5. 뷰를 배치하는 레이아웃

1. 선형으로 배치 – LinearLayout
2. 상대 위치로 배치 – RelativeLayout
3. 겹쳐서 배치 - FrameLayout
4. 표 형태로 배치 – GridLayout
5. 계층 구조로 배치 – ConstraintLayout
6. 전화 앱의 키패드 화면 만들기

1. 선형으로 배치-LinearLayout

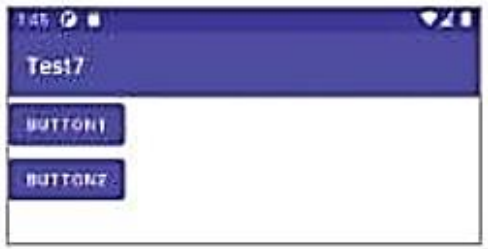
LinearLayout 배치 규칙

- LinearLayout은 뷰를 가로나 세로 방향으로 나열하는 레이아웃
- orientation 속성에 horizontal이나 vertical값으로 방향을 지정

• LinearLayout의 방향 속성 설정

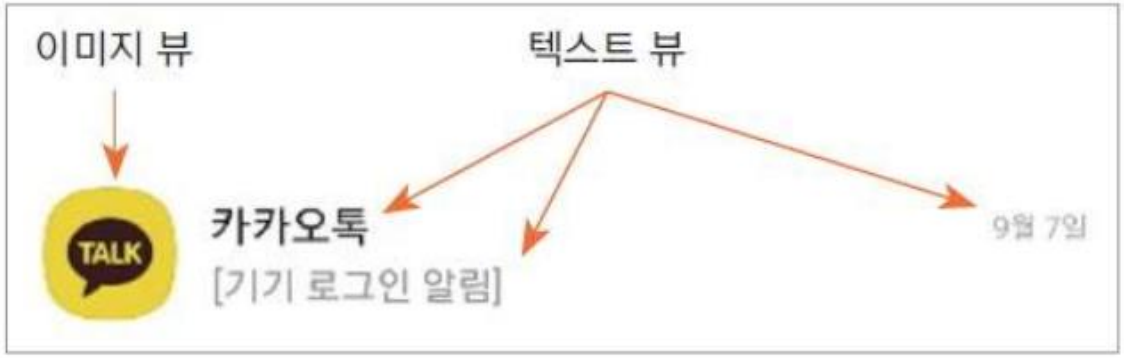
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BUTTON1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BUTTON2" />
</LinearLayout>
```

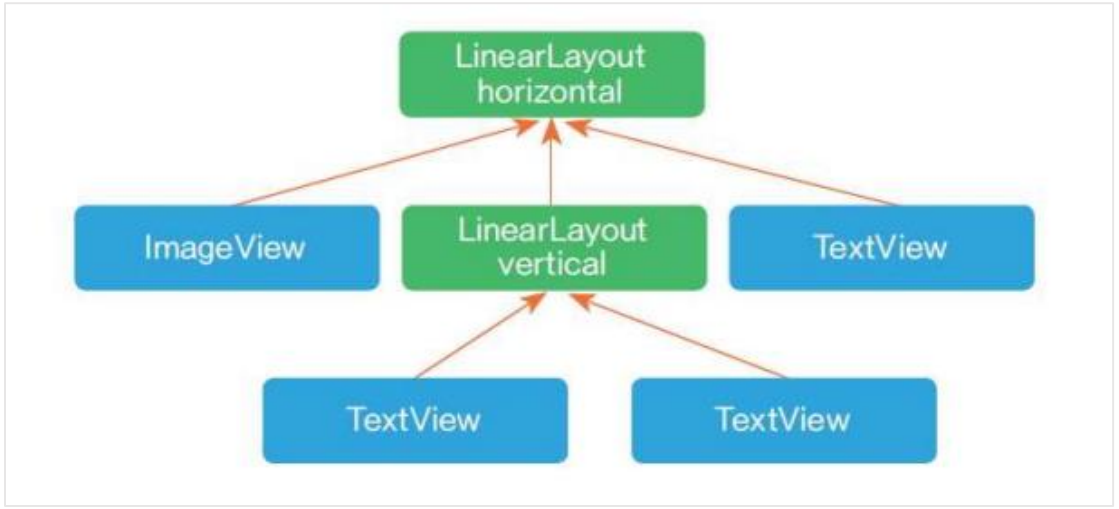
▶ 실행 결과



1. 선형으로 배치-LinearLayout

- LinearLayout을 중첩
- 레이아웃 클래스도 뷰이므로 다른 레이아웃 클래스에 포함할 수 있다.





• LinearLayout 중첩

```
<LinearLayout (...생략...)
    android:orientation="horizontal">
    <ImageView (...생략...) />
    <LinearLayout (...생략...)
        android:orientation="vertical">
        <TextView (...생략...) />
        <TextView (...생략...) />
    </LinearLayout>
    <TextView (...생략...) />
</LinearLayout>
```

▶ 실행 결과

1. 선형으로 배치-LinearLayout

- **여백을 채우는 layout_weight 속성**
  - 뷰 1개로 전체 여백 채우기
    - 여백을 뷰로 채우려면 layout_weight 속성을 사용

• 여백 없이 채우는 weight 속성

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BUTTON1"
        android:layout_weight="1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BUTTON2" />
</LinearLayout>
```

▶ 실행 결과

1. 선형으로 배치-LinearLayout

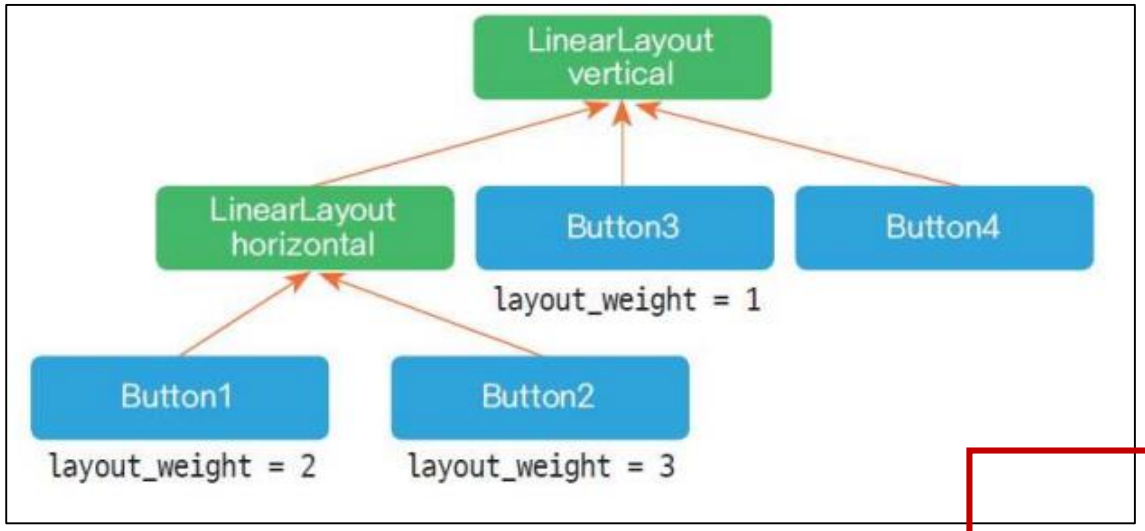
- **뷰 여러 개로 여백을 나누어 채우기**
  - layout_weight 속성에 지정한 숫자는 가중치
  - layout_weight값을 각각 1과 3으로 선언했다면 가로 여백을 각각 1/4만큼, 3/4만큼 나누어 차지

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="BUTTON1"
    android:layout_weight="1" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="BUTTON2"
    android:layout_weight="3" />
```

▶ 실행 결과

1. 선형으로 배치-LinearLayout

- 중첩된 레이아웃에서 여백 채우기



• 중첩된 레이아웃에서 여백 채우기

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

    ...

</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="BUTTON1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:text="BUTTON2" />
</LinearLayout>
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="BUTTON3"
    android:layout_weight="1" />
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="BUTTON4" />
```

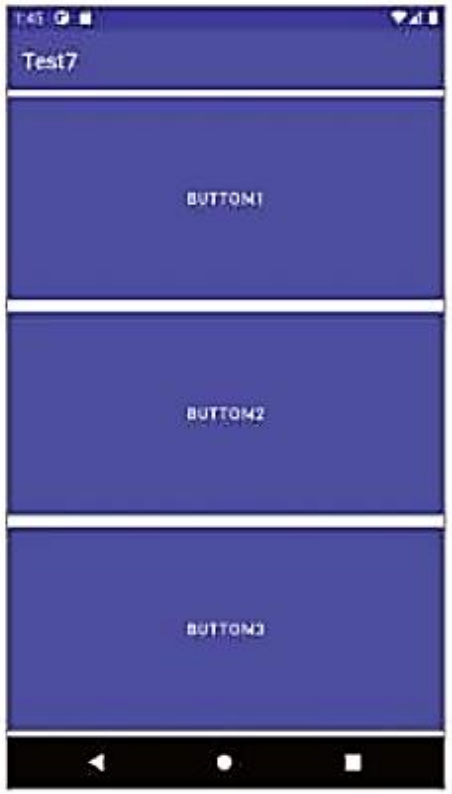
▶ 실행 결과

1. 선형으로 배치-LinearLayout

- **여백 채우기로 뷰의 크기 설정하기**
  - layout_weight 값으로 뷰크기 설정하기

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button1" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button2" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button3" />
</LinearLayout>
```

▶ 실행 결과



1. 선형으로 배치-LinearLayout

- **뷰를 정렬하기**
  - gravity, layout_gravity 속성
  - 뷰에 gravity와 layout_gravity 속성 적용하기
    - gravity 속성의 정렬 대상은 콘텐츠
    - layout_gravity는 뷰 자체를 정렬하는 속성

• 뷰 정렬하기

```
<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="150dp"
        android:layout_height="150dp"
        android:background="#FF0000"
        android:textSize="15dp"
        android:textStyle="bold"
        android:textColor="#FFFFFF"
        android:text="HelloWorld"
        android:gravity="right|bottom"
        android:layout_gravity="center_horizontal" />
</LinearLayout>
```

▶ 실행 결과

1. 선형으로 배치-LinearLayout

- **gravity 속성 적용하기**

• 화면 가운데로 정렬

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">
    <TextView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:background="#FF0000"
        android:textSize="15dp"
        android:textStyle="bold"
        android:textColor="#FFFFFF"
        android:text="HelloWorld"
        android:gravity="right|bottom" />
</LinearLayout>
```

▶ 실행 결과

2. 상대 위치로 배치 - RelativeLayout

- **RelativeLayout 배치 규칙**
  - 상대 뷰의 위치를 기준으로 정렬하는 레이아웃
    - android:layout_above: 기준 뷰의 위쪽에 배치
    - android:layout_below: 기준 뷰의 아래쪽에 배치
    - android:layout_toLeftOf: 기준 뷰의 왼쪽에 배치
    - android:layout_toRightOf: 기준 뷰의 오른쪽에 배치

• 상대 뷰의 오른쪽에 배치

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:id="@+id/testImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="HELLO"
        android:layout_toRightOf="@id/testImage" />
</RelativeLayout>
```

▶ 실행 결과

2. 상대 위치로 배치 - RelativeLayout

- **맞춤 정렬하는 align 속성**
  - 상대 뷰의 어느 쪽에 맞춰서 정렬할지를 정하는 속성
  - android:layout_alignTop: 기준 뷰와 위쪽을 맞춤
  - android:layout_alignBottom: 기준 뷰와 아래쪽을 맞춤
  - android:layout_alignLeft: 기준 뷰와 왼쪽을 맞춤
  - android:layout_alignRight: 기준 뷰와 오른쪽을 맞춤
  - android:layout_alignBaseline: 기준 뷰와 텍스트 기준선을 맞춤

• 기준 뷰와 아래쪽을 맞춰 정렬

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:id="@+id/testImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello"
        android:layout_toRightOf="@id/testImage"
        android:layout_alignBottom="@id/testImage" />
</RelativeLayout>
```

▶ 실행 결과

2. 상대 위치로 배치 - RelativeLayout

- **상위 레이아웃을 기준으로 맞춤 정렬하는 속성**
  - android:layout_alignParentTop: 부모의 위쪽에 맞춤
  - android:layout_alignParentBottom: 부모의 아래쪽에 맞춤
  - android:layout_alignParentLeft: 부모의 왼쪽에 맞춤
  - android:layout_alignParentRight: 부모의 오른쪽에 맞춤
  - android:layout_centerHorizontal: 부모의 가로 방향 중앙에 맞춤
  - android:layout_centerVertical: 부모의 세로 방향 중앙에 맞춤
  - android:layout_centerInParent: 부모의 가로·세로 중앙에 맞춤

• 부모의 오른쪽에 맞춰 정렬

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView

        android:id="@+id/testImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="HELLO"
        android:layout_alignBottom="@id/testImage"
        android:layout_alignParentRight="true" />
</RelativeLayout>
```

▶ 실행 결과

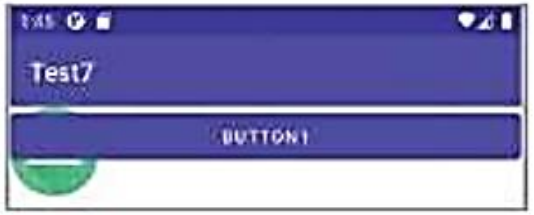
3. 겹쳐서 배치 - FrameLayout

- **FrameLayout 배치 규칙**
  - 카드를 쌓듯이 뷰를 추가한 순서대로 위에 겹쳐서 계속 출력하는 레이아웃
  - 대부분 뷰의 표시 여부를 설정하는 visibility 속성을 함께 사용

• FrameLayout에 버튼과 이미지 추가

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="BUTTON1" />
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
</FrameLayout>
```

▶ 실행 결과



3. 겹쳐서 배치 - FrameLayout

- **visibility 속성 값을 바꾸는 액티비티 코드**

• visibility 속성값을 바꾸는 액티비티 코드

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.button.setOnClickListener {
            binding.button.visibility = View.INVISIBLE
            binding.imageView.visibility = View.VISIBLE
        }
        binding.imageView.setOnClickListener {
            binding.button.visibility = View.VISIBLE
            binding.imageView.visibility = View.INVISIBLE
        }
    }
}
```

버튼은 숨기고 이미지는 보이도록 설정

버튼은 보이고 이미지는 숨기도록 설정

4. 표 형태로 배치 - GridLayout

- **GridLayout 배치 규칙**
  - 테이블 화면을 만드는 레이아웃
  - orientation 속성으로 가로나 세로 방향으로 뷰를 나열
  - 줄바꿈을 자동으로 해줍니다.
    - orientation: 방향 설정
    - rowCount: 세로로 나열할 뷰 개수
    - columnCount: 가로로 나열할 뷰 개수

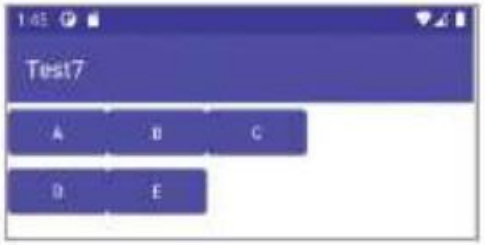
4. 표 형태로 배치 - GridLayout

- 열, 행 개수 지정 예

• 열 개수 지정

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:columnCount="3">
    <Button android:text="A" />
    <Button android:text="B" />
    <Button android:text="C" />
    <Button android:text="D" />
    <Button android:text="E" />
</GridLayout>
```

▶실행 결과



• 행 개수 지정

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:rowCount="3">
    <Button android:text="A" />
    <Button android:text="B" />
    <Button android:text="C" />
    <Button android:text="D" />
    <Button android:text="E" />
</GridLayout>
```

▶실행 결과

4. 표 형태로 배치 - GridLayout

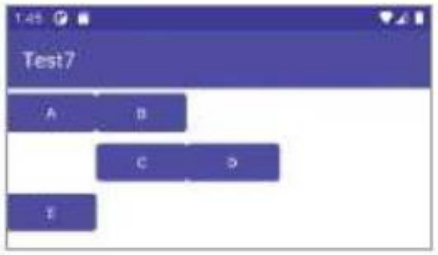
GridLayout 속성

- 특정 뷰의 위치 조정하기
  - layout_row: 뷰가 위치하는 세로 방향 인덱스
  - layout_column: 뷰가 위치하는 가로 방향 인덱스

• 뷰의 위치 조정

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:columnCount="3">
    <Button android:text="A" />
    <Button android:text="B" />
    <Button android:text="C"
        android:layout_row="1"
        android:layout_column="1" />
    <Button android:text="D" />
    <Button android:text="E" />
</GridLayout>
```

▶ 실행 결과



- 특정 뷰의 크기 확장하기
  - layout_gravity 속성을 이용

• 뷰의 크기 확장하기

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:columnCount="3">
    <Button android:text="A" />
    <Button android:text="HELLOWORLD HELLOWORLD" />
    <Button android:text="C" />
    <Button android:text="D" />
    <Button android:text="E"
        android:layout_gravity="fill_horizontal" />
    <Button android:text="F" />
</GridLayout>
```

▶ 실행 결과

4. 표 형태로 배치 - GridLayout

- 한 칸에 뷰를 여러개 표시

• 여백에 다음 뷰 넣기

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:columnCount="3">
    <Button android:text="A" />
    <Button android:text="HELLOWORLD HELLOWORLD" />
    <Button android:text="C" />
    <Button android:text="D" />
    <Button android:text="E" />
    <Button android:text="F"
        android:layout_row="1"
        android:layout_column="1"
        android:layout_gravity="right" />
</GridLayout>
```

▶ 실행 결과

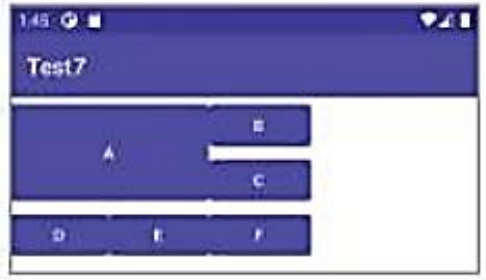
4. 표 형태로 배치 - GridLayout

- **열이나 행 병합하기**
  - layout_columnSpan: 가로로 열 병합
  - layout_rowSpan: 세로로 행 병합

• 열과 행 병합

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:columnCount="3">
    <Button android:text="A"
        android:layout_columnSpan="2"
        android:layout_rowSpan="2"
        android:layout_gravity="fill" />
    <Button android:text="B" />
    <Button android:text="C" />
    <Button android:text="D" />
    <Button android:text="E" />
    <Button android:text="F" />
</GridLayout>
```

▶ 실행 결과



5. 계층 구조로 배치 - ConstraintLayout

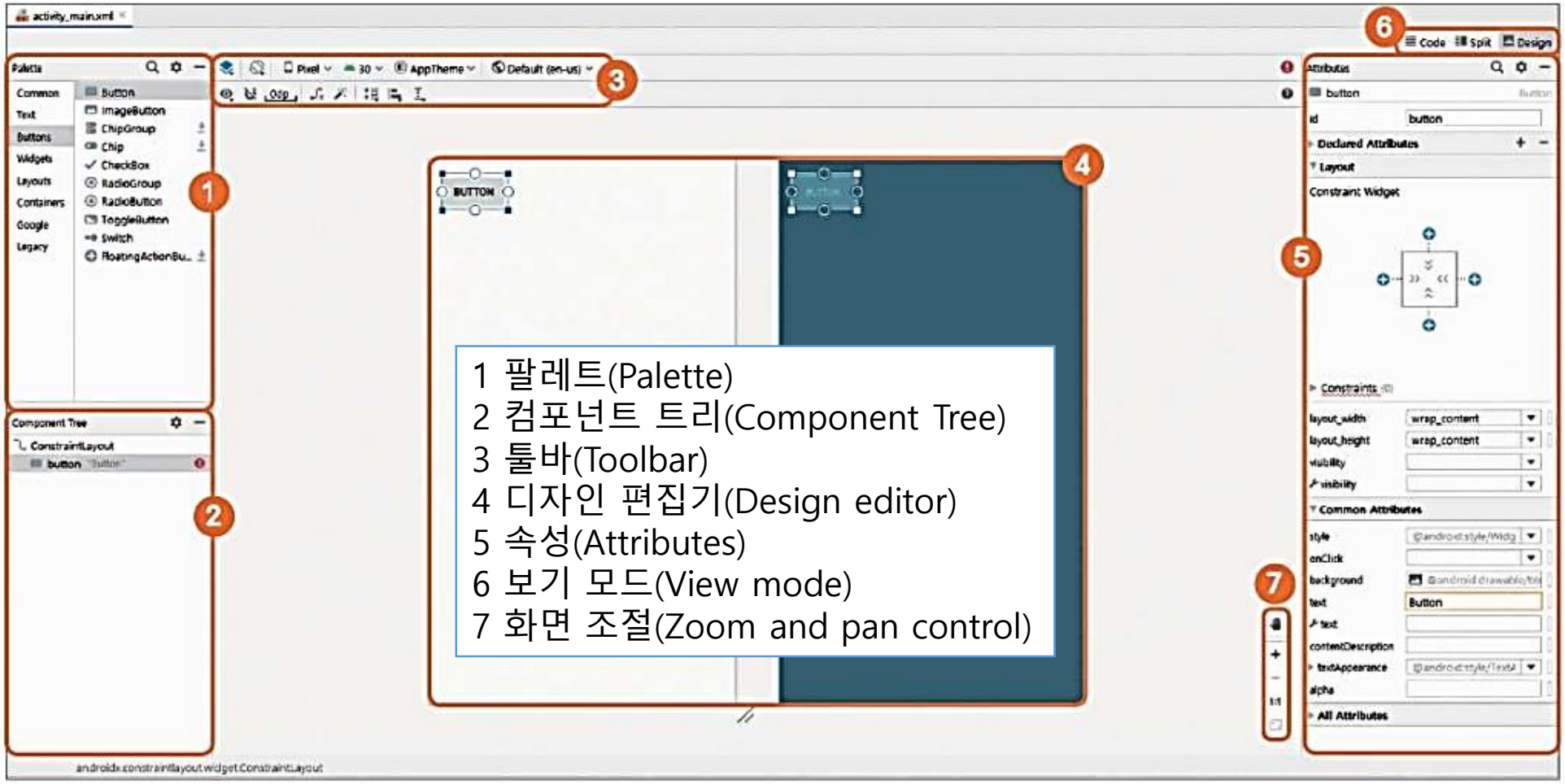
- **androidx에서 제공하는 라이브러리**

• constraintlayout 사용을 위한 빌드 설정

```
implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
```

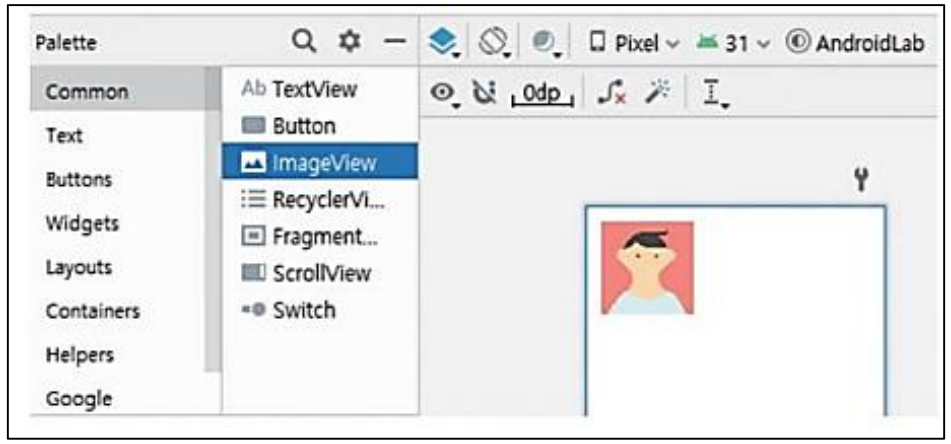
- **레이아웃 편집기에서 레이아웃 구성하기**
  - 상대 위치로 배치하는 RelativeLayout과 비슷하지만 더 많은 속성을 제공
  - 레이아웃 편집기를 제공

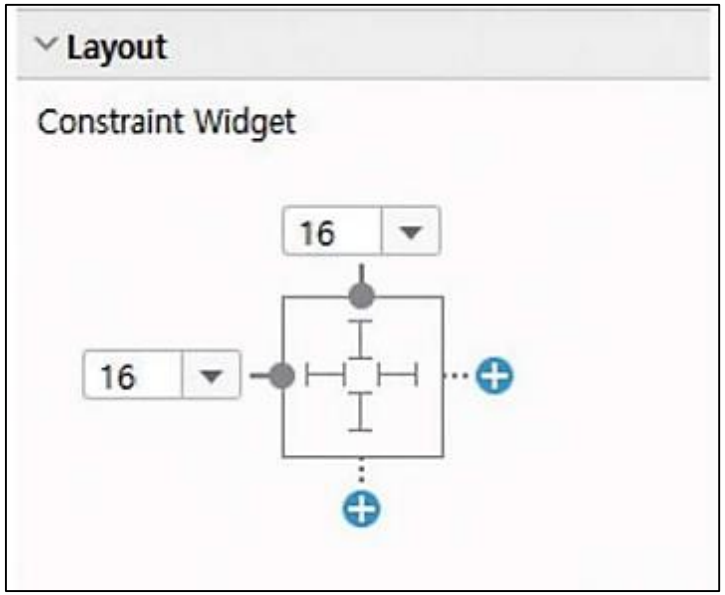
5. 계층 구조로 배치 - ConstraintLayout

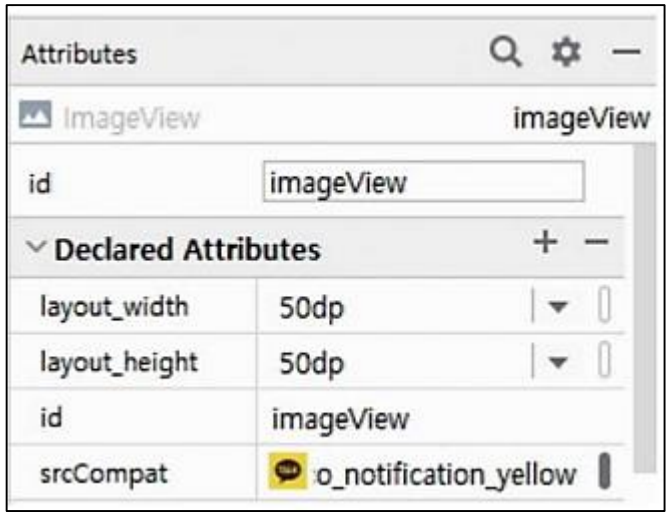


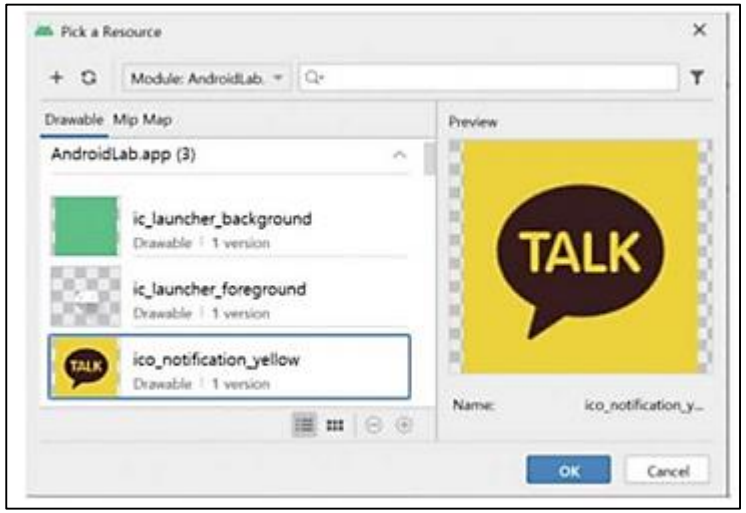
5. 계층 구조로 배치 - ConstraintLayout

- 이미지 추가



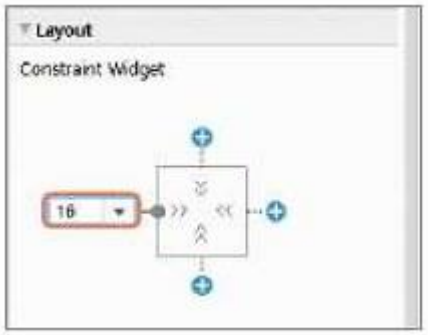


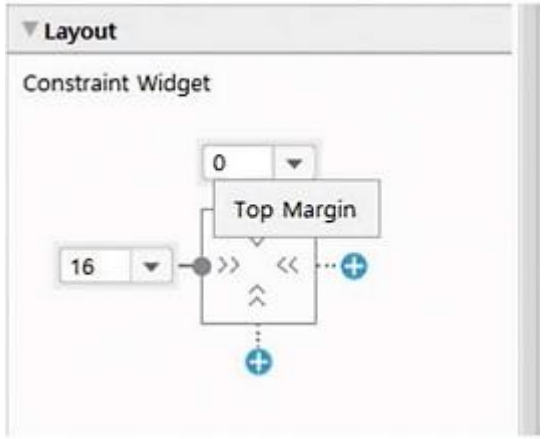




5. 계층 구조로 배치 - ConstraintLayout

- 제목 추가



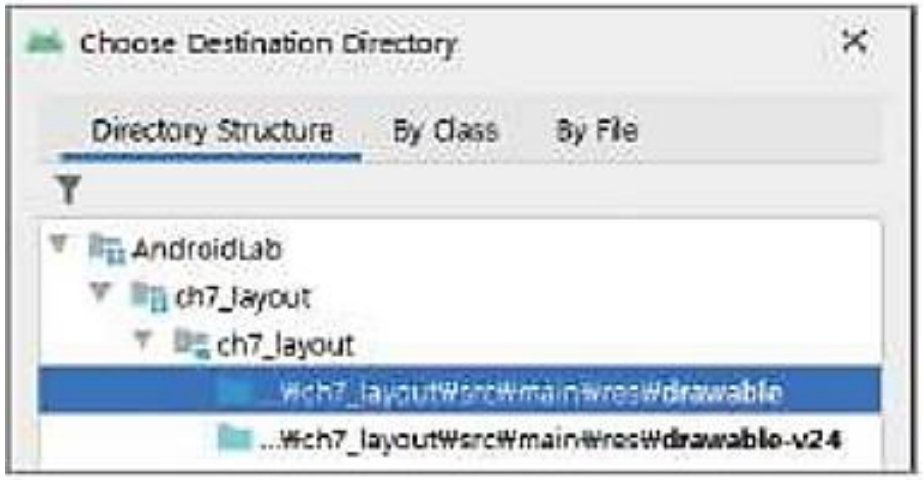


5. 계층 구조로 배치 - ConstraintLayout

- 메시지 추가

실습 : 전화 앱의 키패드 화면 만들기

- **1단계. 새로운 모듈 생성하기**
  - Ch7_Layout이라는 이름으로 새로운 모듈을 만든다.
- **2단계. 실습 파일 복사하기**
  - add.png, back.png, call.png, video.png 파일을 res/drawable 디렉터리에 복사



실습 : 전화 앱의 키패드 화면 만들기

- **3단계. 메인 액티비티 작성하기**
  - activity_main.xml 파일
- **4단계. 앱 실행하기**