CSE 437

Mobile Computing

Activities and Services

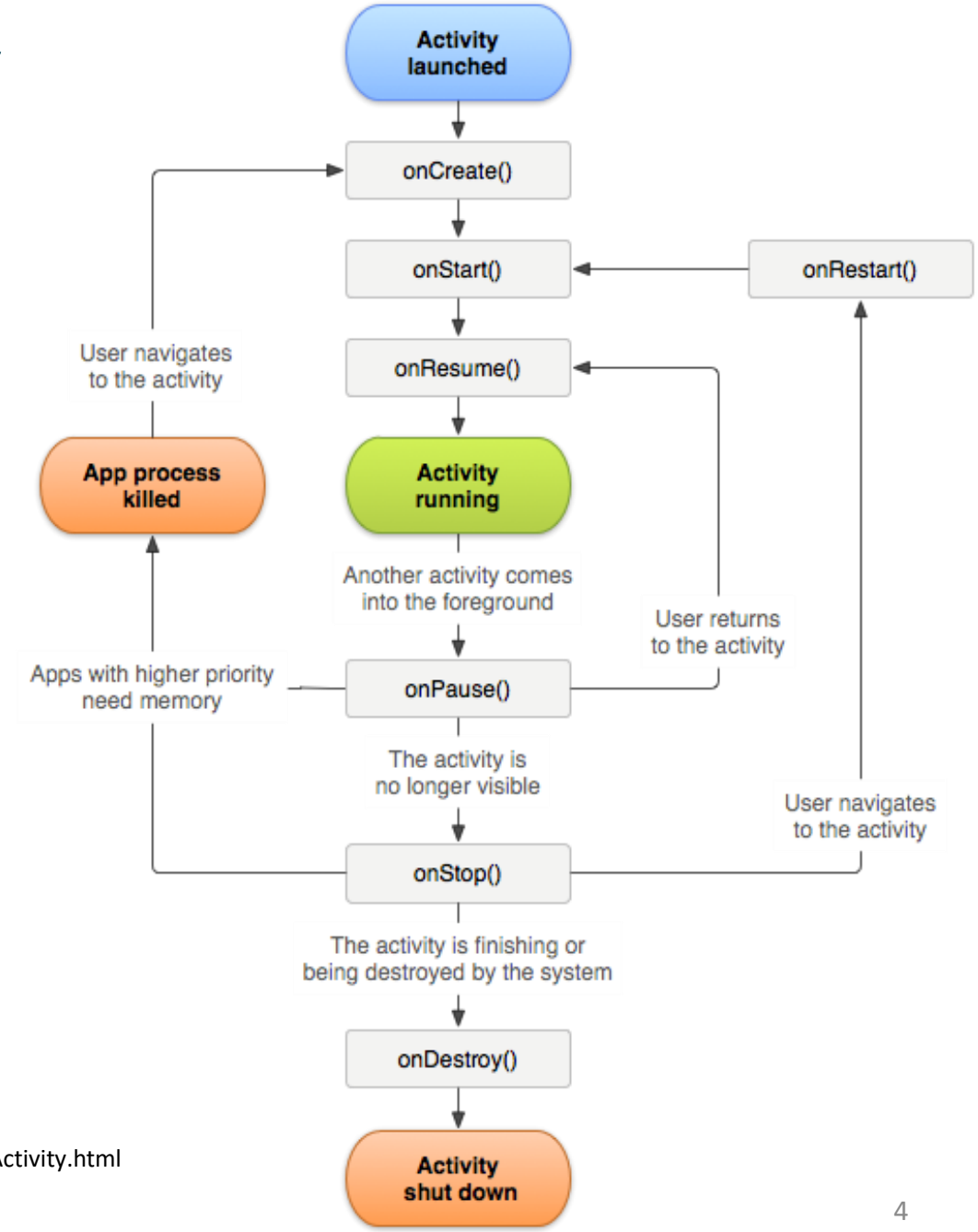
1. The life cycles of an activity
2. Intent and Intent Filters
3. Common Intents
4. Sharing data using Intents
5. Managing Multiple Activities
6. The life cycles of a Fragment
7. Managing Fragments
8. Services

The life cycles of an activity

The life cycles of an activity

States

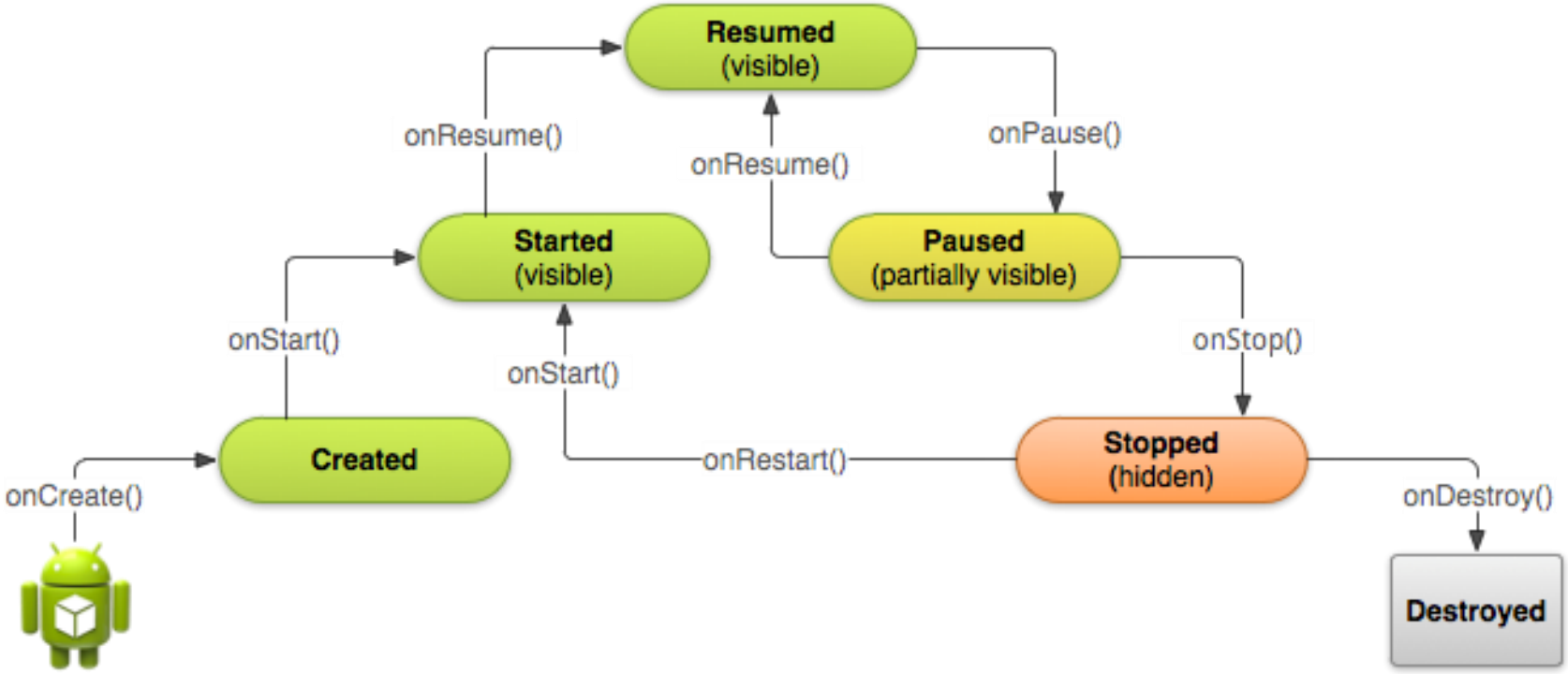
- Activity running
- Activity paused
- Activity stopped
- Activity destroyed



Ref: https://developer.android.com/reference/android/app/Activity.html

The life cycles of an activity

Methods and Key Loops



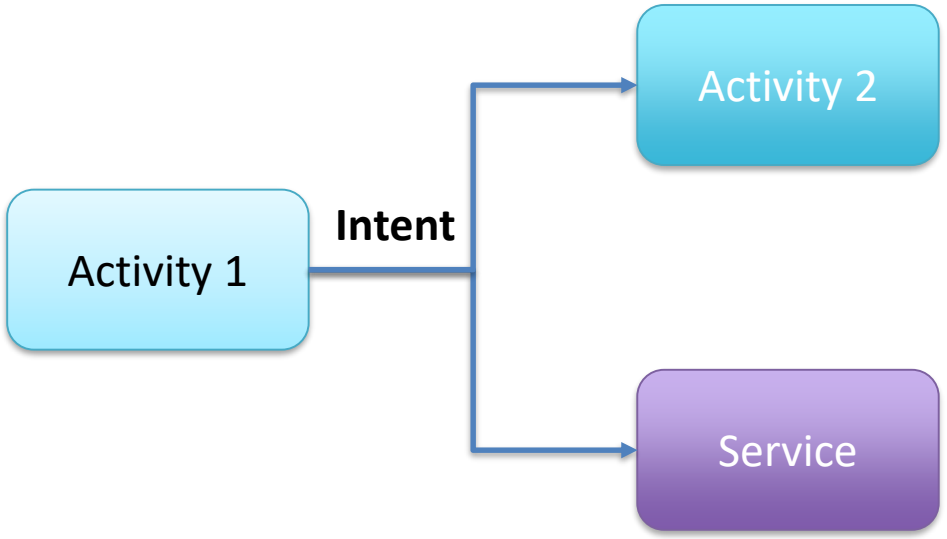
Source: http://developer.android.com/training/basics/activity-lifecycle/starting.html

Intent and Intent Filters

Intents

Use Cases

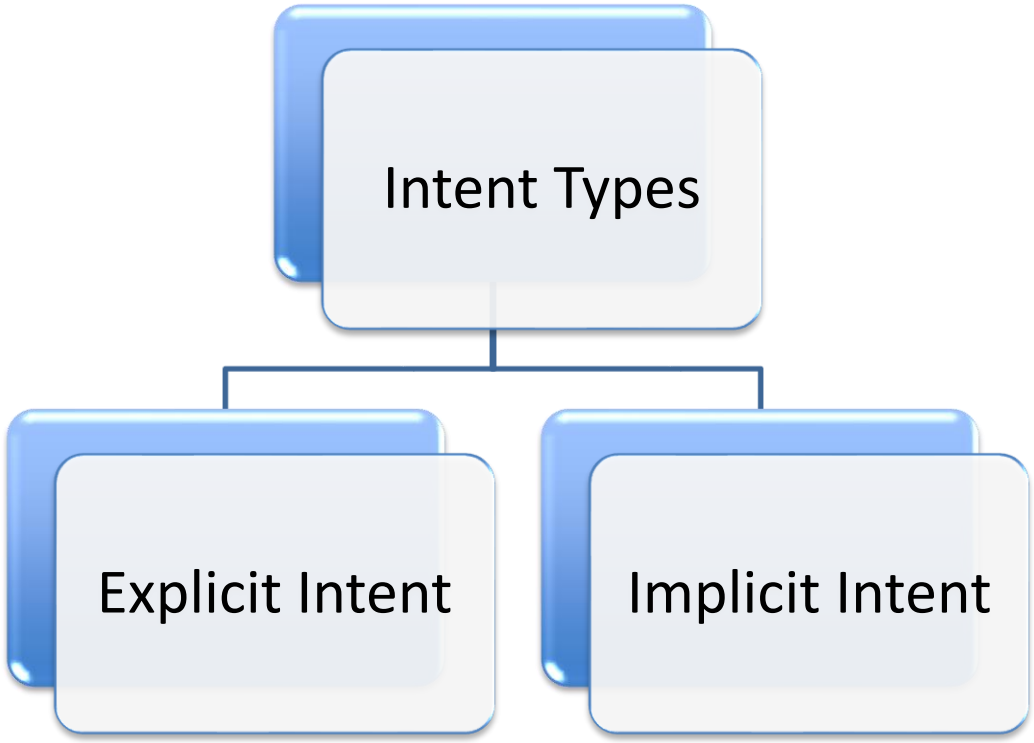
- An Intent is a messaging object you can use to request an action from another app component.
- **Use Cases**
  - **To start an activity**
  - **To start a service**
  - **To deliver a broadcast**



Ref: https://developer.android.com/guide/components/intents-filters.html

Intents

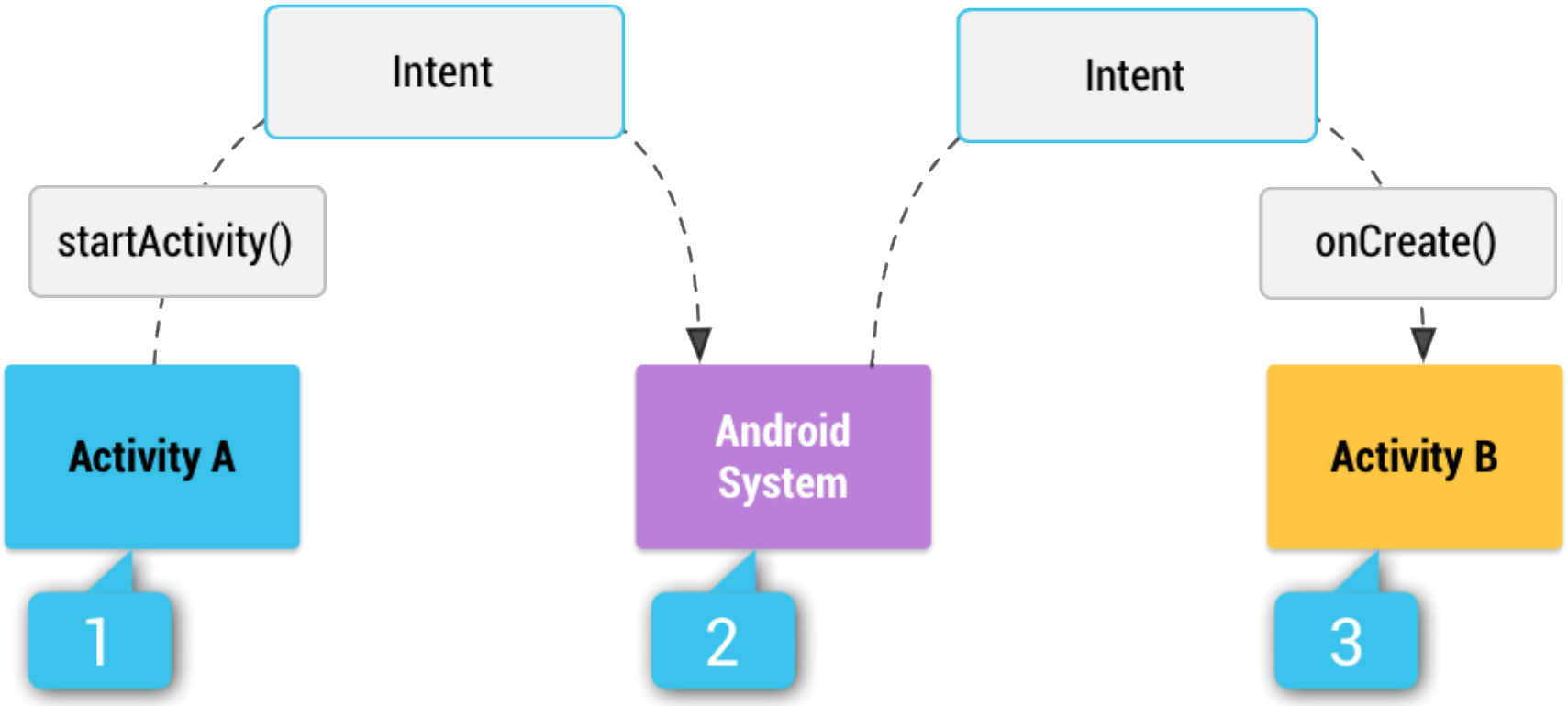
Types



Ref: https://developer.android.com/guide/components/intents-filters.html

Intents

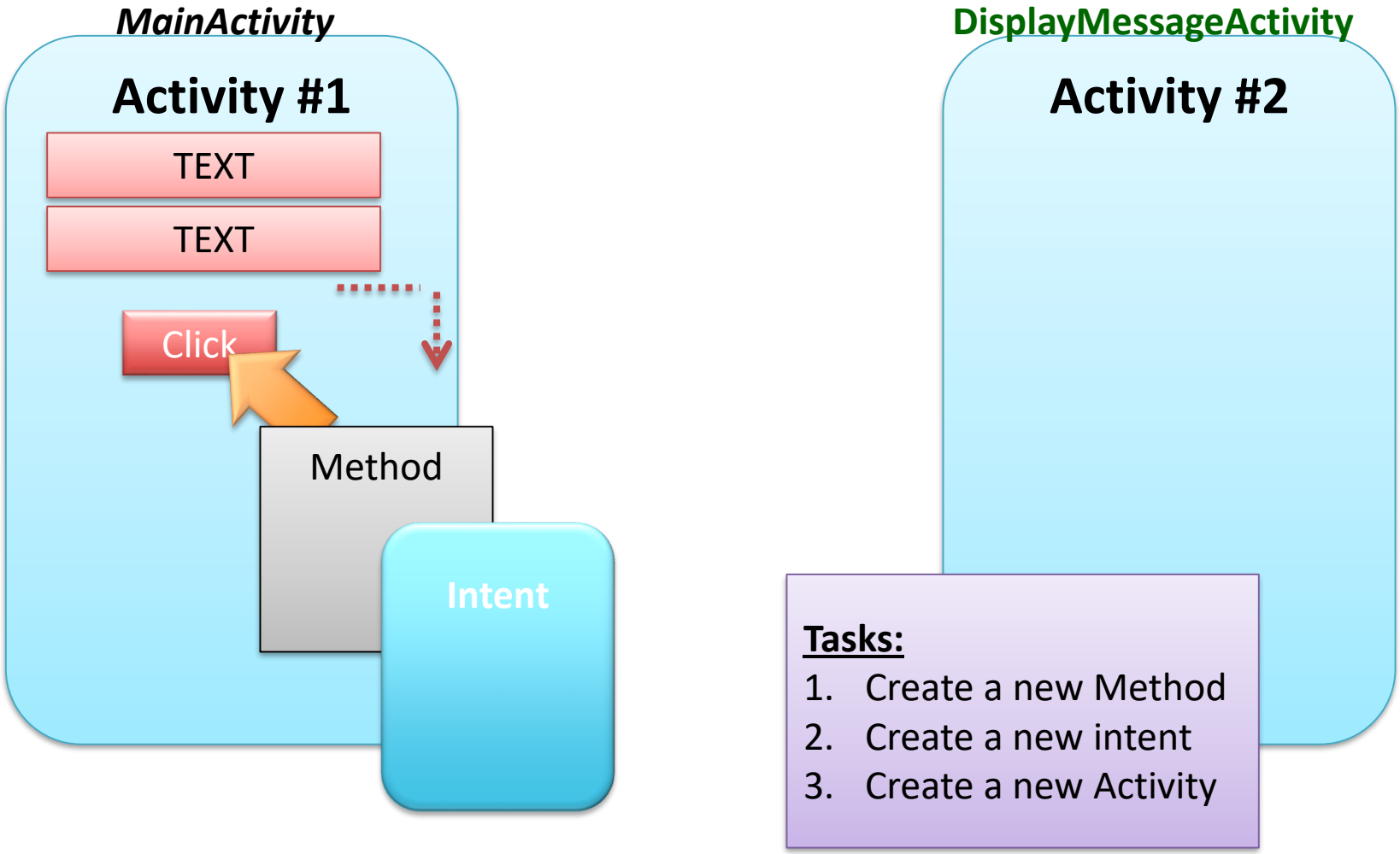
Implicit Intent



Ref: https://developer.android.com/guide/components/intents-filters.html

Intents

Building an Intent

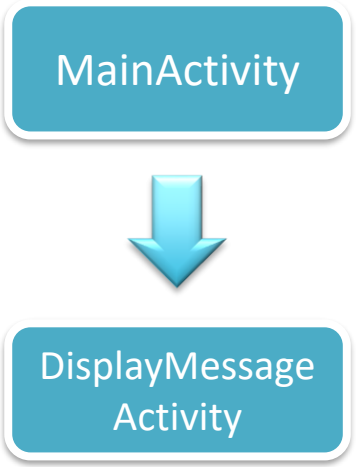


Ref: https://developer.android.com/guide/components/intents-filters.html

Intents

Building an Intent to start an activity called DisplayMessageActivity

```
public class MainActivity extends AppCompatActivity {
    public final static String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /** Called when the user clicks the Send button */
    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.edit_message);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
}
```



Ref: https://developer.android.com/training/basics/firstapp/starting-activity.html

Common Intents

Common Intents

Actions

Task	Action
Create an alarm	ACTION_SET_ALARM
Create a timer	ACTION_SET_TIMER
Show all alarms	ACTION_SHOW_ALARMS
Add a calendar event	ACTION_INSERT
Capture a picture or video and return it	ACTION_IMAGE_CAPTURE or ACTION_VIDEO_CAPTURE
Start a camera app in still image mode	INTENT_ACTION_STILL_IMAGE_CAMERA
Start a camera app in video mode	INTENT_ACTION_VIDEO_CAMERA

Example

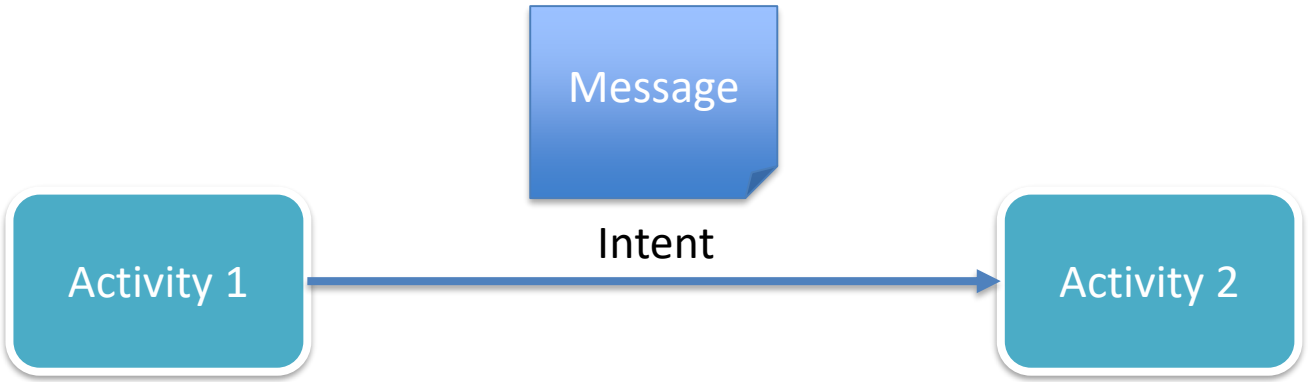
```
public void capturePhoto() {
    Intent intent = new Intent(MediaStore.INTENT_ACTION_STILL_IMAGE_CAMERA);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent);
    }
}
```

Sharing data using Intents

Sharing Data

put/get Extra

- An intent not only allows you to start another activity, but it can carry a bundle of data to the activity as well.



Ref: https://developer.android.com/training/basics/firstapp/starting-activity.html

Sharing Data

Building an Intent to start an activity called DisplayMessageActivity and sending a message to it.

```
public class MainActivity extends AppCompatActivity {
    public final static String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /** Called when the user clicks the Send button */
    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.edit_message);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
}
```

Ref: https://developer.android.com/training/basics/firstapp/starting-activity.html

Sharing Data

Receiving the Intent and Displaying the Message

onCreate() *method for* ***DisplayMessage*** *should look like this:*

```
TextView TextView10;
```

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

        TextView10 = (TextView)findViewById(R.id.TextView10);


        Intent intent02 = getIntent();
        String message = intent02.getStringExtra(
        MainActivity.EXTRA_MESSAGE);
        TextView10.setText(message);

}
```

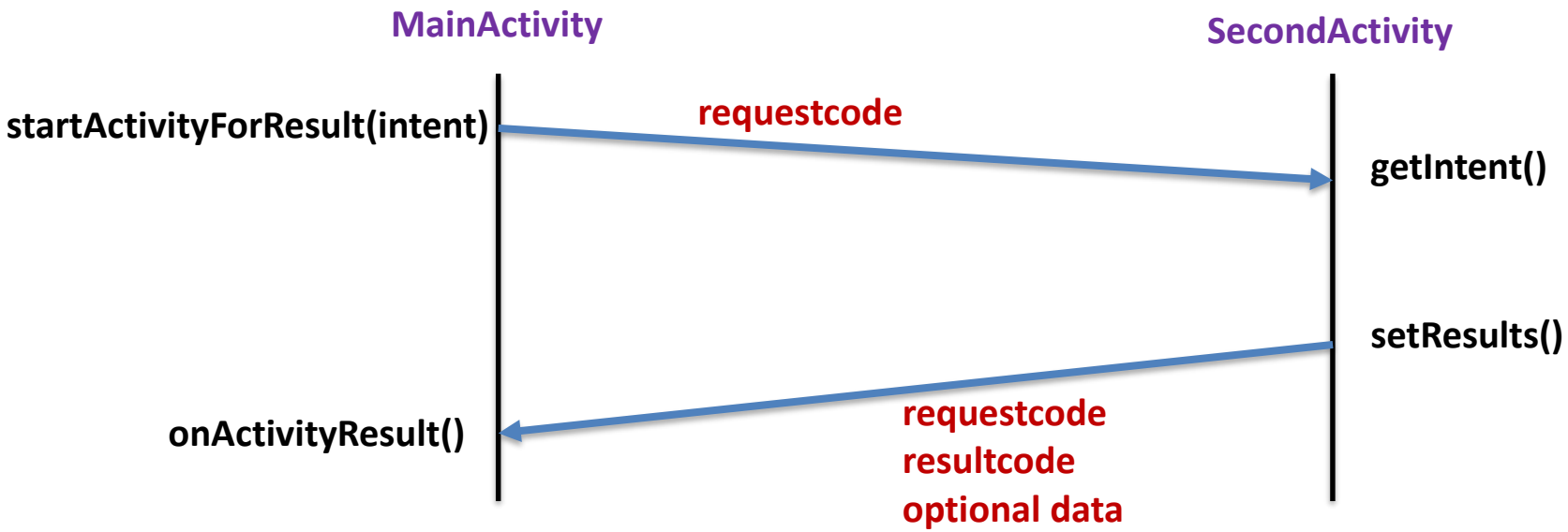
Ref: https://developer.android.com/training/basics/firstapp/starting-activity.html

Managing Multiple Activities

Managing Multiple Activities

Getting a Result from an Activity

- Starting another activity doesn't have to be one-way.
- You can also start another activity and receive a result back.
- To receive a result, call **startActivityForResult()** (instead of **startActivity()**).



https://developer.android.com/training/basics/intents/result.html

Managing Multiple Activities

Start the Activity

Starting an activity that allows the user to pick a contact

```
static final int PICK_CONTACT_REQUEST = 1;  // The request code
...
private void pickContact() {
    Intent pickContactIntent = new Intent(Intent.ACTION_PICK, Uri.parse("content://contacts"));
    pickContactIntent.setType(Phone.CONTENT_TYPE);
     // Show user only contacts w/ phone numbers
    startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
}
```

https://developer.android.com/training/basics/intents/result.html

Managing Multiple Activities

Receive the Result

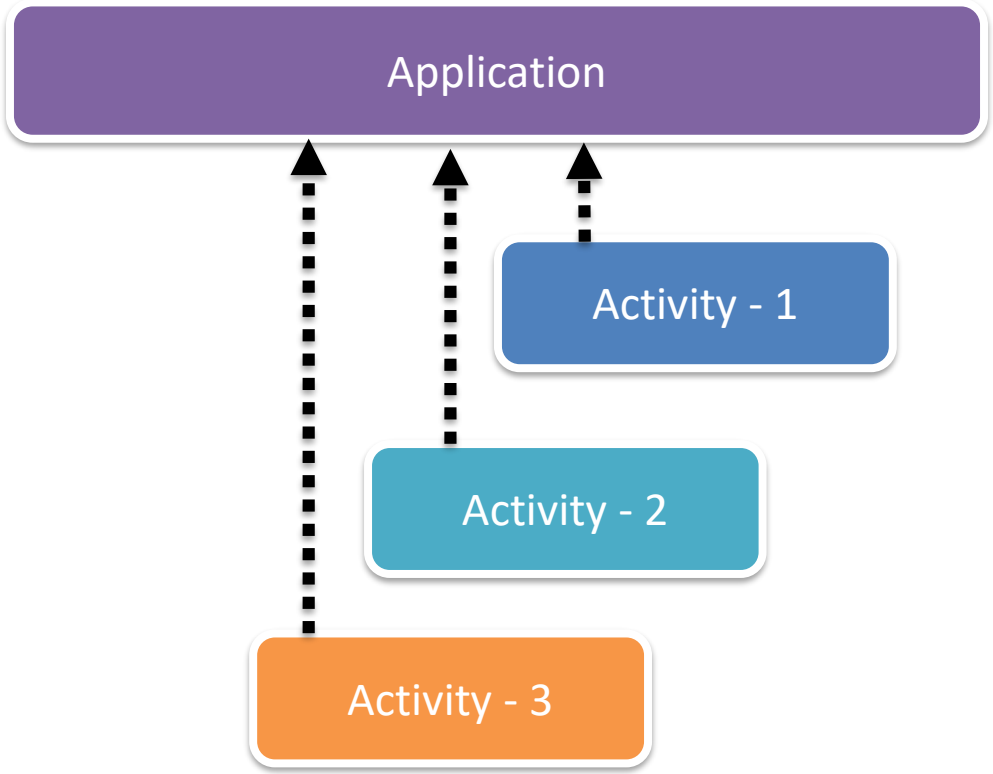
```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   // Check which request we're responding to
   if (requestCode == PICK_CONTACT_REQUEST) {
      // Make sure the request was successful
      if (resultCode == RESULT_OK) {
         // The user picked a contact.
         // The Intent's data Uri identifies which contact was selected.

         // Do something with the contact here (bigger example below)
      }
   }
}
```

https://developer.android.com/training/basics/intents/result.html

Managing Multiple Activities

Activity lifecycle

- Remember that Every Activity has a Life Cycle
- Remember that Only one activity can run in the foreground at one time. The rest are paused or stopped



Managing Multiple Activities

Main Activity

- when we have an application with multiple activities, we need to define the “main” or entry-point activity.
- In the Android framework, the manifest file provides the information about all of the activities that make up an application as well as define the activity that will serve as the entry point.
- The main activity for your app must be declared in the manifest with an **<intent-filter>** that includes the **MAIN** action and **LAUNCHER** category

```
<activity
        android:name="tru.comp2160.lab1.HelloWorldActivity"
        android:label="@string/app_name" >
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />
           <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

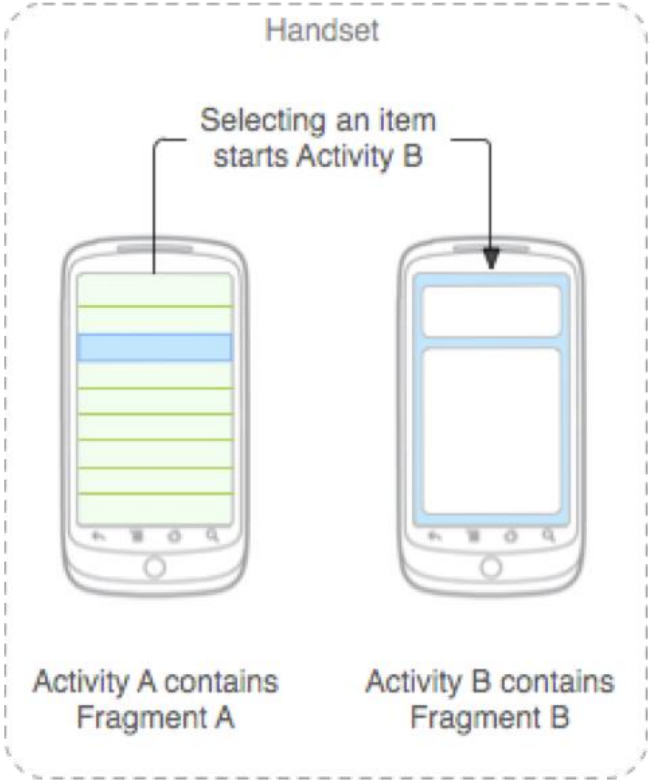
Fragments

The life cycles of a Fragment

Introduction

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

You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.

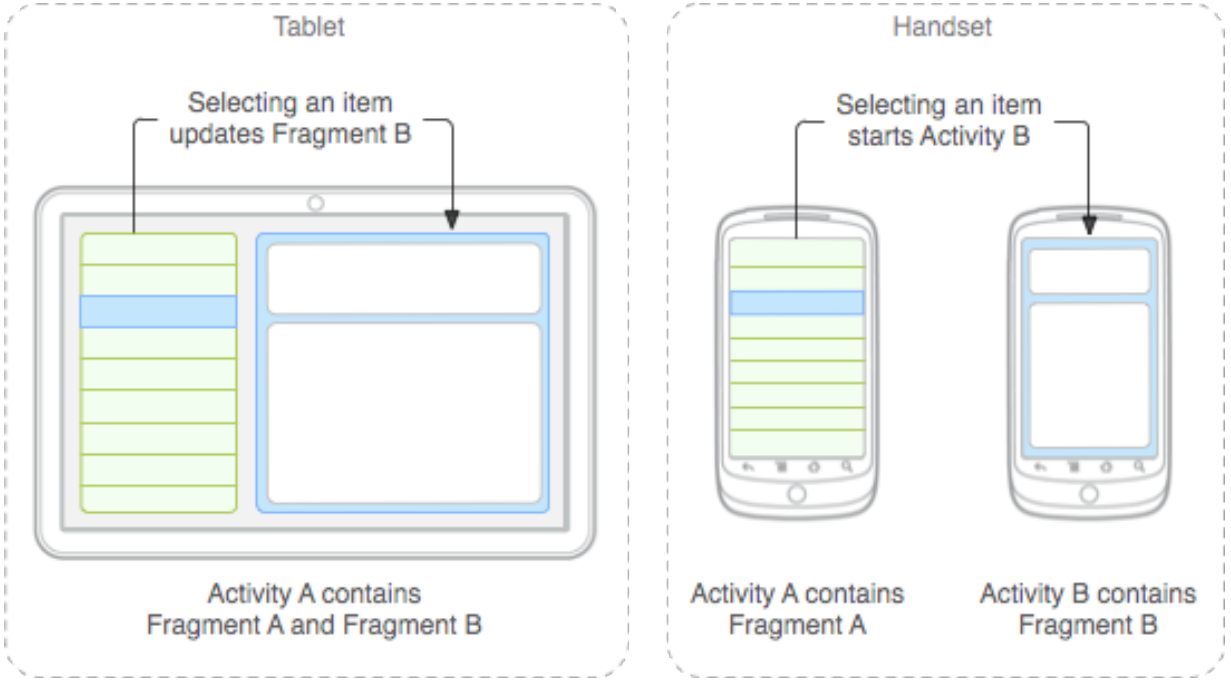


Reference: https://developer.android.com/guide/components/fragments.html

The life cycles of a Fragment

Introduction

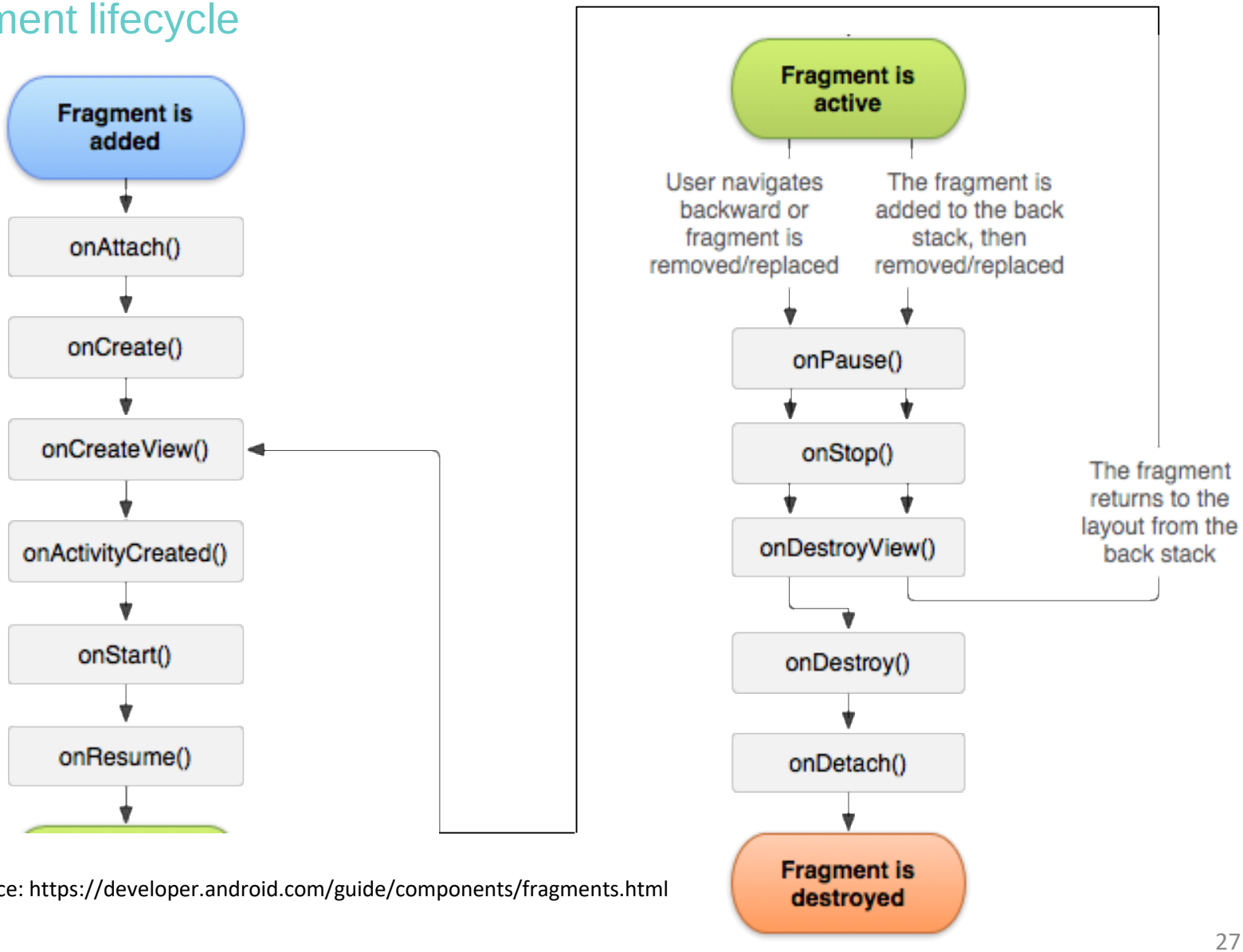
- You should design each fragment as a modular and reusable activity component.
- This is especially important because a modular fragment allows you to change your fragment combinations for different screen sizes.



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

The life cycles of a Fragment

Fragment lifecycle



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

The life cycles of a Fragment

Fragment Common SubClasses

- DialogFragment
  - Displays a floating dialog.
  - Using this class to create a dialog is a good alternative to using the dialog helper methods in the Activity class

- ListFragment
  - Displays a list of items that are managed by an adapter
  - Provides several methods for managing a list view

- PreferenceFragment
  - Displays a hierarchy of Preference objects as a list
  - Useful when creating a "settings" activity for your application.

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

Managing Fragments

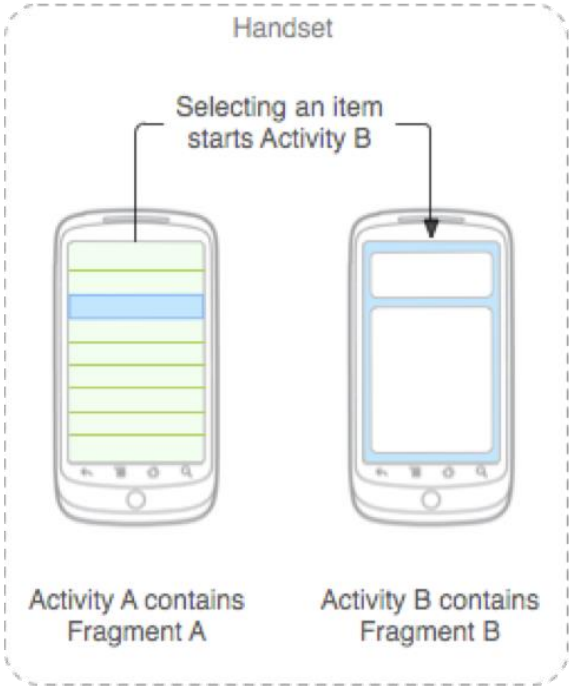
Managing Fragments

Creating Fragments

There are two ways you can add a fragment to the activity layout:

1. Declare the fragment inside the activity's layout file.

2. Or, programmatically add the fragment to an existing ViewGroup.



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

Managing Fragments

Creating Fragments - Static

Declare the fragment inside the activity's layout file

```
<?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>
```

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

Managing Fragments

Creating Fragments - Dynamic

Programmatically add the fragment to an existing ViewGroup

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

```
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

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

Services

Introduction to Services

Types

A service can essentially take two forms:

Started

A service is "started" when an application component (such as an activity) starts it by calling startService().

Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.

Bound

A service is "bound" when an application component binds to it by calling bindService().

A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).

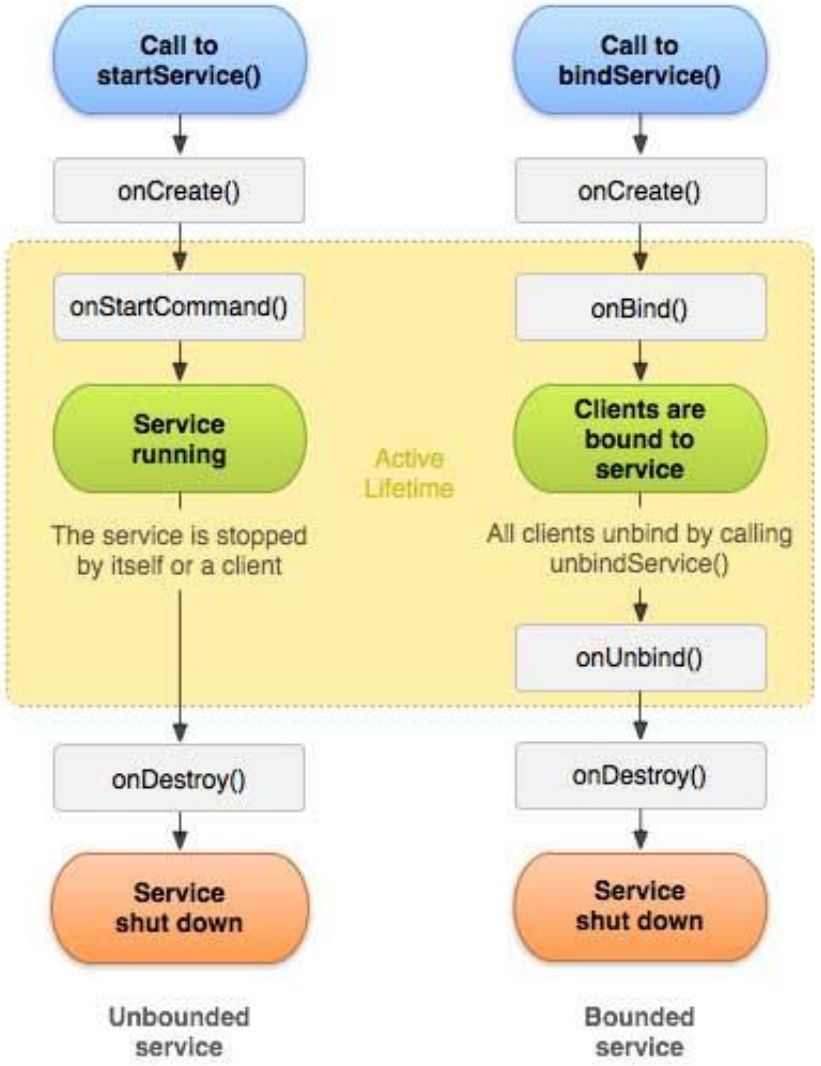
A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

https://developer.android.com/guide/components/services.html

Service Lifecycle

Started vs Bound

Although a started service is stopped by a call to either stopSelf() or stopService(), there is not a respective callback for the service (there's no onStop() callback). So, unless the service is bound to a client, the system destroys it when the service is stopped—onDestroy() is the only callback received.



https://developer.android.com/guide/components/services.html

Questions?