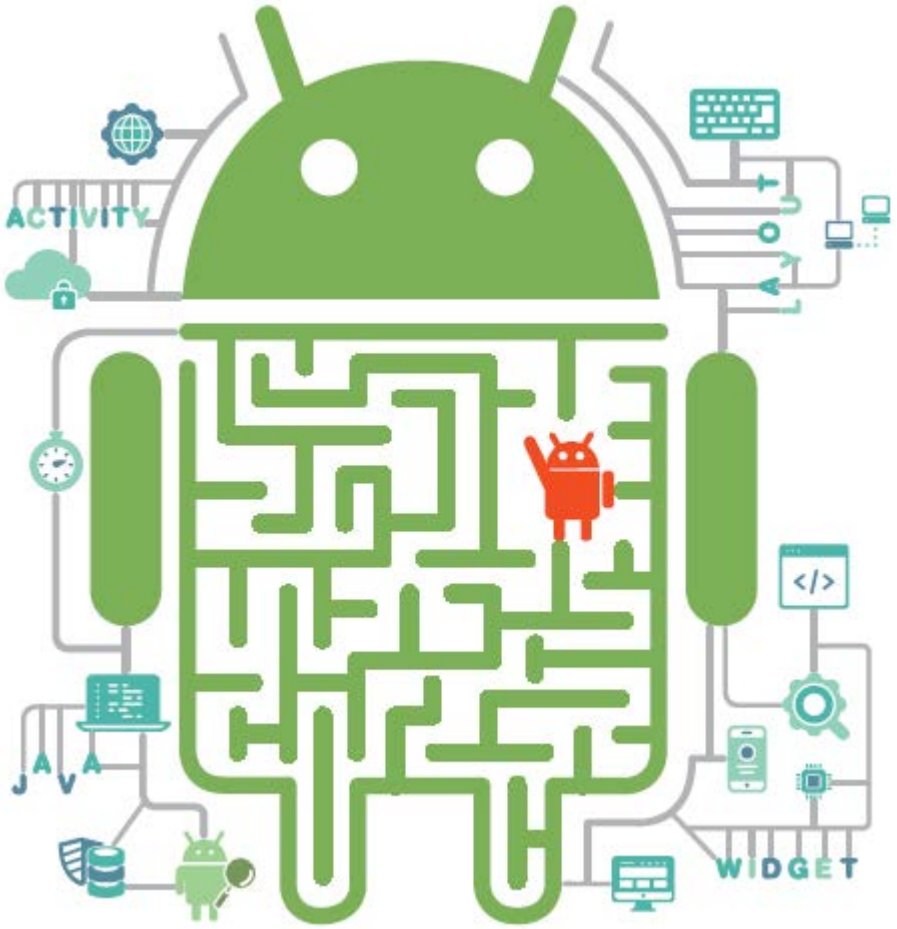
원리를 알면 IT가 맛있다
IT COOKBOOK

단계별로 배우는 안드로이드 프로그래밍

[강의교안 이용 안내]



단계별로 배우는

안드로이드 프로그래밍

Chapter 04. 액티비티(Ⅰ)

한빛아카데미 Hanbit Academy, Inc.

목차

학습목표

- Log 클래스와 logcat 사용법을 익힌다.
- 액티비티 생명주기 개념을 이해한다.
- 액티비티 상태를 저장하고 복원할 수 있다.
- 기본 리소스와 대체 리소스를 활용한다.

01 이벤트 로깅과 logcat ▶ Log 클래스

이벤트 로깅(event logging)

- 발생하는 이벤트(event), 즉 사건을 어딘가에 기록하는 행위
- 기록해둔 이벤트는 나중에 언제든지 살펴볼 수 있고, 이를 통해 문제점 찾기 가능

Log 클래스

- Log 클래스는 안드로이드 앱의 이벤트 로깅에 가장 많이 사용
- 로그 메시지의 중요도에 따라 여러 메서드를 제공
- 단순한 정보 출력부터 심각한 오류 처리에 이르기까지 다양하게 활용

안드로이드 시스템의 구조 관점에서 본 동작 원리

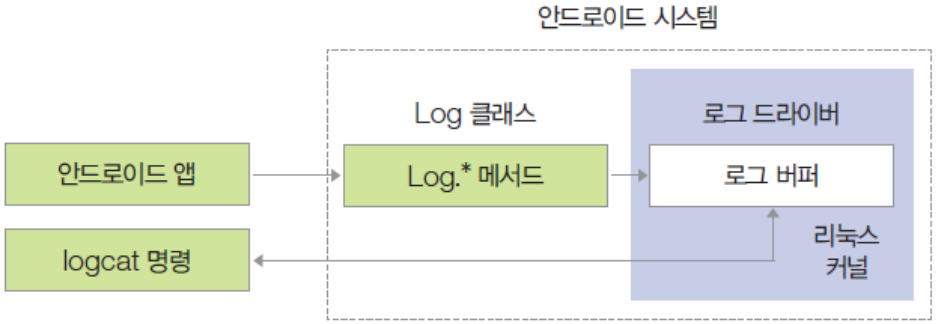


그림 4-1 Log 클래스 동작 원리

01 이벤트 로깅과 logcat ▶ Log 클래스

- 리눅스 커널: 로그 드라이버를 통해 로그 버퍼(log buffer) 제공
- 로그 버퍼의 크기 한정
  - 새로운 내용이 기록되면 오래된 내용부터 덮어쓰기 됨
- 로그 버퍼에 저장된 내용을 보거나 지울 때는 logcat 명령 사용
- 로그 레벨(log level)에 따라 Log 클래스가 제공하는 주요 메서드

표 4-1 Log 클래스의 주요 메서드

메서드 이름	기능
wtf(String tag, String msg)	단언(assert) 메시지를 로그 버퍼에 보낸다. 참고 wtf=What a Terrible Failure
e(String tag, String msg)	에러(error) 메시지를 로그 버퍼에 보낸다.
w(String tag, String msg)	경고(warn) 메시지를 로그 버퍼에 보낸다.
i(String tag, String msg)	정보(info) 메시지를 로그 버퍼에 보낸다.
d(String tag, String msg)	디버그(debug) 메시지를 로그 버퍼에 보낸다.
v(String tag, String msg)	장황한(verbose) 메시지를 로그 버퍼에 보낸다.

01 이벤트 로깅과 logcat ▶ Log 클래스

실습 4-1 LogTest

- res/layout/activity_main.xml에 [표 4-1]의 메서드 개수와 똑같은 여섯 개의 버튼을 배치

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:orientation="vertical">
6       <Button
7           android:id="@+id/btnAssert"
8           android:layout_width="wrap_content"
9           android:layout_height="wrap_content"
10          android:onClick="mOnClick"
11          android:text="Assert"/>
```

01 이벤트 로깅과 logcat ▶ Log 클래스

실습 4-1 LogTest

- res/layout/activity_main.xml에 [표 4-1]의 메서드 개수와 똑같은 여섯 개의 버튼을 배치

```
12      <Button
13          android:id="@+id/btnError"
14          android:layout_width="wrap_content"
15          android:layout_height="wrap_content"
16          android:onClick="mOnClick"
17          android:text="Error"/>
18      <Button
19          android:id="@+id/btnWarn"
20          android:layout_width="wrap_content"
21          android:layout_height="wrap_content"
22          android:onClick="mOnClick"
23          android:text="Warn"/>
24      <Button
25          android:id="@+id/btnInfo"
26          android:layout_width="wrap_content"
27          android:layout_height="wrap_content"
28          android:onClick="mOnClick"
29          android:text="Info"/>
30      <Button
```

01 이벤트 로깅과 logcat ▶ Log 클래스

실습 4-1 LogTest

- res/layout/activity_main.xml에 [표 4-1]의 메서드 개수와 똑같은 여섯 개의 버튼을 배치

```
31          android:id="@+id/btnDebug"
32          android:layout_width="wrap_content"
33          android:layout_height="wrap_content"
34          android:onClick="mOnClick"
35          android:text="Debug"/>
36      <Button
37          android:id="@+id/btnVerbose"
38          android:layout_width="wrap_content"
39          android:layout_height="wrap_content"
40          android:onClick="mOnClick"
41          android:text="Verbose"/>
42  </LinearLayout>
```

01 이벤트 로깅과 logcat ▶ Log 클래스

실습 4-1 LogTest

- **MainActivity 클래스에 다음 코드 작성**

MainActivity.java

```
1   public class MainActivity extends AppCompatActivity {
2
3       private static final String TAG = "LogTest";
4
5       @Override
6       protected void onCreate(Bundle savedInstanceState) {
7           super.onCreate(savedInstanceState);
8           setContentView(R.layout.activity_main);
9       }
10
11      public void mOnClick(View v) {
12          switch (v.getId()) {
13          case R.id.btnAssert:
14              Log.wtf(TAG, "Assert message");
15              break;
16          case R.id.btnError:
17              Log.e(TAG, "Error message");
18              break;
19          case R.id.btnWarn:
20              Log.w(TAG, "Warn message");
```

실습 4-1 LogTest

- **MainActivity 클래스에 다음 코드 작성**

```
21              break;
22          case R.id.btnInfo:
23              Log.i(TAG, "Info message");
24              break;
25          case R.id.btnDebug:
26              Log.d(TAG, "Debug message");
27              break;
28          case R.id.btnVerbose:
29              Log.v(TAG, "Verbose message");
30              break;
31          }
32      }
33  }
```

01 이벤트 로깅과 logcat ▶ Log 클래스

실습 4-1 LogTest

- 앱 실행 후 각 버튼을 클릭
- 안드로이드 스튜디오가 제공하는 도구를 이용

그림 4-2 실행 화면

01 이벤트 로깅과 logcat ▶ Log 클래스

실습 4-1 LogTest

- [View]-[Tool Windows]-[Android Monitor] 메뉴 선택
- 혹은 단축키 Alt+6

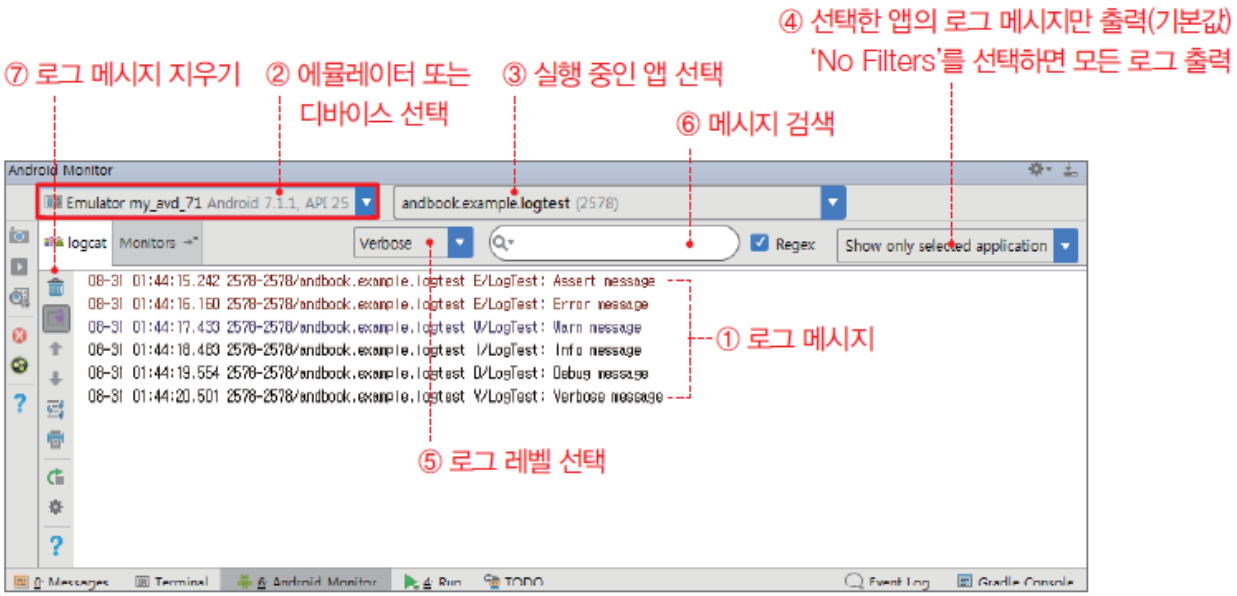


그림 4-3 Android Monitor 화면

실습 4-1 LogTest

Android Monitor[AS 2.3] / Logcat[AS 3.0] 설명

표 4-2 Android Monitor[AS 2.3] / Logcat[AS 3.0] 설명

번호	제목	설명
①	로그 메시지	[날짜 시각] [프로세스ID-스레드ID/] [패키지명] [로그 레벨/태그] [:로그 메시지] 형식으로 로그 메시지가 출력된다. 여기서 '로그 레벨'은 E(Error), W(Warning)처럼 한 글자로 표시된다.
②	에뮬레이터 또는 디바이스 선택	특정 에뮬레이터나 디바이스를 반드시 선택해야 한다. 대부분은 그대로 두면 된다.
③	실행 중인 앱 선택	실행 중인 앱을 반드시 선택해야 한다. 대부분은 그대로 두면 된다.
④	선택한 앱의 로그 메시지만 출력(기본값)'No Filters'를 선택하면 모든 로그 출력	③에서 선택된 앱의 로그 메시지만 보게 되어 있지만(기본값), 'No Filters'를 선택하여 안드로이드 시스템에서 발생하는 모든 메시지를 볼 수도 있다. TIP_ 테스트 도중 앱에서 문제가 발생했지만 로그 메시지로 알 수 없다면 반드시 'No Filters'를 선택해서 전체 메시지를 살펴봐야 한다.
⑤	로그 레벨 선택	로그 레벨을 선택하면 그것과 같거나 더 중요한 로그 메시지만 표시된다.
⑥	메시지 검색	글자를 입력하면 그 글자를 포함하는 로그 메시지만 표시된다.
⑦	로그 메시지 지우기	이전에 발생한 로그 메시지가 많아서 유용한 정보를 찾아내기 힘들다면 테스트에 앞서 기존 로그 메시지를 한꺼번에 지울 수 있다.

실습 4-2 PrintStreamTest

- **System.out과 System.err 객체로 출력하는 앱 작성**
  - 1장의 Hello 예제처럼 프로젝트를 생성하되 이름은 PrintStreamTest
  - res/layout/activity_main.xml에 버튼 두 개 배치
  - MainActivity 클래스에 다음 코드 작성

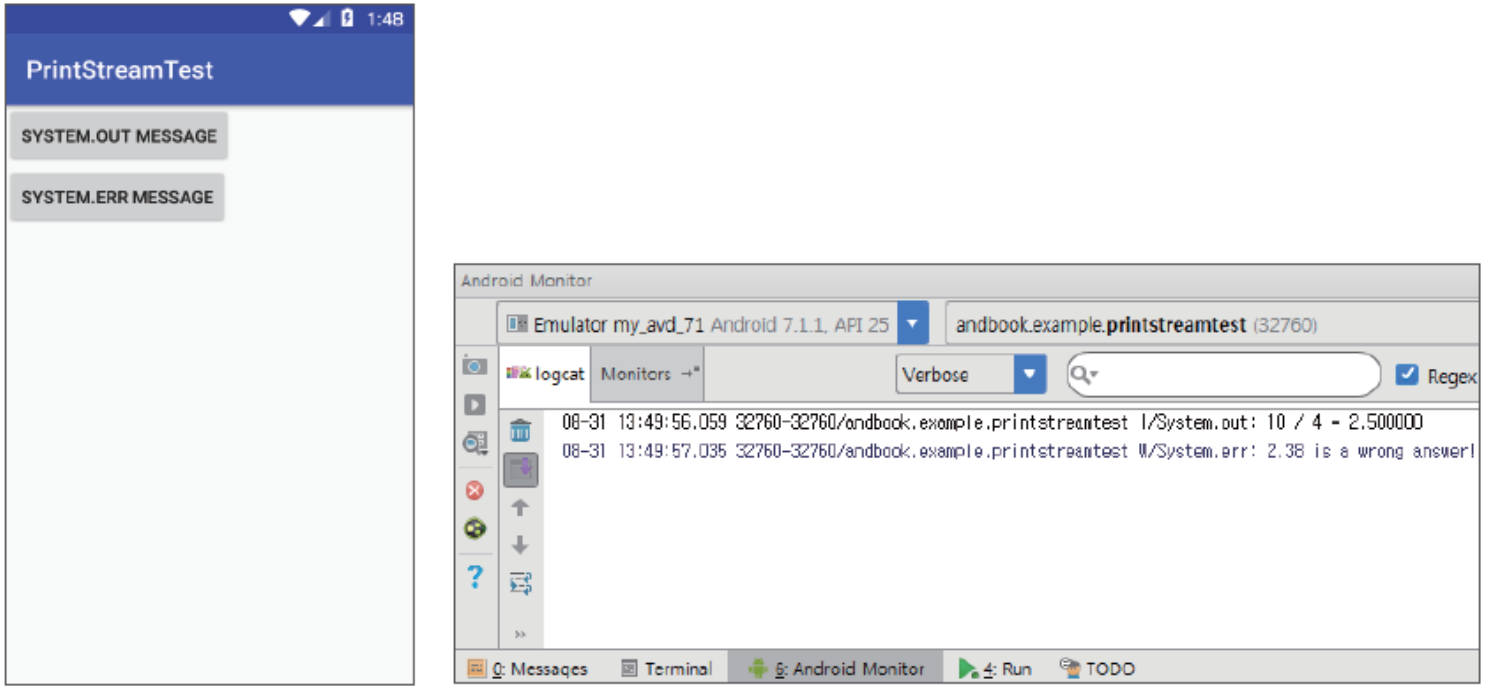
MainActivity.java

```
1   public class MainActivity extends AppCompatActivity {
2
3       @Override
4       protected void onCreate(Bundle savedInstanceState) {
5           super.onCreate(savedInstanceState);
6           setContentView(R.layout.activity_main);
7       }
8
9       public void mOnClick(View v) {
10          switch (v.getId()) {
11          case R.id.btnOut:
12              System.out.printf("%d / %d = %f%n", 10, 4, 10 / 4.0);
13              break;
14          case R.id.btnErr:
15              System.err.printf("%.2f is a wrong answer!%n", 2.378);
16              break;
17          }
18      }
19  }
```

01 이벤트 로깅과 logcat ▶ System.out과 System.err

실습 4-2 PrintStreamTest

- 초기 화면과 버튼 클릭 시 Android Monitor에 출력되는 로그 메시지



(a) 초기 화면 (b) Android Monitor

그림 4-4 실행 화면

02 액티비티 생명주기

- 액티비티는 생성에서 종료까지 여러 상태를 거침
- 이런 상태 전체를 생명주기life cycle라 부름

- 안드로이드 시스템은 액티비티의 상태를 크게 세 종류로 나누어 관리

표 4-3 액티비티 상태

상태	설명
실행 재개Resumed	액티비티가 화면 맨 앞에 있으며 사용자와 상호작용하고 있다.
일시 정지Paused	다른 액티비티가 부분만 가리거나 반투명 액티비티가 부분 혹은 전체를 가려서 현재 액티비티가 보이는 상태이다. 시스템의 가용 메모리가 매우 부족하면 안드로이드 시스템은 일시 정지 상태의 액티비티를 강제로 종료시킬 수 있다.
중단Stopped	다른 액티비티가 완전히 가려서 현재 액티비티가 보이지 않는 상태이다. 시스템의 가용 메모리가 부족하면 안드로이드 시스템은 중단 상태의 액티비티를 강제로 종료시킬 수 있다.

02 액티비티 생명주기

콜백 메서드(callback method)

- 액티비티의 상태가 변할 때마다 액티비티 클래스의 약속된 메서드 호출

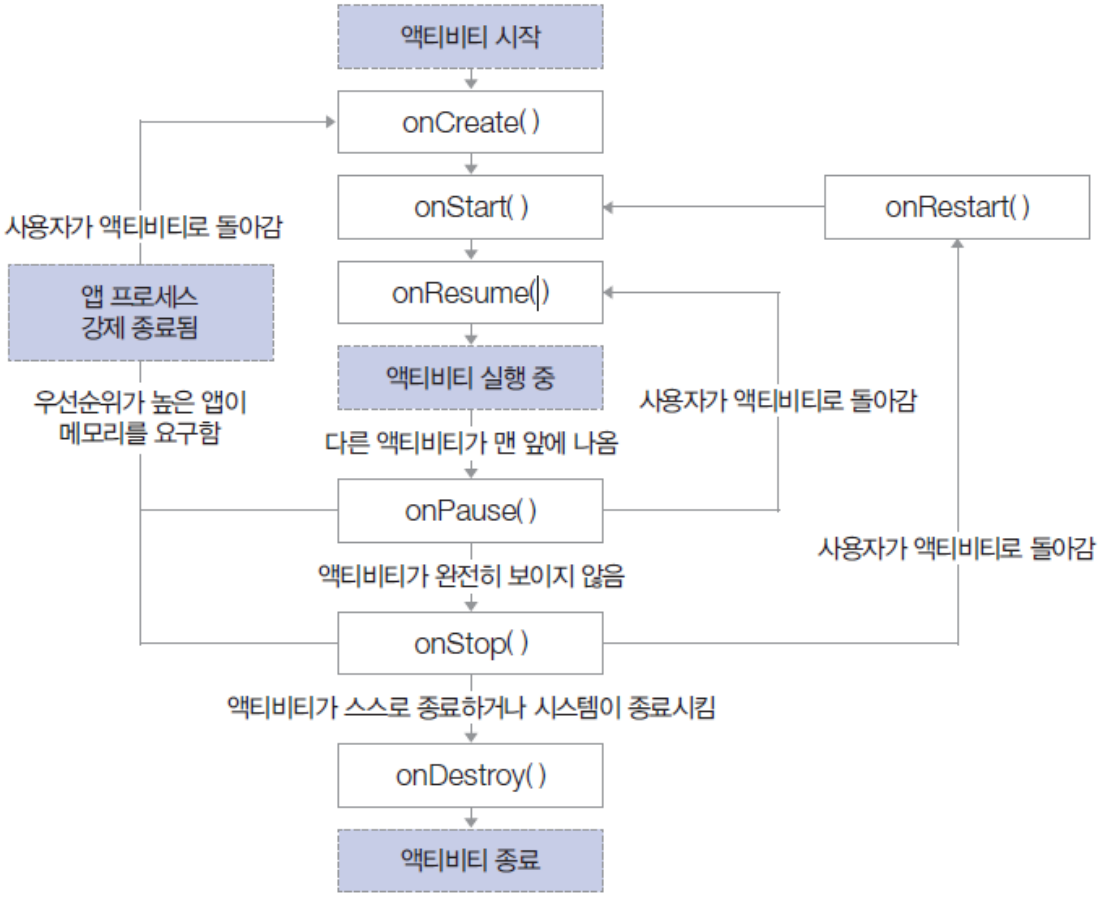


그림 4-5 액티비티의 생명주기와 콜백 메서드

02 액티비티 생명주기

- 액티비티의 생명주기는 크게 세 개의 루프로 나눔
- 응용 프로그램은 안드로이드 시스템이 자동으로 호출해주는 콜백 메서드를 정의함으로써 생명주기에 따라 적절한 작업 가능
- onCreate( )에서 각종 초기화 코드를 수행하고 onDestroy( )에서 각종 정리 코드를 수행 가능

표 4-4 액티비티의 생명주기 루프와 콜백 메서드

분류	구간
전체 생애entire lifetime	onCreate() ~ onDestroy()
가시 생애visible lifetime	onStart() ~ onStop()
전면 생애foreground lifetime	onResume() ~ onPause()

02 액티비티 생명주기

실습 4-3 ActivityCycle

- **MainActivity 클래스에 다음 코드를 작성**

MainActivity.java

```
1   public class MainActivity extends AppCompatActivity {
2
3       private static final String TAG = "ActivityCycle";
4
5       @Override
6       protected void onCreate(Bundle savedInstanceState) {
7           Log.d(TAG, "onCreate() 호출!");
8           super.onCreate(savedInstanceState);
9           setContentView(R.layout.activity_main);
10      }
11
12      @Override
13      protected void onDestroy() {
14          Log.d(TAG, "onDestroy() 호출!");
15          super.onDestroy();
16      }
17
18      @Override
19      protected void onStart() {
20          Log.d(TAG, "onStart() 호출!");
21          super.onStart();
22      }
23
```

02 액티비티 생명주기

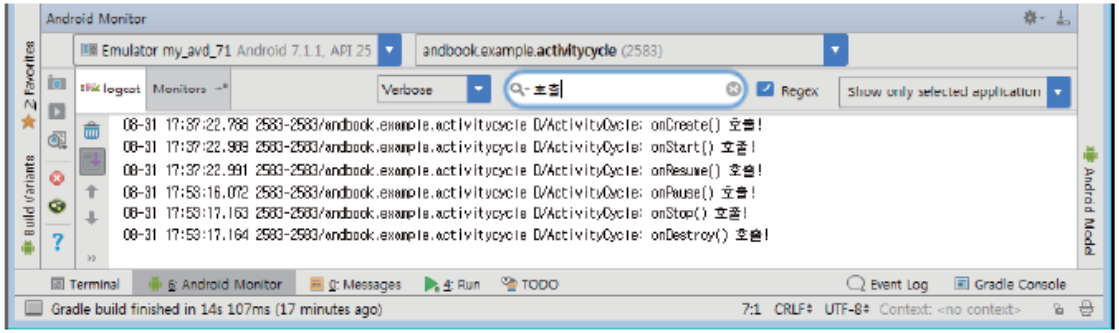
실습 4-3 ActivityCycle

- **MainActivity 클래스에 다음 코드를 작성**

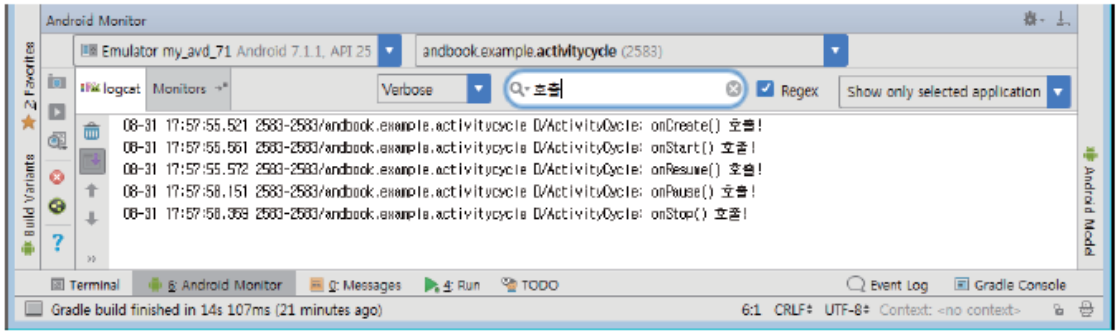
```
24      @Override
25      protected void onStop() {
26          Log.d(TAG, "onStop() 호출!");
27          super.onStop();
28      }
29
30      @Override
31      protected void onResume() {
32          Log.d(TAG, "onResume() 호출!");
33          super.onResume();
34      }
35
36      @Override
37      protected void onPause() {
38          Log.d(TAG, "onPause() 호출!");
39          super.onPause();
40      }
41  }
```

실습 4-3 ActivityCycle

- **앱을 설치한 후 테스트한 결과**



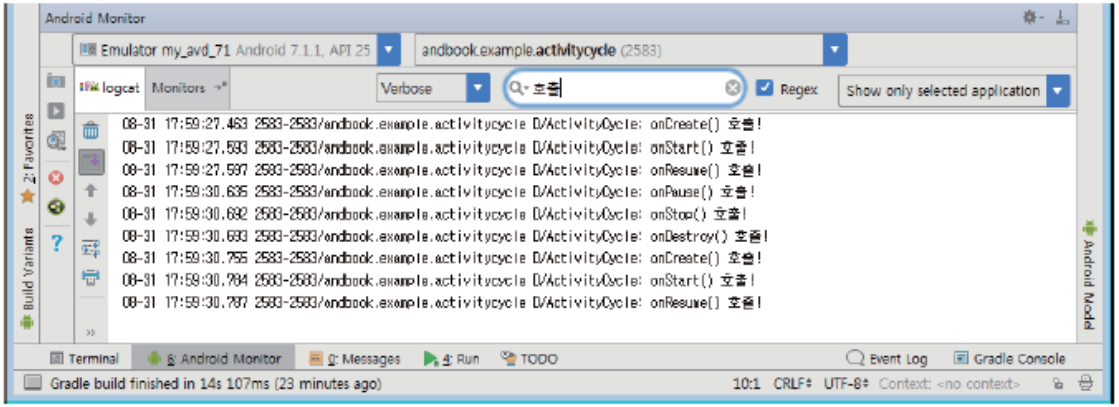
(a) 앱 실행 후 백 버튼 누르기



(b) 앱 실행 후 홈 버튼 누르기

실습 4-3 ActivityCycle

- **앱을 설치한 후 테스트한 결과**



(c) 앱 실행 후 화면 회전하기

그림 4-6 테스트 결과

03 액티비티 상태 유지 ▶ 액티비티 재생성 상황

안드로이드 시스템이 액티비티를 자동으로 재생성하는 대표적인 상황

- 기기를 회전하여 화면 방향을 전환할 때
- 시스템의 언어를 변경할 때
- 시스템의 글꼴 크기를 변경할 때
- 시스템의 디스플레이 크기를 변경할 때(안드로이드 7.0 이상)
- 안드로이드 디바이스의 출력을 TV와 같은 외부 디스플레이로 내보낼 때
- 안드로이드 디바이스를 도킹 장치에 연결하거나 제거할 때
- 외장 키보드를 연결할 때
- 슬라이딩 자판이 내장된 안드로이드 디바이스에서 자판을 꺼내거나 집어넣을 때
- 기기가 켜진 상태에서 SIM 카드를 교체하여 통신사를 변경할 때

03 액티비티 상태 유지 ▶ 액티비티 재생성 상황

실습 4-4 ActivityRecreate

- res/layout/activity_main.xml에 EditText와 RadioButton 배치

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:orientation="vertical">
6       <EditText
7           android:id="@+id/edit1"
8           android:layout_width="match_parent"
9           android:layout_height="wrap_content"
10          android:hint="id가 있는 경우"/>
11      <EditText
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:hint="id가 없는 경우"/>
```

03 액티비티 상태 유지 ▶ 액티비티 재생성 상황

실습 4-4 ActivityRecreate

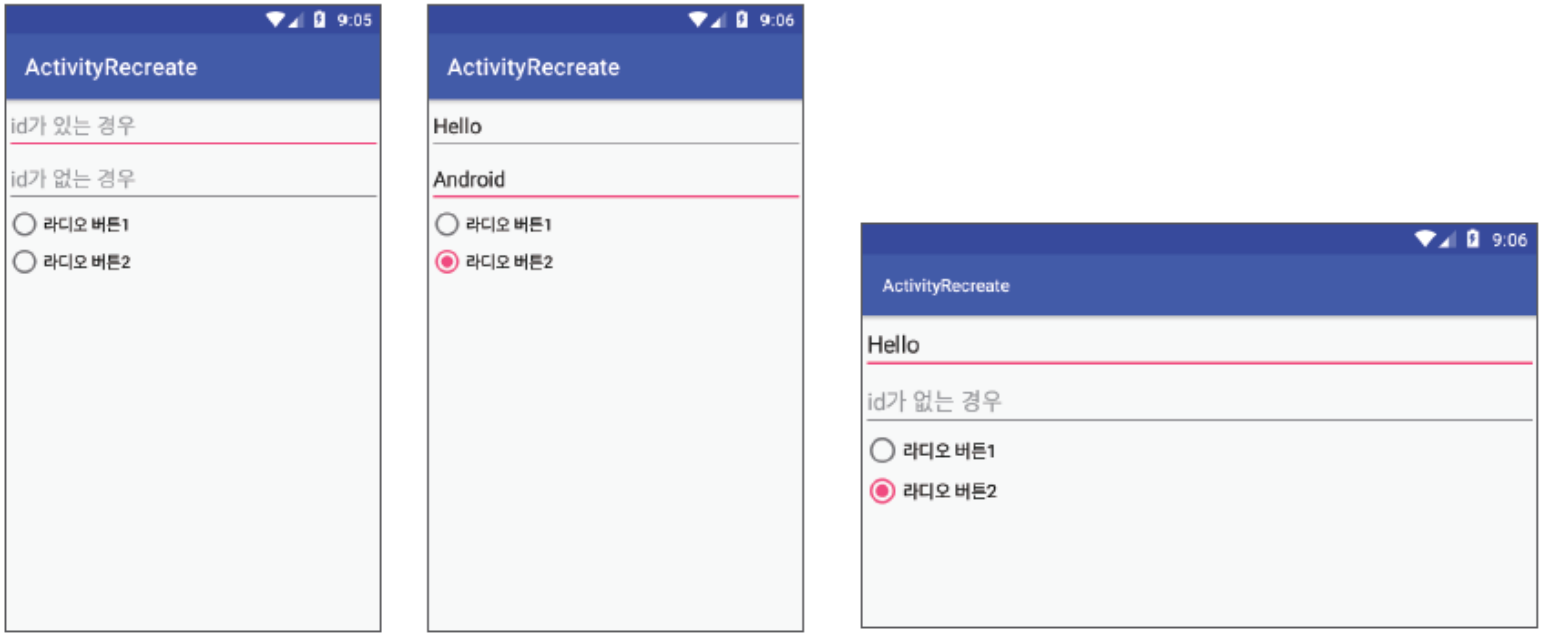
- res/layout/activity_main.xml에 EditText와 RadioButton 배치

```
15      <RadioGroup
16          android:layout_width="wrap_content"
17          android:layout_height="wrap_content">
18          <RadioButton
19              android:id="@+id/radio1"
20              android:layout_width="wrap_content"
21              android:layout_height="wrap_content"
22              android:text="라디오 버튼1"/>
23          <RadioButton
24              android:id="@+id/radio2"
25              android:layout_width="wrap_content"
26              android:layout_height="wrap_content"
27              android:text="라디오 버튼2"/>
28      </RadioGroup>
29  </LinearLayout>
```

03 액티비티 상태 유지 ▶ 액티비티 재생성 상황

실습 4-4 ActivityRecreate

실행 화면



(a) 초기 화면 (b) 텍스트 입력, 라디오 버튼 선택 (c) 화면 회전

그림 4-7 실행 화면

- 안드로이드 시스템은 액티비티가 재생성되는 상황에서 상태를 저장하고 복원할 수 있도록 다음과 같은 코드 구조 제공

```
1   public class MainActivity extends AppCompatActivity {
2
3       @Override
4       protected void onCreate(Bundle savedInstanceState) {
5           super.onCreate(savedInstanceState);
6           setContentView(R.layout.activity_main);
7
8           if (savedInstanceState != null) {
9               // savedInstanceState 객체에 대해 get~() 메서드를
10              // 호출하여 데이터를 읽는다.
11          }
12      }
13
14      @Override
15      protected void onSaveInstanceState(Bundle outState) {
16          super.onSaveInstanceState(outState);
17          // outState 객체에 대해 put~() 메서드를
18          // 호출하여 데이터를 저장한다.
19      }
20  }
```

03 액티비티 상태 유지 ▶ 액티비티 재생성 상황

Bundle 클래스는 자바 언어의 기본형과 배열을 모두 지원

표 4-5 Bundle 클래스의 주요 메서드

종류	메서드

데이터 저장	void **putBoolean**(String key, boolean value) // 자바 기본형
	void **putBooleanArray**(String key, boolean[] value) // 자바 기본형의 배열
	void **putByte**(String key, byte value)
	void **putByteArray**(String key, byte[] value)
	void **putChar**(String key, char value)
	void **putCharArray**(String key, char[] value)
	void **putShort**(String key, short value)
	void **putShortArray**(String key, short[] value)
	void **putInt**(String key, int value) // 자바 기본형
	void **putIntArray**(String key, int[] value) // 자바 기본형의 배열
	void **putLong**(String key, long value)
	void **putLongArray**(String key, long[] value)
	void **putFloat**(String key, float value)
	void **putFloatArray**(String key, float[] value)
	void **putDouble**(String key, double value)
	void **putDoubleArray**(String key, double[] value)
	void **putString**(String key, String value) // 자바 문자열
	void **putStringArray**(String key, String[] value) // 자바 문자열의 배열

데이터 읽기	boolean **getBoolean**(String key) // 자바 기본형
	boolean **getBoolean**(String key, boolean defaultValue) // 자바 기본형; 기본값 지정
	boolean[] **getBooleanArray**(String key) // 자바 기본형의 배열
	…
	int **getInt**(String key) // 자바 기본형
	int **getInt**(String key, int defaultValue) // 자바 기본형; 기본값 지정
	int[] **getIntArray**(String key) // 자바 기본형의 배열
	…
	String **getString**(String key) // 자바 기본형
	String **getString**(String key, String defaultValue) // 자바 문자열; 기본값 지정
	String[] **getStringArray**(String key) // 자바 문자열의 배열

03 액티비티 상태 유지 ▶ 액티비티 재생성 상황

실습 4-5 SaveState

- onCreate( )와 onSaveInstanceState ( ) 조합하여 액티비티의 상태 유지

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="wrap_content">
5       <Button
6           android:id="@+id/btnIncrease"
7           android:layout_width="wrap_content"
8           android:layout_height="wrap_content"
9           android:onClick="mOnClick"
10          android:text="숫자 증가"/>
11      <TextView
12          android:id="@+id/textNumber"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:layout_gravity="center_vertical"
16          android:layout_marginLeft="8dp"
17          android:textSize="24dp"/>
18  </LinearLayout>
```

03 액티비티 상태 유지 ▶ 액티비티 재생성 상황

실습 4-5 SaveState

- MainActivity 클래스에 다음 코드 작성

MainActivity.java

```
1   public class MainActivity extends AppCompatActivity {
2
3       private int mNumber;
4       private TextView mTextNumber;
5
6       @Override
7       protected void onCreate(Bundle savedInstanceState) {
8           super.onCreate(savedInstanceState);
9           setContentView(R.layout.activity_main);
10
11          if (savedInstanceState != null) {
12              mNumber = savedInstanceState.getInt("number", 0);
13          }
```

03 액티비티 상태 유지 ▶ 액티비티 재생성 상황

실습 4-5 SaveState

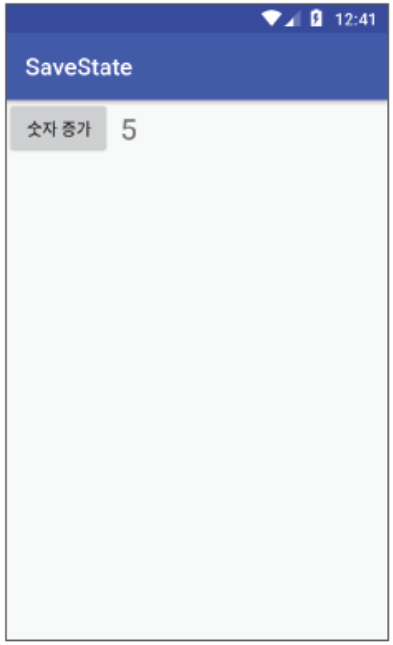
- MainActivity 클래스에 다음 코드 작성

```
14
15          mTextNumber = (TextView) findViewById(R.id.textNumber);
16          mTextNumber.setText(mNumber + "");
17      }
18
19      @Override
20      protected void onSaveInstanceState(Bundle outState) {
21          super.onSaveInstanceState(outState);
22          outState.putInt("number", mNumber);
23      }
24
25      public void mOnClick(View v) {
26          mNumber++;
27          mTextNumber.setText(mNumber + "");
28      }
29  }
```

실습 4-5 SaveState

실행 화면

(a) 초기 화면



(b) 버튼을 다섯 번 클릭

(c) 화면 회전

(d) 화면 회전(자바 코드 주석 처리 시)

04 대체 리소스 정의 ▶ 기본 리소스

리소스 중 res 폴더에 있는 것

표 4-6 리소스 종류

리소스	설명	최초 소개 위치
애니메이션 (animation)	애니메이션 효과를 정의한다.	7장
컬러 상태 목록 (color state list)	뷰가 상태에 따라 서로 다른 색상을 가지게 한다.	없음
비트맵 드로어블 (bitmap drawable)	그래픽 파일(예: *.png)이다.	2장
나인-패치 드로어블 (9-patch drawable)	자연스런 크기 조절이 가능한 그래픽 파일이다.	4장
XML 드로어블 (XML drawable)	그래픽 파일을 조합해서 효과를 내거나 간단한 도형을 정의한다.	없음
레이아웃 (layout)	응용 프로그램 UI를 정의한다.	1장
메뉴 (menu)	응용 프로그램 메뉴를 정의한다.	6장
문자열 (string)	문자열을 정의한다.	4장
문자열 배열 (string array)	문자열 배열을 정의한다.	7장
스타일 (style)	UI 요소들의 외양을 정의한다.	7장
기타 리소스	색상, 치수, ID, 정수 등 다양한 값을 정의한다.	7장

비트맵 드로어블(bitmap drawable)

- 그래픽 파일을 나타내는 리소스
- 지원되는 확장자로는 .png(권장) .jpg (가능) .gif (비권장)
- 비트맵 드로어블은 자바 코드에서 BitmapDrawable 객체로 표현

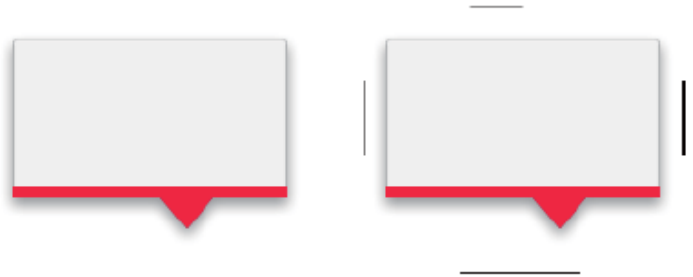
```
<ImageView
    android:id="@+id/img"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/test"/>
```

- res/drawable/test.png 파일을 자바 코드에서 사용하는 예

```
// 방법1
ImageView iv1 = (ImageView) findViewById(R.id.img);
iv1.setImageResource(R.drawable.test);
// 방법2
ImageView iv2 = (ImageView) findViewById(R.id.img);
BitmapDrawable drawable = (BitmapDrawable) getResources().getDrawable(R.drawable.test);
iv2.setImageDrawable(drawable);
```

나인-패치 드로어블(9-patch drawable)

- 특정 영역만 늘어나거나 줄어드는 그래픽 파일
- 뷰의 배경 그림으로 사용하기에 적합
- 나인-패치는 일반 PNG 이미지의 외곽에 1픽셀 두께의 테두리를 추가한 것
- 안드로이드 스튜디오에 내장된 기능을 사용하면 일반 PNG 파일을 읽어서 나인-패치를 생성



(a) 일반 PNG 이미지 (b) 나인-패치

그림 4-9 일반 PNG 이미지와 나인-패치

04 대체 리소스 정의 ▶ 기본 리소스

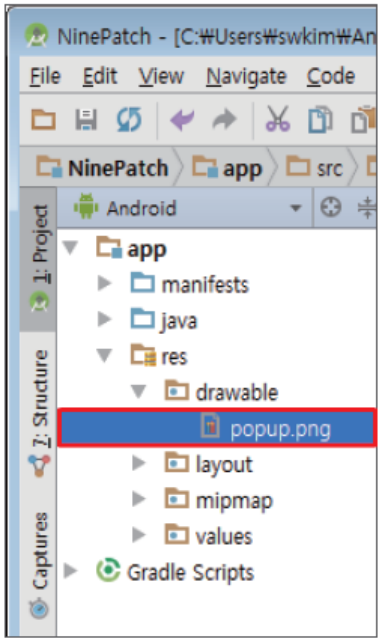
- **왼쪽과 위쪽의 검은 선**
  - 나인-패치의 크기를 변경할 때 늘어나거나 줄어들 세로와 가로 영역을 지정
- **오른쪽과 아래쪽의 검은 선**
  - 나인-패치를 뷰의 배경으로 사용할 때 뷰의 내용이 표시될 세로와 가로 영역 지정
- **오른쪽과 아래쪽의 검은 선이 없을 때**
  - 왼쪽과 위쪽의 검은 선이 역할을 대신함

그림 4-10 나인-패치 영역 설명

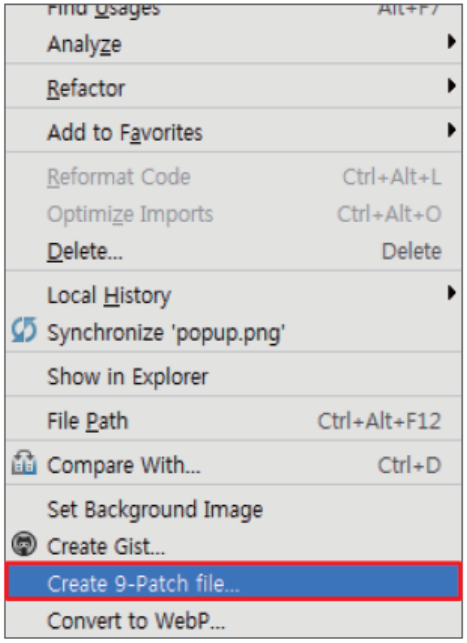
실습 4-6 NinePatch

- 나인-패치를 직접 정의하고 텍스트뷰의 배경으로 사용하는 앱 작성

(a) popup.png 파일



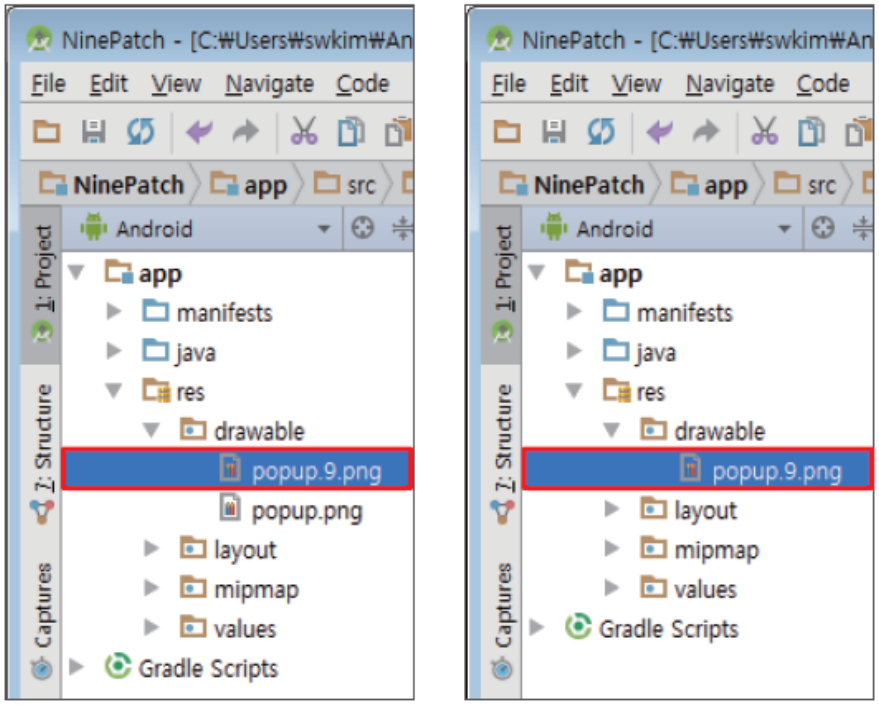
(b) 프로젝트의 res/drawable 폴더



(c) [Create 9-Patch file…] 메뉴 선택

실습 4-6 NinePatch

- 나인-패치를 직접 정의하고 텍스트뷰의 배경으로 사용하는 앱 작성

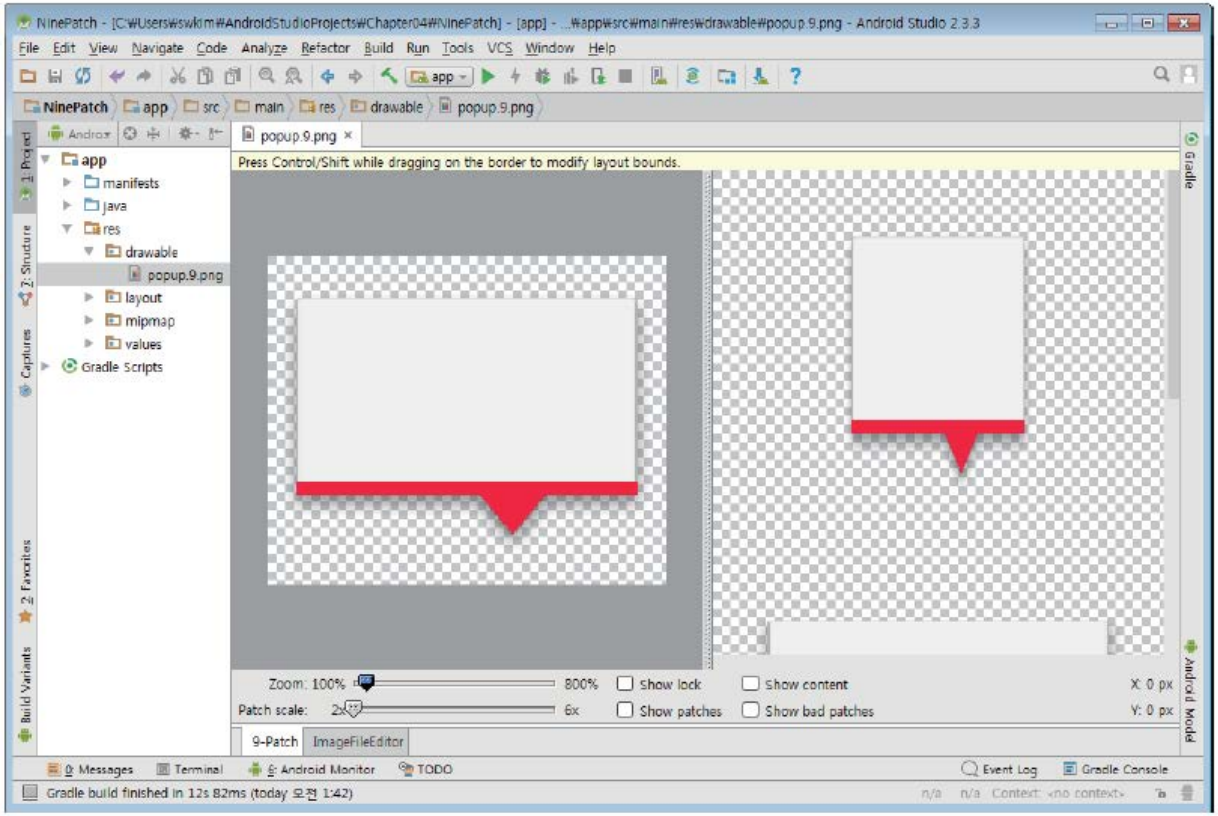


(d) 생성된 popup.9.png (e) popup.png 삭제 후

그림 4-11 일반 PNG 이미지로 나인-패치 생성

실습 4-6 NinePatch

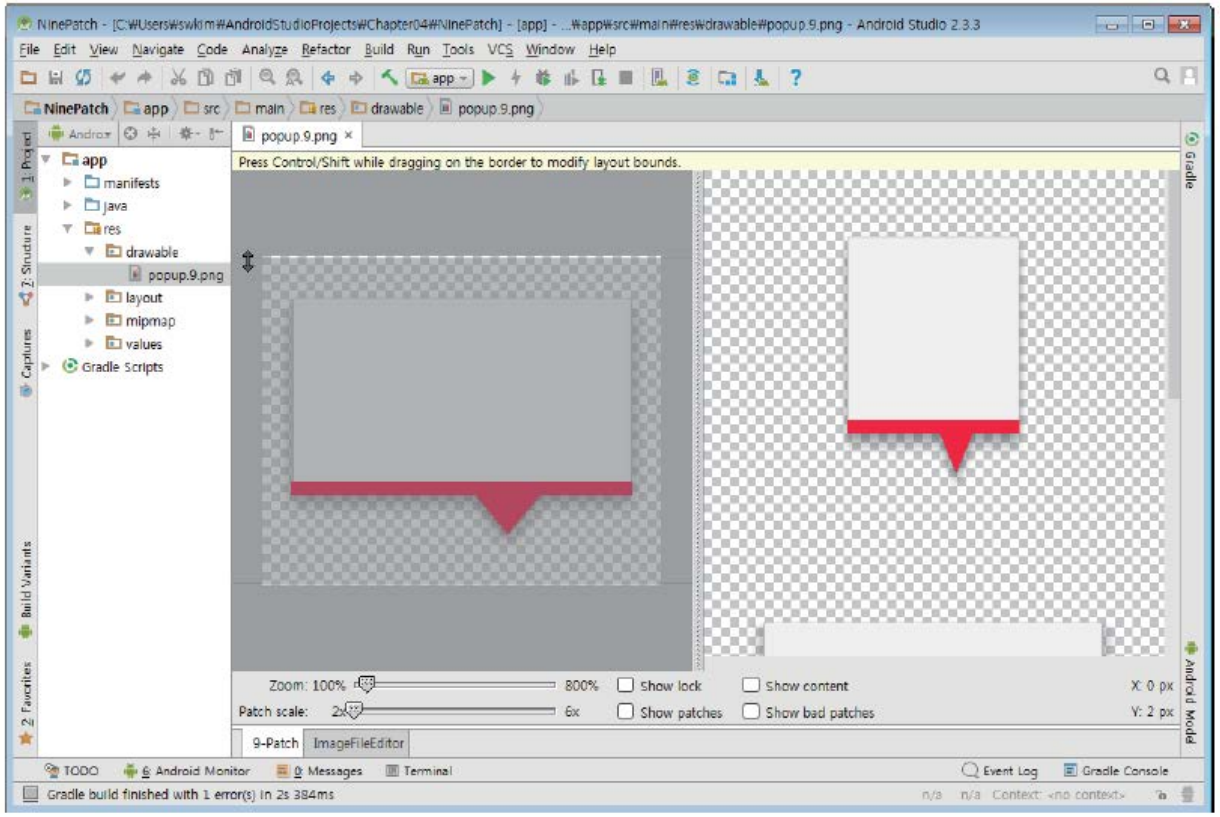
초기 화면



(a) 초기 화면

실습 4-6 NinePatch

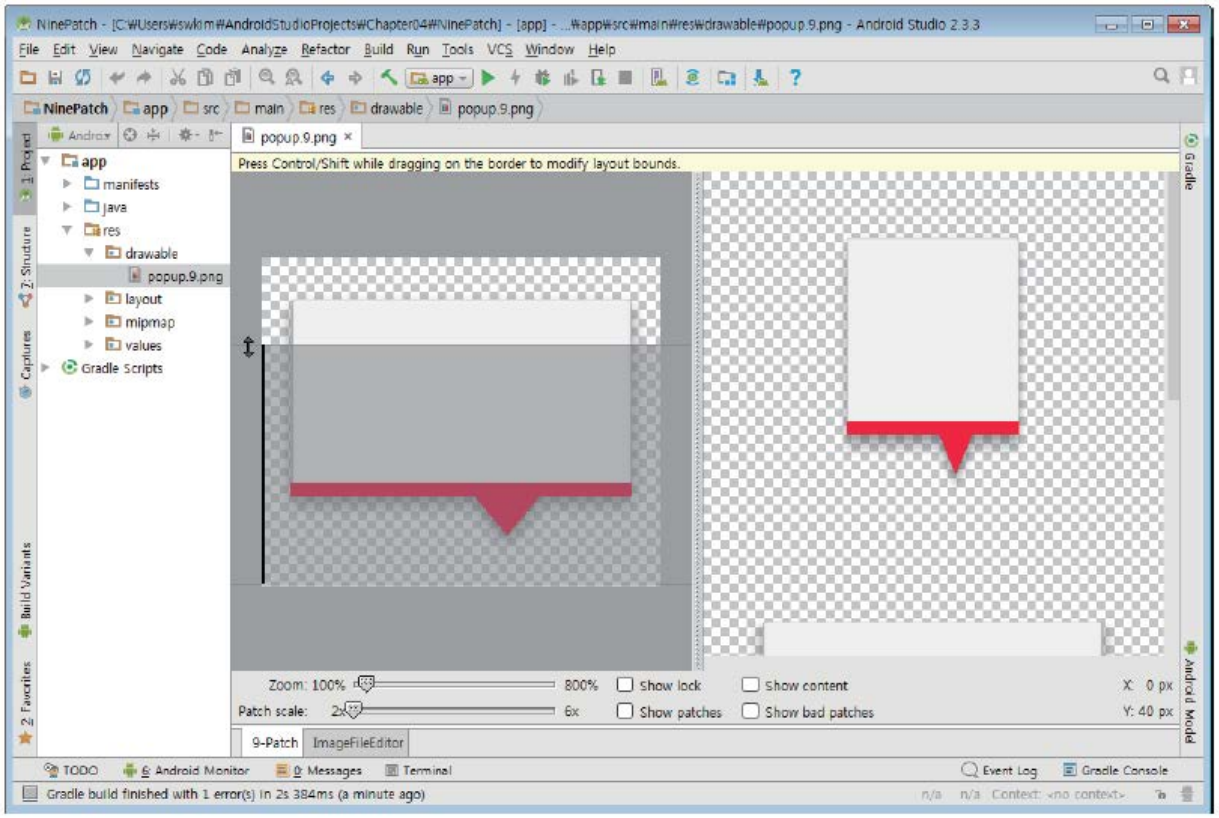
- 왼쪽 빈 공간에 마우스를 가져다 대면 커서 모양 바뀜



(b) 테두리 그리기 직전

실습 4-6 NinePatch

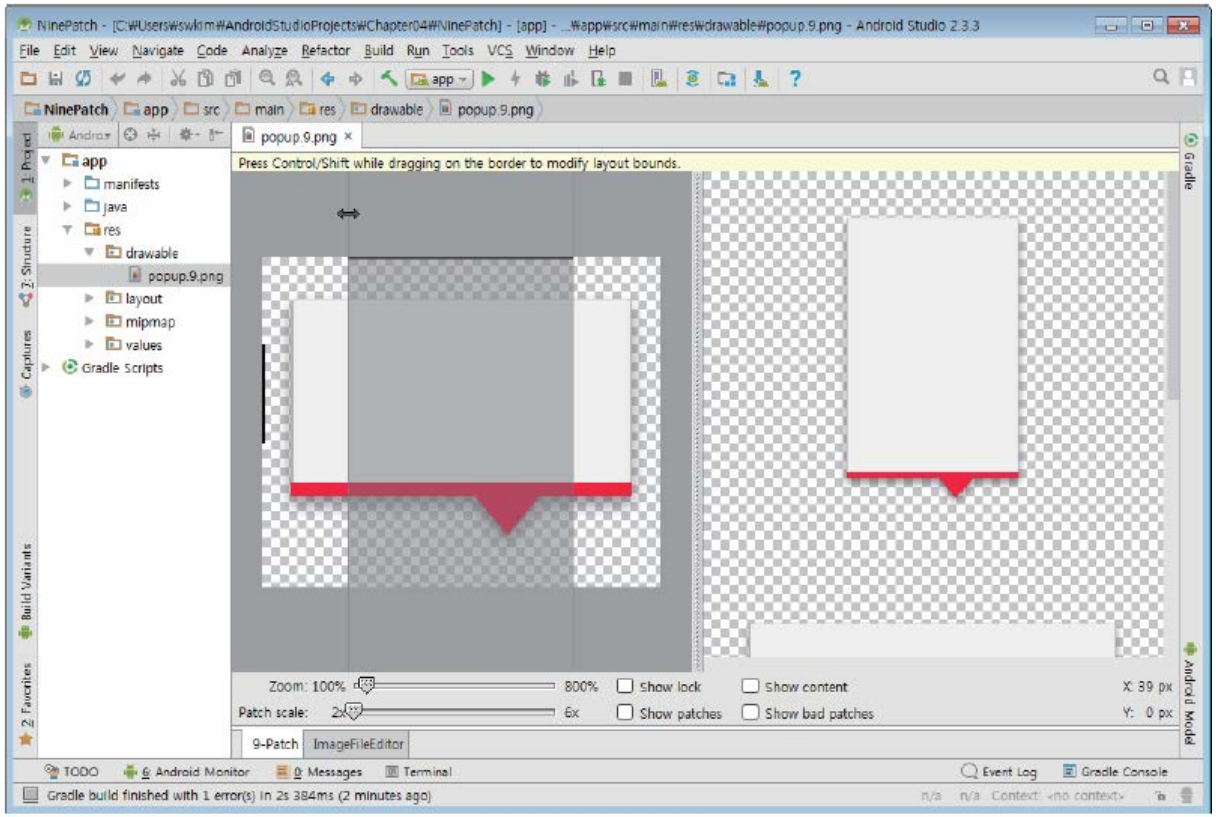
- 마우스 왼쪽 버튼을 클릭한 상태에서 움직여 테두리 그리기



(c) 테두리 그리기 시작

실습 4-6 NinePatch

- 왼쪽과 위쪽에 테두리를 그린 상태



(d) 왼쪽과 위쪽 테두리

그림 4-12 나인-패치 편집

실습 4-6 NinePatch

res/layout/activity_main.xml 수정

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       android:layout_width="match_parent"
5       android:layout_height="match_parent"
6       android:background="#ffffff"
7       android:orientation="vertical">
8       <TextView
9           android:layout_width="wrap_content"
10          android:layout_height="wrap_content"
11          android:background="@drawable/popup"
12          android:text="안녕하세요.\n정말 반가워요.\n그동안 잘 지냈어요?\n만나서 식사할까요?"
13          android:textSize="18dp"/>
14  </LinearLayout>
```

실습 4-6 NinePatch

실행 화면

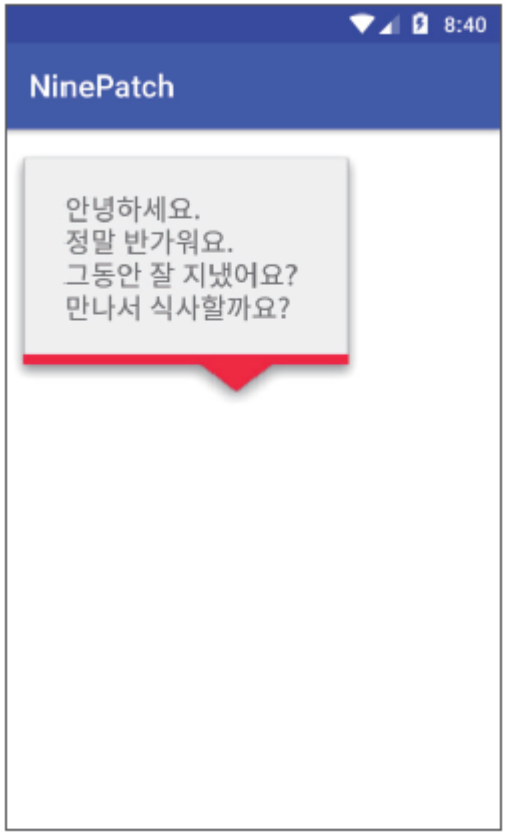


그림 4-13 실행 화면

04 대체 리소스 정의 ▶ 기본 리소스

- 문자열 리소스는 res/values/파일명.xml 파일에 정의

- 문자열 리소스를 res/values/strings.xml 파일에 정의한 것

```
<resources>
    <string name="app_name">StringTest</string>
    <string name="msg1">Hello, Android!</string>
    <string name="msg2">I\'ll be back.</string>
    <string name="msg3">This is a \"good thing\".</string>
    <string name="msg4"><b>Think</b> like a man of <i>action</i>
        <u>and</u> <b>act</b> like man of <i>thought</i>.</string>
</resources>
```

- 문자열 리소스를 XML에서 참조할 때는 @string/리소스명 형식 사용

04 대체 리소스 정의 ▶ 기본 리소스

- 문자열 리소스를 레이아웃에서 사용하는 예

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/msg1"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/msg2"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/msg3"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/msg4"/>
</LinearLayout>
```

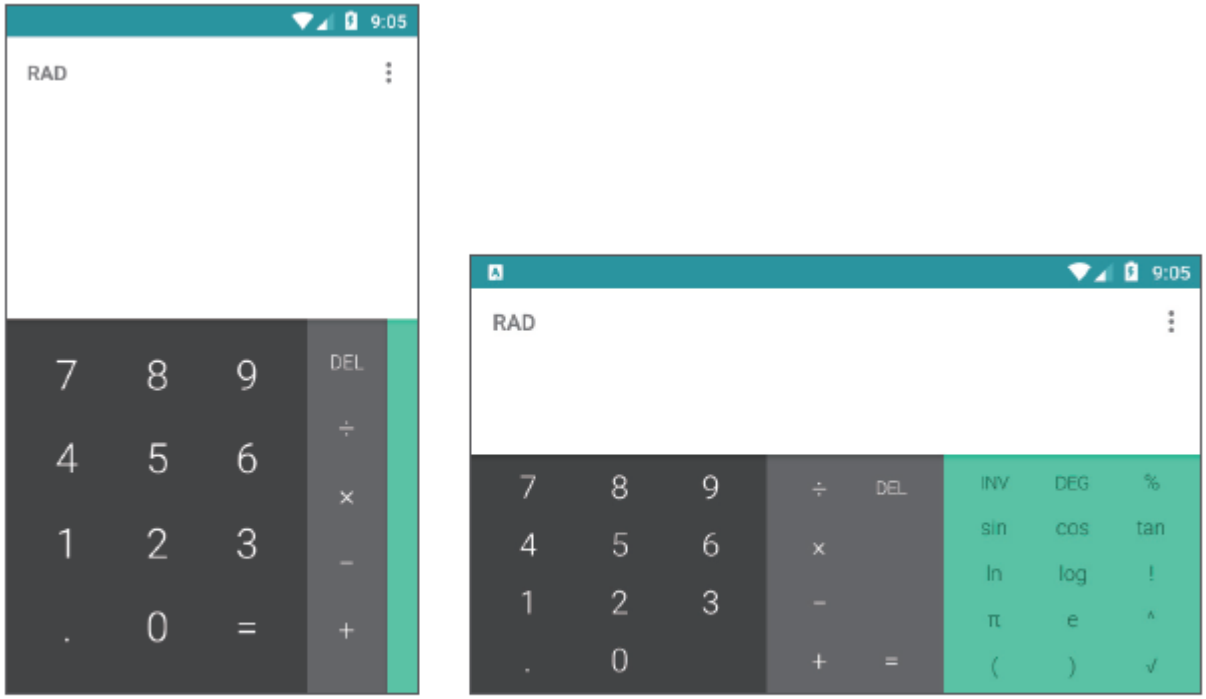
04 대체 리소스 정의 ▶ 기본 리소스

- 문자열 리소스를 자바 코드에서 Toast 메시지로 표시하는 예

```
// 방법1
Toast.makeText(this, R.string.msg4, Toast.LENGTH_LONG).show();
// 방법2
String str1 = getResources().getString(R.string.msg4);
Toast.makeText(this, str1, Toast.LENGTH_LONG).show();
// 방법3
CharSequence str2 = getResources().getText(R.string.msg4);
Toast.makeText(this, str2, Toast.LENGTH_LONG).show();
```

- 안드로이드 앱은 시스템의 구성 변화에 따라 자동으로 반응

- 화면을 회전하면 레이아웃이 달라짐



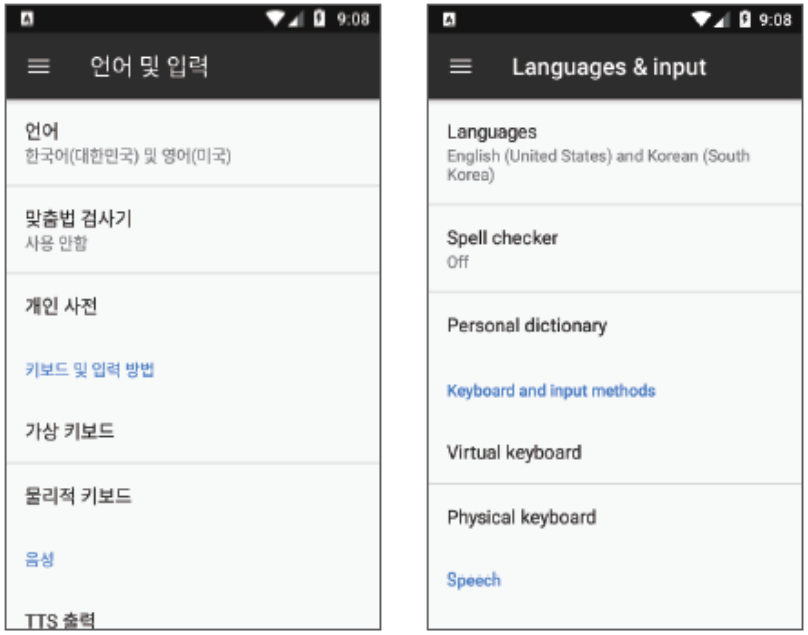
(a) 세로 방향 (b) 가로 방향

그림 4-14 화면 회전에 따른 레이아웃 변경

04 대체 리소스 정의 ▶ 대체 리소스

- 안드로이드의 '설정' 앱 자체도 시스템의 구성 변화에 자동 반응

- 시스템 언어를 변경하면 표시되는 언어도 그에 따라 바뀜



(a) 한국어 (b) 영어

그림 4-15 시스템 언어 변경에 따른 표시 언어 변경

04 대체 리소스 정의 ▶ 대체 리소스

실습 4-7 AlternativeRsrc

자동으로 화면 구성을 변경하는 앱 작성

- 1장의 Hello 예제처럼 프로젝트를 생성하되 이름은 AlternativeRsrc
- testpic이라는 이름의 그림 파일(확장자는 .png 또는 .jpg 권장)을 준비
- 윈도우즈 탐색기에서 Ctrl+C로 복사한 후 AlternativeRsrc 프로젝트의 res/drawable 폴더에 Ctrl+V로 붙여넣기

- res/values/strings.xml 파일에 다음 한 줄을 추가

strings.xml

```
1   <resources>
2       <string name="app_name">AlternativeRsrc</string>
3       <string name="pic_text">F-22 Raptor is a fifth-generation stealth fighter aircraft.</string>
4   </resources>
```

실습 4-7 AlternativeRsrc

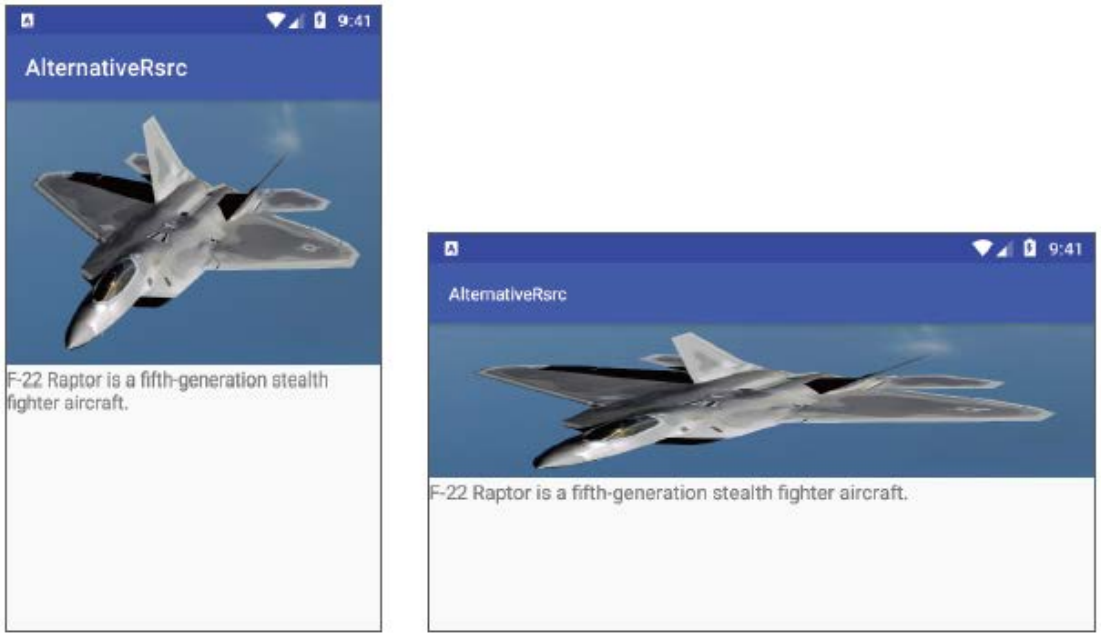
res/layout/activity_main.xml 파일 수정

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:orientation="vertical">
6       <ImageView
7           android:layout_width="match_parent"
8           android:layout_height="0dp"
9           android:layout_weight="1"
10          android:scaleType="fitXY"
11          android:src="@drawable/testpic"/>
12      <TextView
13          android:layout_width="match_parent"
14          android:layout_height="0dp"
15          android:layout_weight="1"
16          android:text="@string/pic_text"
17          android:textSize="16dp"/>
18  </LinearLayout>
```

실습 4-7 AlternativeRsrc

실행 화면

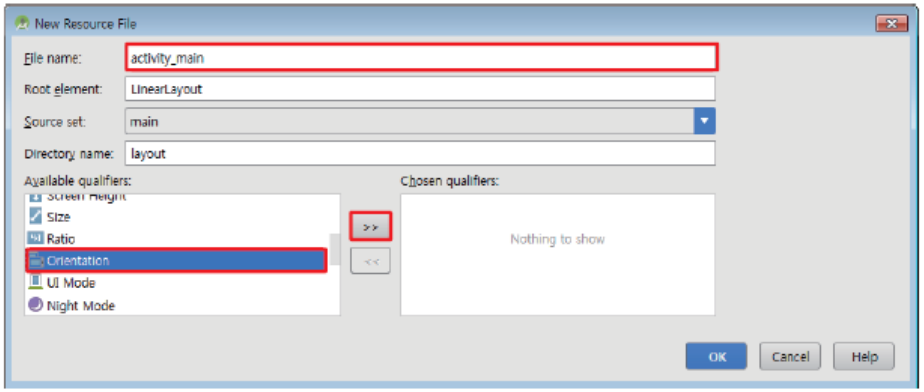


(a) 세로 방향 (b) 가로 방향

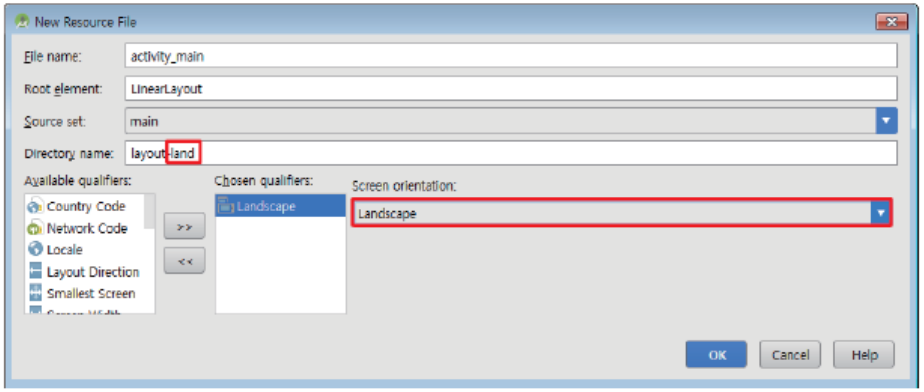
그림 4-16 실행 화면

실습 4-7 AlternativeRsrc

activity_main.xml 생성



(a) 파일 이름 입력 후 Orientation 추가



(b) Orientation 기본값 변경

그림 4-17 가로 방향 레이아웃 리소스 생성

실습 4-7 AlternativeRsrc

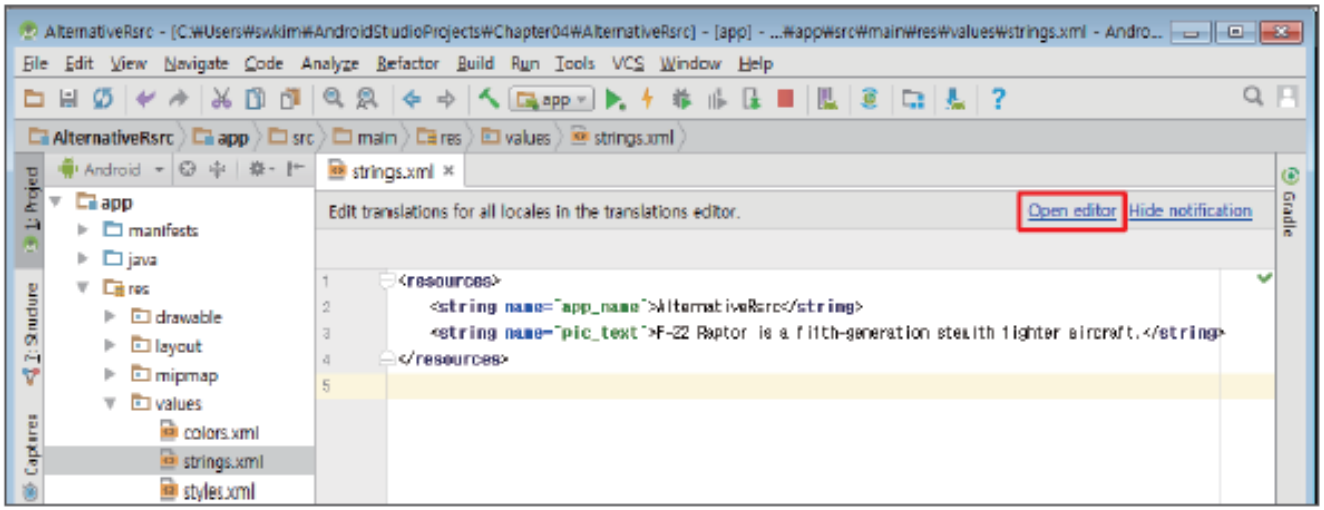
- res/layout/activity_main.xml (land) 파일 열고 입력

activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:orientation="horizontal">
6       <ImageView
7           android:layout_width="0dp"
8           android:layout_height="match_parent"
9           android:layout_weight="1"
10          android:scaleType="fitXY"
11          android:src="@drawable/testpic"/>
12      <TextView
13          android:layout_width="0dp"
14          android:layout_height="match_parent"
15          android:layout_weight="1"
16          android:text="@string/pic_text"
17          android:textSize="16dp"/>
18  </LinearLayout>
```

실습 4-7 AlternativeRsrc

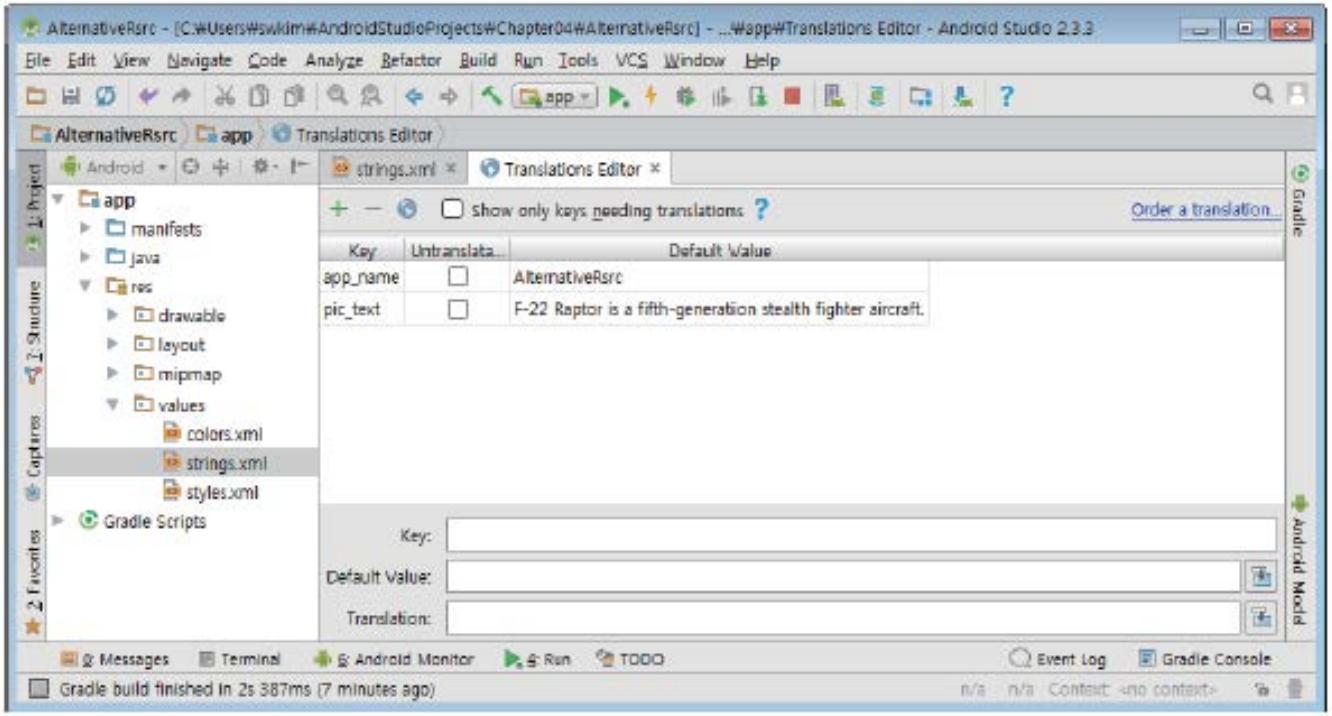
- 한국어 문자열 리소스 추가



(a) string.xml을 열고 Open editor 클릭

실습 4-7 AlternativeRsrc

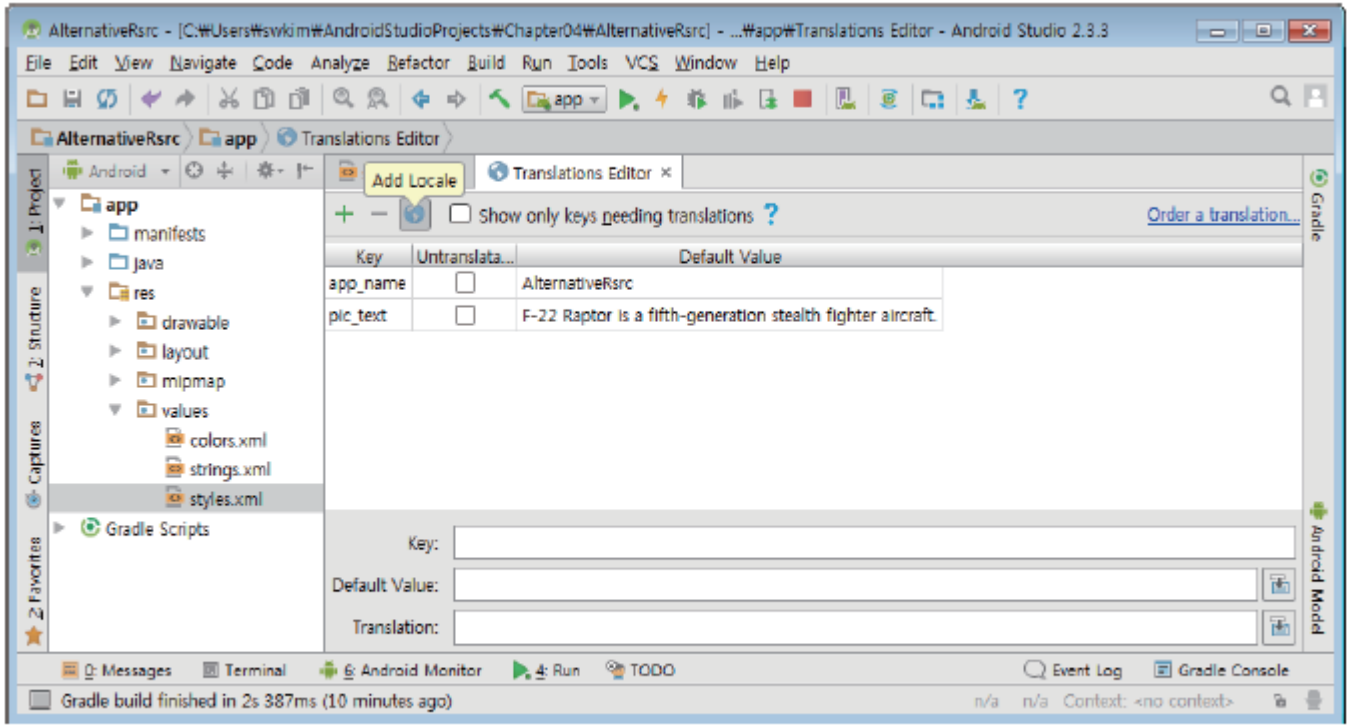
- 한국어 문자열 리소스 추가



(b) Translations Editor 실행

실습 4-7 AlternativeRsrc

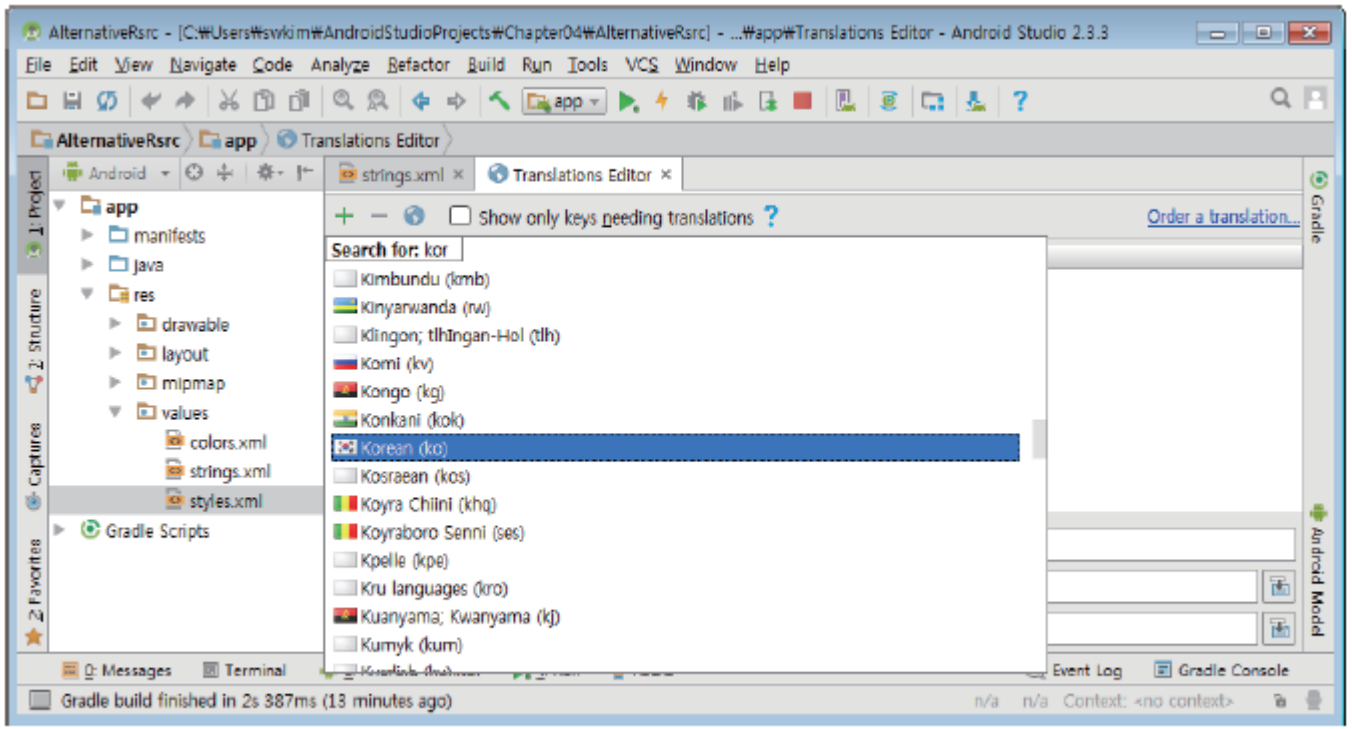
- 한국어 문자열 리소스 추가



(c) [Add Locale] 아이콘 클릭

실습 4-7 AlternativeRsrc

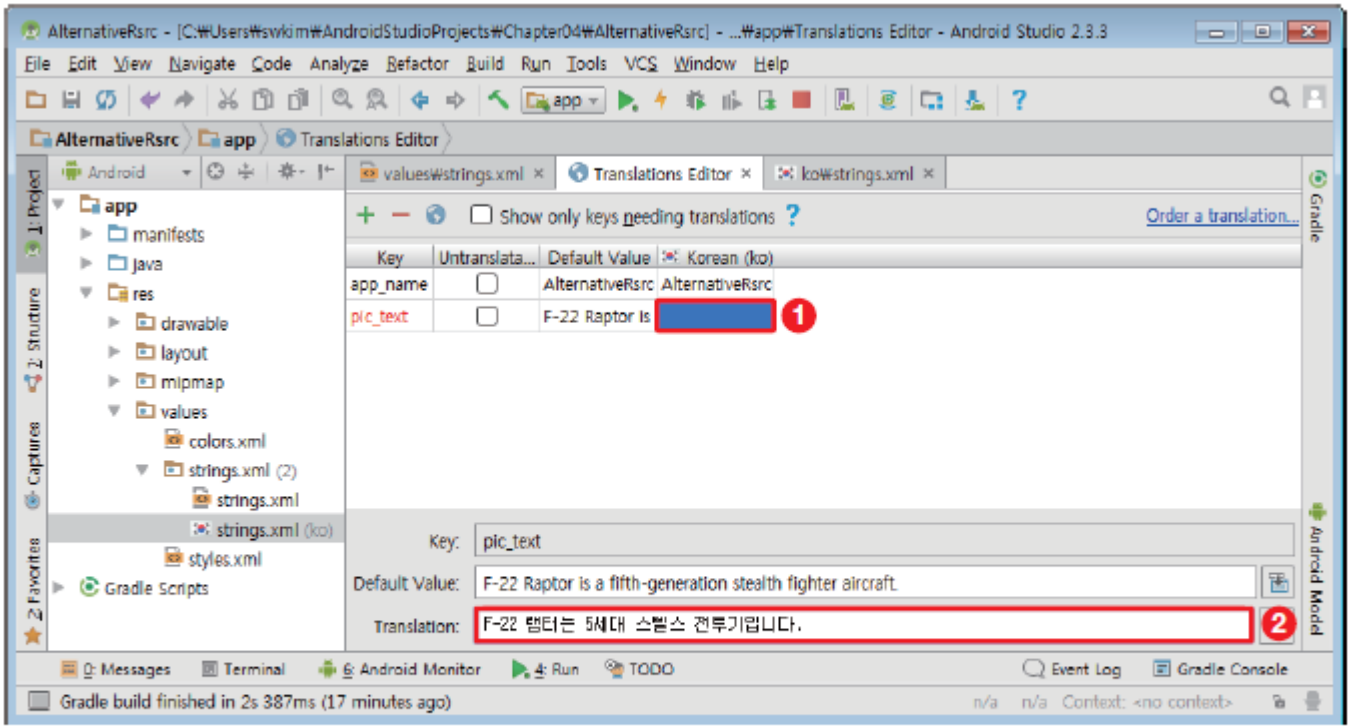
한국어 문자열 리소스 추가



(d) "Korean (ko)" 선택

실습 4-7 AlternativeRsrc

- 한국어 문자열 리소스 추가



(e) 한국어 번역 입력

그림 4-18 한국어 문자열 리소스 생성

실습 4-7 AlternativeRsrc

실행 화면

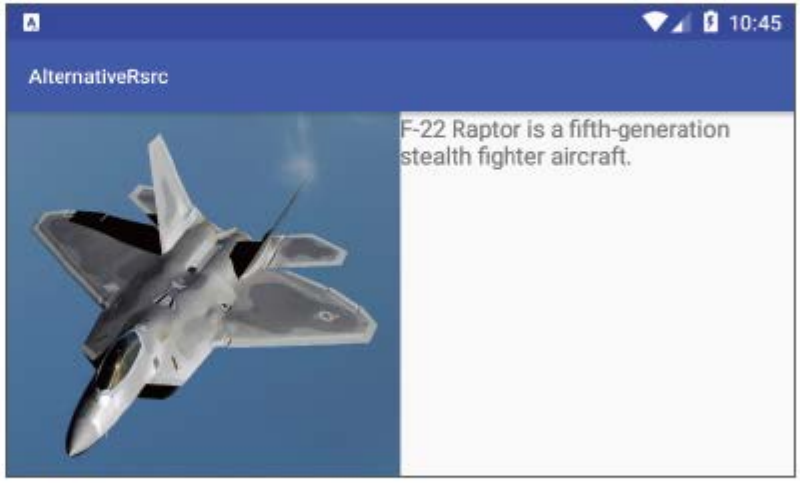
(a) 세로 방향, 한국어

(b) 가로 방향, 한국어

실습 4-7 AlternativeRsrc

실행 화면

(c) 세로 방향, 영어



(d) 가로 방향, 영어

그림 4-19 실행 화면

실습 4-7 AlternativeRsrc

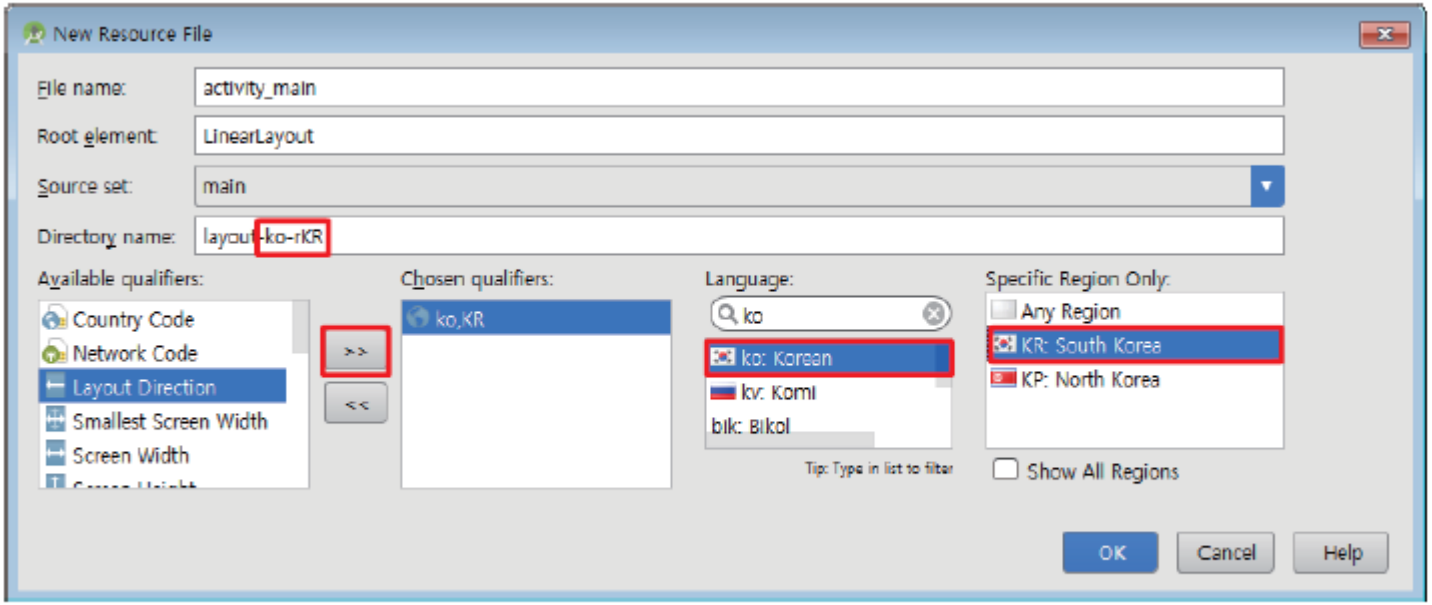
대체 리소스의 종류

표 4-7 대체 리소스 종류

시스템 구성	수식어 사용 예	설명
언어와 지역 (Language and region)	-ko (한국어) -ko-rKR (한국어-대한민국) ...	두 글자의 언어 코드와 더불어 두 글자의 지역 코드를 r을 붙여서 지정한다. 지역 코드는 생략할 수 있다. 언어 또는 언어와 지역이 일치하면 해당 리소스가 선택된다.
가용 폭 (Available width)	-w820dp (가용폭 820dp) ...	디바이스의 화면 폭이 지정 값을 만족하면 해당 리소스가 선택된다.
화면 방향 (Screen orientation)	-port (세로 방향) -land (가로 방향)	화면 방향이 지정 값과 일치하면 해당 리소스가 선택된다.
화면 픽셀 밀도 (Screen pixel density)	-ldpi -mdpi -hdpi -xhdpi -xxhdpi -xxxhdpi -nodpi -tvdpi	디바이스의 픽셀 밀도에 따라 적절한 리소스가 선택된다. 가장 낮은 픽셀 밀도(-ldpi)부터 가장 높은 픽셀 밀도(-xxxhdpi)까지 지원된다. -nodpi 수식어가 붙은 drawable 폴더에 비트맵 리소스를 넣어두면 디바이스의 픽셀 밀도와 상관없이 원래 크기를 유지한다. -tvdpi는 이름대로 안드로이드 TV를 위한 것으로 일반 앱에서는 거의 사용하지 않는다.
플랫폼 버전 (Platform version)	-v7 (안드로이드 2.1) -v8 (안드로이드 2.2) ...	안드로이드 플랫폼 버전이 일치하면 해당 리소스가 선택된다.

실습 4-7 AlternativeRsrc

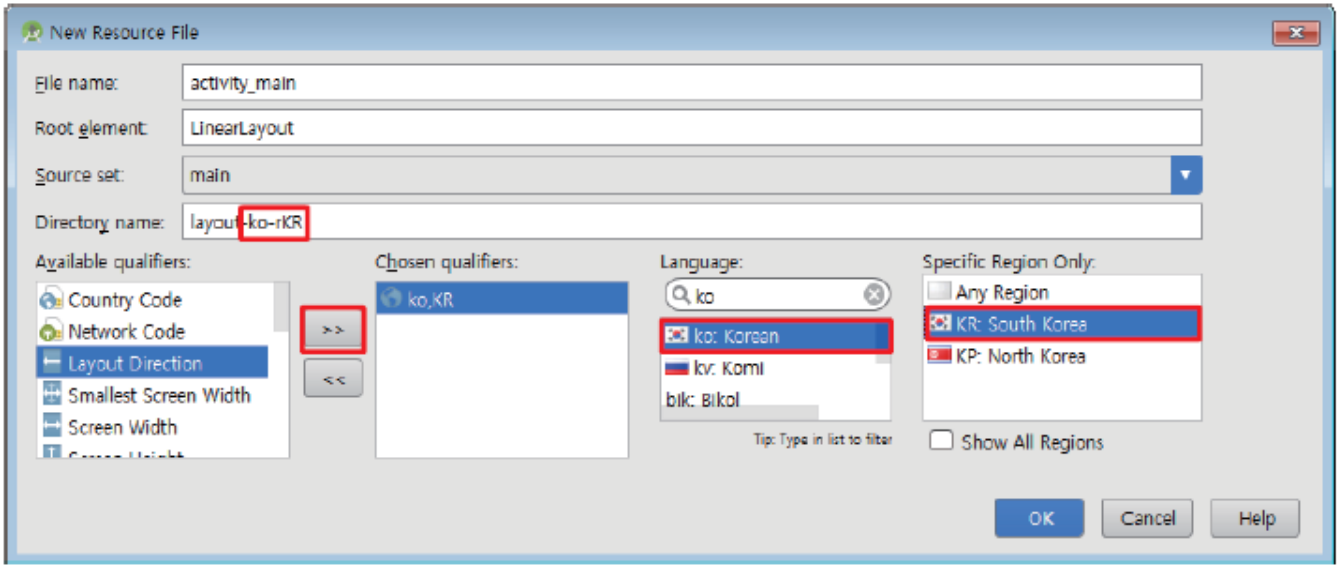
레이아웃 리소스 정의



(a) 언어와 지역 선택

실습 4-7 AlternativeRsrc

- 레이아웃 리소스 정의

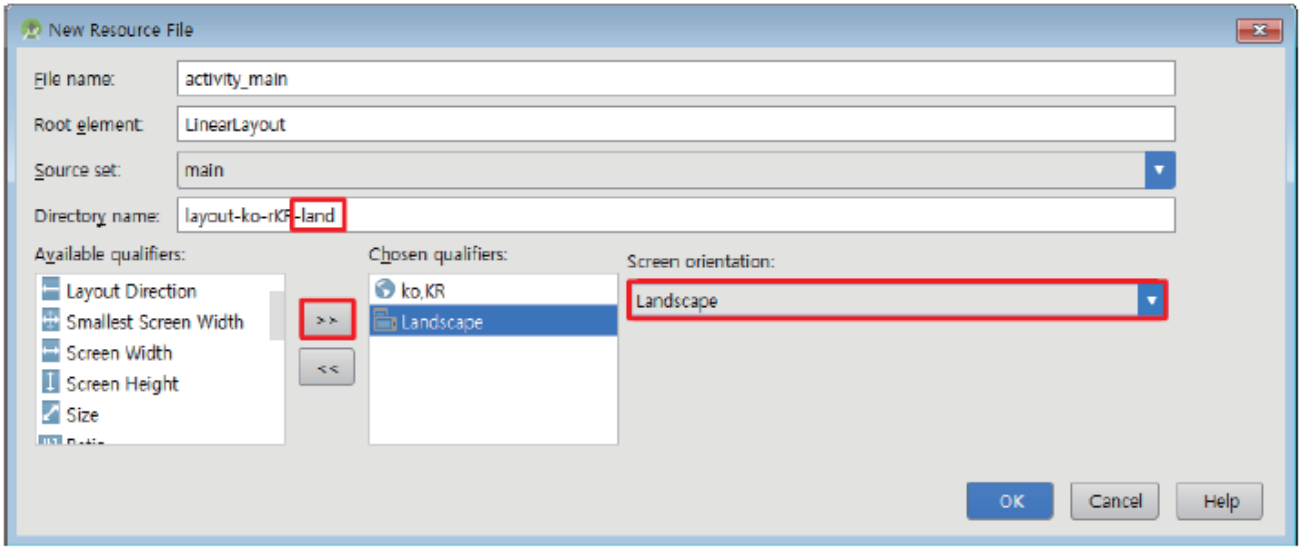


(a) 언어와 지역 선택

- Available qualifiers에서 'Locale' 선택 후 >> 버튼 클릭
- Language에서 "ko" 입력하여 검색 후 'ko: Korean' 선택
- Specific Region Only에서 'KR: South Korea' 선택

실습 4-7 AlternativeRsrc

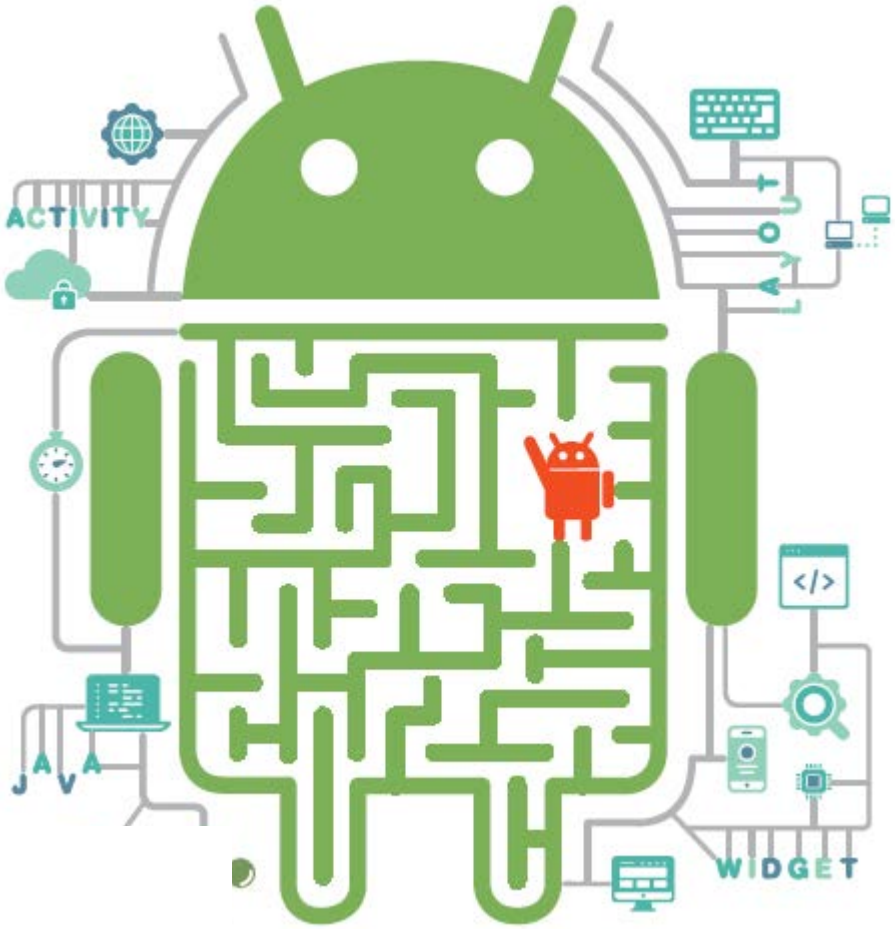
레이아웃 리소스 정의



(b) 화면 방향 선택

- Available qualifiers에서 'Orientation' 선택 후 >> 버튼 클릭
- Screen orientation에서 'Landscape' 선택

그림 4-20 대체 리소스 정의



단계별로 배우는

안드로이드 프로그래밍