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 4-1. Application Basics

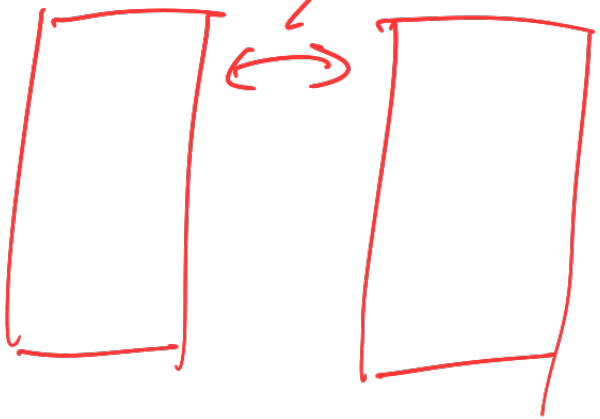
Prof. Younhyun Jung

Email) Younhyun.jung@gachon.ac.kr

가천대학교 소프트웨어학과
Gachon University

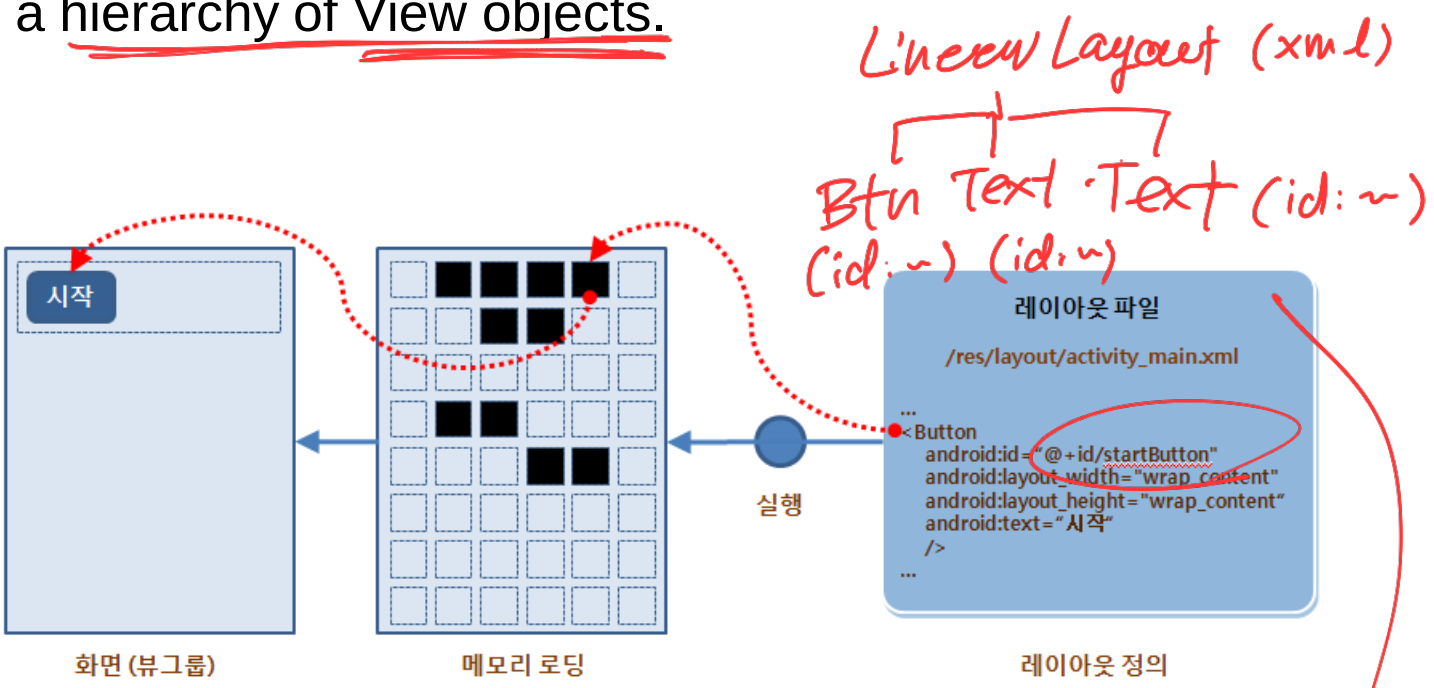
Contents

- Understanding Layout Inflation
- Activity & Intent & Activity Lifecycle
- Concept of Service & Broadcast



Understanding Layout Inflation

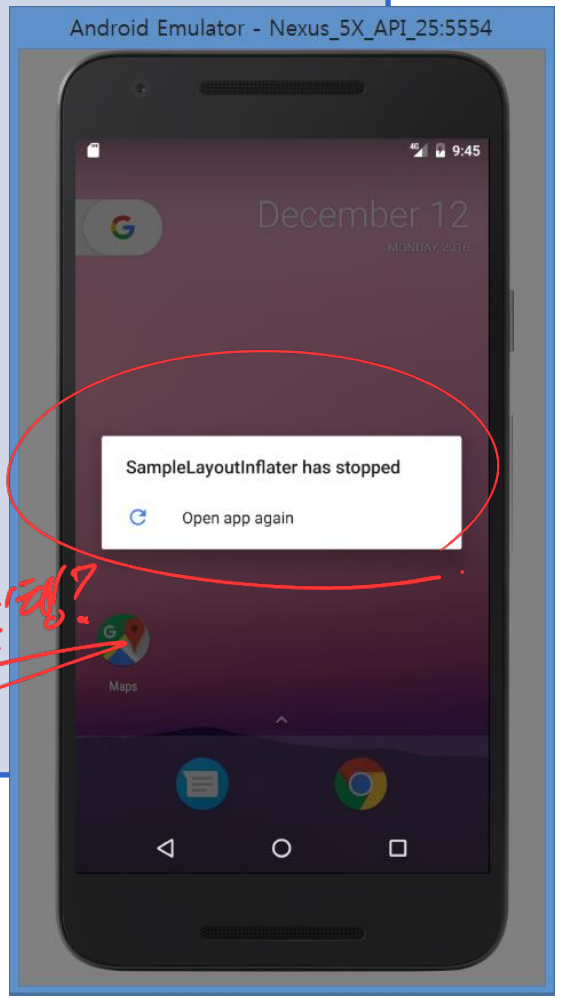
- Layout Inflation?
  - Layout inflation is the term used within the context of Android to indicate when an XML layout resource is parsed and converted into a hierarchy of View objects.



2017, 정재곤, "Do it! 안드로이드 앱 프로그래밍(개정4판)", 이지스퍼블리싱(주)

Understanding Layout Inflation

```
public class MainActivity extends AppCompatActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Button button = (Button) findViewById(R.id.button);
    button.setText("Start " );

    setContentView(R.layout.activity_main);
  }
}
```

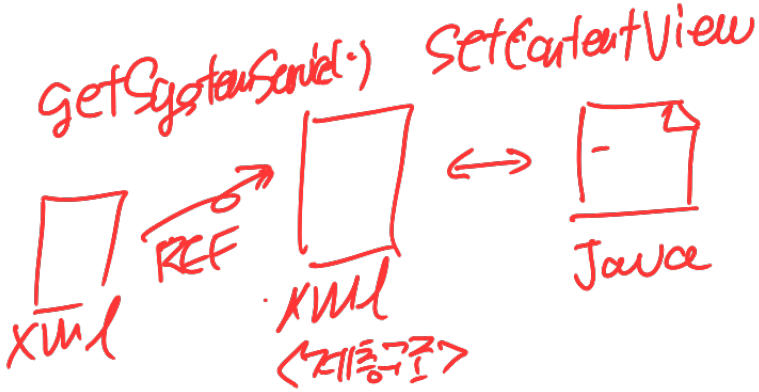


Why???

Understanding Layout Inflation

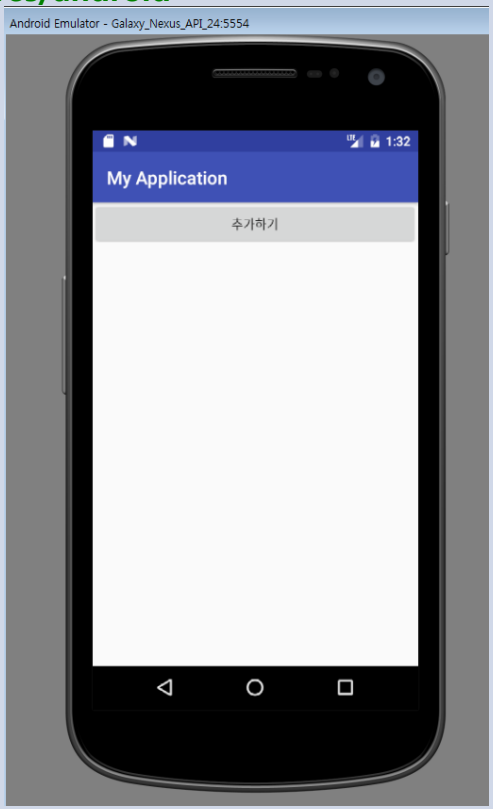
- For main Layout Inflation (e.g., activity_main.xml) : Conduct Layout Inflation by using setContentView method
  - setContentView(R.layout.activity_main);

- For sub Layout Inflation (e.g., sub1.xml) : Conduct manually Layout Inflation
  - getSystemService(Context.Layout_INFLATER_SERVICE)



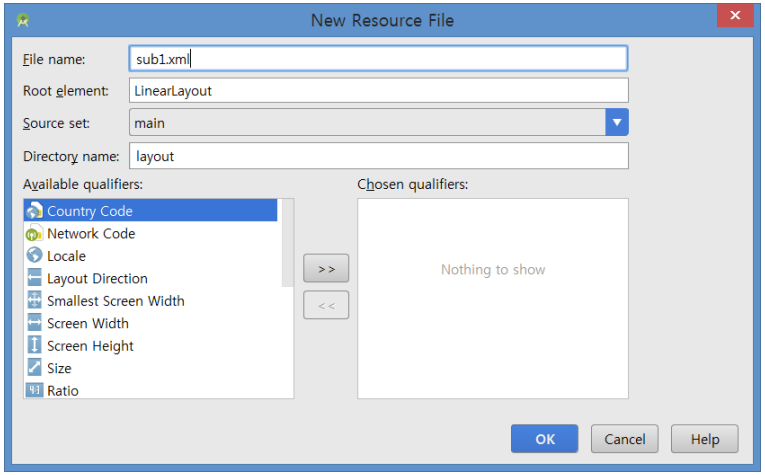
Exercise

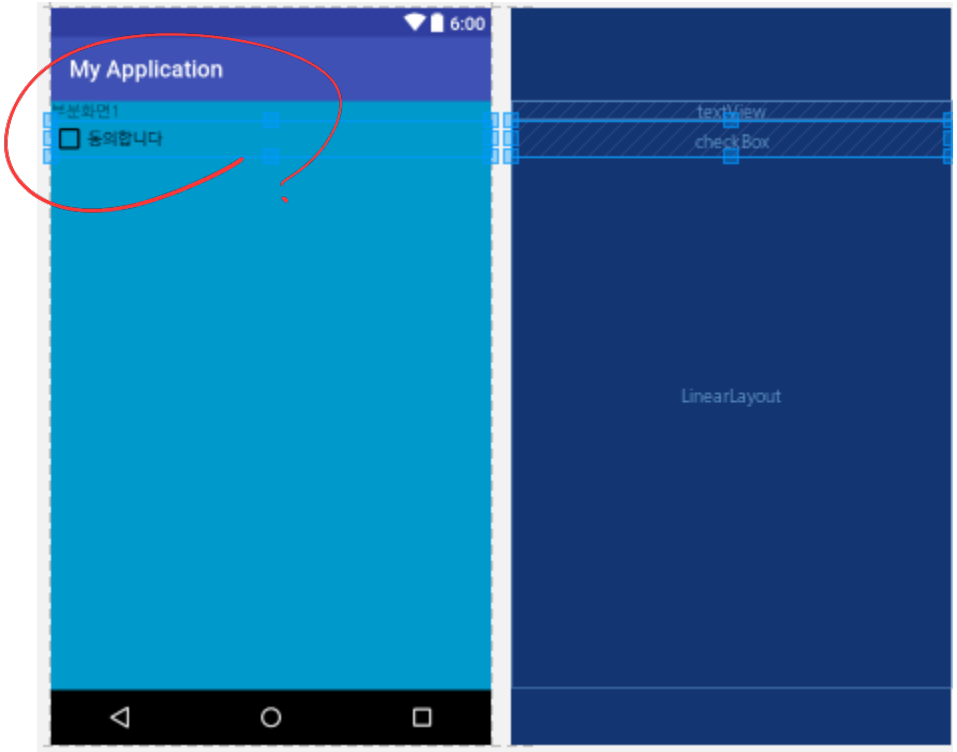
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

<Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="추가하기" />

    <LinearLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

    </LinearLayout>

</LinearLayout>
```



Cont.

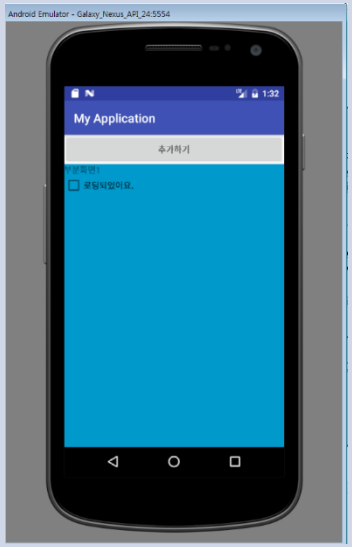
- Making sub1.xml file and then add the textview & checkbox widget as shown below:





Cont.

```
LinearLayout container;
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  container = (LinearLayout) findViewById(R.id.container);

  Button button = (Button) findViewById(R.id.button);
  button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
      inflater.inflate(R.layout.sub1, container, true);

      CheckBox checkBox = (CheckBox) container.findViewById(R.id.checkBox);
      checkBox.setText("로딩되었어요.");
    }
  });
}
```



System 서비스를 통해서 Layout 변경.

https://developer.android.com/reference/android/view/LayoutInflater

Android App Components

- Android App components are the essential building blocks of an Android app.
  - An Android application consists of one or more *core components*.
- Four core components
  - ***1. Activity*** → 실행화면
  - ***2. Service***
  - ***3. Broadcast receiver***
  - ***4. Content provider***

Application Building Blocks

Activity

- UI Component typically corresponding to one screen.

Service

- Faceless task that runs in the background.

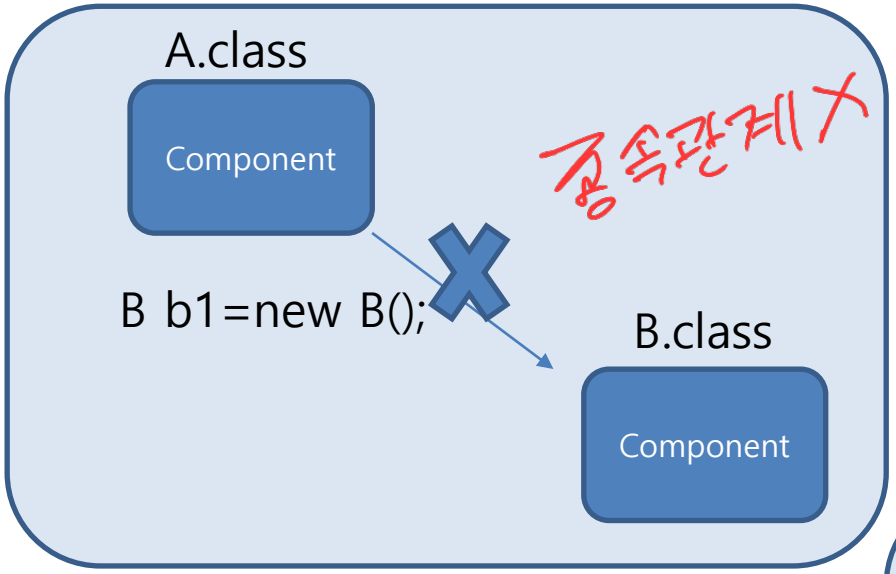
Content Provider

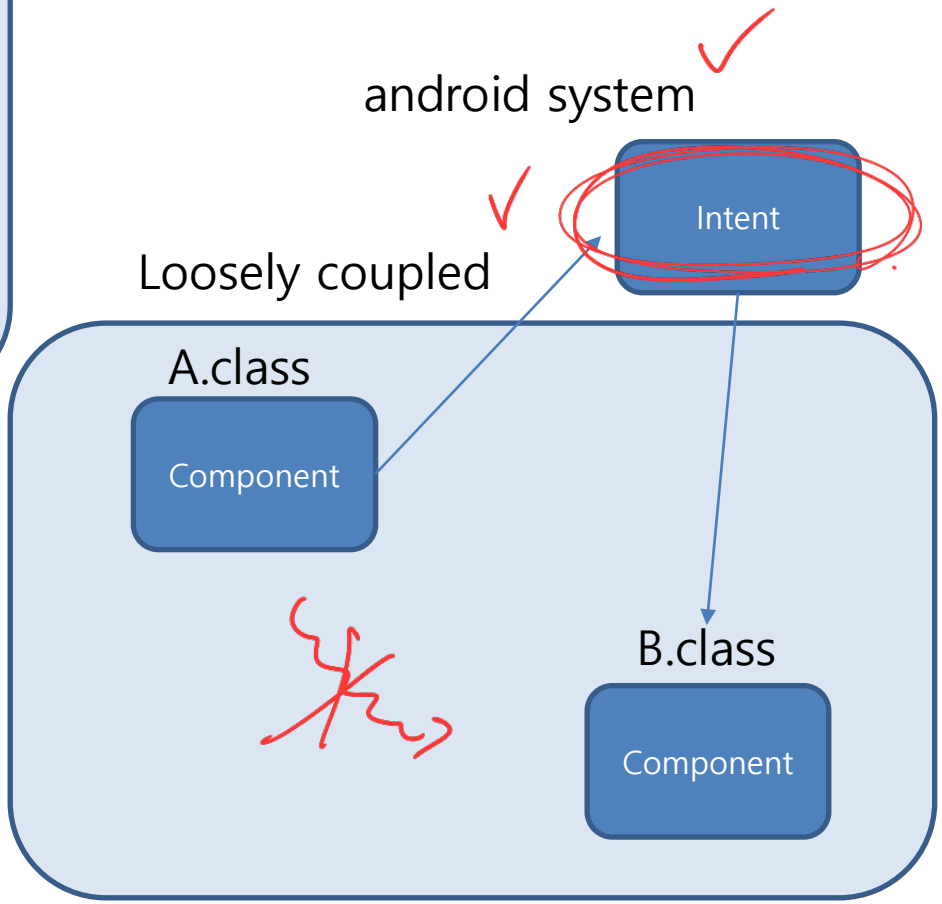
- Enable applications to share data.

Broadcast Receiver

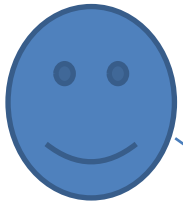
- Responds to notifications or status changes.

Components based Development in Android

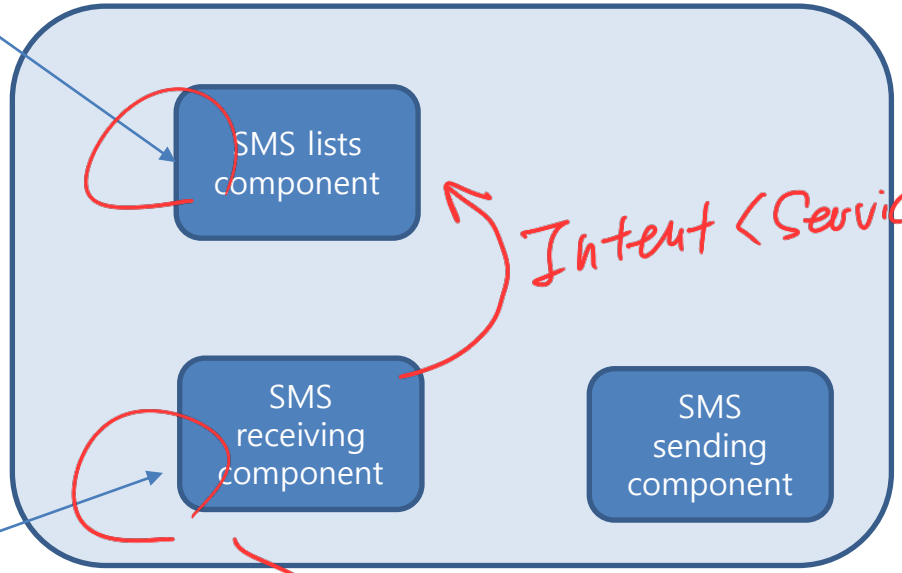




Components based Development in Android

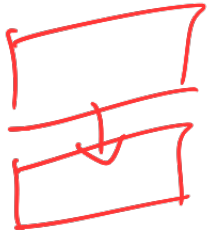


Starting an app by a user



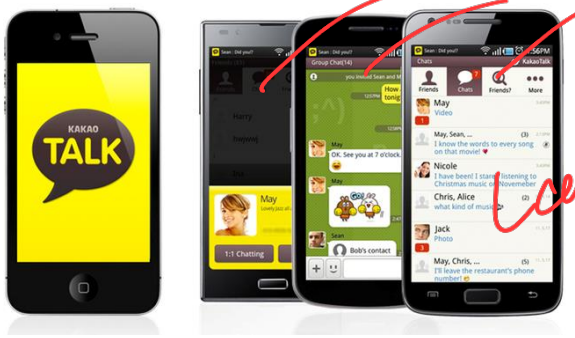
Intent <Service>

Starting an app when receiving SMS



Activity

- Primary class for User Interaction (UI)
- An activity represents a single screen with a user interface.
  - An *activity* usually shows a *single visual user interface* (GUI).
  - An activity is roughly equivalent to a Windows-Form
    - (or a Web-page).
- A typical Android application consists of *one or more* activities.
    - c.f., A web-site consists of one or more web-pages

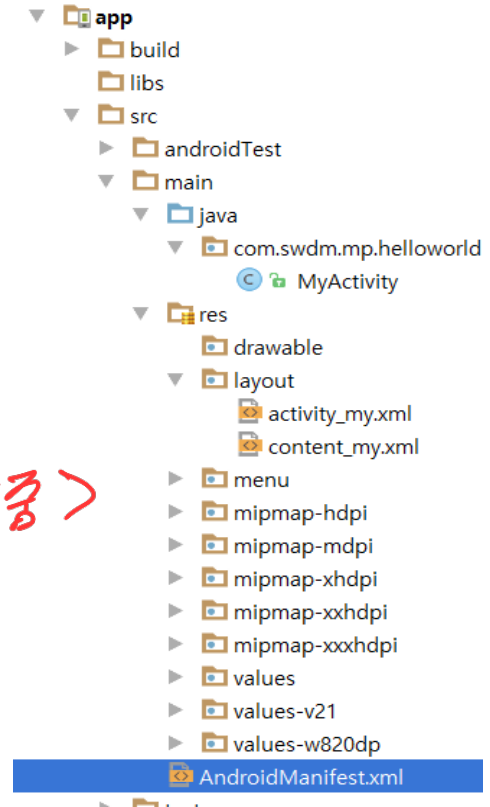


NOTE: 안드로이드에서는 일반적으로 한 화면을 'activity'라고 부름. 이 화면들이 모두 독립적으로 동작할 수 있게 만들어져 있으며 안드로이드 시스템에서 관리 됨. 따라서, 스마트폰 단말에 설치 된 앱 안에서는 어떤 화면들이 포함 되 있는지 안드로이드 OS가 알고 있어야 함

Activity

- An Android application is typically a collection of activities, an activity corresponds to a screen.
  - Of course, apps may have other components – Services, Broadcast Receivers, Content Providers.
  - An Android application consists of one or more *components* that are defined in the application's manifest file.

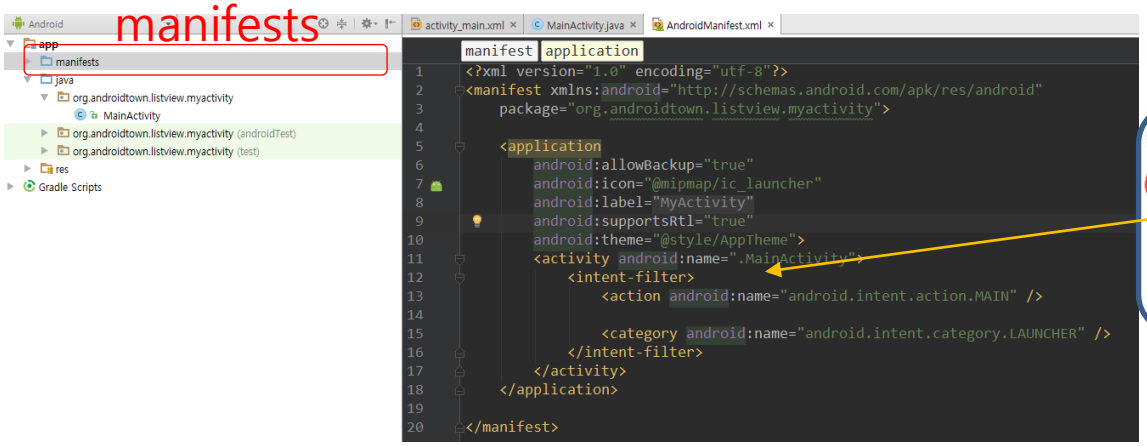
컴포넌트들이 정의됨!
(자동으로 AS가 추가 해줌)



Activity

- Although an application consists of one or more activities, ***each activity is independent of the others***.
- Typically, *one of the activities is marked as the first* one that should be presented to the user when the application is launched.
  - No main() function unlike conventional C, Java …

manifests



- Activities that can initiate applications have filter with:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name=
        "android.intent.category.LAUNCHER" />
</intent-filter>
```

처음 실행할 activity를 설정.

Activity

- Activities are independent of each other; however they usually cooperate exchanging data and actions.
- Moving from one activity to another is accomplished by asking the current activity to execute an ***intent*.**
- Intents enable activities to interact with each other in an **asynchronous** mode.

비동기 처리방식

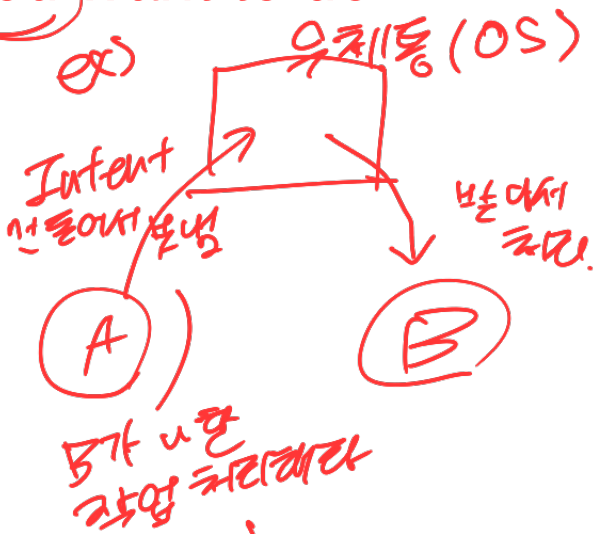
- What is an intent???

intent를 실행하는 대상 Activity에게

Activity간 정보 교환.

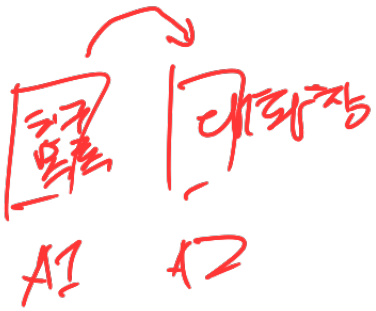
Concept of Intents

- An Intent in Android describes "*what you want to do*"
  - This may look like
    - "*I want to look up a contact record*," or
    - "*Please launch this website*," or
    - "*Show the Order Confirmation Screen.*"



- An **Intent** is a declaration of need.
- An **intent** is an abstract description of an operation to be performed.

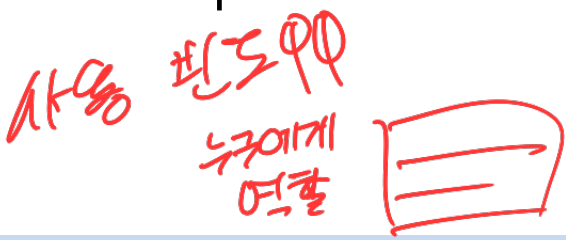
Concept of Intents (Cont.)



- An App can contain **zero or more activities**.
  - You need to navigate from one activity to another

> - You navigate between activities through what is known as an **Intent**

- Understanding the concept of Intent
  - This is very important but somewhat abstract concept – is not easy
  - Let's try it through experiencing it !

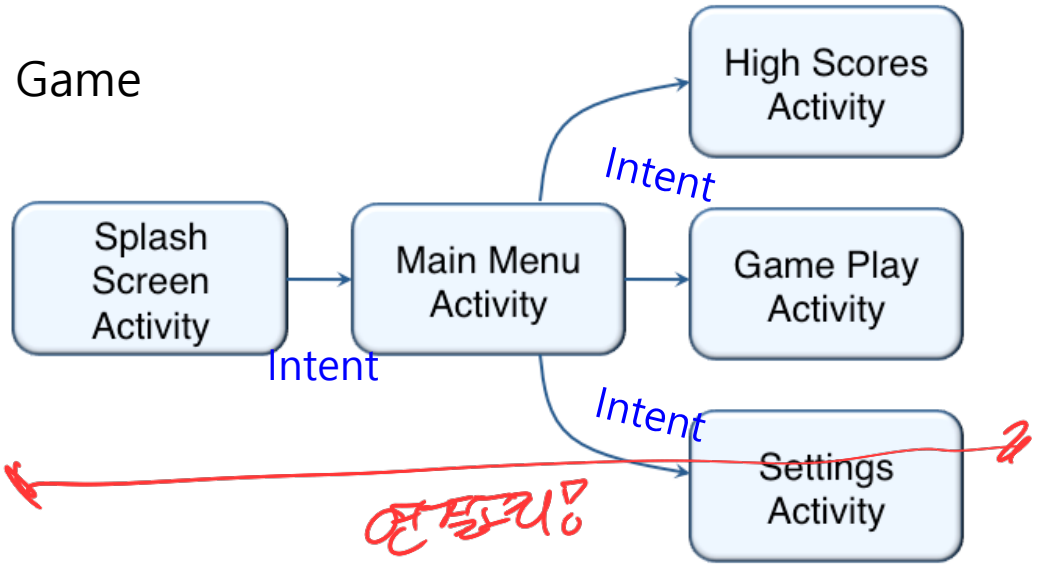


NOTE : Intent는 데이터를 전달하기 위해 안드로이드에서 만들어준 객체 타입. 화면을 띄울 때 사용하는 startActivity라는 함수 상자로 넘겨주면 안드로이드 시스템으로 전달됨

Concept of Intents (Cont.)

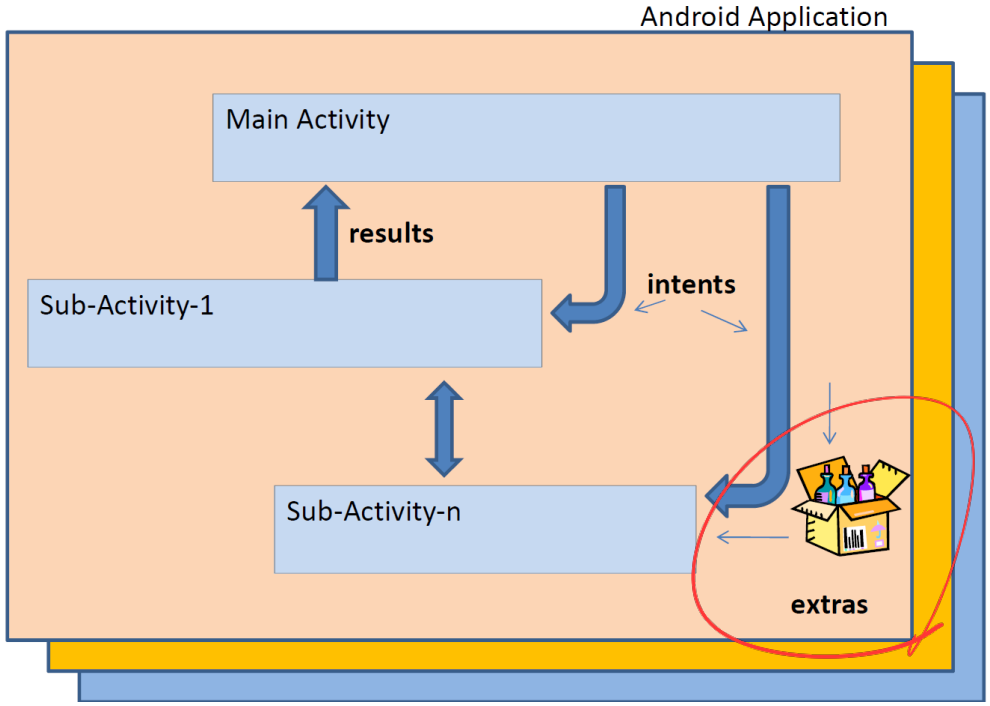
- Its most significant use is in the *launching of activities*, where it can be thought of as the glue between activities.
  - It enables/allows
    - activation of different components – activities, services, and broadcast receivers
    - Communication between loosely-connected components

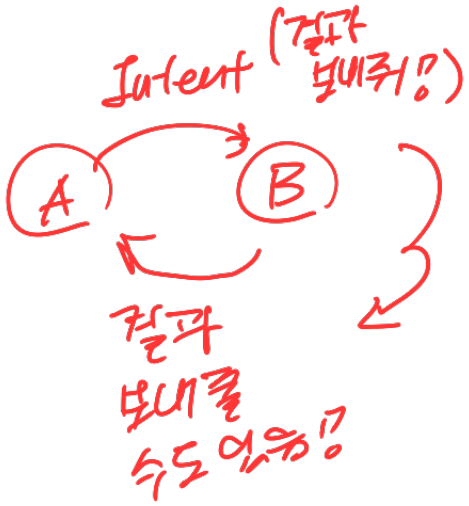
Example: Typical Game



Intents

- Android Activities



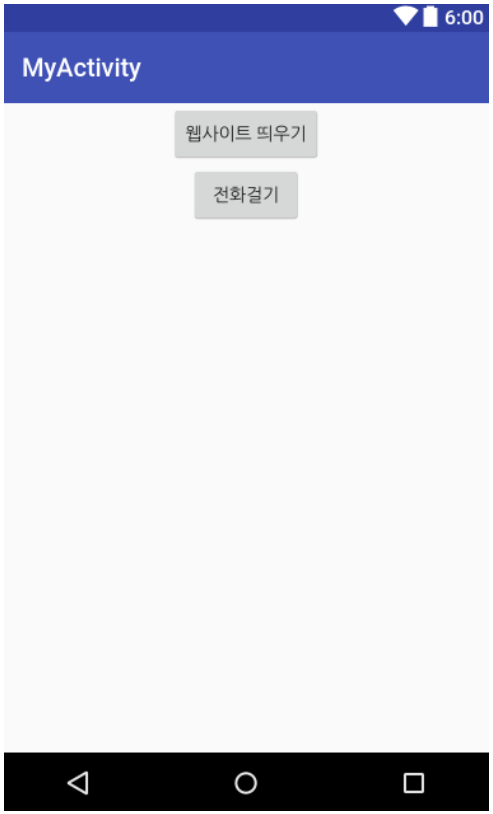


Activities call each other using Intents. An intent may include basic and extra data elements. The called activity may return a result to the caller.

Exercise

- How to call built-in app in smartphone using Intent
  - Understanding basic concept of intent
  - Make a xml file as follows:

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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="웹사이트 띄우기"
        android:layout_gravity="center"/>

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="전화걸기"
        android:layout_gravity="center"/>

</LinearLayout>
```



Exercise

- How to call built-in app in smartphone using Intent
  - Make a java file:

"alt+enter" for automatic import

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

```
public class MainActivity extends AppCompatActivity {
    LinearLayout container;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button1 = findViewById(R.id.button1);
        Button button2 = findViewById(R.id.button2);
```

Inflation

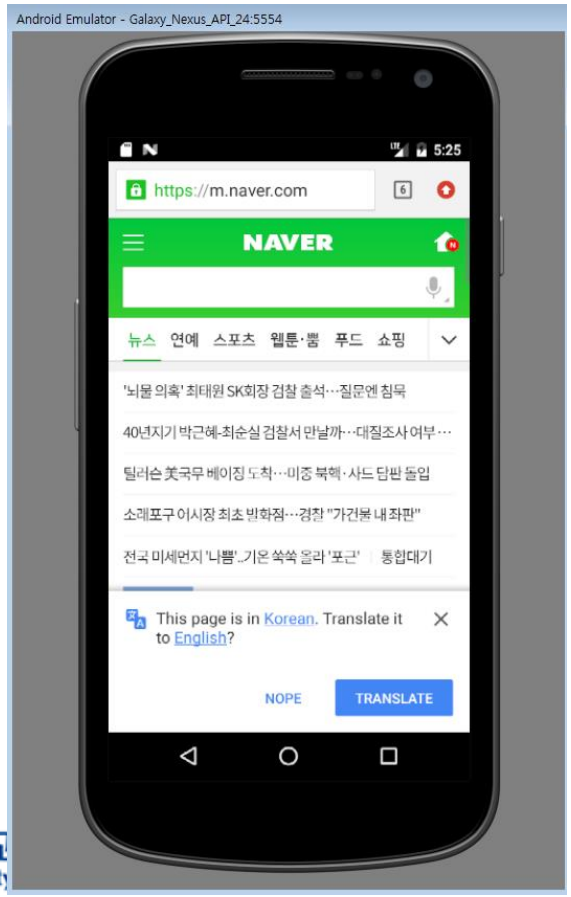
Exercise

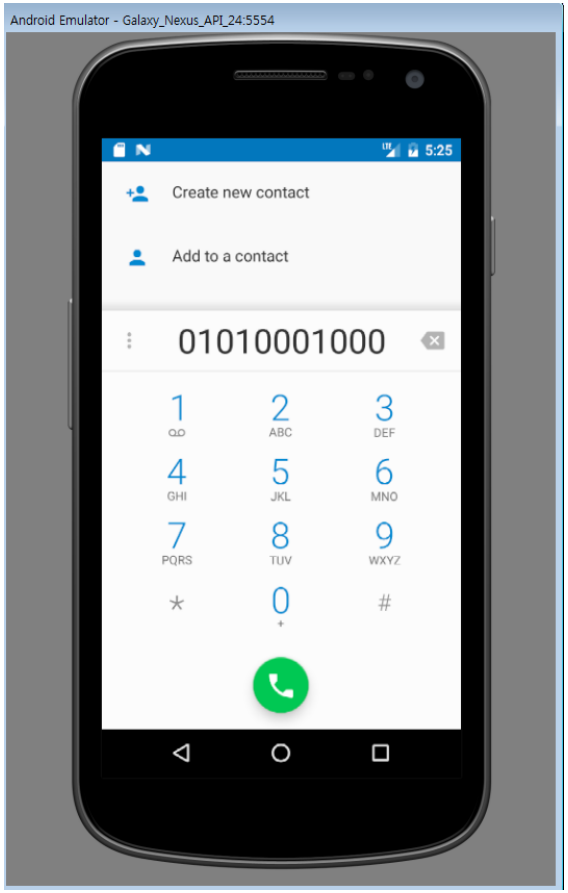
- How to call built-in app in smartphone using Intent
  - Make a java file & RUN!!

```
button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://m.naver.com"));
                startActivity(intent);
            }
        });

        button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("tel:01010001000"));
                startActivity(intent);
            }
        });
    }
}
```

Exercise

- How to call built-in app in smartphone using Intent
  - Make a java file & RUN!!

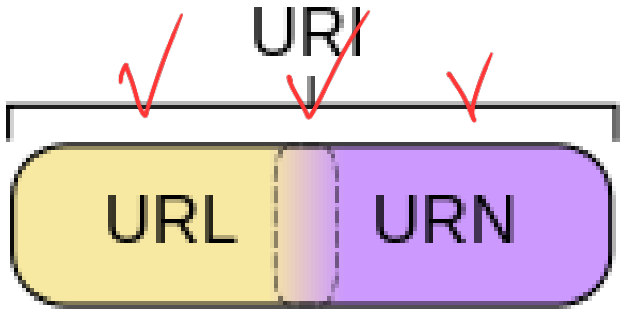




Appendix: URI?

import android.net.Uri;

- Uniform Resource Identifier (URI)
  - a string of characters used to identify a name or a web resource.
- a uniform resource locator (URL) + A uniform resource name (URN)
  - url : like a person's address
    - e.g. file:///home/username/books/
  - urn : like a person's name
    - e.g., book ISBN : ISBN 0-486-27557-4



Concept of Intents (Cont.)

- Typically an intents is called as follows:

```
Intent myActivity = new Intent (action, data);
startActivity (myActivity);
```

Built-in
or
user-created activity

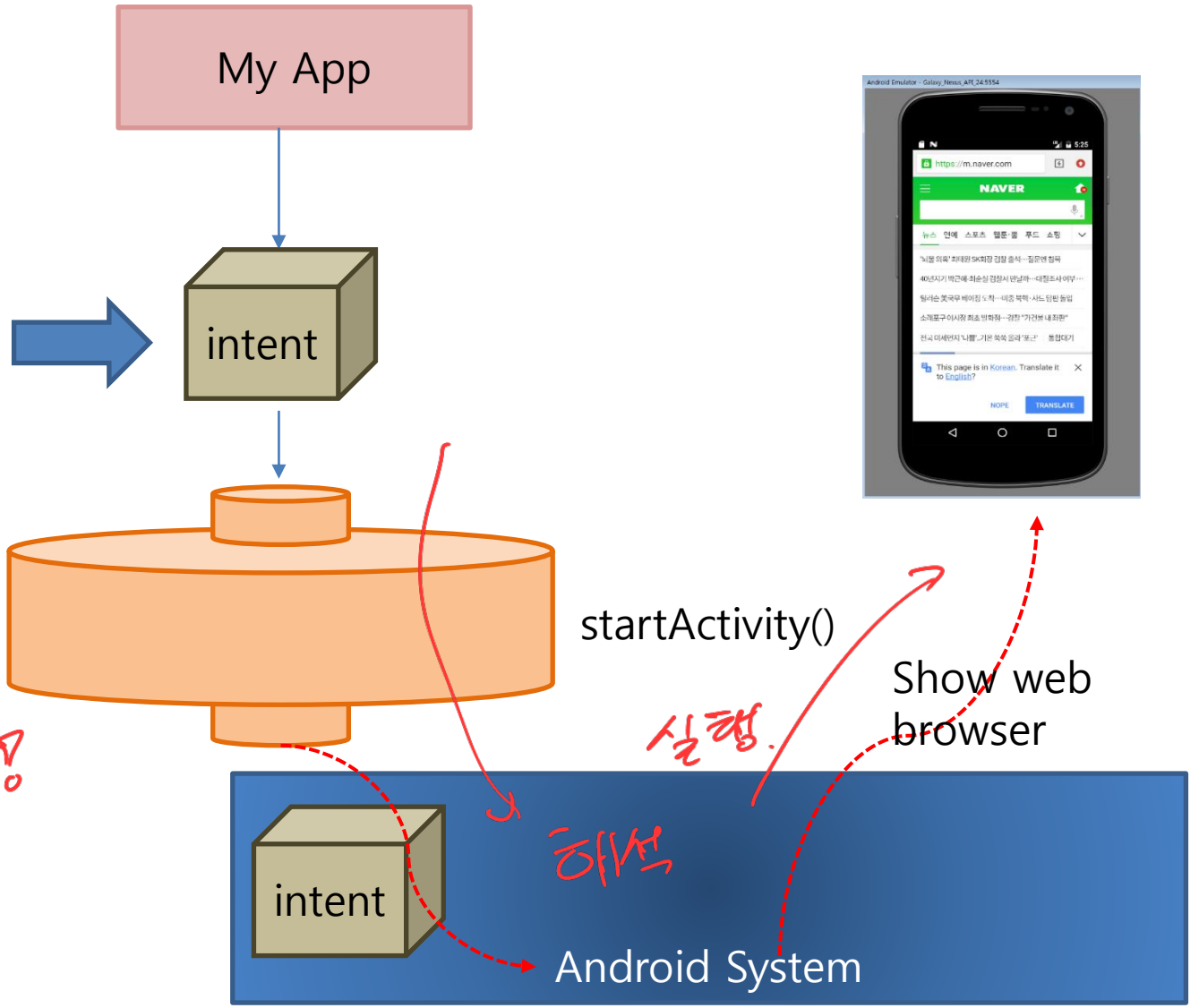
Primary data (as an URI)
tel://
http://
sendto://

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://m.naver.com"));
startActivity(intent);
```

Concept of Intents (Cont.)

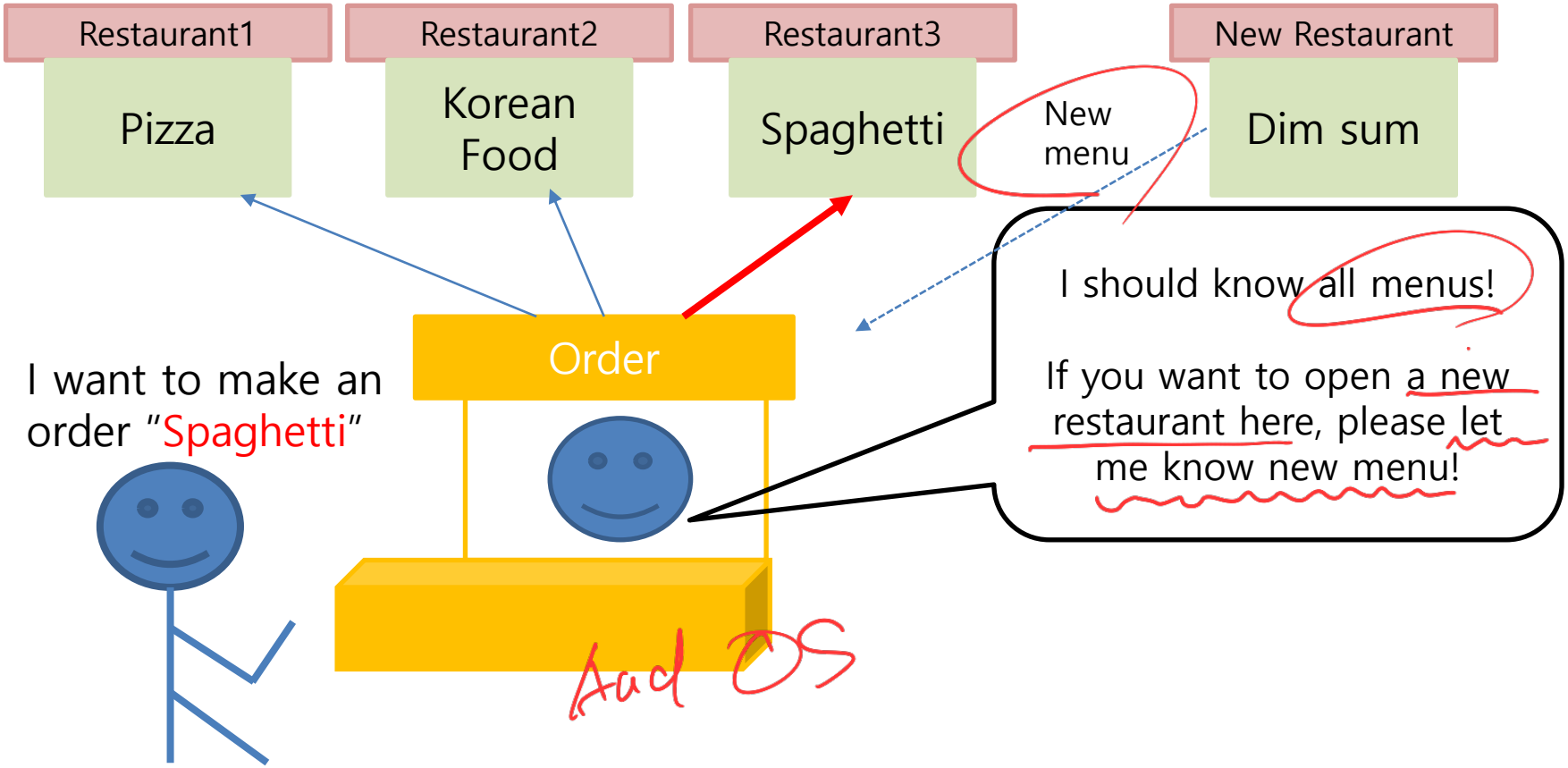
Put
- Action: `Intent.ACTION_VIEW`
- Data: `http://m.naver.com`

in intent

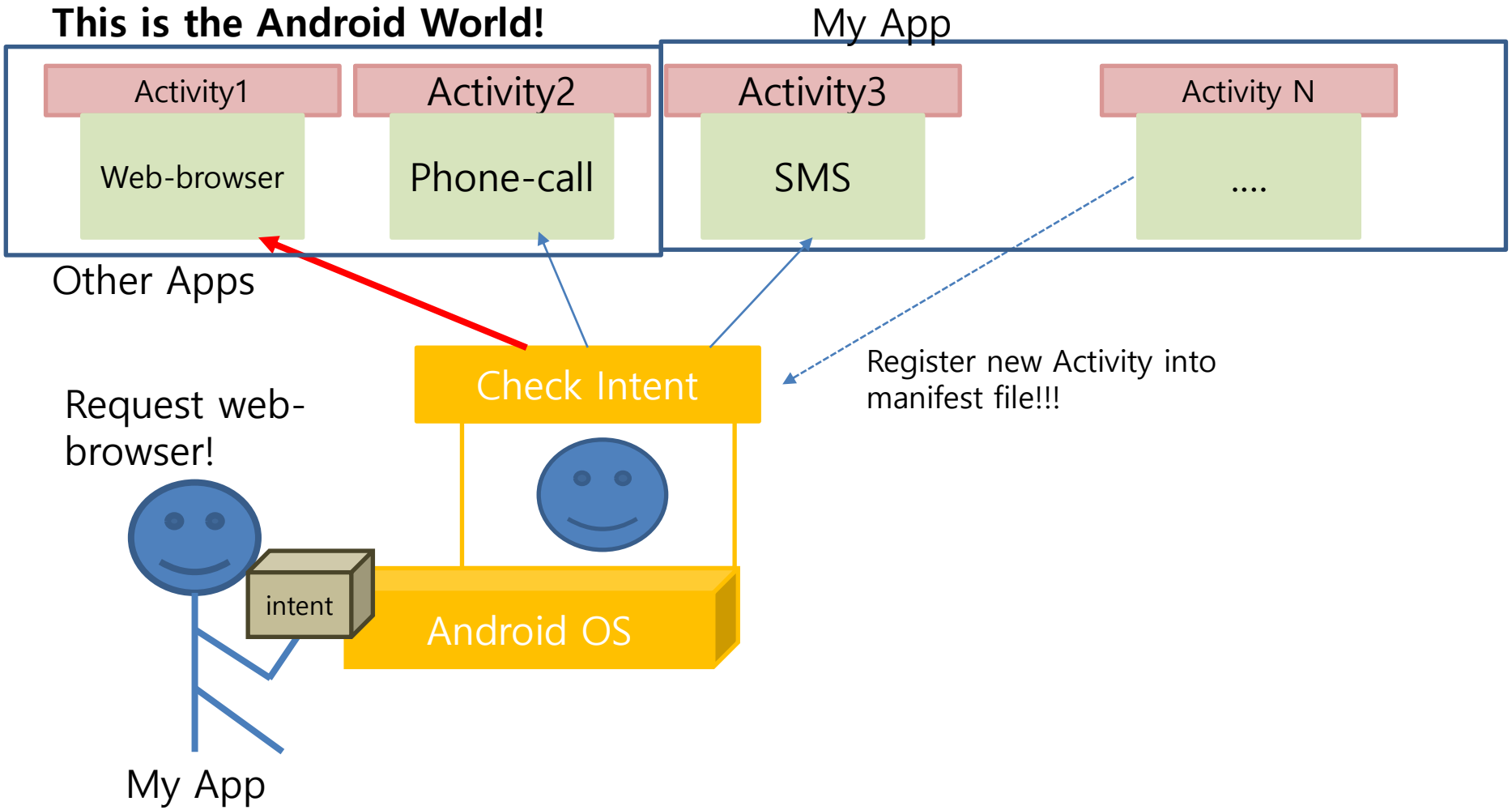


Concept of Intents (Cont.)

Let's imagine concept of intents in Real World



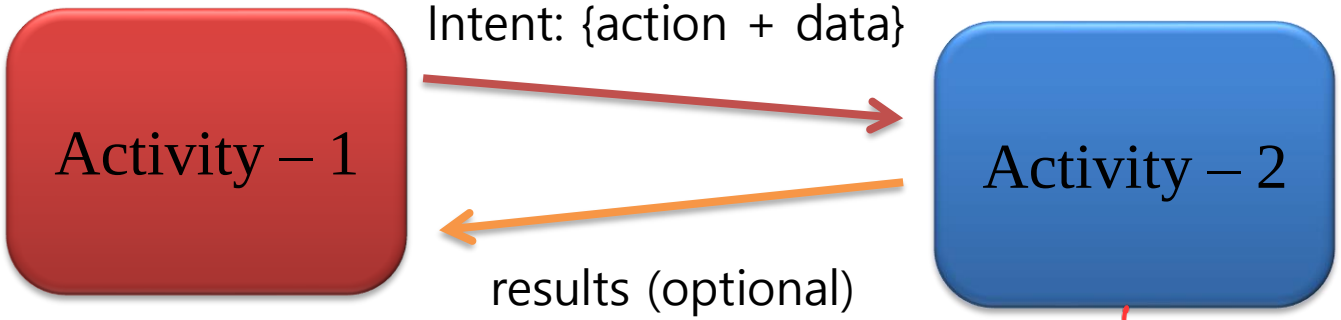
Concept of Intents (Cont.)



Concept of Intents (Cont.)

- The primary pieces of information (two main arguments) in an intent are:

1. **action** :
   The built-in action to be performed, such as **ACTION_VIEW, ACTION_EDIT, ACTION_MAIN**, … o r *user-created-activity*
2. **data** :
   The primary data to operate on, such as a phone number to be called (expressed as a **URI**).



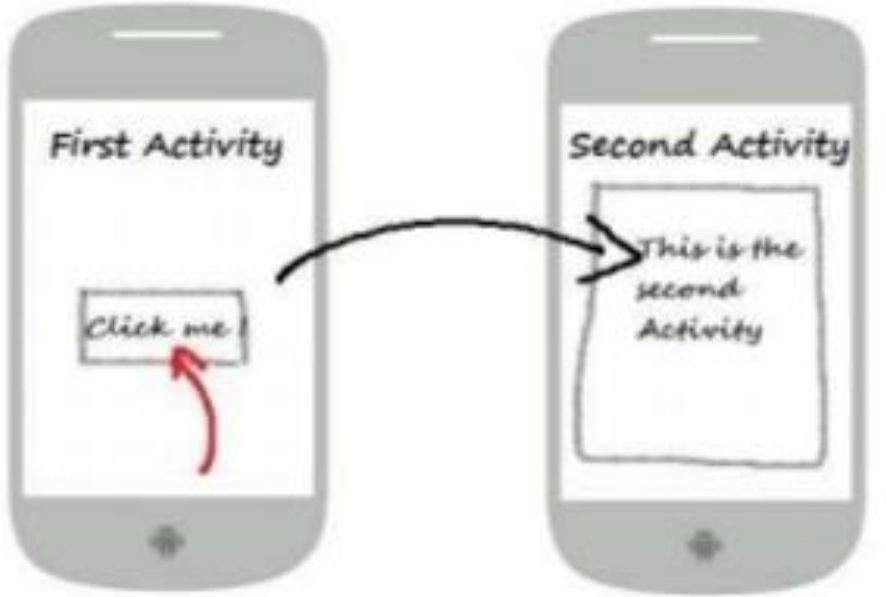
Intents

- Implicit Intent

암묵적

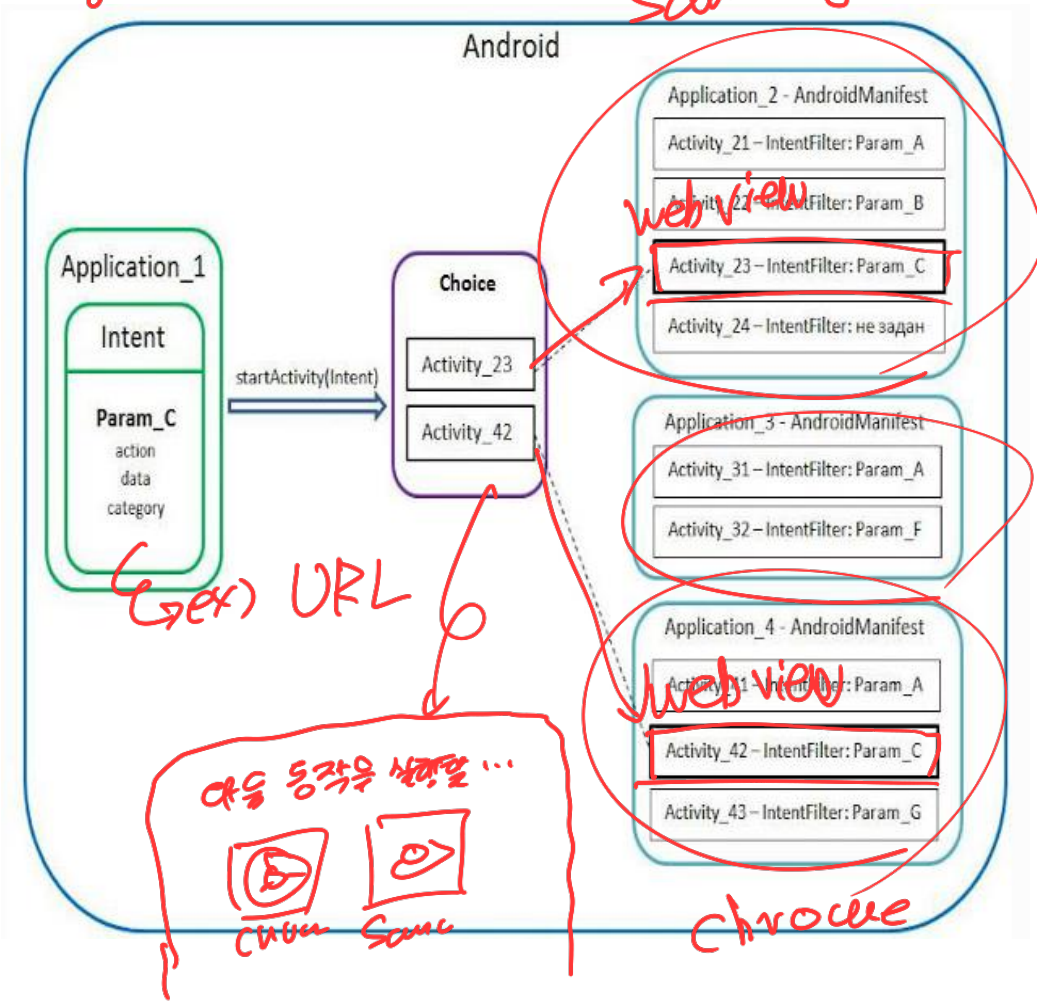
- Explicit Intent

명시적



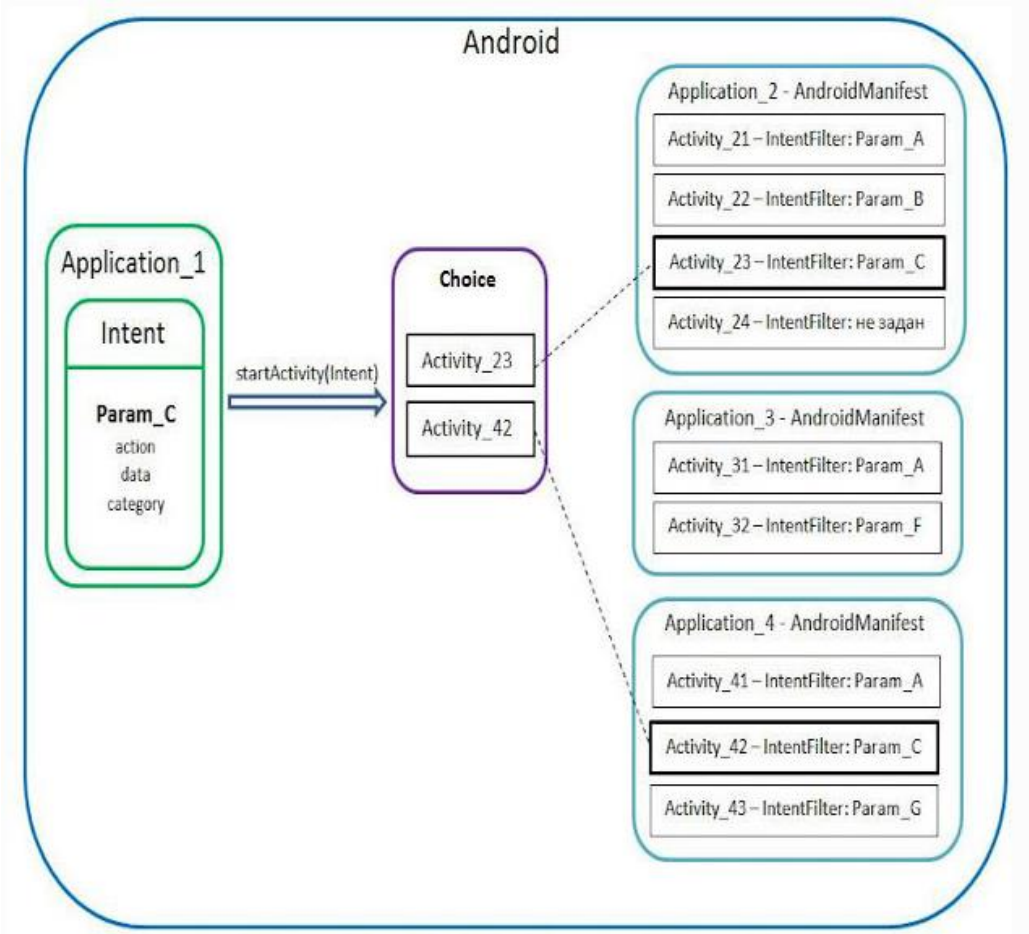
Implicit Intents

- Implicit Intent (Cont.)
  - These intents do not name a target and the field for the component name is left blank. Implicit intents are often used to activate components in other applications.
  - For example if we want to open a url in a browser than it will be implicit intent and we will not mentioned which activity will be responsible for this action
  - Implicit Intent by specifying a URI
    - Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(http://www.facebook.com));
    - startActivity(i);



Implicit Intents

- Implicit Intent
  - Here, we did not specify the activity that will be responsible for opening the url of facebook. So Android system will handle this action by performing mediation role.
    - Intent filter can be used for finding relevant activity by checking ACTION argument in intent filter.
  - If multiple activity is responsible to handle this action then a dialog will show to the user for the selection of appropriate activity

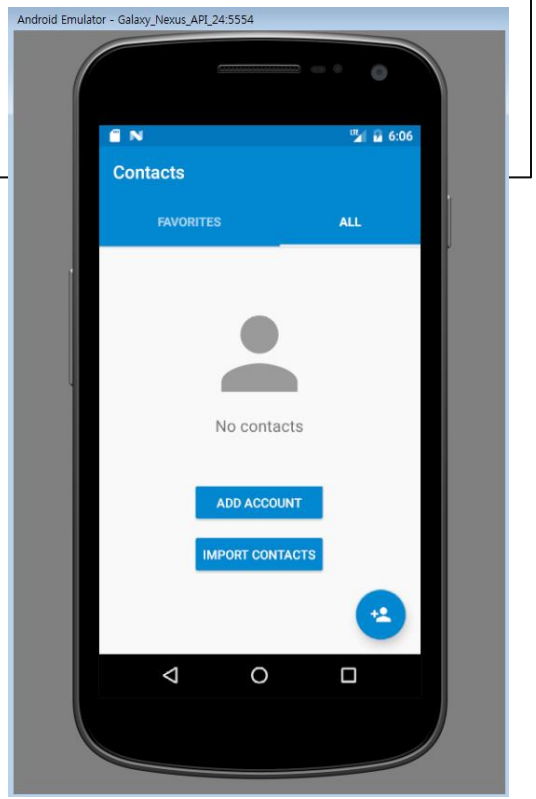


Example of Implicit Intent (1)

- Following fragments calls an **Intent** whose job is to invoke a built-in task (*ACTION_VIEW*) and explore the *Contacts* available in the phone.

```
Intent intent = new Intent(
     intent.ACTION_VIEW,
     Uri.parse("content://contacts/people"));
startActivity(intent);
```

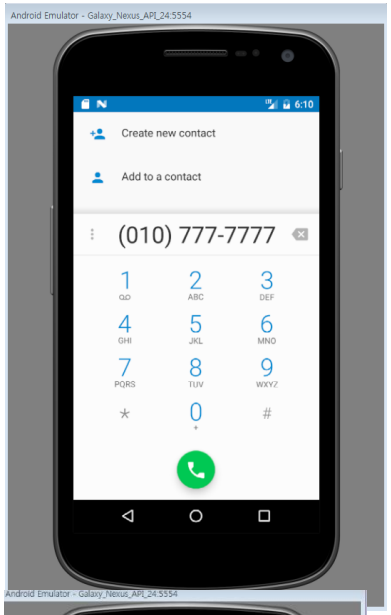
parse() method of the Uri class to convert a URL string into a Uri object.



Example of Implicit Intent (2)

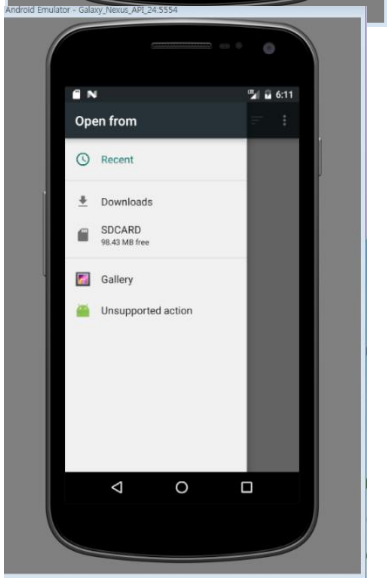
- Example :
  - Display the phone dialer with the given number filled in.
    - Show the Dial-pad

```
Intent intent =
new Intent (Intent.ACTION_DIAL,
                          Uri.parse( "tel:010-777-7777") );
startActivity(intent);
```



- Show picture gallery

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_GET_CONTENT);
intent.setType("image/pictures/*");
startActivity(intent);
```



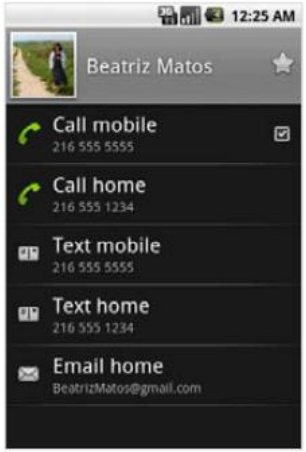
More Example

- Example : show all your contacts

```
String myData = "content://contacts/people/";
Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
                                Uri.parse(myData));
startActivity(myActivity2);
```

- Example : Show a particular contacts

```
String myData = "content://contacts/people/8";
Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
                                Uri.parse(myData));
startActivity(myActivity2);
```



- Example : Edit a particular contacts – /people/N

```
String myData = "content://contacts/people/8";
Intent myActivity2 = new Intent(Intent.ACTION_EDIT,
                                Uri.parse(myData));
startActivity(myActivity2);
```

Common Built-in Standard Actions

`Intent.ACTION_?`

List of standard actions that Intents can use for launching activities (usually through ***startActivity(Intent).***

ACTION_MAIN	ACTION_ANSWER
ACTION_VIEW	ACTION_INSERT
ACTION_ATTACH_DATA	ACTION_DELETE
ACTION_EDIT	ACTION_RUN
ACTION_PICK	ACTION_SYNC
ACTION_CHOOSER	ACTION_PICK_ACTIVITY
ACTION_GET_CONTENT	ACTION_SEARCH
ACTION_DIAL	ACTION WEB SEARCH
ACTION_CALL	ACTION_FACTORY_TEST
ACTION_SEND	
ACTION_SENDTO	

다양하게 준비되어 있다 ~

For a list of actions, see:
http://developer.android.com/reference/android/content/Intent.html

Secondary Attributes

- In addition to the primary *action/data* attributes, there are a number of *extra attributes (i.e., additional parameters)* that you can also include with an intent, such as:
  - 1. Category 2. Components 3. Type 4. Extras, ..

- Example: ACTION_SENDTO
  - Sending a text message (using extra attributes)

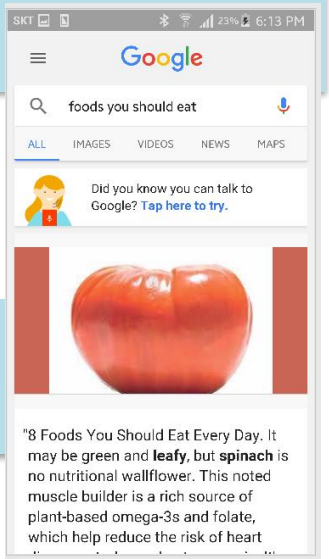
약속이 되어있음 !?

```
Intent intent = new Intent( Intent.ACTION_SENDTO,
                            Uri.parse("smsto:0105551234"));
intent.putExtra("sms_body","are we playing golf next Saturday?");
startActivity(intent);
```

- Web Search : **ACTION_WEB_SEARCH**
  - Passing a string as an ***Extra*** argument for a Google Search.

URI추가 안함

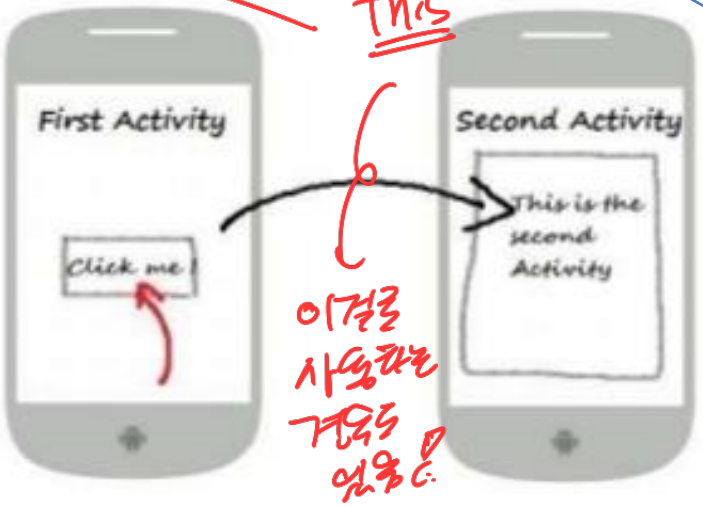
```
Intent intent = new Intent (Intent.ACTION_WEB_SEARCH );
intent.putExtra(SearchManager.QUERY, "foods you should eat");
startActivity(intent);
```



Explicit Intents

- Explicit Intent
  - These intents designate the target component by its name and they are typically used for application-internal messages-such as an activity starting a subordinate service or launching a sister activity.

e.g., `Intent intent = new Intent(getApplicationContext(), MenuActivity.class);`



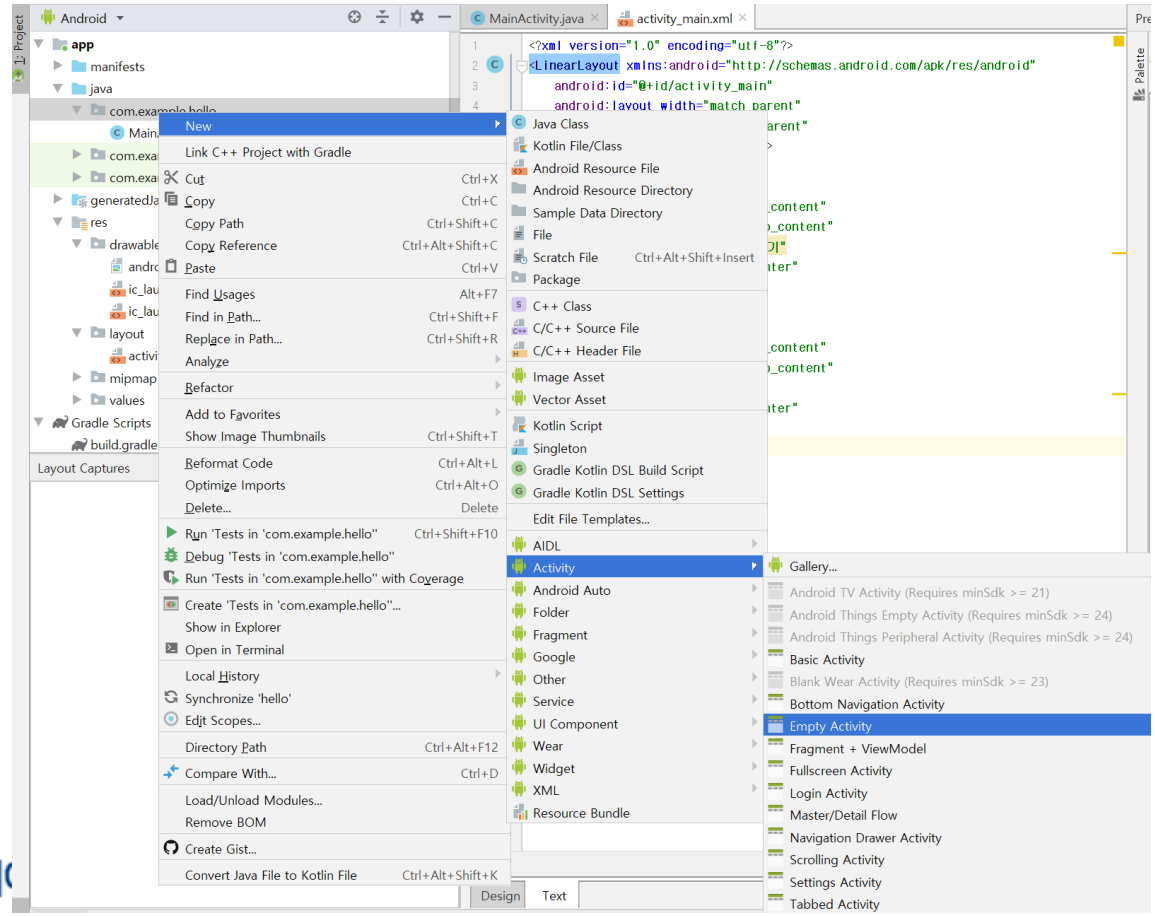
Context는 시스템 레벨의 기능 및 정보를 이용할 때 사용

Exercises

- Now Let's understand Explicit Intent which can be used for switching between Activities
  - Procedure for new Activity
    - Making a New Activity
    - Making a New Layout for New Activity
    - Register info of New Activity at Manifests
    - Call New activity

Exercises

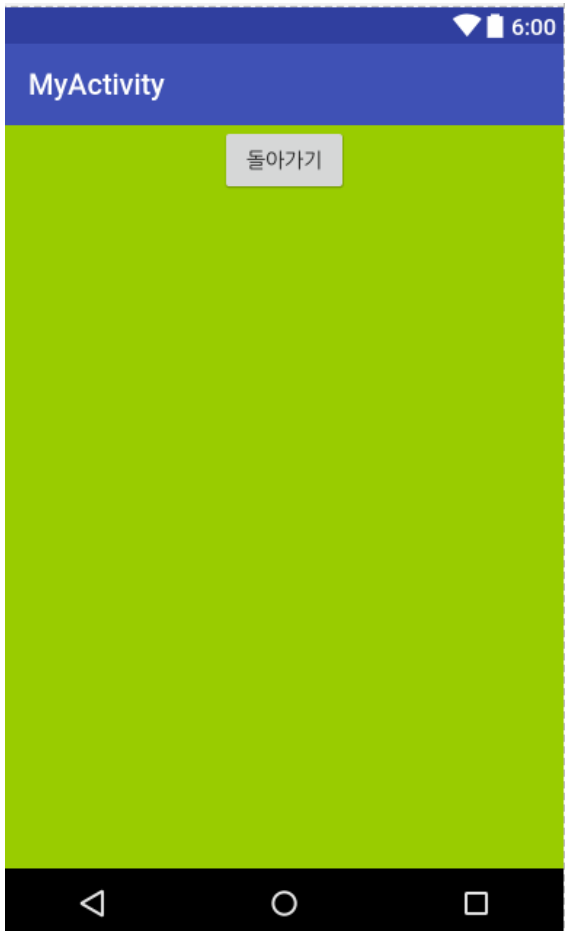
- Making a NewActivity → Now you have two activities (MainActivity & NewActivity)



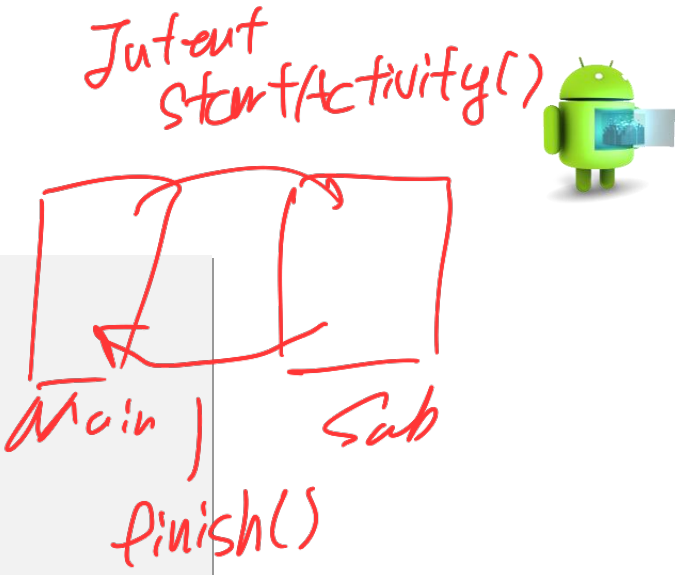
Exercises

- Design a layout for an activity_new.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_new"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@android:color/holo_green_light">
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="돌아가기"
        android:layout_gravity="center"
        />
</LinearLayout>
```

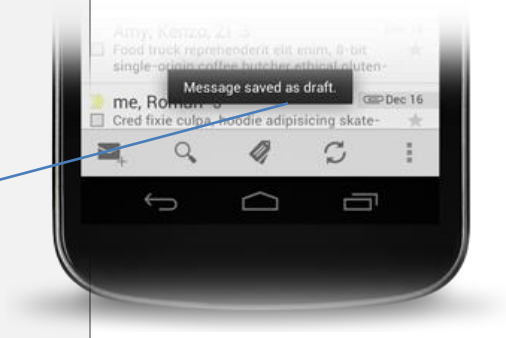


Exercises



- NewActivity.java

```
package com.example.hello;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class NewActivity extends AppCompatActivity {

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

        Button button = findViewById(R.id.button2);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(),
"돌아가기버튼이 눌렸어요", Toast.LENGTH_LONG).show();
                finish();
            }
        });
    }
}
```



Remove current activity

Exercises

- Manifests : Register info of New Activity at Manifests
  - Latest version may automatically add below info for new activity

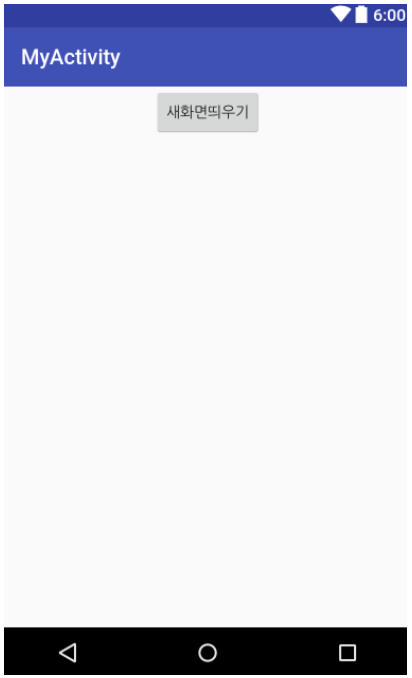
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.hello">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".NewActivity"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

And Studio 자동화됨?

Exercises

- Call New activity
  - activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="새 화면 띄우기"
        />

</LinearLayout>
```



Exercises

- Call New activity
  - MainActivity.java

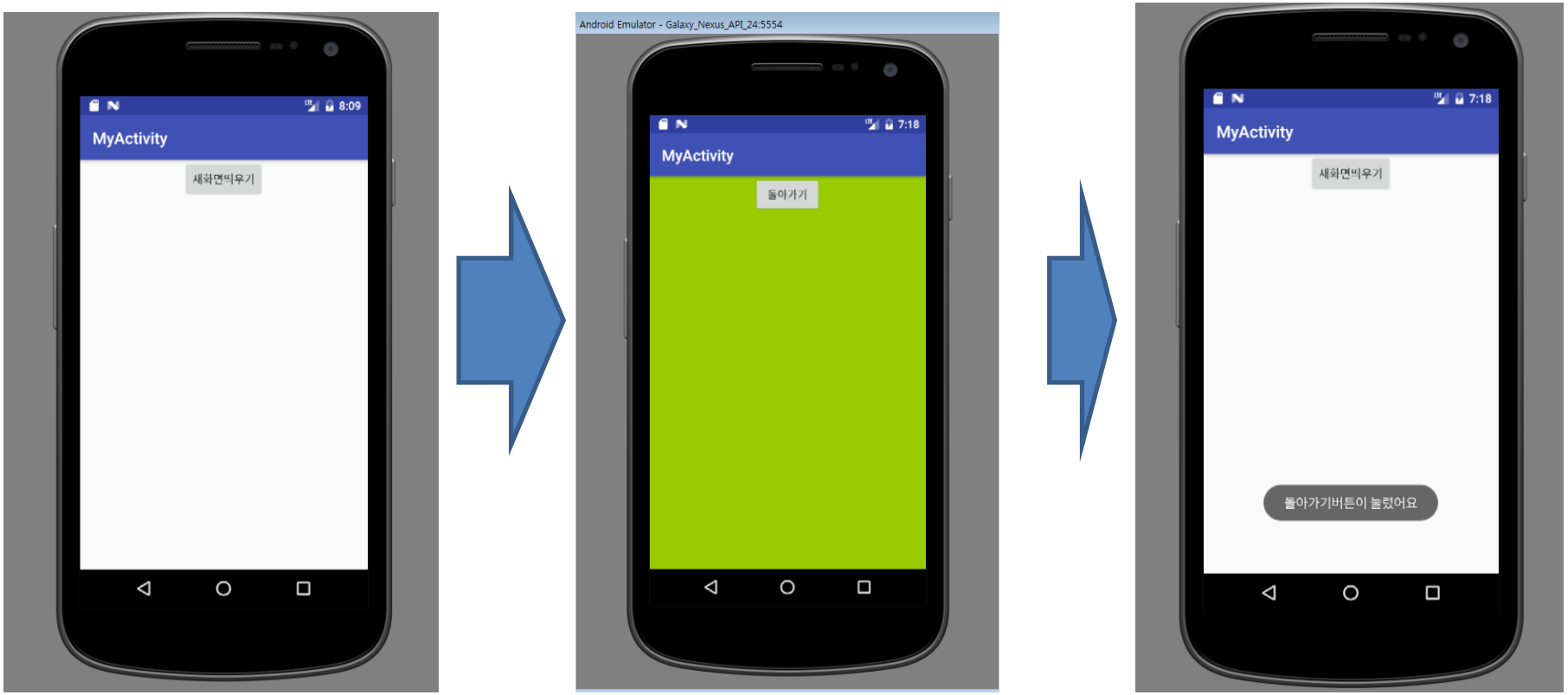
```java
package com.example.hello;
import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

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

        Button button1 = findViewById(R.id.button1);

        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), NewActivity.class);
                startActivity(intent);
            }
        });

    }
}
```

지금 연습 의미..?

Intent. ACTION.VIEW
(→ 미리 지정

Exercises

- Call New activity
  - RUN!!



Launching Activity by Class Name

- You can start activities in several ways.
- method: **`startActivity()`**

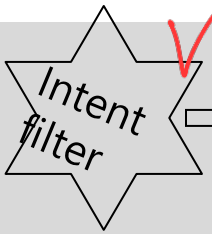
```
startActivity(new Intent("com.examp.ACTIVITY2"));
```

or

```
startActivity(new Intent(this, Activity2.class));
```

getApplicationContext()

- (if the activity `Activity2` is defined within the same project)



AndroidManifest.xml

```
<activity android:name=".Activity2"
        android:label="Activity 2">
    <intent-filter>
        <action android:name="com.example.usingintent.ACTIVITY2"/>
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

<intent-filter> element

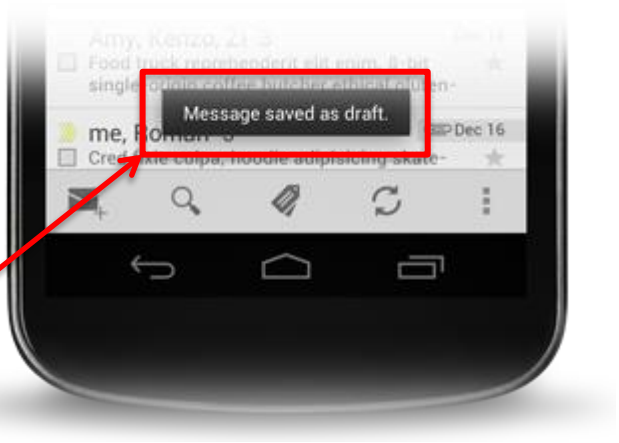
- <intent-filter> element defines how your activity can be invoked by another activity.
  - In your **AndroidManifest.xml** file

```
<activity android:name=".Activity2"
      android:label="Activity 2">
    <intent-filter>
      <action android:name="com.example.usingintent.ACTIVITY2"/>
      <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
  </activity>
```

Action to react

    - Via this name, other activities that wish to call this activity will invoke the activity
  - The category for the intent filter is android.intent.category.DEFAULT.
    - You need to add this to the intent filter so that this activity can be started by another activity using the startActivity() method

Toast (a small popup msg-dialog)



- A Toast
  - provides simple feedback about an operation in a small popup.
  - It only fills the amount of space required for the message and the current activity remains visible and interactive.
- c.f. for Android developers
  - It may be used for helping debug along with Log.d().

How to Use a Toast

- instantiate a Toast object with makeText() method.
  - **three parameters**:
    (the application Context, the text message, the duration for the toast)
    - `Toast.LENGTH_LONG  or Toast.LENGTH_SHORT or second`
  - returns: a properly initialized Toast object.
    - **You can display the toast notification with show()**

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;
Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

or (by chaining)

```
Toast.makeText(context, text, duration).show();
```

```
Or Simply,
•   Toast.makeText(this, text, Toast.LENGTH_SHORT).show();

•   Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT).show();
```

Exercise

- We can extend previous exercise in order to deliver additional information between activities using intent.

MainActivity.java → onClick method

```
@Override
public void onClick(View v) {
    Intent intent = new Intent(getApplicationContext(), NewActivity.class);
    startActivityForResult(intent,1);
}
```

Request result (using request code)

You can define request code which should be different from activities

NewActivity.java → onClick method

```
@Override
public void onClick(View v) {
    Toast.makeText(getApplicationContext(), "돌아가기버튼이 눌렸어요", Toast.LENGTH_LONG).show();

    Intent intent = new Intent();
    intent.putExtra("name", "mike");
    setResult(RESULT_OK, intent);
    finish();
}
```

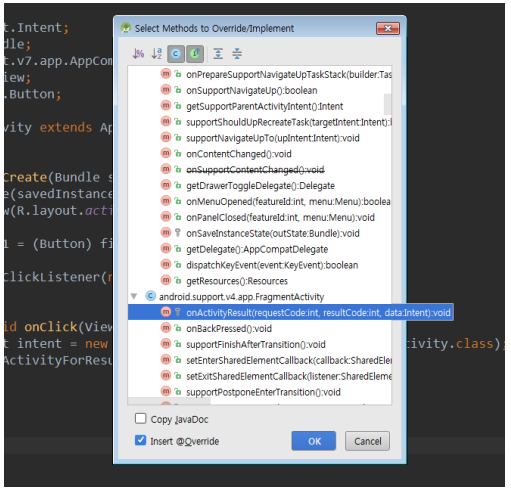
Response to request by using intent

Exercise (Cont.)

- Add method to extract result received from NewActivity through intent
  - **You can easily add Override Methods [Right button click →Generate→Override Methods]**

MainActivity.java

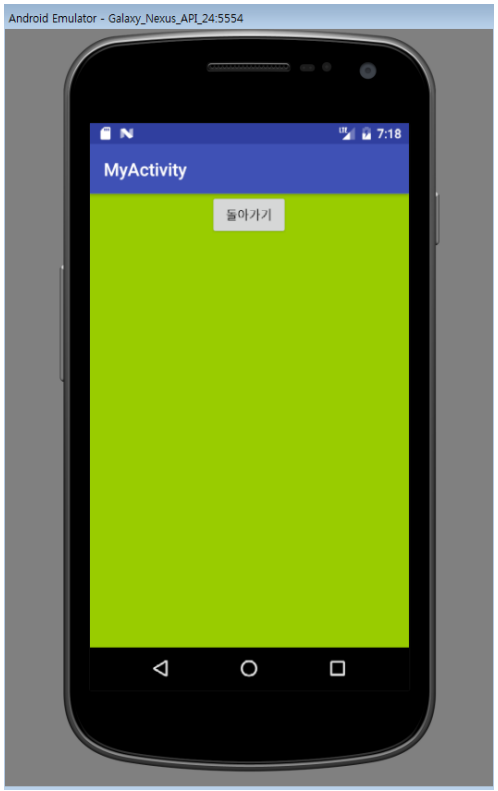
```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    String outName = data.getStringExtra("name");
    Toast.makeText(this, "전달받은 name 값:"+outName, Toast.LENGTH_LONG).show();
}
```

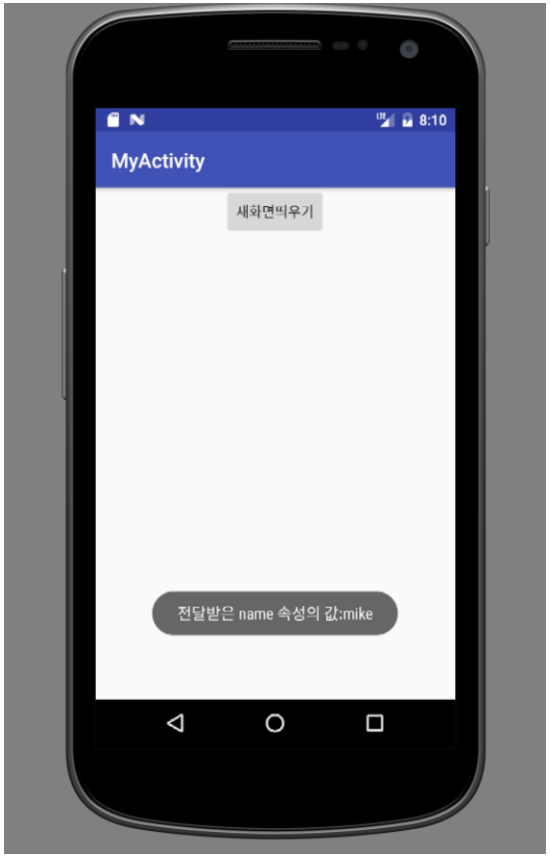


주의! onActivityResult는 onCreate method안에 있으면 안됨!

Exercise (Cont.)

- RUN!!





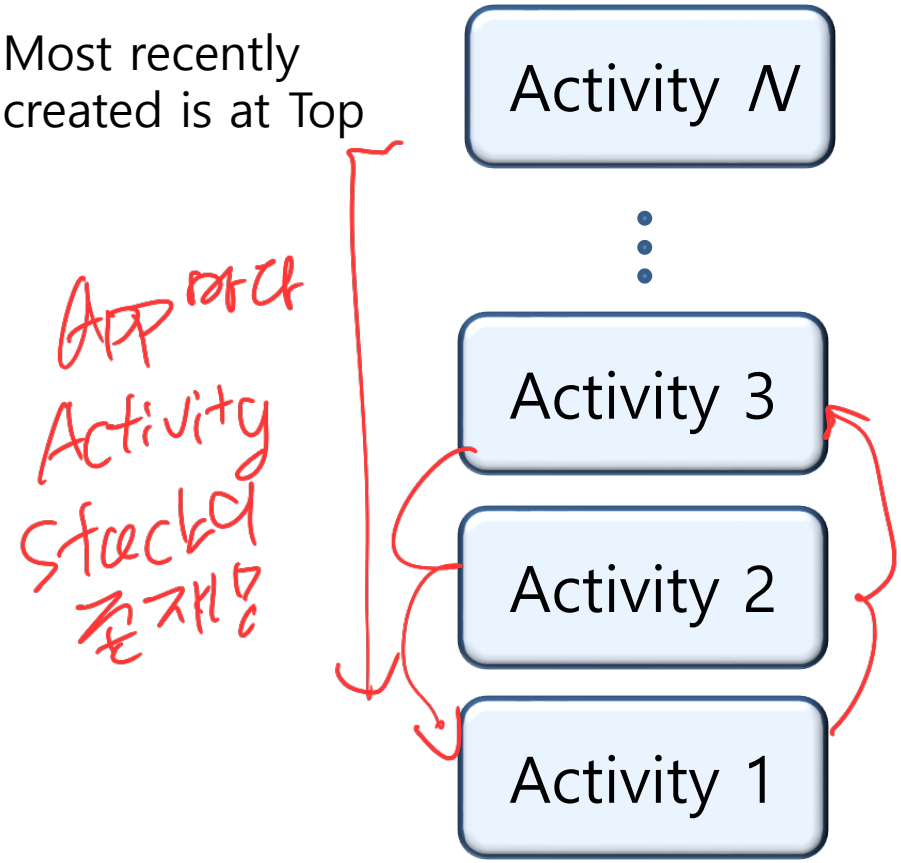
Activity Stack

- During the execution of an App,

Activities in the system are managed as an ***activity stack***.

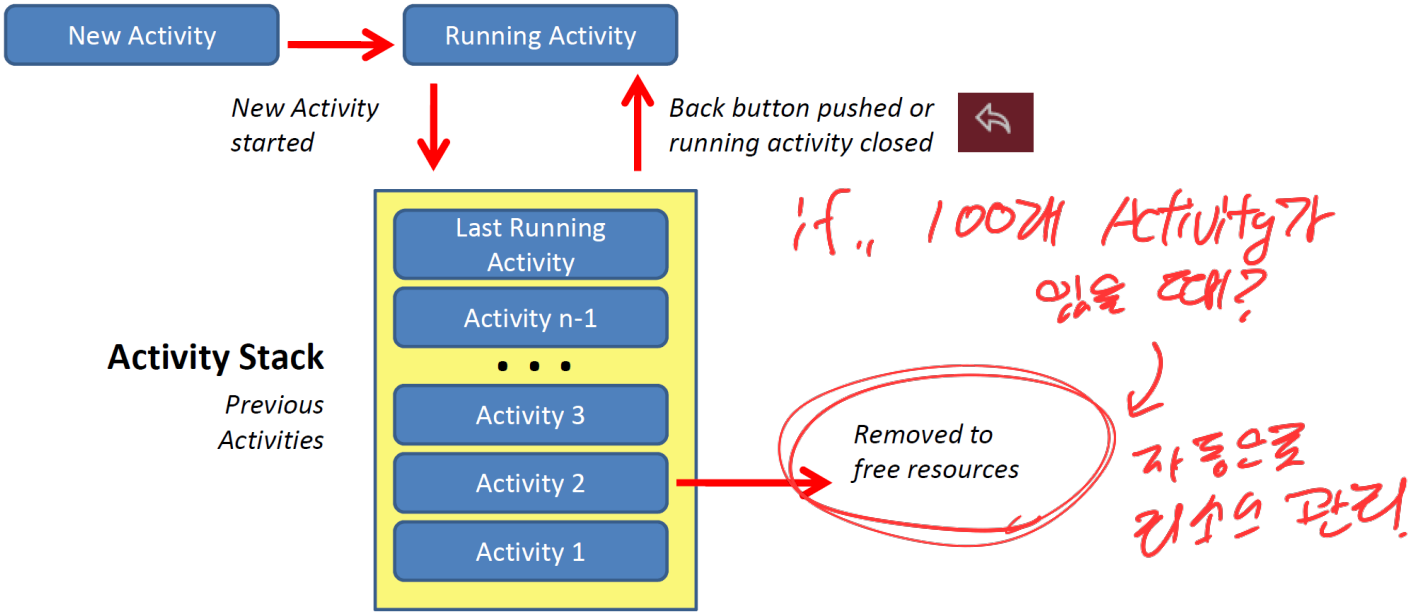
- When a new activity is *started*, it is placed on the *top* of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.
- If the user presses the *Back Button* the next activity on the stack moves up and becomes active.

Most recently created is at Top



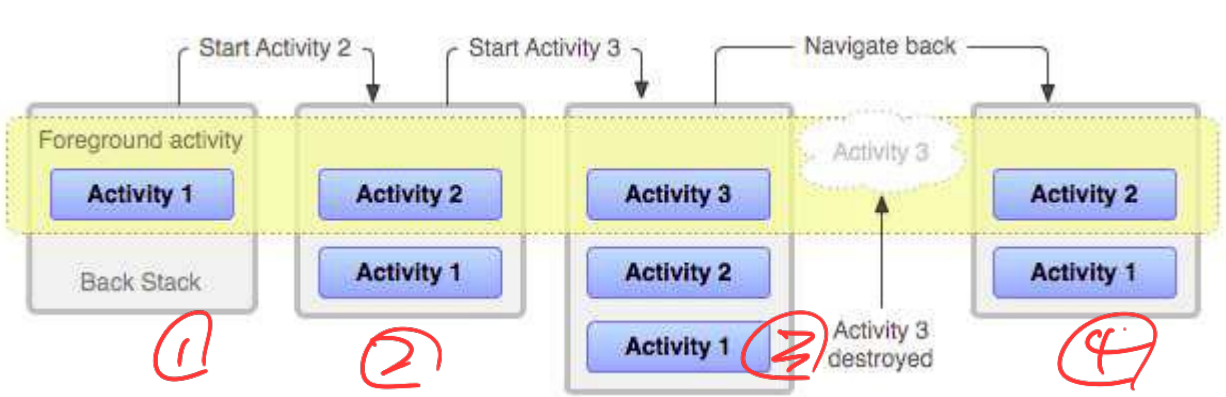
Activity Stack

- Conceptually laid out as a stack
  - The activity on top of the stack is visible/in foreground
  - Background activities are stopped but state is retained
  - Back button resumes previous Activity in the stack



Activity Stack (cont.)

- Conceptually laid out as a stack
  - The activity on top of the stack is visible/in foreground
  - Background activities are stopped but state is retained
  - Back button resumes previous Activity in the stack



  - HOME button moves app and its activities in background

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)