첫 번째 애플리케이션 (2/2)

Mobile Software
2019 Fall

What to do?

- **Android XML**
- 소스 코드 분석
  - package, import
  - 클래스 정의 및 상속
  - 메소드 재정의
  - setContentView 메소드
- MVC

Android XML : content_main.xml (1/2)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="com.google.android.material.appbar.Ap
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main"
    android:background="#ff2438">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

- 2개의 view를 정의
  - 계층 구조
  - **ConstraintLayout**
    - 외부 라이브러리에 정의
    - Android SDK에 아직 포함되지 않았음
  - **TextView**
    - Android SDK에 포함

- **Root element**
  - **ConstraintLayout**
  - Root element는 하나 뿐임
  - Root element에서만 name space 속성을 지정

Android XML : content_main.xml (2/2)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="com.google.android.material.appbar.Ap
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main"
    android:background="#ff2438">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

xmlns:**android**
➔ Android SDK에 정의된 tag 이름 및 속성을 사용
➔ **ns** : namespace
➔ xmlns:**xxx** : xxx 는 prefix

xmlns:**app**
➔외부 라이브러리에서 정의한 tag 이름 및 속성을 사용

xmlns:**tools**
➔ Android studio에서만 사용
➔ Preview 화면에서 XML로 설계한 UI 를 보여줄 때 사용

What to do?

- Android XML
- **소스 코드 분석**
  - **package, import**
  - **클래스 정의 및 상속**
  - **메소드 재정의**
  - **setContentView 메소드**
- UI 구현을 위한 코딩 스타일
- MVC

MainActivity.kt 소스 코드 (1/6)

```
package edu.incheon.firstapplication

import ...

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setSupportActionBar(toolbar)

        val str = getString(R.string.hello_msg)
        textView.text = str

        fab.setOnClickListener {...}
    }
```

- **package**
  - 클래스를 보관하는 container
  - 다른 사람 것과 중복되지 않도록 container 이름은 고유한 명칭을 지정
  - 주로 domain 이름을 패키지 이름 앞에 사용
  - **edu.incheon.프로젝트이름**

```
import android.os.Bundle
import com.google.android.material.snackbar.Snackbar
import androidx.appcompat.app.AppCompatActivity
import android.view.Menu
import android.view.MenuItem
import android.view.View

import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.content_main.*
```

- **import**
  - **패키지_이름.클래스_이름;**
  - 외부에서 정의한 클래스를 사용할 때 필요

MainActivity.kt 소스 코드 (2/6)

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setSupportActionBar(toolbar)

        val str = getString(R.string.hello_msg)
        textView.text = str

        fab.setOnClickListener {...}
    }

    fun convertCurrency(view: View) {...}
```

Compat → *compatible* (호환 가능한)

- **MainActivity** 클래스 : **파일 이름 = 클래스 이름**
  - **':' ➔ 클래스 상속 기호** (**자식 클래스 : 부모 클래스**)
    - 부모 클래스로부터 상속 받을 때 사용하는 기호.
- **AppCompatActivity**
  - 부모 클래스(**super class**)
  - 사용자 인터페이스(UI)를 갖는 **Activity** 컴포넌트
  - **AppCompatActivity ( ) 에서 괄호는 생성자(constructor) 호출**

MainActivity.kt 소스 코드 (3/6)

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)

    val str = getString(R.string.hello_msg)
    textView.text = str

    fab.setOnClickListener {...}
}
```

클래스에서 정의한 함수는 객체지향언어(OOP)에서는 method라 부름

- **override**
  - 부모 클래스(super class)로부터 상속받은 method를 자식 클래스에서 재정의
    - 부모 클래스 : **AppCompatActivity**
- **fun :** 함수 (function) 정의
- **변수 또는 파라미터 정의**
  - **파라미터 이름 ':' 데이터 타입** ➔savedInstanceState: Bundle?
  - **변수 이름 ':' 데이터 타입** ➔

```
val str: String = getString(R.string.hello_msg)
```

MainActivity.kt 소스 코드 (4/6)

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)

    val str = getString(R.string.hello_msg)
    textView.text = str

    fab.setOnClickListener {...}
}
```

이 method는 리턴 값이 없음.

- 메소드 이름이 **onXXX, OnXXX** 이면 **Event handler**
  - 이벤트가 발생했을 때 해당 이벤트를 처리하는 **callback** 메소드

- **onCreate**( ) 메소드는 언제, 누가 호출하는가?
  - Activity가 생성될 때 한 번 호출됨.Android system에서 호출.
    - 사용자 코드에서 이 메소드를 직접 호출할 수 없음.
  - **Activity 의 lifecycle**(생명 주기)과 관련 있음.

잠깐! Callback 메소드가 뭐예요?

- 모양 : **onXXX**
- 의미
  - Don't call us. We call you.
    - "결과가 나오면, 문자로 알려줄게."
    - 코드를 내가 짰지만, 이 method를 내가 호출할 수 없어!
  - 이 method가 언제 호출될지 어떻게 알지?
    - 걱정 말고 **onCreate**만 정확히 구현해!
      - **Android System**이 정확히 **onCreate** 를 호출해 줄 꺼야!
- 특징
  - There is no need for polling.
    - You will get the results as soon as they are available.

- 나만의 Callback 메소드 만들고 싶어?
  - Interface를 사용하면 돼!

MainActivity.kt 소스 코드 (5/6)

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)

    val str = getString(R.string.hello_msg)
    textView.text = str

    fab.setOnClickListener {...}
}
```

super
➔ **슈퍼 클래스**(super class)
➔ **AppCompatActivity**

- *dot-notation*
  - 객체(object)의 method나 property(또는 field)를 호출하는 문장 구조
    Object-이름 **.** **메소드_이름** (…);
    Object-이름 **.** **멤버_필드_이름**;
  - 상속받은 슈퍼 클래스의 method를 호출할 경우
    super . 메소드_이름 (…);

- 위 문장은 슈퍼 클래스인 **AppCompatActivity** 클래스의 **onCreate** 메소드를 호출

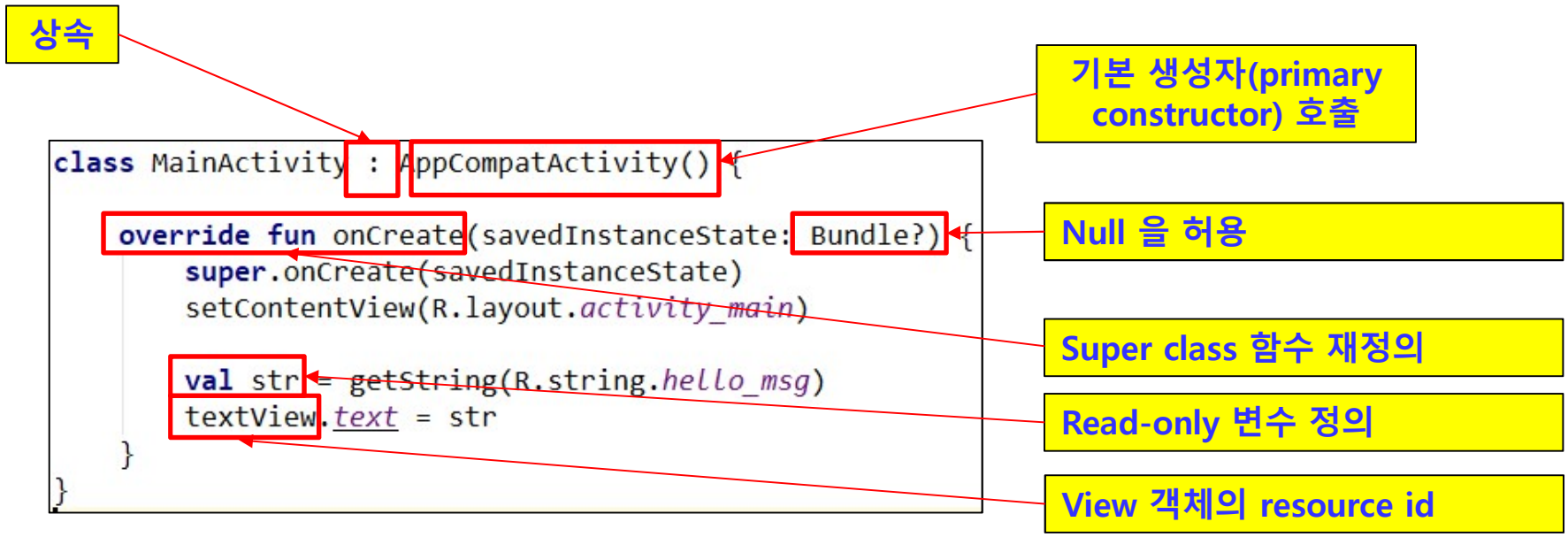
MainActivity.kt 소스 코드 (6/6)

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)

    val str = getString(R.string.hello_msg)
    textView.text = str

    fab.setOnClickListener {...}
}
```

- **setContentView**
  - UI 화면을 출력.
  - UI를 정의한 layout XML 파일 내용에 맞게 출력.
    - 이 메소드를 호출하지 않으면 화면에 아무 것도 나타나지 않음.
- **R. layout. activity_main**
  - **activity_main**은 UI 화면 내용을 정의한 XML 파일의 이름
  - 이 XML 파일에 해당하는 resource-id(**정수형 상수**) : **R.java**에 정의됨.

```
public static final int activity_main=0x7f0b001c;
```

MainActivity.kt 소스 코드 : 요약



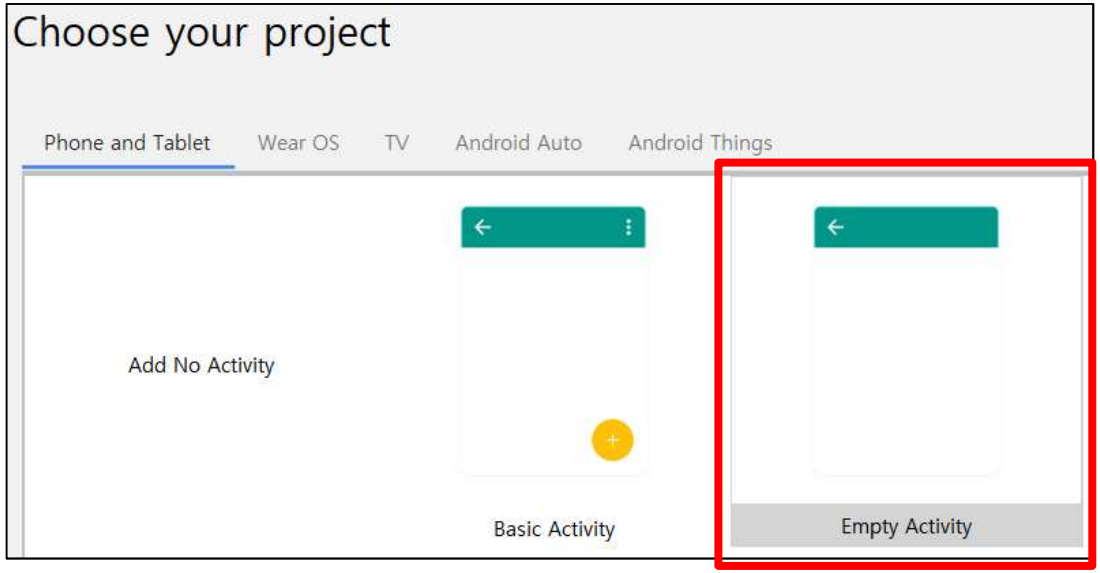
Android App.은 어디에서부터 실행이 시작될까?

- **Android App.은 main() 메소드가 없다.**
  - Android 컴포넌트 중 UI를 갖는 Activity부터 먼저 실행
    - activity가 여러 개 있으면
      - 가장 먼저 실행할 activity를 Manifest 파일에서 지정
  - Activity 클래스에서는
    - **onCreate**( ) 메소드가 가장 먼저 실행됨.

Practice 2a: Do and see it

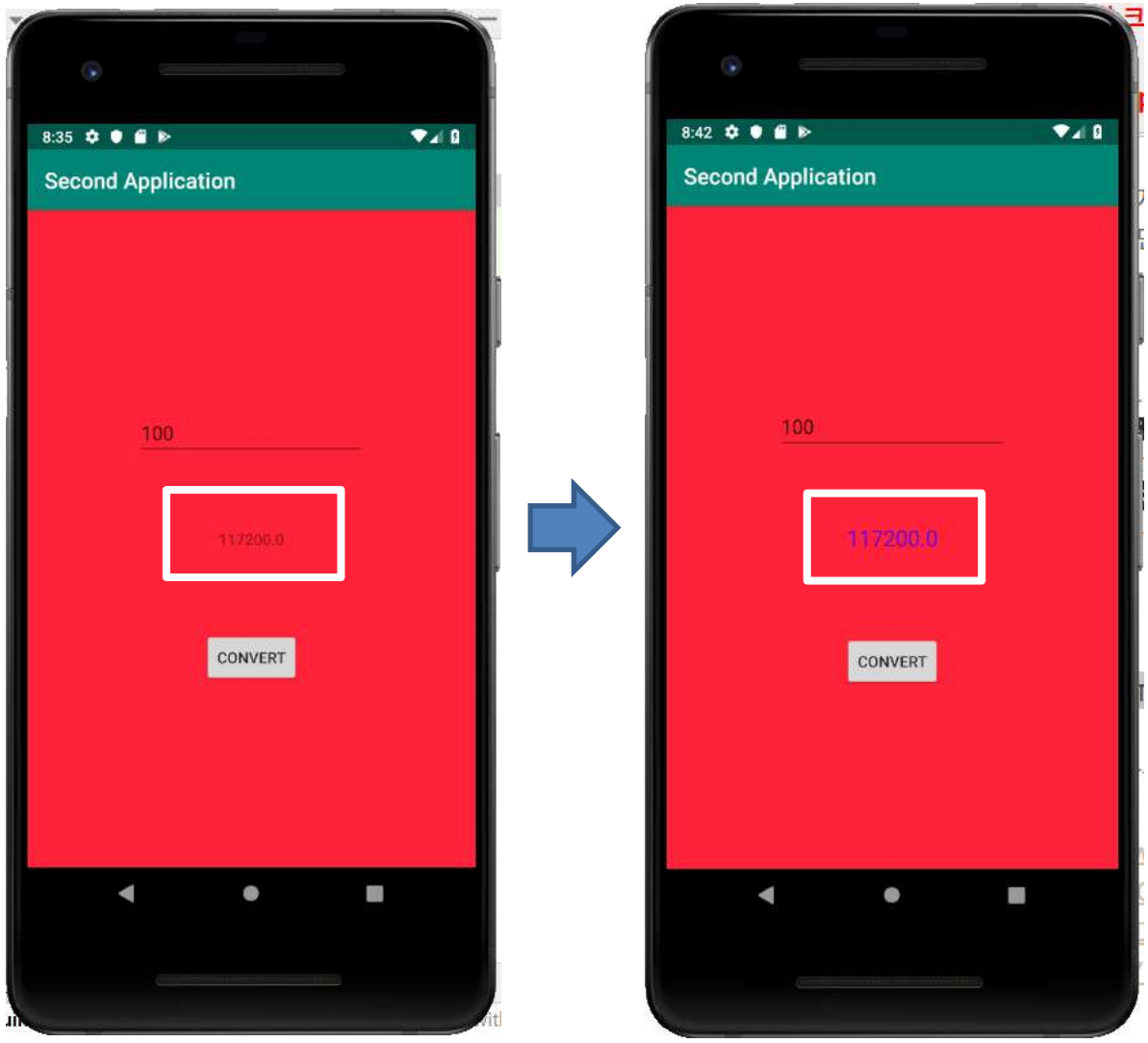
- **Basic Activity**를 사용하여 First Application 프로젝트를 만들었다.

- 이번엔 **Empty Activity**를 사용하여 Second Application 프로젝트를 만들어보자.
  - 화면 구성은 같음
    - 위젯은 TextView, Button, EditText를 사용
  - 환율 변환 기능도 같음



Practice 2b: Do and see it

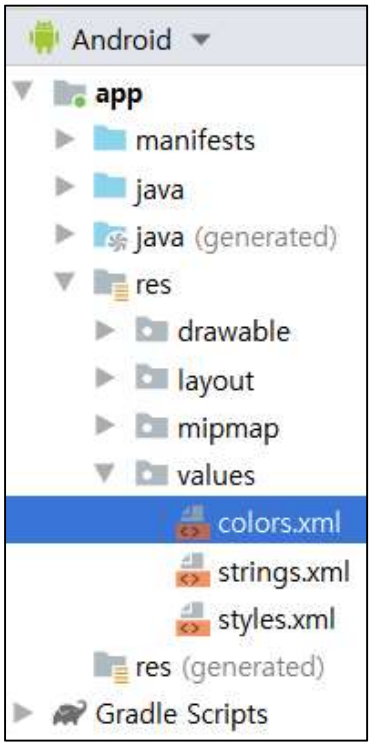
- 배경 화면이 빨간색이라 환율 금액을 알아보기 힘들다.
- 텍스트 속성을 바꿔보자
  - 글자 색상
  - 글자 크기



Practice 2c: TextView 객체 속성 설정

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {...}

    fun convertCurrency(view: View) {
        if (dollarText.text.isNotEmpty()) {
            val dollarValue = dollarText.text.toString().toFloat()
            val wonValue = dollarValue * 1172f
            textView.setTextColor(Color.parseColor("#6200ea"))
            textView.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20f)
            textView.text = wonValue.toString()


        } else {
            textView.text = "No Value"
        }
    }
}
```

res 폴더에 TextView 객체 속성 값을 저장하고 이를 코드에서 참조하자! (1/2)



activity_main.xml | MainActivity.kt | colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
    <color name="colorTextView">#6200ea</color>
</resources>
```

```
textView.setTextColor(Color.parseColor("#6200ea"))
```

↓

```
val color = resources.getColor(R.color.colorTextView)
```

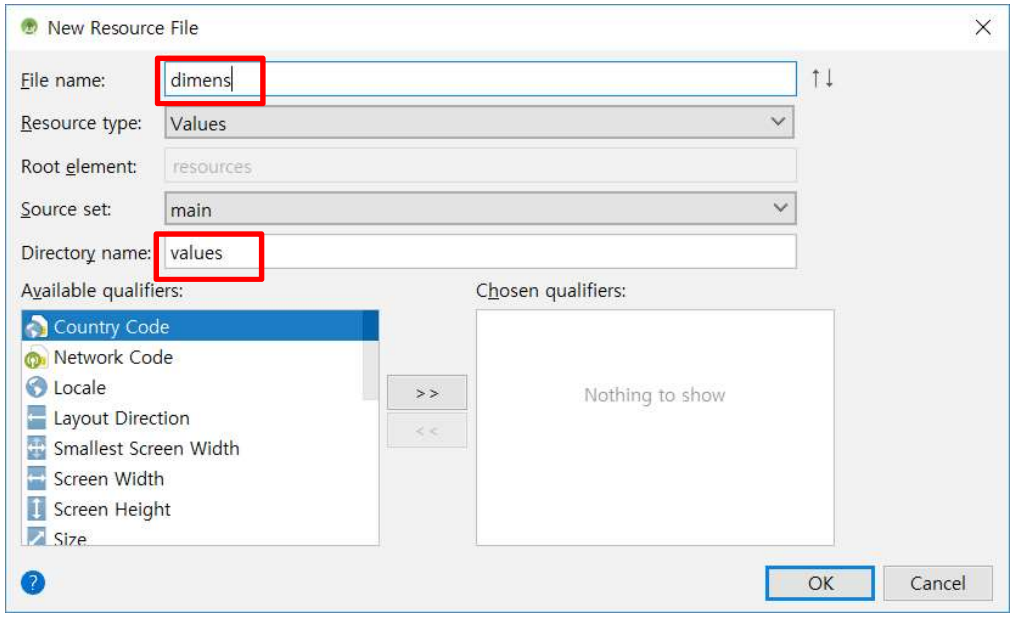
deprecated

↓

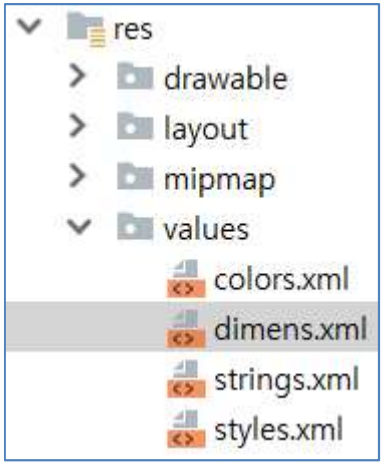
```
val color = ContextCompat.getColor(this, R.color.colorTextView)
textView.setTextColor(color)
```

values 폴더에 dimens.xml 추가

res
> 마우스 오른쪽 버튼
> New Resource File



res/dimens.xml



res 폴더에 TextView 객체 속성 값을 저장하고 이를 코드에서 참조하자! (2/2)

2가지 방식으로 구현 가능. 차이점은 return 값의 type이 다름.

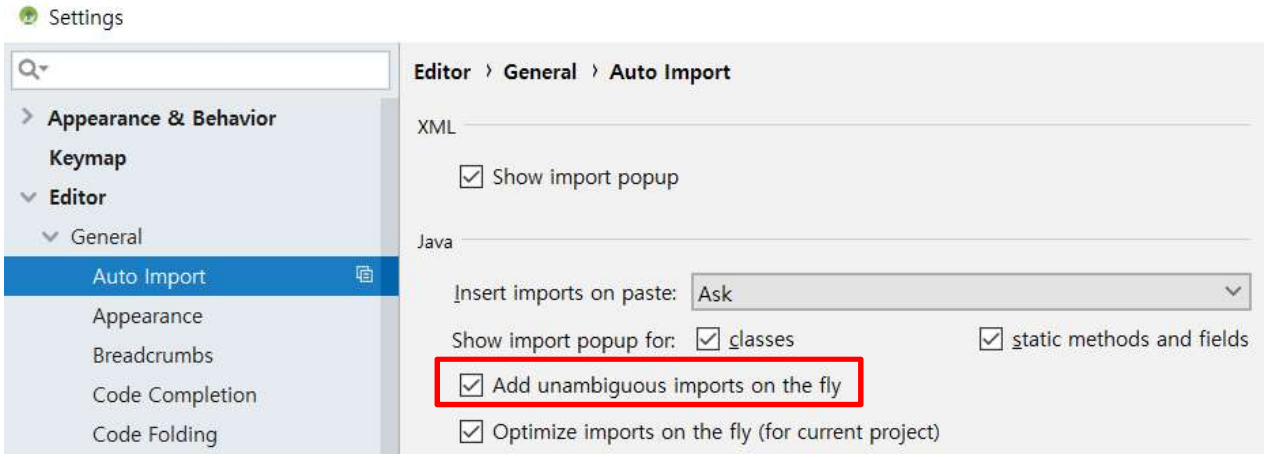
```
val dp = resources.getDimension(R.dimen.textView_size)
textView.setTextSize(TypedValue.COMPLEX_UNIT_DIP, dp)
```

```
val dp = resources
    .getDimensionPixelOffset(R.dimen.textView_size)
    .toFloat()
textView.setTextSize(TypedValue.COMPLEX_UNIT_DIP, dp)
```

위 2개의 메소드 실행 결과 return 되는 값의 단위는 dp
직접 텍스트의 폰트 크기를 지정했을 때 값의 단위는 px

```
textView.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20f)
```

Auto import 설정



잠깐! 코딩할 때 유용한 기능

- Java 소스 코드에서
  Ctrl-Q : 해당 method에 대한 설명
  Ctrl-P : method에서 입력해야 할 parameter

- 코드를 수정하고 나서 indent를 맞출 때
  Code > Reformat code

- 코드 블록을 주석(comment) 처리하고 싶으면
  Code > Comment with Block Comment
- 파일 이름을 바꾸고 싶으면
  파일 이름 클릭 > 마우스 오른쪽 버튼 > Refactor

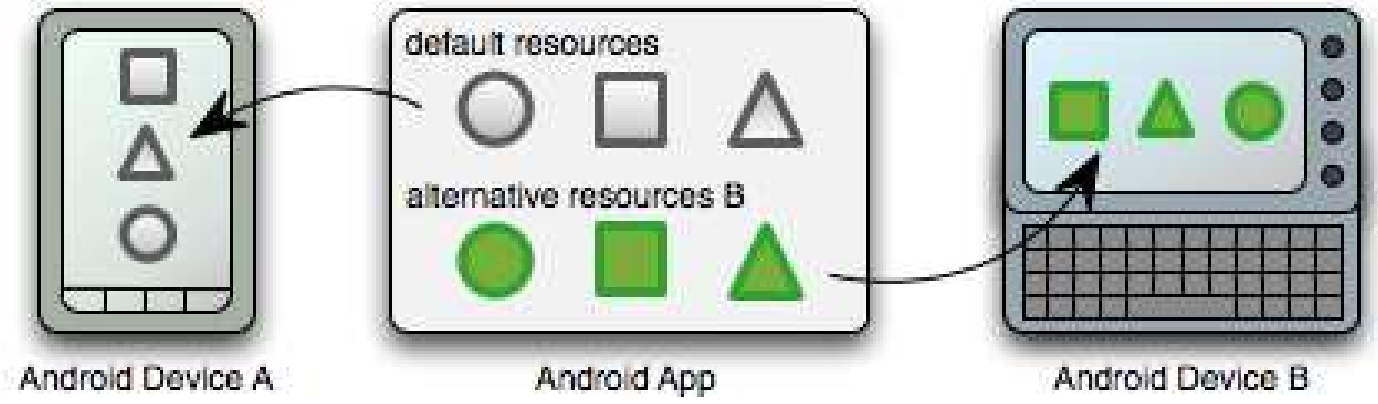
- XML에서 주석 처리하려면
  <!-- -->로 주석 처리할 부분을 둘러 쌈

What to do?

- Android XML
  - xmlns:android, xmlns:app, xmlns:tools
- 소스 코드 분석
  - package, import
  - 클래스 정의 및 상속
  - 메소드 재정의
  - setContentView 메소드
- **MVC**

코드와 리소스를 분리하는 이유

- Android가 다양한 장치에 탑재되면서
  - 언어나 화면 크기에 따라 리소스를 다르게 하는 것이 필요
- Android 에서는 XML을 이용하여 UI 화면을 구현하는 방법을 선호
  - App.의 UI와 business logic을 분리



Model-View-Controller(MVC)