소프트웨어 실습 2

Chapter 5

2022.10

YangMin Lee

manson23@nate.com

S03(공대 1호관), 301-01호실 (yanwenry)

05 레이아웃 익히기

학습목표

- 레이아웃의 개념을 이해한다.
- 다양한 레이아웃으로 화면을 구성한다.
- Kotlin 코드만으로 화면을 작성한다.

차례

01 레이아웃의 기본개념

레이아웃 계층도

```
java.lang.Object
    └ android.view.View
        └ android.widget.ViewGroup
            └ android.widget.LinearLayout
                └ android.widget.TableLayout
            └ android.widget.RelativeLayout
            └ android.widget.FrameLayout
            └ android.widget.GridLayout
```

- 레이아웃은 ViewGroup 클래스로부터 상속받으며 내부에 무엇을 담는 용도로 쓰임
  - » 레이아웃 안에 존재하는 위젯을 배치하게 해줌

01 레이아웃의 기본개념

- 레이아웃의 대표적인 속성
  - **orientation** : 레이아웃 안에 배치할 위젯의 수직 또는 수평 방향을 설정
  - **gravity** : 레이아웃 안에 배치할 위젯의 정렬 방향을 좌측, 우측, 중앙 등으로 설정
  - **padding** : 레이아웃 안에 배치할 위젯의 여백을 설정
  - **layout_weight** : 레이아웃이 전체 화면에서 차지하는 공간의 가중값을 설정하는데, 여러 개의 레이아웃이 중복될 때 주로 사용
  - **baselineAligned** : 레이아웃 안에 배치할 위젯을 보기 좋게 정렬함
  - 레이아웃도 View 클래스의 하위 클래스이므로 View 클래스의 XML 속성과 메소드를 모두 사용할 수 있음.

01 레이아웃의 종류

- 자주 사용되는 레이아웃
  - 리니어레이아웃(LinearLayout)
  - 렐러티브레이아웃(RelativeLayout)
  - 프레임레이아웃(FrameLayout)
  - 테이블레이아웃(TableLayout)
  - 그리드레이아웃(GridLayout) 등

그림 5-1 레이아웃의 종류

01 레이아웃의 종류

- **리니어레이아웃(선형 레이아웃)**
  - 레이아웃의 왼쪽 위부터 아래쪽 또는 오른쪽으로 차례로 배치
- **렐러티브레이아웃(상대 레이아웃)**
  - 위젯 자신이 속한 레이아웃의 상하좌우 위치를 지정하여 배치하거나 다른 위젯으로부터 상대적인 위치를 지정
- **테이블레이아웃**
  - 행과 열의 개수를 지정한 테이블 형태로 위젯을 배열
- **그리드레이아웃**
  - 테이블레이아웃과 비슷하지만 행 또는 열을 확장하여 다양하게 배치할 때 더 편리함
- **프레임레이아웃**
  - 위젯을 왼쪽 위에 일률적으로 겹쳐서 배치하여 중복되어 보이는 효과를 낼 수 있음
  - 여러 개의 위젯을 배치한 후 상황에 따라서 필요한 위젯을 보이는 방식에 주로 활용

01 기본 리니어레이아웃의 형태

- **orientation 속성**
  - 리니어레이아웃의 가장 기본적인 속성
  - 값으로 vertical과 horizontal을 지정할 수 있음
  - vertical
    - » 리니어레이아웃 안에 포함될 위젯의 배치를 왼쪽 위부터 수직 방향으로 쌓음
  - horizontal
    - » 리니어레이아웃 안에 포함될 위젯의 배치를 왼쪽 위부터 수평 방향으로 쌓음

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    ~~~ 이곳에 위젯 배치 ~~~

</LinearLayout>
```

01 기본 리니어레이아웃의 형태

예제 5-1 orientation 속성이 vertical인 XML 코드

```
1  <LinearLayout
2      android:orientation="vertical" >
3      <Button
4          android:layout_width="wrap_content"
5          android:layout_height="wrap_content"
6          android:text="Button" />
7      <TextView
8          android:text="TextView" />
9      <CheckBox
10         android:text="CheckBox" />
11     <RadioButton
12         android:text="RadioButton" />
13     <Switch
14         android:text="Switch" />
15 </LinearLayout>
```

예제 5-2 orientation 속성이 horizontal인 XML 코드

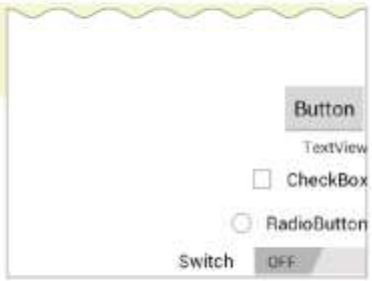
```
1  <LinearLayout
2      android:orientation="horizontal" >
3      <Button
4          android:layout_width="wrap_content"
5          android:layout_height="wrap_content"
6          android:text="Button" />
7      <TextView
8          android:text="TextView" />
9          ~~~ 생략 ~~~
10 </LinearLayout>
```

01 기본 리니어레이아웃의 형태

- **gravity와 layout_gravity 속성**
  - **gravity 속성**
    - » 레이아웃 안의 위젯을 어디에 배치할 것인지를 결정
    - » 값으로 left, right, center, top, bottom 등을 사용
    - » 2개를 조합하여 right|bottom처럼 사용할 수도 있음
      - » **right | bottom** : 오른쪽 아래에 정렬한다

예제 5-3 gravity 속성의 XML 코드

```
1  <LinearLayout
2      android:orientation="vertical"
3      android:gravity="right|bottom" >
4      <Button
5          android:layout_width="wrap_content"
6          android:layout_height="wrap_content"
7          android:text="Button" />
8      <TextView
9          android:text="TextView" />
10         ~~~ 생략 ~~~
11 </LinearLayout>
```



01 기본 리니어레이아웃의 형태

- **layout_ gravity 속성**
  - » gravity 속성이 자신에게 포함된 자식(주로 위젯)을 어디에 위치시킬지를 결정한다면, layout_ gravity 속성은 자신의 위치를 부모(주로 레이아웃)의 어디에 위치시킬지 결정함
  - » 그래서 gravity는 레이아웃에, layout_ gravity는 위젯에 주로 지정함

예제 5-4 layout_gravity 속성의 XML 코드

```
1  <LinearLayout
2      android:orientation="vertical" >
3      <Button
4          android:layout_gravity="right"
5          android:text="오른쪽" />
6      <Button
7          android:layout_gravity="center"
8          android:text="중앙" />
9      <Button
10         android:layout_gravity="left"
11         android:text="왼쪽" />
12 </LinearLayout>
```

01 기본 리니어레이아웃의 형태

▶ 직접 풀어보기 5-1

리니어레이아웃으로 다음 화면을 구성하는 XML을 작성하라.

- 리니어레이아웃의 orientation은 vertical로 한다.
- 버튼 3개를 생성하고 버튼의 layout_width는 110dp, layout_height는 100dp로 한다.

HINT 버튼에 gravity와 layout_gravity를 모두 설정해야 한다.

그림 5-2 리니어레이아웃 활용

01 기본 리니어레이아웃의 형태

- **baselineAligned 속성**
  - 크기가 다른 위젯들을 보기 좋게 정렬해주는 것으로 true와 false 값을 지정할 수 있음
    - [그림 5-3]의
      (a) : 리니어레이아웃에 baselineAligned를 false로 지정
      (b) : 생략하거나 true로 지정

(a) false로 지정　　(b) 생략하거나 true로 지정

그림 5-3 baselineAligned 속성

02 중복 리니어레이아웃의 형태

- **한 화면에서 위젯을 수평과 수직으로 다양하게 배치해야 하는 경우**
  - 리니어레이아웃 안에 리니어레이아웃을 생성하는 방식을 사용
    - 바깥의 큰 리니어레이아웃을 3개의 작은 리니어레이아웃으로 분류한 후 각 리니어레이아웃 안에 필요한 위젯을 배치하는 방식을 주로 사용

```
<LinearLayout>

    <LinearLayout>
        위젯…
    </LinearLayout>

    <LinearLayout>
        위젯…
    </LinearLayout>

    <LinearLayout>
        위젯…
    </LinearLayout>

</LinearLayout>
```

그림 5-4 다양한 배치의 레이아웃

02 중복 리니어레이아웃의 형태

- **layout_weight 속성**
  - 리니어레이아웃을 여러 개 사용할 경우 각 레이아웃의 크기를 지정해야 함
  - 레이아웃을 화면 전체에 채워야 하기 때문에 주로 전체 화면에 대한 비율(%)로 지정함

예제 5-5 3개의 레이아웃으로 구분한 XML 코드

```
1  <LinearLayout
2      android:layout_width="match_parent"
3      android:layout_height="match_parent"
4      android:orientation="vertical" >
5      <LinearLayout
6          android:layout_width="match_parent"
7          android:layout_height="match_parent"
8          android:gravity="center"
9          android:orientation="vertical" >
10         <Button
11             android:text="버튼1" />
12         <Button
13             android:text="버튼2" />
14     </LinearLayout>
15     <LinearLayout
16         android:layout_width="match_parent"
17         android:layout_height="match_parent"
18         android:background="#00FF00"
19         android:gravity="center"
20         android:orientation="horizontal" >
```

02 중복 리니어레이아웃의 형태

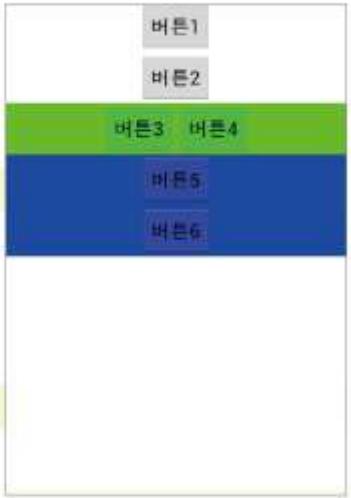
```
21          <Button
22              android:text="버튼3" />
23          <Button
24              android:text="버튼4" />
25      </LinearLayout>
26      <LinearLayout
27          android:layout_width="match_parent"
28          android:layout_height="match_parent"
29          android:background="#0000FF"
30          android:gravity="center"
31          android:orientation="vertical" >
32          <Button
33              android:text="버튼5" />
34          <Button
35              android:text="버튼6" />
36      </LinearLayout>
37  </LinearLayout>
```

02 중복 리니어레이아웃의 형태

- 첫 번째 레이아웃의 버튼만 보이는 것을 해결하기 위해 [예제 5-5]의 7행, 17행, 28행의 android:layout_height=“match_ parent”를 android:layout_height=wrap_ content”로 바꿈
- 그리고 레이아웃마다 구분되어 보이도록 내부에 있는 3개의 레이아웃에 background 속성을 지정함

예제 5-6 layout_height를 wrap_content로 변경

```
1  <LinearLayout
2      android:orientation="vertical" >
3      <LinearLayout
4          android:layout_width="match_parent"
5          android:layout_height="wrap_content"
6          ~~~ 생략 ~~~
7      </LinearLayout>
8      <LinearLayout
9          android:layout_width="match_parent"
10         android:layout_height="wrap_content"
11         ~~~ 생략 ~~~
12     </LinearLayout>
13     <LinearLayout
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         ~~~ 생략 ~~~
17     </LinearLayout>
18 </LinearLayout>
```



02 중복 리니어레이아웃의 형태

- 3개의 레이아웃이 모두 보이기는 하지만 화면을 꽉 채우지는 않음
- layout_weight 속성으로 각 레이아웃의 가중값을 지정
  - » layout_weight를 모두 1로 지정하여 세 레이아웃의 높이를 동일하도록 함

예제 5-7 layout_weight를 1로 지정

```
1  <LinearLayout
2      android:orientation="vertical" >
3      <LinearLayout
4          android:layout_width="match_parent"
5          android:layout_height="match_parent"
6          android:layout_weight="1"
7          ~~ 생략 ~~
8      </LinearLayout>
9      <LinearLayout
10         android:layout_width="match_parent"
11         android:layout_height="match_parent"
12         android:layout_weight="1"
13         ~~ 생략 ~~
14     </LinearLayout>
15     <LinearLayout
16         android:layout_width="match_parent"
17         android:layout_height="match_parent"
18         android:layout_weight="1"
19         ~~ 생략 ~~
20     </LinearLayout>
21 </LinearLayout>
```

02 중복 리니어레이아웃의 형태

▶ 직접 풀어보기 5-2

리니어레이아웃으로 다음 화면을 구성하는 XML을 작성하라. 단, 레이아웃이 구분되어 보이도록 각각 다른 색으로 지정한다.

HINT 레이아웃 안에 레이아웃을 여러 번 중첩해도 된다.

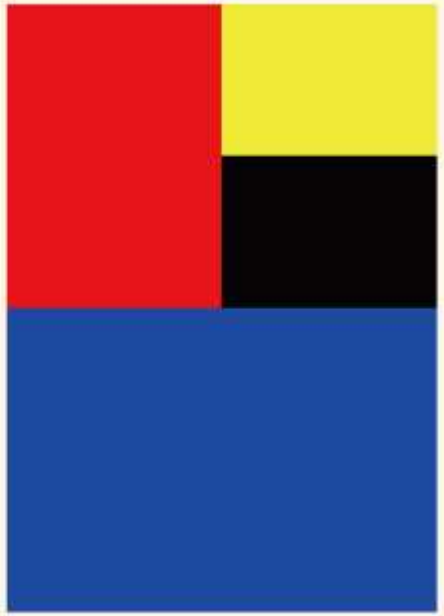


그림 5-5 중첩 리니어레이아웃

03 Kotlin 코드로 화면 만들기

- 지금까지 작성한 프로젝트는 기본적으로 activity_main.xml에서 화면을 구성한 후 Kotlin 파일(MainActivity.kt)의 setContentView() 메소드로 화면을 출력함

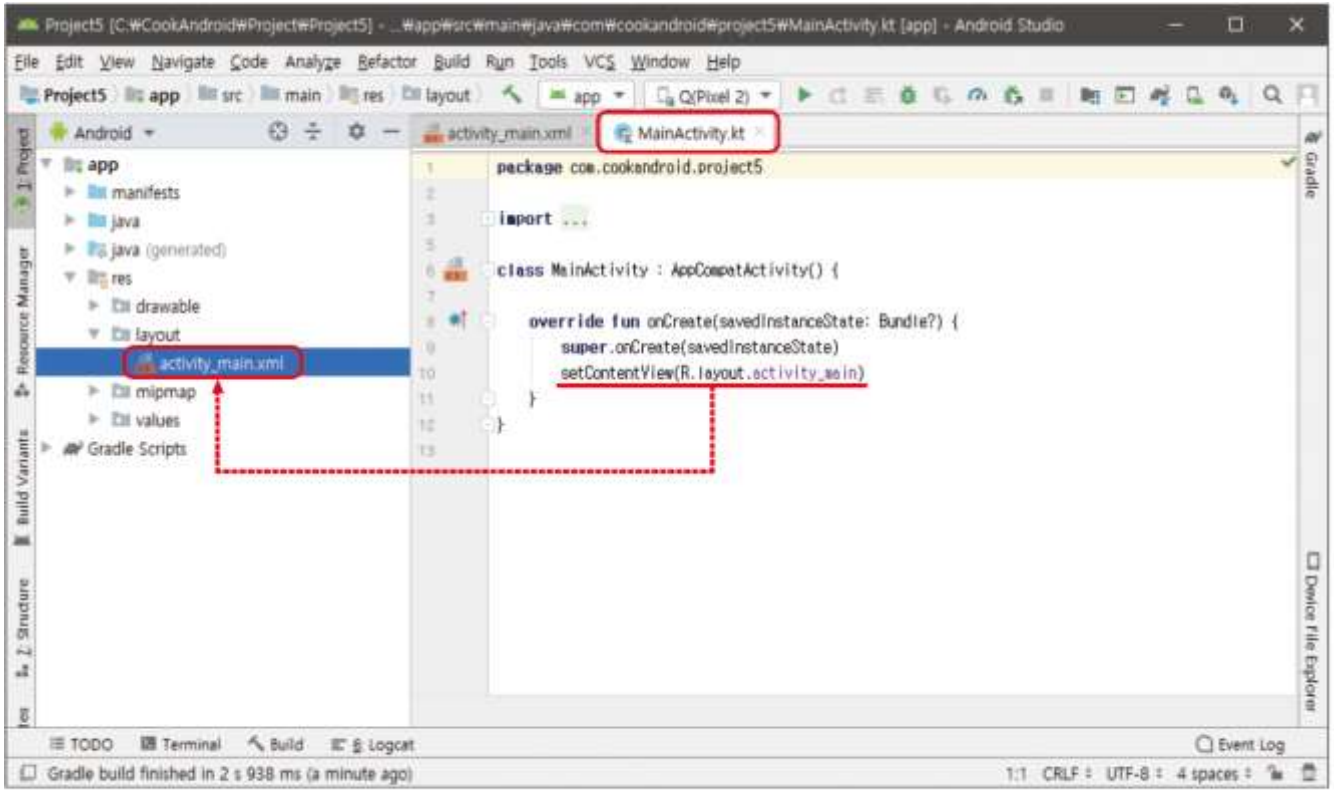


그림 5-6 XML과 Kotlin 코드의 동작

03 Kotlin 코드로 화면 만들기

- **<실습 5-1> XML 없이 화면 코딩하기**
  - 버튼을 클릭하면 토스트 메시지가 출력되는 화면을 Kotlin만으로 코딩하기

- 안드로이드 프로젝트 생성
  - (1) 새 프로젝트 만들기
    - » 프로젝트 이름 : 'Project5_1'
    - » 패키지 이름 : 'com.cookandroid. project5_1'
    - » 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

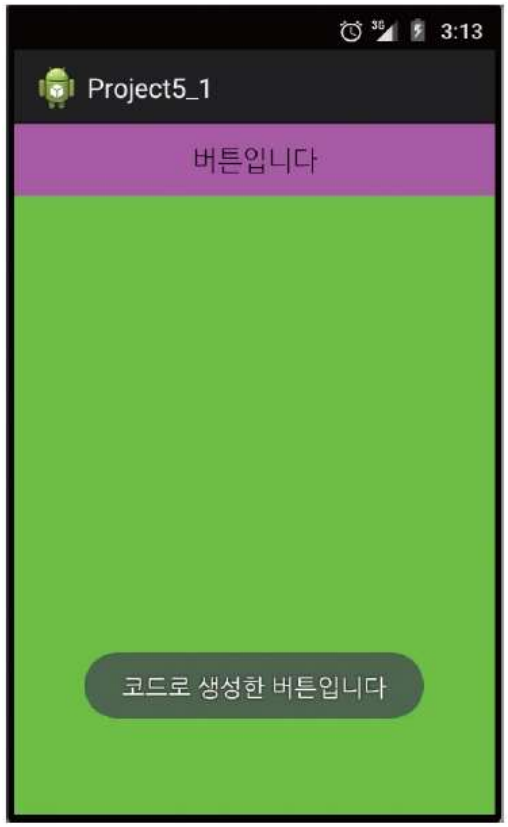


그림 5-7 Kotlin 코드만 사용한 프로젝트

03 Kotlin 코드로 화면 만들기

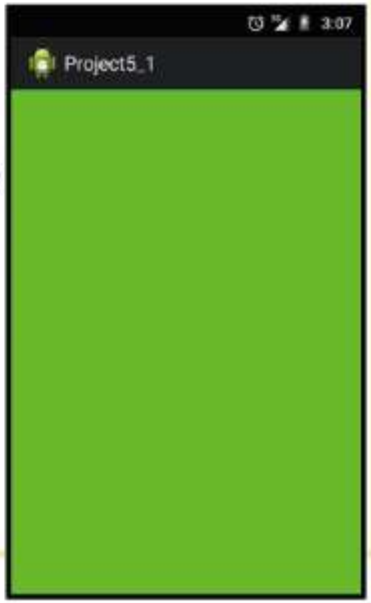
- 화면 디자인 및 편집
  - (2) 이번 실습에는 XML 파일이 필요 없음
  - Android Studio의 Project Tree에서 [app]-[res] -[layout]-[activity_main.xml]을 마우스 오른쪽 버튼으로 클릭하고 [Delete]를 선택하여 삭제함
    - » [Delete] 창이 나오면 체크 버튼이 모두 켜진 상태에서 <OK>를 클릭
    - » [Usage Detected] 창이 나오면 <Delete Anyway>를 클릭

- Kotlin 코드 작성 및 수정
  - (3) [app]-[java]-[패키지 이름]-[MainActivity]를 열기
    - » 앞에서 activity_main.xml 파일을 삭제했기 때문에 프로젝트를 실행하면 setContentView() 메소드에서 오류가 발생함
    - » 일단 오류가 발생한 행 앞에 //를 붙여 주석으로 처리한 후 다음 작업을 진행
  - (4) 리니어레이아웃을 생성하는 코드를 작성하고 실행

03 Kotlin 코드로 화면 만들기

예제 5-8 리니어레이아웃을 생성하는 Kotlin 코드

```
1  public override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      // setContentView(R.layout.activity_main);
4
5      val params = LinearLayout.LayoutParams(
6                   LinearLayout.LayoutParams.MATCH_PARENT,
7                   LinearLayout.LayoutParams.MATCH_PARENT)
8
9      val baseLayout = LinearLayout(this)
10     baseLayout.orientation = LinearLayout.VERTICAL
11     baseLayout.setBackgroundColor(Color.rgb(0, 255, 0))
12     setContentView(baseLayout, params)
13 }
```



03 Kotlin 코드로 화면 만들기

- (5) 버튼을 만들고, 버튼을 클릭했을 때의 토스트 메시지를 작성
  - » onCreate() 메소드 안에 이어서 코딩

예제 5-9 버튼을 생성하는 Kotlin 코드

```
1  var btn = Button(this)
2  btn.text = "버튼입니다"
3  btn.setBackgroundColor(Color.MAGENTA)
4  baseLayout.addView(btn)
5
6  btn.setOnClickListener {
7      Toast.makeText(applicationContext,
8          "코드로 생성한 버튼입니다", Toast.LENGTH_SHORT).show()
9  }
```

- 프로젝트 실행 및 결과 확인
  - (6) [Run As]-[Run 'app']을 선택하거나 [Run 'app'] 아이콘을 클릭하여 프로젝트를 실행하고 버튼을 클릭해 보기

03 Kotlin 코드로 화면 만들기

▶ 직접 풀어보기 5-3

다음 화면을 XML 파일 없이 Kotlin 코드만으로 작성하라.

- 레이아웃에 에디트텍스트 1개, 버튼 1개, 텍스트뷰 1개를 생성한다.
- 버튼을 클릭하면 에디트텍스트에 쓰인 문자열이 텍스트뷰에 나타나게 한다.

그림 5-8 XML 파일 없이 프로젝트 생성

01 렐러티브레이아웃

- **렐러티브레이아웃의 상하좌우에 배치**
  - 렐러티브레이아웃 안에 포함된 위젯의 속성 중 부모(레이아웃)의 어느 위치에 배치할지를 결정하는 속성은 모두 일곱 가지임.

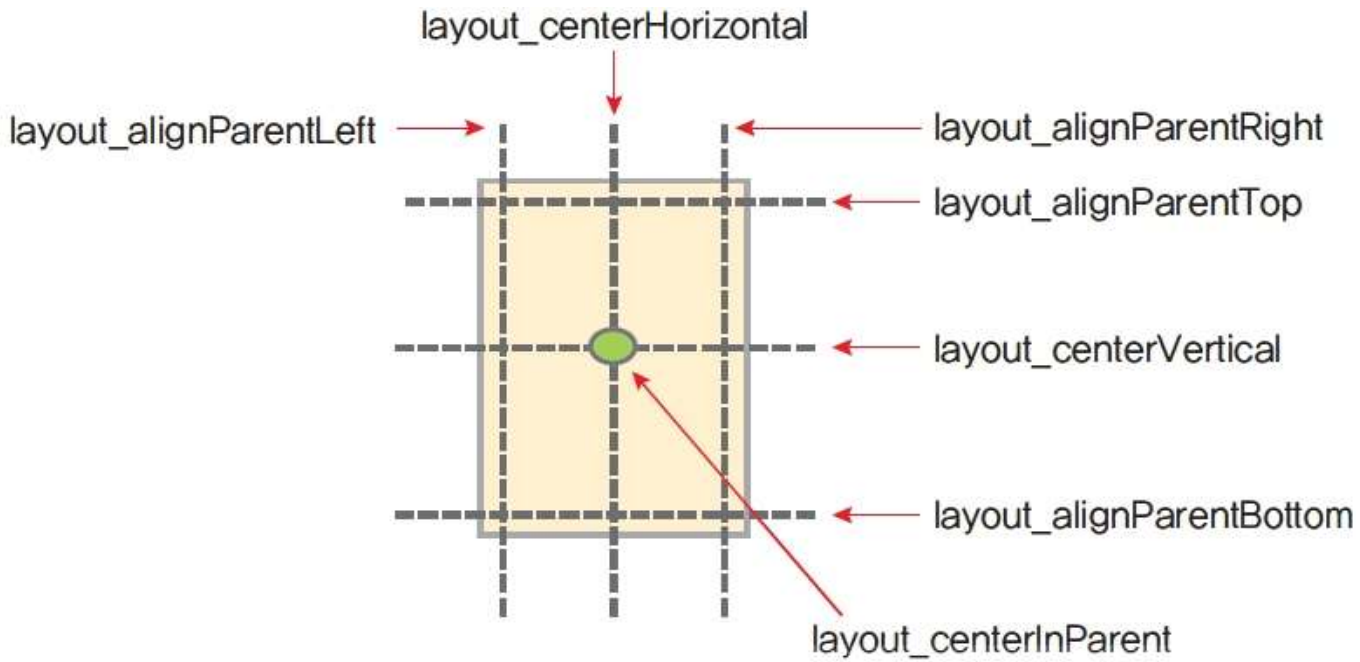


그림 5-9 부모(레이아웃)의 위치를 적용할 때의 속성

01 렐러티브레이아웃

예제 5-10 렐러티브레이아웃의 XML 코드 1

```
1  <RelativeLayout xmlns:android="http://~~~"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent" >
4      <Button
5          android:layout_alignParentTop="true"
6          android:layout_centerHorizontal="true"
7          android:text="위쪽" />
8      <Button
9          android:layout_alignParentLeft="true"
10         android:layout_centerVertical="true"
11         android:text="좌측" />
12     <Button
13         android:layout_centerInParent="true"
14         android:text="중앙" />
15     <Button
16         android:layout_alignParentRight="true"
17         android:layout_centerVertical="true"
18         android:text="우측" />
19     <Button
20         android:layout_alignParentBottom="true"
21         android:layout_centerHorizontal="true"
22         android:text="아래" />
23 </RelativeLayout>
```

01 렐러티브레이아웃

▪ 다른 위젯의 상대 위치에 배치

- 렐러티브레이아웃 안에서 다른 위젯의 특정한 곳에 배치하는 방법도 있음
- 각 속성의 값에는 다른 위젯의 아이디를 지정하면 되는데 “@+id/기준 위젯의 아이디”와 같은 형식으로 사용함
- 다른 위젯과 관련된 속성은 다음 그림과 같음

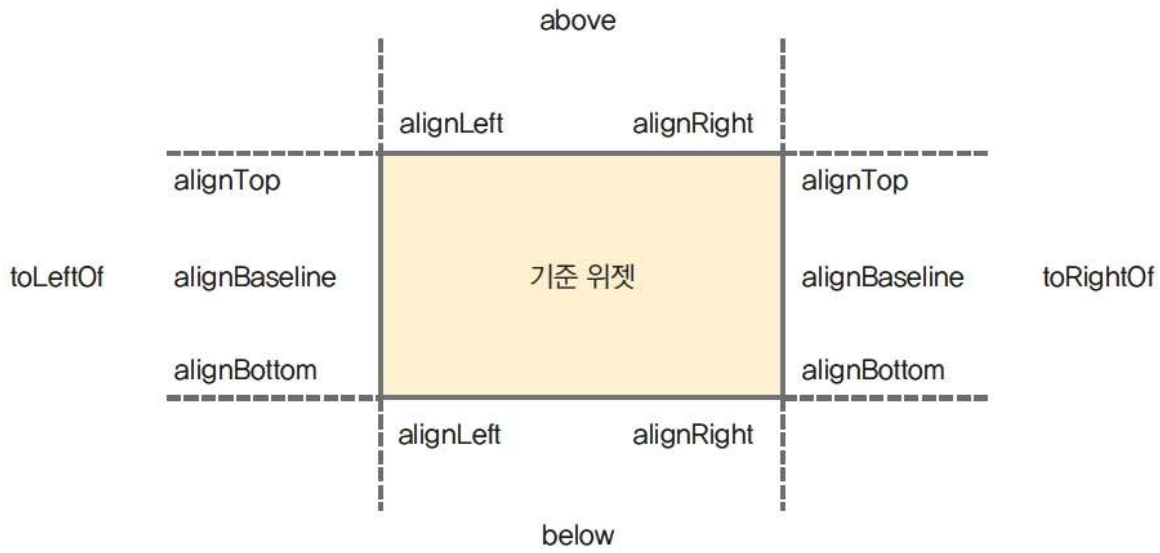


그림 5-10 다른 위젯의 상대적인 위치를 적용할 때의 속성

01 렐러티브레이아웃

예제 5-11 렐러티브레이아웃의 XML 코드 2

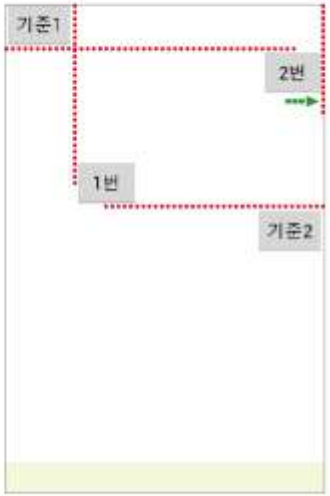
```
1  <RelativeLayout xmlns:android="http://~~~"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent" >
4      <Button
5          android:id="@+id/baseBtn"
6          android:layout_width="150dp"
7          android:layout_height="150dp"
8          android:layout_centerHorizontal="true"
9          android:layout_centerVertical="true"
10         android:text="기준위젯" />
11     <Button
12         android:layout_alignTop="@+id/baseBtn"
13         android:layout_toLeftOf="@+id/baseBtn"
14         android:text="1번" />
15         ~~~ 생략(버튼 2개) ~~~
16     <Button
17         android:layout_above="@+id/baseBtn"
18         android:layout_alignLeft="@+id/baseBtn"
19         android:text="4번" />
20     <Button
21         android:layout_alignRight="@+id/baseBtn"
22         android:layout_below="@+id/baseBtn"
23         android:text="5번" />
24     <Button
25         android:layout_above="@+id/baseBtn"
26         android:layout_toRightOf="@+id/baseBtn"
27         android:text="6번" />
28 </RelativeLayout>
```

01 렐러티브레이아웃

- [예제 5-12]는 여러 위젯에 대한 상대적인 위치를 지정하는 방식을 보여줌
  - » <1번>은 <기준1>의 오른쪽과 <기준2>의 위쪽에 맞춰서 배치했으며, <2번>은 <기준1>의 아래쪽과 레이아웃의 오른쪽

예제 5-12 렐러티브레이아웃 속성을 조합한 XML 코드

```
1  <RelativeLayout xmlns:android="http://~~~"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent" >
4      <Button
5          android:id="@+id/baseBtn1"
6          android:layout_alignParentLeft="true"
7          android:layout_alignParentTop="true"
8          android:text="기준1" />
9      <Button
10         android:id="@+id/baseBtn2"
11         android:layout_alignParentRight="true"
12         android:layout_centerVertical="true"
13         android:text="기준2" />
14     <Button
15         android:layout_above="@+id/baseBtn2"
16         android:layout_toRightOf="@+id/baseBtn1"
17         android:text="1번" />
18     <Button
19         android:layout_alignParentRight="true"
20         android:layout_below="@+id/baseBtn1"
21         android:text="2번" />
22 </RelativeLayout>
```



01 렐러티브레이아웃

▶ 직접 풀어보기 5-4

다음 화면의 XML 코드를 중복 리니어레이아웃과 렐러티브레이아웃으로 각각 작성하라. 텍스트뷰 1개, 에디트텍스트 1개, 버튼 2개로 구성한다.

그림 5-11 레이아웃 비교

02 테이블레이아웃

- **테이블레이아웃**
  - 위젯을 표형태로 배치할 때 주로 사용
  - <TableRow>와 함께 사용
    - 행의 수 : <TableRow>의 수
    - 열의 수 : <TableRow> 안에 포함된 위젯의 수

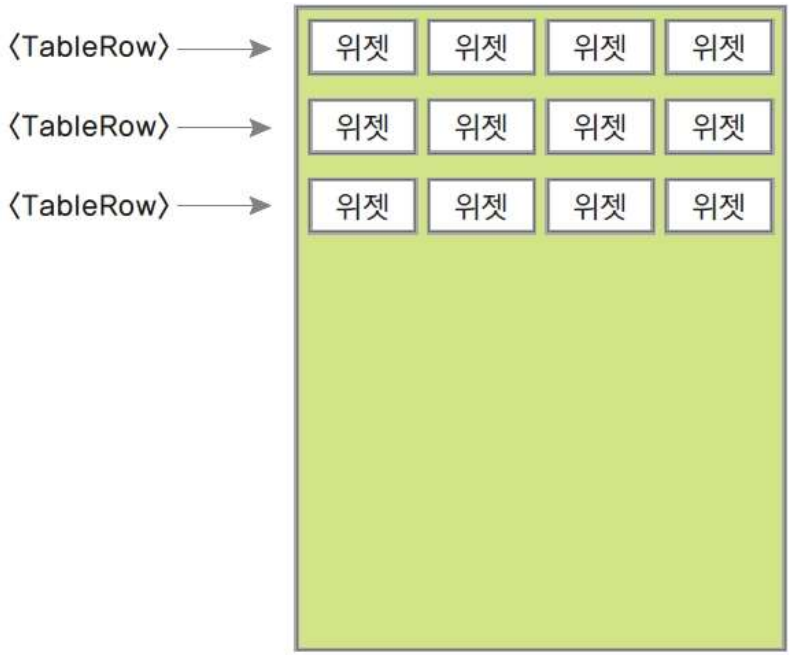


그림 5-12 테이블레이아웃의 개요

02 테이블레이아웃

- **테이블레이아웃의 속성**
  - **layout_span**
    - 테이블레이아웃 안에 포함된 위젯에 설정하는 속성
    - 열을 합쳐서 표시하라는 의미
      - » 예를 들어 layout_span="2"는 현재 셀부터 2개의 셀을 합쳐서 표시함
  - **layout_column**
    - 테이블레이아웃 안에 포함된 위젯에 설정하는 속성
    - 지정된 열에 현재 위젯을 표시
  - **stretchColumns**
    - <TableLayout> 자체에 설정하는 속성
    - 지정된 열의 너비를 늘리라는 의미
      - » stretchColumns="*"는 각 셀을 모두 같은 크기로 확장하여 전체 화면이 꽉 차는 효과를 냄
      - » 열 번호는 0번부터 시작

02 테이블레이아웃

예제 5-13 테이블레이아웃의 XML 코드

```
1  <TableLayout xmlns:android="http://~~~"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent" >
4
5      <TableRow>
6          <Button
7              android:layout_width="60dp"
8              android:text="1" />
9          <Button
10             android:layout_width="60dp"
11             android:layout_span="2"
12             android:text="2" />
13         <Button
14             android:layout_width="60dp"
15             android:text="3" />
16     </TableRow>
17
18     <TableRow>
19         <Button
20             android:layout_width="60dp"
21             android:layout_column="1"
22             android:text="4" />
23         <Button
24             android:layout_width="60dp"
25             android:text="5" />
26         <Button
27             android:layout_width="60dp"
28             android:text="6" />
29     </TableRow>
30 </TableLayout>
```

02 테이블레이아웃

<실습 5-2> 테이블레이아웃을 활용한 계산기 앱 만들기

- 테이블레이아웃을 활용하여 숫자 버튼도 있는 계산기 만들기
- 비슷한 형태의 버튼이 여러 개 나오는데, Kotlin 코드로 버튼을 배열로 처리하는 기법도 함께 익힘

안드로이드 프로젝트 생성

- (1) 새 프로젝트 만들기
  - » 프로젝트 이름 : ‘Project5_2’
  - » 패키지 이름 : ‘com.cookandroid.project5_
  - » 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

그림 5-13 테이블레이아웃 계산기 앱 결과 화면

02 테이블레이아웃

- 화면 디자인 및 편집
  - (2) [app]-[res]-[layout]-[activity_main.xml]을 열고, 아래쪽의 [Text] 탭을 클릭하여 화면을 코딩. 다음과 같이 화면을 구성
    - » TableLayout 1개와 TableRow 9개로 구성
    - » 에디트텍스트 2개, 숫자 버튼 10개, 연산 버튼 4개, 텍스트뷰 1개를 생성
    - » 각 연산 버튼 위젯에 layout_margin을 적절히 지정(예: 5dp)
    - » 결과를 보여줄 TextView는 색상을 빨간색으로, 글자 크기를 20dp 정도로 지정
    - » 각 위젯의 id는 위에서부터 차례로 Edit1, Edit2, BtnNum0~9, BtnAdd, BtnSub, BtnMul, BtnDiv, TextResult로 지정

02 테이블레이아웃

예제 5-14 activity_main.xml

```
1  <TableLayout xmlns:android="http://~~~"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent" >
4
5      <TableRow>
6          <EditText
7              android:id="@+id/Edit1"
8              android:layout_span="5"
9              android:hint="숫자1 입력" />
10     </TableRow>
11
12     ~~~ 생략(TableRow 1개, EditText 1개) ~~~
13
14     <TableRow>
15         <Button
16             android:id="@+id/BtnNum0"
17             android:text="0" />
18         ~~~ 생략(숫자 Button 4개) ~~~
19     </TableRow>
20
21     <TableRow>
22         ~~~ 생략(숫자 Button 5개) ~~~
23     </TableRow>
24
25     <TableRow>
26         <Button
27             android:id="@+id/BtnAdd"
28             android:layout_margin="5dp"
29             android:layout_span="5"
30             android:text="더하기" />
31     </TableRow>
32
33     ~~~ 생략(TableRow 3개, 연산 Button 3개) ~~~
34
35     <TableRow>
36         <TextView
37             android:id="@+id/TextResult"
38             android:layout_margin="5dp"
39             android:layout_span="5"
40             android:text="계산 결과 : "
41             android:textColor="#FF0000"
42             android:textSize="20dp" />
43     </TableRow>
44 </TableLayout>
```

02 테이블레이아웃

- Kotlin 코드 작성 및 수정
  - (3) [app]-[java]-[패키지 이름]-[MainActivity]를 열기
  - (4) 다음과 같은 변수를 전역변수로 선언
    - » 숫자 버튼을 제외한 activity_main.xml의 7개 위젯에 대응할 위젯 변수 7개
    - » 입력될 2개 문자열을 저장할 문자열 변수 2개
    - » 계산 결과를 저장할 정수 변수 1개
    - » 10개 숫자 버튼을 저장할 버튼 배열
    - » 10개 버튼의 id를 저장할 정수형 배열
    - » 증가값으로 사용할 정수 변수

02 테이블레이아웃

예제 5-15 Kotlin 코드 1

```
1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3      lateinit internal var edit1 : EditText
4      lateinit internal var edit2 : EditText
5      lateinit internal var btnAdd : Button
6      lateinit internal var btnSub : Button
7      lateinit internal var btnMul : Button
8      lateinit internal var btnDiv : Button
9      lateinit internal var textResult : TextView
10     lateinit internal var num1 : String
11     lateinit internal var num2 : String
12     internal var result : Int? = null
13     internal var numButtons  = ArrayList<Button>(10)
14     internal var numBtnIDs = arrayOf(R.id.BtnNum0, R.id.BtnNum1, R.id.BtnNum2,
15                         R.id.BtnNum3, R.id.BtnNum4, R.id.BtnNum5, R.id.BtnNum6,
16                         R.id.BtnNum7, R.id.BtnNum8, R.id.BtnNum9)
17     internal var I : Int = 0 // 증가값 용도
18
19     override fun onCreate(savedInstanceState: Bundle?) {
20         ~~~ 생략 ~~~
```

02 테이블레이아웃

- (5) 메인 함수 onCreate() 내부를 코딩함
  - » 숫자 버튼이 없다고 가정하고 연산 버튼을 터치했을 때의 내용을 코딩
  - » 숫자 버튼이 없는 상태의 계산기 코딩은 이미 [실습 4-1]에서 [illegible]일함(다음 예제는

예제 5-16 Kotlin 코드 2

```
1  ~~~ 생략 ~~~
2  override fun onCreate(savedInstanceState: Bundle?) {
3      super.onCreate(savedInstanceState)
4      setContentView(R.layout.activity_main)
5
6      title = "테이블레이아웃 계산기"
7
8      edit1 = findViewById<EditText>(R.id.Edit1)
9      edit2 = findViewById<EditText>(R.id.Edit2)
10     btnAdd = findViewById<Button>(R.id.BtnAdd)
11     ~~~ 생략(연산 버튼 3개 대입) ~~~
12     textResult = findViewById<TextView>(R.id.TextResult)
13
14     btnAdd.setOnTouchListener { view, motionEvent ->
15         num1 = edit1.text.toString()
16         num2 = edit2.text.toString()
17         result = Integer.parseInt(num1) + Integer.parseInt(num2)
18         textResult.text = "계산 결과 : " + result.toString()
19         false
20     }
21     ~~~ 생략(연산 버튼 3개 터치 람다식) ~~~
22 }
```

02 테이블레이아웃

- (6) 지금까지 작성한 프로젝트를 실행해봄
  - » 숫자 버튼을 클릭하는 동작을 제외하고 에디트텍스트에 숫자를 직접 입력하면 잘 동작함
  - » 현재까지의 결과는 [실습 4-1]과 동일함

- (7) 숫자 버튼 10개를 배열 변수에 대입한 후 각 버튼의 클릭 이벤트 람다식을 만들어봄
  - » 원칙적으로 10개 각 버튼에 대해 코딩해야 하지만 코드가 길어지기 때문에 배열을 작성함
  - » 배열 처리이므로 for 문이나 while 문을 사용함
    - » 다음 코드를 onCreate( ) 안에 작성

02 테이블레이아웃

예제 5-17 Kotlin 코드 3

```
1  for (i in 0..9 step 1) {
2      numButtons.add(findViewById<Button>(numBtnIDs[i]))
3  }
4
5  for (i in 0..numBtnIDs.size-1 step 1) {
6      numButtons[i].setOnClickListener {
7          if (edit1.isFocused == true) {
8              num1 = edit1.text.toString() + numButtons[i].getText().toString()
9              edit1.setText(num1)
10         } else if (edit2.isFocused == true) {
11             num2 = edit2.text.toString() + numButtons[i].getText().toString()
12             edit2.setText(num2)
13         } else {
14             Toast.makeText(applicationContext,
15                 "먼저 에디트텍스트를 선택하세요", Toast.LENGTH_SHORT).show()
16         }
17     }
18 }
```

02 테이블레이아웃

- **프로젝트 실행 및 결과 확인**
  - (8) 완성된 코드를 실행하면 계산기가 정상적으로 동작함
  - 에디트텍스트에 포커스가 없는 상태에서 숫자 버튼을 클릭하면 메시지가 나타남

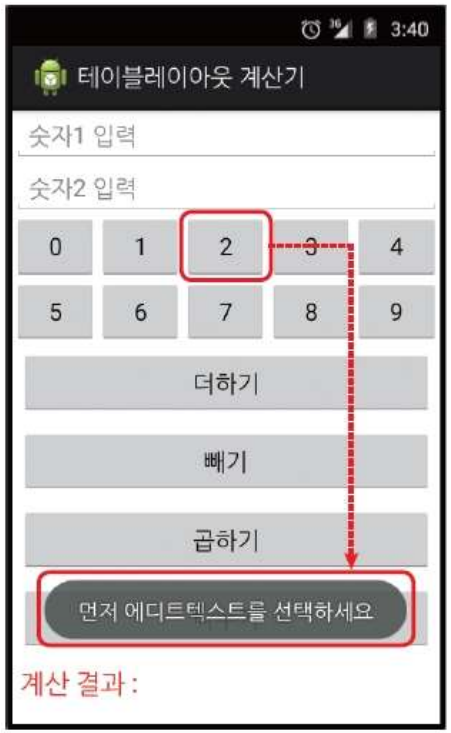


그림 5-14 에디트텍스트에 포커스가 없는 상태에서 클릭

03 그리드레이아웃

- **그리드레이아웃**
  - 테이블레이아웃과 마찬가지로 위젯을 표 형태로 배치할 때 사용하지만 좀 더 직관적임
  - 테이블레이아웃에서는 다소 어려웠던 행 확장도 간단하게 할 수 있어서 유연한 화면 구성에 적합함
    - » 2 행 3열을 지정하려면 layout_row 속성은 1로, layout_column 속성은 2로 설정(0부터 시작한다)

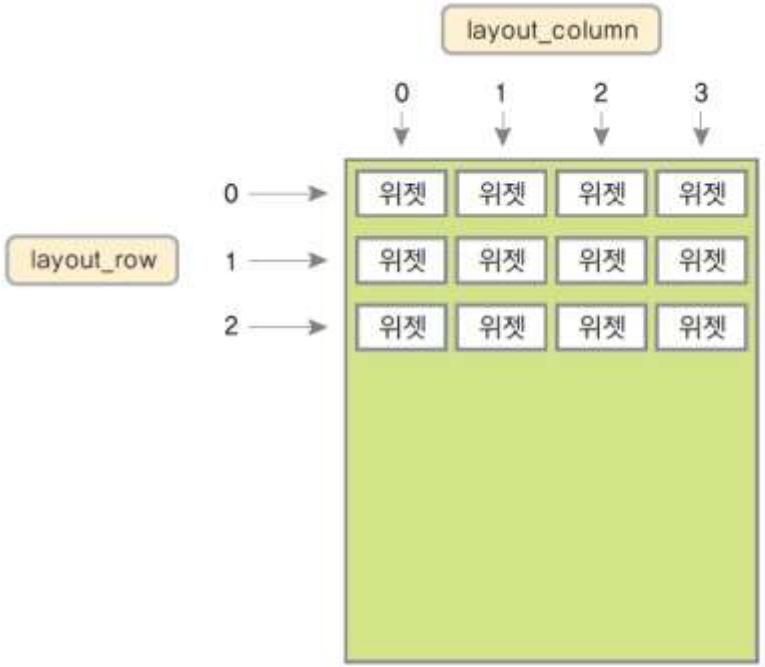


그림 5-15 그리드레이아웃의 개요

03 그리드레이아웃

- **그리드레이아웃의 속성**
  - <GridLayout> 자체에 자주 사용되는 속성
    - **rowCount**: 행의 수
    - **columnCount**: 열의 수
    - **orientation**: 그리드를 수평 방향으로 우선할 것인지, 수직 방향으로 우선할 것인지를 결정
    - **layout_row**: 자신이 위치할 행 번호(0번부터 시작)
    - **layout_column**: 자신이 위치할 열 번호(0번부터 시작)
    - **layout_rowSpan**: 행을 지정된 수만큼 확장함
    - **layout_columnSpan**: 열을 지정된 수만큼 확장함
    - layout_ **gravity**: 주로 fill, fill_vertical, fill_horizontal 등으로 지정
      - » layout_rowSpan이나 layout_ columnSpan으로 행 또는 열이 확장되었을 때 위젯을 확장된 셀에 꽉 채우는 효과를 냄

03 그리드레이아웃

예제 5-18 그리드레이아웃의 XML 코드

```
1  <GridLayout xmlns:android="http://~~~"
2      android:columnCount="4"
3      android:rowCount="2" >
4      <Button
5          android:layout_column="0"
6          android:layout_row="0"
7          android:layout_rowSpan="2"
8          android:layout_gravity="fill_vertical"
9          android:text="1" />
10     <Button
11         android:layout_column="1"
12         android:layout_row="0"
13         android:layout_columnSpan="2"
14         android:layout_gravity="fill_horizontal"
15         android:text="2" />
16     <Button
17         android:layout_column="3"
18         android:layout_row="0"
19         android:text="3" />
20     <Button
21         android:layout_column="1"
22         android:layout_row="1"
23         android:text="4" />
24     ~~~ 생략 ~~~
25 </GridLayout>
```

03 그리드레이아웃

▶ **직접 풀어보기 5-5**

[실습 5-2]를 그리드레이아웃으로 변경하여 실행하라.

HINT Kotlin 코드는 고칠 필요가 없고 XML만 변경하면 된다. XML 위젯의 id도 동일하게 사용한다.

그림 5-16 그리드레이아웃 계산기 앱

04 프레임레이아웃

- **프레임레이아웃**
  - 단순히 레이아웃 안의 위젯을 왼쪽 상단부터 겹쳐서 출력함
  - 프레임레이아웃 그 자체를 사용하기보다는 탭위젯 등과 혼용할 때 유용함

그림 5-17 프레임레이아웃의 개요

04 프레임레이아웃

- **프레임레이아웃의 속성**
  - <FrameLayout>에서 가끔 사용하는 속성
    - **foreground**: 프레임레이아웃의 전경 이미지를 지정함.
    - **foregroundGravity**: 전경 이미지의 위치를 지정함.
      - » fill, right, left, top, bottom 등의 값을 사용함.

예제 5-19 프레임레이아웃의 XML 코드

```
1  <FrameLayout xmlns:android="http://~~~"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent"
4      android:foreground="@drawable/dog"
5      android:foregroundGravity="center|fill_horizontal" >
6      <RatingBar
7          android:id="@+id/ratingBar1" >
8      <ImageView
9          android:src="@drawable/ic_launcher" />
10     <CheckBox
11         android:text="CheckBox" />
12 </FrameLayout>
```

Thank You !