3장 안드로이드 프로그램의 첫걸음

시작하면서

- **목차**
  - 프로젝트의 생성하기
  - 프로젝트 파일 및 소스 코드 이해
  - 코드로 문자열 표시하기
  - 문자열 출력 프로그램 응용
  - 프로젝트에 새로운 파일/속성 추가

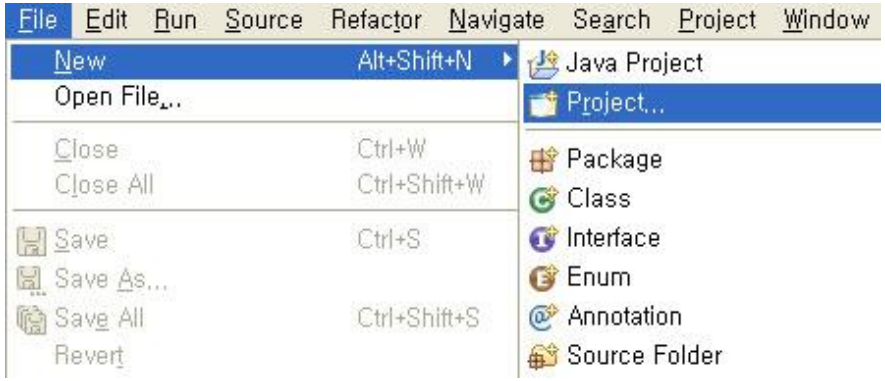
프로젝트 생성하기

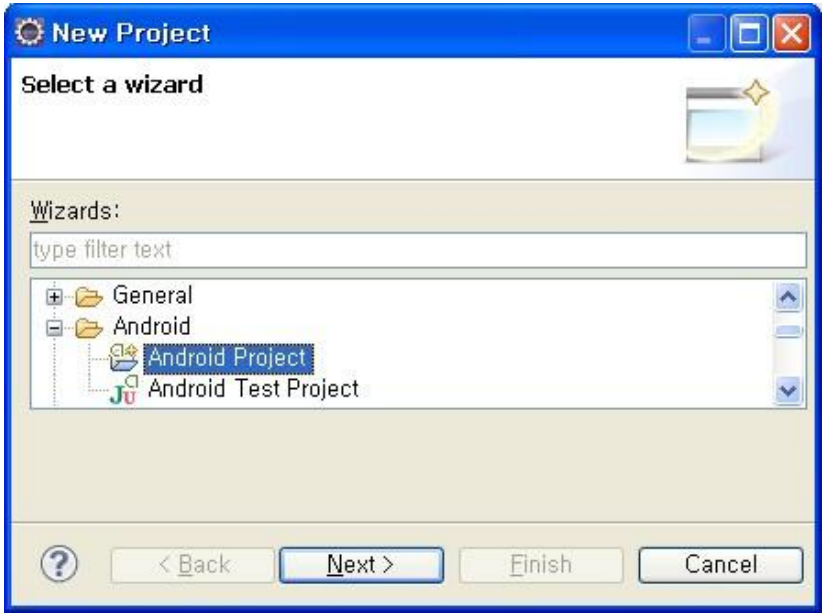
프로젝트 생성하기

- **<실습 5-1>: Android 프로젝트의 생성과 에뮬레이터 구동**

  (1)[그림 3-1](a)처럼 안드로이드 프로젝트 생성 아이콘 클릭

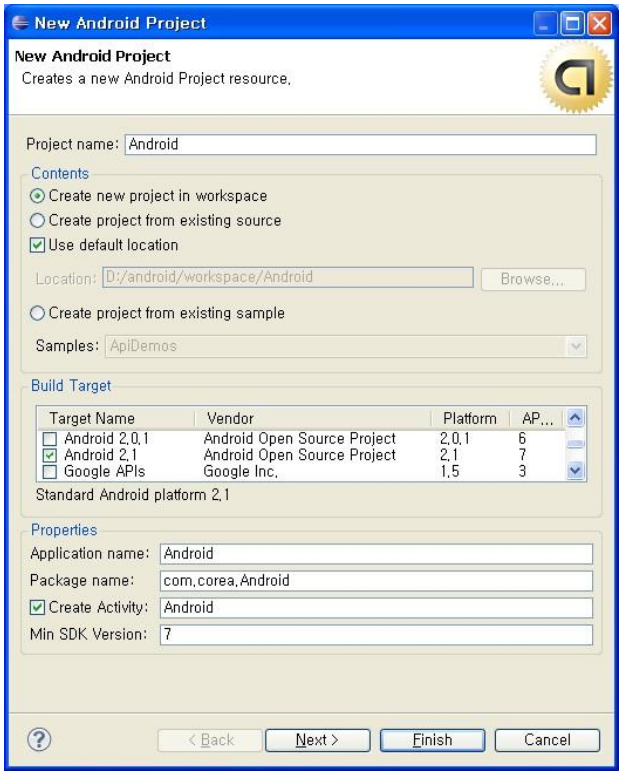
[그림 3-1](b)처럼 이클립스에서 메뉴 선택





프로젝트 생성하기

(2) ‘New Android Project' 마법사에서
Project name/Application name/Activity name: 'Android’ 입력
Package name: ’com.corea.Android' 입력
Target name: 'Android 2.2' 체크



프로젝트 생성하기

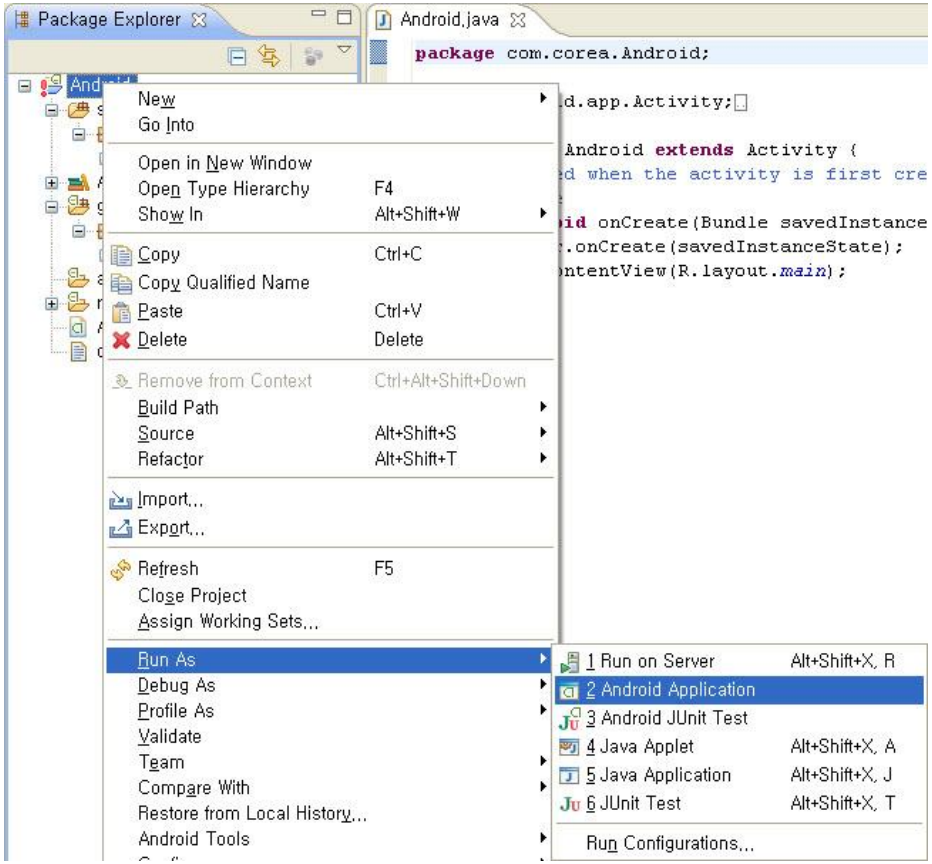
(3) 패키지 익스플로러 창에서 생성한 'Android' 프로젝트 확인
- 오른쪽 하단: 폴더 왼쪽에 !/x이 나타나면 오류가 있다는 표시

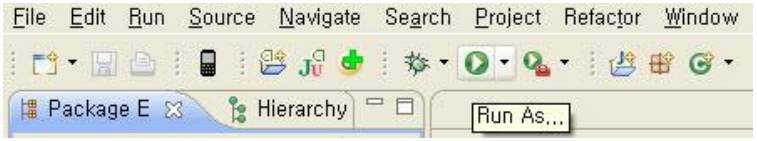
프로젝트 생성하기

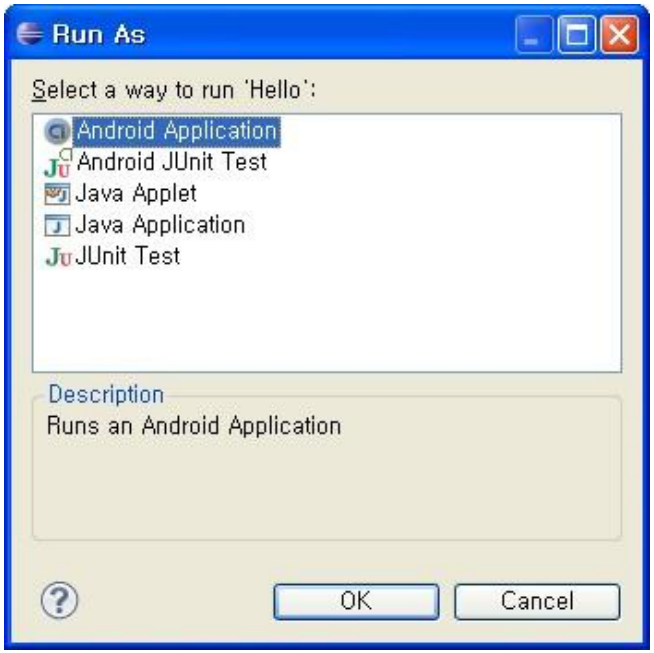
(4) 완성된 프로젝트를 에뮬레이터에 구동

1)이클립스의 패키지 익스플로러: [Run As]

2)이클립스 툴바: [Run As]







프로젝트 생성하기

(5) 에뮬레이터 부팅

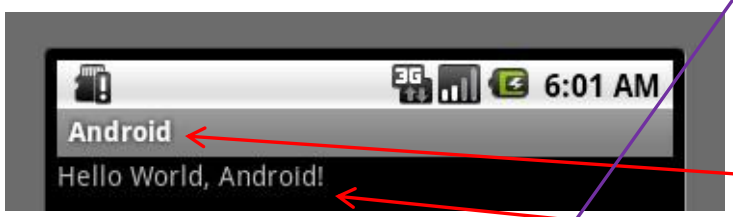
- 이클립스 하단부의 콘솔탭을 관찰

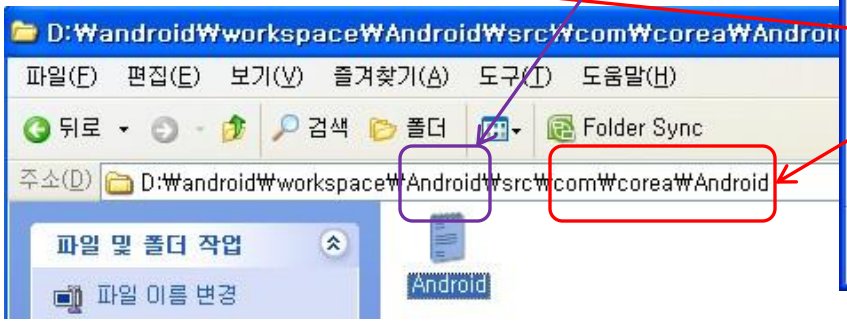
프로젝트 생성하기

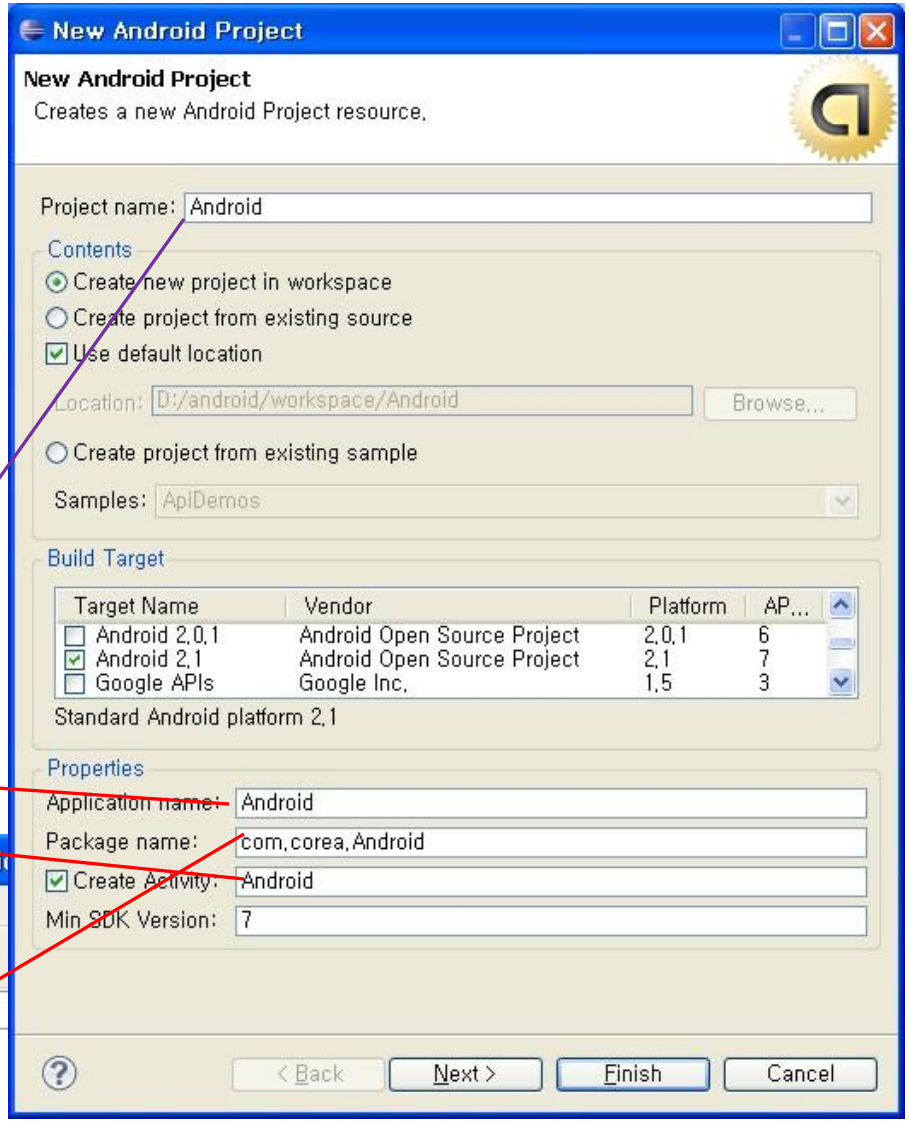
(6) 실행 결과

- 이클립스 하단부의 콘솔탭을 관찰
- 애플리케이션 수행 결과가 나타나지 않을 경우:
  락이 걸려 있는 것이므로 [MENU] 버튼을 클릭하여 해제

프로젝트 파일 경로


6:01 AM
Android
Hello World, Android!


D:₩android₩workspace₩Android₩src₩com₩corea₩Androi
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
뒤로 검색 폴더 Folder Sync
주소(D) D:₩android₩workspace₩Android₩src₩com₩corea₩Android
파일 및 폴더 작업
파일 이름 변경
Android


New Android Project
New Android Project
Creates a new Android Project resource.
Project name: Android
Contents
Create new project in workspace
Create project from existing source
Use default location
Location: D:/android/workspace/Android
Browse...
Create project from existing sample
Samples: ApiDemos
Build Target
Target Name Vendor Platform AP...
Android 2.0.1 Android Open Source Project 2.0.1 6
Android 2.1 Android Open Source Project 2.1 7
Google APIs Google Inc. 1.5 3
Standard Android platform 2.1
Properties
Application name: Android
Package name: com.corea.Android
Create Activity: Android
Min SDK Version: 7
< Back
Next >
Finish
Cancel

프로젝트 파일 및 소스 코드 이해

프로젝트 구성 파일

- **안드로이드 애플리케이션의 구성**
  - 새로운 프로젝트를 생성하면 다음의 폴더와 파일이 자동 생성됨
    - 소스 코드를 위한 /src와 /gen 폴더
    - 리소스 관리를 위한 /drawable, /layout, /values, /assets 폴더
      - XML, PNG, JPEG 등의 다양한 종류의 리소스 파일을 지원
    - 안드로이드 라이브러리
    - AndroidManifest.xml

프로젝트 구성 파일

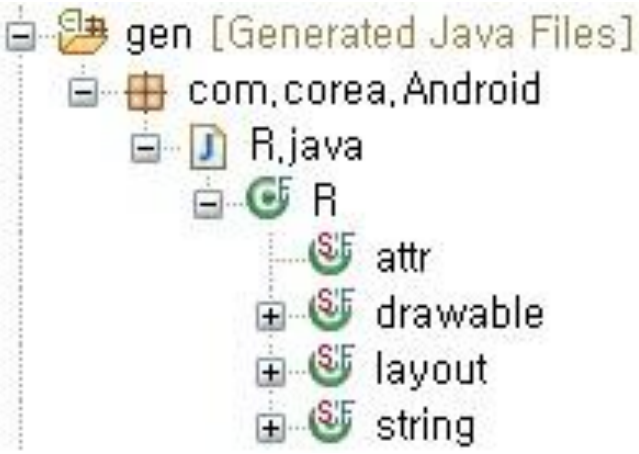
- **src 폴더**
  - 액티비티, 서비스, 컨텐트 프로바이더 등 애플리케이션 구성에 필요한 자바 형식의 소스 파일들로 구성
    - Android 프로젝트의 경우: 자동으로 생성된 액티비티에 대한 소스 코드 Android.java가 자동 생성

Android.java

```
package com.corea.Android;

import android.app.Activity;

public class Android extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

프로젝트 구성 파일

- **gen 폴더**
  - 자동 생성 코드는 Android.java와 같은 일반 소스 코드에게 외부 리소스 파일에 대한 정보를 제공
  - 안드로이드 애플리케이션은 소스 코드 외에 레이아웃, 그림, 문자열 등 여러 외부 리소스를 포함
  - R.java: 여러 외부 리소스 주소를 포함하므로 임의 변경 금지

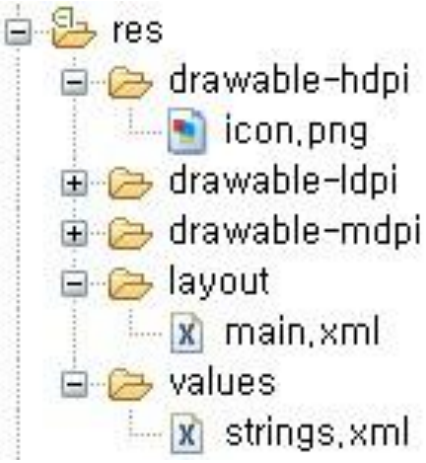


Android.java | R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.

package com.corea.Android;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

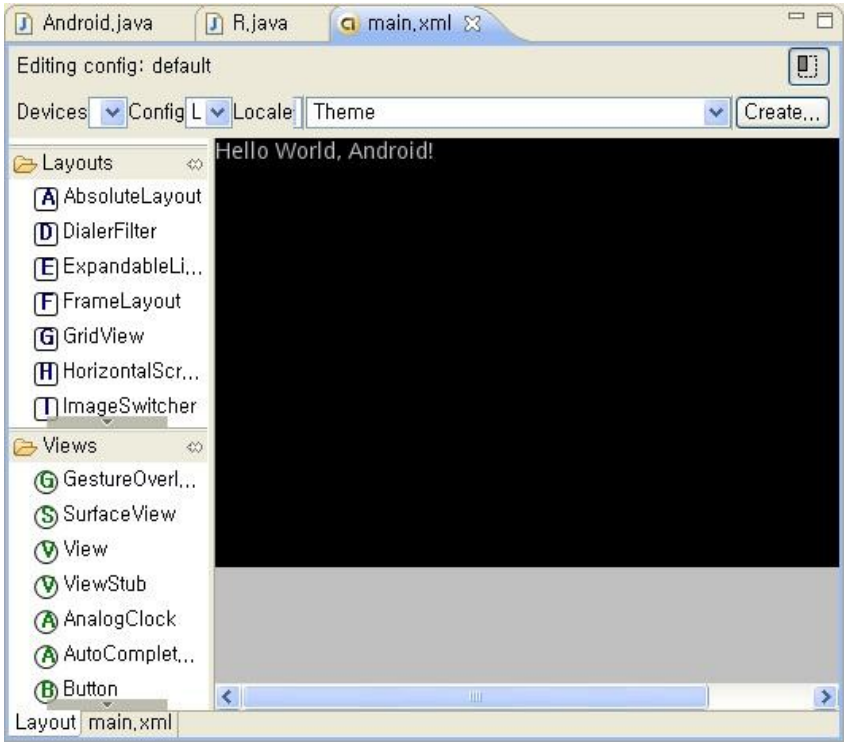
프로젝트 구성 파일

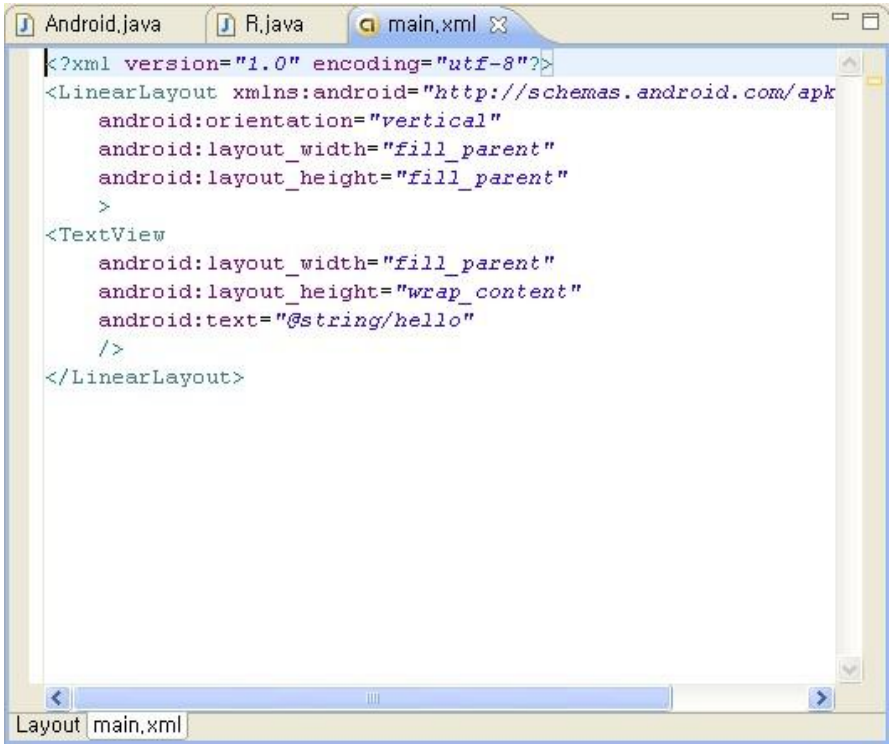
- **res 폴더**
  - 이미지 파일 저장: /drawable
    - jpg, png 등의 파일 형식을 지원
    - Android 프로젝트에는 어플리케이션의 아이콘인 icon.png를 포함
    - 리소스를 추가하면 해당 파일의 이름을 리소스 명으로 등록
    - 파일 이름은 모두 소문자 및 일부 한정된 특수문자([a-z0-9_.])로 구성
    - 해상도(고해상도, 중해상도, 저해상도)에 따라 /drawable-hdpi, /drawable-mdpi, /drawable-ldpi 폴더로 관리



프로젝트 구성 파일

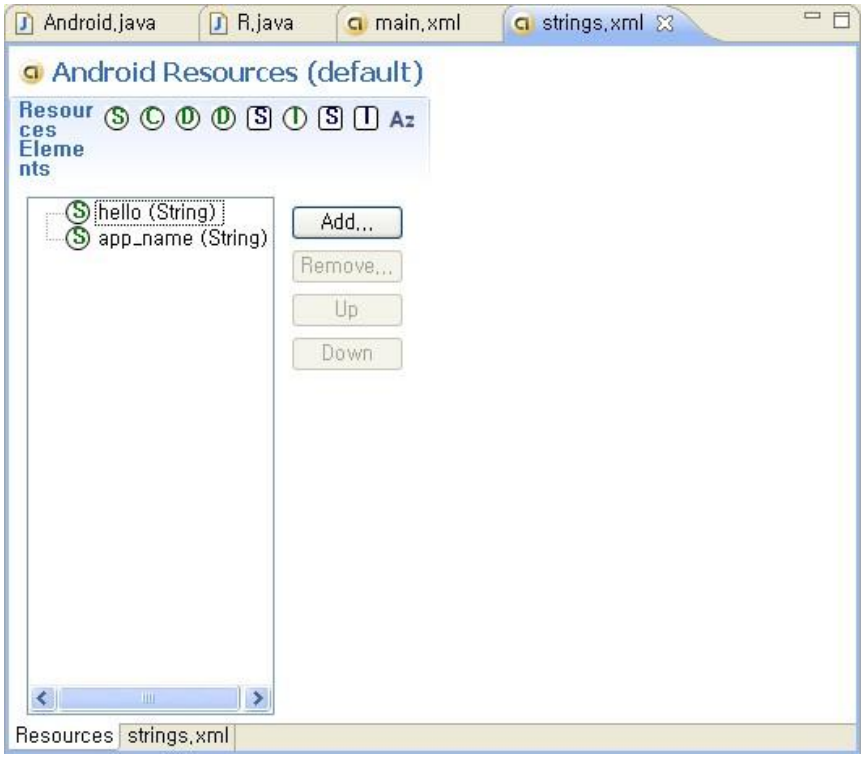
- 레이아웃: /layout
  - 어플리케이션의 화면을 구성하는 레이아웃 파일들을 포함
  - Android 프로젝트에는 Android 액티비티의 레이아웃인 main.xml 파일 포함

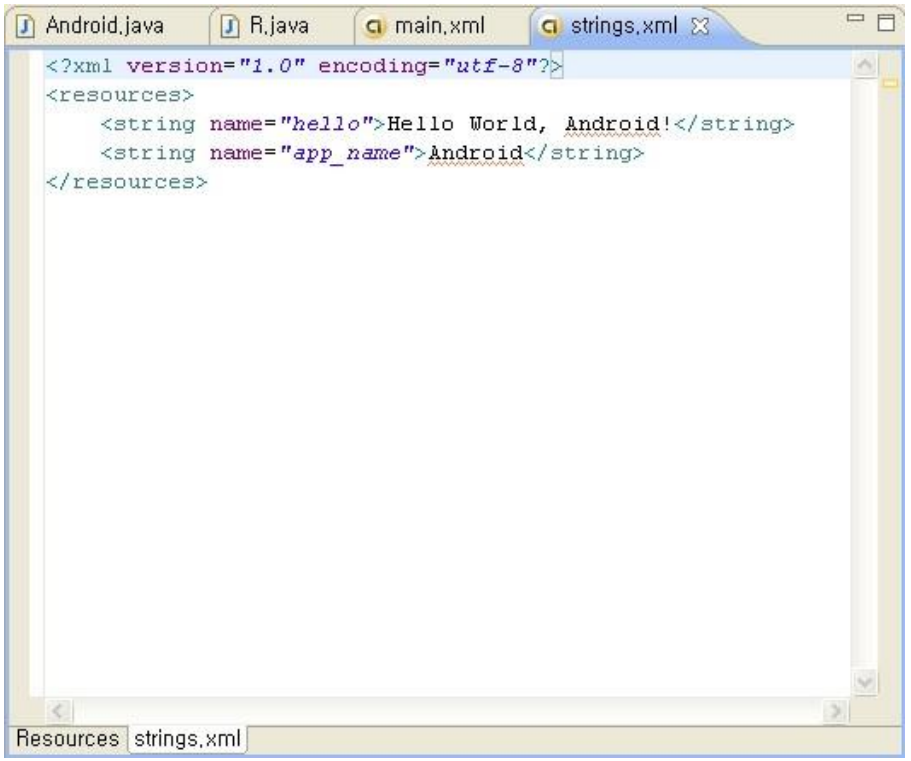




프로젝트 구성 파일

- 문자열 등 저장: /values
  - 문자열, 배열 등 기타 애플리케이션에서 사용하는 여러 값을 보관
  - Android 프로젝트에는 strings.xml 파일에 문자열 등 저장





프로젝트 구성 파일

- 매니페스트 파일: AndroidManifest.xml
  - 애플리케이션의 프로필 역할
  - 애플리케이션 이름, 버전, 컴포넌트(액티비티, 서비스 등) 정의, 사용권한, 사용하는 라이브러리 등 애플리케이션의 구성 정보를 포함

프로그램 소스 이해

- 프로그램 소스

```
package com.corea.Android;

import android.app.Activity;
import android.os.Bundle;

public class Android extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```
#include <stdio.h>

int main()
{
    printf(Hello World, Android!₩n");
    return 0;
}
```

프로그램 소스 이해

- **import android.app.Activity**
  - Activity란 응용 프로그램에서 하나의 화면을 구성하는 단위
  - Activity란 사용자 인터페이스 컴포넌트를 화면에 표시하고, 사용자의 반응을 처리
  - 애플리케이션 = 다수의 Activity의 집합

- **import android.os.Bundle**
  - Bundle은 프로그램 실행에 필요한 정보(예를 들면, toString, getString, getShort 등)를 포함하는 클래스
  - C 언어의 stdio.h에 대응

프로그램 소스 이해

- **onCreate(Bundle savedInstanceState)**
  - 액티비티가 실행될 때 호출
  - 액티비티의 UI를 적용하기 위한 setContentView()와 같은 메소드를 호출
  - Bundle 객체는 애플리케이션의 실행/종료에 따른 상태 정보를 저장
- **setContentView(R.layout.main)**
  - R 클래스의 내부 클래스인 layout 클래스의 main 필드를 화면에 출력
  - R.java 파일은 프로젝트내 여러 리소스 파일의 주소를 포함

프로그램 소스 이해

- **R.java**

R 클래스는 각종 리소스에 접근할 수 있는 포인터 주소가 저장
→ 수정 금지

```
package com.corea.Android;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

프로그램 소스 이해

레이아웃

- 뷰(view)는 화면 상에서 UI를 표시하는 기본 단위.
  예)텍스트뷰, 편집텍스트, 그림, 버튼 등
- 레이아웃(layout)은 각 뷰를 화면 상에 배치하고 구성하는 일종의 뷰그룹(View Group)
- 레이아웃은 XML로 표현. 코드로 표현할 수 있으나 복잡

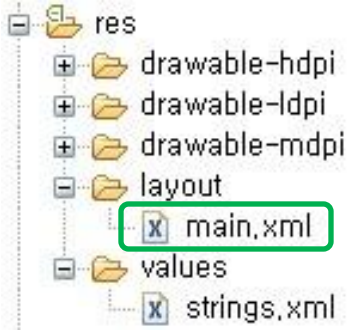
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"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

안드로이드 이름공간

컨테이너, 즉 부모가 가지는 길이를 모두 할당

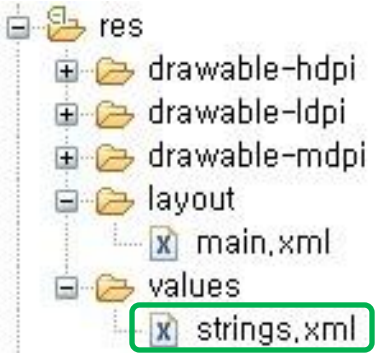
뷰의 넓이와 높이 지정

해당 뷰가 그려질 수 있는 필요한 길이만 할당



프로그램 소스 이해

- **hello와 문자열 값**




```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Android!</string>
    <string name="app_name">Android</string>
</resources>
```

프로그램 소스 이해

AndroidMenifest.xml

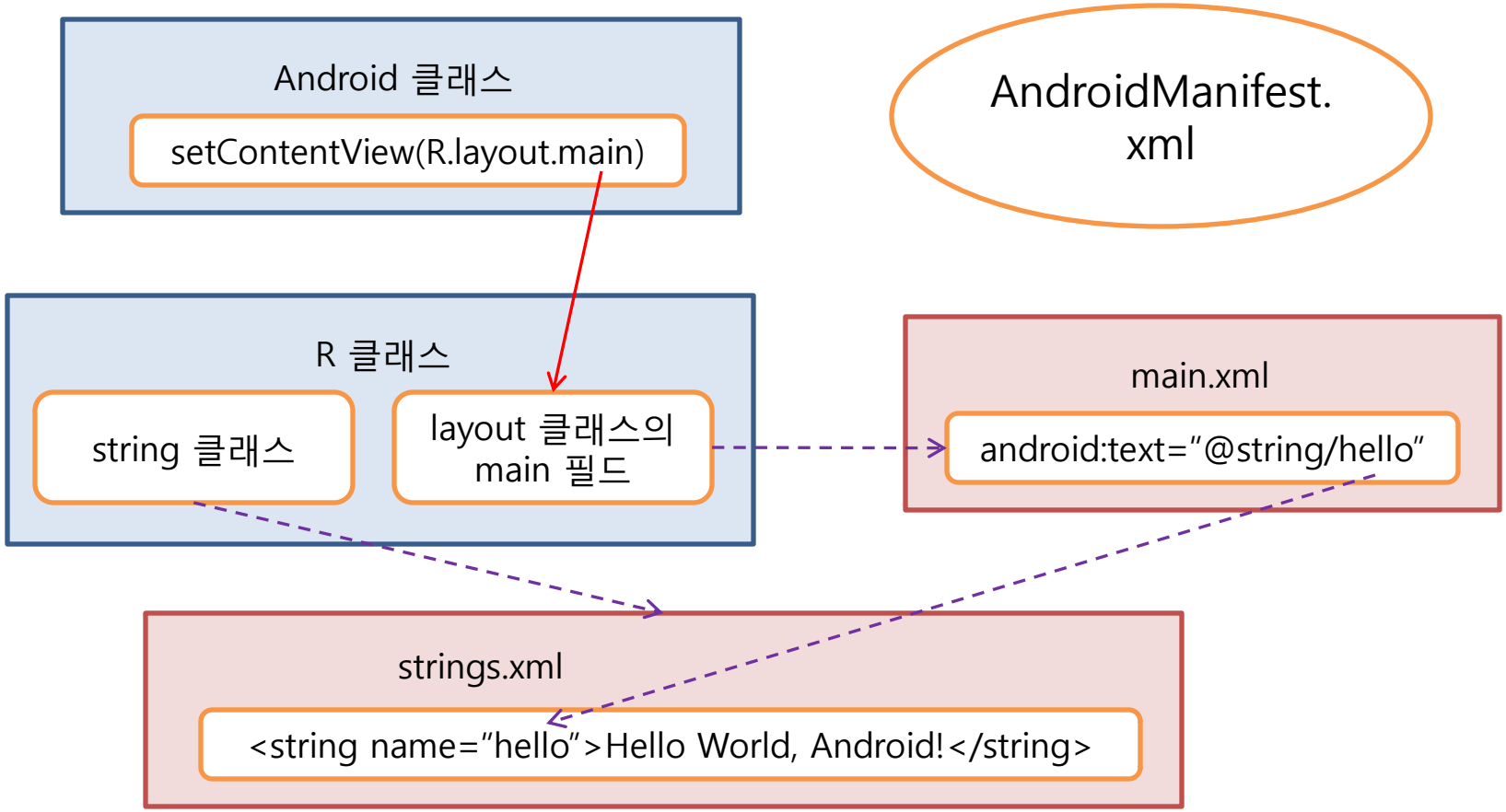
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.corea.Android"  ⟶ ❶ 패키지 이름
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
       android:label="@string/app_name">  ⟶ ❷ 애플리케이션 이름
        <activity android:name=".Android"
                  android:label="@string/app_name">
            <intent-filter>  ⟶ ❸ 인텐트 필터 정의
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="7" />

</manifest>
```

프로그램 소스 이해

- **AndroidMenifest.xml**
  - 매니페스트 파일은 안드로이드가 애플리케이션을 구동시 필요한 구성 정보
  - 주액티비티, 라이브러리 및 모듈, 접근권한 등을 서술
  - 내용
    - Android 애플리케이션이 하나의 액티비티로 구성
    - 인텐트 필터는 액티비티, 서비스, 브로드캐스트 리시버가 반응할 수 있는 인텐트 형식 기술
    - android.intent.action.MAIN은 애플리케이션의 진입점(entry point)을 의미
    - android.intent.category.LAUNCHER는 액티비티를 런처(launcher)에 노출
    - android.intent.action.MAIN과 android.intent.category.LAUNCHER의 조합은 해당 액티비티를 애플리케이션 시작시 처음 시작될 액티비티를 지정
    - 기호 @는 리소스 파일에서 접근할 수 있는 정보를 의미

프로그램 소스 이해

문자열 출력 과정



코드로 문자열 표시하기

코드로 문자열 표시하기

- UI를 코드에 통합한 방식

◉ **<실습 3-2>: Hello 프로젝트**

(1) <실습3-1>의 과정 (1)을 수행

(2) <실습3-1>의 과정 (2)를 수행,
Project name/Application name/Create Activity 항목: Hello 입력
Package name 항목: com.corea.Hello 입력

(3) 이클립스의 패키지 익스플로러 창에 방금 생성한 'Hello'라는 프로젝트 생성 확인. src 폴더 아래에 있는 Hello.java를 더블 클릭

(4) import 문장 옆의 십자가 동그라미를 클릭하여 import 문장을 펼침

(5) 소스 코드를 수정.

```
setContentView(R.layout.main);
⇓
TextView hello = new TextView(this);
hello.setText("Hello, Android!");
setContentView(hello);
```

(6) [Ctrl] + [Shift] + [o]를 클릭

(7) <실습 3-1>의 과정 (4)~(6)을 수행.

<실습 3-1>과 동일한 "Hello, Android!"라는 문자열이 에뮬레이터로 출력

프로그램 소스 수정

- 프로젝트의 소스 코드 수정 절차

main.xml | strings.xml | Android.java

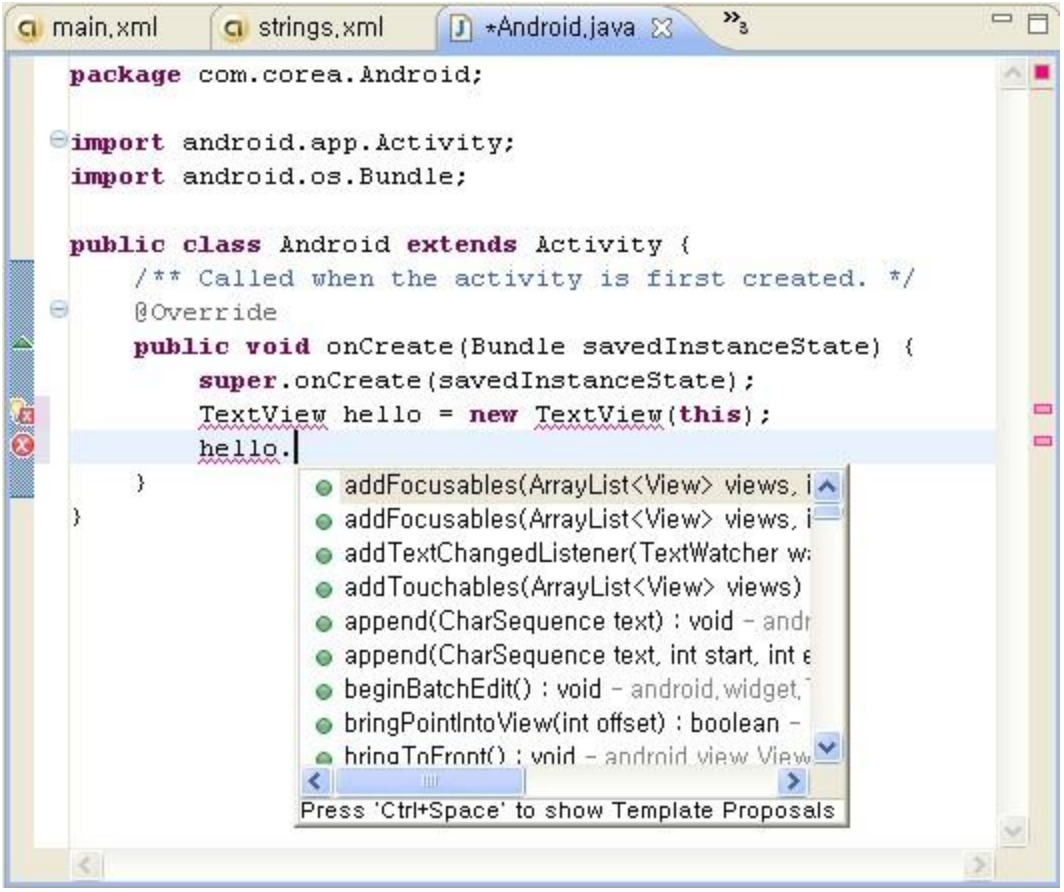
```
package com.corea.Android;

import android.app.Activity;
import android.os.Bundle;

public class Android extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

main.xml | strings.xml | *Android.java

```
package com.corea.Android;

import android.app.Activity;
import android.os.Bundle;

public class Android extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView hello = new TextView(this);
        hello.setText("Hello World, Android!");
        setContentView(hello);
    }
}
```

프로그램 소스 수정

- 필요한 메소드 선택



프로그램 소스 수정

- 자동으로 필요한 import 문장 추가
  - ctrl + shift + o

*Hello.java

```
1  package com.corea.Hello;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5
6  public class Hello extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         TextView hello = new TextView(this);
12         hello.setText("Hello, Android!");
13         setContentView(hello);
14     }
15 }
```

- 누르기 전

*Hello.java

```
1  package com.corea.Hello;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.TextView;
6
7  public class Hello extends Activity {
8      /** Called when the activity is first created. */
9      @Override
10     public void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         TextView hello = new TextView(this);
13         hello.setText("Hello, Android!");
14         setContentView(hello);
15     }
16 }
```

누른 후

프로그램 소스 수정

- **<실습 3-2> 코드로 문자열 표시하기**
  - <실습 3-1>과 동일한 방식으로 실행
  - 실행결과는 동일
  - 차이점
    - 리소스 클래스인 R.java를 사용하지 않았음
    - 프로그램 소스와 사용자 인터페이스를 분리하지 않았음

프로그램 소스 분석

- 수정된 소스 코드

```
package com.corea.Android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Android extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView hello = new TextView(this);
        hello.setText("Hello World, Android!");
        setContentView(hello);
    }
}
```

프로그램 소스 분석

- **TextView**
  - 화면에 문자열을 표시
  - 생성자

```
Public TextView(Context context)
Public TextView(Context context, AttributeSet attrs)
Public TextView(Context context, AttributeSet attrs, int defStyle)
```

  - TextView(this)에서 this는 현 애플리케이션 지칭
  - Activity 클래스의 상속도

```
Java.lang.Object
    android.content.Context
        android.content.ContextWrapper
            android.view.ContextThemeWrapper
                android.app.Activity
```

프로그램 소스 분석

- **setText()**
  - 형식

```
public void setText(CharSequence text)
public void setText(int resid)
```

- **setContentView()**
  - 레이아웃 혹은 뷰(view)를 액티비티의 화면에 표시
  - 형식

```
public void setContentView(int layoutResID)
public void setContentView(View view)
```

  - hello는 TextView
  - TextView는 View의 하위클래스

문자열 출력 프로그램 응용

색상과 크기 변화

- **<실습 3-3>: 색상과 크기 변화**

(1) <실습3-1>의 과정 (1) 수행

(2) <실습3-1>의 과정 (2) 수행

Project name/Application name/Create Activity 항목: ColorSize 입력

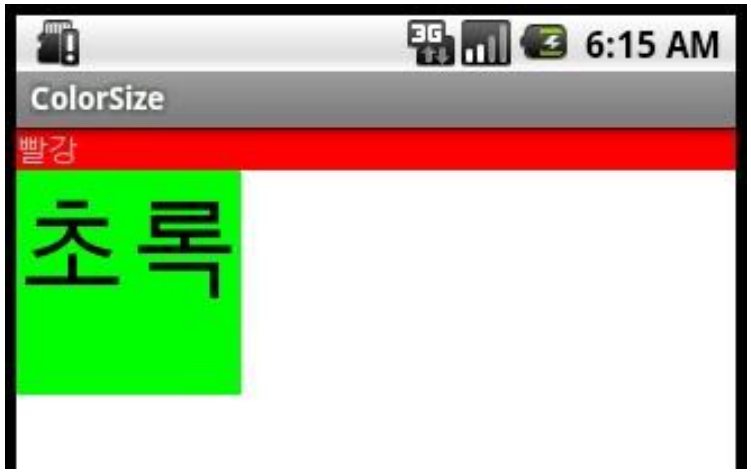
Package name 항목: com.corea.ColorSize 입력

(3) 이클립스의 패키지 익스플로러 창에 방금 생성한 'ColorSize'라는 프로젝트 생성 확인. res/layout 폴더 아래의 main.xml를 더블 클릭

(4) <코드 3-7>과 같이 main.xml을 수정

(5) <실습 3-1>의 과정 (4)~(6) 수행

(6) 옆 그림처럼 에뮬레이터에 출력



색상과 크기 변화

- <코드 3-7> main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffffff">
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="빨강"
    android:background="#ffff0000" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="100dp"
    android:text="초록"
    android:textColor="#ff000000"
    android:textSize="50sp"
    android:background="#ff00ff00" />
</LinearLayout>
```

TextView 속성과 메소드

TextView 속성과 메소드: <표 3-1>

속성	메소드	의미
background	setBackgroundColor()	배경색 설정
gravity	setGravity()	문자열이 표시될 위치
text	setText()	문자열 표시
textColor	setTextColor()	문자색 설정
textSize	setTextSize()	문자의 크기 설정
textStyle	setTextStyle()	문자 스타일 설정
visibility	setVisibility()	문자열의 가시성 설정

- Gravity 속성: 뷰에서 텍스트 위치
  - top, bottom, left, right, center, center_vertical, center_horizontal
  - fill, fill_vertivcal, fill_horizontal
- Visibility 속성: visible, invisible(안 보이나 공간 차지), gone(안보이고 공간 차지하지 않음)

색상과 크기 표현

- **색상**
  - 색상 값은 항상 #로 시작
  - 비트 수와 알파(투명도) 여부에 따라 4가지 형태
    - #RGB
    - #ARGB
    - #RRGGBB
    - #AARRGGBB
    - 여기서 알파 값은 클수록 불투명
- **크기**
  - 픽셀: px
  - 인치: in
  - 밀리미터: mm
  - 포인트: pt
  - 밀도 독립 픽셀: dp(density-independent pixel)
  - 축척 독립 픽셀: sp(scale-independent pixel)

1dp는 160dpi화면에서 1px에 대응

이미지 출력 프로그램

- **<실습 3-4>: 이미지 출력 프로그램**

(1) <실습3-1>의 과정 (1) 수행

(2) <실습3-1>의 과정 (2) 수행

Project name/Application name/Create Activity 항목:WaveLogo 입력

Package name 항목: com.corea.WaveLogo 입력

(3) 이클립스의 패키지 익스플로러 창에 방금 생성한 'WaveLogo'라는 프로젝트 생성 확인. res/layout 폴더 아래의 main.xml를 더블 클릭

(4) 이미지 wavelogo를 res/drawable-ldpi에 복사

이미지 출력 프로그램

(5) 패키지 익스플로러 창의 res/drawable-ldpi 폴더 확인-마우스 오른 쪽 버튼을 클릭하여 팝업창이 나타나면 Refresh 항목을 선택

(6) res/drawable-ldpi 폴더 아래에 wavelog라는 이미지를 관찰

이미지 출력 프로그램

(7) res/layout 폴더 아래의 main.xml 더블 클릭, 이클립스 자바 편집기 하단의 main.xml 탭 클릭

(8) <코드 3-8>과 같이 main.xml 수정(07~11라인)

```
<?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”>
<ImageView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/wavelogo”
/>
</LinearLayout>
```

이미지 출력 프로그램

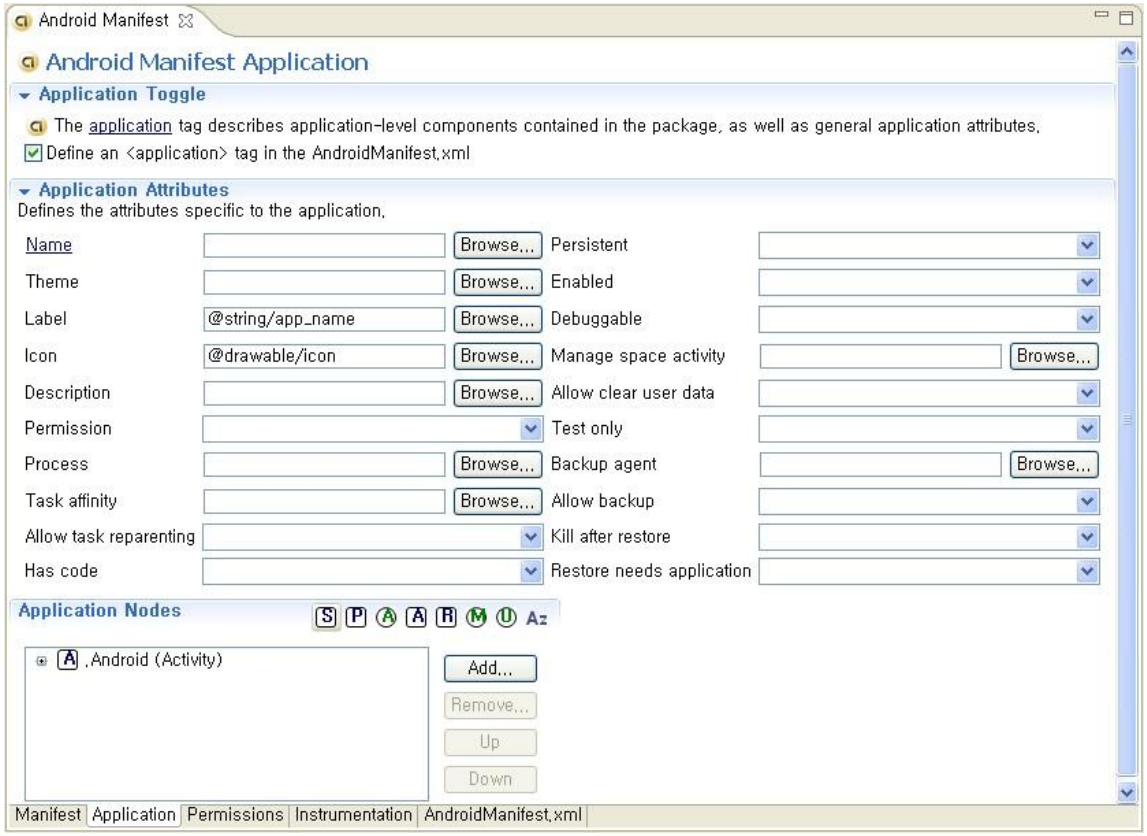
(9) <실습 3-1>의 과정 (4)~(6) 수행, 아래 그림처럼 이미지가 에뮬레이터에 출력

프로젝트에 파일/속성 추가

액티비티 추가

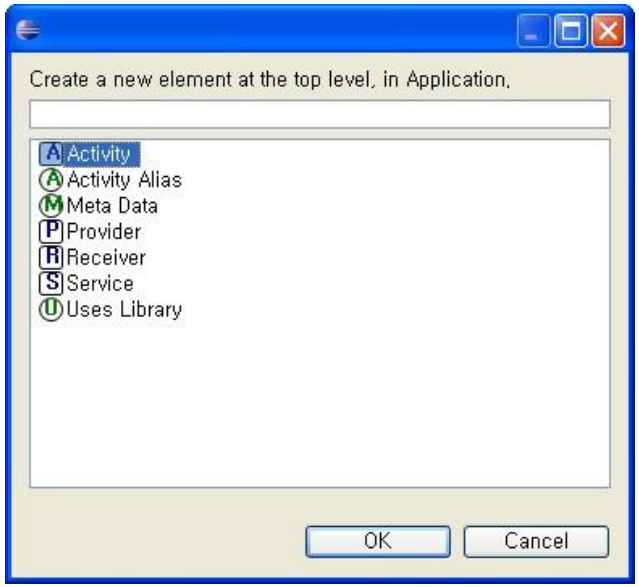
- **<실습 3-5>: 액티비티 추가**

  (1) 패키지 익스플로러에서 액티비티를 추가하고자 하는 프로젝트(여기서는 Android)의 AndroidManifest.xml 파일을 더블 클릭하여 열음

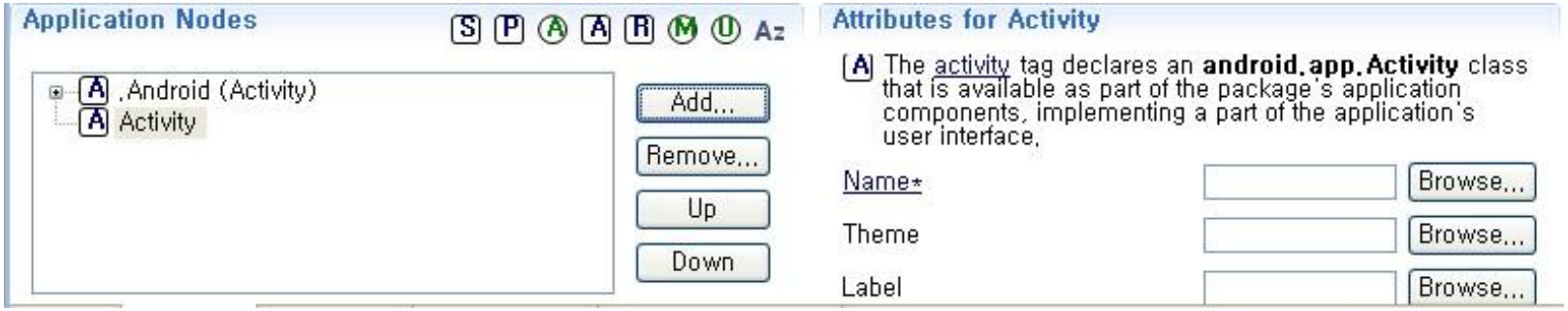


액티비티 추가

(2) 마법사에서 Activity 항목 선택 후 [OK] 버튼 클릭



(3) Application Nodes 부분에 새로운 Activity를 위한 노드 생성 확인



액티비티 추가

(4) 실질적인 자바 클래스 생성을 위하여 [그림 3-27]의 오른쪽 화면에서 Name* 부분 클릭

(5) 'New Java Class'라는 마법사가 나타남,

Name 항목에 클래스 이름 부여: 'NewAndroid' 입력

'Inherited abstracted methods' 항목 체크 후, [Finish] 버튼 클릭

→ 이클립스의 자바 편집기 창에 NewAndroid.java의 소스 코드 생성됨

*Android Manifest | NewAndroid.java

```
1  package com.corea.Android;
2
3  import android.app.Activity;
5
6  public class NewAndroid extends Activity {
7
8      /** Called when the activity is first created. */
9      @Override
10     public void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12
13         // TODO Auto-generated method stub
14     }
15
16 }
```

액티비티 추가

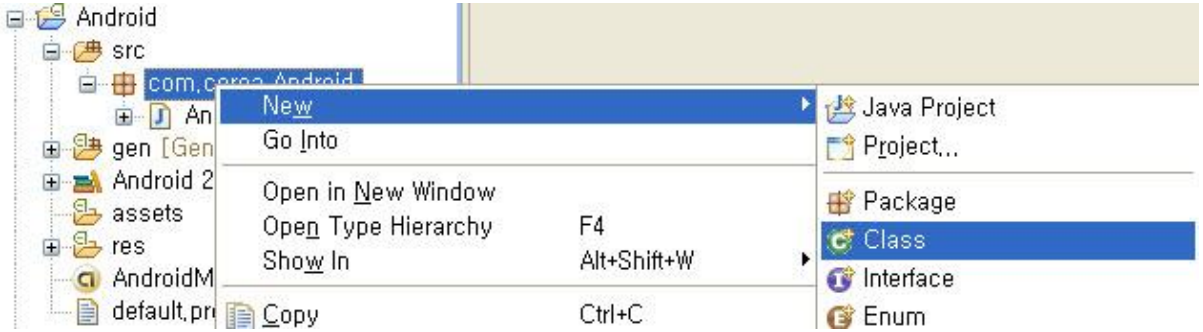
(6) AndroidManifest 파일을 클릭한 후 Androidmanifest.xml 탭을 선택
→15번 줄에 NewAndroid 액티비티를 위한 태그의 추가 확인

자바 파일 추가

- **<실습 3-6>: 자바 파일 추가**

  (1) Android 프로젝트 선택. src 폴더에 속한 패키지(com.corea.Android)를 오른쪽 마우스로 클릭

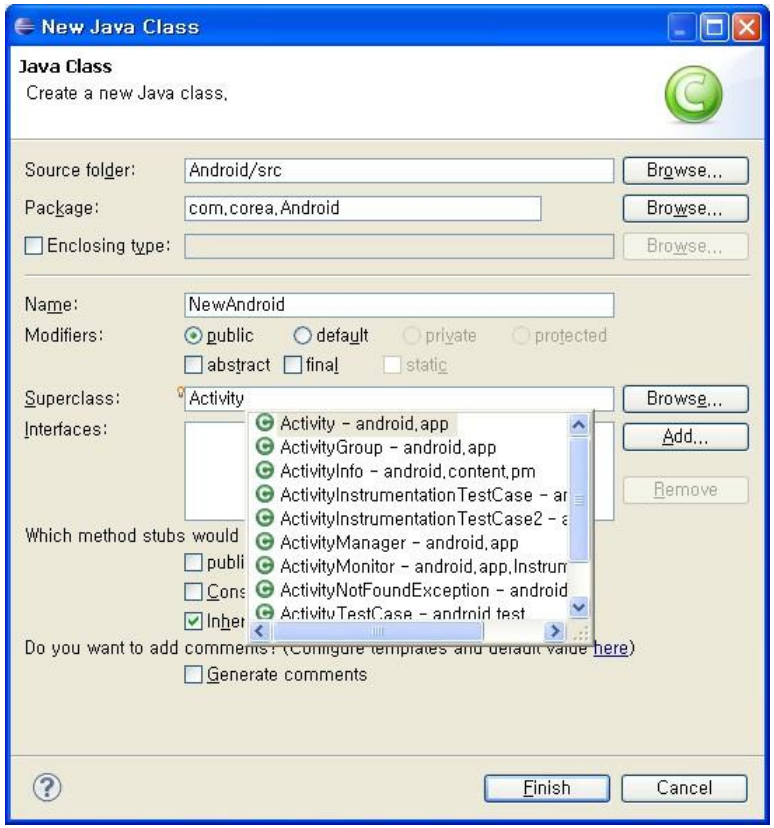
  (2) 나타난 팝업 창에서 New → Class 항목 선택



자바 파일 추가

(3) 'New Java Class' 마법사가 나타나면 클래스 이름 입력

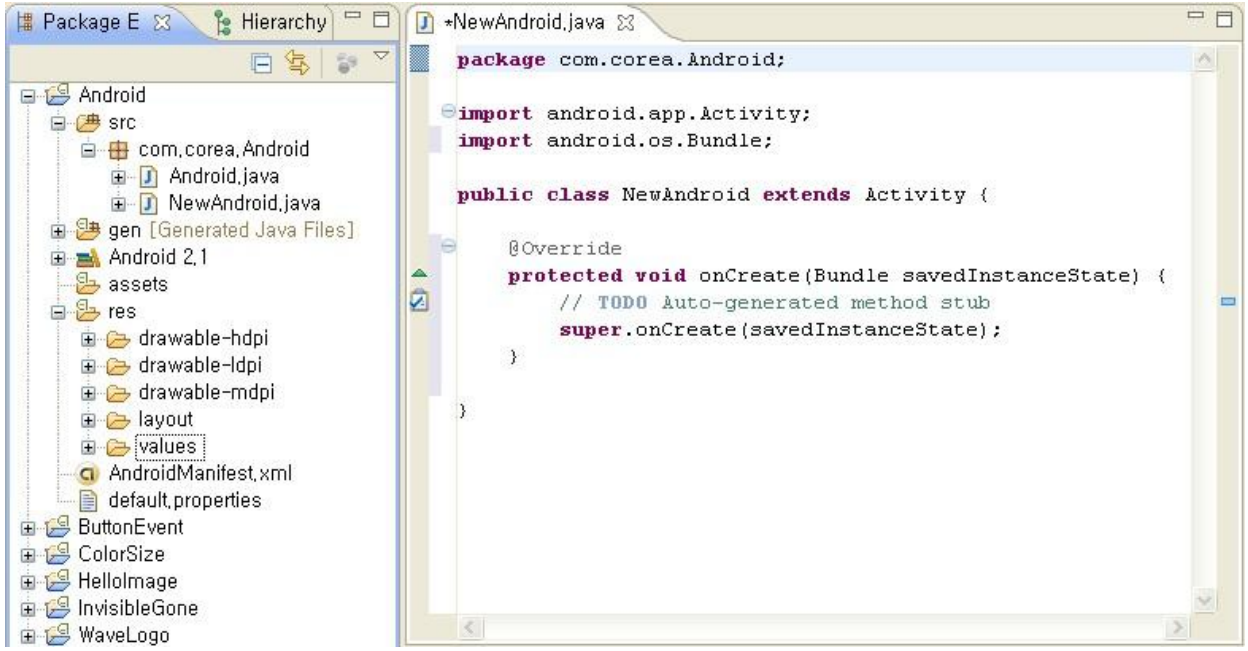
- 수퍼클래스 항목: 'Activity'를 수퍼클래스 이름으로 입력
- [Ctrl] + [Space] 키 클릭, Activity와 관련된 다양한 라이브러리가 나타나면 'Activity - android.app' 선택



자바 파일 추가

(4) 'New Java Class' 마법사 하단의 [Finish] 버튼 클릭

→ 자바 편집기에 기본 소스가 보여짐



기타

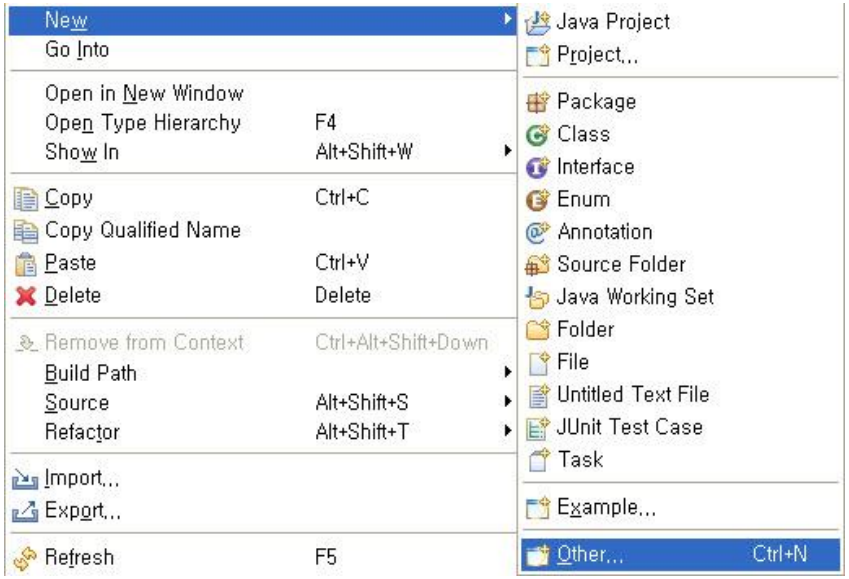
- XML 파일 추가
- 매니페스트 파일에 권한 추가
- 매니페스트 파일에 외부 라이브러리 추가

XML 파일 추가

- **<실습 3-7>: XML 파일 추가**

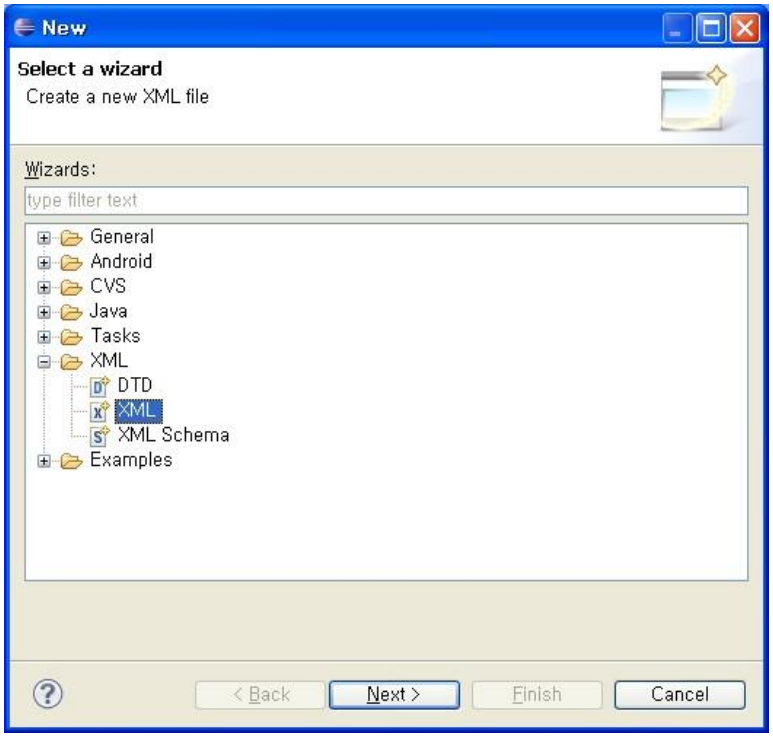
  (1) Android 프로젝트 선택. src 폴더에 속한 패키지(com.corea.Android)를 오른쪽 마우스로 클릭

  (2) 나타난 팝업 창에서 New → Other 항목 선택



XML 파일 추가

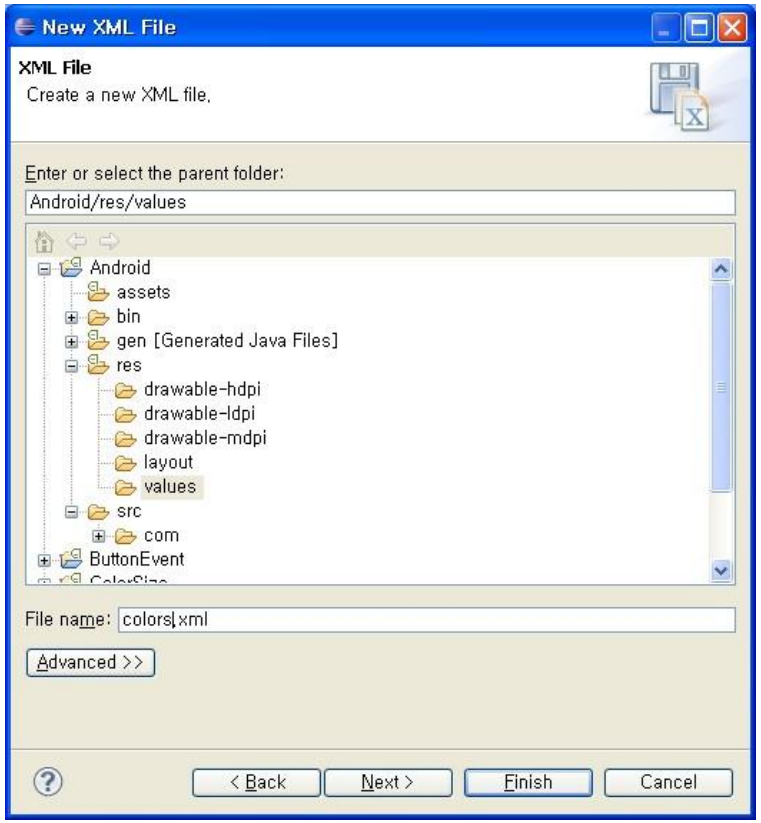
(3) 'New' 마법사가 나타나면 XML → XML 항목 선택 후 [Next] 버튼 클릭



XML 파일 추가

(4) 'New XML File' 마법사가 나타나면 적절한 파일 이름(예, colors.xml)을 입력한 후 [Finish] 버튼 클릭

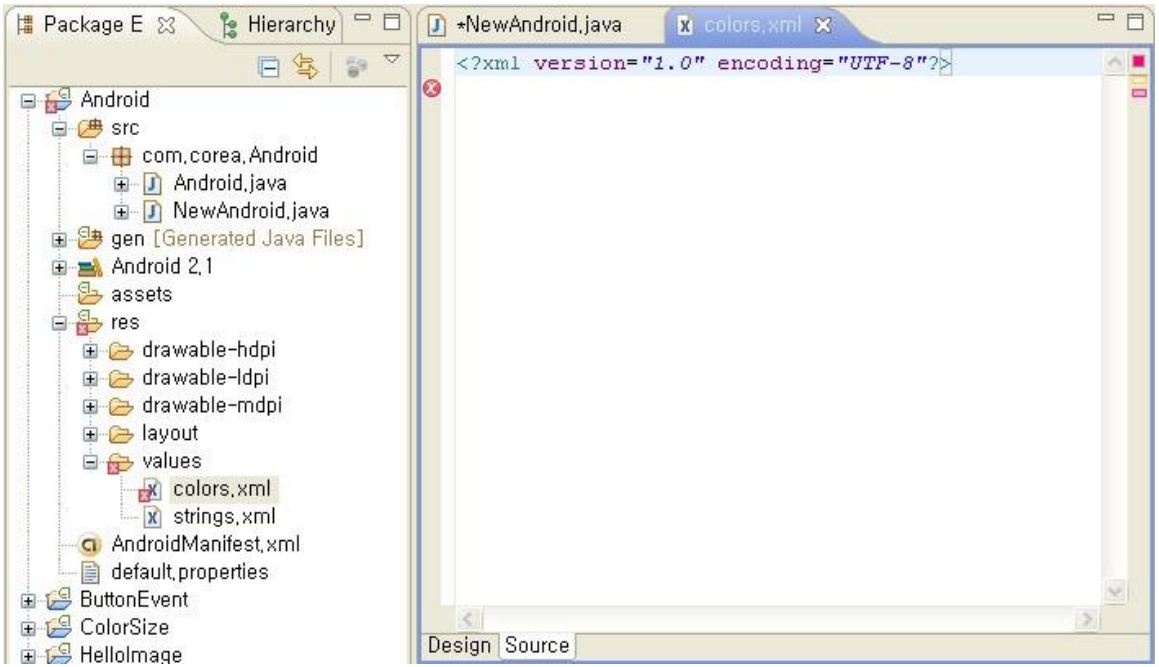
- 필요하다면 XML 파일이 추가될 폴더 위치 변경



XML 파일 추가

(5) 추가된 XML 파일 관찰

- 생성된 파일이 완전한 내용이 아니기 때문에 오류가 표시
- 적합한 내용을 완성하면 오류 표시가 사라짐

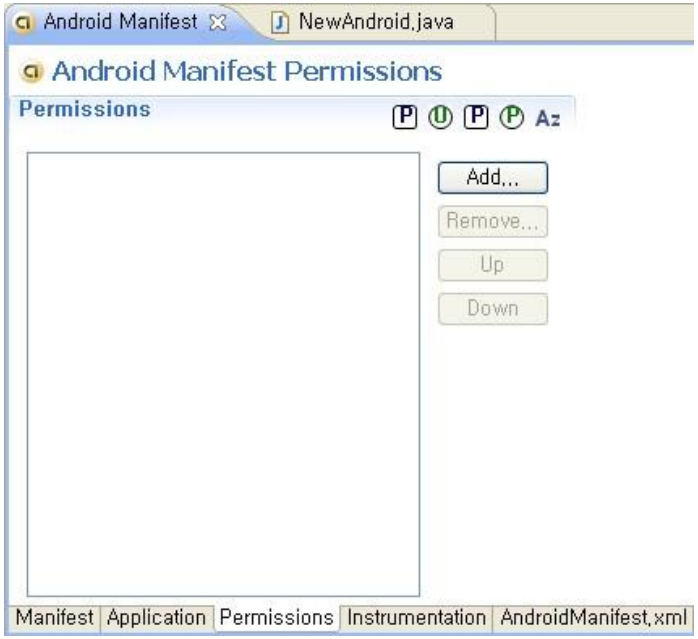


매니페스트 파일에 권한 추가

- **<실습 3-8>: 권한 추가**

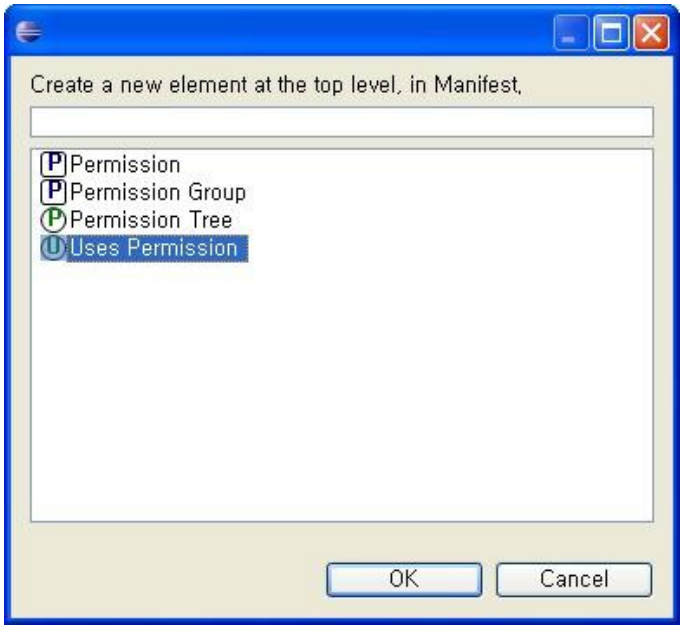
  (1) 매니페스트 파일을 열고, 매니페스트 편집기의 권한 탭 클릭

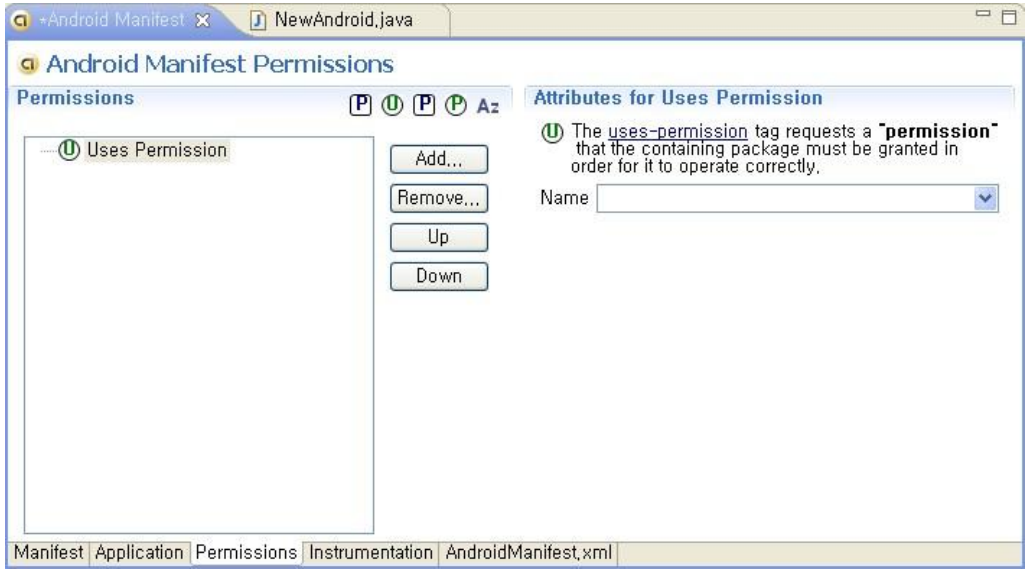
  - 애플리케이션에게 사용 권한을 추가하기 위하여 [Add] 버튼 클릭



매니페스트 파일에 권한 추가

(2) 'Uses Permission' 선택 후 [OK] 버튼 클릭



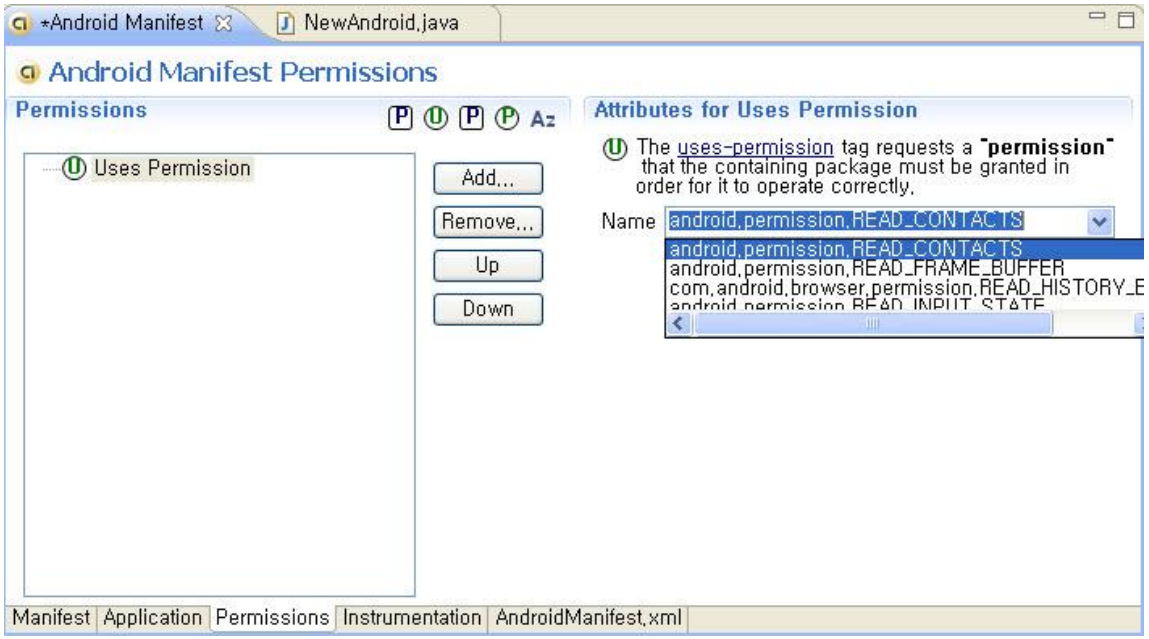


(3) 매니페스트 편집기 오른쪽에 'Attributes for Uses Permission'에 관한 내용이 보여짐

매니페스트 파일에 권한 추가

(4) Name 항목에 애플리케이션에서 필요한 권한을 선택하면 매니페스트 파일에 추가됨.

- 권한 추가 후 AndroidManifest.xml 탭을 클릭하여 추가된 권한 확인



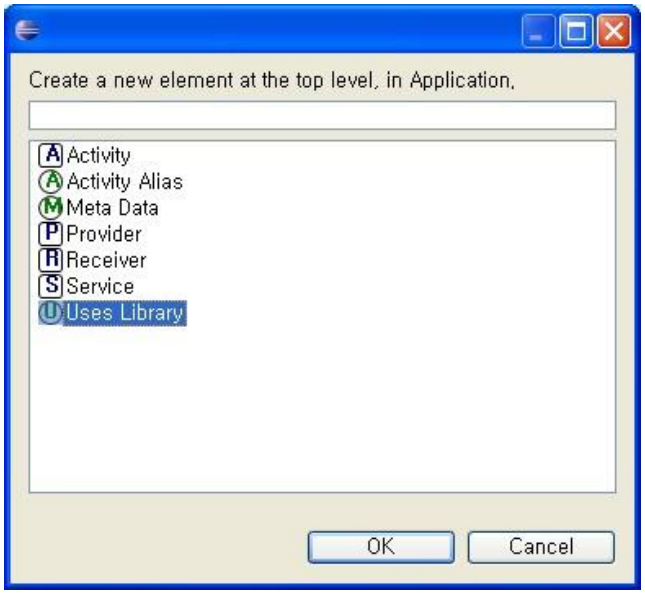
외부 라이브러리 추가

- **<실습 3-9>: 외부 라이브러리 추가**

  (1) 매니페스트 파일을 열고 매니페스트 편집기의 Application 탭을 클릭
  - 편집기 하단의 'Application Nodes' 화면의 [Add] 버튼 클릭

외부 라이브러리 추가

(2) 'Uses Library' 항목 선택 후 [OK] 버튼 클릭



(3) 매니페스트 편집기 오른쪽에 'Attributes for Uses Library'라는 속성 창이 나타나면, Name 항목에 필요한 외부 라이브러리를 선택, Required 항목에 true 값 선택

→ AndroidManifest.xml 탭을 클릭하여 추가된 라이브러리 확인