

Samsung Software GN Membership Android CodeLab

Teacher : 23-2 Jake Yoon, Michael Kim

Date : 2014. 04. 17

안녕하세요. 23-2 기 윤재석, 김기범 입니다. 오늘은 특별히 SelfStudy 형태로 문서를 제공해드리도록 하겠습니다. 오늘 다룰 주제는 Event Handling 과 ListView 를 배워보겠습니다. XML 을 통해서 화면 설계한 것을 Java 코딩을 통해 동적으로 변환시키는 것을 해보려고 합니다. 먼저 github 에서 최신버전의 소스를 받기 위해 git pull 을 입력해주세요.

```
D:\#Study>cd gnssm_android_codelab

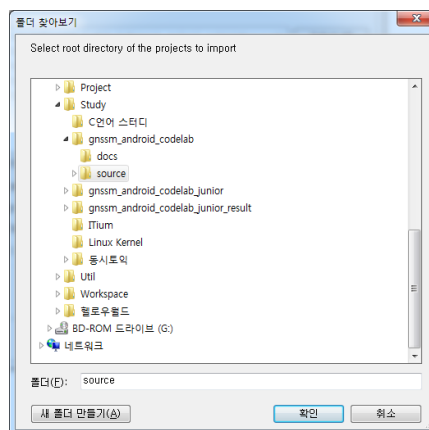
D:\#Study\gnssm_android_codelab>ls
README.md docs source

D:\#Study\gnssm_android_codelab>git pull
From https://github.com/yjaeseok/gnssm_android_codelab
1f47a3f..ae67c4b master -> origin/master
Already up-to-date.

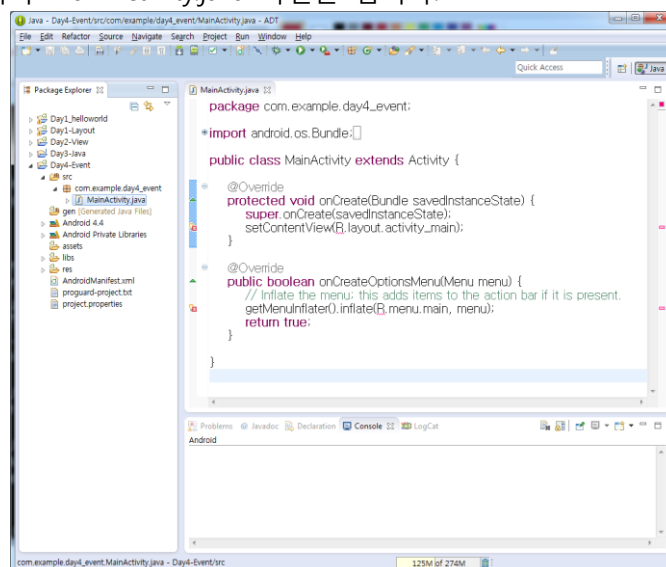
D:\#Study\gnssm_android_codelab>
```

Eclipse 를 실행합니다.

좌측에 있는 Package Explorer 에서 Day4-Event 프로젝트가 보이시나요? 보이지 않는다면 File > Import > Existing Projects into Workspace > Select root directory > Browser 를 눌러서 source 폴더를 선택해줍니다.



Package Explorer 에서 MainActivity.java 파일을 엽니다.



MainActivity 는 Activity 를 상속받은 클래스입니다. 즉, Activity 가 가지고 있는 특성인 여러 View 를 포함할 수 있습니다. 이전 수업자료를 토대로 다시 생각해보면, Activity 는 onCreate()라는 함수를 실행하면서 시작하고 onStart() -> onResume() -> onPause() -> onStop() -> onDestroy() 로 끝나는 Activity LifeCycle 이라는 개념을 가지고 있습니다.

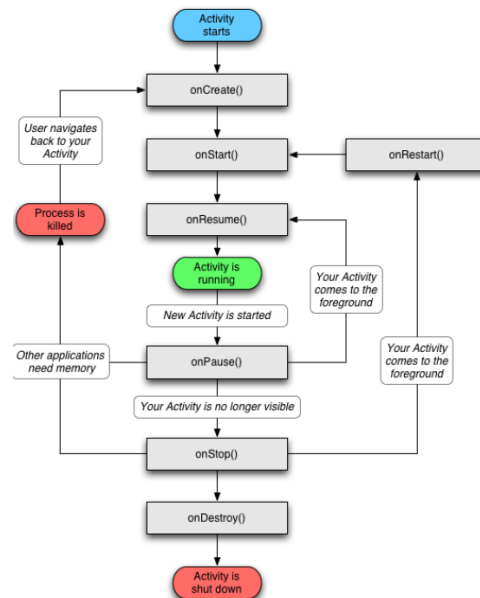
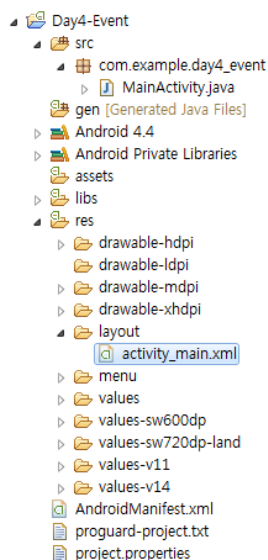


Figure 1 Activity LifeCycle

즉 Activity 가 생성되면 최초 실행되는 함수는 onCreate() 입니다. 현재 소스에는 onCreate()가 실행되면 setContentView(R.layout.activity_main) 함수를 실행하도록 코딩 되어있습니다. 여기서 말하는 setContentView() 라는 함수는 Activity 의 화면을 그릴 XML 파일을 지정해주고, 그 파일로 화면을 그릴 수 있도록 세팅해주는 것을 합니다. 여기서 말하는 R.layout.activity_main 이라는 것은 R.java 파일에 선언되어있는 변수를 의미하며, R.java 는 res 폴더아래에 존재하는 리소스나 객체를 참조하기 위한 메모리 주소를 저장하고 있습니다.

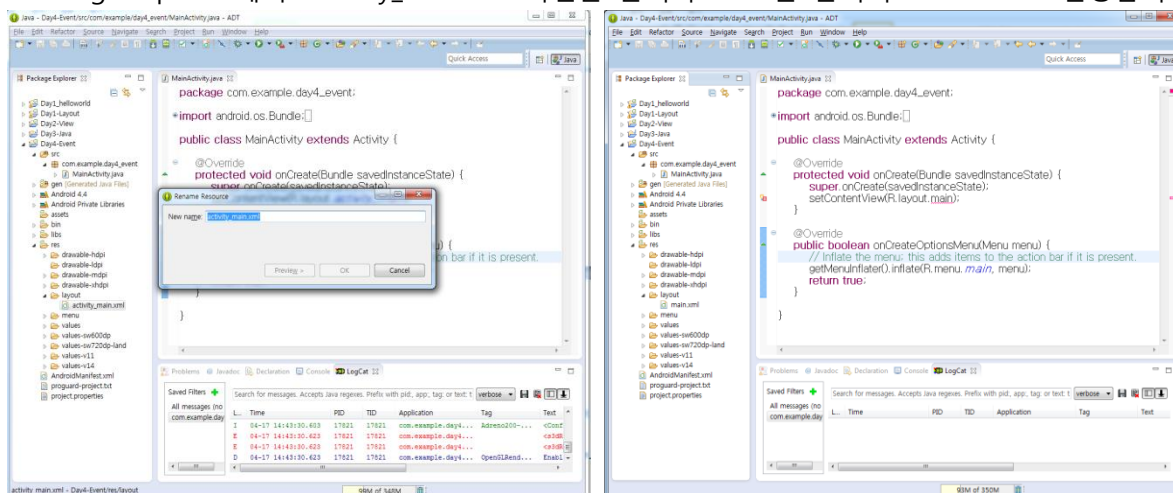


그래서 좌측에 보이는 것처럼 layout 폴더 아래에 activity_main.xml 이라는 파일이 하나 존재하는 것을 볼 수 있고, R.layout.activity_main 이라고 말하는 것은 res/layout/activity_main.xml 파일을 의미하는 용도로 쓰입니다.

즉 만약에 activity_main.xml 파일을 main.xml 이라고 수정한다면 R.layout.activity_main 을 R.layout.main 으로 수정해줘야합니다.

실제로 activity_main.xml 파일을 main.xml 로 이름을 변경해보겠습니다.

Package Explorer 에서 acitivity_main.xml 파일을 클릭하고 F2 를 눌러서 main.xml 로 변경합니다.



MainActivity.java 파일에 보면 setContentView()안에 R.layout.activity_main 이 R.layout.main 으로 자동으로 수정된 것을 확인할 수 있습니다. (ADT 가 업데이트되면서 이런 것도 해주네요 +_+)
즉 R.java 파일은 리소스와 연관된다는 것을 눈으로 확인할 수 있었습니다.

Event

이어서 Event 를 본격적으로 Event 를 배워보기 위해 Event 개념을 익혀보도록 하겠습니다.

Event 는 사용자가 버튼을 클릭하거나 스크린을 터치하거나, 리스트를 스크롤링 하는 등의 여러가지 행동을 했을 때, 그 행동을 감지해서 처리하는 유용한 방식을 말합니다. 안드로이드 프레임워크는 FIFO 방식의 Event Queue 처리 방식을 지향합니다. 즉 먼저 발생한 이벤트가 먼저 처리되는 형식으로 진행된다는 것입니다.

안드로이드에서는 이벤트를 관리하기 위해서 세 가지 컨셉을 제공합니다.

- Event Listeners : Event Listeners 는 특정 이벤트가 발생했을 때, 알림을 받는 객체를 말하며 주로 interface 로 작성되며, 구현된 클래스가 Event 를 받을 수 있게 됩니다.
- Event Listener Registration : Event Listener Registration 은 보통 setEventListener() 와 같은 형태로 제공되며, Event Listeners 를 등록시키기 위한 과정으로 쓰입니다.
- Event Handlers : Event Listener 내에 구현되어있는 실제 이벤트가 발생시에 처리해야 할 함수들을 말합니다.

Event Listeners 와 Handlers 의 예를 보도록 하겠습니다.

Event Handler	Event Listener & Description
onClick()	OnClickListener() OnClickListener()는 사용자가 버튼, 텍스트, 이미지 등을 클릭하거나 터치하거나 포커싱 하였을 때 호출됩니다.
onLongClick()	OnLongClickListener() OnLongClickListener()는 사용자가 버튼, 텍스트, 이미지 등을 클릭하거나 터치하거나 포커싱 하였을 때 호출됩니다. Click 과 다른 점은 길게 클릭되었을 때 호출된다는 것입니다.
onFocusChange()	OnFocusChangeListener() OnFocusChangeListener()는 View 가 Focus 를 얻거나 Focus 를 잃었을 때 호출됩니다.
onTouch()	OnTouchListener() 스크린 또는 View 를 터치하거나, 제스처를 행하였을 때 호출됩니다..

Event Listener Registration 에는 5 가지 방법이 존재합니다. 하지만 우리는 1 가지에 대해서만 배워보겠습니다.

이전 Java 특강 시간에 Interface 라는 것을 배우면서 Person 클래스에 Walkable 과 같은 Interface 를 구현했던 것을 기억하시나요? 혹시 기억이 가물가물하실 분들을 위해서 다시 이야기를 하면 Interface 란 어떠한 기능(함수)을 할 수 있는 클래스를 구분하기 위해서 사용됩니다. 예를 들어, Person 이라는 클래스가 Walk()라는 기능을 할 수 있다면 Walkable 이라는 Interface 를 구현해주면 됩니다.

예 1) Person 클래스와 Walkable 인터페이스

```

public class Person {
    private String name;
    private String sex;
    private int age;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}

public interface Walkable {
    public void Walk();
}

```

예 2) Walkable 인터페이스를 구현한 Person 클래스

(강의자료/src/Day3-Java/Person.java)

```

public class Person implements Walkable {
    private String name;
    private String sex;
    private int age;

    @Override
    public void Walk() {
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}

```

< 반드시 인터페이스의
함수를 구현해줘야함

위의 예에서 볼 수 있듯이 interface 를 구현한다는 이야기는 해당 interface 에 선언되어있는 함수를 해당 클래스가 반드시 구현(재정의)해줘야 한다는 이야기입니다. 이 말은 다시 말하면, 해당 클래스에는 interface 에 있는 함수가 구현(재정의)되어있다는 말입니다. 이 개념을 정확히 이해하셔야 Event 라는 것의 동작을 이해하시기에 편할 것입니다.

지금부터는 본격적으로 Android 어플리케이션에 interface 개념을 활용하여 Event 라는 것을 동작시켜보겠습니다. 먼저 res/layout/main.xml 파일을 열고 아래와 같이 입력합니다.

```

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

    <RadioGroup android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:gravity="center" android:id="@+id/radioGroup">

        <RadioButton android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/radio_short"
            android:text="짧게 표시" android:checked="true"/>

        <RadioButton android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/radio_long"
            android:text="길게 표시" />

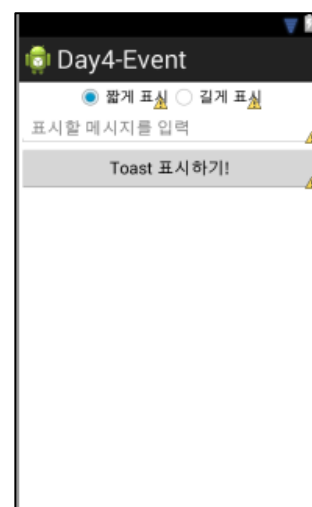
    </RadioGroup>

    <EditText android:layout_height="wrap_content"
        android:id="@+id/TextInput"
        android:layout_width="fill_parent"
        android:hint="표시할 메시지를 입력"/>

    <Button android:layout_height="wrap_content"
        android:id="@+id/ShowToast"
        android:text="Toast 표시하기!!"
        android:layout_width="fill_parent"
        android:layout_gravity="center"/>

</LinearLayout>

```

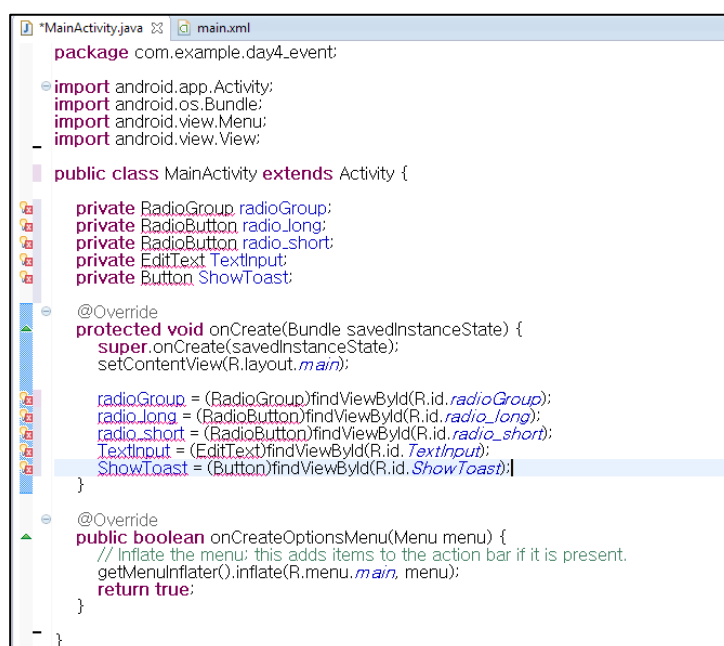


소스는 상당히 간단합니다. LinearLayout 으로 세로 방향으로 RadioGroup, EditText, Button 을 생성하였습니다. 각 객체에는 id 라는 속성을 사용하여 각각 radioGroup, radio_short, radio_long, TextInput, ShowToast 라고 id 를 할당해주었습니다. 이렇게 id 를 할당해주면 어플리케이션을 실행했을 때 R.java 파일에 id 밑에 해당 id 들이 생성되게 되며, MainActivity.java 와 같은 java 파일에서 XML 객체들을 접근할 수 있게 됩니다. (즉, 객체의 값을 얻거나 동적으로 변형할 수 있게 됩니다.)

main.xml 파일을 저장하고 안드로이드 어플리케이션을 실행해보도록 하겠습니다. 실행 후 gen/com.example.day4_event/R.java 파일을 열어보면 id 밑에 radioGroup, radio_short, radio_long, TextInput, ShowToast 가 추가된 것을 확인할 수 있습니다.

```
public static final class id {
    public static final int ShowToast=0x7f080004;
    public static final int TextInput=0x7f080003;
    public static final int action_settings=0x7f080005;
    public static final int radioGroup=0x7f080000;
    public static final int radio_long=0x7f080002;
    public static final int radio_short=0x7f080001;
}
```

즉, 이제 MainActivity.java 파일에서 해당 객체에 접근할 수 있는데 접근하기 위해서는 findViewById() 라는 함수를 사용해야합니다. (이 함수는 엄청 많이 써야하므로 귀찮아하실 꺼예요 ππ) findViewById() 함수를 활용하여 객체를 얻어보도록 하겠습니다. onCreate()함수 안에 아래와 같이 입력합니다.



```
*MainActivity.java  main.xml
package com.example.day4_event;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;

public class MainActivity extends Activity {

    private RadioGroup radioGroup;
    private RadioButton radioLong;
    private RadioButton radioShort;
    private EditText TextInput;
    private Button ShowToast;

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

        radioGroup = (RadioGroup)findViewById(R.id.radioGroup);
        radioLong = (RadioButton)findViewById(R.id.radio_long);
        radioShort = (RadioButton)findViewById(R.id.radio_short);
        TextInput = (EditText)findViewById(R.id.TextInput);
        ShowToast = (Button)findViewById(R.id.ShowToast);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

즉 radioGroup, radio_long, radio_short, TextInput, ShowToast 라는 이름으로 객체를 접근할 수 있습니다. (이름은 짓기 나름입니다.) 하지만 현재 해당 클래스에 대한 패키지가 import 되어있지 않아 빨간 줄이 그어져 있네요. Ctrl + Shift + O 를 눌러서 자동으로 해당 클래스가 포함된 패키지를 import 합니다.

```

*MainActivity.java  main.xml
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;

public class MainActivity extends Activity {

    private RadioGroup radioGroup;
    private RadioButton radio_long;
    private RadioButton radio_short;
    private EditText TextInput;
    private Button ShowToast;

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

        radioGroup = (RadioGroup)findViewById(R.id.radioGroup);
        radio_long = (RadioButton)findViewById(R.id.radio_long);
        radio_short = (RadioButton)findViewById(R.id.radio_short);
        TextInput = (EditText)findViewById(R.id.TextInput);
        ShowToast = (Button)findViewById(R.id.ShowToast);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}

```

TextInput.setText("I'm JakeYoon"); radio_long.setChecked(true); setBackgroundColor(Color.BLACK)과 같은 코드를 작성함을 통해서 객체들의 속성들을 변경시킬 수 있습니다.

지금부터는 앞에서 설명했던 EventListener 를 붙여보도록 하겠습니다. EventListener 를 붙이기 위해서 setOnClickListener() 함수를 호출해야 합니다. setOnClickListener()는 이름에 쓰여있는 것처럼 Click Event 가 발생했을 때 호출해주는 Listener 입니다. setOnClickListener() 함수의 인자로 this 를 넣어주었는데, this 란 MainActivity 클래스 자신을 의미합니다.

```

*MainActivity.java  main.xml
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;

public class MainActivity extends Activity {

    private RadioGroup radioGroup;
    private RadioButton radio_long;
    private RadioButton radio_short;
    private EditText TextInput;
    private Button ShowToast;

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

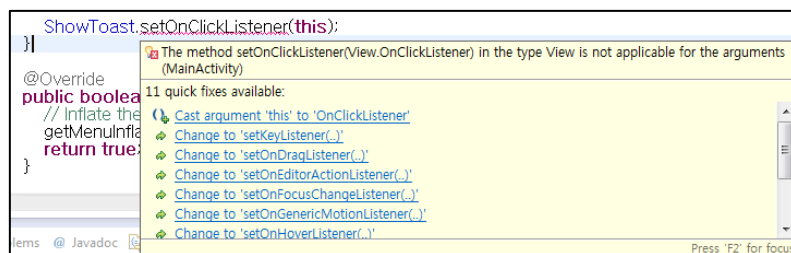
        radioGroup = (RadioGroup)findViewById(R.id.radioGroup);
        radio_long = (RadioButton)findViewById(R.id.radio_long);
        radio_short = (RadioButton)findViewById(R.id.radio_short);
        TextInput = (EditText)findViewById(R.id.TextInput);
        ShowToast = (Button)findViewById(R.id.ShowToast);

        ShowToast.setOnClickListener(this);
    }

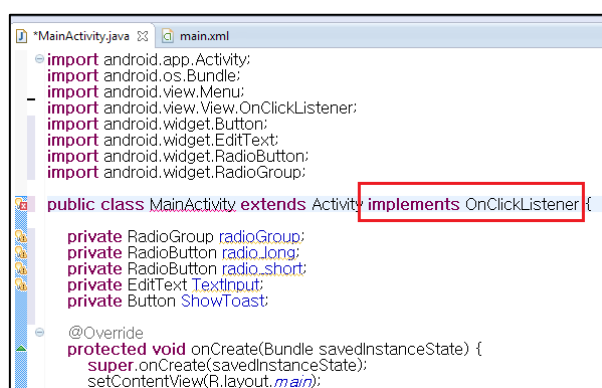
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}

```

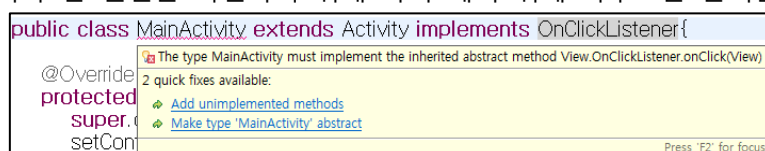
그러나 setOnClickListener()함수에 에러가 있다고 뜹니다. 에러부분에 마우스를 올려놓으면 아래와 같은 내용을 확인할 수 있습니다.



에러의 내용은 setOnClickListener()의 인자로 쓰인 MainActivity 가 적절하지 않다는 것입니다. setOnClickListener()의 인자로써 OnClickListener 인터페이스를 구현한 것만 들어올 수 있는데 MainActivity 는 OnClickListener 의 기능을 할 수 있지 않기 때문입니다. 따라서 위와 같이 코드를 작성하기 위해서는 MainActivity 가 OnClickListener 의 기능을 할 수 있도록 구현해주어야 합니다. 구현하기 위해서 아래와 같이 작성합니다.



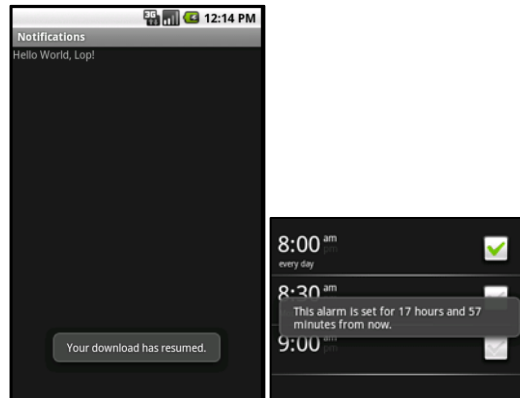
OnClickListener 인터페이스를 구현하였으므로 ShowToast.setOnClickListener(this)에는 에러가 사라진 것을 확인할 수 있습니다. 하지만, 이번에는 MainActivity 에 에러가 생긴 것을 확인할 수 있네요. (ㅠㅠ) 에러가 난 원인을 확인하기 위해 다시 에러 위에 마우스를 올려봅니다.



에러의 원인은 OnClickListener 를 구현했기 때문에 OnClickListener 인터페이스에 선언되어있는 함수를 구현해주어야 한다는 것입니다. 아래에 Add unimplemented methods 를 클릭합니다. 클릭하면 에러가 사라지면서 하단에 onClick() 이라는 함수가 생성된 것을 확인할 수 있습니다.



onClick() 함수는 버튼이 클릭되었을 때 호출되게 됩니다. 버튼이 클릭되었을 때를 감지하기 위해서 Toast 라는 것을 활용하도록 하겠습니다. Toast 는 어떤 이벤트가 발생하였을 때, 사용자에게 알려주기 위해서 사용하는 일종의 알림 창 입니다.



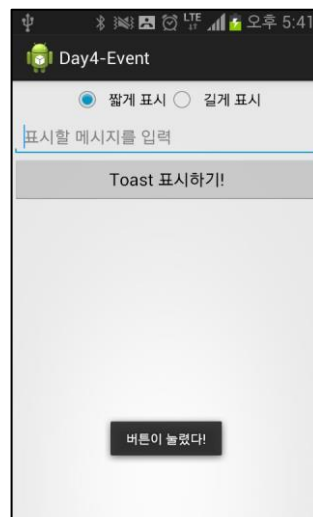
Toast 는 위 그림들과 같은 용도로 사용되며, UI Thread 에서 실행되어야 한다는 조건을 가지고 있습니다.

Toast 를 생성하고 표현하는 방법은 간단합니다. `Toast.makeText(Context context, CharSequence text, int duration).show();` 와 같은 형태로 사용합니다. `makeText()` 함수는 Toast 를 만드는 데 쓰이는 함수로 UI Thread 를 담당할 Context, 표현할 Text, 표현하는 시간 (`Toast.LENGTH_LONG`, `Toast.LENGTH_SHORT`) 3 가지를 인자로 갖습니다.

`onClick()` 함수 내에 아래와 같이 입력하고 어플리케이션을 실행합니다.

```
@Override
public void onClick(View v) {
    Toast.makeText(MainActivity.this, "버튼이 눌렀대", Toast.LENGTH_LONG).show();
}
```

버튼을 눌러보면 아래와 같이 알림이 뜨는 것을 확인할 수 있습니다.



그럼 이제 버튼을 눌렀으니 버튼에 뜨는 Text 를 동적으로, 눌린 Toast 의 노출 시간을 동적으로 변형해보도록 하겠습니다.

아래와 같이 코드를 작성합니다.

```
@Override
public void onClick(View v) {
    Toast.makeText(MainActivity.this,
        TextInput.getText().toString(),
        radio_long.isChecked()?Toast.LENGTH_LONG:Toast.LENGTH_SHORT);
}
```

첫 번째 인자는 MainActivity.this 로 이전과 동일합니다.

두 번째 인자는 TextInput.getText().toString() 으로 EditText 에 입력된 텍스트 값을 의미합니다.

세 번째 인자는 radio_long.isChecked()?Toast.LENGTH_LONG:Toast.LENGTH_SHORT 으로 삼항 연산자를 사용해서 radio_long 이 체크된 상태이면 Toast.LENGTH_LONG 을 체크되지 않은 상태이면 Toast.LENGTH_SHORT 을 갖도록 작성하였습니다.

실행 후 Text 를 변형해가면서 실행해보시기 바랍니다. ^-^;;

오늘 실습은 여기까지~ 참 쉽고 간단하죠? 이어서 오늘 숙제를 드리도록 하겠습니다. !!

오늘 숙제는 계산기 만들기입니다. 아래와 같은 UI 를 구성하시고, 덧셈, 뺄셈, 곱셈, 나눗셈 사칙연산을 할 수 있는 계산기를 만들어주세요. 기능은 윈도우에 있는 계산기를 참고하시면 될 것 같습니다. DELETE 를 누르면 입력된 버퍼의 내용을 비우도록 구현해야겠죠? 여러분의 실력을 믿습니다.

제출기한은 다음 주 화요일(22 일) 자정까지이며,

제출하는 곳은 수요일반은 윤재석 회원의 아웃룩으로, 목요일반은 김기범 회원의 아웃룩으로 제출하시면 됩니다. 수고 많이 하셨습니다.

