가천대학교
Gachon University

학생 여러분 반갑습니다.

다른 친구들이 입장할 때까지 조금 기다려 주십시오.

곧 모바일 프로그래밍 수업을 시작합니다.

음소거( )가 되었는지 확인 바랍니다.

모바일 프로그래밍
화목(1,2교시)/ 화목(3,4교시)
정윤현 (AI/소프트웨어학부)

Mobile Programming

Android Programming

Chap 4-2. Application Basics

Prof. Younhyun Jung

Email) younhyun.jung@gachon.ac.kr

가천대학교 소프트웨어학과
Gachon University

Activity Life Cycle : Activity States

- Activities are created, suspended, resumed & destroyed as necessary when an application executes
  - Understanding the life cycle of an activity is vital to ensuring that your application works correctly.

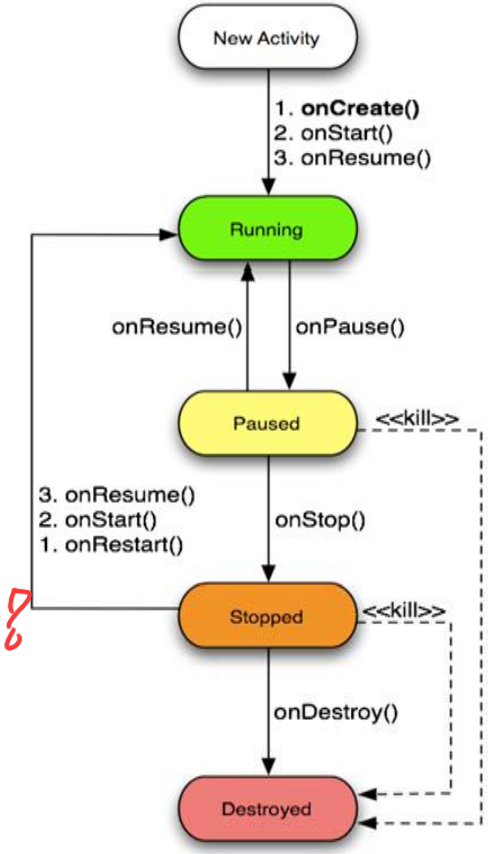
<> 액티비티가 어떤 상태인지 알고 있어야 함!

Activity's Life Cycle States

- An activity has essentially three states

1. Active/Running
2. Paused → focus 잃었을 때
3. Stopped

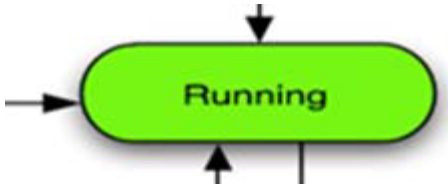
Simplified State Transition Diagram



각 상태가 변할 때 항상 호출된다!

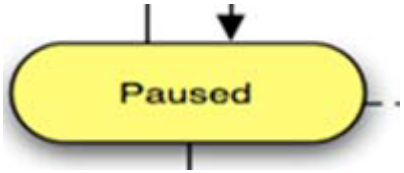
MUST READ:
Training:
http://developer.android.com/training/basics/activity-lifecycle/index.html
http://developer.android.com/reference/android/app/Activity.html
http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

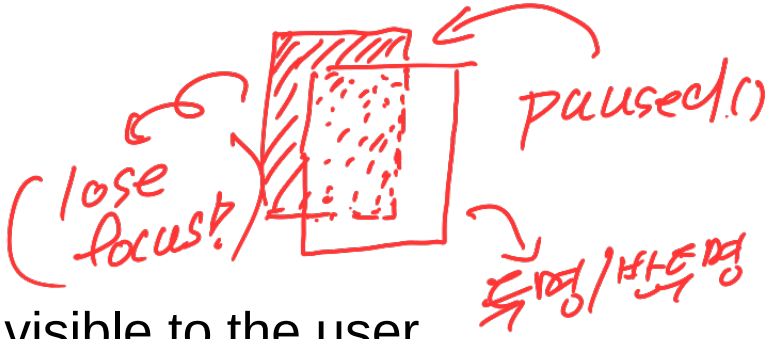
Life Cycle State : Active or Running



- It is active or running when it is in the foreground of the screen (at the top of the activity stack for the current task).
- This is the activity that is the focus for the user's actions.

Life Cycle State : Paused





- It is **paused** if it has lost **focus** but is still visible to the user.
  - That is, another activity lies on top of it and that new activity either is transparent or doesn't cover the full screen.
- A paused activity is completely alive
  - maintaining its state information and attachment to the window manager

NOTE: Paused activities can be killed by the system when available memory becomes extremely low.

자원이 많이 부족할 때 될 수 있음!?

Life Cycle State : Stopped

- An Activity is stopped if it is completely obscured (hidden) by another activity.
- Although stopped, it still retains all state and member information.
- However, it is no longer visible to the user so its window is hidden.

NOTE: Stopped activities can be killed by the system when available memory becomes extremely low.

Activity Life Cycle Events

- Life Cycle States
  - When progressing from one state to the other,
    the OS notifies the application of the changes by issuing calls to the following transition methods (state-change handling methods or callbacks):

void onCreate(Bundle savedInstanceState) void onStart() void onRestart() void onResume()	void onPause() void onStop() void onDestroy()

Base class Activity

- `Activity` base class defines a series of events that governs the life cycle of an activity.

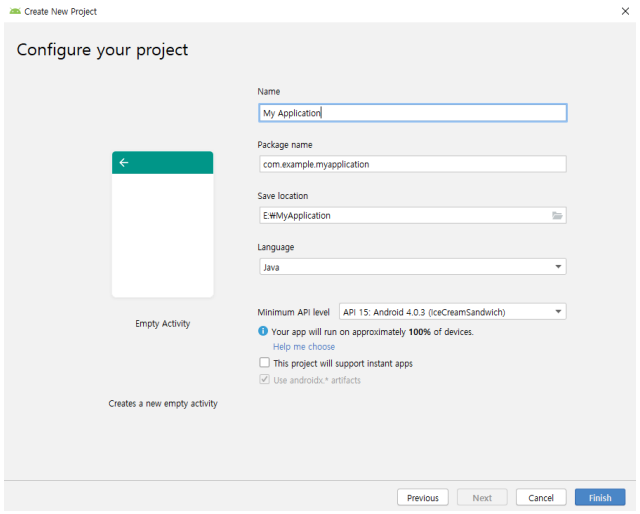
- The `Activity` class's basic methods are
  - **onCreate() :** Called when the activity is first created
  - **onStart() :** Called when the activity becomes visible to the user
  - **onResume() :** Called when the activity starts interacting with the user
  - **onPause() :** Called when the current activity is being paused
  - **onStop() :** Called when the activity is no longer visible to the user
  - **onDestroy() :** Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
  - **onRestart() :** Called when the activity has been stopped and is restarting again

Exercise

Goal : Understanding the Life Cycle of an Activity

TRY IT OUT:

Step 1.



Step 2.

See:
http://developer.android.com/reference/android/util/Log.html

Exercise (Cont.)

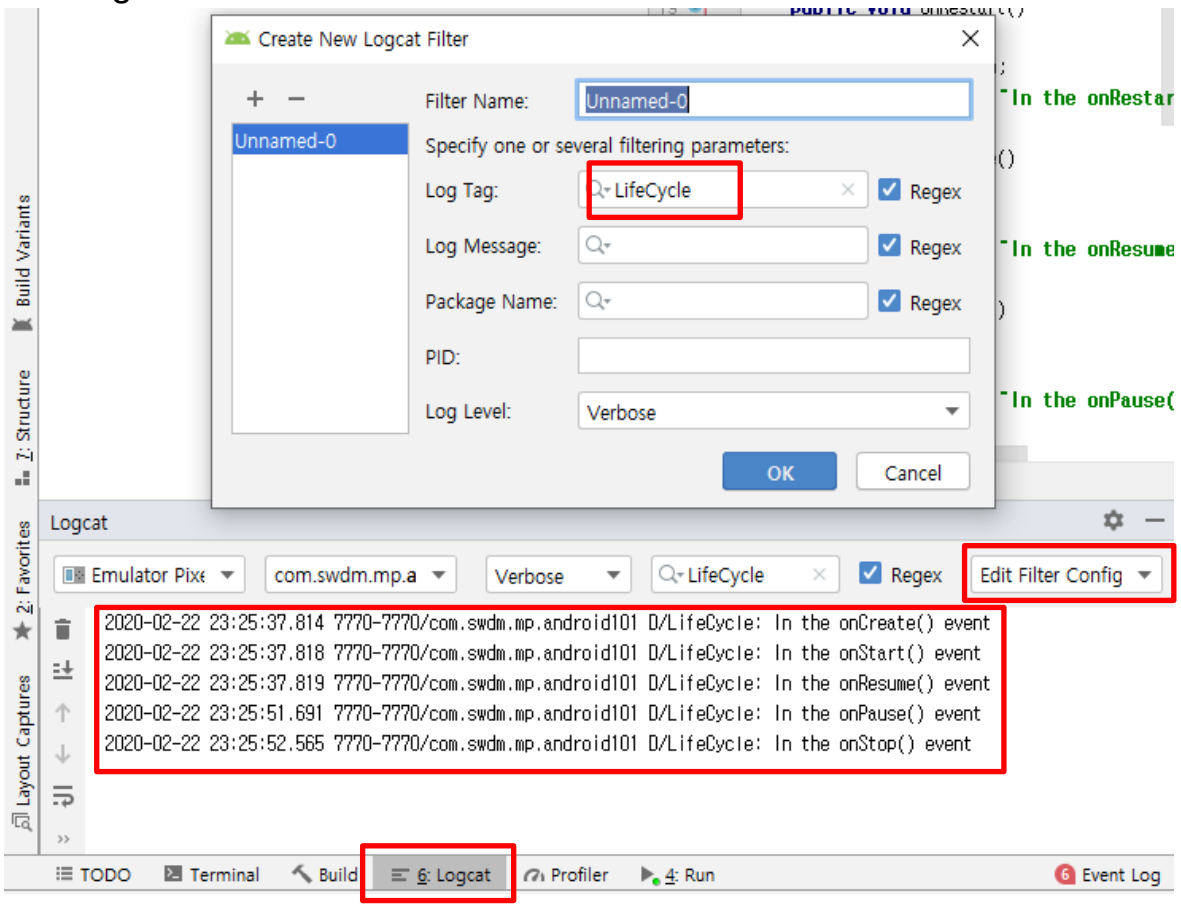
- In the **MainActivity.java** file, add the following codes in red color:

```
package com.swdm.mp.android101; // your own package name
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends AppCompatActivity {
        String tag = "LifeCycle";
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                Log.d(tag, "In the onCreate() event");
        }
        public void onStart()
        {
                super.onStart();
                Log.d(tag, "In the onStart() event");
        }
        public void onRestart()
        {
                super.onRestart();
                Log.d(tag, "In the onRestart() event");
        }
```

```
  public void onResume()
  {
        super.onResume();
        Log.d(tag, "In the onResume() event");
  }
  public void onPause()
  {
        super.onPause();
        Log.d(tag, "In the onPause() event");
  }
  public void onStop()
  {
        super.onStop();
        Log.d(tag, "In the onStop() event");
  }
  public void onDestroy()
  {
        super.onDestroy();
        Log.d(tag, "In the onDestroy() event");
   }
}
```

Exercise (Cont.)

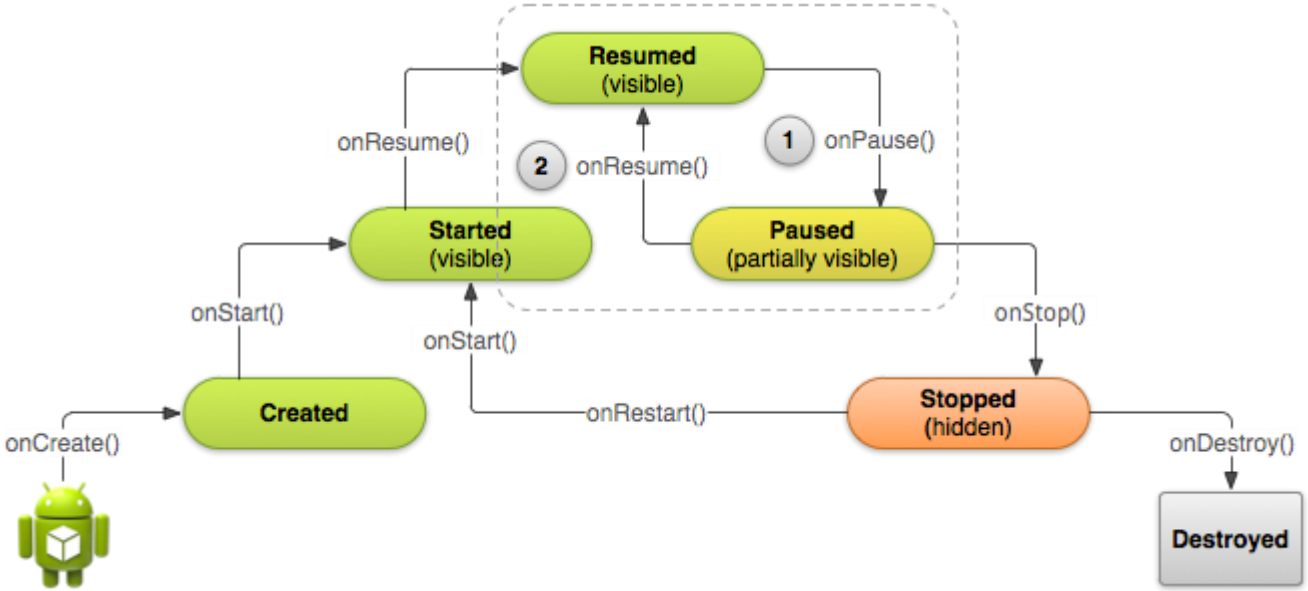
- RUN!!
- Check the logcat
  - Edit filter to represent selected Log
  - And Press Back Button



Lifetime Summary

Complete / Visible / Foreground Lifetime

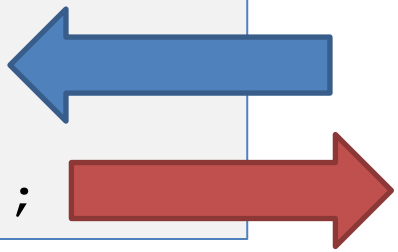
- An activity begins its lifecycle when entering the onCreate() state.
- If not interrupted or dismissed, the activity performs its job and finally terminates and releases its acquired resources when reaching the onDestroy() event.



Android Bundles

- The Android Bundle container is a simple mechanism used to pass data between activities.
- A **Bundle** is a type-safe collection of **<name, value>** pairs.
- There is a set of ***putXXX*** and ***getXXX*** methods to store and retrieve (single and array) values of primitive data types from/to the bundles. For example

```
Bundle myBundle = new Bundle();
myBundle.putDouble ("var1", 3.1415);
...
Double v1 = myBundle.getDouble("var1");
```



Android Intents & Bundles

Activity1: Sender → **Activity2: Receiver**

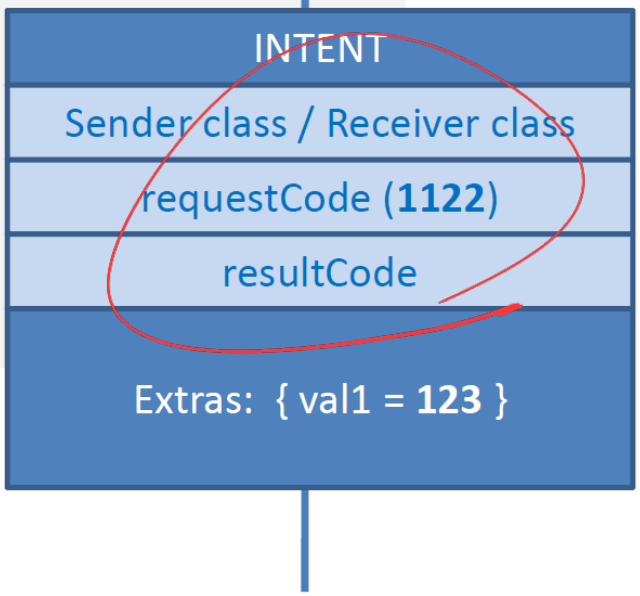
```
Intent myIntentA1A2 = new Intent  (Activity1.this,
                                   Activity2.class);

Bundle myBundle1 = new Bundle();
myBundle1.putInt ("val1", 123);

myIntentA1A2.putExtras(myBundle1);

startActivityForResult(myIntentA1A2, 1122);
```

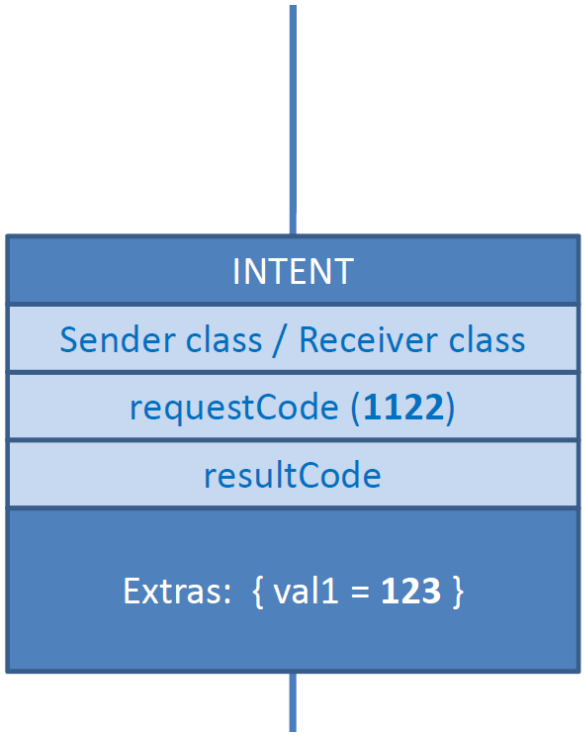
Bundle를 더 많은 데이터 넣을 수 있을?



Android Intents & Bundles

Activity1: Sender

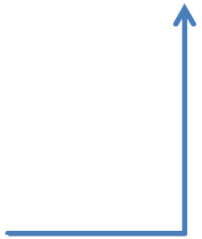
Activity2: Receiver




```
Intent myCallerIntent  = getIntent();

Bundle myBundle = myCallerIntent.getExtras();

int val1 = myBundle.getInt("val1");
```



Android Intents & Bundles

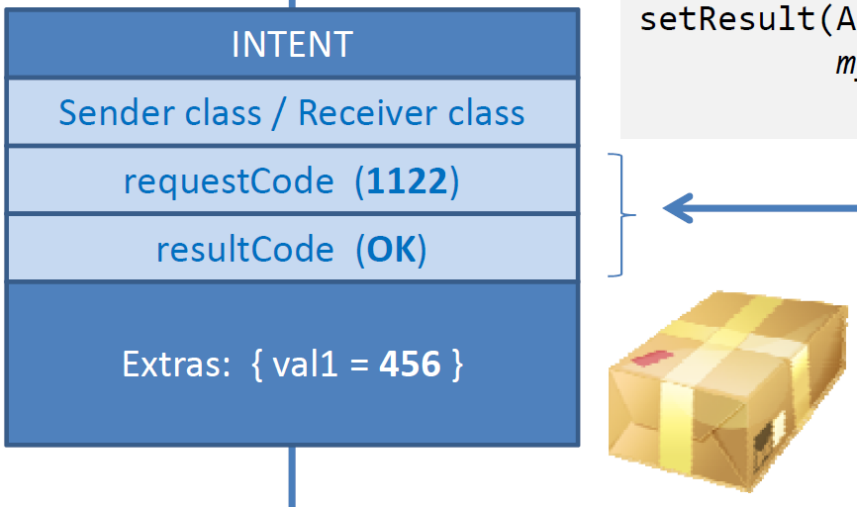
Activity1: Sender

Activity2: Receiver

```
myBundle.putString("val1", 456 );

myCallerIntent.putExtras(myBundle);

setResult(Activity.RESULT_OK,
          myCallerIntent);
```

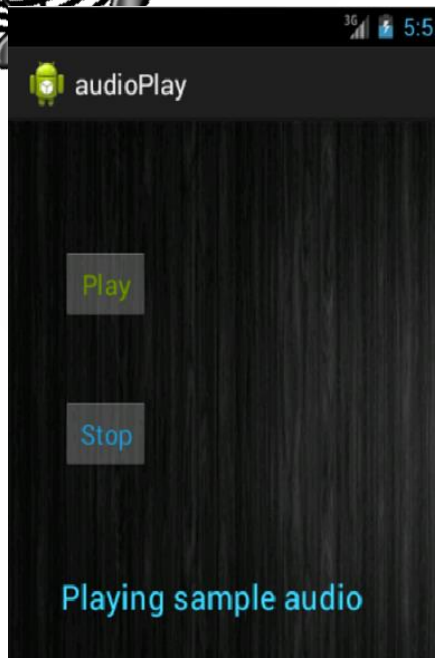


통신

Service

- A service is a special type of component that does not have a visual user interface.
- Runs in the background to perform long-running operations
- Applications can start their own services or interact with other services already active.
- Example:
  - Background GPS service
    - Run in the background, detecting satellites, phone towers or Wi-Fi AP location information
    - Service periodically broadcasts location coordinates to any applications listening for that kind of data
    - An App may opt for binding to the running GPS service
  - Playing sound/music

Service



- Example
  - A music service (Pandora radio) and GPS location run in the background. The selected music station is heard while other GUIs are shown on the screen.

Starting a Service

- Call **startService**

```
startService(new Intent(this, MyService.class));
```

The onStartCommand() method is called when you start the service explicitly using the startService() method.

- Call **stopService**

```
stopService(new Intent(this, MyService.class));
```

The onDestroy() method is called when the service is stopped using the stopService() method or stopSelf() method. This is where you clean up the resources used by your service.

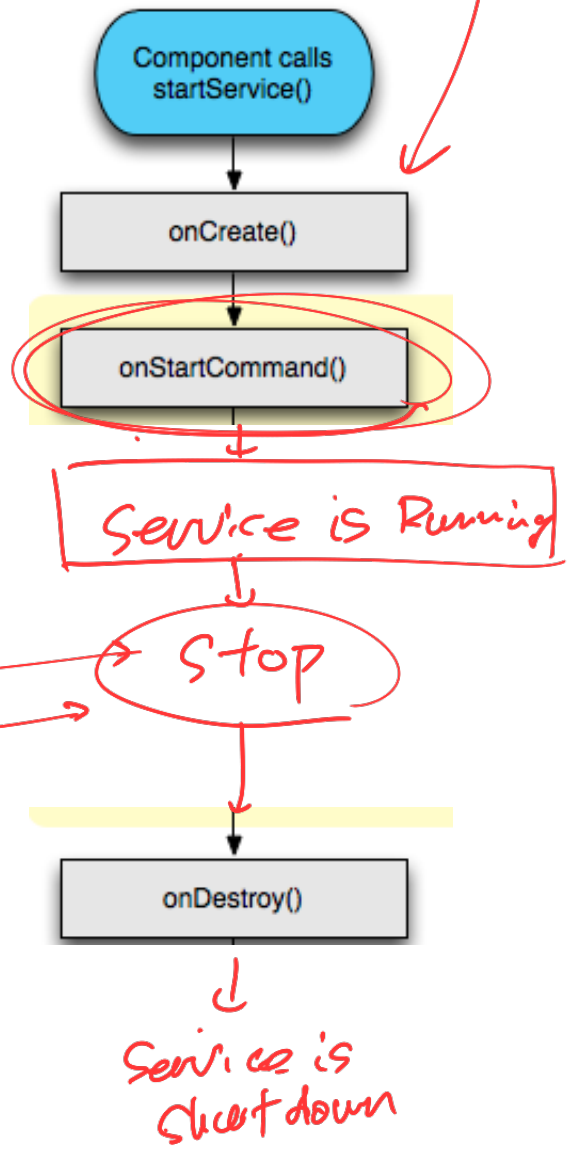
Service Life Cycle

- A Service has three lifecycle methods:

- **1.void onCreate()**

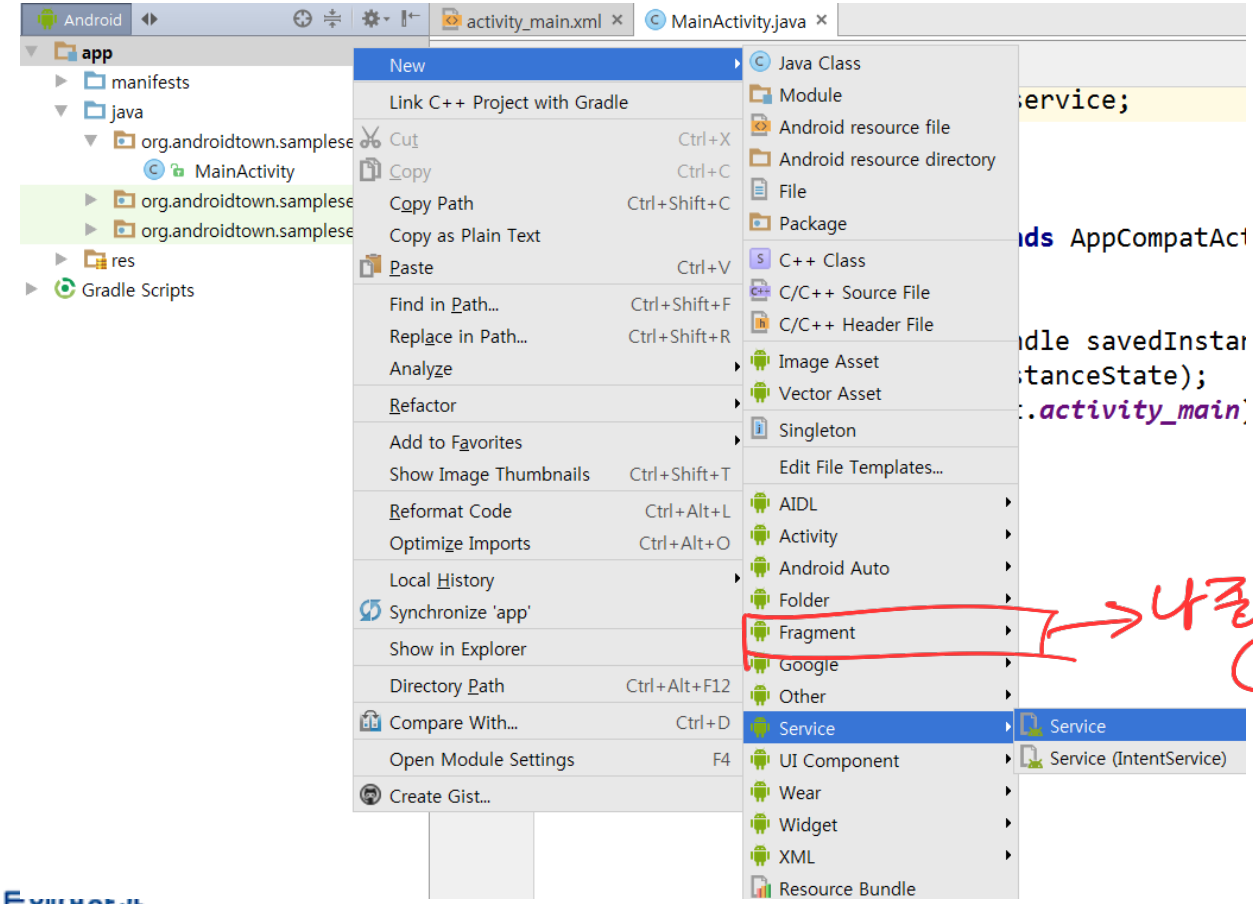
- **2.void onStartCommand()**
  - called when the service is started by another component via a call to the **startService()** method.

- **3.void onDestroy()**
  - The service will continue running until stopService() or stopSelf() is called.



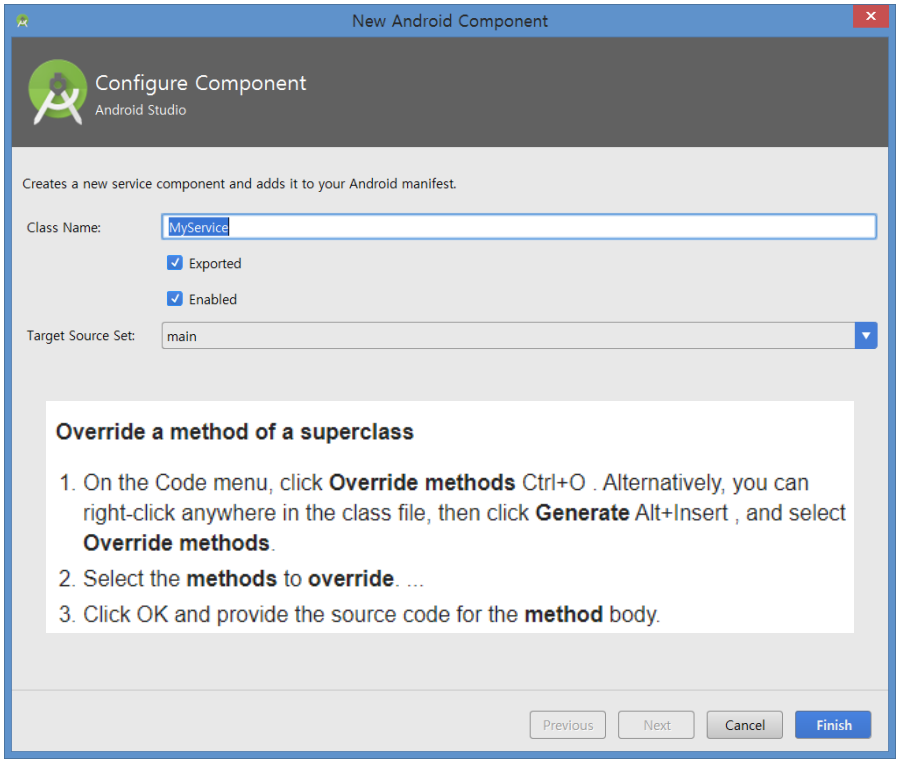
Exercise

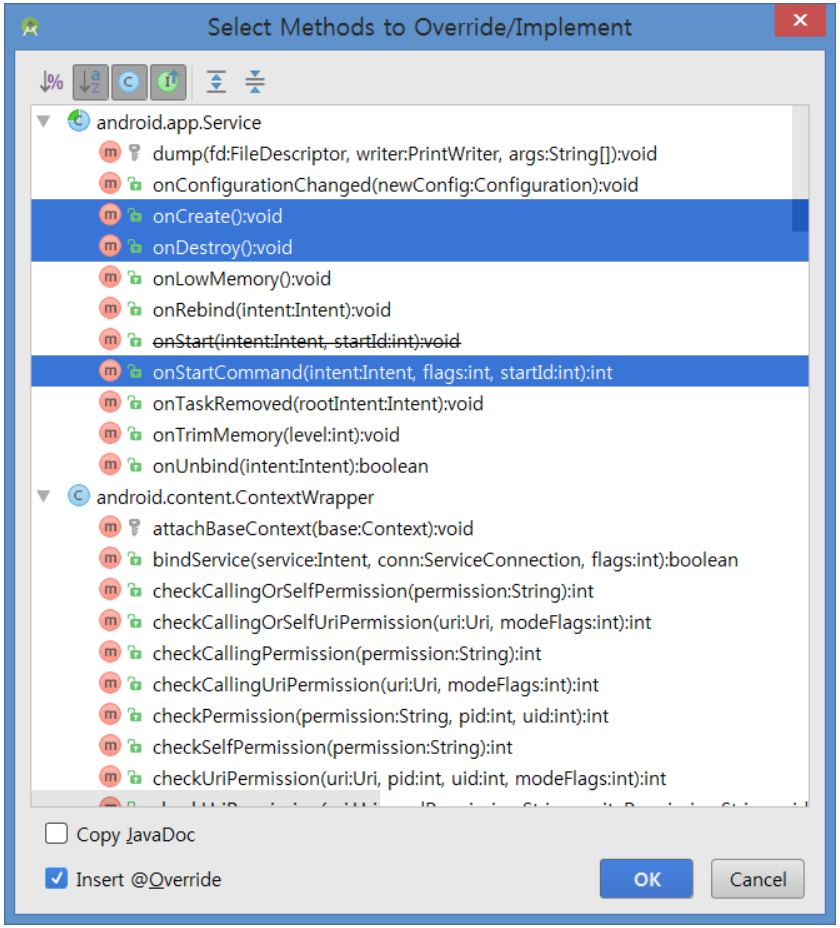
- Add new service (New→Service→Service)



Exercise (cont.)

- (Ctrl+O) Add three methods to override (onCreate, onDestory, onStartCommand)





Exercise (cont.)

- Add methods to override

```
package com.example.service_test;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service {
    public MyService() {
    }

      @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        return START_STICKY;
      }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

START_STICKY
tells the OS to recreate the service after it has enough memory and call onStartCommand() again with a null intent.
서비스가 강제종료 되었을 경우, 재시작함

Exercise (cont.)

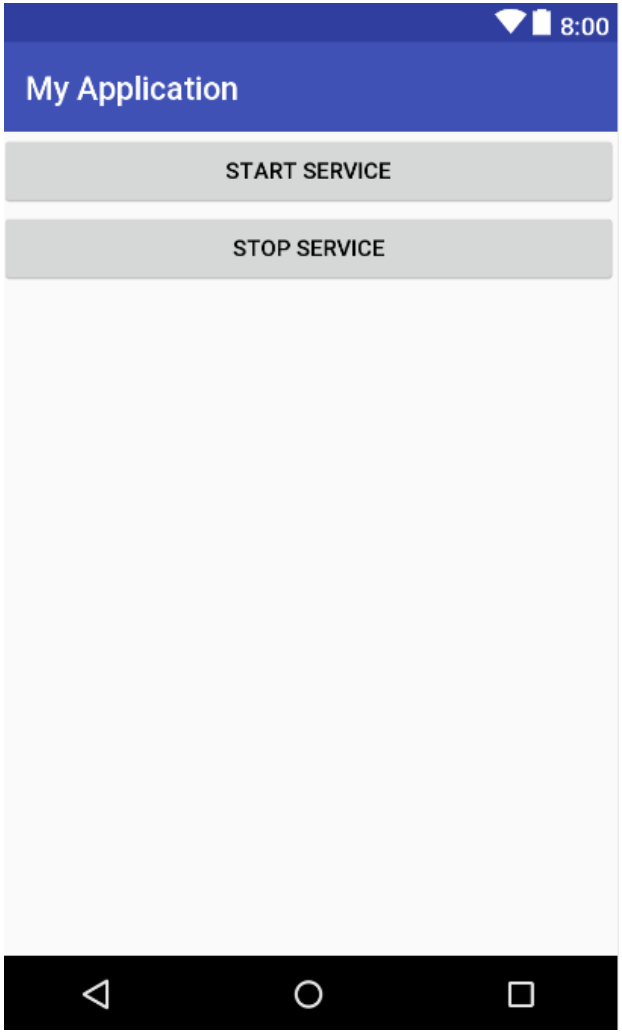
- xml file

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/btnStartService"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Start Service" />

    <Button
        android:id="@+id/btnStopService"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Stop Service" />

    </LinearLayout>
```



Exercise (cont.)

- MainActivity

```
package com.example.service_test;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnStart = findViewById(R.id.btnStartService);
        btnStart.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View v) {
                startService(new Intent(getApplicationContext(), MyService.class));
            }
        });

        Button btnStop = findViewById(R.id.btnStopService);
        btnStop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                stopService(new Intent(getApplicationContext(), MyService.class));
            }
        });
    }
}
```

startActivity(new Intent(context, class))

Then, RUN!

References

- See
  - http://developer.android.com/training/basics/activity-lifecycle/index.html
  - http://developer.xamarin.com/guides/android/application_fundamentals/activity_lifecycle/

- "Beginning Android 4 Application Development," Ch-2.
- 2017, 정재곤, "Do it! 안드로이드 앱 프로그래밍(개정4판)", 이지스퍼블리싱(주)