to manage fragments in activity.

❖ obtain the fragments with findFragmentById() or findFragmentByTag()

❖ Pop fragments off the back stack, with popBackStack() (simulating a Back command by the user).

FRAGMENT TRANSACTIONS

- FragmentTransaction API can be used to

❖ Add fragment

❖ Remove fragment

❖ Replace fragment

❖ You call addToBackStack(), in order to add the transaction to a back stack of fragment transactions.

❖ You must call commit() in the end in order to save all changes.

EXAMPLE

- Create activity that contain fragment container
- Create two/ three fragment classes along with its xml file
- Add and replace fragment dynamically

ACTIVITY TO FRAGMENT COMMUNICATION

- Activity can get reference to fragment using FragmentManager
- Fragment myFrag =
  supportFragmentManager.findFragmentById(R.id.fragment);
- Then invoke any method in the fragment
- myFrag.doSomething();

```
FragmentManager manager = getSupportFragmentManager();
Fragment fragment =
manager.findFragmentById(R.id.fragment);
if(fragment instanceof Fragment2){
    ((Fragment2) fragment).displayMessage(" message from
Activity");
}
```

REFERENCES

1. https://developer.android.com/guide
2. https://developer.android.com/training/basics/fragments/communicating.html#DefineInterface