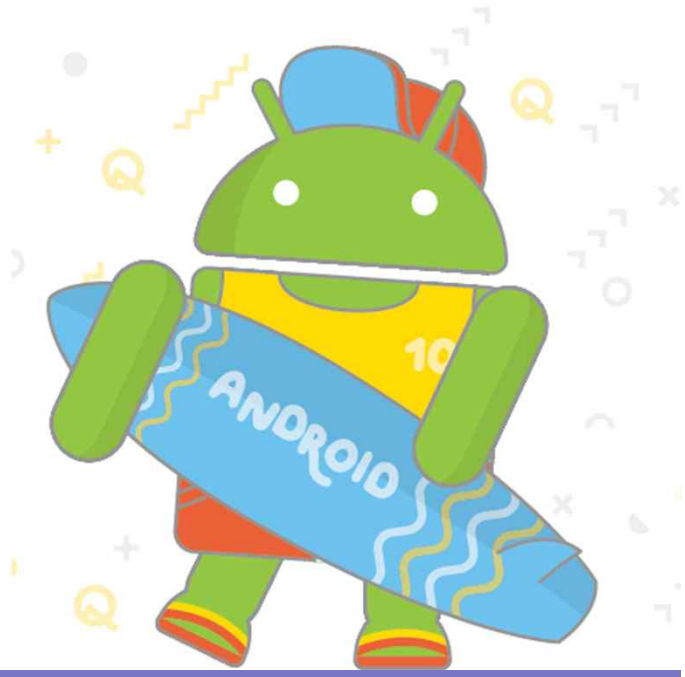
# 07 메뉴와 대화상자



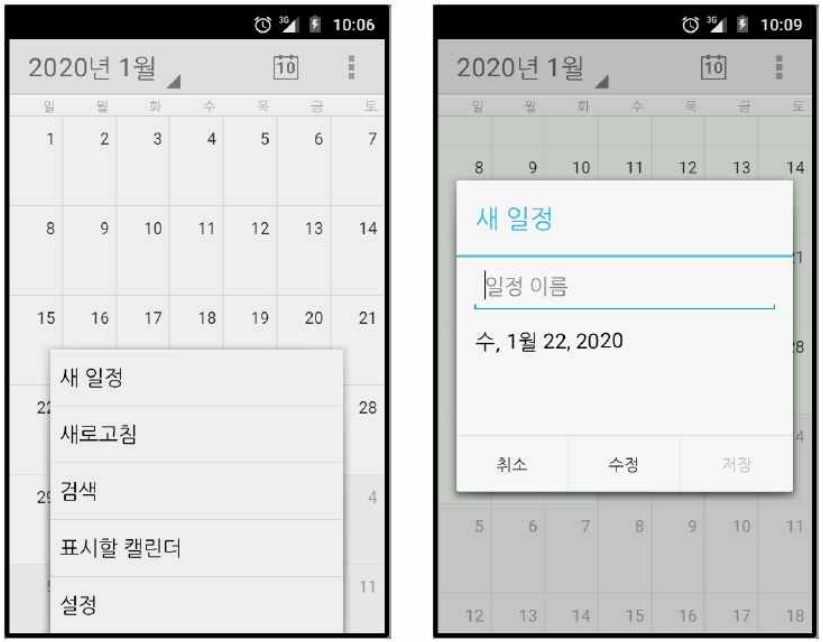
# 학습목표

- 메뉴를 작성하고 사용하는 방법을 익힌다.
- 토스트의 다양한 출력 방법을 알아본다.
- 대화상자 사용법을 익힌다.

# 차례

# 메뉴

- 옵션 메뉴(option menu)
  - 키패드의 메뉴 버튼 또는 화면 오른쪽 위의 메뉴 아이콘을 눌렀을 때 화면 하단에 나오는 메뉴
  - 목록이 많으면 위쪽으로 쌓여 출력됨
    - 화면이 넘어갈 정도로 목록이 많으면 스크롤해서 선택 가능함
- 컨텍스트 메뉴(context menu)
  - 위젯 등을 롱클릭하면 나오는 메뉴
    - 일반적으로 제목과 함께 화면의 중앙에 출력됨




(a) 옵션 메뉴 (b) 컨텍스트 메뉴(날짜를 롱클릭한 화면)

그림 7-1 옵션 메뉴와 컨텍스트 메뉴의 예

# 01 XML을 이용한 옵션 메뉴

- 메뉴 XML 파일을 이용하는 방식

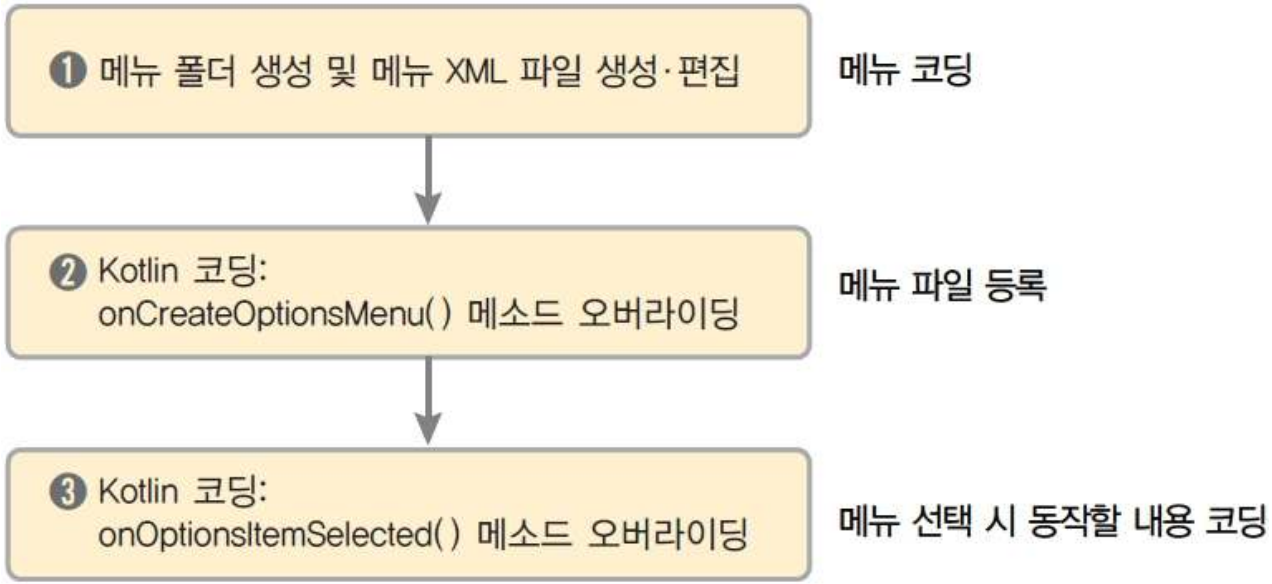



그림 7-2 옵션 메뉴 설정 순서(XML 파일 이용)

# 01 XML을 이용한 옵션 메뉴

- 메뉴 XML 파일 형식

```
<menu>
        <item
            android:id="@+id/항목1아이디"
            android:title="항목1 제목"/>
        <item
            android:id="@+id/항목2아이디"
            android:title="항목2 제목"/>
</menu>
```

# 01 XML을 이용한 옵션 메뉴

- onCreateOptionsMenu() 메소드
  - Activity 클래스에서 오버라이딩함
  - 앱이 실행되면 메뉴의 내용을 XML 파일에서 자동으로 읽어옴
  - 메소드에 코딩할 내용은 다음과 같이 거의 고정되어 있으므로 그대로 사용하면 되지만, 4행의 '메뉴XML아이디'는 화면에 보여줄 메뉴 XML 파일의 id로 바꿔야 함

```
override fun onCreateOptionsMenu(menu: Menu?) : Boolean {
    super.onCreateOptionsMenu(menu)
    var mInflater = menuInflater
    mInflater.inflate(R.menu.메뉴XML아이디, menu)
    return true
}
```

# 01 XML을 이용한 옵션 메뉴

- onOptionsItemSelected( ) 메소드
  - Activity 클래스에서 오버라이딩함
  - 메뉴를 선택했을 때 어떤 동작을 할 것인지를 담고 있음
  - 메뉴는 항목이 여러 개 나오기 때문에 보통 메소드 내부에서 when 문을 사용함

```
override fun onOptionsItemSelected(item: MenuItem) : Boolean {
    when (item.itemId) {
        R.id.항목1아이디 -> {
            // 항목1을 선택했을 때 실행할 코드
            return true
        }
        R.id.항목2아이디 -> {
            // 항목2를 선택했을 때 실행할 코드
            return true
        }
    }
    return false
}
```

  - 위 코드에서 '항목1아이디', '항목2아이디'를 메뉴 XML 파일의 해당 id로 바꾸고 각 항목을 선택할 때 실제로 실행될 코드를 작성함

# 01 XML을 이용한 옵션 메뉴

## <실습 7-1> 배경색 변경 앱 만들기

### 안드로이드 프로젝트 생성

- (1) 새 프로젝트를 만들기
  - 프로젝트 이름 : 'Project7_1'
  - 패키지 이름 : 'com.cookandroid. project7_1'
  - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

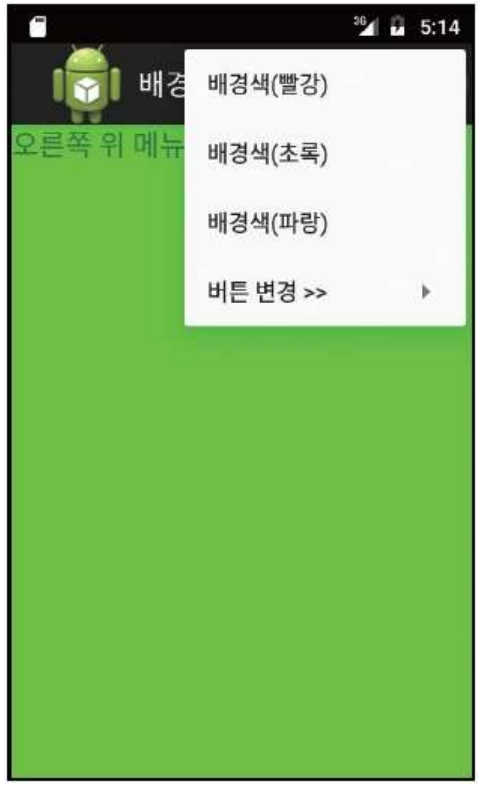


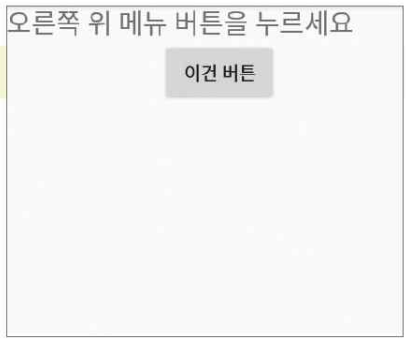

그림 7-3 배경색 변경 앱 결과 화면

# 01 XML을 이용한 옵션 메뉴

- 화면 디자인 및 편집
  - (2) [app]-[res]-[layout]-[activity_main.xml]을 열고 아래쪽의 [Text] 탭을 클릭하여 화면을 간단히 코딩함
    - 바깥 리니어레이아웃의 id는 baseLayout
    - 텍스트뷰 1개와 버튼 1개 생성
      - » 버튼의 id는 button1

**예제 7-1 activity_main.xml**

```
1  <LinearLayout
2      android:id="@+id/baseLayout" >
3      <TextView
4          android:text="오른쪽 위 메뉴 버튼을 누르세요" />
5      <Button
6          android:id="@+id/button1"
7          android:text="이건 버튼" />
8  </LinearLayout>
```



# 01 XML을 이용한 옵션 메뉴

- (3) menu 폴더가 없다면 생성해야 함
  - Project Tree의 [app]-[res]에서 마우스 오른쪽 버튼을 클릭하고 [New]-[Android Resource Directory]를 선택
  - [New Resource Directory] 창에서 Resource type으로 'menu'를 선택하고 <OK>를 클릭

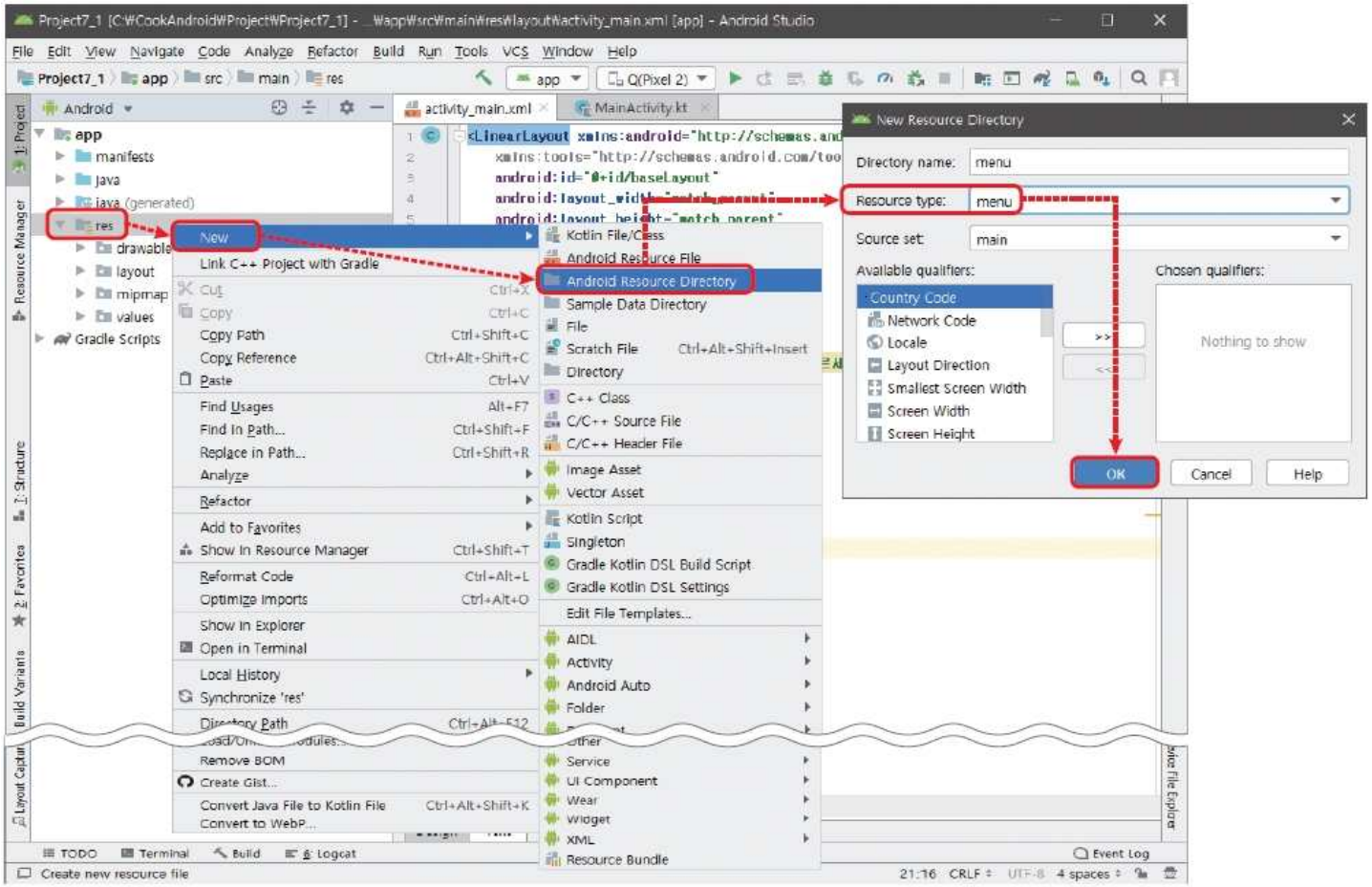



그림 7-4 메뉴 폴더 생성

# 01 XML을 이용한 옵션 메뉴

- (4) 메뉴 XML 파일 만들기
  - Project Tree의 [app]-[res]-[menu]에서 마우스 오른쪽 버튼을 클릭하고 [New]-[Menu resource file]을 선택
  - [New Resource File] 창에서 새로운 파일 이름 'menu1.xml'을 입력하고 <OK>를 클릭

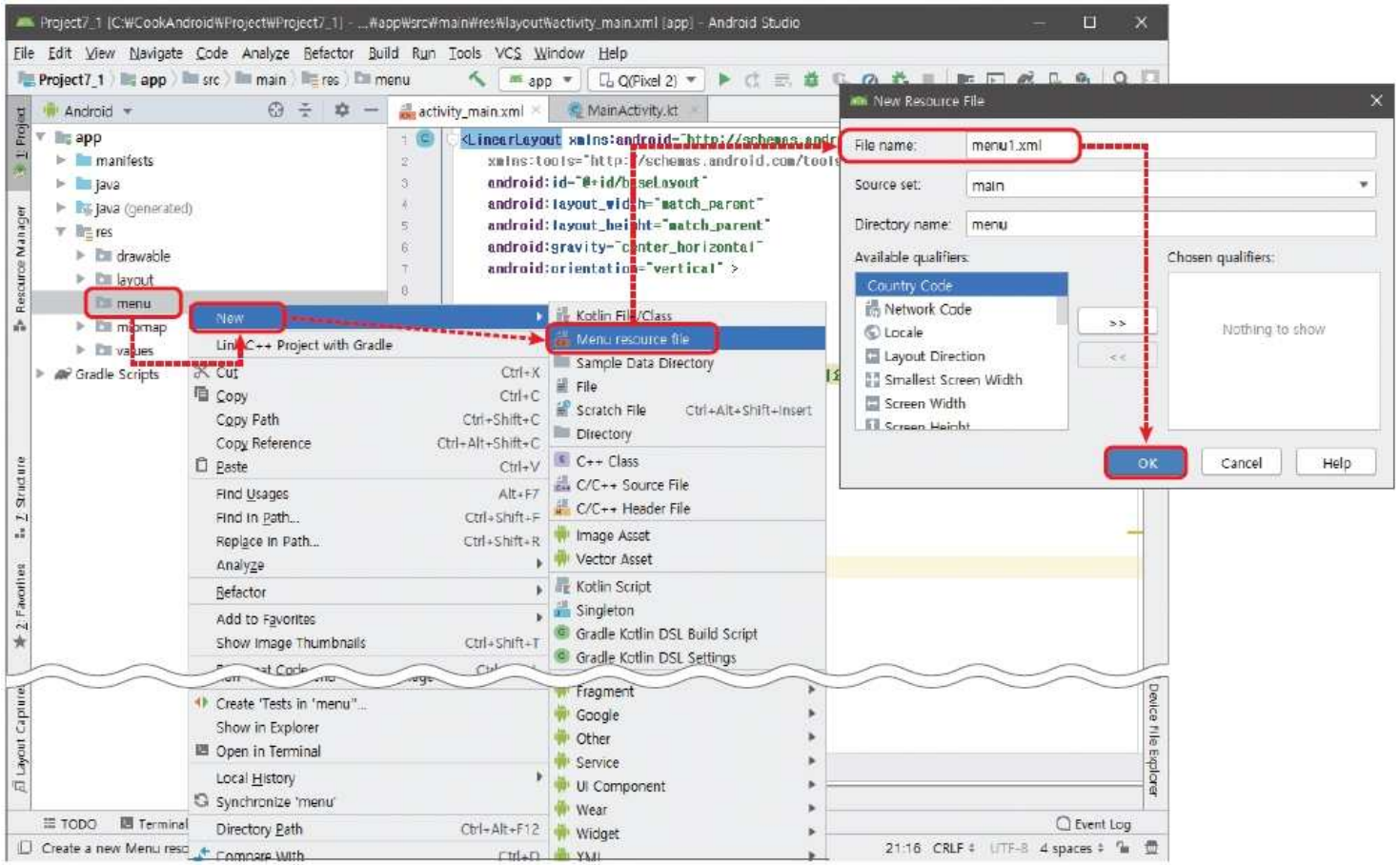



그림 7-5 메뉴 XML 파일 생성

# 01 XML을 이용한 옵션 메뉴

- (5) 아래쪽의 [Text] 탭을 클릭하고 메뉴 XML 파일을 코딩함.

예제 7-2 menu1.xml

```
1  <menu xmlns:android="http://schemas.android.com/apk/res/android" >
2
3      <item
4          android:id="@+id/itemRed"
5          android:title="배경색(빨강)">
6      </item>
7      <item
8          android:id="@+id/itemGreen"
9          android:title="배경색(초록)">
10     </item>
11     <item
12         android:id="@+id/itemBlue"
13         android:title="배경색(파랑)">
14     </item>
15     <item android:title="버튼 변경 >> ">
16         <menu>
17             <item
18                 android:id="@+id/subRotate"
19                 android:title="버튼 45도 회전"/>
20             <item
21                 android:id="@+id/subSize"
22                 android:title="버튼 2배 확대"/>
23         </menu>
24     </item>
25
26 </menu>
```

# 01 XML을 이용한 옵션 메뉴

- Kotlin 코드 작성 및 수정
  - (6) [app]-[java]-[패키지 이름]-[MainActivity]를 열기
  - (7) 다음 내용을 코딩
    - activity_main.xml의 레이아웃과 버튼에 대응할 전역변수 2개 선언
    - 메인 함수 onCreate( ) 안에서 위젯 변수 2개에 위젯을 대입

예제 7-3 Kotlin 코드 1

```
1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3
4      lateinit var baseLayout : LinearLayout
5      lateinit var button1 : Button
6
7      override fun onCreate(savedInstanceState: Bundle?) {
8          super.onCreate(savedInstanceState)
9          setContentView(R.layout.activity_main)
10         title = "배경색 바꾸기"
11         baseLayout = findViewById<LinearLayout>(R.id.baseLayout)
12         button1 = findViewById<Button>(R.id.button1)
13     }
14
15 }
```

# 01 XML을 이용한 옵션 메뉴

- (8) 옵션 메뉴를 등록하는 메소드 onCreateOptionsMenu( )를 부모 클래스인 Activity 클래스로부터 오버라이딩함
  - 자동 완성 기능 사용
    - onCreate() 메소드 밖에 커서를 두고 [Code]-[Override Methods]를 선택하고 [android.app.Activity] 하위의 onCreate OptionsMenu() 메소드를 선택함

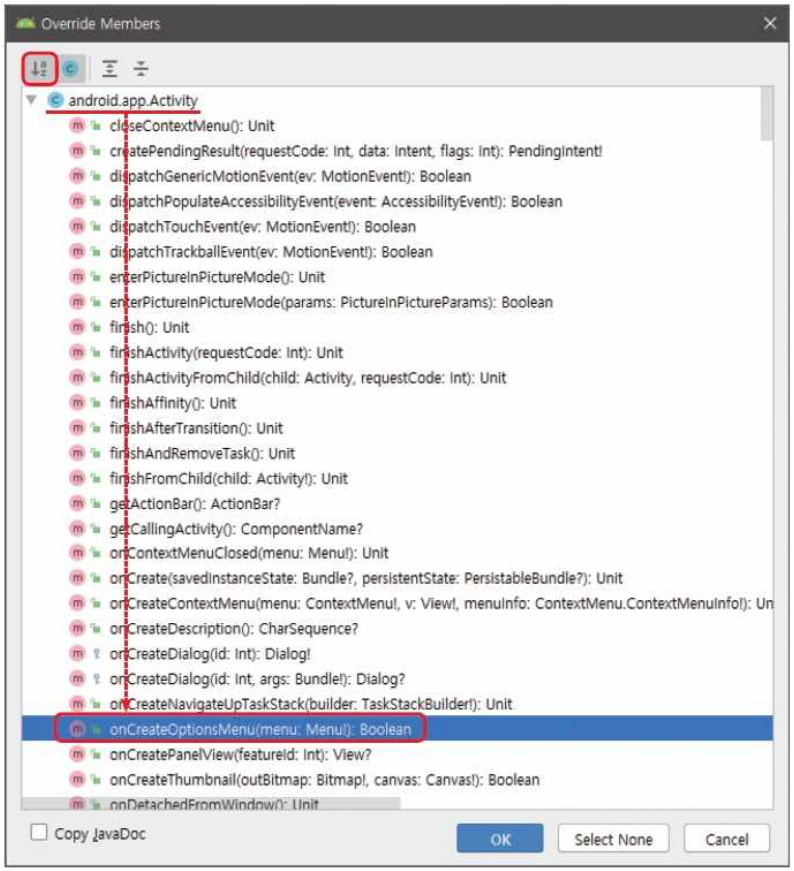



그림 7-6 onCreateOptionsMenu( ) 메소드 오버라이딩

# 01 XML을 이용한 옵션 메뉴

- (9) 자동 완성된 코드에 나머지 코딩

예제 7-4 Kotlin 코드 2

```
1           ~~~ 생략([예제 7-3]과 동일) ~~~
2           baseLayout = findViewById<LinearLayout>(R.id.baseLayout)
3           button1 = findViewById<Button>(R.id.button1)
4       }
5
6       override fun onCreateOptionsMenu(menu: Menu?) : Boolean {
7           super.onCreateOptionsMenu(menu)
8           var mInflater = menuInflater
9           mInflater.inflate(R.menu.menu1, menu)
10          return true
11      }
12
13  }
```

# 01 XML을 이용한 옵션 메뉴

- (10) 메뉴를 클릭했을 때 동작할 메소드 onOptionsItemSelected( )를 코딩
  - 커서를 onCreate() 밖에 두고 [Code]-[Override Methods]를 선택하여 onOptionsItemSelected( )를 자동 완성한 후 when 문을 사용하여 나머지 코딩함

예제 7-5 Kotlin 코드 3

```
        ~~~ 생략([예제 7-4]와 동일) ~~~
        mInflater.inflate(R.menu.menu1, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem) : Boolean {
        when (item.itemId) {
            R.id.itemRed -> {
                baseLayout.setBackgroundColor(Color.RED)
                return true
            }
            ~~~ 생략(itemGreen, itemBlue의 when 문) ~~~
            R.id.subRotate -> {
                button1.rotation = 45f
                return true
            }
            R.id.subSize -> {
                button1.scaleX = 2f
                return true
            }
        }
        return false
    }
```

# 01 XML을 이용한 옵션 메뉴

- 프로젝트 실행 및 결과 확인
  - (11) 완성된 코드를 실행한 후 키패드의 메뉴 버튼을 누르고 각 항목을 선택하면 [그림 7-3]과 같이 배경색이나 버튼 모양이 바뀔 것

# 02 Kotlin 코드만 이용한 옵션 메뉴

- onCreateOptionsMenu( ) 메소드 안에서 메뉴 XML 파일에 접근하는 대신에 직접 menu.add( ) 메소드로 메뉴 항목을 추가하고 onOptionsItemSelected( ) 메소드의 when 문을 코드에서 지정한 항목의 id 번호로 변경하여 구현

**예제 7-6 Kotlin 코드만으로 구성한 옵션 메뉴**

```
1  override fun onCreateOptionsMenu(menu: Menu?) : Boolean {
2      super.onCreateOptionsMenu(menu)
3
4      menu!!.add(0, 1, 0, "배경색 (빨강)")
5      menu!!.add(0, 2, 0, "배경색 (초록)")
6      menu!!.add(0, 3, 0, "배경색 (파랑)")
7
8      var sMenu : SubMenu = menu.addSubMenu("버튼 변경 >>")
9      sMenu.add(0, 4, 0, "버튼 45도 회전")
10     sMenu.add(0, 5, 0, "버튼 2배 확대")
11
12     return true
13 }
14
15 override fun onOptionsItemSelected(item: MenuItem): Boolean {
16     when (item.itemId) {
17         1-> {
18             baseLayout.setBackgroundColor(Color.RED)
19             return true
20         }
21         ~~~ 생략(when의 변숫값을 2, 3, 4, 5로 변경) ~~~
```

# 02 Kotlin 코드만 이용한 옵션 메뉴

▶ **직접 풀어보기 7-1**

[실습 7-1]을 다음과 같이 수정하라.

- 레이아웃은 RelativeLayout을 사용하고 텍스트뷰, 에디트텍스트, 이미지뷰를 적절히 배치한다.
- 에디트텍스트에 각도를 입력하고 옵션 메뉴의 [그림 회전]을 선택하면 해당 각도만큼 이미지뷰가 회전한다.
- 한라산, 추자도, 범섬 옵션 메뉴는 라디오버튼과 같이 3개 중 하나만 체크하고, 선택하면 이미지뷰가 해당 이미지로 바뀌게 한다(이미지나 이미지 이름이 달라도 된다).

HINT 1 메뉴 XML에서 여러 항목을 묶으려면 〈group〉…〈/group〉을 이용한다. 예를 들면 다음과 같다.

```
〈group
      android:checkableBehavior="single"〉
      〈item
          android:id="@+id/item1"
          android:title="한라산"
          android:checked="true"〉
      〈/item〉
      ~~~ 생략 ~~~
〈/group〉
```

HINT 2 이미지뷰의 이미지를 변경하려면 setImageResource( ) 메소드를 사용한다.

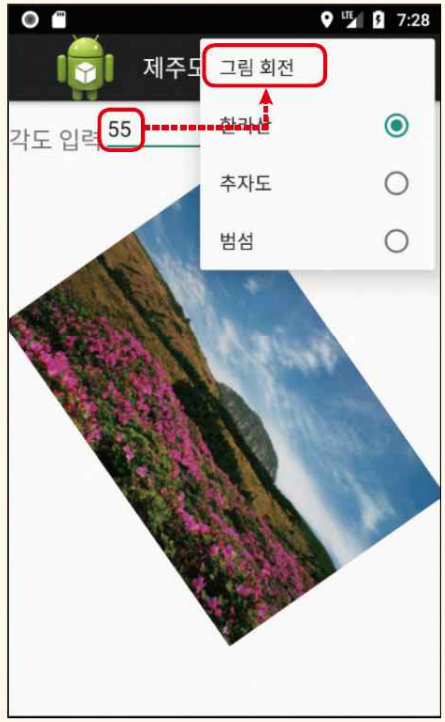



그림 7-7 제주도 풍경 앱

# 03 XML을 이용한 컨텍스트 메뉴

- 컨텍스트 메뉴에서 메뉴 XML 파일을 이용하는 방식은 옵션 메뉴와 비슷함
- 단, 여러 개의 위젯에 메뉴를 설정할 수 있으므로 onCreate( ) 메소드에서 컨텍스트 메뉴를 registerForContextMenu( )로 등록해야 함

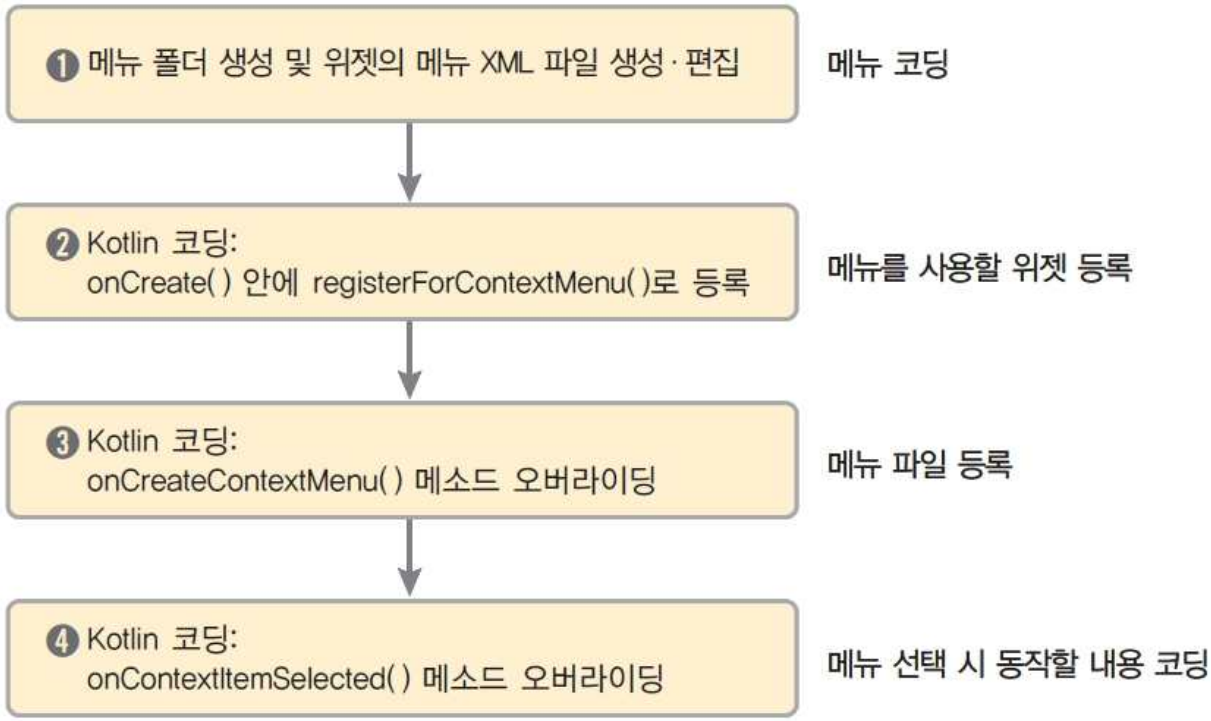



그림 7-8 컨텍스트 메뉴 설정 순서(XML 파일 이용)

# 03 XML을 이용한 컨텍스트 메뉴

- 메뉴 XML 파일은 컨텍스트 메뉴가 나오게 할 위젯마다 별도의 파일로 만들어야 함
  - 메뉴 XML의 문법은 옵션 메뉴와 동일함

– onCreateContextMenu( ) 메소드에는 위젯별로 컨텍스트 메뉴가 나타나야 하므로 위젯별 컨텍스트 메뉴를 if 문으로 등록해야 함

```
override fun onCreateContextMenu(menu: ContextMenu?, v: View?,
             menuInfo: ContextMenu.ContextMenuInfo? ) {
    super.onCreateContextMenu(menu, v, menuInfo)

    var mInflater = this.menuInflater
    if (v === 위젯1) {
        mInflater.inflate(R.menu.첫번째메뉴XML파일, menu)
    }
    if (v === button2) {
        mInflater.inflate(R.menu.두번째메뉴XML파일, menu)
    }
}
```

# 03 XML을 이용한 컨텍스트 메뉴

## <실습 7-2> 배경색 변경 및 버튼 변경 앱 만들기

- [실습 7-1]과 비슷하게 동작하는 컨텍스트 메뉴 만들어 보기
- <배경색 변경>을 롱클릭하면 세 가지 컨텍스트 메뉴가 나옴
- <버튼 변경>을 롱클릭하면 버튼을 회전시키거나 크기를 변경하는 두 가지 컨텍스트 메뉴가 나옴

## 안드로이드 프로젝트 생성

- (1) 새 프로젝트 만들기
  - 프로젝트 이름 : 'Project7_2'
  - 패키지 이름 : 'com.cookandroid.project7_2'
  - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

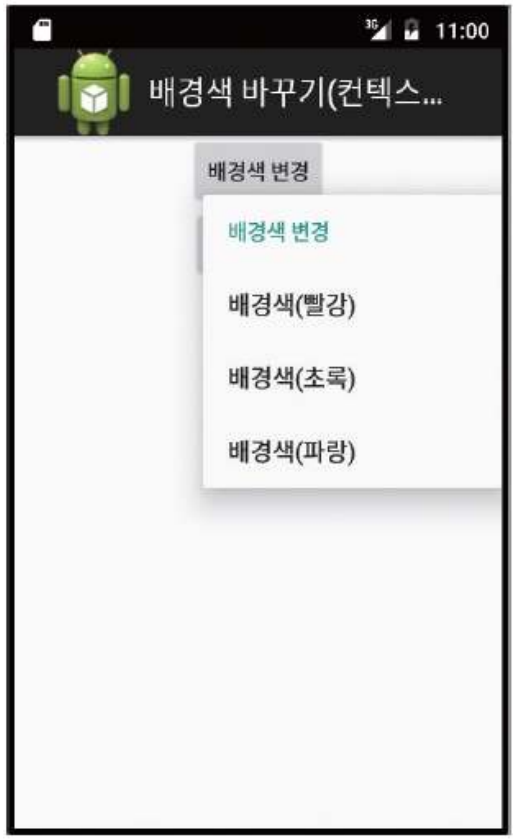



그림 7-9 배경색 변경 및 버튼 변경 앱 결과 화면

# 03 XML을 이용한 컨텍스트 메뉴

- 화면 디자인 및 편집
  - (2) [app]-[res]-[layout]-[activity_main.xml]을 열고, 아래쪽의 [Text] 탭을 클릭하여 화면을 간단히 코딩함
    - 레이아웃의 id : baseLayout
    - 버튼 2개 생성, 버튼의 id : button1, button2

예제 7-7 activity_main.xml

```
1  <LinearLayout
2      android:gravity="center_horizontal"
3      <Button
4          android:id="@+id/button1"
5          android:text="배경색 변경" />
6      <Button
7          android:id="@+id/button2"
8          android:text="버튼 변경" />
9  </LinearLayout>
```

# 03 XML을 이용한 컨텍스트 메뉴

- (3) [res]-[menu] 폴더를 만들고, 폴더에 menu1.xml과 menu2.xml 파일 만들기
- (4) menu1.xml은 배경색 변경과 관련된 3개 항목을, menu2.xml은 버튼과 관련된 2개 항목을 생성함
  - 즉 [실습 7-1]에서 1개였던 메뉴 XML 파일을 2개로 분리함.

**예제 7-8 menu1.xml**

```
1  <menu xmlns:android="http://schemas.android.com/apk/res/android" >
2      <item
3          android:id="@+id/itemRed"
4          android:title="배경색(빨강)">
5      </item>
6      ~~~ 생략(배경색 항목 2개) ~~~
7  </menu>
```

**예제 7-9 menu2.xml**

```
1  <menu xmlns:android="http://schemas.android.com/apk/res/android" >
2      <item
3          android:id="@+id/subRotate"
4          android:title="버튼 45도 회전"/>
5      <item
6          android:id="@+id/subSize"
7          android:title="버튼 2개 확대"/>
8  </menu>
```

# 03 XML을 이용한 컨텍스트 메뉴

- Kotlin 코드 작성 및 수정
  - (5) [app]-[java]-[패키지 이름]-[MainActivity]를 열기
  - (6) 다음 내용을 코딩
    - activity_main.xml의 레이아웃과 버튼 2개에 대응할 전역변수 3개 선언
    - 메인 함수 onCreate( ) 안에서 위젯 변수 3개에 위젯 대입
    - 버튼 위젯 변수 2개를 registerForContextMenu( )에 등록

# 03 XML을 이용한 컨텍스트 메뉴

예제 7-10 Kotlin 코드 1

```
1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3      lateinit  var baseLayout : LinearLayout
4      lateinit  var button1 : Button
5      lateinit  var button2 : Button
6
7      override fun onCreate(savedInstanceState: Bundle?) {
8          super.onCreate(savedInstanceState)
9          setContentView(R.layout.activity_main)
10         title = "배경색 바꾸기(컨텍스트 메뉴)"
11         baseLayout = findViewById<LinearLayout>(R.id.baseLayout)
12                       as LinearLayout
13         button1 = findViewById<Button>(R.id.button1) as Button
14         registerForContextMenu(button1)
15         button2 = findViewById<Button>(R.id.button2) as Button
16         registerForContextMenu(button2)
17     }
18
19 }
```

# 03 XML을 이용한 컨텍스트 메뉴

- (7) 컨텍스트 메뉴를 등록하는 메소드 onCreateContextMenu( )를 부모 클래스인 Activity 클래스로부터 오버라이딩함
- (8) 자동 완성된 메소드 내부를 코딩함
  - [실습 7-1]과 비슷하지만 선택한 위젯에 따라 인플레이트하는 메뉴 XML 파일을 다르게 함

예제 7-11 Kotlin 코드 2

```
override fun onCreateContextMenu(menu: ContextMenu?, v: View?,
                menuInfo: ContextMenu.ContextMenuInfo? ) {
    super.onCreateContextMenu(menu, v, menuInfo)

    var mInflater = this.menuInflater
    if (v === button1) {
        menu!!.setHeaderTitle("배경색 변경")
        mInflater.inflate(R.menu.menu1, menu)
    }
    if (v === button2) {
        mInflater.inflate(R.menu.menu2, menu)
    }
}
```

# 03 XML을 이용한 컨텍스트 메뉴

- (9) 실제로 동작할 메소드 onContextItemSelected( )를 같은 방식으로 자동 완성한 후 코딩함
  - 메소드의 내용은 [실습 7-1]의 onOptionsItemSelected( )와 거의 비슷함
  - 단, [예제 7-5]에서 14행과 18행의 button1을 button2로 변경

■ 프로젝트 실행 및 결과 확인

- (10) 완성된 코드를 실행하고 각 버튼을 롱클릭하여 컨텍스트 메뉴를 테스트하면 [그림 7-9]와 같음

▶ 직접 풀어보기 7-2

[실습 7-2]의 컨텍스트 메뉴를 메뉴 XML 파일 없이 Kotlin 코드로만 완성하라. [예제 7-6]을 참조한다.

# 토스트

- 토스트(toast)
  - 화면에 잠깐 나타났다 사라지는 메시지
  - 사용자가 인식해야 할 작은 메시지를 보여줄 때 사용하면 편리함
  - 프로그래머가 디버깅 용도로 사용하기에도 적당함

  - 토스트의 가장 일반적인 사용 형식

```
Toast.makeText(Context context, String message, int duration).show()
```

  - context : 현재 액티비티(화면)를 표시하기 위해 this를 주로 사용
    - 버튼을 클릭했을 때 내 부 클래스에서 토스트를 출력하기 위해 applicationContext를 사용
  - duration : 화면에 나타나는 시간
    - Toast.LENGTH_LONG 또는 Toast.LENGTH_SHORT 중 하나 사용
  - show() : 생성된 토스트를 화면에 보여주기 위함.

# 토스트

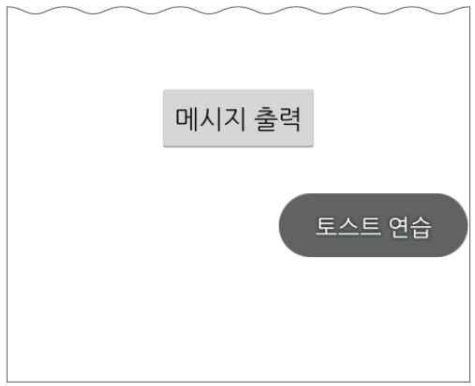
- 토스트는 기본적으로 화면의 중앙 하단 부근에 나타남
  - setGravity( ) 메소드 : 사용하면 위치를 변경함

```
Toast.setGravity(int gravity, int xOffset, int yOffset)
```

  - gravity : 화면의 상단, 중앙, 하단 등을 지정
  - xOffset, yOffset : 떨어진 거리를 나타냄

**예제 7-12 토스트 연습용 activiting_main.xml 코드**

```
1  <LinearLayout
2      android:gravity="center" >
3
4      <Button
5          android:id="@+id/button1"
6          android:text="메시지 출력" />
7
8  </LinearLayout>
```



# 토스트

**예제 7-13 토스트 연습용 Kotlin 코드**

```
1  public override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      setContentView(R.layout.activity_main)
4      title = "토스트 연습"
5
6      var button1 = findViewById<Button>(R.id.button1)
7
8      button1.setOnClickListener {
9          var tMsg = Toast.makeText(applicationContext,
10                         "토스트 연습", Toast.LENGTH_SHORT)
11
12         var display = (getSystemService(Context.WINDOW_SERVICE) as
13                         WindowManager).defaultDisplay
14         var xOffset = (Math.random() * display.width).toInt()
15         var yOffset = (Math.random() * display.height).toInt()
16
17         tMsg.setGravity(Gravity.TOP or Gravity.LEFT, xOffset, yOffset)
18         tMsg.show()
19     }
20 }
```

# 01 기본 대화상자

- 대화상자(dialog)
  - 화면에 메시지를 나타낸 후 확인이나 취소 같은 사용자의 선택을 받아들이는 경우에 사용함
  - 토스트보다 좀 더 강력한 메시지를 보여줄 때 적당함
  - 사용자에게 내용을 좀 더 확실히 주지시켜야 할 때 혹은 계속 진행할지 여부를 선택하게 할 때 사용함

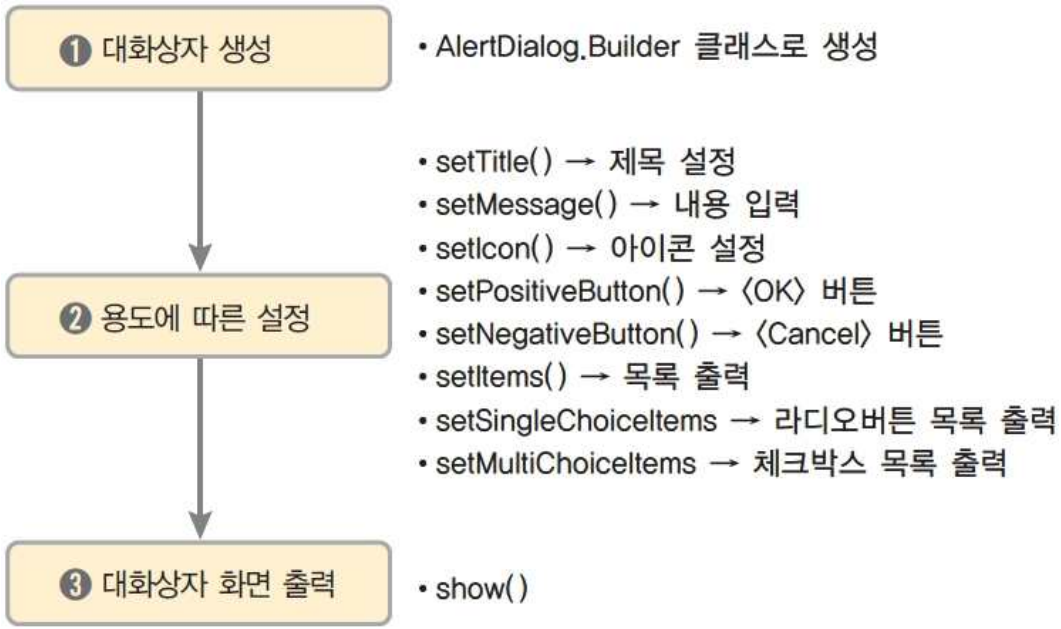



그림 7-10 대화상자 설정 순서

# 01 기본 대화상자

- AlertDialog.Builder 클래스
  - 안드로이드에서 대화상자를 생성할 때 주로 사용는 클래스
  - 다양한 메소드를 이용하여 대화상자를 꾸미고 최종적으로는 show() 메소드로 대화상자를 화면에 보이도록 함

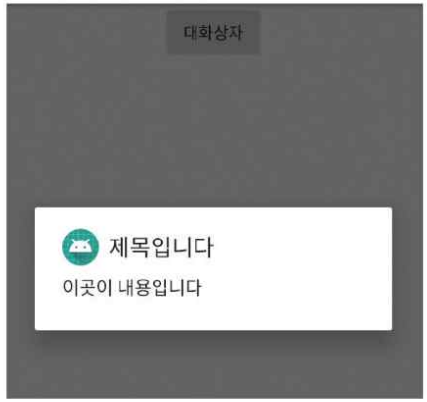
예제 7-14 대화상자 연습용 activity_main.xml 코드

```
1  <LinearLayout
2      android:gravity="center_horizontal" >
3      <Button
4          android:id="@+id/button1"
5          android:text="대화상자" />
6  </LinearLayout>
```

# 01 기본 대화상자

**예제 7-15** 기본적인 대화상자의 Kotlin 코드

```
1  public override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      setContentView(R.layout.activity_main)
4      var button1 = findViewById<Button>(R.id.button1)
5
6      button1.setOnClickListener {
7          var dlg = AlertDialog.Builder(this@MainActivity)
8          dlg.setTitle("제목입니다")
9          dlg.setMessage("이곳이 내용입니다")
10         dlg.setIcon(R.mipmap.ic_launcher)
11         dlg.show()
12     }
13 }
```

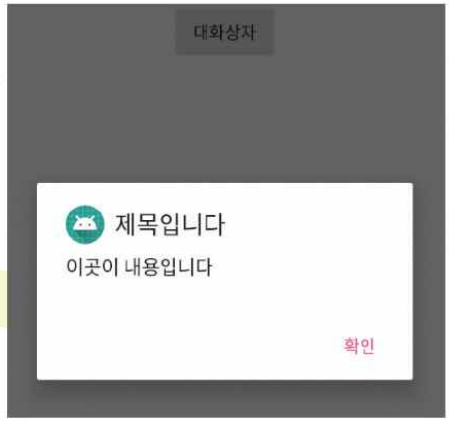


# 01 기본 대화상자

- setPositiveButton( ) 메소드
  - <확인>이나 <닫기>를 1개만 사용할 때 사용
  - setPositiveButton("문자열", 람다식) 형태로 사용

**예제 7-16 버튼이 1개인 대화상자의 Kotlin 코드**

```
1  button1.setOnClickListener {
2      var dlg = AlertDialog.Builder(this@MainActivity)
3      dlg.setTitle("제목입니다")
4      dlg.setMessage("이곳이 내용입니다")
5      dlg.setIcon(R.mipmap.ic_launcher)
6      dlg.setPositiveButton("확인", null)
7      dlg.show()
8  }
```




  - 현재는 람다식 부분에 null을 입력했으므로 <확인>을 클릭해도 대화상자만 닫힐 뿐 아무 동작도 일어나지 않음

# 01 기본 대화상자

- <확인>을 클릭한 후 다음 동작을 해야 하는 경우 [예제 7-16]의 6행을 다음과 같이 수정함

**예제 7-17 버튼 클릭 시 동작하는 대화상자의 Kotlin 코드**

```
1  dlg.setPositiveButton("확인") { dialog, which ->
2      Toast.makeText(this@MainActivity,
3          "확인을 눌렀네요", Toast.LENGTH_SHORT).show()
4  }
```

- <확인>과 <취소> 버튼이 있는 대화상자를 만들려면 setNegativeButton( ) 메소드를 추가함
  - 형식은 setPositiveButton( )과 동일함

# 02 목록 대화상자

– 대화상자에 리스트 형태의 목록을 출력하고 그중 하나를 선택하게 할 수 있는 예제

**예제 7-18 기본 목록 대화상자의 Kotlin 코드**

```
1  button1.setOnClickListener {
2      var versionArray = arrayOf("오레오", "파이", "큐(10)")
3      var dlg = AlertDialog.Builder(this@MainActivity)
4      dlg.setTitle("좋아하는 버전은?")
5      dlg.setIcon(R.mipmap.ic_launcher)
6      dlg.setItems(versionArray) { dialog, which ->
7          button1.text = versionArray[which]
8      }
9      dlg.setPositiveButton("닫기", null)
10     dlg.show()
11 }
```

# 02 목록 대화상자

- setSingleChoiceItems( )
  - 항목을 선택해도 대화상자가 닫히지 않도록 setItems( ) 대신 사용
  - 라디오버튼과 같은 형태로 출력됨
  - setSingleChoiceItems(문자열 배열, 초기 선택 인덱스) { }로 파라미터가 2개임

예제 7-19 라디오버튼 목록 대화상자의 Kotlin 코드

```
1  dlg.setSingleChoiceItems(versionArray, 0) { dialog, which ->
2      button1.text = versionArray[which]
3  }
```

# 02 목록 대화상자

- setMultiChoiceItems( )
  - 여러 개를 동시에 선택할 때 사용
  - 체크박스 형태로 표시 됨

**예제 7-20 체크박스 목록 대화상자의 Kotlin 코드**

```
1  button1.setOnClickListener {
2      var versionArray = arrayOf("오레오", "파이", "큐(10)")
3      var checkArray = booleanArrayOf(true, false, false)
4      var dlg = AlertDialog.Builder(this@MainActivity)
5      dlg.setTitle("좋아하는 버전은?")
6      dlg.setIcon(R.mipmap.ic_launcher)
7      dlg.setMultiChoiceItems(versionArray,
8          checkArray) { dialog, which, isChecked ->
9          button1.text = versionArray[which]
10     }
11     dlg.setPositiveButton("닫기", null)
12     dlg.show()
13 }
```

# 02 목록 대화상자

## <실습 7-3> 사용자 정보 입력 앱 만들기

- 그림과 색상이 들어간 토스트, 사용자 정보를 입력받는 대화상자를 만들어보기

### 안드로이드 프로젝트 생성

- (1) 새 프로젝트 만들기
  - 프로젝트 이름 : 'Project7_3'
  - 패키지 이름 : 'com.cookandroid.project7_3'
  - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름


그림 7-11 사용자 정보 입력 앱 결과 화면

# 02 목록 대화상자

- 화면 디자인 및 편집
  - (2) [app]-[res]-[layout]-[activity_main.xml]을 열고, 아래쪽의 [Text] 탭을 클릭하여 화면을 간단히 코딩함
    - 텍스트뷰 2개와 버튼 1개 생성
    - 위젯의 id : tvName, tvEmail, button1

예제 7-21 activity_main.xml

```
1  <LinearLayout
2      android:gravity="center_horizontal" >
3      <TextView
4          android:id="@+id/tvName"
5          android:text="사용자 이름" />
6      <TextView
7          android:id="@+id/tvEmail"
8          android:text="이메일" />
9      <Button
10         android:id="@+id/button1" />
11 </LinearLayout>
```

# 02 목록 대화상자

- (3) 대화상자에서 사용할 레이아웃 XML 파일 만들기
  - [app]-[res]-[layout]에서 마우스 오른쪽 버튼을 클릭하고 [New]-[Layout resource file]을 선택

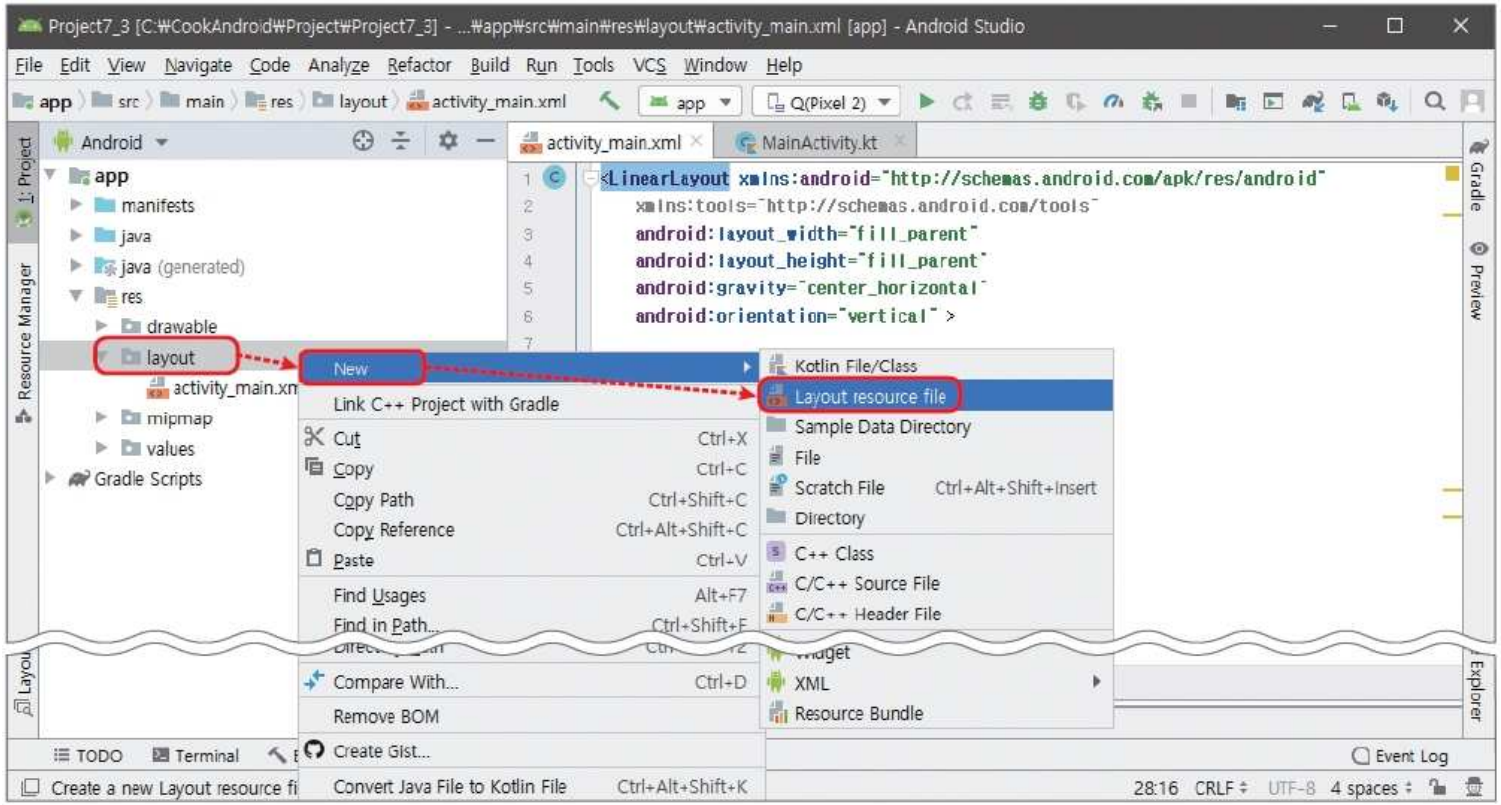



그림 7-12 레이아웃 XML 파일 생성 1

# 02 목록 대화상자

- (4) [New Resource File] 창에서 아래 내용을 입력하고 <OK> 클릭
  - File name : 'dialog1.xml'
  - Root element : 'Linear Layout'

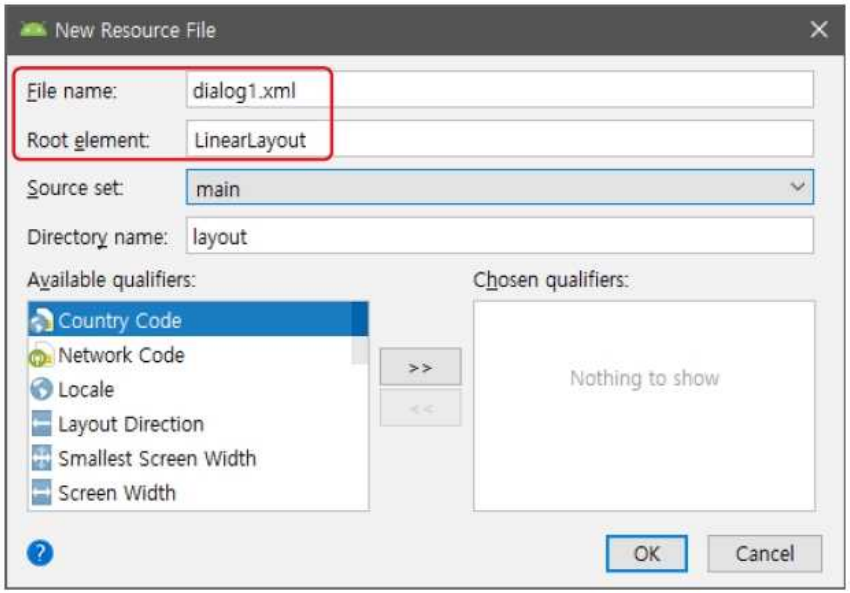


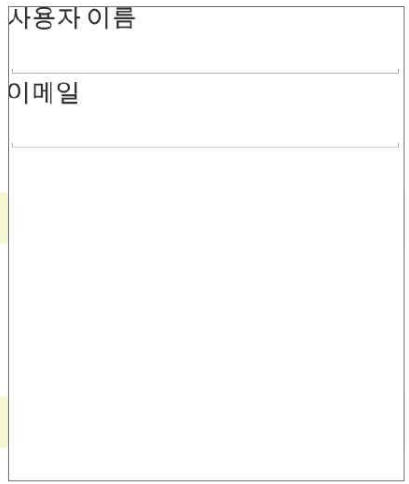

그림 7-13 레이아웃 XML 파일 생성 2

# 02 목록 대화상자

- (5) 대화상자용 dialog1.xml 파일을 코딩함
  - 텍스트뷰, 에디트텍스트, 텍스트뷰, 에디트텍스트의 순서로 생성
  - 에디트텍스트의 위젯 id : dlgEdt1, dlgEdt2

예제 7-22 dialog1.xml

```
1  <LinearLayout>
2      <TextView
3          android:text="사용자 이름" />
4      <EditText
5          android:id="@+id/dlgEdt1" />
6      <TextView
7          android:text="이메일" />
8      <EditText
9          android:id="@+id/dlgEdt2" />
10 </LinearLayout>
```



# 02 목록 대화상자

- (6) dialog1.xml과 같은 방식으로 toast1.xml 파일을 생성하고 코딩함. 먼저 토스트에 표시 될 그림 파일을 drawable 폴더에 복사해놓음
  - 이미지뷰, 텍스트뷰, 이미지뷰의 순서로 생성함
  - 텍스트뷰의 id : toastText1
  - 레이아웃의 배경을 빨간색으로 지정

예제 7-23 toast1.xml

```
1  <LinearLayout
2      android:background="#ff0000"
3      android:gravity="center" >
4      <ImageView
5          android:src="@drawable/btn_star_big_on" />
6      <TextView
7          android:id="@+id/toastText1"
8          android:textSize="20dp"
9          android:text="TextView" />
10     <ImageView
11         android:src="@drawable/btn_star_big_on" />
12 </LinearLayout>
```

# 02 목록 대화상자

- Kotlin 코드 작성 및 수정
  - (7) [app]-[java]-[패키지 이름]-[MainActivity]를 열기
  - (8) 다음 내용을 코딩
    - activity_main.xml의 텍스트뷰 2개, 버튼 1개에 대응할 전역변수 3개 선언
    - dialog1.xml의 에디트텍스트에 대응할 전역변수 2개 선언
    - toast1.xml의 텍스트뷰 1개에 대응할 전역변수 1개 선언
    - dialog1.xml과 toast1.xml을 인플레이트할 뷰 변수 2개 선언
    - 메인 함수 onCreate( ) 안에서 activity_main.xml의 위젯 변수 3개에 위젯 대입

# 02 목록 대화상자

예제 7-24 Kotlin 코드 1

```
1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3      lateinit var tvName : TextView
4      lateinit var tvEmail : TextView
5      lateinit var button1 : Button
6      lateinit var dlgEdtName : EditText
7      lateinit var dlgEdtEmail : EditText
8      lateinit var toastText : TextView
9      lateinit var dialogView : View
10     lateinit var toastView : View
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         setContentView(R.layout.activity_main)
15         title = "사용자 정보 입력"
16
17         tvName = findViewById<TextView>(R.id.tvName)
18         tvEmail = findViewById<TextView>(R.id.tvEmail)
19         button1 = findViewById<Button>(R.id.button1)
20
21     }
22 }
```

# 02 목록 대화상자

- (9) onCreate() 메소드 안에 코딩함
  - button1 변수를 클릭했을 때의 람다식을 작성
    - onClick() 메소드 안에 대화상자를 만들고 <확인>과 <취소>를 추가
    - <확인>, <취소>를 클릭했을 때 실행할 내용에는 일단 null을 입력
    - 실행한 후 버튼을 클릭하면 dialog1.xml의 내용이 대화상자로 나타남

예제 7-25 Kotlin 코드 2

```
1  button1.setOnClickListener {
2      dialogView = View.inflate(this@MainActivity, R.layout.dialog1, null)
3      var dlg = AlertDialog.Builder(this@MainActivity)
4      dlg.setTitle("사용자 정보 입력")
5      dlg.setIcon(R.drawable.ic_menu_allfriends)
6      dlg.setView(dialogView)
7      dlg.setPositiveButton("확인", null)
8      dlg.setNegativeButton("취소", null)
9      dlg.show()
10 }
```

# 02 목록 대화상자

- (10) 대화상자의 <확인>을 클릭하면 대화상자에서 입력한 사용자 이름과 이메일이 메인 화면 (activity_main.xml)의 텍스트뷰에 쓰이도록 코딩함
  - [예제 7-25]의 7행을 다음 코드로 대체

예제 7-26 Kotlin 코드 3

```
1  dlg.setPositiveButton("확인") { dialog, which ->
2      dlgEdtName = dialogView.findViewById<EditText>(R.id.dlgEdt1)
3      dlgEdtEmail = dialogView.findViewById<EditText>(R.id.dlgEdt2)
4
5      tvName.text = dlgEdtName.text.toString()
6      tvEmail.text = dlgEdtEmail.text.toString()
7  }
```

# 02 목록 대화상자

- (11) 대화상자의 <취소>를 클릭했을 때 toast1.xml이 토스트 메시지로 나오도록 설정
  - 역시 toast1.xml을 인플레이트하는 방식을 사용

예제 7-27 Kotlin 코드 4

```
1  dlg.setNegativeButton("취소") { dialog, which ->
2      var toast = Toast(this@MainActivity)
3      toastView = View.inflate(this@MainActivity, R.layout.toast1,null)
4      toastText = toastView .findViewById<TextView>(R.id.toastText1)
5      toastText.text = "취소했습니다"
6      toast.view = toastView
7      toast.show()
8  }
```


- 프로젝트 실행 및 결과 확인
  - (12) 완성된 코드를 실행하고 버튼을 클릭하여 대화상자와 토스트를 테스트하면 [그림 7-11]과 같음

# 02 목록 대화상자

▶ **직접 풀어보기 7-3**

[실습 7-3]을 다음과 같이 수정하라.

- activity_main.xml의 텍스트뷰를 에디트텍스트로 변경한다.
- 〈여기를 클릭〉을 클릭하면 activity_main.xml의 에디트텍스트 내용이 대화상자의 에디트텍스트에 나타난다.
- 대화상자에서 〈확인〉을 클릭하면 대화상자의 에디트텍스트 내용이 activity_main.xml의 에디트텍스트 내용으로 변경된다.
- 대화상자에서 〈취소〉를 클릭하면 토스트가 화면의 임의 위치에 나타난다.


그림 7-14 수정된 사용자 정보 입력 앱

# Thank You !