Android application 만들기

3. Layout

Linear layout
Constraint layout

안드로이드에서 코드와 디자인을 연결하는 방법

기능 코드는 java파일에 구현. 각종 디자인 관련 파일은 res 폴더 안에 있다.

-> 디자인과 코드를 어떻게 연결할까?

res 폴더 안의 파일들은 자동으로 아이디가 부여 되며 자바 코드에서는 다음과 같이 아이디를 이용해서 사용 가능

R.카테고리.아이디

카테고리: 주로 res 안의 폴더 이름을 사용하며 string, color 등의 카테고리도 있다.

아이디: 주로 확장자를 제외한 파일이름. 일부 xml 파일 안의 name 속성을 아이디로 사용하기도 한다. 자동으로 아이디를 인식하므로 파일 이름은 영어 소문자와 _ 와 숫자만으로 작성한다.

예를 들어 res/layout 폴더 안의 activity_main.xml의 아이디는

R.layout.activity_main

자바 코드에서 레이아웃 부르기

화면을 디자인 하는 파일. xml로 작성하며 다양한 양식을 제공하며 res/layout 폴더 안에 작성.

Activity는 반드시 하나의 레이아웃이 필요하며 코드에서는 다음 부분에서 사용할 레이아웃을 지정.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

res/layout 폴더 내에 사용하지 않는 xml 파일이 있는 것은 괜찮지만 내용에 오류가 있을 경우 전체 컴파일이 되지 않으므로 주의한다.

레이아웃의 구조

내부에 위젯들을 포함하며 다른 레이아웃을 포함할 수 있다.

레이아웃 파일은 트리구조로 기술하며 최상위 항목은 반드시 레이아웃을 사용한다.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

레이아웃에 위젯 하나가 있는 가장 단순한 형태

레이아웃과 위젯의 기본 속성

Left/Start

위젯이나 레이아웃에서 정렬이나 위치를 지정할 때 사용하는 표현

초기 버전 안드로이드 라이브러리에서는 Left/Right를 사용했지만 반전 언어 지원을 위해 Start/End로 수정되었다.

Minimum SDK가 낮은 경우 Left와 Start를 모두 표기해 줘야 정상 동작하고 Minimum SDK가 높은 경우에는 Start/End만 사용해도 된다.

android:layout_width 와 android:layout_height

사이즈를 지정하는 속성. 필수 속성이다.

- match_parent: 부모의 사이즈에 맞춰 달라는 것으로 margin, padding을 적용한 나머지 공간을 가득 채운다.
- warp_content: 위젯의 내용물 사이즈에 맞게 지정하는 것.
- 직접 수치 입력: 사이즈를 직접 지정하는 것으로 dp 단위를 사용한다.

레이아웃과 위젯의 기본 속성

dp

다양한 화면 해상도를 가지는 안드로이드 디바이스를 지원하기 위한 단위.

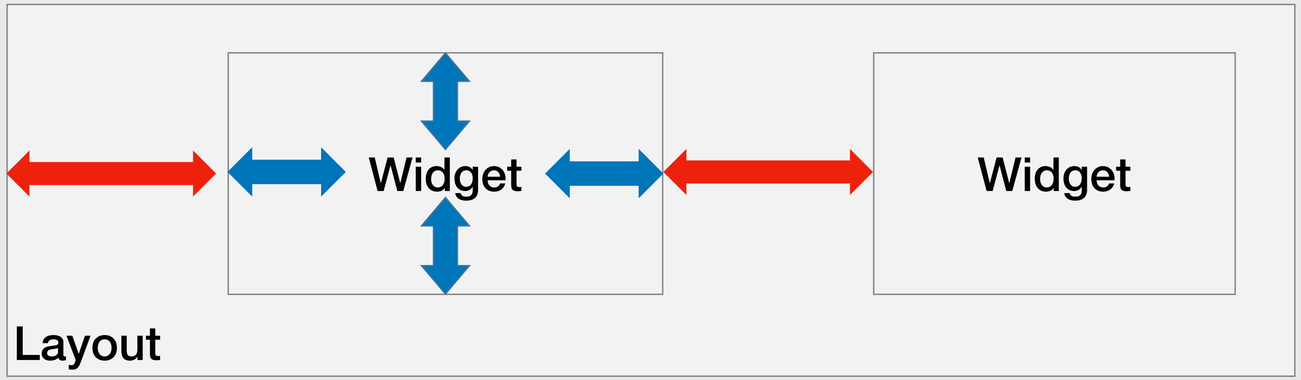
pixel로 사이즈를 지정할 경우 고해상도 디바이스로 갈수록 작게 출력되지만

dp로 사이즈를 지정할 경우 어느 정도 사이즈가 유지된다.

android:layout_margin 과 android:padding

margin은 부모 레이아웃이나 다른 위젯과의 간격, padding은 자기 자식 요소들과 경계와의 간격. dp로 지정한다.

layout_marginTop/Left/Bottom/End와 같이 네 방향에 각각 다른 값을 지정할 수 있다.



레이아웃과 위젯의 기본 속성

android:gravity

자식 요소의 정렬 방법. 좌우에 대한 설정과 상하에 대한 설정을 지정할 수 있으며 | 연산자를 사용한다.

center, centerHorizontal, centerVertical 등을 이용해 가운데 정렬.

bottom, top, start(left), end(right) 등을 사용할 수 있다.

android:id

레이아웃에서 다른 위젯을 배치하는 기준점으로 사용할 때, 자바 코드에서 이 위젯이나 레이아웃을 제어할 때 부여한다.

일반적인 자바의 변수 이름과 같은 규칙을 사용한다.

레이아웃 내에서 유일하면 되지만 특별한 문제가 없다면 프로젝트 내에서 유일하게 작성하는 것을 추천한다.

새 아이디를 부여할 때는 **@+id/만들고 싶은 아이디** 의 형식으로 적고

이미 만들어진 아이디를 참조할 때는 **@id/아이디** 로 가능하다. 그러나 참조하는 위젯의 코드 라인이 아래쪽에 있을 때는

반드시 **@+id/아이디** 로 작성해야 한다.

레이아웃과 위젯의 기본 속성

태그 이름

위젯이나 레이아웃은 내부적으로 자바 클래스와 대응된다.

클래스 이름을 태그로 지정하게 되는데 android SDK에 기본 포함 된 레이아웃이나 위젯은 클래스 이름만 바로 적고 추가된 라이브러리에 든 위젯이나 레이아웃은 패키지 이름을 포함한 전체 경로를 적는다.

이름만 적는 경우

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 ……>

</LinearLayout>
```

패키지와 클래스 이름을 다 적는 경우

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
 ……>

</androidx.constraintlayout.widget.ConstraintLayout>
```

많이 쓰는 레이아웃

Linear layout

가로 또는 세로 한 방향으로만 위젯이나 자식 레이아웃을 추가할 수 있는 레이아웃.

사용이 간편해 팝업 등에 사용하기에 좋다.

Relative layout

다른 위젯이나 레이아웃을 참조하여 위치를 정하는 레이아웃. 기존 프로젝트를 업데이트 하는 경우라면 만날 수 있다.

Constraint layout

각 위젯의 제어점을 이용하여 위치를 정하는 레이아웃. 최신 버전의 안드로이드 스튜디오를 사용해 프로젝트를 만들 경우 기본 레이아웃이다.

Linear layout

android:orientation

가로 또는 세로로 위젯을 배치하는 레이아웃이기 때문에 리니어 레이아웃의 필수 속성이다.

horizontal, vertical 중 하나를 사용한다.

horizontal

vertical

Linear layout: 실습

파일 추가

res/layout 폴더에서 마우스 우클릭 > New > Layout resource file

File name: **activity_main_linear**

Root element: **LinearLayout(입력되어 있음. 확인)**

Source set: main

Directory name: layout

Linear layout: 실습

activity_main_linear.xml (android:orientation="vertical" 확인)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</LinearLayout>
```

Activity에서 새 레이아웃 사용하도록 연결하기

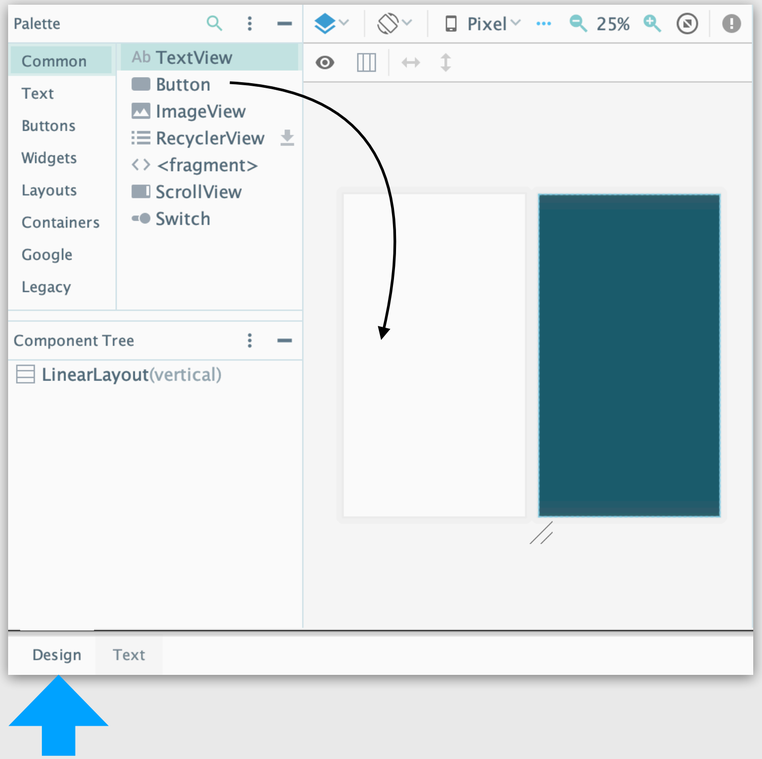
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main_linear);
}
```

AVD 또는 실제 디바이스에서 새 코드를 실행해 빈 화면이 출력되는지 확인한다.

Linear layout: 실습

버튼 3개 추가하기

Design 탭에서 위젯을 드래그로 추가하여 대략적인 위치를 잡은 다음 Text 탭에서 상세 위치와 속성을 설정한다.

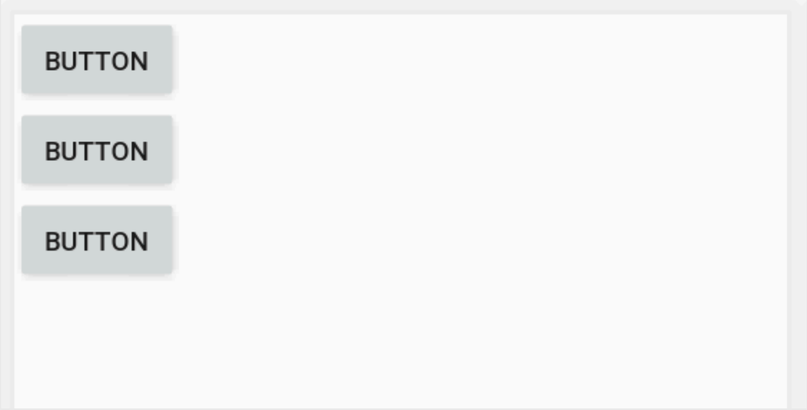



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
</LinearLayout>
```

Linear layout: 실습

버튼 3개 속성 변경하기

각 버튼의 android:layout_width 속성을 wrap_content로 변경하여 다음과 같은 화면이 나오도록 수정한다.

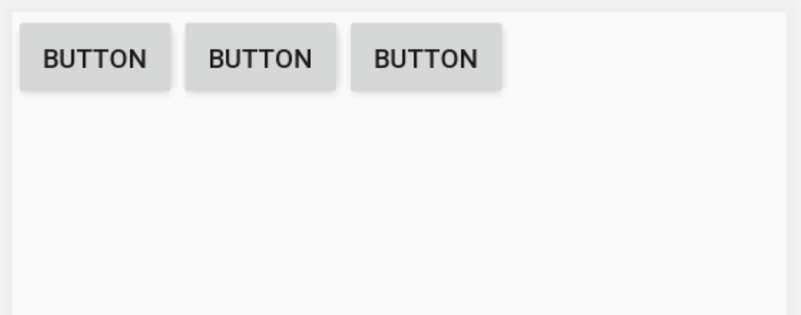



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
</LinearLayout>
```

Linear layout: 실습

Layout 속성 변경하기

LinearLayout의 android:orientation 속성을 horizontal로 변경하여 다음과 같은 화면을 확인한다.




```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
</LinearLayout>
```

Linear layout: 실습

Layout 속성 변경하기

LinearLayout에 android:gravity 속성을 center_horizontal, center 등으로 추가 해 본다.

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

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

Constraint Layout

Linear layout은 사용하기 편하지만 복잡한 화면을 디자인하면 Depth가 늘어나는 단점이 있다.

다른 Android SDK의 기본 layout들은 margin, padding등을 이용해 위젯을 배치하므로 화면 사이즈에 유연하게 대응하기 힘들었다. (예를 들어 전체 화면 높이의 30% 위치에 로고 그림 그리기)

Android에서 추가로 제공하는 레이아웃으로

Build.gradle 에 dependency를 추가해야 사용할 수 있다.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
}
```

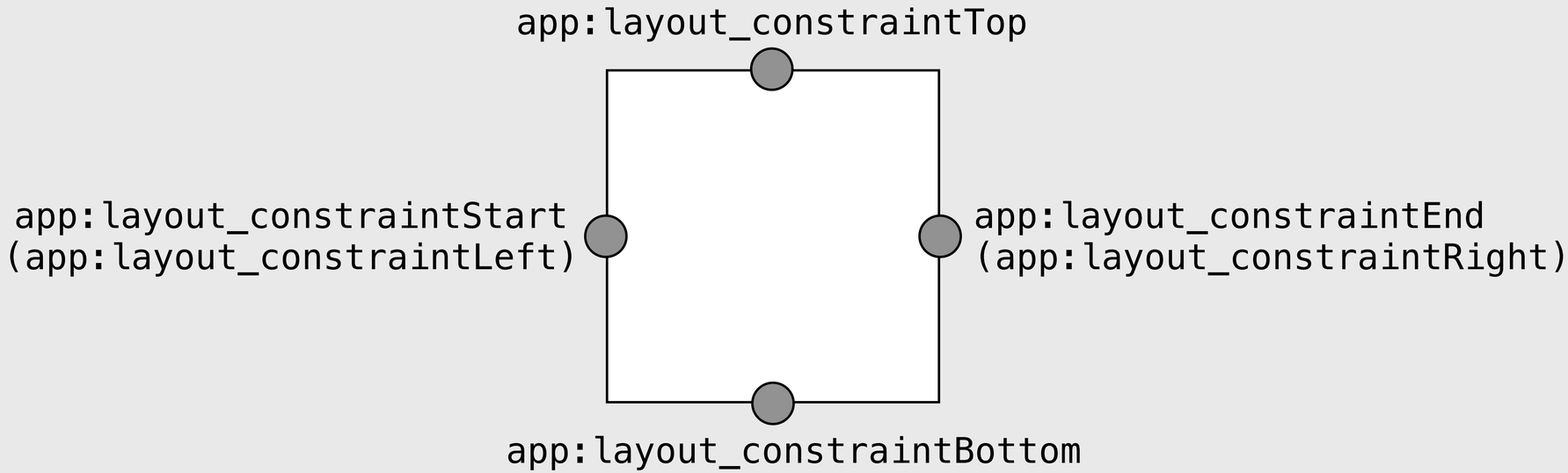
Constraint Layout

Constraint layout에서 각 레이아웃이나 위젯은 상, 하, 좌, 우(Top, Bottom, Start, End)의 제약점을 가진다.

이 제약점을 부모 레이아웃이나 다른 위젯 등에 두 방향(horizontal, vertical)을 연결해 자신의 위치를 지정한다.

Android SDK의 기본 레이아웃이 아니라 추가 라이브러리이므로

Constraint layout 전용 속성은 namespace로 android 대신 app을 사용한다.

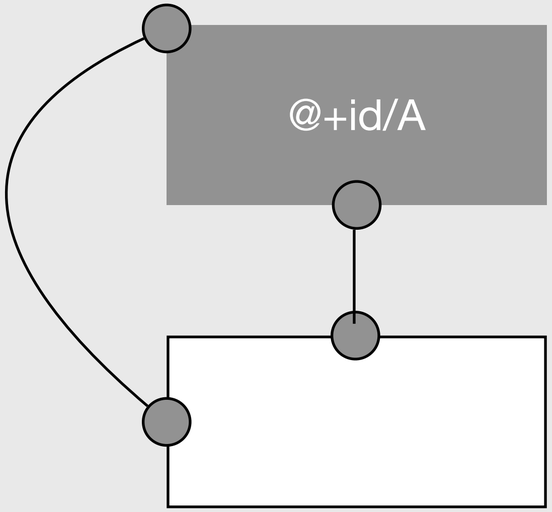


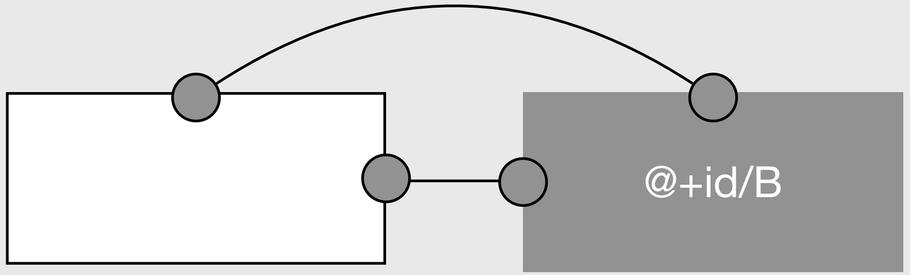
Constraint Layout

자신의 제약점을 먼저 두고 상태의 제약점을 적는 방식으로 기술한다. 아이디로 상대를 지정하며 부모는 parent라 적는다.

app:layout_constraintTop_toBottomOf=“@+id/A”

app:layout_constraintStart_toStartOf=“@+id/A”





app:layout_constraintTop_toTopOf=“@+id/B”

app:layout_constraintEnd_toStartOf=“@+id/B”

Constraint Layout

Constraint layout의 각 선들은 장력을 가진 스프링 같은 속성을 가진다.

app:layout_constraintStart_toStartOf=“parent”

app:layout_constratintTop_toTopOf=“parent”

와 같이 적을 경우 스프링의 힘으로 당겨져 부모 레이아웃의 좌 상단 모서리에 위젯이 위치한다.

app:layout_constraintStart_toStartOf=“parent”

app:layout_constraintEnd_toEndOf=“parent”

의 경우 가로로 가운데 정렬이 된다.

android studio의 Design 탭에서 드래그 & 드롭으로 제약점을 바로 연결할 수 있다.

Constraint Layout

activity_main.xml

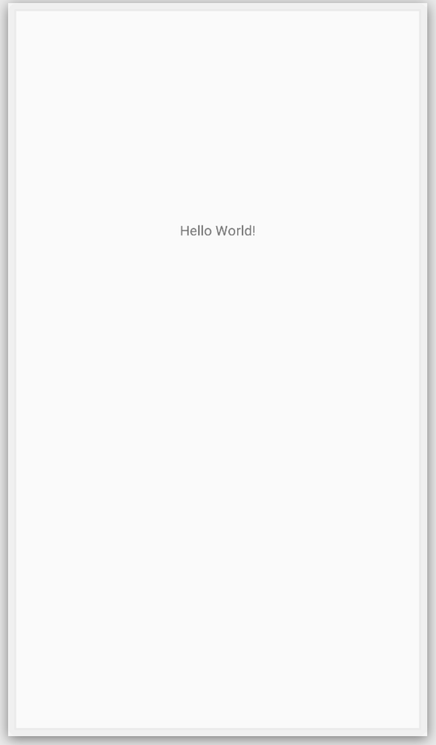
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

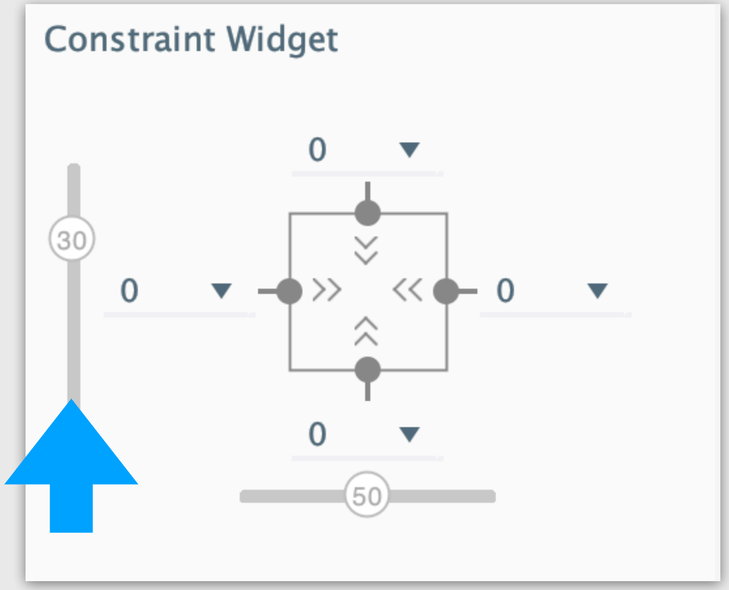
Constraint Layout

상, 하 또는 좌, 우가 동시에 연결 된 경우 힘의 크기를 조정할 수 있는데 이를 bias 라고 한다. 디바이스의 화면 사이즈가 달라져도 비율을 유지하므로 편리하다.

Design 탭에서 위젯을 클릭하면 우측에 다음 메뉴가 출력되며

bias, constraint 제거하기, 네 방향의 margin 설정하기 등을 할 수 있다.



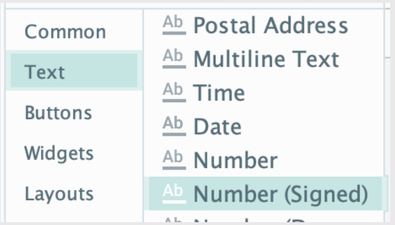


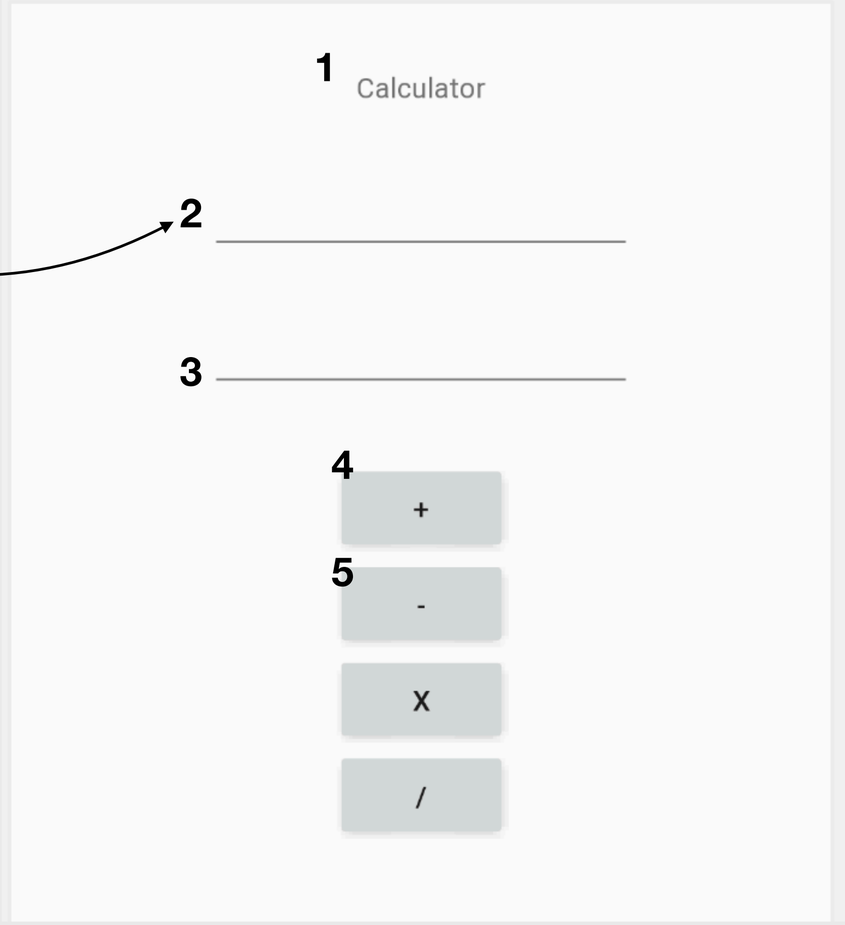

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.3" />
```

Constraint Layout: 실습

activity_main.xml을 다음과 같이 디자인하고

앱을 실행했을 때 그 화면이 나오도록 수정하라.

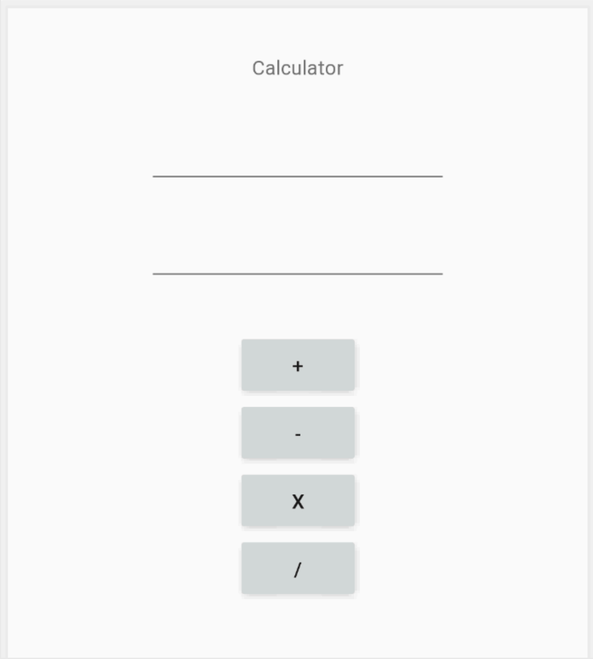




1. @+id/textView : 출력 글자를 Calculator로 수정하고 위쪽 마진 32dp로 위쪽 정렬, 좌우 가운데 정렬
2. @+id/editTextNum1: Text 카테고리에서 Number (Signed) 선택. textView 아래쪽에 마진 32dp, 좌우 가운데 정렬
3. @+id/editTextNum2: 좌우 가운데 정렬 위쪽에 마진 24dp
4. @+id/buttonAdd: Button 카테고리에서 Button 선택. editTextNum2 와 32dp 마진. 좌우 가운데 정렬
5. @+id/buttonSub, @+id/buttonMul, @+id/buttonDiv 로 설정. 마진은 없고 화면과 같이 정렬하고 좌우는 가운데 정렬

Constraint Layout: 실습

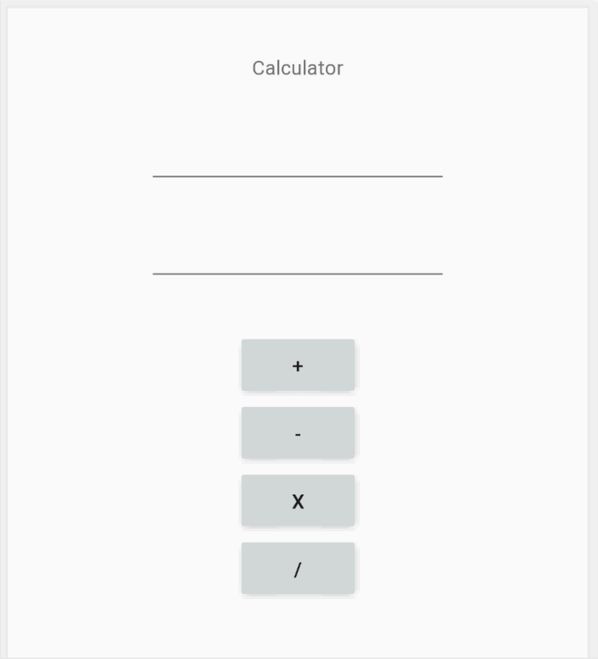
activity_main.xml을 다음과 같이 디자인하고 앱을 실행했을 때 그 화면이 나오도록 수정하라.




```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="Calculator"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Constraint Layout: 실습

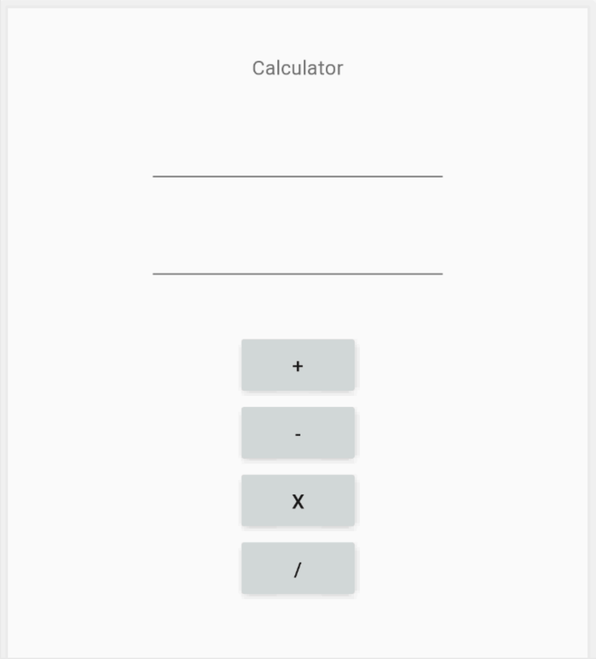
activity_main.xml을 다음과 같이 디자인하고 앱을 실행했을 때 그 화면이 나오도록 수정하라.




```
<EditText
    android:id="@+id/editTextNum1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:ems="10"
    android:inputType="numberSigned"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />

<EditText
    android:id="@+id/editTextNum2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:ems="10"
    android:inputType="numberSigned"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextNum1" />
```

Constraint Layout: 실습

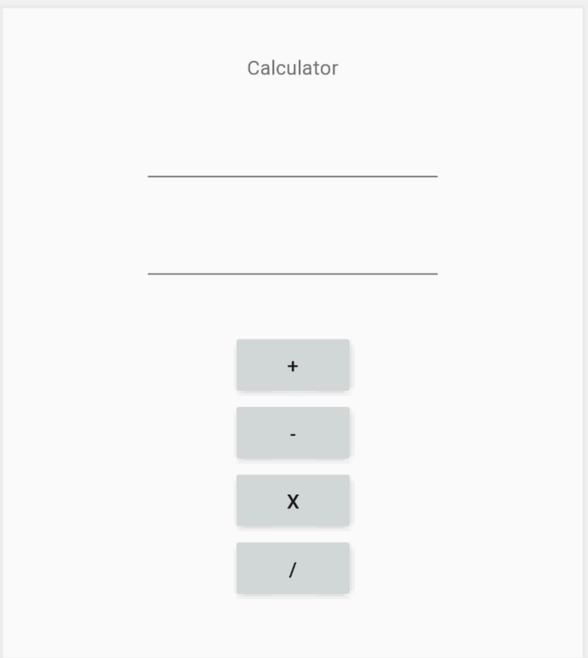
activity_main.xml을 다음과 같이 디자인하고 앱을 실행했을 때 그 화면이 나오도록 수정하라.




```
<Button
    android:id="@+id/buttonAdd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:text="+"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextNum2" />

<Button
    android:id="@+id/buttonSub"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="-"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/buttonAdd" />
```

Constraint Layout: 실습

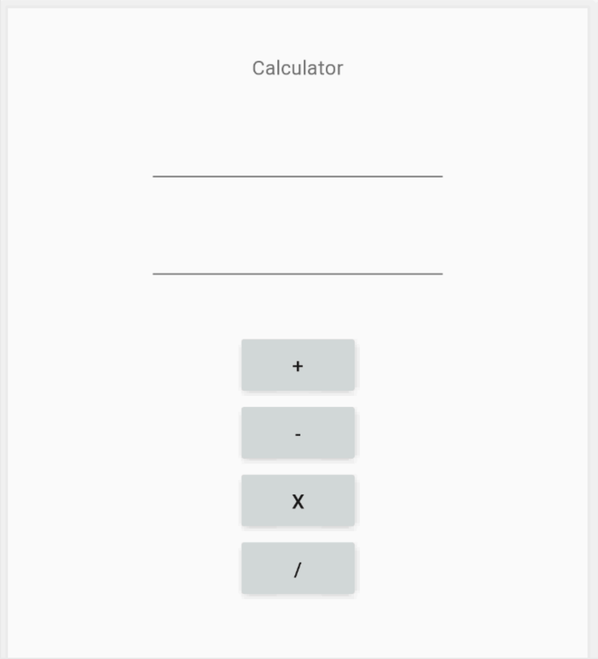
activity_main.xml을 다음과 같이 디자인하고 앱을 실행했을 때 그 화면이 나오도록 수정하라.




```
    <Button
        android:id="@+id/buttonMul"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="x"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonSub" />

    <Button
        android:id="@+id/buttonDiv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="/"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonMul" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Constraint Layout: 실습

activity_main.xml을 다음과 같이 디자인하고 앱을 실행했을 때 그 화면이 나오도록 수정하라.




```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```