
Kotlin을 이용한 Android 프로그래밍

로또 번호 생성 앱 만들기

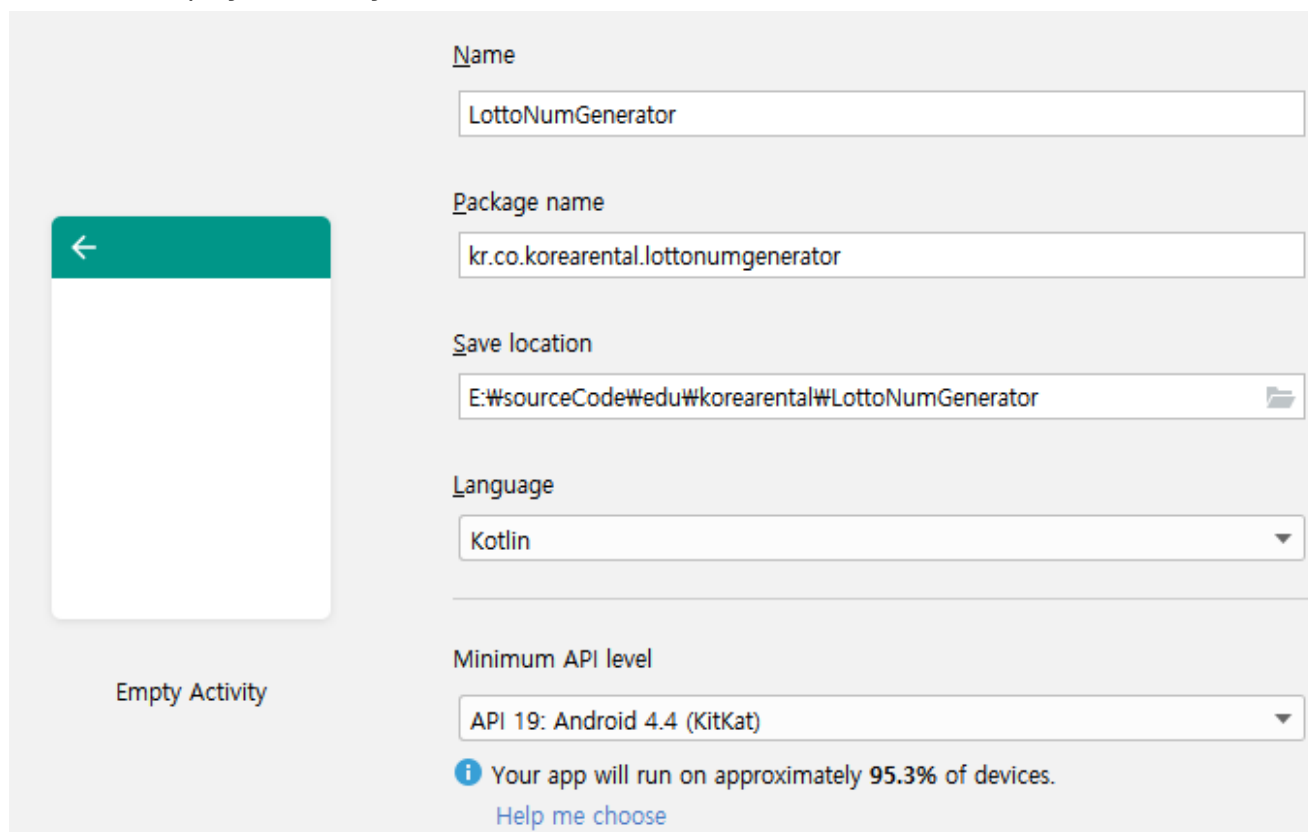
Contents

- I. 1~45까지의 숫자 중 6개를 사용자에게 보여줌
- II. 로또 번호 생성 방법
 - I. 랜덤으로 로또 번호를 생성하여 추천
 - II. 사용자의 생일을 입력 받아 별자리에 맞는 로또 번호를 추천
 - III. 사용자의 이름을 입력 받아서 이름에 맞는 로또 번호를 추천

로또 번호 생성 앱 만들기

▶ 프로젝트 생성

- ▶ 프로젝트 명 : LottoNumGenerator
- ▶ minSdkVersion : 19(Android 4.4 KitKat)
- ▶ 기본 액티비티 : EmptyActivity



Name

LottoNumGenerator

Package name

kr.co.korearental.lottonumgenerator

Save location

E:\sourceCode\Wedu\korearental\LottoNumGenerator

Language

Kotlin

Minimum API level

API 19: Android 4.4 (KitKat)

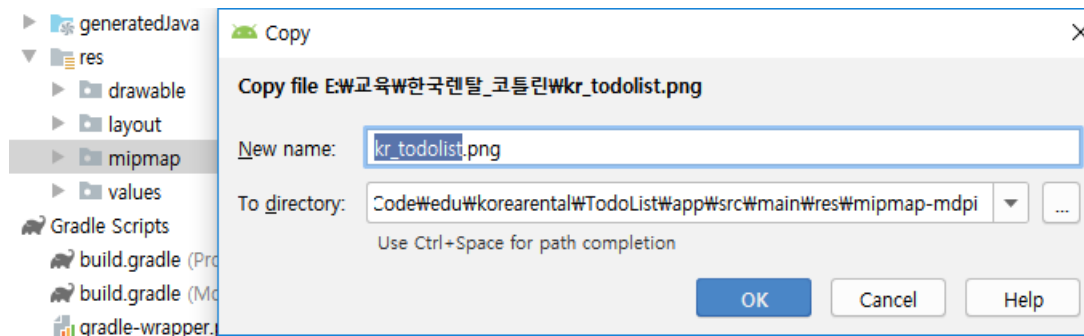
i Your app will run on approximately 95.3% of devices.
[Help me choose](#)

로또 번호 생성 앱 만들기

▶ 프로젝트 설정

▶ kr_lotto.png로 아이콘 변경

▶ res/mipmap 에 아이콘 이미지 복사



▶ 매니페스트 수정

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/kr_todolist"
    android:label="ToDoList"
    android:roundIcon="@mipmap/kr_todolist"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
```

로또 번호 생성 앱 만들기

▶ 액티비티 추가

▶ 로또 번호 생성 앱은 4개의 액티비티로 이루어져 있으므로 액티비티 추가

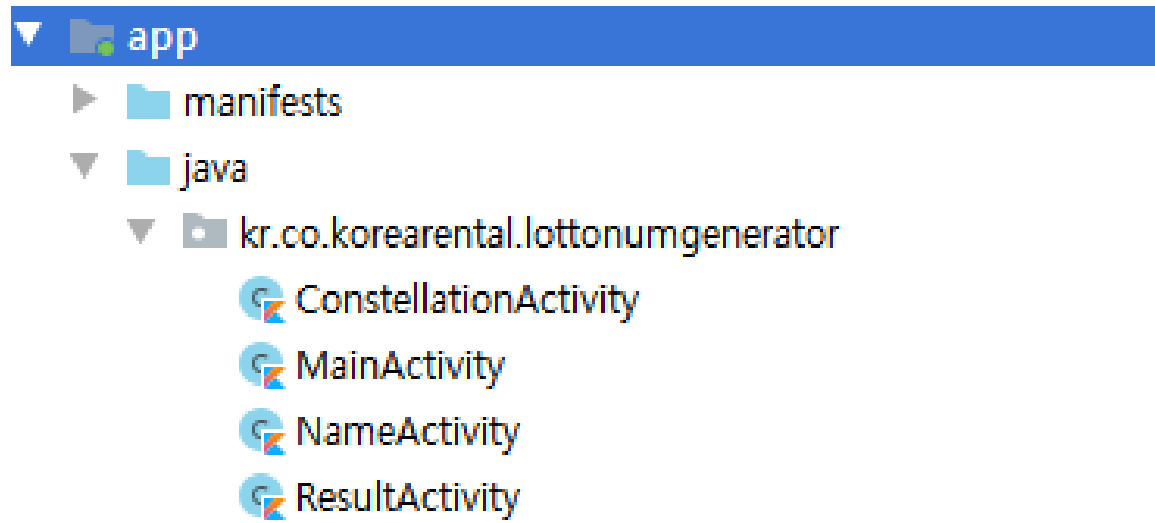
▷ MainActivity : 첫 화면 - 랜덤 번호 생성, 별자리로 번호 생성, 이름으로 번호 생성

▷ ConstellationActivity : 생년월일 입력 화면 - 별자리를 알아보기 위하여 생년월일 입력

▷ NameActivity : 이름 입력 화면 - 이름을 응용하여 로또 번호 생성

▷ ResultActivity : 로또 번호 생성 결과 화면 - 3가지 방법으로 생성된 로또 번호 출력

▶ New - Activity - Empty Activity



로또 번호 생성 앱 만들기

▶ 매니페스트 확인

▶ 매니페이스에 액티비티가 정상적으로 추가되었는지 확인

▶ 종종 자동으로 추가되지 않는 경우 발생

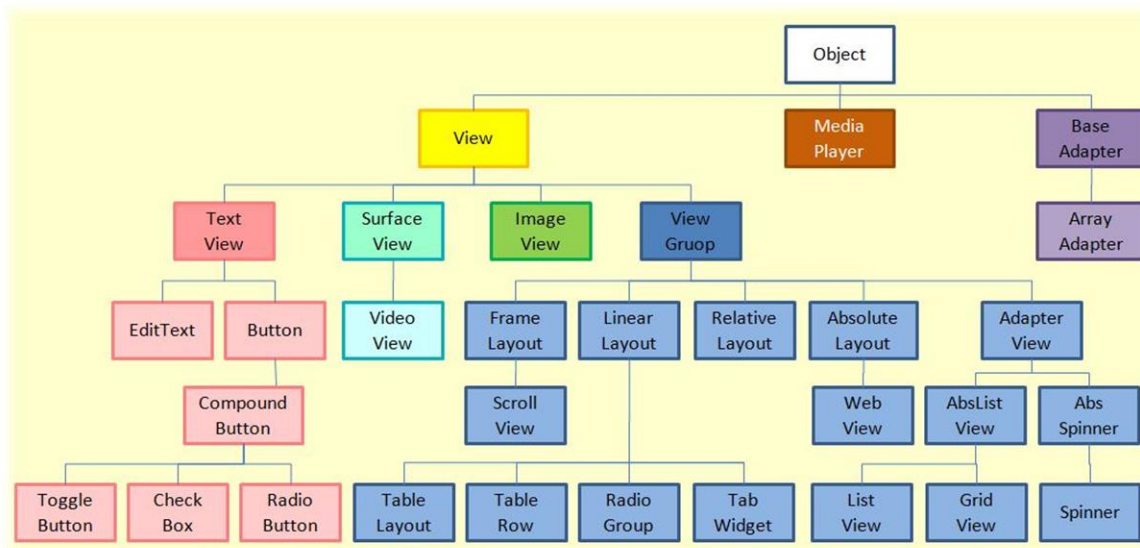
▶ activity를 xml을 사용하지 않고 kotlin으로만 만들거나 외부에서 파일을 카피하는 경우 직접 추가

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="LottoNumGenerator"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".ResultActivity">
    </activity>
    <activity android:name=".NameActivity">
    </activity>
    <activity android:name=".ConstellationActivity">
    </activity>
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN">
            </action>
            <category android:name="android.intent.category.LAUNCHER">
            </category>
        </intent-filter>
    </activity>
</application>
```

로또 번호 생성 앱 만들기

▶ 뷰(View)의 속성

- ▶ 뷰는 화면에 표시되는 가장 기본적인 요소
- ▶ xml파일에서 사용되는 CardView, TextView, ImageView 등의 UI요소들은 모양은 다르지만 모두 View라는 공통점이 있음
 - ▶ UI요소들은 모두 View를 상속 받음
- ▶ View는 기본적으로 직사각형 형태의 영역과 자기 자신을 화면에 그리기 위한 정보로 구성
 - ▶ 필요에 따라 다른 View와 구분하기 위한 자신만의 ID를 갖음



로또 번호 생성 앱 만들기

▶ 뷰 그룹과 레이아웃

- ▶ ViewGroup은 View를 상속 받으면서 여러 개의 UI요소를 포함하는 View
 - ▷ 뷰 그룹은 여러 개의 뷰들을 자식 뷰로 가지면서 각 뷰의 위치를 조정하고 관리
 - 뷰 그룹에 포함되어 있는 뷰를 자식 뷰라고 하며 뷰 그룹을 부모뷰라고 함
 - ▷ 여러 개의 뷰를 묶어서 관리하는 경우 UI를 쉽게 작성하고 유지/보수를 편리하게 해줌
- ▶ 뷰 그룹도 뷰를 상속 받기 때문에 내부에 뷰 그룹을 포함할 수 있음
 - ▷ 뷰 그룹이 뷰 그룹을 포함할 수 있는 구성은 안드로이드 UI를 매우 유연하게 작성하는데 도움
- ▶ 뷰 그룹은 뷰를 그룹 형태로 포함시킬 수 있는 개념이며 실제로 사용되는 것은 뷰 그룹을 상속 받은 레이아웃(layout)임
- ▶ 프로젝트에서는 가장 자주 사용되는 LinearLayout, RelativeLayout, ConstraintLayout을 사용

로또 번호 생성 앱 만들기

▶ LinearLayout 개요

- ▶ 내부에 포함된 자식 뷰를 선형으로 배치시키는 레이아웃
- ▶ 선형으로 배치하기 때문에 가로 또는 세로로만 배치 가능
- ▶ 프로젝트에서는 MainActivity를 선형 레이아웃으로 구성
- ▶ activity_main.xml을 선택한 후 text모드에서 기본 레이아웃인 constraintlayout을 linearlayout으로 변경

```
<?xml version="1.0" encoding="utf-8"?>  
<android.support.constraint.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

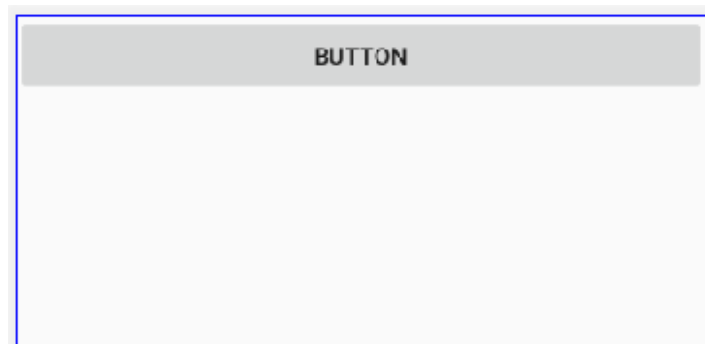


```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

- ▶ 디자인 모드에서 텍스트 뷰를 삭제하고 버튼을 추가
- ▶ 내부에서 선형으로 자식 뷰를 구성하기 때문에 버튼을 화면 어디에 추가하더라도 상단에 배치됨



- ▶ 리니어레이아웃은 기본적으로 방향 속성(orientation)이 가로이기 때문에 버튼을 추가하면 가로로 정렬됨



로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ orientation 속성을 vertical로 변경

▷ none : 속성값을 지정하지 않는 경우 기본값은 horizontal

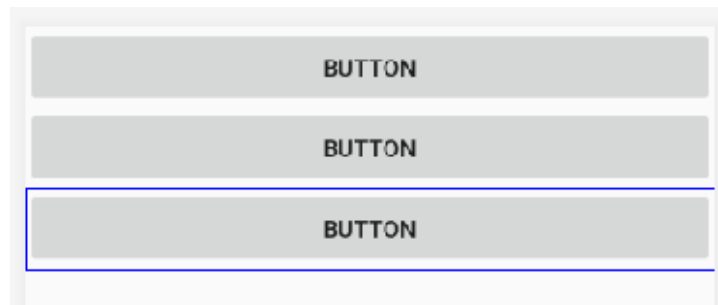
▷ horizontal : 자식 뷰를 가로로 순차 정렬

▷ vertical : 자식 뷰를 세로로 순차 정렬

▶ 안드로이드 스튜디오 버전마다 자동으로 지정되는 오리엔테이션 값이 다를 수 있음



▶ 버튼을 지우고 다시 추가



로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

- ▶ 실습에서 확인한 것과 같이 세로나 가로 방향으로 뷰를 순차적으로 배치할 때 유용
- ▶ 리니어 레이아웃은 초기부터 널리 사용된 뷰 그룹이며 복잡한 레이아웃도 속성의 설정에 따라 충분히 구현 가능
- ▶ 리니어 레이아웃에서 뷰를 가운데로 배치하기 위하여 gravity와 layout_gravity속성의 개념을 알고 있어야 함

로또 번호 생성 앱 만들기

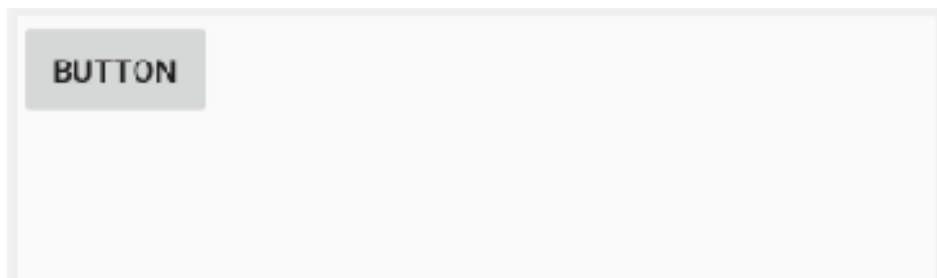
▶ LinearLayout 다루기

▶ gravity

- ▶ 객체가 자신의 경계 내에서 x축과 y축의 내용을 배치하는 방법을 지정
- ▶ 자식 뷰의 가로, 세로 정렬에 영향을 줌

▶ layout_gravity

- ▶ 구성 요소를 셀 그룹에 배치하는 방법을 지정
- ▶ 실습을 위하여 화면에서 버튼을 하나만 남기고 해당 버튼의 너비와 높이를 wrap_content로 변경

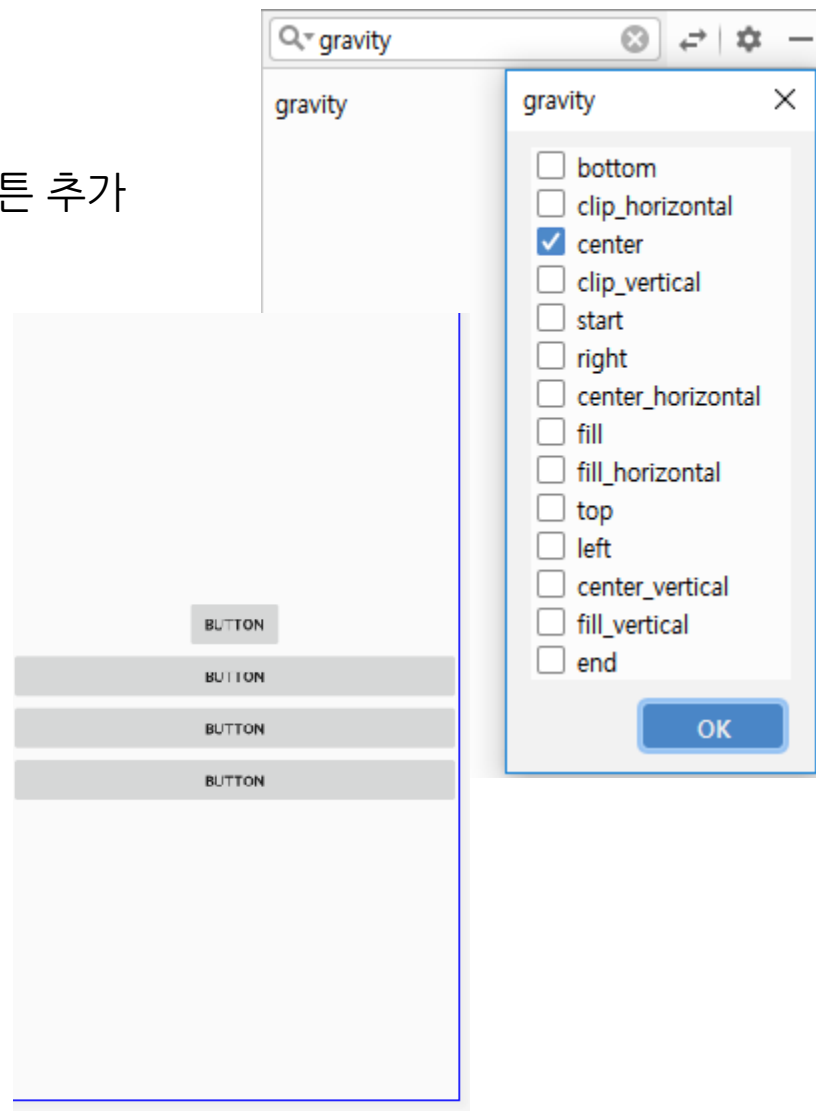


로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ gravity 설정

- ▶ 레이아웃의 gravity속성을 center로 설정한 후 버튼 추가
- ▶ 레이아웃 내부의 자식 뷰들이 가운데로 정렬됨
- ▶ gravity속성은 지정한 뷰 입장에서 콘텐츠를 정렬

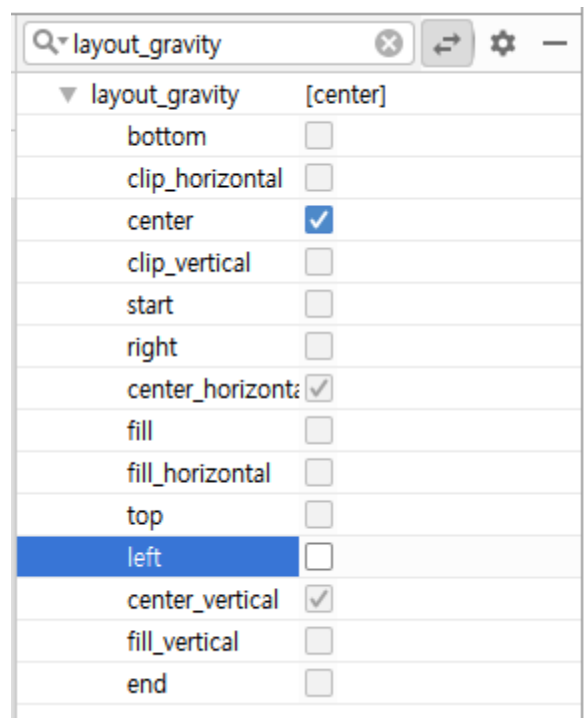
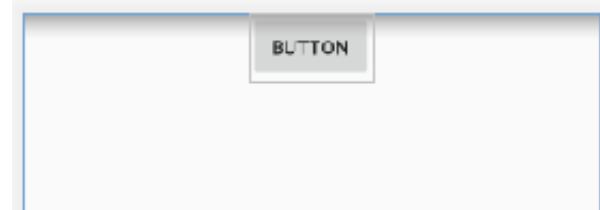


로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ layout_gravity 설정

- ▶ 새로 추가한 버튼을 모두 삭제하고 기존의 버튼만 남김
- ▶ 레이아웃의 gravity속성을 원래대로 수정
 - center를 체크 해제
- ▶ 버튼의 layout_gravity속성을 center로 설정
- ▶ 버튼이 가로 기준으로는 가운데로 정렬되지
만 세로는 가운데로 가지 않음
 - 레이아웃이 세로로 순차적 배치를 하기 때문
- ▶ 레이아웃의 오리엔테이션을 가로로 변경
- ▶ layout_gravity속성은 자신의 위치를 부모
뷰를 기준으로 정렬하는 속성



로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ layout_margin(외부 여백)

▶ 뷰의 상하좌우에 추가 공간을 지정

▶ 레이아웃을 vertical로 설정하고 버튼의 layout_gravity의 속성을 해제



▶ 버튼의 왼쪽 layout_margin속성을 100dp로 설정

- 버튼의 왼쪽 여백이 설정한 크기만큼 공간이 생김



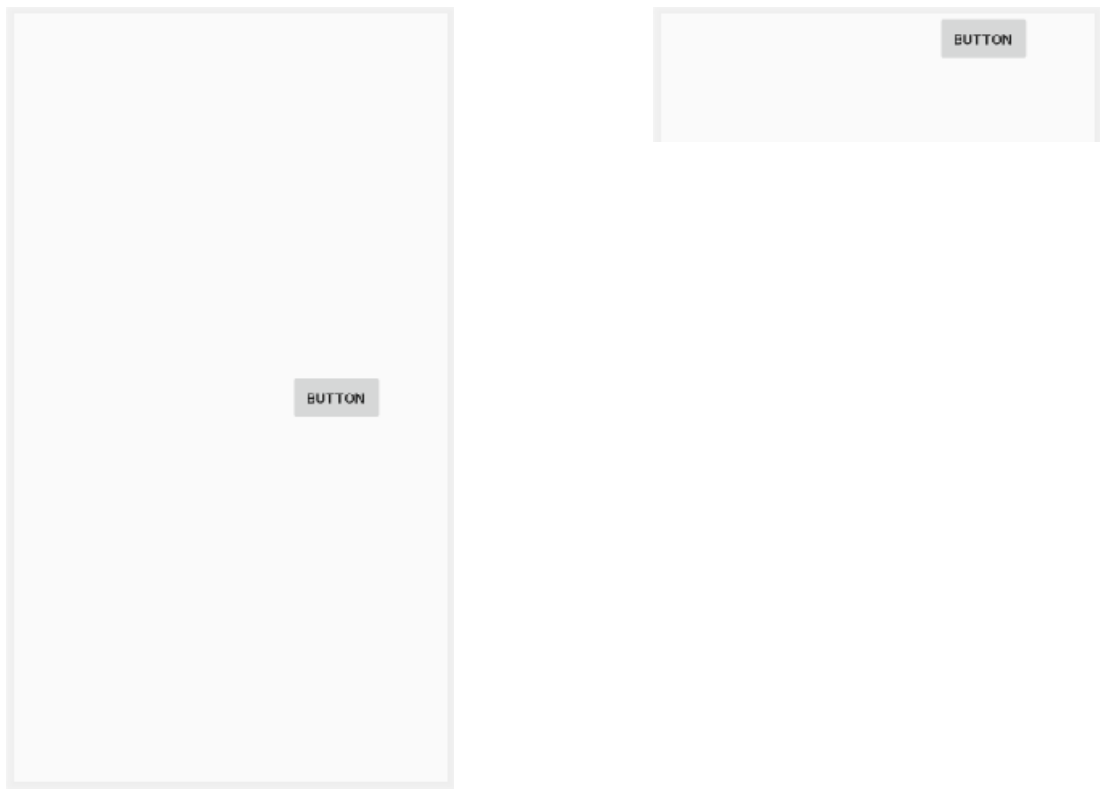
▼ Layout_Margin	[?, 100dp, ?, ?, ?]
all	
bottom	
end	
left	100dp
right	
start	
top	

로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ 앞서 배운 정렬과 여백 설정의 적용 순서 확인

▶ 레이아웃의 gravity속성과 버튼의 layout_gravity속성을 center로 적용 후 상태 확인



▶ 배치기준이 먼저 적용되고 이후에 margin값이 적용됨

로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ padding(내부 여백)

- ▶ 뷰의 가장자리와 내부의 내용사이에 간격을 지정
- ▶ 화면 상태를 초기화한 후 버튼의 왼쪽 padding을 60dp로 설정



- ▶ 결론적으로 margin은 외부 여백만큼 view를 옮기는 것이고 padding은 내부 콘텐츠에 여백을 설정에서 뷰와의 간격을 조절

로또 번호 생성 앱 만들기

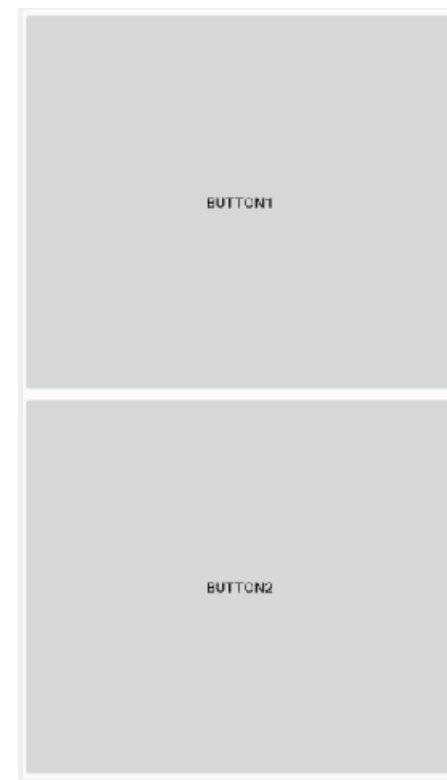
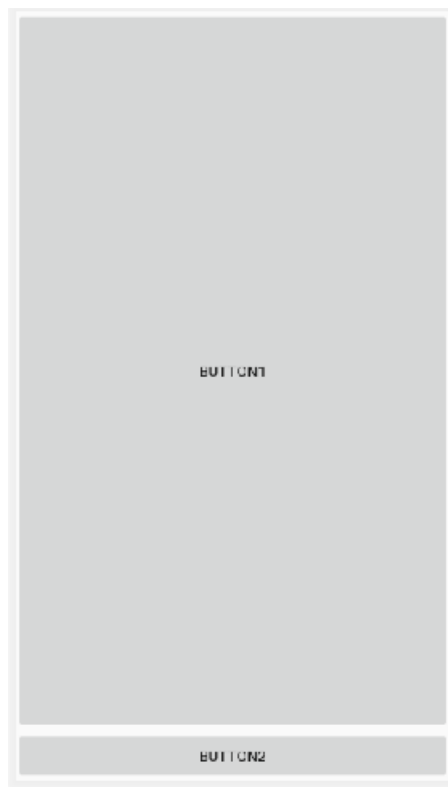
▶ LinearLayout 다루기

▶ weight 속성

▷ weight 속성의 기본 값은 0이며 button1의 layout_weight 속성을 1로 변경

▷ button2의 속성도 1로 변경

- 두개의 버튼이 1:1로 화면 구성

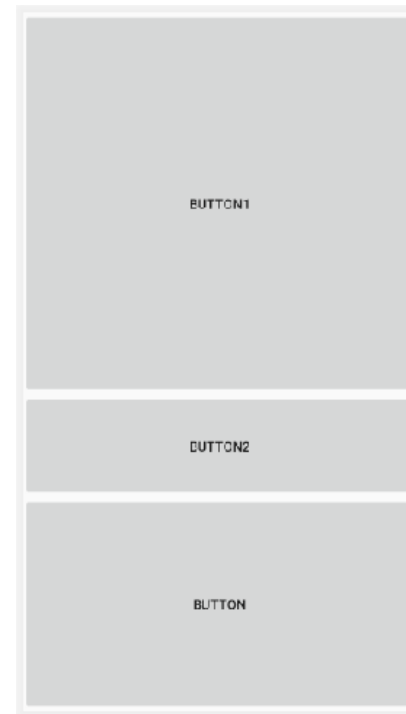
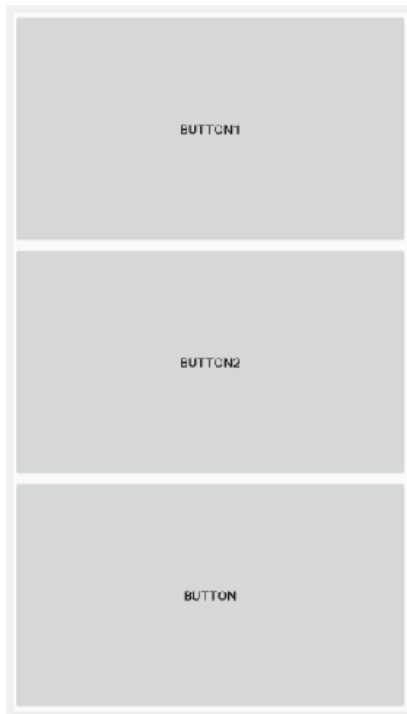


로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ weight 속성

- ▶ button3을 추가한 후 weigh속성을 1로 설정하여 1:1:1 구성 확인
- ▶ button1을 60, button2을 10, button1을 30으로 수정
 - 각각 화면을 60%, 10%, 30%씩 차지

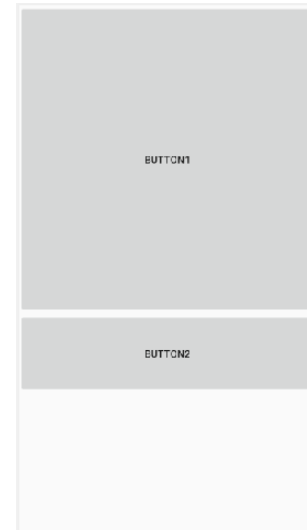
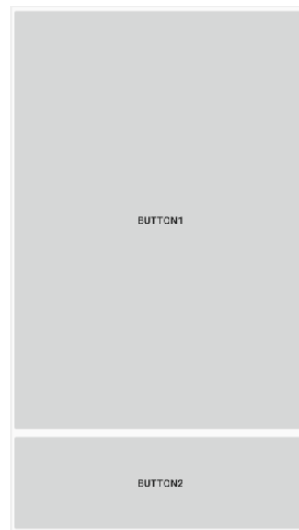
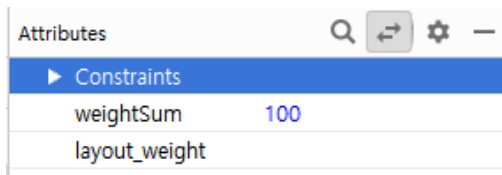


로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ weight 속성

- ▶ 앞의 상황에서 button3만 삭제하면 button1,2의 구성 비율은 6:1로 변경
- ▶ button3이 삭제되더라도 각각 60%, 10%를 유지하기 위하여 레이아웃의 weightSum속성을 100으로 설정
 - 비율 정해지는 뷰의 사이즈를 match_parent로 설정하지 않도록 주의
 - 안드로이드 개발자 레퍼런스에서는 0dp로 설정하도록 권장
 - 30%의 남은 공간 유지



로또 번호 생성 앱 만들기

▶ LinearLayout 다루기

▶ 레이아웃 중첩 사용

▶ 레이아웃 내부에 또다른 다른 레이아웃을 넣는 것을 의미

- 뷰 그룹은 뷰를 상속 받기 때문에 레이아웃 내부에 레이아웃을 포함 가능

▶ 리니어 레이아웃의 경우 가로나 세로 등 한 방향으로만 뷰를 정렬하기 때문에 유연한 구성을 위하여 중첩하여 사용

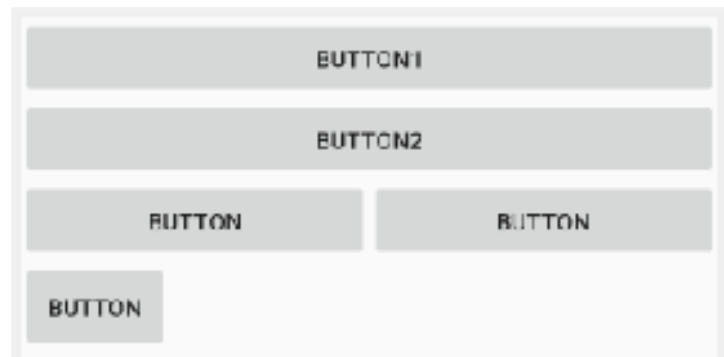
▶ 화면의 버튼 weight 속성을 각각 0으로 설정하고 높이를 wrap_content로 변경

▶ 두번째 버튼 아래에 리니어 레이아웃 추가

- 현재 레이아웃이 vertical상태이므로 horizontal로 추가
- layout_height를 wrap_content로 변경

▶ 중첩된 레이아웃 내부에 버튼 2개 추가

▶ 중첩된 레이아웃 아래에 버튼 추가



실습 문제1

▶ 리니어 레이아웃을 아래와 같이 중첩하여 구성하고 색상 값을 입력하여 구분하도록 하시오.

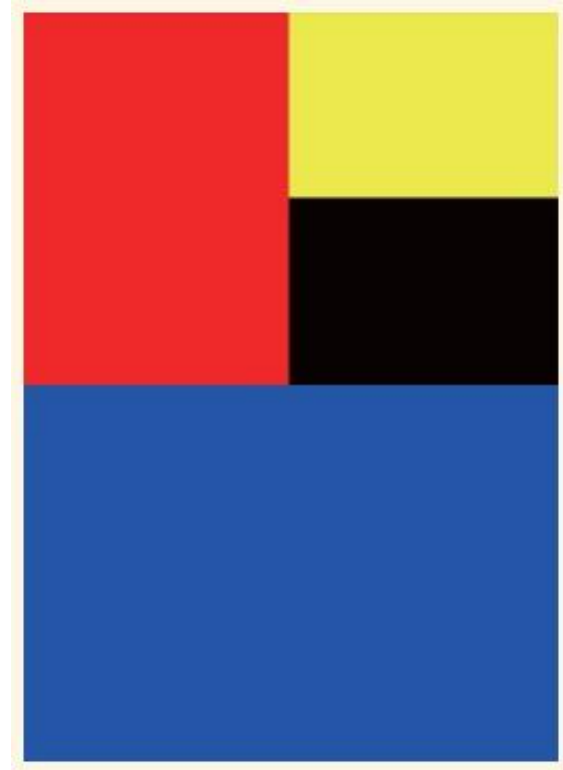
▶ 노란색 #ffff00

▶ 검정색 #000000

background



#ffff00



실습 문제2

▶ 리니어 레이아웃으로 다음 화면을 구성하는 .xml을 작성하시오.

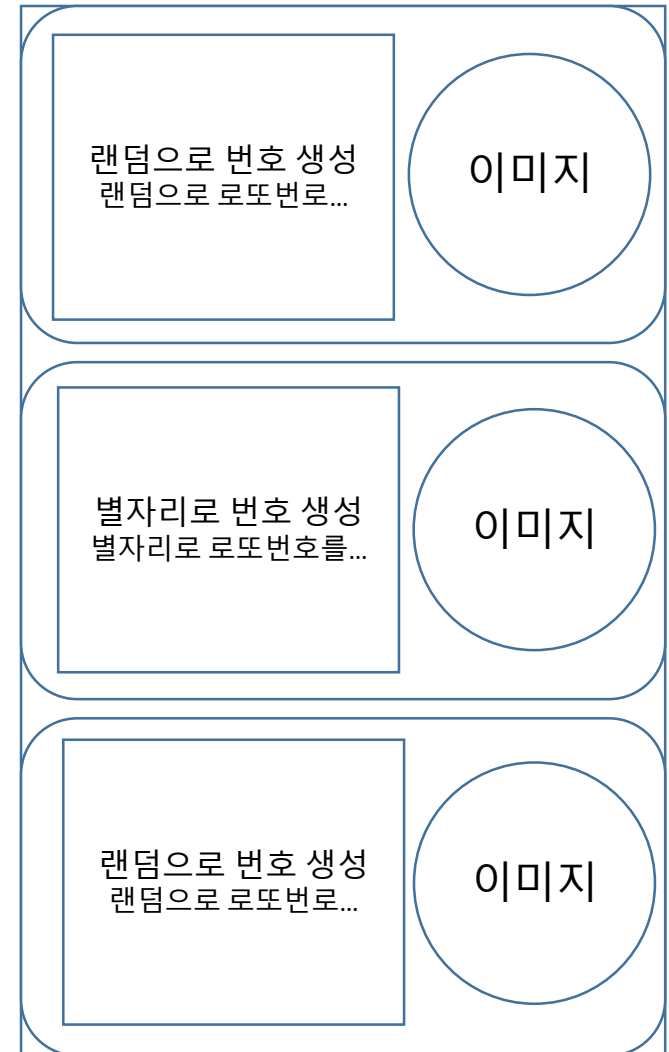
- ▶ 레이아웃의 orientation은 vertical로 설정
- ▶ 버튼 3개를 생성하고 버튼 크기는 아래와 같이 설정
 - ▷ layout_width = 110dp
 - ▷ layout_height = 110dp
- ▶ 버튼의 gravity와 layout_gravity를 모두 설정



로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

- ▶ LinearLayout(vertical)으로 구성하고 내부에 CardView를 3개 추가
- ▶ CardView는 1:1:1비율로 배치
- ▶ CardView 내부에 LinearLayout을 넣고 그 내부에 이미지 뷰와 텍스트 뷰 2개를 담을 수 있는 LinearLayout 추가



로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

- ▶ activity_main.xml의 text모드에서 ConstraintLayout을 LinearLayout으로 변경 후 TextView 삭제
- ▶ orientation 속성을 vertical로 설정

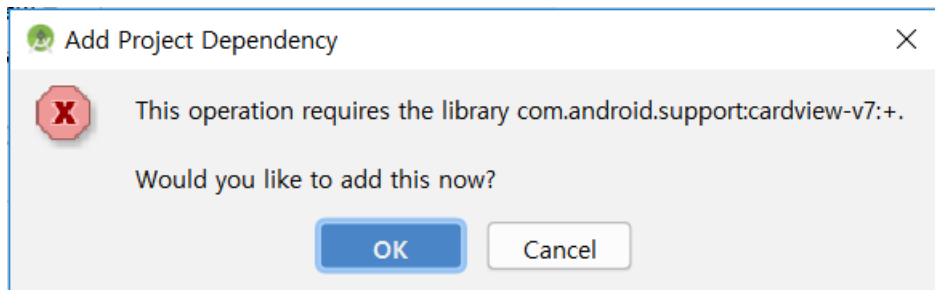
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" android:orientation="vertical"
    android:background="#00000000">
</LinearLayout>
```

로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

▶ 카드뷰 추가

- ▶ 랜덤으로 번호 생성, 별자리로 번호 생성, 이름으로 번호 생성
- ▶ 안드로이드 5.0부터 Material 디자인으로 변경되면서 CardView가 추가됨
- ▶ 카드뷰는 테두리의 라운드 처리가 가능하고 그림자, 터치 애니메이션 등이 적용된 반응형 UI
- ▶ 카드뷰를 추가할 경우, 하위 호환성을 위하여 라이브러리 의존성 주입



▶ 모듈 수준의 gradle에서 추가 확인

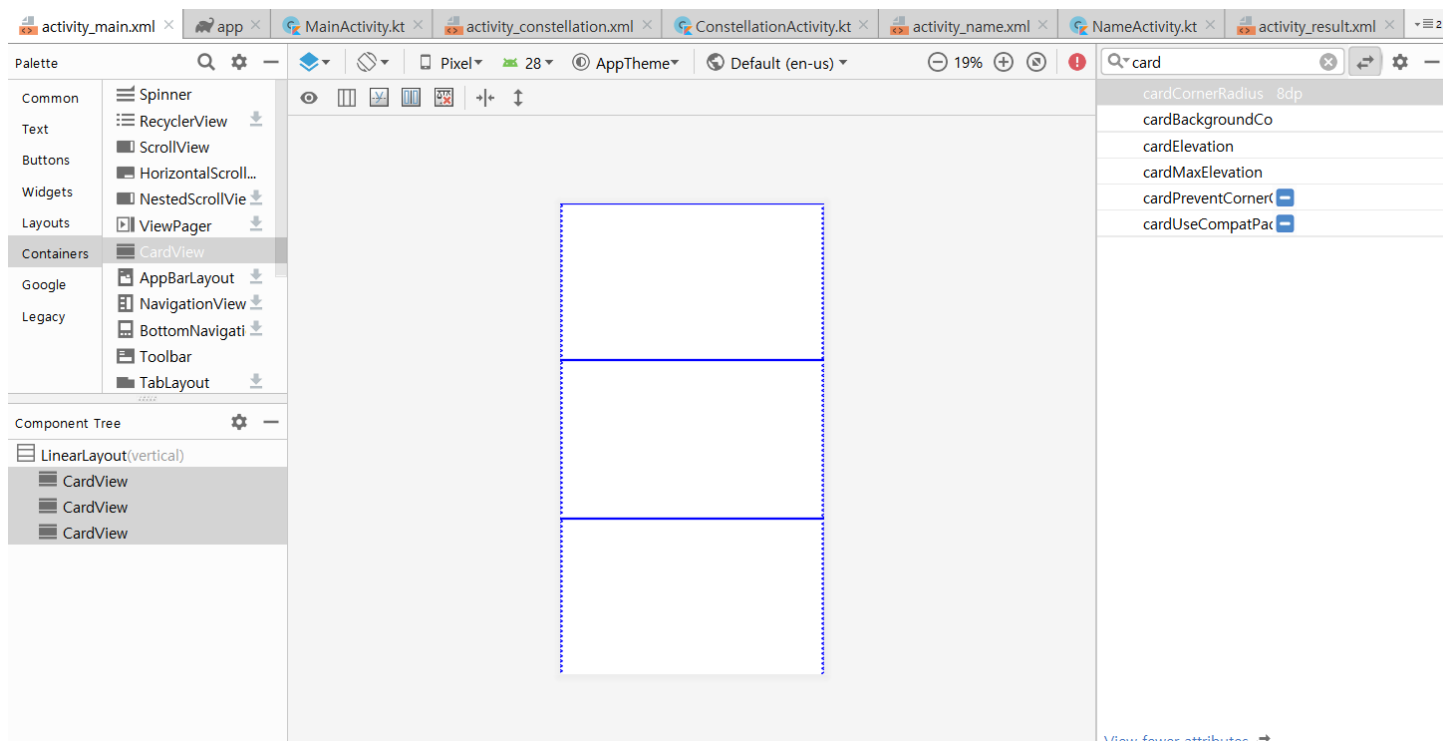
```
androidTestImplementation 'com.android.support.test:runner:1.0.2'  
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
implementation 'com.android.support:cardview-v7:28.0.0'
```

로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

▶ 카드뷰 3개 추가

- ▶ 1:1:1비율로 설정 / 높이 0dp / 그리고 레이아웃 weightSum 설정으로 3으로 지정
- ▶ 카드 뷰 사이에 약간의 여백을 추가하기 위하여 2,3번째 카드 뷰에 margin/top = 2dp
- ▶ 모서리의 둥근 효과를 위하여 cardCornerRadius 속성을 8dp로 설정



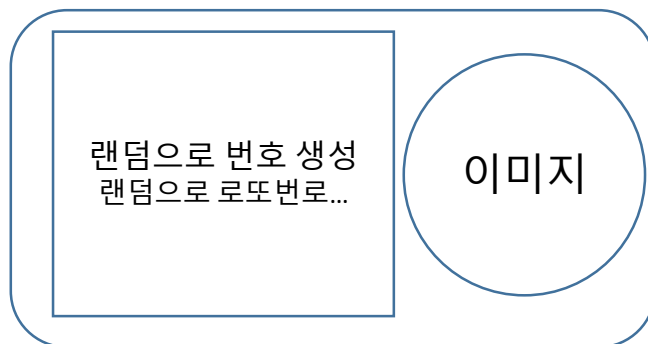
로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

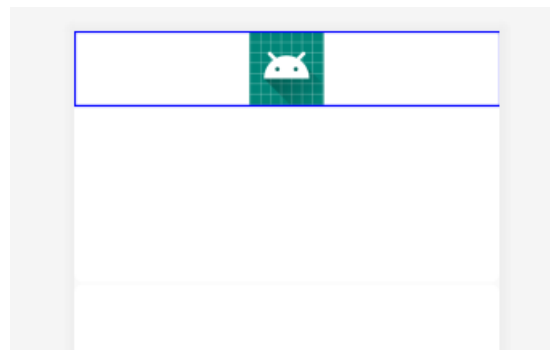
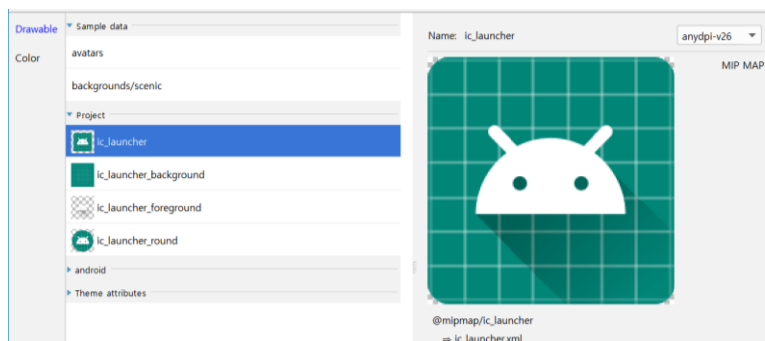
▶ 카드뷰 구성

▷ 카드 뷰의 왼쪽은 설명(텍스트뷰2개)이고 오른쪽은 이미지 뷰를 삽입할 예정

▷ 이미지를 고정으로 설정한 후 텍스트 크기를 가변적으로 설정할 예정



▷ 카드 뷰에 리니어 레이아웃(horizontal) 추가한 후 내부에 이미지 뷰 추가



로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

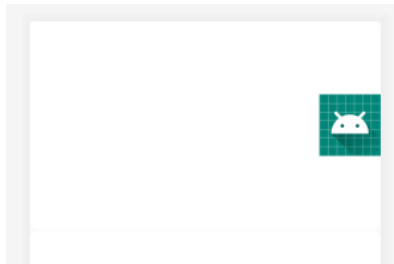
▶ 카드뷰 구성

▶ 카드 뷰내의 레이아웃의 속성을 아래와 같이 변경

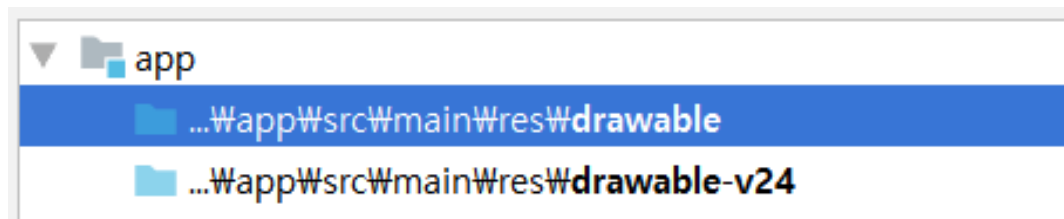
- gravity = right & center_vertical

▶ 이미지 뷰 좌측에 리니어 레이아웃(vertical) 추가 후 너비를 wrap_content로 변경

▶ 추가한 이미지 뷰의 너비와 높이를 140dp로 고정 / weight 속성을 1로 설정



▶ 강의 홈페이지의 drawable.zip을 다운로드 후 프로젝트에 모두 추가



로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

▶ 이미지 뷰의 이미지 변경

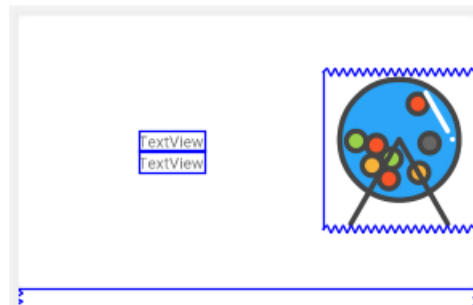
▷ scrCompat 속성에서 lottery 이미지를 선택한 후 좌측에 있는 레이아웃의 weight를 2로 변경

▷ 좌측의 레이아웃에 TextView 2개 배치



▷ 텍스트 뷰가 한쪽으로 쏠려 있으므로 텍스트 뷰를 관리하는 레이아웃의 gravity = center

▷ 텍스트 뷰의 width = wrap_content

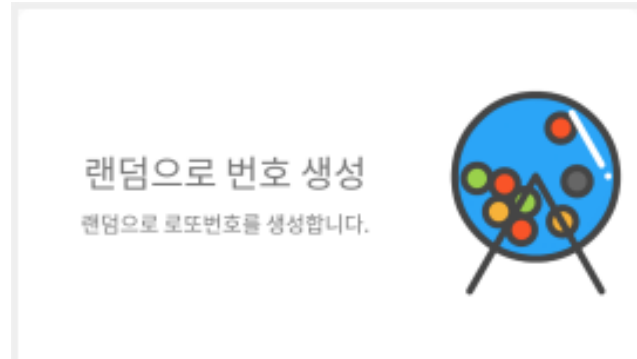


로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

▶ 텍스트 뷰 조정

- ▶ 첫번째 텍스트 뷰의 textSize = 24sp로 설정하고 텍스트뷰 사이에 여백을 주기 위하여 두번째 텍스트 뷰의 margin_top 속성을 8dp로 설정
- ▶ 텍스트를 보기와 같이 변경



- ▶ 나머지 카드뷰에도 리니어 레이아웃을 복사하여 붙여넣기
 - 각각의 카드뷰에 있는 이미지 변경 - constellation, name

로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

▶ 카드뷰 모서리 설정

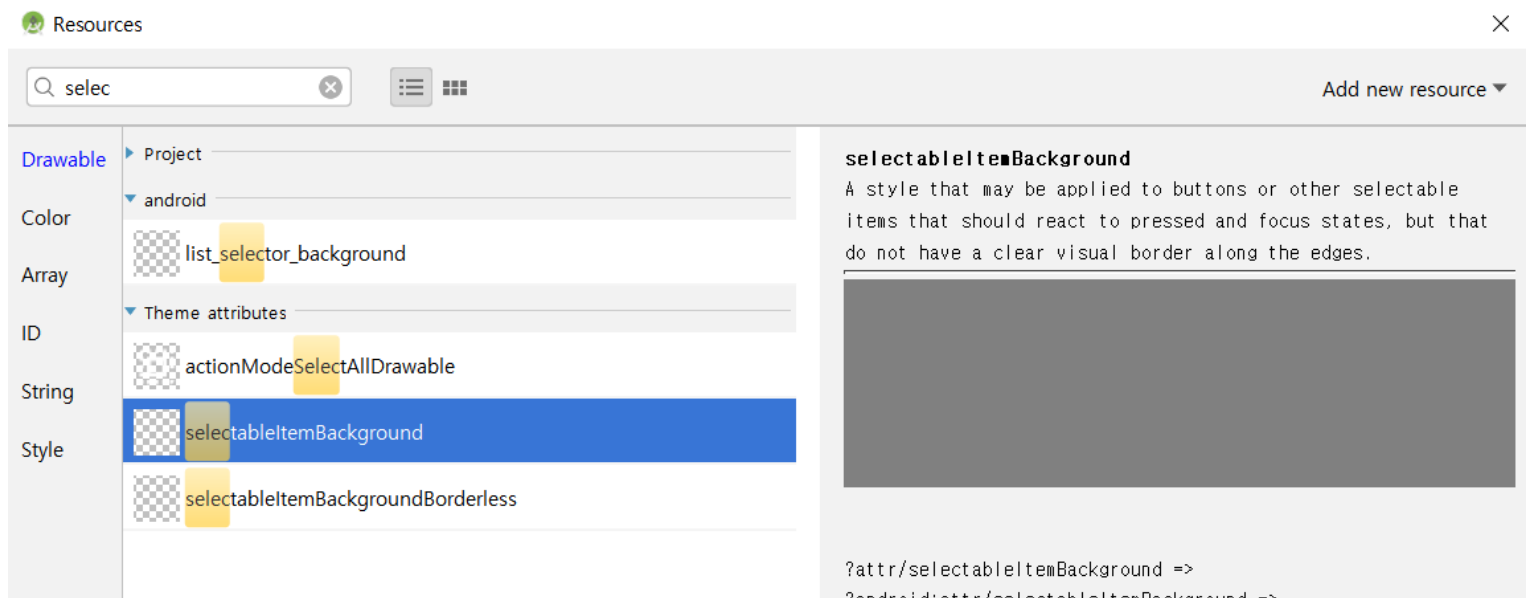
- ▶ `cardUseCompatPadding` 항목에 체크
- ▶ 2,3번째 이미지가 약간 커보이므로 각각의 이미지 뷰에 `padding / all = 8dp`씩 적용



로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

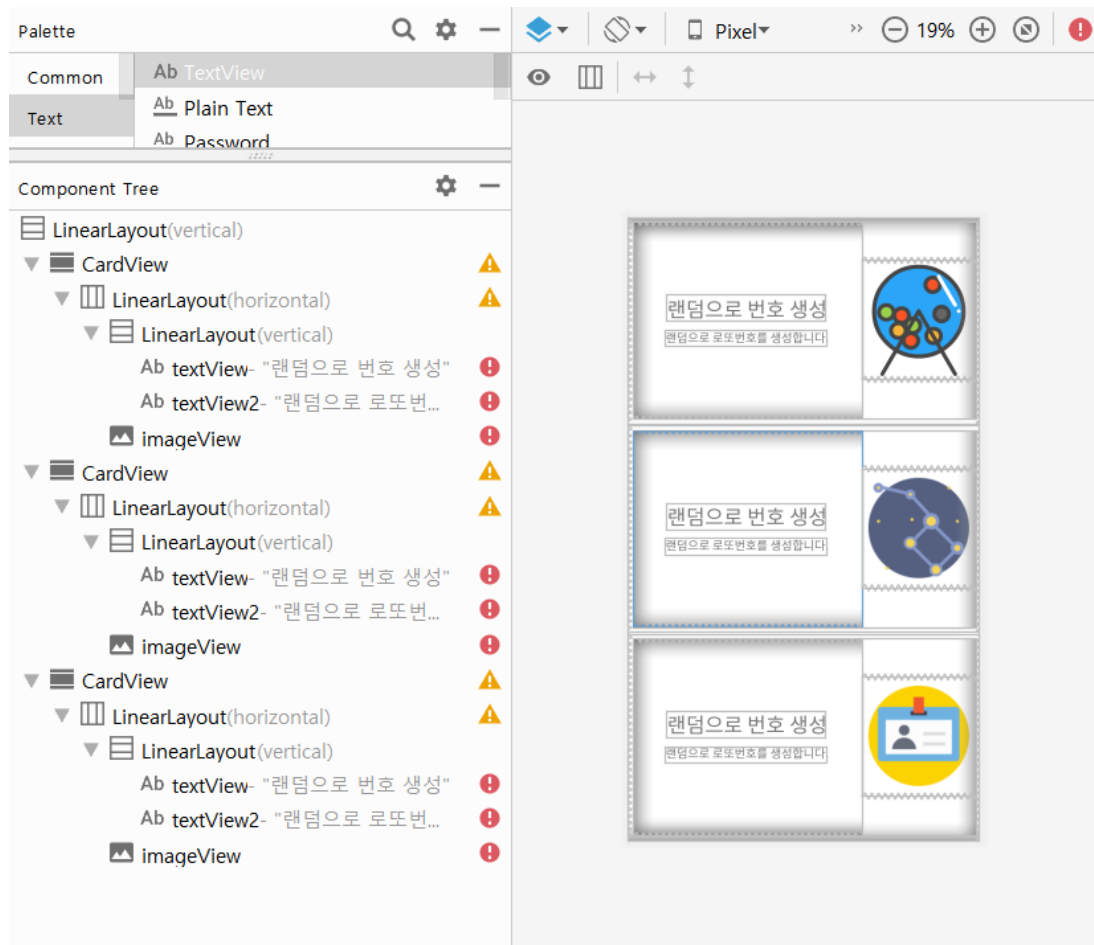
- ▶ 카드 클릭 시 피드백 효과를 위하여 cardView의 foreground와 background 속성에 selectableItemBackground를 추가



로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

▶ 카드 뷰는 기본적으로 Click 이벤트가 비활성화 되어 있기 때문에 clickable 속성을 true로 설정



로또 번호 생성 앱 만들기

▶ 메인 화면 UI 작성

- ▶ 복사/붙여넣기로 인한 ID 중복 제거를 위하여 빨간색 느낌표가 있는 것을 확인 - 에러
- ▶ 해당 View의 ID는 사용하는 것이 아니므로 모두 삭제
- ▶ 여기까지 진행하였으면 첫 화면 디자인 완성! - 실행하여 확인해 직접 확인해보세요.

로또 번호 생성 앱 만들기

▶ Relative 레이아웃으로 별자리 입력 화면 구현

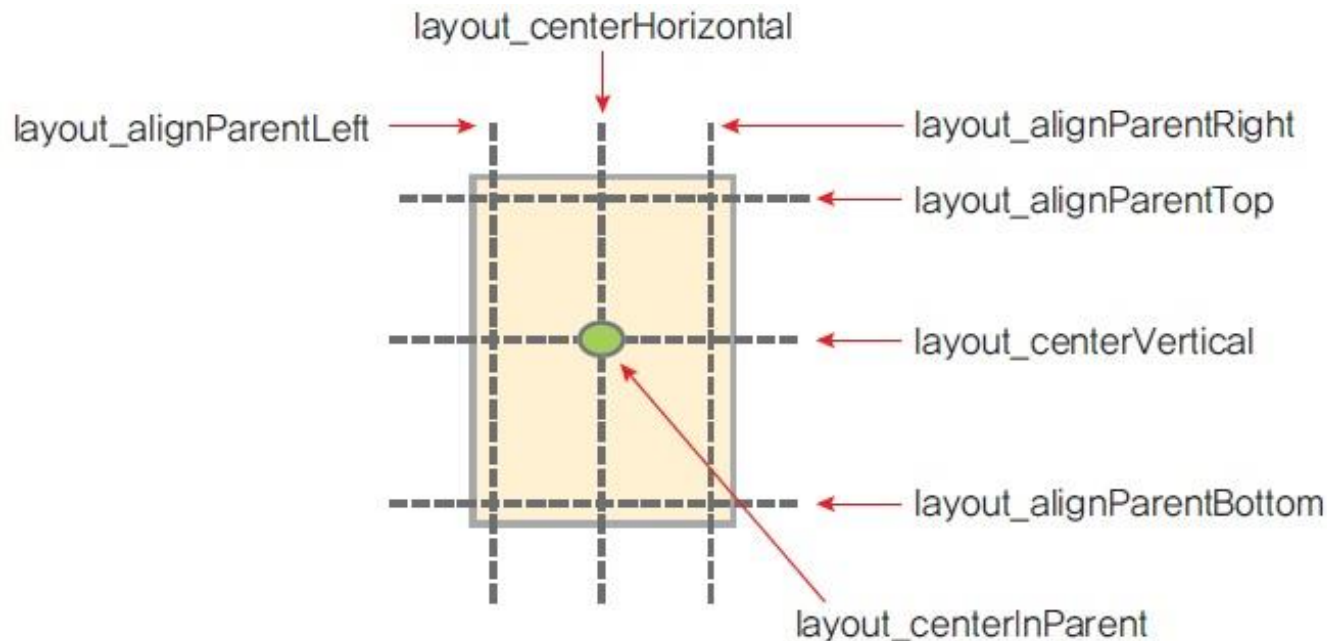
▶ RelativeLayout는 부모 뷰나 다른 뷰와의 상대적 위치 관계로 뷰를 배치

▷ 부모 뷰를 기준으로 배치하는 방법

▷ 부모 뷰가 같은 형제 뷰를 기준으로 배치하는 방법

▶ 부모 뷰를 기준으로 상대적 위치를 지정하는 속성

▷ 해당 속성이 true 경우 적용



로또 번호 생성 앱 만들기

▶ Relative 레이아웃으로 별자리 입력 화면 구현

▶ 부모 뷰를 기준으로 상대적 위치를 지정하는 속성

▶ 해당 속성이 true 경우 적용



부모뷰



자식뷰



`android:layout_alignParentLeft`



`android:layout_alignParentTop`



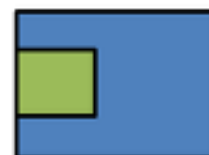
`android:layout_alignParentRight`



`android:layout_alignParentBottom`



`android:layout_centerHorizontal`



`android:layout_centerVertical`



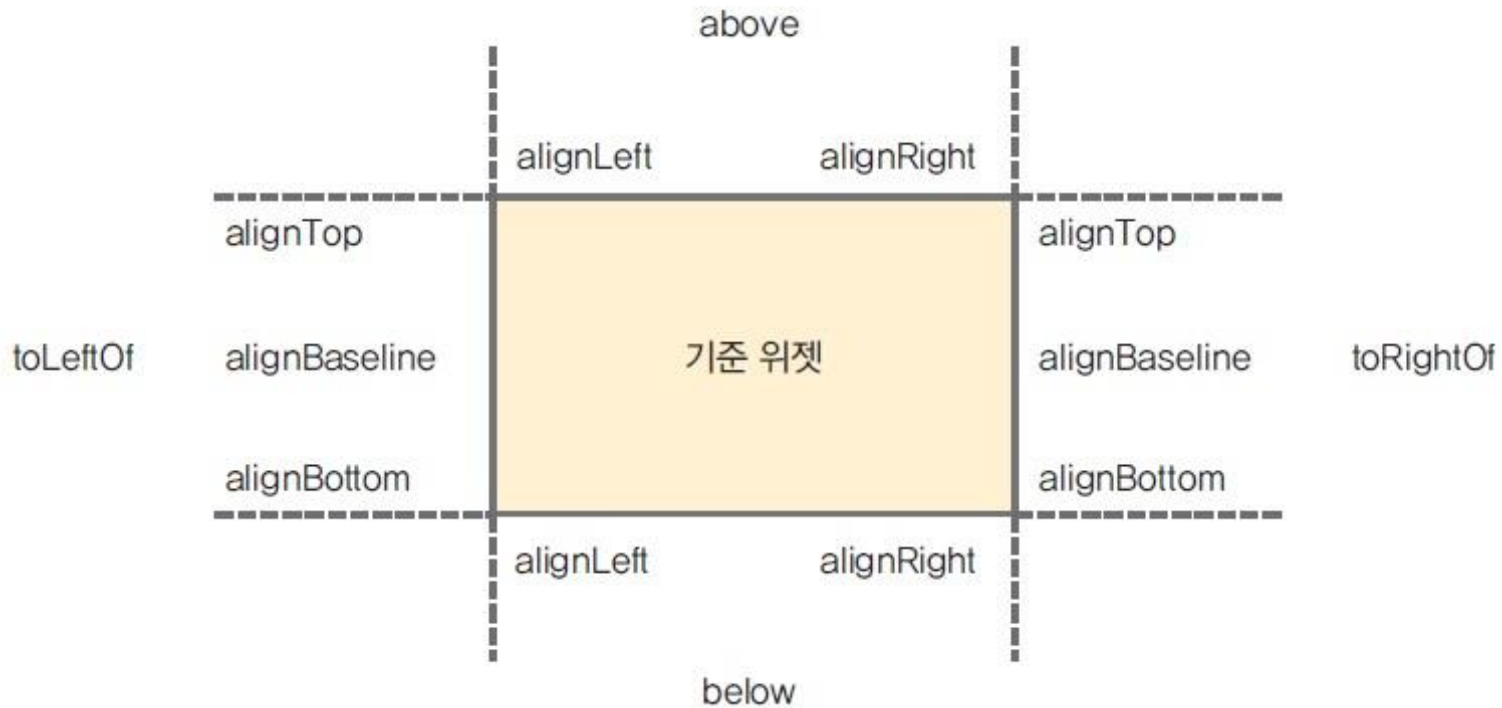
`android:layout_centerInParent`

로또 번호 생성 앱 만들기

▶ Relative 레이아웃으로 별자리 입력 화면 구현

▶ 부모 뷰가 동일한 형제 뷰끼리 상대적인 위치를 지정할 때 속성

▶ `android:layout_above = @+id/형제뷰 아이디`

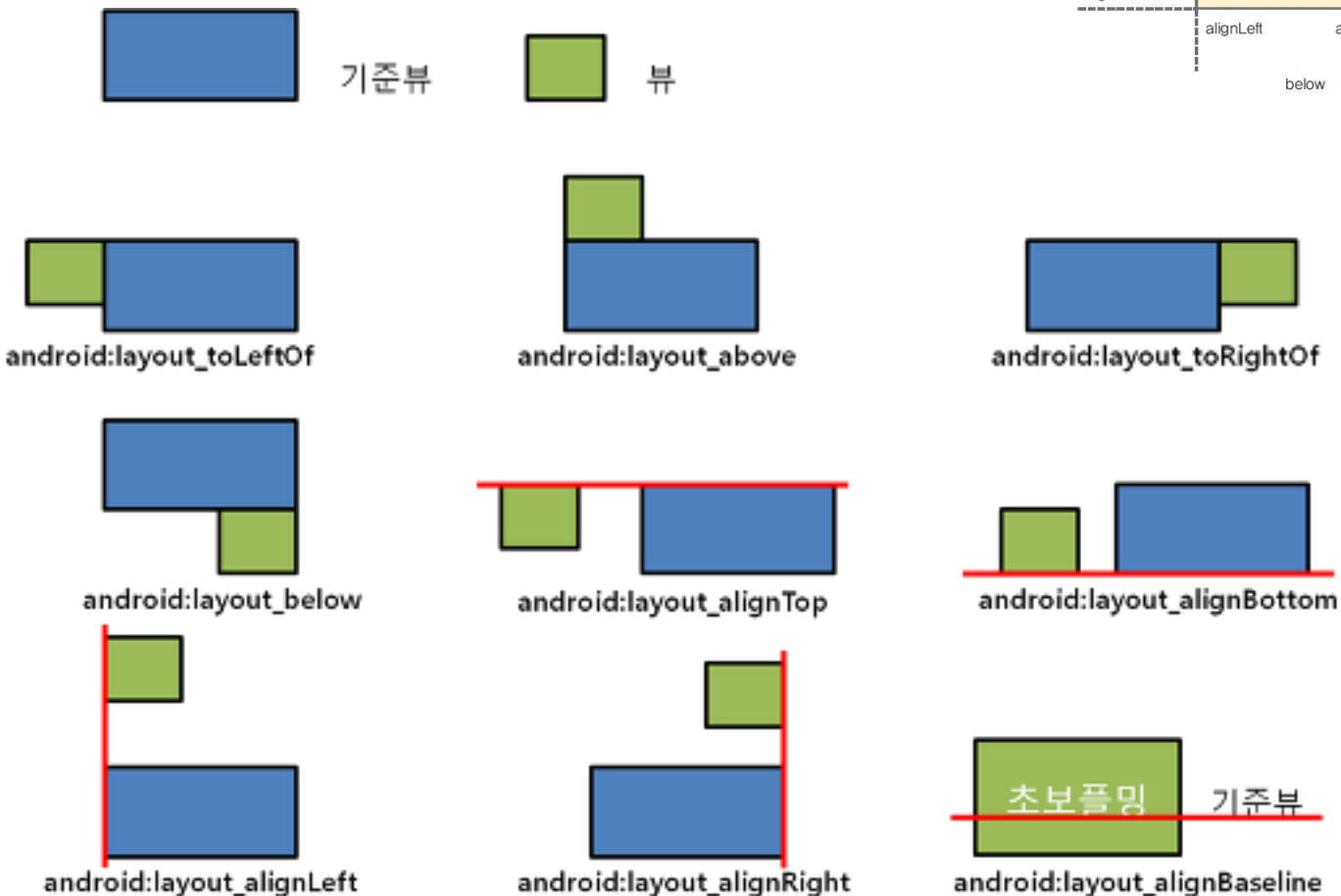
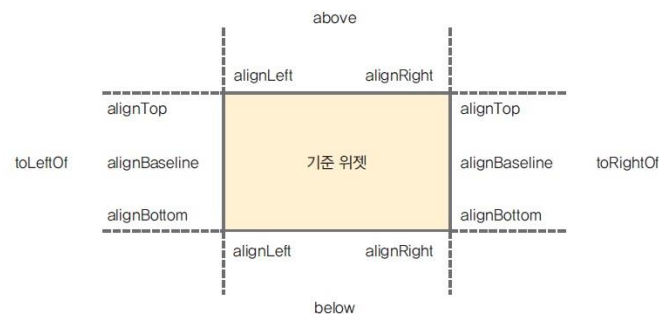


로또 번호 생성 앱 만들기

▶ Relative 레이아웃으로 별자리 입력 화면 구현

▶ 부모 뷰가 동일한 형제 뷰끼리 상대적인 위치를 지정할 때 속성

▶ `android:layout_above = @+id/형제뷰 아이디`



로또 번호 생성 앱 만들기

▶ 부모 뷰 기준 배치

▶ activity_constellation.xml을 상대 레이아웃으로 구성

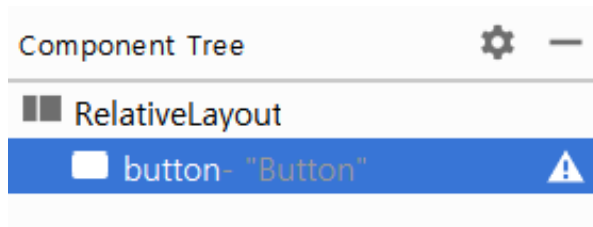
▶ 상대 레이아웃으로 변경

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ConstellationActivity">

</RelativeLayout>
```

▶ 상대 레이아웃에 뷰를 추가할 경우 다른 뷰와의 관계나 margin등이 자동으로 설정되어 번거로우므로 component tree에 추가

▶ 버튼 추가



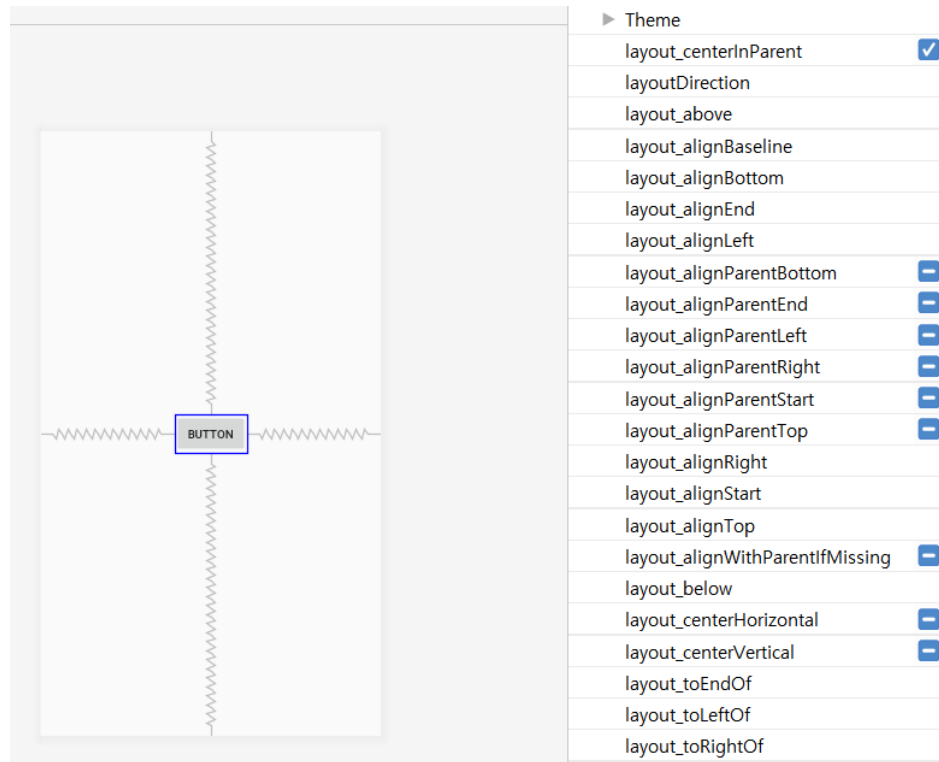
로또 번호 생성 앱 만들기

▶ 부모 뷰 기준 배치

▶ 버튼의 `layout_centerInParent` 속성을 `true`로 설정

▷ 부모 뷰를 기준으로 정중앙에 배치

▷ 멀티해상도를 완벽하게 지원하기 때문에 Device를 변경해도 위치 변화 없음

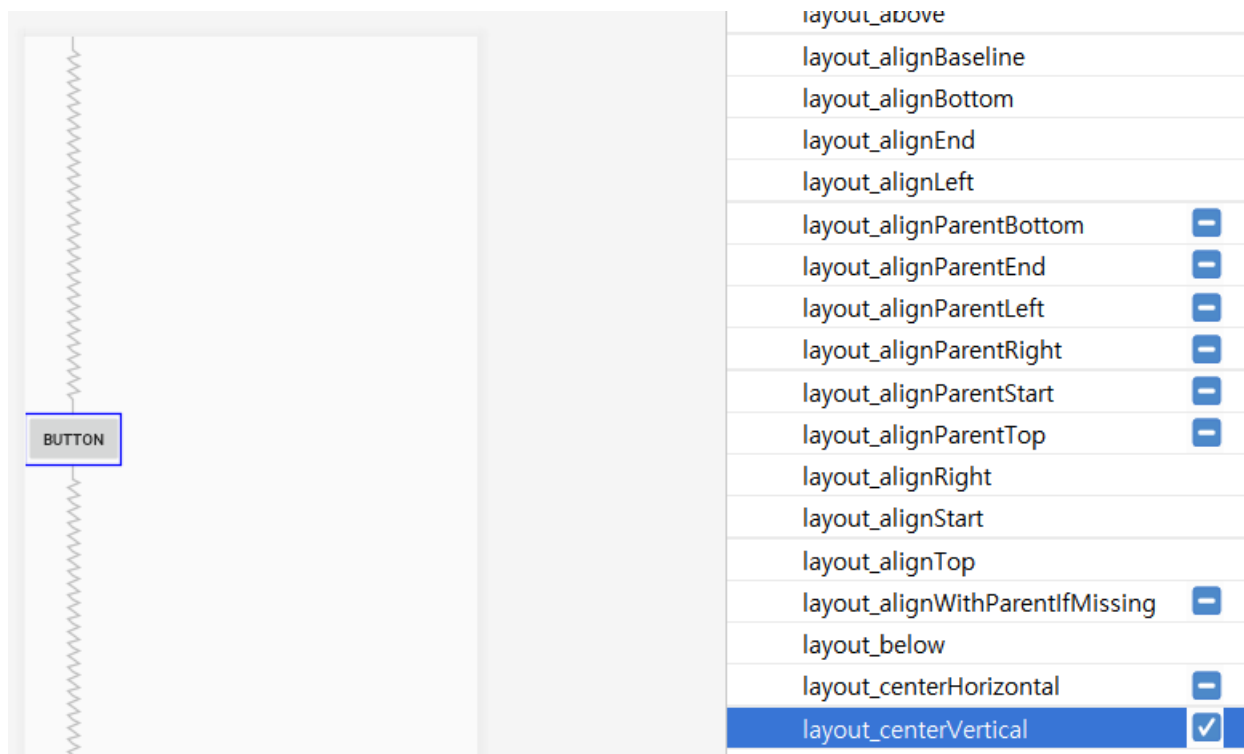


로또 번호 생성 앱 만들기

▶ 부모 뷰 기준 배치

▶ 버튼의 `layout_centerInParent` 속성을 해제하고 `layout_centerVertical` 속성 체크

▷ 부모 뷰 기준으로 가로 좌측, 세로 정중앙으로 배치



로또 번호 생성 앱 만들기

▶ 형제 뷰 기본 배치

▶ 예제

```
1 <RelativeLayout xmlns:android="http://www."
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="기준 위젯" />
11    <Button
12        android:layout_alignTop="@+id/baseBtn"
13        android:layout_toLeftOf="@+id/baseBtn"
14        android:text="1번" />
```



로또 번호 생성 앱 만들기

▶ 형제 뷰 기본 배치

▶ 예제

```
15      ~~~~ 중간 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>
```



실습 문제3

- ▶ 보기와 같이 상대 레이아웃으로 작성하시오.
- ▶ 기준1,2는 부모 Relative layout 기준으로 작성
- ▶ 1,2버튼은 기준 버튼의 상대적 위치로 작성

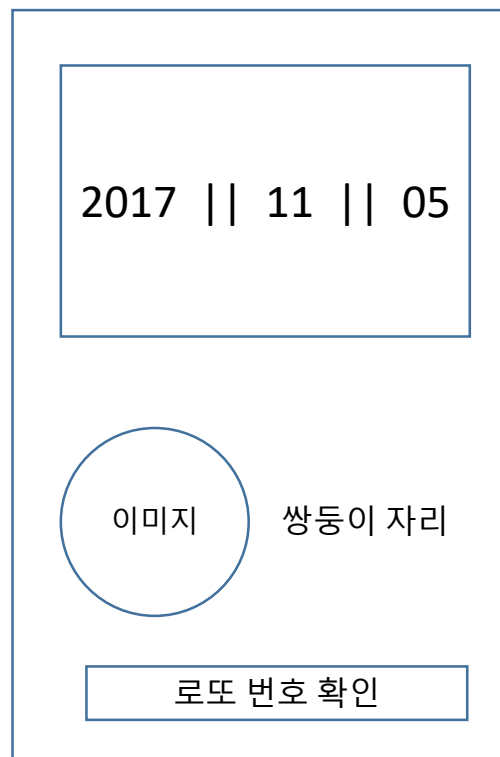


로또 번호 생성 앱 만들기

▶ Relative 레이아웃으로 앱 화면 구성

▶ 별자리 입력 화면 구성

- ▶ 상대 레이아웃으로 화면을 구성할 경우 기준으로 선정할 View를 먼저 찾음
- ▶ 아래 로또 확인 버튼을 부모 뷰의 하단과 일치 시키고 그 위에 이미지와 텍스트를 배치한 후 그 위에 달력 위젯을 배치
- ▶ RelativeLayout은 디자인 모드로 할 경우 자동 관계 설정으로 인하여 번거로움
- ▶ 텍스트 모드에서 진행



로또 번호 생성 앱 만들기

▶ Relative 레이아웃으로 앱 화면 구성

▶ 버튼 삽입

```
<Button
    android:id="@+id/goResultButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_margin="16dp"
    android:text="로또번호확인" />
```

▶ 달력 삽입

```
<DatePicker
    android:id="@+id/datePicker"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:calendarViewShown="false"
    android:datePickerMode="spinner" />
```


로또 번호 생성 앱 만들기

▶ Relative 레이아웃으로 앱 화면 구성

▶ 이미지 뷰 삽입

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="120dp"
    android:layout_height="match_parent"
    android:layout_above="@+id/goResultButton"
    android:layout_below="@+id/datePicker"
    android:layout_margin="16dp"
    android:src="@drawable/constellation" />
```

로또 번호 생성 앱 만들기

▶ Relative 레이아웃으로 앱 화면 구성

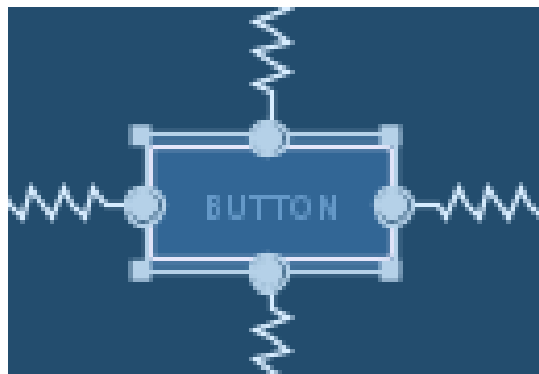
▶ 텍스트 뷰 삽입

```
<android.support.v7.widget.AppCompatTextView
    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/imageView"
    android:layout_alignTop="@+id/imageView"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="32dp"
    android:layout_toRightOf="@+id/imageView"
    android:gravity="center"
    android:maxLines="1"
    android:text="쌍둥이자리"
    android:textColor="@android:color/black"
    app:autoSizeMaxTextSize="48sp"
    app:autoSizeMinTextSize="24sp"
    app:autoSizeStepGranularity="1sp"
    app:autoSizeTextType="uniform" />
```

로또 번호 생성 앱 만들기

▶ ConstraintLayout으로 이름 입력 화면 구현

- ▶ ConstraintLayout은 드래그 앤 드롭하여 개발하기에 가장 좋은 레이아웃
- ▶ google에서 사용을 강력하게 권장
- ▶ 적어도 하나 씩의 수평제약과 수직 제약을 가져야 함
 - ▷ 이들은 View의 세로축과 가로축의 위치를 각각 지정
- ▶ 제약 조건들은 부모 뷰나 다른 View를 기준으로 설정
- ▶ 제약 조건을 추가할 때는 블루프린트 화면을 사용하면 더욱 편리하게 추가
- ▶ 제약 조건에서 뷰는 4개의 원형 꼭지점을 가짐
 - ▷ 각각 constraintTop, constraintBottom, constraintLeft, constraintRight 속성을 의미



로또 번호 생성 앱 만들기

▶ ConstraintLayout으로 이름 입력 화면 구현

▶ 제약 속성

`app:layout_constraintBottom_toBottomOf="@id/target"` 지정한 뷰의 하단과 자신의 하단을 제약으로 연결

`app:layout_constraintBottom_toTopOf="@id/target"` 지정한 뷰의 하단과 자신의 상단을 제약으로 연결

`app:layout_constraintEnd_toEndOf="@id/target"`

`app:layout_constraintEnd_toStartOf="@id/target"`

`app:layout_constraintLeft_toLeftOf="@id/target"`

`app:layout_constraintLeft_toRightOf="@id/target"`

`app:layout_constraintRight_toLeftOf="@id/target"`

`app:layout_constraintRight_toRightOf="@id/target"`

`app:layout_constraintStart_toEndOf="@id/target"`

`app:layout_constraintStart_toStartOf="@id/target"`

`app:layout_constraintTop_toBottomOf="@id/target"`

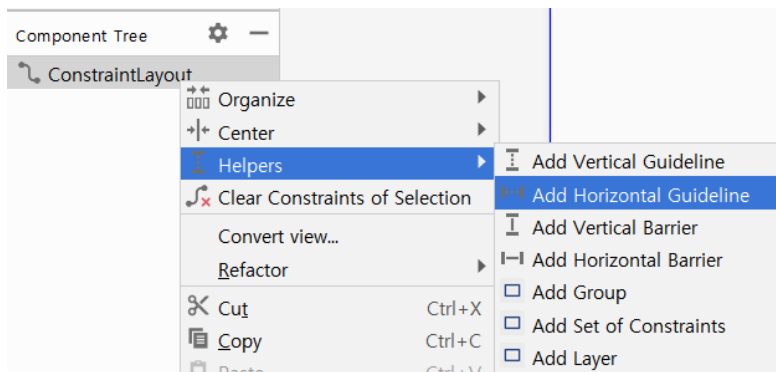
`app:layout_constraintTop_toTopOf="@id/target"` 지정한 뷰의 상단과 자신의 상단을 제약으로 연결

로또 번호 생성 앱 만들기

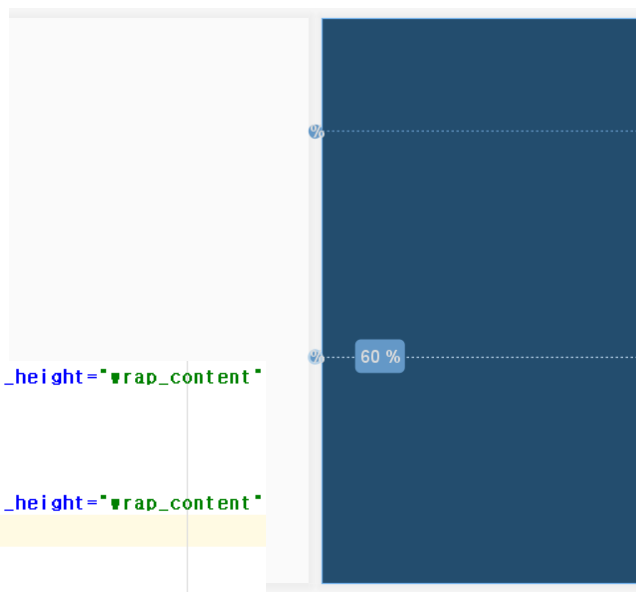
▶ ConstraintLayout으로 이름 입력 화면 구현

▶ 가이드 라인 생성

- ▶ 가이드 라인은 뷰를 쉽게 배치할 수 있도록 기준선을 추가하는 것
- ▶ 가이드라인을 2개 추가하고 id를 guideline1, guideline2로 지정
 - 퍼센트 기준으로 각각 20%, 60% 위치에 각각 배치
- ▶ 컴포넌트 트리에서 레이아웃 우 클릭 후 추가



```
<android.support.constraint.Guideline android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:id="@+id/guideline1"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.2"/>
<android.support.constraint.Guideline android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:id="@+id/guideline2"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.6"/>
```



로또 번호 생성 앱 만들기

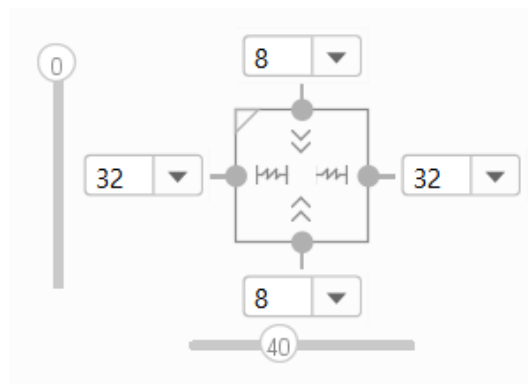
▶ ConstraintLayout으로 이름 입력 화면 구현

▶ 이미지 뷰 추가

- ▶ 이미지 뷰를 가이드 라인 사이에 배치하고 이미지 지정 - name
- ▶ 이미지 뷰의 상하 제약을 두개의 가이드 라인과, 좌우 제약을 부모와 연결
- ▶ 이미지 뷰의 너비와 높이는 match_parent로 설정

▶ 에디트 텍스트 추가

- ▶ 팔레트에서 plain Text를 가이드 라인2의 아래에 배치하고 id를 editText로 변경
- ▶ 상단 제약을 가이드 라인2와 연결하고 좌우는 부모와 연결
- ▶ editText의 Vertical Bias를 0으로 설정(제약의 범위 내에서 뷰의 위치 서정)
- ▶ EditText의 너비를 match_parent로 설정
- ▶ 좌우 제약을 32dp로 설정
- ▶ hint = “이름을 변경하세요.”
- ▶ text는 삭제



로또 번호 생성 앱 만들기

▶ ConstraintLayout으로 이름 입력 화면 구현

▶ vertical 가이드라인 추가

- ▷ id : guideline3

- ▷ 속성을 %로 변경하고 50% 위치에 배치

▶ 버튼 추가

- ▷ 버튼 2개를 에디트텍스트 하단에 수평으로 배치

 - id = goButton, backButton

- ▷ goButton의 좌측 제약은 에디트 텍스트와, 우측은 vertical 가이드와 연결

- ▷ backButton의 우측 제약은 에디트 텍스트와, 좌측은 vertical 가이드와 연결

- ▷ 두 버튼의 상단은 에디트 텍스트와, 하단은 부모 뷰와 연결

- ▷ 너비를 각각 match_... 로 설정

- ▷ vertical bias를 0으로 변경

로또 번호 생성 앱 만들기

▶ ConstraintLayout으로 이름 입력 화면 구현

▶ 버튼 추가(계속)

▷ goButton의 좌측 제약을 0dp로 변경

▷ backButton의 우측제약을 0dp로 변경

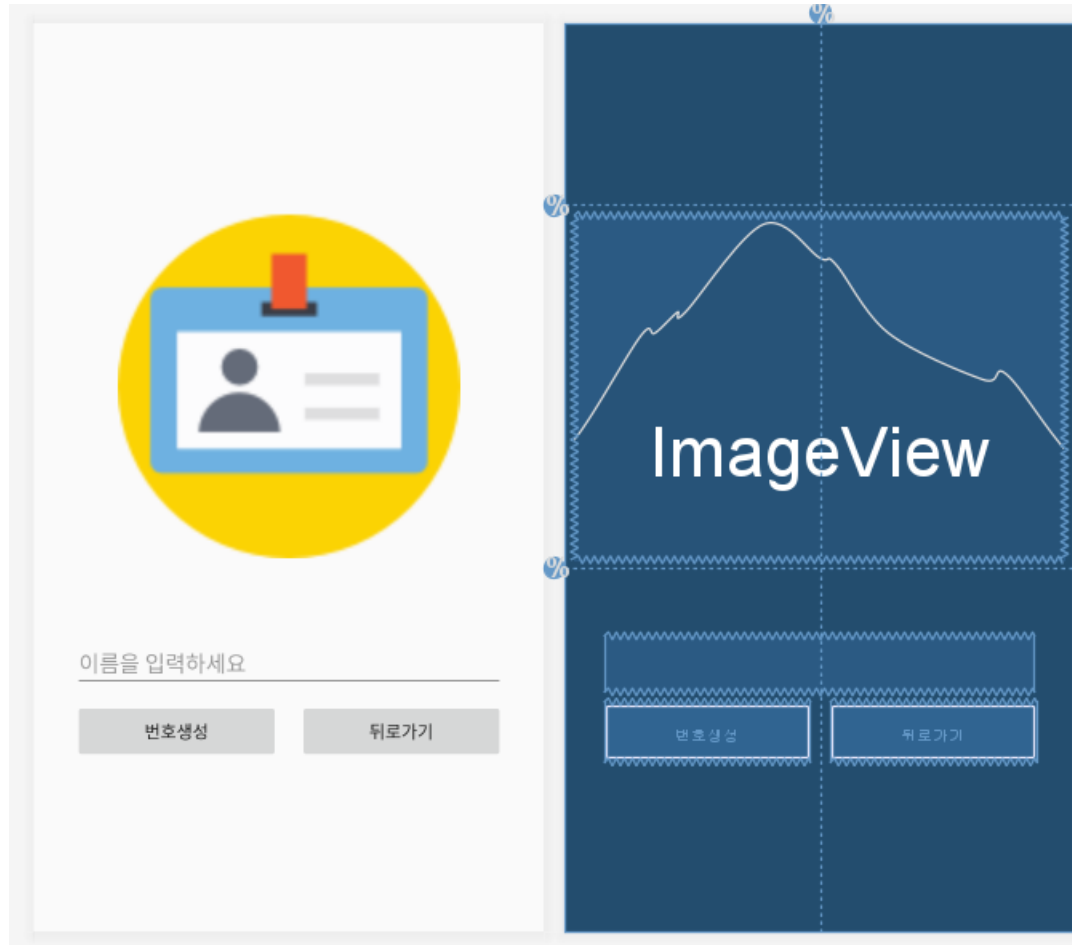
▷ 버튼 텍스트 변경

■ goButton = “번호생성“, backButton = “뒤로가기”

로또 번호 생성 앱 만들기

▶ ConstraintLayout으로 이름 입력 화면 구현

▶ 화면 완성



로또 번호 생성 앱 만들기

▶ 테마 설정

- ▶ 안드로이드에서는 스타일(style)과 테마(Theme)을 xml로 분리하여 관리 가능 - 재사용
 - ▷ 스타일은 뷰의 요소에, 테마는 전체 앱 또는 액티비티에 적용가능한 스타일
- ▶ 일반적으로 앱에서 사용할 기본 컬러들을 테마에서 지정
 - ▷ 사용하는 색상의 통일감이 디자인적으로 중요
 - ▷ 안드로이드는 colorPrimary, colorPrimaryDark, colorAccent의 세 가지 요소를 기반으로 UI를 구성하도록 플랫폼화 되어 있음
 - 버튼, 에디트 텍스트 등의 기본 컴포넌트의 전반적인 요소가 적용됨
 - ▷ google에서는 컬러 조합에 대한 가이드를 제공 - material color

로또 번호 생성 앱 만들기

▶ 색상 컨셉 설정

▶ 컬러 컨셉 추가

▷ app - res - values - colors.xml 파일을 수정

▷ 각각의 화면에서 사용하도록 접미사(Result, Constellation, Name)를 붙여서 분류

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <!-- 결과화면에서 사용할 컬러 -->
    <color name="colorPrimaryResult">#2196F3</color>
    <color name="colorPrimaryDarkResult">#1976D2</color>
    <color name="colorAccentResult">#FF4081</color>
    <!-- 별자리 입력화면에서 사용할 컬러 -->
    <color name="colorPrimaryConstellation">#673AB7</color>
    <color name="colorPrimaryDarkConstellation">#512DA8</color>
    <color name="colorAccentConstellation">#7E57C2</color>
    <color name="colorContentConstellation">#556080</color>
    <color name="colorButtonConstellation">#673AB7</color>
    <!-- 이름 입력화면에서 사용할 컬러 -->
    <color name="colorPrimaryName">#F9A11F</color>
    <color name="colorPrimaryDarkName">#EF6C00</color>
    <color name="colorAccentName">#FFC107</color>
    <color name="colorContentName">#fcd837</color>
    <color name="colorButtonName">#F9A11F</color>
</resources>
```

로또 번호 생성 앱 만들기

▶ 테마 설정

- ▶ 테마를 생성하는 일반적인 방법은 기본적으로 생성되는 style.xml 파일을 편집하는 것
- ▶ app - res - values - styles.xml 파일을 수정
 - ▷ 최초의 스타일 이름은 AppTheme이고 이후에는 AppTheme.[이름] 형태로 지정
 - ▷ AppTheme의 기본적인 요소들은 상속받고 추가로 지정한 속성만 변경
 - ▷ 지정하지 않은 속성들은 AppTheme와 동일하게 사용

```
<resources>
```

```
<!-- Base application theme. -->
```

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

```
<!-- Customize your theme here. -->
```

```
<item name="colorPrimary">@color/colorPrimary</item>
```

```
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>
```

```
<item name="colorAccent">@color/colorAccent</item>
```

```
</style>
```

```
<!-- 결과화면 테마 -->
```

```
<style name="AppTheme.Result" parent="AppTheme">
```

```
<!-- Customize your theme here. -->
```

```
<item name="colorPrimary">@color/colorPrimaryResult</item>
```

```
<item name="colorPrimaryDark">@color/colorPrimaryDarkResult</item>
```

```
<item name="colorAccent">@color/colorAccentResult</item>
```

```
</style>
```

로또 번호 생성 앱 만들기

▶ 테마 설정

▶ 테마를 생성하는 일반적인 방법은 기본적으로 생성되는 style.xml 파일을 편집하는 것

▶ app - res - values - styles.xml 파일을 수정(계속)

```
<!-- 별자리 입력화면 테마 -->
<style name="AppTheme.Constellation" parent="AppTheme">
    <item name="colorPrimary">@color/colorPrimaryConstellation</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDarkConstellation</item>
    <item name="colorAccent">@color/colorAccentConstellation</item>
    <!-- 버튼의 컬러 -->
    <item name="colorButtonNormal">@color/colorButtonConstellation</item>
    <!-- 버튼등 텍스트 컬러 -->
    <item name="android:textColor">@android:color/white</item>
</style>

<style name="AppTheme.Name" parent="AppTheme">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimaryName</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDarkName</item>
    <item name="colorAccent">@color/colorAccentName</item>
    <!-- 버튼의 컬러 -->
    <item name="colorButtonNormal">@color/colorButtonName</item>
    <!-- 버튼등 텍스트 컬러 -->
    <item name="android:textColor">@android:color/white</item>
</style>

</resources>
```

로또 번호 생성 앱 만들기

▶ 테마 설정

▶ Activity와 테마를 연결하기 위하여 매니페스트 파일 수정

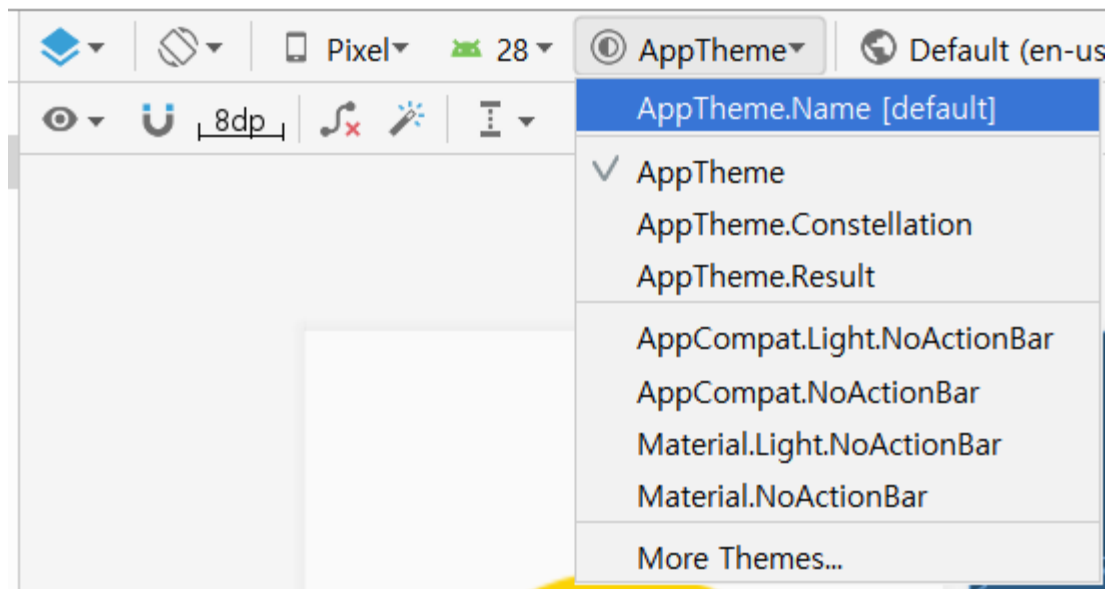
```
<!-- AppTheme.Constellation 테마지정 -->
<activity
    android:name=".ConstellationActivity"
    android:theme="@style/AppTheme.Constellation" />
<!-- AppTheme.Name 테마지정 -->
<activity
    android:name=".NameActivity"
    android:theme="@style/AppTheme.Name" />
<!-- AppTheme.Result 테마지정 -->
<activity
    android:name=".ResultActivity"
    android:theme="@style/AppTheme.Result" />
```

로또 번호 생성 앱 만들기

▶ 테마 설정

▶ 미리 보기

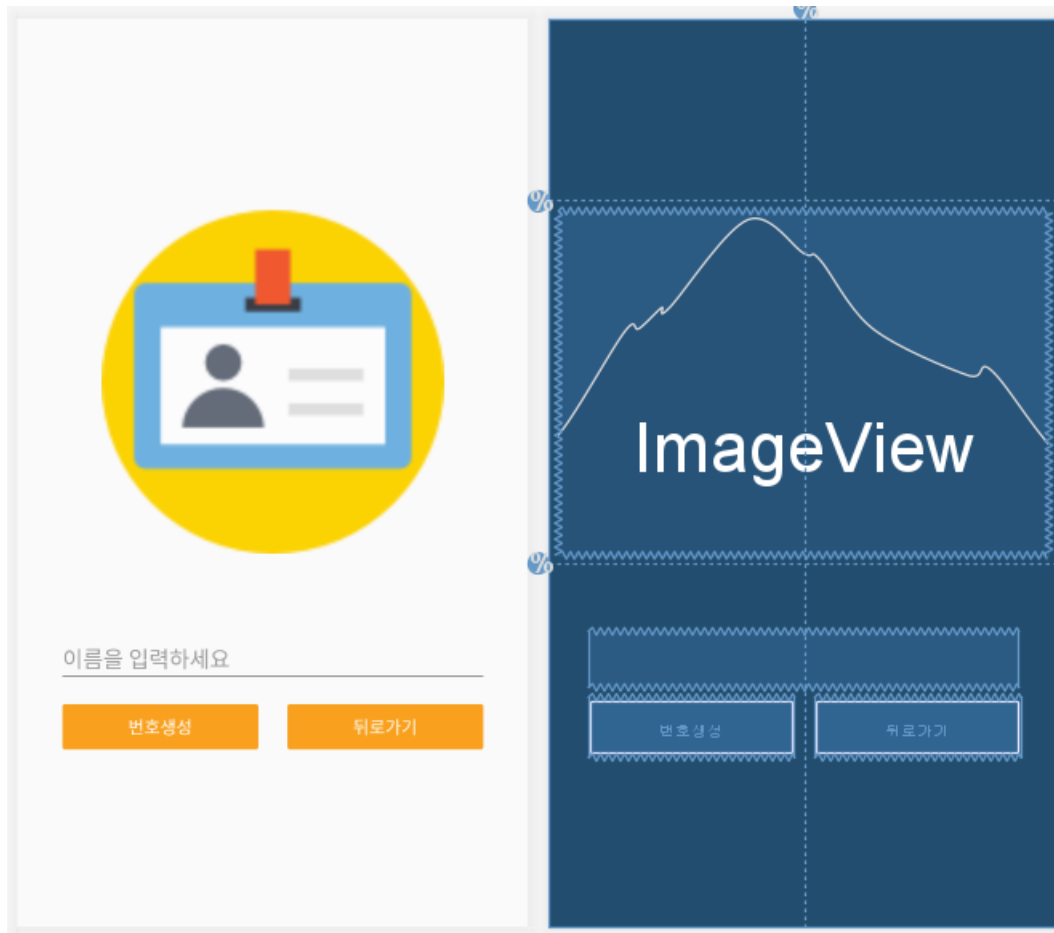
▶ 디자인 탭의 상단에 테마를 변경할 수 있는 툴 요소(App Theme)를 클릭하고 해당 xml 선택



로또 번호 생성 앱 만들기

▶ 테마 설정

▶ 미리 보기



로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

- ▶ activity_result.xml 파일을 열고 UI 구성

- ▶ 가이드라인 배치

- ▶ 여러 해상도에서 자연스러운 UI를 생성하기 위하여 가이드라인을 퍼센트로 설정

- ▶ horizontal 가이드 라인을 2개 추가하고 id를 guideline1, guideline2로 지정

- guideline1을 퍼센트 속성으로 변경하고 30% 위치로 배치

- guideline2을 퍼센트 속성으로 변경하고 50% 위치로 배치

- ▶ vertical 가이드 라인을 2개 추가하고 id를 guideline3, guideline4로 지정

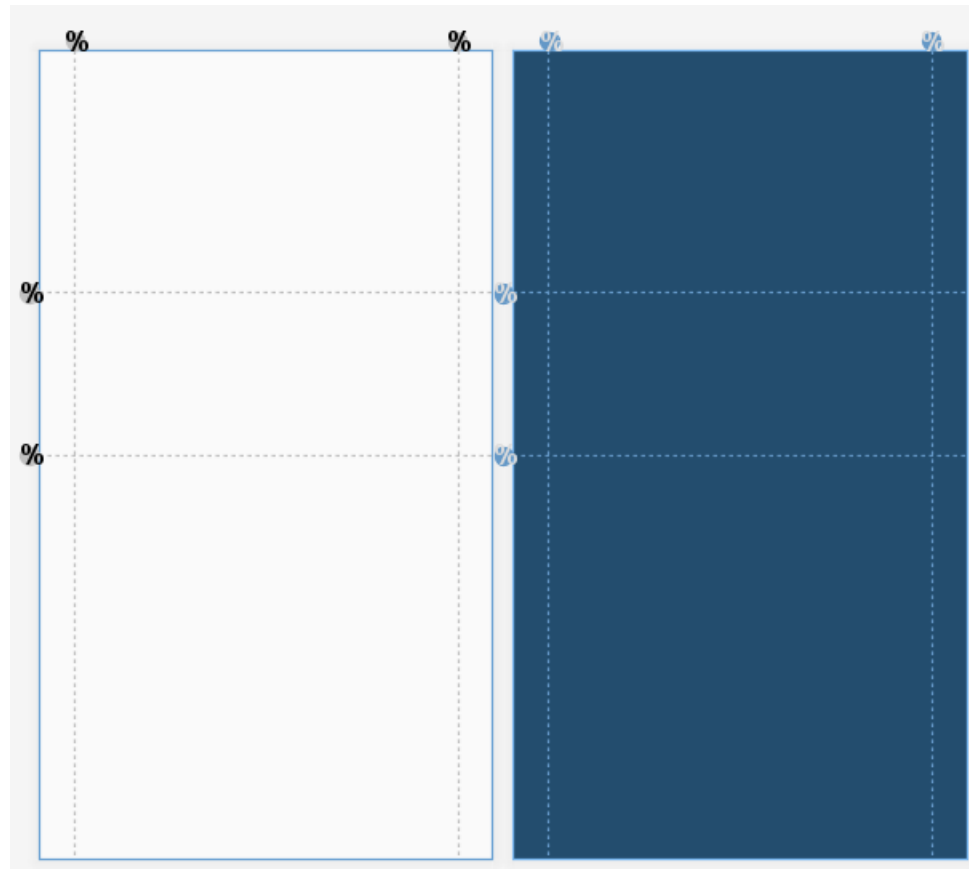
- guideline3을 퍼센트 속성으로 변경하고 8% 위치로 배치

- guideline4을 퍼센트 속성으로 변경하고 92% 위치로 배치

로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ 가이드라인 배치

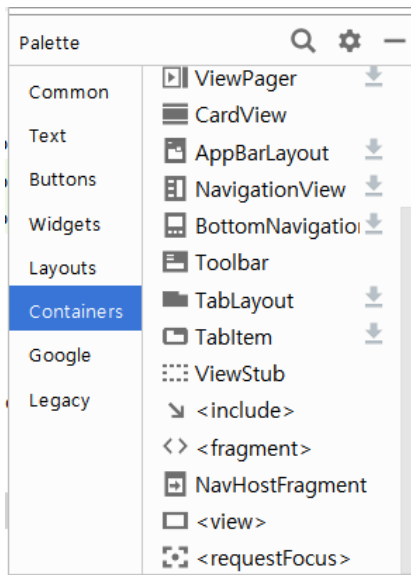


로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ AppCompatActivity 생성

- ▶ View의 크기에 따라 Text크기가 변경되도록 AppCompatActivity 사용
 - View의 크기가 작게 변경될 때 Text의 크기가 고정이라면 문자열이 잘리거나 줄 바꿈이 일어남
- ▶ 구형의 안드로이드 버전에서도 해당 기능이 적용되게 하려면 AppCompatActivity를 사용해야 함
- ▶ AppCompatActivity는 SDK에 기본적으로 포함되지 않으므로 view 항목을 사용

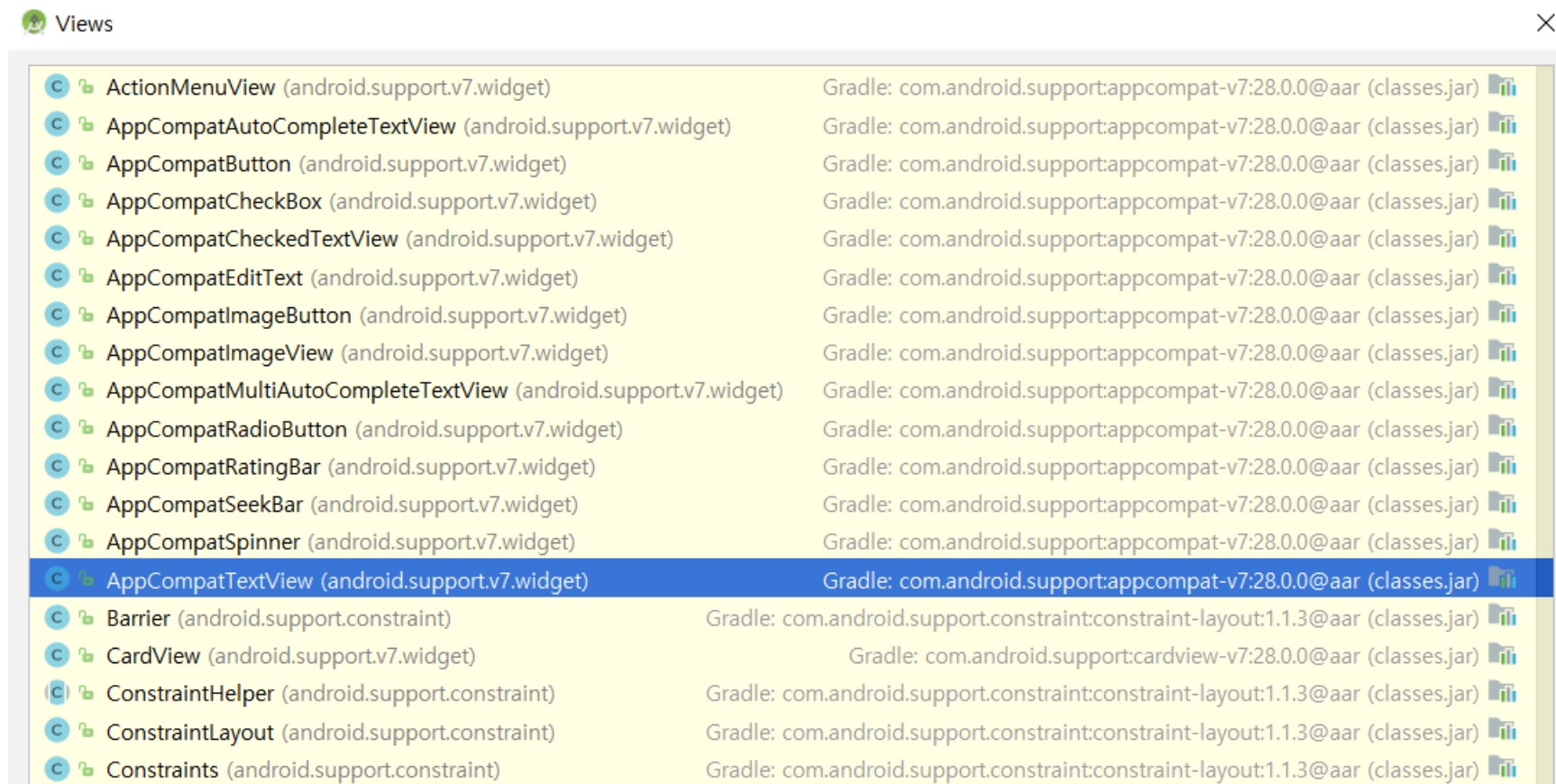


로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ AppCompatActivity 생성

▶ view를 화면에 드래그 앤 드랍하면 실제로 구현된 클래스를 선택할 수 있음



로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ AppCompatActivity 생성

▶ 해당 뷰를 추가하면 버그로 인하여 id의 형식이 잘못 지정되어 있으므로 수정

- id="@+id/view" → android:id="@+id/resultLabel"

▶ 좌측과 우측을 각각 guideline3, guideline4와 연결

▶ 좌우 여백을 16dp로 설정

▶ 상단을 부모와, 하단은 guideline1과 연결

▶ 가로 세로를 match_...로 설정

▶ textSize = 60sp, text = “홍길동님의 ₩n 로또 번호 입니다.”

▶ gravity = center / maxLine = 3

▶ autoSizeMaxTextSize = 60sp / autoSizeMinTextSize = 16sp

▶ autoSizeStepGranularity = 1sp / autoSizeTextType = uniform

로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

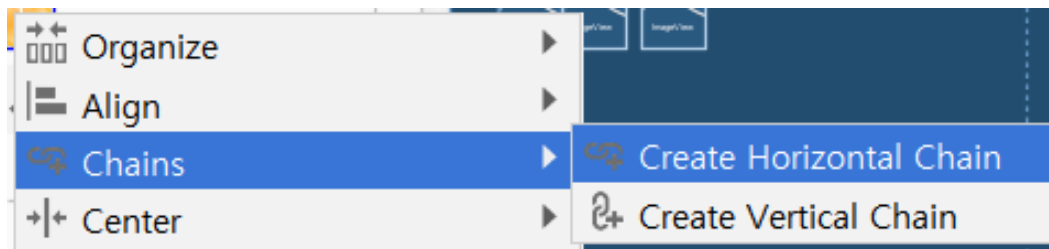
▶ 로또 공 이미지 배치

▷ 결과로 나오는 로또 공 이미지를 UI에 배치

▷ 이미지 뷰 6개를 guideline1과 guideline2 사이에 배치

- id = imageView01, imageView02, imageView03, imageView04, imageView05, imageView06

▷ shift 키를 이용하여 이미지 뷰를 동시에 선택하고 horizontal 체인으로 연결



▷ 체인 전체의 좌우 제약을 각각 guideline3과 guideline4와 연결

▷ 각 이미지 뷰를 아래와 같이 설정

- 상하제약을 guideline1과 guideline2와 연결
- 너비와 높이는 match_... / 좌우 여백은 8dp ➔ 이미지뷰는 자동으로 커지고 여백이 있어서 서로 붙지 않음

로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ 하단 고정 이미지 추가

- ▶ 이미지 뷰를 guideline2 하단에 추가하고 id = imageView10, 이미지는 money로 설정
- ▶ 좌측과 우측 제약을 각각 guideline3, guideline4와 연결
- ▶ 상하 제약을 각각 guideline2와 부모 뷰에 연결
- ▶ 너비와 높이는 match_...로 설정

로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ 화면 완성



로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ 인텐트를 활용한 UI 연결

▶ activity_main.xml의 각 카드 뷰에 id 설정

- randomCard, constellationCard, nameCard

▶ MainActivity.kt에 이벤트 리스너를 달고 화면을 넘기는 코드 작성

```
// 랜덤으로 번호 생성 카드의 클릭 이벤트 리스너
randomCard.setOnClickListener {
    // ResultActivity 를 시작하는 Intent 생성
    val intent = Intent(this, ResultActivity::class.java)
    // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행
    startActivity(intent)
}
// 별자리로 번호 생성 카드의 클릭 이벤트 리스너
constellationCard.setOnClickListener {
    // ConstellationActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행
    startActivity(Intent(this, ConstellationActivity::class.java))
}
// 이름으로 번호 생성 카드의 클릭 이벤트 리스너
nameCard.setOnClickListener {
    // NameActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행
    startActivity(Intent(this, NameActivity::class.java))
}
```

로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ ConstellationActivity, NameAcitivity에 결과 화면으로 전환되는 리스너 구현

▶ ConstellationActivity

```
goResultButton.setOnClickListener {  
    // ResultActivity 를 시작하는 Intent 생성  
    val intent = Intent(this, ResultActivity::class.java)  
  
    // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행  
    startActivity(intent)  
}
```

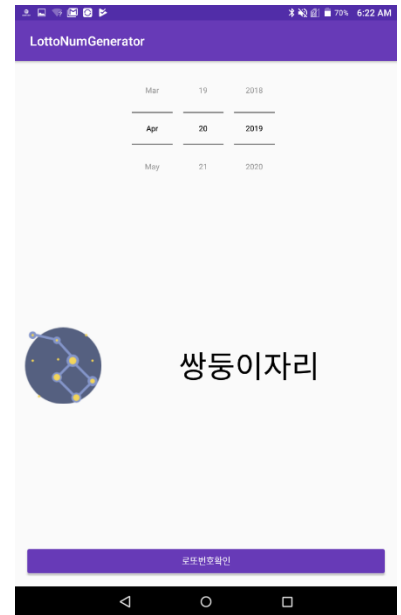
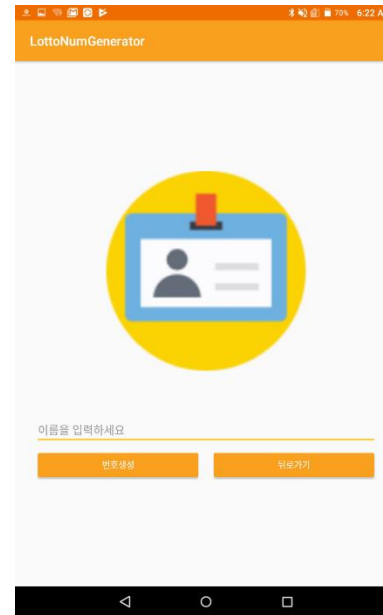
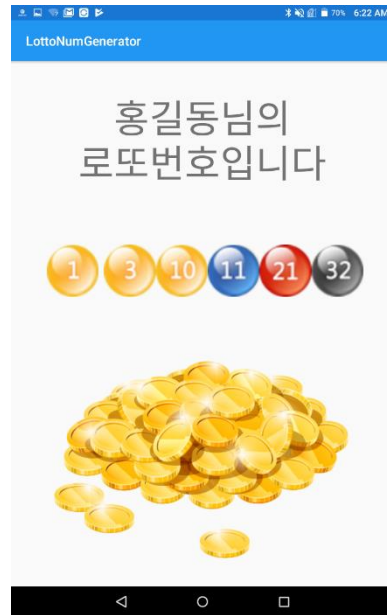
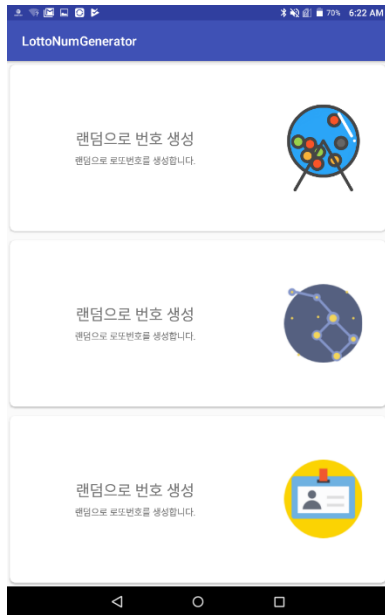
▶ NameAcitivity

```
goResultButton.setOnClickListener {  
    // ResultActivity 를 시작하는 Intent 생성  
    val intent = Intent(this, ResultActivity::class.java)  
  
    // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행  
    startActivity(intent)  
}  
// 뒤로가기 버튼의 클릭이벤트 리스너 설정  
backButton.setOnClickListener {  
    // 액티비티 종료  
    finish()  
}
```

로또 번호 생성 앱 만들기

▶ 결과 화면 만들기

▶ 실행하여 화면 전환 확인



로또 번호 생성 앱 만들기

▶ 난수 만들기

- ▶ 난수는 프로그램에서 무작위로 추출된 난수
- ▶ 컴퓨터에서는 seed값이 같으면 동일한 난수를 생성하기 때문에 엄밀히 말하면 진정한 난수를 존재하지 않음
 - ▷ 컴퓨터는 입력에 대한 출력이 항상 동일하기 때문
- ▶ 의사 난수
 - ▷ 입력 값을 매번 다르게 입력하여 출력된 난수
- ▶ Random 클래스나 Math 클래스로 난수를 생성
 - ▷ 본 프로젝트에서는 Random 클래스를 사용

로또 번호 생성 앱 만들기

▶ ResultActivity.kr 수정

▶ MainActivity.kr 파일에 난수 생성 코드 작성

```
/**
 * 랜덤으로 1 ~ 45 번호중 하나의 번호를 생성하는 함수
 */
fun getRandomLottoNumber(): Int {
    // Random.nextInt 는 0 ~ 전달받은 파라미터 값 미만의 번호를 생성
    // ex) Random().nextInt(10) 은 0 ~ 9 까지의 무작위 수를 반환
    // 1 ~ 45 까지의 번호를 생성하려면 파라미터의 45 를 넣고 결과값의 1을 더한다.
    return Random().nextInt(45) + 1
}
```

로또 번호 생성 앱 만들기

▶ ResultActivity.kr 수정

▶ MainActivity.kr 파일에 난수 생성 코드 작성

```
fun getRandomLottoNumbers(): MutableList<Int> {  
    // 무작위로 생성된 로또 번호를 저장할 가변 리스트 생성  
    val lottoNumbers = mutableListOf<Int>()           //변경 가능한 리스트 타입  
  
    // 6번 반복하는 for 문  
    for (i in 1..6) {  
        // 랜덤한 번호를 임시로 저장할 변수를 생성  
        var number = 0  
        do {  
            // 랜덤한 번호를 추출해 number 변수에 저장  
            number = getRandomLottoNumber()  
  
            // lottoNumbers 에 number 변수의 값이 없을때까지 반복  
        } while (lottoNumbers.contains(number)) //동일한 수가 있다면 true  
  
        // 이미 뽑은 리스트에 없는 번호가 나올때까지 반복했으므로 중복이 없는 상태  
        // 추출된 번호를 뽑은 리스트에 추가  
        lottoNumbers.add(number)  
    }  
    return lottoNumbers  
}
```

로또 번호 생성 앱 만들기

▶ MainActivity.kr 수정

▶ 생성된 로또 번호를 인텐트로 ResultActivity에 전달

```
randomCard.setOnClickListener {  
    // ResultActivity 를 시작하는 Intent 생성  
    val intent = Intent(this, ResultActivity::class.java)  
  
    // intent 의 결과 데이터를 전달한다.  
    // int 의 리스트를 전달하므로 putIntegerArrayListExtra 를 사용한다.  
    intent.putIntegerArrayListExtra("result", ArrayList(getRandomLottoNumbers()))  
  
    // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행  
    startActivity(intent)  
}
```

로또 번호 생성 앱 만들기

▶ ResultActivity.kr 수정

▶ 넘겨받은 인텐트 데이터로 로또번호에 맞는 이미지 출력

```
val lottoImageStartId = R.drawable.ball_01

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_result)

    // 전달받은 결과 배열을 가져온다.
    val result = intent.getIntegerArrayListExtra("result")

    // 전달받은 결과가 있는 경우에만 실행
    result?.let {
        // 결과에 맞게 로또 공 이미지를 업데이트한다.
        // 전달받은 결과는 정렬되어 있지않으므로 정렬해서 전달한다.
        updateLottoBallImage(result.sortedBy { it })
    }
}
```


로또 번호 생성 앱 만들기

▶ ResultActivity.kr 수정

▶ 전달 받은 값으로 로또 공의 이미지를 변경하는데 1번 공의 이미지를 기준으로 상대적인 아이디를 검색

▶ 이미지 아이디는 리소스 이름 오름차순으로 되어있고 실제 리소스 아이디는 정수임을 이용

```
//val lottoImageStartId = R.drawable.ball_01
/**
 * 결과에 따라 로또 공 이미지를 업데이트한다.
 */
fun updateLottoBallImage(result: List<Int>){
    // 결과의 사이즈가 6개 미만인경우 예러가 발생할 수 있으므로 바로 리턴한다.
    if(result.size < 6) return

    // ball_01 이미지 부터 순서대로 이미지 아이디가 있기 때문에
    // ball_01 아이디에 결과값 -1 을 하면 목표하는 이미지가 된다
    // ex) result[0] 이 2번 공인 경우 ball_01 에서 하나 뒤에 이미지가 된다.
    imageView01.setImageResource(lottoImageStartId + (result[0] - 1))
    imageView02.setImageResource(lottoImageStartId + (result[1] - 1))
    imageView03.setImageResource(lottoImageStartId + (result[2] - 1))
    imageView04.setImageResource(lottoImageStartId + (result[3] - 1))
    imageView05.setImageResource(lottoImageStartId + (result[4] - 1))
    imageView06.setImageResource(lottoImageStartId + (result[5] - 1))
}
```

로또 번호 생성 앱 만들기

▶ Shuffle 클래스

▶ 랜덤 값을 추출하기 위한 다른 방법

▶ 앞서 추출한 랜덤 값이 특정 범위에서 무작위로 수 하나를 추출했다면 Shuffle은 이미 존재하는 집합을 섞어서 필요한 만큼 추출

▶ 1~45까지의 번호를 갖는 집합에서 마구 섞은 후 앞에서 순서대로 6개를 자름(추출)

```
/**
 * Shuffle 을 사용해 로또 번호 생성
 */
fun getShuffleLottoNumbers(): MutableList<Int> {
    // 1 ~ 45 번에 로또 번호를 저장할 리스트 생성
    val list = mutableListOf<Int>()

    // 1~45 까지 for 문을 돌면서 리스트에 로또 번호 저장
    for(number in 1..45){
        list.add(number)
    }

    // 리스트를 무작위로 섞는다.
    list.shuffle()

    // 리스트를 앞에서부터 순서대로 6개를 잘라 결과 반환
    return list.subList(0, 6)
}
```

로또 번호 생성 앱 만들기

▶ Shuffle 클래스

▶ ResultActivity.kr 수정

▶ getRandomLottoNumbers() → getShuffleLottoNumbers()

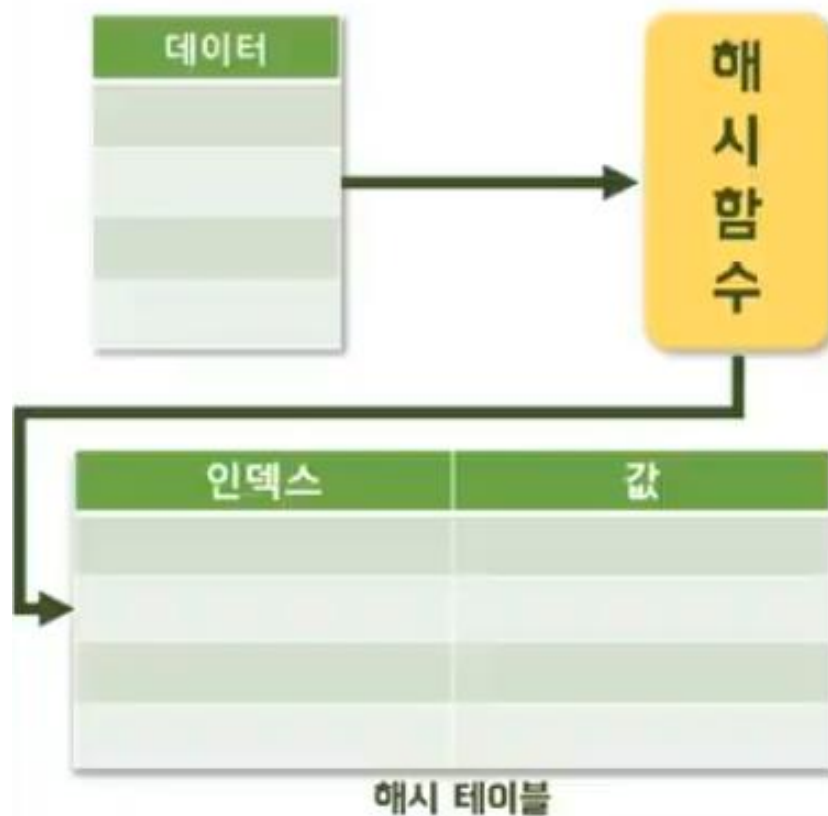
```
randomCard.setOnClickListener {  
  
    // ResultActivity 를 시작하는 Intent 생성  
    val intent = Intent(this, ResultActivity::class.java)  
  
    // intent 의 결과 데이터를 전달한다.  
    // int 의 리스트를 전달하므로 putIntegerArrayListExtra 를 사용한다.  
    //intent.putIntegerArrayListExtra("result", ArrayList(getRandomLottoNumbers()))  
    intent.putIntegerArrayListExtra("result", ArrayList(getShuffleLottoNumbers()))  
  
    // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행  
    startActivity(intent)  
}
```

▶ 랜덤보다 셔플을 사용하는 것이 추출 횟수가 증가할수록 중복될 확률이 낮아짐

로또 번호 생성 앱 만들기

▶ 해시코드(Hash Code) 개요

- ▶ 랜덤으로 번호를 보여주는 것과 달리 별자리나 이름은 입력한 값에 따라 동일한 값을 출력해야 함
- ▶ 코틀린에서는 HashMap과 hashCode()메소드를 이용하여 해시 코드를 사용



로또 번호 생성 앱 만들기

▶ 이름에 따른 로또 번호 생성

▶ 입력 받은 이름의 Hash 코드를 구한 후 Random()의 seed로 사용

▶ 이름 마다 다른 랜덤 값 출력

▶ NameActivity.kt에 아래 코드 삽입

```
/**
 * 입력받은 이름에 대한 해시코드를 사용하여 로또 번호를 섞고 결과를 반환
 */
fun getLottoNumbersFromHash(name: String): MutableList<Int> {
    // 1 ~ 45 번에 로또 번호를 저장할 리스트 생성
    val list = mutableListOf<Int>()

    // 1~45 까지 for 문을 돌면서 리스트에 로또 번호 저장
    for (number in 1..45) {
        list.add(number)
    }

    // 리스트를 무작위로 섞는다. 이때 섞는 기준으로 Random(SEED) 를 사용
    list.shuffle(Random(name.hashCode().toLong()))

    // 리스트를 앞에서부터 순서대로 6개를 잘라서 결과 반환
    return list.subList(0, 6)
}
```

로또 번호 생성 앱 만들기

▶ 이름에 따른 로또 번호 생성

▶ NameActivity.kt코드에 ResultActivity.kt로 로또 번호 추출결과를 인텐트로 전달하는 코드 삽입

```
goButton.setOnClickListener {  
    // ResultActivity 를 시작하는 Intent 생성  
    val intent = Intent(this, ResultActivity::class.java)  
  
    // intent 의 결과 데이터를 전달  
    // int 의 리스트를 전달하므로 putIntegerArrayListExtra 를 사용  
    // 전달하는 리스트는 이름의 해시코드로 생성한 로또번호  
    intent.putIntegerArrayListExtra("result",  
        ArrayList(getLottoNumbersFromHash(editText.text.toString())))  
  
    // 입력받은 이름을 추가로 전달  
    intent.putExtra("name", editText.text.toString())  
  
    // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행  
    startActivity(intent)  
}
```

로또 번호 생성 앱 만들기

▶ 매일 다른 로또 번호 생성

- ▶ 이름은 항상 같으므로 이름과 오늘의 날짜를 조합하여 매일 다른 로또 번호 생성
- ▶ 이름 문자열의 해시 값을 받는 것(name.hashCode())처럼 오늘 날짜의 년월일 조합과 이름을 합쳐서 해시 값을 추출 → targetString.hashCode()

```
/**
 * 입력받은 이름에 대한 해시코드를 사용하여 로또 번호를 섞고 결과를 반환한다.
 */
fun getLottoNumbersFromHash(name: String): MutableList<Int> {
    // 1 ~ 45 번에 로또 번호를 저장할 리스트 생성
    val list = mutableListOf<Int>()

    // 1~45 까지 for 문을 돌면서 리스트에 로또 번호 저장
    for (number in 1..45) {
        list.add(number)
    }
    // SimpleDateFormat 은 날짜의 시간값을 포맷화된 텍스트 형태로 바꿔주는 클래스
    // 현재 Date 의 "yyyy-MM-dd" 문자열과 이름 문자열을 합친다 → 2019-04-26홍길동.hashCode()
    val targetString = SimpleDateFormat("yyyy-MM-dd", Locale.KOREA).format(Date()) + name
    //시간 마다 다른 번호로 하려면 HH 포맷 추가
    // 리스트를 무작위로 섞는다. 이때 섞는 기준으로 Random(SEED) 를 사용한다
    // SEED 값은 전달받은 이름과 오늘의 해당하는 "yyyy-MM-dd" 를 합친 문자열의 해시코드를 사용한다.
    list.shuffle(Random(targetString.hashCode().toLong()))

    // 리스트를 앞에서부터 순서대로 6개를 잘라 결과 반환
    return list.subList(0, 6)
}
```

로또 번호 생성 앱 만들기

▶ 이름과 날짜의 결과를 받아서 출력하는 코드 추가

▶ ResultActivity.kt

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_result)

    // 전달받은 결과 배열을 가져온다.
    val result = intent.getIntegerArrayListExtra("result")

    // 전달받은 이름을 가져온다.
    val name = intent.getStringExtra("name")

    // 결과화면 기본 텍스트
    resultLabel.text = "랜덤으로 생성된\n로또번호입니다"

    // name 이 전달된 경우 결과화면의 텍스트를 변경
    if(!TextUtils.isEmpty(name)){
        resultLabel.text = "${name} 님의\n${SimpleDateFormat("yyyy년 MM월 dd일").format(Date())}\n
로또 번호입니다"
    }
    // 전달받은 결과가 있는 경우에만 실행
    result?.let {
        // 결과에 맞게 로또 공 이미지를 업데이트한다.
        // 전달받은 결과는 정렬되어 있지않으므로 정렬해서 전달한다.
        updateLottoBallImage(result.sortedBy { it })
    }
}
```


로또 번호 생성 앱 만들기

▶ 이름이 입력되지 않으면 결과를 확인할 수 없도록 코드 수정

▶ NameActivity.kt의 onCreate()에 리스너 등록

```
goButton.setOnClickListener {  
    // 입력된 이름이 없으면 토스트 메시지 출력후 리턴  
    if(TextUtils.isEmpty(editText.text.toString())) {  
        Toast.makeText(applicationContext, "이름을 입력하세요.", Toast.LENGTH_SHORT).show()  
        return@setOnClickListener  
    }  
  
    // ResultActivity 를 시작하는 Intent 생성  
    val intent = Intent(this, ResultActivity::class.java)  
  
    // intent 의 결과 데이터를 전달한다.  
    // int 의 리스트를 전달하므로 putIntegerArrayListExtra 를 사용한다.  
    // 전달하는 리스트는 이름의 해시코드로 생성한 로또번호  
    intent.putIntegerArrayListExtra("result",  
        ArrayList(getLottoNumbersFromHash(editText.text.toString())))  
  
    // 입력받은 이름을 추가로 전달한다.  
    intent.putExtra("name", editText.text.toString())  
  
    // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행  
    startActivity(intent)  
}
```

로또 번호 생성 앱 만들기

▶ 이름과 로또번호를 전달받아서 결과화면에 출력

▶ ResultActivity.kt

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_result)

    // 전달받은 결과 배열을 가져온다.
    val result = intent.getIntegerArrayListExtra("result")

    // 전달받은 이름을 가져온다.
    val name = intent.getStringExtra("name")

    // 결과화면 기본 텍스트
    resultLabel.text = "랜덤으로 생성된\n로또번호입니다"

    // name 이 전달된 경우 결과화면의 텍스트를 변경
    if(!TextUtils.isEmpty(name)){
        resultLabel.text = "${name} 님의\n${SimpleDateFormat("yyyy년 MM월 dd일").format(Date())}\n
로또 번호입니다"
    }

    // 전달받은 결과가 있는 경우에만 실행
    result?.let {
        // 결과에 맞게 로또 공 이미지를 업데이트한다.
        // 전달받은 결과는 정렬되어 있지않으므로 정렬해서 전달한다.
        updateLottoBallImage(result.sortedBy { it })
    }
}
```

로또 번호 생성 앱 만들기

▶ 별자리에 따른 로또 번호 생성

- ▶ 앞서 이름으로 입력받은 문자열과 날짜 조합에 대한 해시 코드 값을 랜덤 함수의 seed값으로 사용하고 결과로 로또 번호를 생성하였음
- ▶ 별자리에 따른 로또 번호도 날짜와 별자리의 조합으로 추출

▶ ConstellationActivity.kt

//전달받은 월정보, 일정보 기준으로 별자리를 반환한다.

```
fun makeConstellationString(month: Int, day: Int): String {  
    // 전달받은 월 정보와 일 정보를 기반으로 정수형태의 값을 생성 ex) 1월 5일 -> 105, 11월 1일 -> 1101  
    val target = "${month + 1}${String.format("%02d", day)}".toInt()  
    when (target) {  
        in 101..119 -> return "염소자리"  
        in 120..218 -> return "물병자리"  
        in 219..320 -> return "물고기자리"  
        in 321..419 -> return "양자리"  
        in 420..520 -> return "황소자리"  
        in 521..621 -> return "쌍둥이자리"  
        in 622..722 -> return "게자리"  
        in 723..822 -> return "사자자리"  
        in 823..923 -> return "처녀자리"  
        in 924..1022 -> return "천칭자리"  
        in 1023..1122 -> return "전갈자리"  
        in 1123..1224 -> return "사수자리"  
        in 1225..1231 -> return "염소자리"  
        else -> return "기타별자리"  
    }  
}
```

로또 번호 생성 앱 만들기

▶ 별자리에 따른 로또 번호 생성

▶ 생년월일을 입력하는 뷰인 DatePicker의 날짜가 변하면 별자리를 표시하는 텍스트를 변경

▶ ConstellationActivity.kt

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_constellation)  
  
    // 로또번호 확인 버튼의 클릭이벤트 리스너 설정  
    goResultButton.setOnClickListener {  
  
        // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행  
        val intent = Intent(this, ResultActivity::class.java)  
        startActivity(intent)  
    }  
  
    // 현재 DatePicker 의 월, 일 정보로 별자리 텍스트 변경  
    textView.text = makeConstellationString(datePicker.month, datePicker.dayOfMonth)  
  
    //뒤에서 계속
```

로또 번호 생성 앱 만들기

▶ 별자리에 따른 로또 번호 생성

▶ 생년월일을 입력하는 뷰인 DatePicker의 날짜가 변하면 별자리를 표시하는 텍스트를 변경

▶ ConstellationActivity.kt

//앞에서 이어서

```
// DatePicker 의 날짜가 변화하면 별자리를 보여주는 텍스트뷰도 변경
val calendar = Calendar.getInstance()
datePicker.init(calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH), object : CalendarView.OnDateChangeListener,
DatePicker.OnDateChangeListener {
    override fun onChanged(view: DatePicker?, year: Int, monthOfYear: Int, dayOfMonth:
Int) {
        // 변경된 시점의 DatePicker 의 월, 일 정보로 별자리 텍스트 변경
        textView.text = makeConstellationString(datePicker.month, datePicker.dayOfMonth)
    }

    override fun onSelectedDayChange(view: CalendarView?, year: Int, month: Int, dayOfMonth:
Int) {
    }
})
})
```

로또 번호 생성 앱 만들기

▶ 별자리에 따른 로또 번호 생성

▶ OnDataChangedListener에서 바뀐 날짜의 별자리로 텍스트를 변경

▶ 이후에는 이름으로 로또번호를 생성하는 방법이 비슷

▶ 이름대신 별자리를 사용

▶ NameActivity에 있는 로또 생성 함수를 ConstellationActivity에 똑같이 사용하는 것이 효율적

//입력받은 별자리에 대한 해시코드를 사용하여 로또 번호를 섞고 결과를 반환한다.

```
fun getLottoNumbersFromHash(str: String): MutableList<Int> {  
    // 1 ~ 45 번에 로또 번호를 저장할 리스트 생성  
    val list = mutableListOf<Int>()  
  
    // 1~45 까지 for 문을 돌면서 리스트에 로또 번호 저장  
    for (number in 1..45) {  
        list.add(number)  
    }  
    // SimpleDateFormat 은 날짜의 시간값을 포맷화된 텍스트 형태로 바꿔주는 클래스  
    // 현재 Date 의 "yyyy-MM-dd" 문자열과 이름 문자열을 합친다 → 2019-04-26-홍길동.hashCode()  
    val targetString = SimpleDateFormat("yyyy-MM-dd", Locale.KOREA).format(Date()) + str  
    //시간마다 다른 번호로 하려면 HH 포맷 추가  
    // 리스트를 무작위로 섞는다. 이때 섞는 기준으로 Random(SEED) 를 사용한다  
    // SEED 값은 전달받은 이름과 오늘의 해당하는 "yyyy-MM-dd" 를 합친 문자열의 해시코드를 사용한다.  
    list.shuffle(Random(targetString.hashCode().toLong()))  
  
    // 리스트를 앞에서부터 순서대로 6개를 잘라 결과 반환  
    return list.subList(0, 6)  
}
```

로또 번호 생성 앱 만들기

▶ 별자리에 따른 로또 번호 생성

- ▶ ConstellationActivity의 onCreate()의 버튼 리스너에 로또번호를 결과화면으로 전달하고 전환되는 코드 추가

```
// 로또번호 확인 버튼의 클릭이벤트 리스너 설정
goResultButton.setOnClickListener {
    // ResultActivity 를 시작하는 Intent 생성
    val intent = Intent(this, ResultActivity::class.java)

    // intent 의 결과 데이터를 전달한다.
    // int 의 리스트를 전달하므로 putIntegerArrayListExtra 를 사용한다.
    // 전달하는 리스트는 별자리의 해시코드로 생성한 로또번호
    intent.putIntegerArrayListExtra("result",
    ArrayList(getLottoNumbersFromHash(makeConstellationString(datePicker.month, datePicker.dayOfMonth))))

    // 별자리를 추가로 전달한다.
    intent.putExtra("constellation", makeConstellationString(datePicker.month,
    datePicker.dayOfMonth))

    // ResultActivity 를 시작하는 Intent 를 만들고 startActivity 로 실행
    startActivity(intent)
}
```

로또 번호 생성 앱 만들기

▶ 별자리에 따른 로또 번호 생성

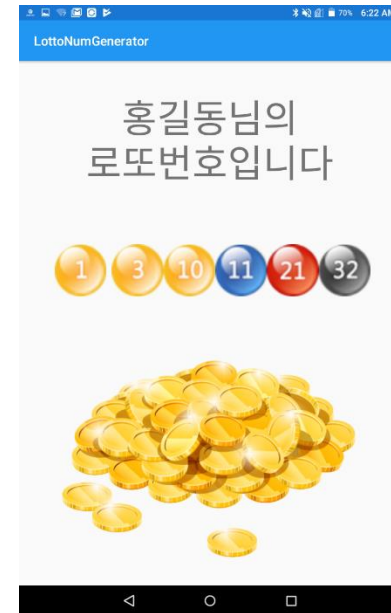
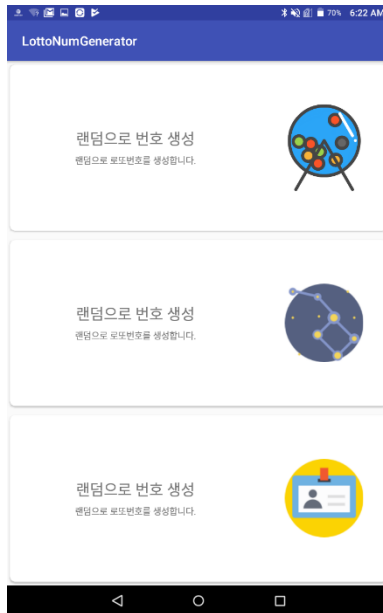
▶ 별자리 정보와 로또번호를 ResultActivity에서 출력

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_result)
    // 전달받은 결과 배열을 가져온다.
    val result = intent.getIntegerArrayListExtra("result")
    // 전달받은 이름을 가져온다.
    val name = intent.getStringExtra("name")
    // 전달받은 별자리를 가져온다
    val constellation = intent.getStringExtra("constellation")
    // 결과화면 기본 텍스트
    resultLabel.text = "랜덤으로 생성된\n로또번호입니다"
    // name 이 전달된 경우 결과화면의 텍스트를 변경
    if(!TextUtils.isEmpty(name)){
        resultLabel.text = "${name} 님의\n${SimpleDateFormat("yyyy년 MM월 dd일").format(Date())}\n
로또 번호입니다"
    }
    // 별자리가 전달된 경우 텍스트 변경
    if(!TextUtils.isEmpty(constellation)){
        resultLabel.text = "${constellation} 의\n${SimpleDateFormat("yyyy년 MM월 dd일
").format(Date())}\n로또 번호입니다"
    }
    // 전달받은 결과가 있는 경우에만 실행
    result?.let {
        // 결과에 맞게 로또 공 이미지를 업데이트한다.
        // 전달받은 결과는 정렬되어 있지않으므로 정렬해서 전달한다.
        updateLottoBallImage(result.sortedBy { it })
    }
}
```


로또 번호 생성 앱 만들기

▶ 별자리에 따른 로또 번호 생성

▶ 실행 확인



로또 번호 생성 앱 만들기

▶ 중복 코드 제거

▶ NameActivity와 ConstellationActivity에 로또 번호를 생성하는 코드가 동일한 형식으로 존재

```
/**
 * 랜덤으로 추출하여 6개의 로또번호를 만드는 함수
 */
fun getRandomLottoNumbers(): MutableList<Int>

/**
 * 랜덤으로 1 ~ 45 번호중 하나의 번호를 생성하는 함수
 */
fun getRandomLottoNumber(): Int

/**
 * Shuffle 을 사용해 로또 번호 생성
 */
fun getShuffleLottoNumbers(): MutableList<Int>

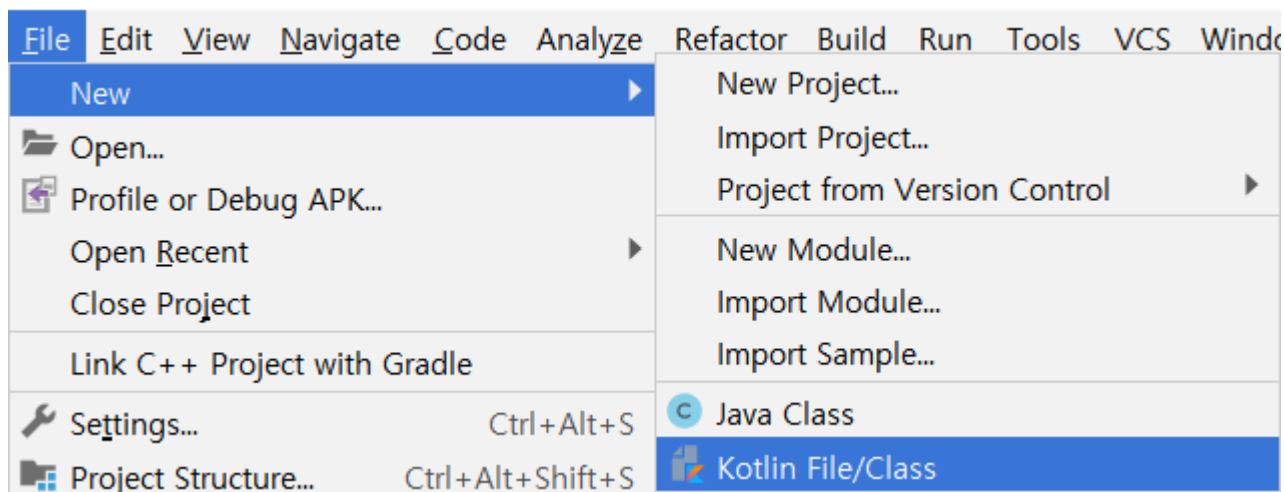
/**
 * 입력받은 이름에 대한 해시코드를 사용하여 로또 번호를 섞고 결과를 반환한다.
 */
fun getLottoNumbersFromHash(str: String): MutableList<Int>
```

로또 번호 생성 앱 만들기

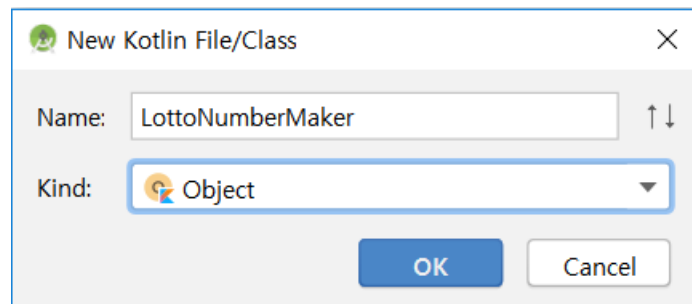
▶ 중복 코드 제거

▶ 로또 번호 생성 코드를 LottoNumberMaker 클래스에 작성

▶ 코틀린 클래스 생성



▶ 따로 선언하지 않고 바로 사용할 수 있도록 object 타입으로 생성



로또 번호 생성 앱 만들기

▶ LottoNumberMaker

▶ 중복되는 코드를 제거하고 재사용성을 높임

- ▷ 번호 추출 방식을 변경해야 한다면 2개의 클래스 내부의 코드를 모두 변경해야 하는 번거로움

▶ 기능

- ▷ 문자열을 입력받고 로또 번호를 생성하는 함수

- ▷ 랜덤, 해시코드 사용 등, 로또 번호를 사용하는 모든 기능을 해당 클래스에 구현

- ▷ `import java.text.SimpleDateFormat`

- ▷ `import java.util.*`

로또 번호 생성 앱 만들기

▶ LottoNumberMaker 클래스 내부에 구현

▶ 랜덤으로 추출하여 6개의 로또번호를 만드는 함수

```
/**
 * 랜덤으로 추출하여 6개의 로또번호를 만드는 함수
 */
fun getRandomLottoNumbers(): MutableList<Int> {
    // 무작위로 생성된 로또 번호를 저장할 가변 리스트 생성
    val lottoNumbers = mutableListOf<Int>()

    // 6번 반복하는 for 문
    for (i in 1..6) {
        // 랜덤한 번호를 임시로 저장할 변수를 생성
        var number = 0
        do {
            // 랜덤한 번호를 추출해 number 변수에 저장
            number = getRandomLottoNumber()

            // lottoNumbers 에 number 변수의 값이 없을때까지 반복
        } while (lottoNumbers.contains(number))

        // 이미 뽑은 리스트에 없는 번호가 나올때까지 반복했으므로 중복이 없는 상태
        // 추출된 번호를 뽑은 리스트에 추가
        lottoNumbers.add(number)
    }
    return lottoNumbers
}
```

로또 번호 생성 앱 만들기

▶ LottoNumberMaker 클래스 내부에 구현

▶ 랜덤으로 1 ~ 45 번호중 하나의 번호를 생성하는 함수

```
/**
 * 랜덤으로 1 ~ 45 번호중 하나의 번호를 생성하는 함수
 */
fun getRandomLottoNumber(): Int {
    // Random.nextInt 는 0 ~ 전달받은 파라미터 값 미만의 번호를 생성
    // ex) Random().nextInt(10) 은 0 ~ 9 까지의 무작위 수를 반환
    // 1 ~ 45 까지의 번호를 생성하려면 파라미터의 45 를 넣고 결과값의 1을 더한다.
    return Random().nextInt(45) + 1
}
```

로또 번호 생성 앱 만들기

▶ LottoNumberMaker 클래스 내부에 구현

▶ Shuffle 을 사용해 로또 번호 생성

```
/**
 * Shuffle 을 사용해 로또 번호 생성
 */
fun getShuffleLottoNumbers(): MutableList<Int> {
    // 1 ~ 45 번에 로또 번호를 저장할 리스트 생성
    val list = mutableListOf<Int>()

    // 1~45 까지 for 문을 돌면서 리스트에 로또 번호 저장
    for(number in 1..45){
        list.add(number)
    }

    // 리스트를 무작위로 섞는다.
    list.shuffle()

    // 리스트를 앞에서부터 순서대로 6개를 잘라 결과 반환
    return list.subList(0, 6)
}
```

로또 번호 생성 앱 만들기

▶ LottoNumberMaker 클래스 내부에 구현

▶ 입력받은 이름에 대한 해시코드를 사용하여 로또 번호를 섞고 결과를 반환하는 함수

```
/**
 * 입력받은 이름에 대한 해시코드를 사용하여 로또 번호를 섞고 결과를 반환한다.
 */
fun getLottoNumbersFromHash(str: String): MutableList<Int> {
    // 1 ~ 45 번에 로또 번호를 저장할 리스트 생성
    val list = mutableListOf<Int>()

    // 1~45 까지 for 문을 돌면서 리스트에 로또 번호 저장
    for (number in 1..45) {
        list.add(number)
    }

    // SimpleDateFormat 은 날짜의 시간값을 포맷화된 텍스트 형태로 바꿔주는 클래스
    // 현재 Date 의 "yyyy-MM-dd" 문자열과 이름 문자열을 합친다
    val targetString = SimpleDateFormat("yyyy-MM-dd", Locale.KOREA).format(Date()) + str

    // 리스트를 무작위로 섞는다. 이때 섞는 기준으로 Random(SEED) 를 사용한다
    // SEED 값은 전달받은 이름과 오늘의 해당하는 "yyyy-MM-dd" 를 합친 문자열의 해시코드를 사용한다.
    list.shuffle(Random(targetString.hashCode().toLong()))

    // 리스트를 앞에서부터 순서대로 6개를 잘라 결과 반환
    return list.subList(0, 6)
}
```


로또 번호 생성 앱 만들기

▶ MainActivity 수정

- ▶ 코드 내부에서 로또 번호 생성 메소드가 필요 없으므로 제거하고 LottoNumberMaker 클래스를 이용
- ▶ Kotlin은 기본 접근 제어자가 public이므로 같은 프로젝트 내부에서는 그래도 함수를 사용할 수 있지만 명시적으로 클래스 이름을 붙여주면 이름이 중복되는 다른 메소드와 구분할 수 있음

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    randomCard.setOnClickListener {  
        val intent = Intent(this, ResultActivity::class.java)  
        intent.putIntegerArrayListExtra("result", ArrayList(LottoNumberMaker.getShuffleLottoNumbers()))  
  
        startActivity(intent)  
    }  
    constellationCard.setOnClickListener {  
        startActivity(Intent(this, ConstellationActivity::class.java))  
    }  
    nameCard.setOnClickListener {  
        startActivity(Intent(this, NameActivity::class.java))  
    }  
}
```

로또 번호 생성 앱 만들기

▶ NameActivity 수정

▶ 앞서 수정한 MainActivity 와 동일한 맥락으로 코드 수정

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_name)

    goButton.setOnClickListener {
        if(TextUtils.isEmpty(editText.text.toString())) {
            Toast.makeText(applicationContext, "이름을 입력하세요.", Toast.LENGTH_SHORT).show()
            return@setOnClickListener
        }
        val intent = Intent(this, ResultActivity::class.java)
        intent.putIntegerArrayListExtra("result",
            ArrayList(LottoNumberMaker.getLottoNumbersFromHash(editText.text.toString())))

        intent.putExtra("name", editText.text.toString())
        startActivity(intent)
    }

    backButton.setOnClickListener {
        finish()
    }
}
```

로또 번호 생성 앱 만들기

▶ ConstellationActivity 수정

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_constellation)

    goResultButton.setOnClickListener {
        val intent = Intent(this, ResultActivity::class.java)
        intent.putIntegerArrayListExtra("result",
            ArrayList(LottoNumberMaker.getLottoNumbersFromHash(
                makeConstellationString(datePicker.month, datePicker.dayOfMonth))))
        intent.putExtra("constellation", makeConstellationString(datePicker.month,
            datePicker.dayOfMonth))
        startActivity(intent)
    }
    textView.text = makeConstellationString(datePicker.month, datePicker.dayOfMonth)
    val calendar = Calendar.getInstance()
    datePicker.init(calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
        calendar.get(Calendar.DAY_OF_MONTH), object : CalendarView.OnDateChangeListener,
        DatePicker.OnDateChangedListener {
            override fun onChanged(view: DatePicker?, year: Int, monthOfYear: Int,
                dayOfMonth: Int) {
                textView.text = makeConstellationString(datePicker.month, datePicker.dayOfMonth)
            }
            override fun onSelectedDayChange(view: CalendarView?, year: Int, month: Int,
                dayOfMonth: Int) {
            }
        })
}
```

로또 번호 생성 앱 만들기

▶ ConstellationActivity 수정

▶ 별자리를 출력하는 코드는 수정할 부분이 없으며 존재의 확인만 진행

```
fun makeConstellationString(month: Int, day: Int): String {  
    val target = "${month + 1}${String.format("%02d", day)}.toInt()  
    when (target) {  
        in 101..119 -> return "염소자리"  
        in 120..218 -> return "물병자리"  
        in 219..320 -> return "물고기자리"  
        in 321..419 -> return "양자리"  
        in 420..520 -> return "황소자리"  
        in 521..621 -> return "쌍둥이자리"  
        in 622..722 -> return "게자리"  
        in 723..822 -> return "사자자리"  
        in 823..923 -> return "처녀자리"  
        in 924..1022 -> return "천칭자리"  
        in 1023..1122 -> return "전갈자리"  
        in 1123..1224 -> return "사수자리"  
        in 1225..1231 -> return "염소자리"  
        else -> return "기타별자리"  
    }  
}
```

로또 번호 생성 앱 만들기

▶ ResultActivity 수정

▶ 넘겨 받은 별자리를 화면에 표시

```
val lottoImageStartId = R.drawable.ball_01

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_result)

    val result = intent.getIntegerArrayListExtra("result")
    val name = intent.getStringExtra("name")
    val constellation = intent.getStringExtra("constellation")

    resultLabel.text = "랜덤으로 생성된\n로또번호입니다"
    if(!TextUtils.isEmpty(name)){
        resultLabel.text = "${name} 님의\n${SimpleDateFormat("yyyy년 MM월 dd일").format(Date())}\n로또
번호입니다"
    }
    if(!TextUtils.isEmpty(constellation)){
        resultLabel.text = "${constellation} 의\n${SimpleDateFormat("yyyy년 MM월 dd일
").format(Date())}\n로또 번호입니다"
    }
    result?.let {
        updateLottoBallImage(result.sortedBy { it })
    }
}
```

로또 번호 생성 앱 만들기

▶ ResultActivity 수정

- ▶ 공의 이미지를 표현하는 코드는 따로 변경할 필요가 없으며 확인만 진행

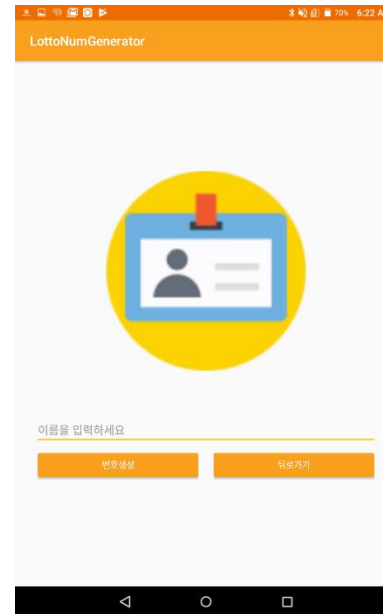
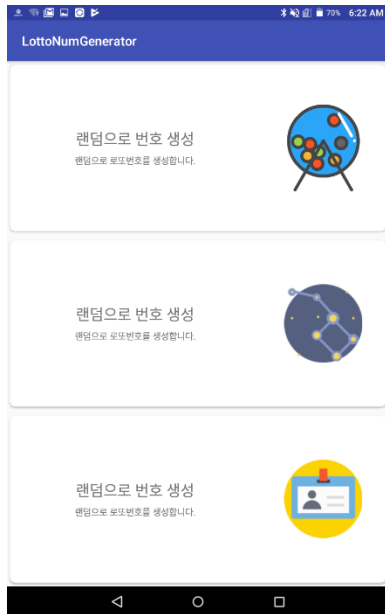
```
fun updateLottoBallImage(result: List<Int>){  
    // 결과의 사이즈가 6개 미만인경우 예러가 발생할 수 있으므로 바로 리턴한다.  
    if(result.size < 6) return  
  
    // ball_01 이미지 부터 순서대로 이미지 아이디가 있기 때문에  
    // ball_01 아이디에 결과값 -1 을 하면 목표하는 이미지가 된다  
    // ex) result[0] 이 2번 공인 경우 ball_01 에서 하나뒤에 이미지가 된다.  
    imageView01.setImageResource(lottoImageStartId + (result[0] - 1))  
    imageView02.setImageResource(lottoImageStartId + (result[1] - 1))  
    imageView03.setImageResource(lottoImageStartId + (result[2] - 1))  
    imageView04.setImageResource(lottoImageStartId + (result[3] - 1))  
    imageView05.setImageResource(lottoImageStartId + (result[4] - 1))  
    imageView06.setImageResource(lottoImageStartId + (result[5] - 1))  
}
```

로또 번호 생성 앱 만들기

▶ 모든 기능 구현 완료

▶ 랜덤, 이름으로, 별자리로...

▶ 실행하기



로또 번호 생성 앱 만들기

▶ 앱의 이름 변경

▶ 앱의 이름은 매니페스트 파일에서 관리

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
```

▶ string 리소스로서 xml에 있는 값을 참조 → strings.xml

```
<resources>
    <string name="app_name">LottoNumGenerator</string>
</resources>
```


로또 번호 생성 앱 만들기

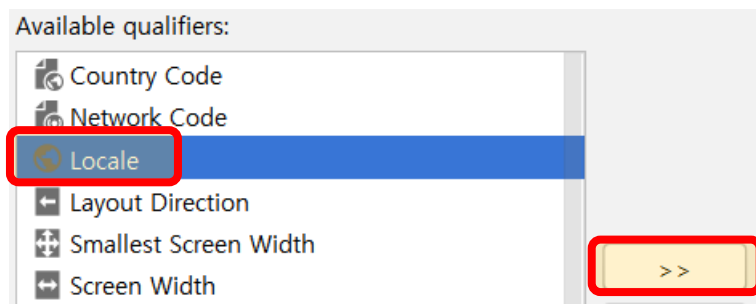
▶ 앱의 이름 변경

▶ 안드로이드는 문자열을 리소스로 관리하는 것을 권장

▶ 다국어를 쉽게 관리

▶ res의 values에서 우클릭 - [new] - [Values resource file]

▶ locale 선택 후 화살표 클릭



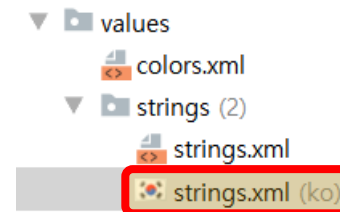
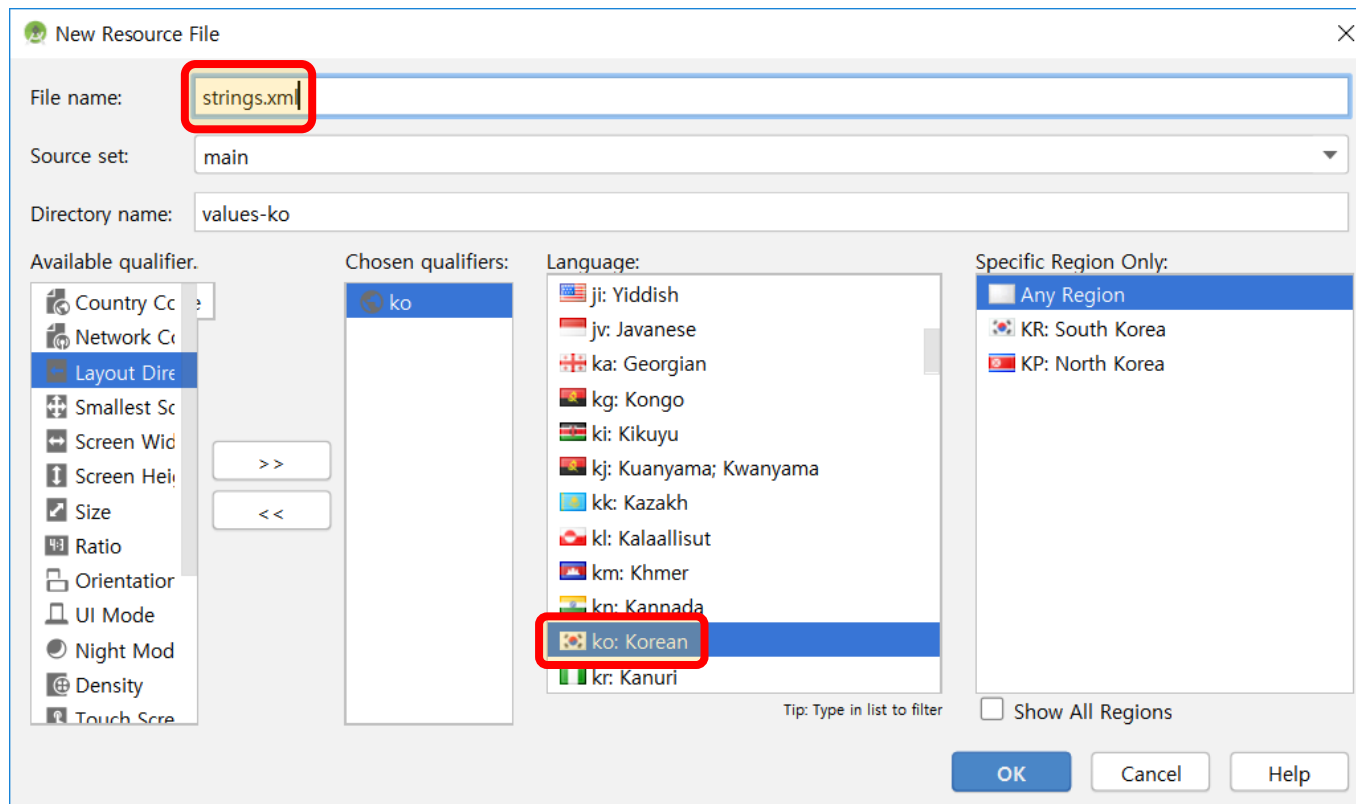
로또 번호 생성 앱 만들기

▶ 앱의 이름 변경

▶ 언어선택 화면에서 한국 선택

▶ 파일 이름은 strings.xml로 지정

▶ 같은 리소스 이름이더라도 대한민국에서는 strings.xml(ko)파일을 참조



로또 번호 생성 앱 만들기

▶ 앱의 이름 변경

▶ 앱의 한국어 이름은 한국로또 영어이름은 Korea Lotto로 지정

▷ strings.xml

```
<resources>  
    <string name="app_name">Korea Lotto</string>  
</resources>
```

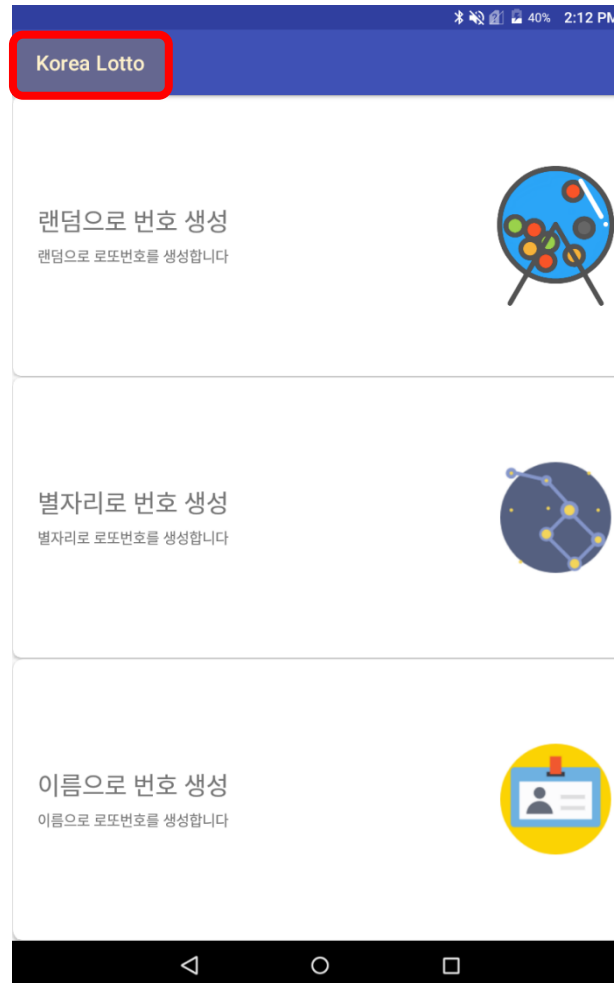
▷ strings.xml(ko)

```
<resources>  
    <string name="app_name">한국로또</string>  
</resources>
```

로또 번호 생성 앱 만들기

▶ 앱의 이름 변경

- ▶ 설정에서 선택한 Language에 따라서 앱의 이름이 다르게 표기

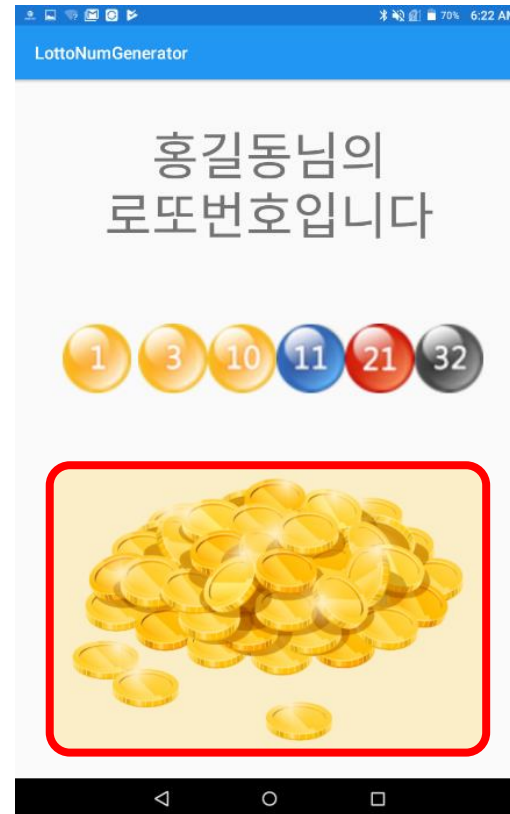
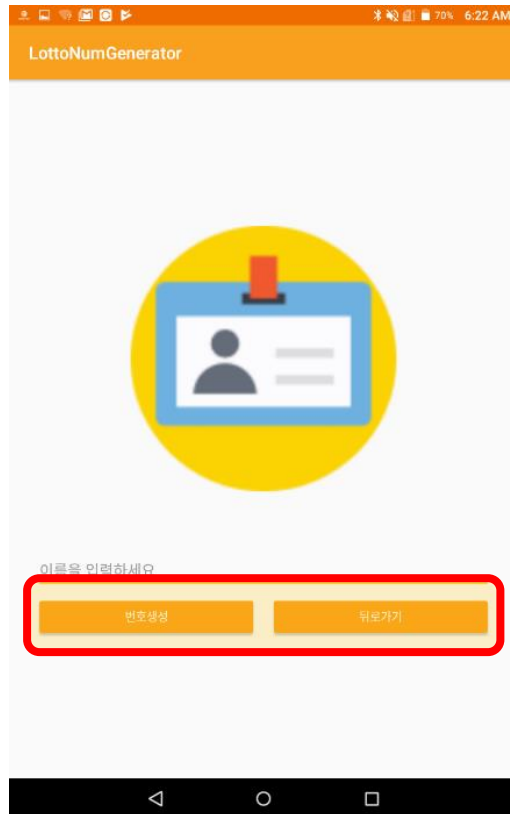


과제

▶ 아래의 요구사항을 확인하고 앱을 수정하시오.

1. 별자리 화면에서 뒤로가기 버튼을 제거한 후 번호 생성 버튼의 크기를 적절하기 확대
2. 결과 화면에서 돈 이미지를 클릭했을 때 이전화면으로 돌아가는 기능 구현

▶실로폰 프로젝트 참고



과제

▶ 아래의 요구사항을 확인하고 앱을 수정하시오.

3. anko 라이브러리를 추가한 후 Intent, toast 사용 간소화

▶비만도 계산기 프로젝트 참고

4. 로또 번호 생성시 사용하는 seed를 기존의 년월일 데이터에서 시분 데이터가 추가되도록 수정

▶yyyy-MM-dd ➔ yyyy-MM-dd-HH-mm

5. sharedPreferences를 사용하여 최근에 입력한 이름과 별자리 정보를 저장 및 자동 설정

▶비만도 계산기 프로젝트 참고

과제

▶ 아래의 요구사항을 확인하고 앱을 수정하시오.

6. 월/일 정보를 입력 받는 editText를 추가하고 입력한 날짜 데이터가 달력에 적용되도록 버튼을 추가하시오.

▶ 달력의 날짜를 변경할 경우 반대로 editText에 적용되도록 작성하시오.



Q & A
