

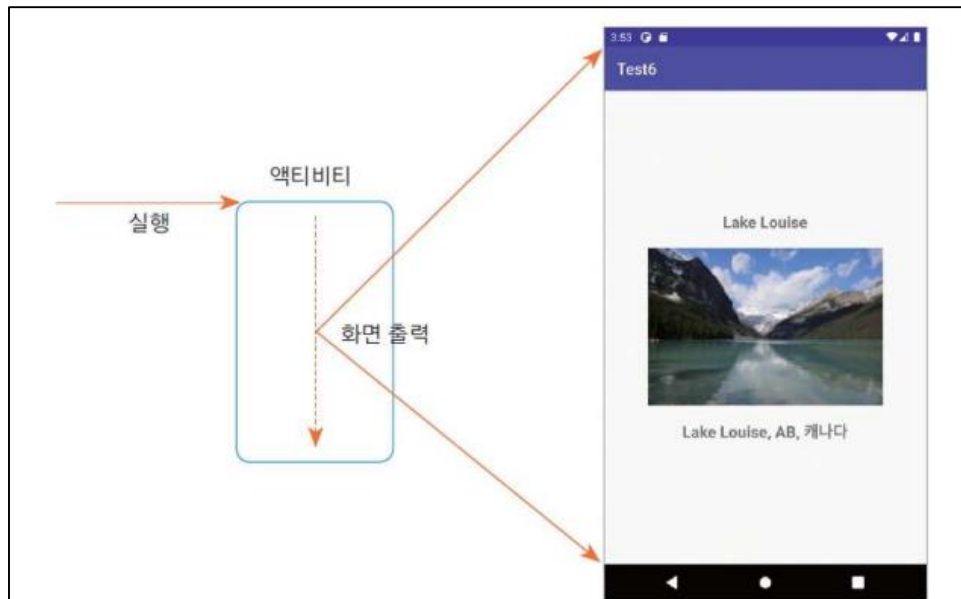
4. 뷰를 이용한 화면 구성

1. 화면을 구성하는 방법
2. 뷰 클래스
3. 기본적인 뷰 살펴보기
4. 뷰 바인딩
5. 카카오톡 비밀번호 확인 화면 만들기

1. 화면을 구성하는 방법

■ 액티비티-뷰 구조

- 화면을 출력하는 컴포넌트는 액티비티
- 화면에 내용을 표시하려면 뷰 클래스를 이용



1. 화면을 구성하는 방법

■ 레이아웃 XML로 화면 구성하기

- 뷰를 XML의 태그로 명시해 화면을 구성하는 방법

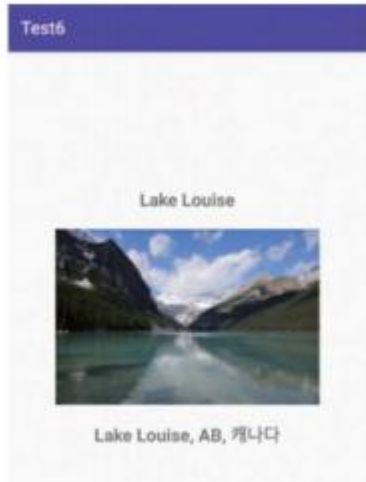
```
class MainActivity: AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        // 화면 출력 XML 명시  
        setContentView(R.layout.activity_main)  
    }  
}
```

```
<?xml version="1.0" encoding="utf-8"?>  
<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="wrap_content"  
        android:layout_height="wrap_content"  
        android:textStyle="bold"  
        android:text="Lake Louise" />  
    <ImageView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:src="@drawable/lake_1" />  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:textStyle="bold"  
        android:text="Lake Louise, AB, 캐나다" />  
</LinearLayout>
```

1. 화면을 구성하는 방법

■ 액티비티 코드로 화면 구성하기

- 화면을 구성하는 뷰 클래스를 액티비티 코드에서 직접 생성



```
class Test1Activity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        1  
        setContentView(layout)  
    }  
}
```

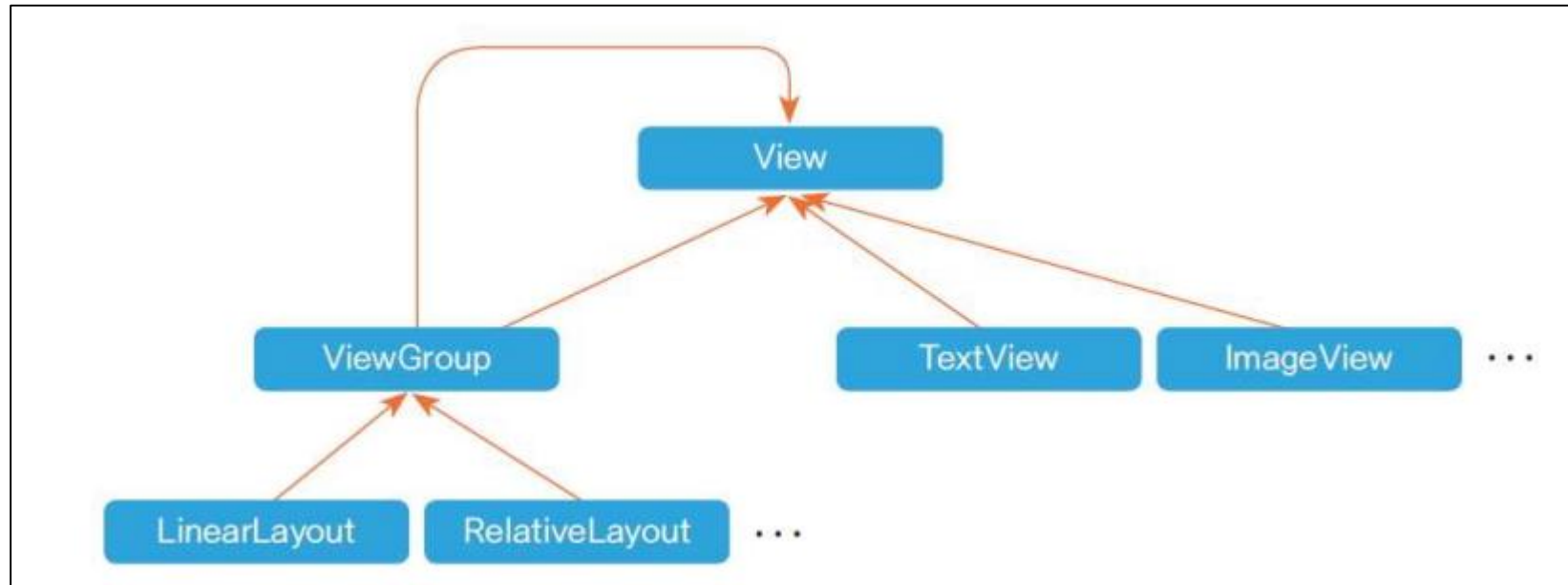
```
val name = TextView(this).apply {  
    typeface = Typeface.DEFAULT_BOLD  
    text="Lake Louise"  
}  
val image = ImageView(this).also {  
    it.setImageDrawable(ContextCompat.getDrawable(this, R.drawable.lake_1))  
}  
val address = TextView(this).apply {  
    typeface = Typeface.DEFAULT_BOLD  
    text = "Lake Louse, AB, 캐나다"  
}  
val layout = LinearLayout(this).apply {  
    orientation = LinearLayout.VERTICAL  
    gravity = Gravity.CENTER  
    addView(name, WRAP_CONTENT, WRAP_CONTENT)  
    addView(image, WRAP_CONTENT, WRAP_CONTENT)  
    addView(address, WRAP_CONTENT, WRAP_CONTENT)  
}
```

2. 뷰 클래스

■ 뷰 클래스의 기본 구조

■ 뷰 객체의 계층 구조

- View: 모든 뷰 클래스의 최상위 클래스입니다. 액티비티는 View의 서브 클래스만 화면에 출력
- ViewGroup: 자체 UI는 없이 다른 뷰 여러 개를 묶어서 제어할 목적으로 사용
- extView: 특정 UI를 출력할 목적으로 사용하는 클래스



2. 뷰 클래스

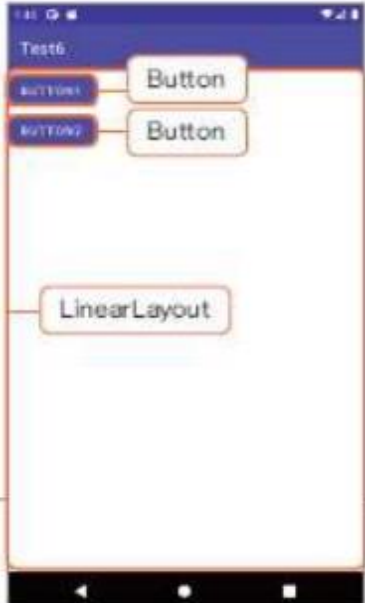
■ 레이아웃 클래스

- ViewGroup 클래스의 하위인 레이아웃 클래스는 화면 자체가 목적이 아니라 다른 뷰(TextView, ImageView 등) 객체 여러 개를 담아서 한꺼번에 제어할 목적으로 사용

• 레이아웃 클래스에 뷰 포함

```
<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>
```

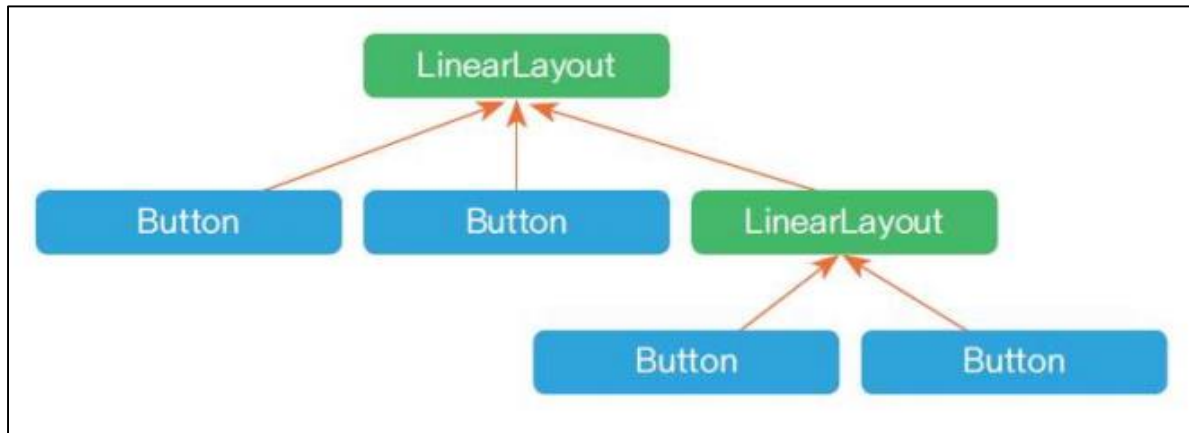
▶ 실행 결과



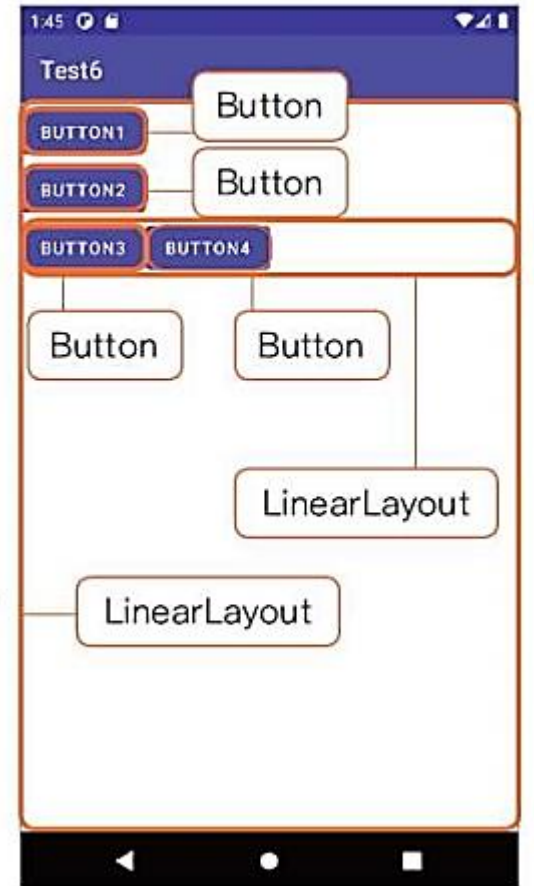
2. 뷰 클래스

■ 레이아웃 중첩

- 뷰의 계층 구조는 레이아웃 객체를 중첩해서 복잡하게 구성 가능



```
<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
        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:text="BUTTON3" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="BUTTON4" />
    </LinearLayout>
</LinearLayout>
```



2. 뷰 클래스

■ 레이아웃 XML의 뷰를 코드에서 사용하기

- 객체를 식별하기 위한 식별자 값을 지정하기 위한 속성이 id
- XML에 id 속성을 추가하면 자동으로 R.java 파일에 상수 변수로 추가
- 코드에서 findViewById() 함수를 이용해 객체 획득

• id 속성 부여

```
<TextView
    android:id="@+id/text1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="hello" />
```

• 코드에서 XML에 입력한 객체 사용법

// XML 화면 출력

setContentView(R.layout.activity_main) — 액티비티 화면 출력(뷰 객체 생성)

// id값으로 뷰 객체 획득

val textView1: TextView = findViewById(R.id.text1)

• 제네릭으로 가져온 뷰 객체

// XML 화면 출력, 뷰 객체 생성 완료

setContentView(R.layout.activity_main)

// id값으로 뷰 객체 획득

val textView1 = findViewById<TextView>(R.id.text1)

2. 뷰 클래스

■ 뷰의 크기를 지정하는 방법

- 뷰가 화면에 나올 때 어떤 크기로 보여야 하는지는 필수 정보
- 크기를 설정하는 속성은 layout_width, layout_height
 - 수치
 - match_parent
 - wrap_content

• 크기 지정 예

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#ffff00">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BUTTON1"
        android:backgroundTint="#0000ff" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="BUTTON2"
        android:backgroundTint="#ff0000" />
</LinearLayout>
```

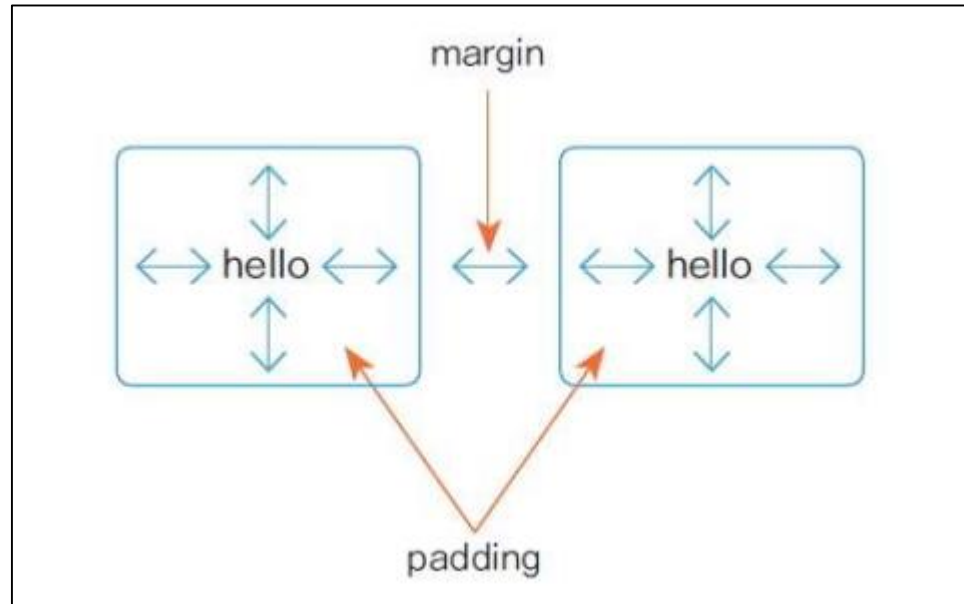
▶ 실행 결과



2. 뷰 클래스

■ 뷰의 간격 설정

- 뷰의 간격은 margin과 padding 속성으로 설정
- margin, padding 속성을 이용하면 간격이 네 방향 모두 같은 크기로 설정
- paddingLeft, paddingRight, paddingTop, paddingBottom와 layout_marginLeft, layout_marginRight, layout_marginTop, layout_marginBottom 속성을 이용 가능

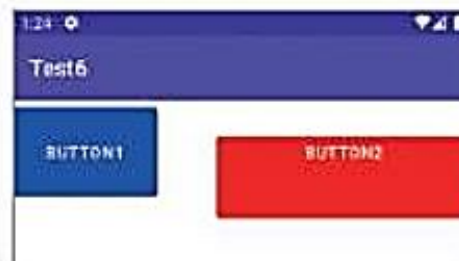


2. 뷰 클래스

■ 뷰 간격 설정

```
<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:backgroundTint="#0000ff"
        android:padding="30dp" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="BUTTON2"
        android:backgroundTint="#ff0000"
        android:paddingBottom="50dp"
        android:layout_marginLeft="50dp" />
</LinearLayout>
```

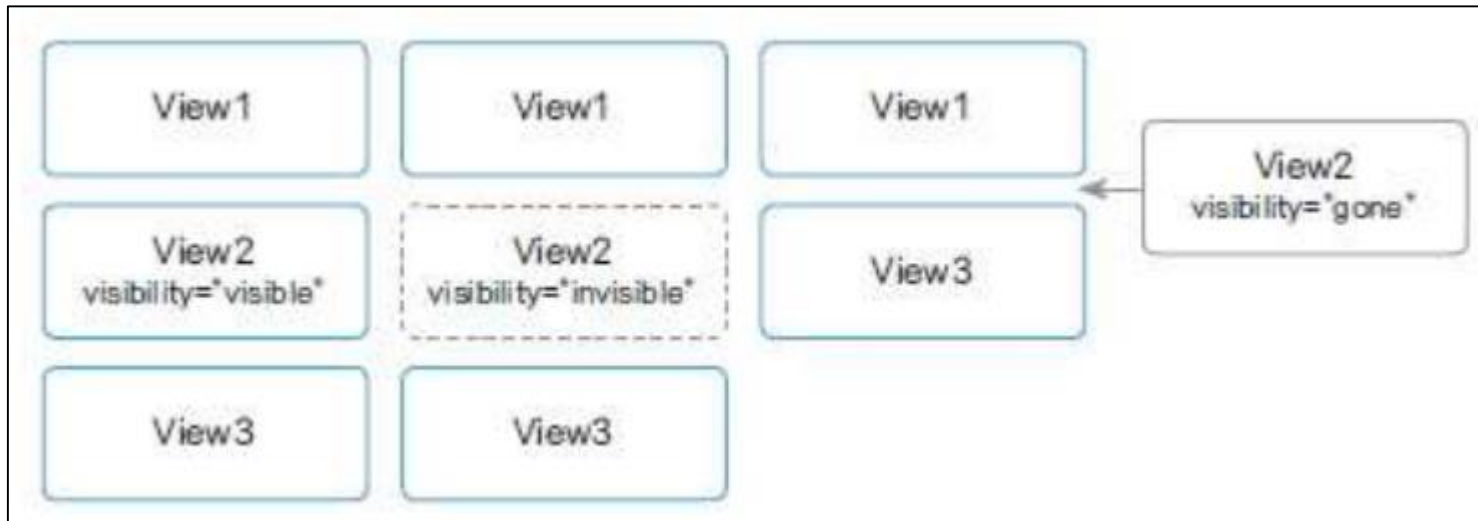
▶ 실행 결과



■ 뷰의 표시 여부 설정: visibility 속성

- visibility 속성은 뷰가 화면에 출력되어야 하는지를 설정
- visible, invisible, gone으로 설정
- invisible은 뷰가 화면에 보이지 않지만 자리는 차지
- gone으로 설정하면 자리조차 차지하지 않음

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="BUTTON2"  
    android:visibility="invisible" />
```



2. 뷰 클래스

- 뷰의 표시 여부 설정: **visibility** 속성

- 코드에서 뷰의 visibility 속성을 조정하려면 뷰의 visibility 속성값을 View.VISIBLE이나 View.INVISIBLE로 설정

• 코드에서 visibility 속성값 변경

```
visibleBtn.setOnClickListener {  
    targetView.visibility = View.VISIBLE  
}  
invisibleBtn.setOnClickListener {  
    targetView.visibility = View.INVISIBLE  
}
```

3. 기본적인 뷰 살펴보기

■ 텍스트 뷰

- TextView는 문자열을 화면에 출력하는 뷰
- android:text 속성 : TextView에 출력할 문자열을 지정
 - android:text="helloworld"
 - android:text="@string/hello"
- android:textColor 속성 : 문자열의 색상을 지정
 - android:textColor="#FF0000"
- android:textSize 속성 : 문자열의 크기를 지정
 - android:textSize="20sp"
- android:textStyle 속성 : 문자열의 스타일을 지정
 - android:textStyle="bold"
 - bold, italic, normal 중에서 선택

• 텍스트 뷰 속성

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="helloworld"
    android:textColor="#FF0000"
    android:textSize="20sp"
    android:textStyle="bold" />
```

▶ 실행 결과



3. 기본적인 뷰 살펴보기

■ 텍스트 뷰

- android:autoLink 속성 : 출력할 문자열을 분석해 특정 형태의 문자열에 자동 링크를 추가
 - android:autoLink="web"
 - web, phone, email 등을 사용

• 자동 링크 속성

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="이지스퍼블리싱 - 웹사이트 : http://easyspub.com, 전화번호 : 325-1722, 이메일 : easy@easyspub.co.kr"
    android:autoLink="web|email|phone" />
```

▶ 실행 결과



3. 기본적인 뷰 살펴보기

■ 텍스트 뷰

- android:maxLines 속성 : 문자열이 특정 줄까지만 나오도록 하는 속성
 - android:maxLines="3"
- android:ellipsize 속성 : 문자열이 더 있다는 것을 표시하기 위한 줄임표(...)를 추가
 - end, middle, start 값 지정

▶ 실행 결과



```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/long_text"
    android:singleLine="true"
    android:ellipsize="middle" />

<View
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:background="#000000" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/long_text"
    android:maxLines="3"
    android:ellipsize="end" />
```

3. 기본적인 뷰 살펴보기

■ 이미지 뷰

- 이미지를 화면에 출력하는 뷰
- android:src 속성 : 출력할 이미지를 설정
 - android:src="@drawable/image3"
- android:maxWidth, android:maxHeight, android:adjustViewBounds 속성 : 이미지의 최대 크기를 지정
 - maxWidth, maxHeight 속성은 android:adjustViewBounds 속성과 함께 사용
 - true로 설정하면 이미지의 가로세로 길이와 비례해 뷰의 크기를 맞춤

• 이미지 크기에 뷰의 크기 맞추기

```
<ImageView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:maxWidth="100dp"  
    android:maxHeight="100dp"  
    android:adjustViewBounds="true"  
    android:src="@drawable/test"  
    android:background="#0000ff"/>
```

▶ 실행 결과



3. 기본적인 뷰 살펴보기

■ 버튼, 체크박스, 라디오 버튼

- Button : 사용자 이벤트를 처리
- CheckBox : 다중 선택을 제공하는 뷰
- RadioButton : 단일 선택을 제공하는 뷰
 - 라디오 버튼 RadioGroup과 함께 사용하며 그룹으로 묶은 라디오 버튼 중 하나만 선택할 수 있게 설정

```
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="check2" />

<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="radio1" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="radio2" />
```

3. 기본적인 뷰 살펴보기

■ 에디트 텍스트

- 글을 입력할 수 있는 뷰
- android:lines, android:maxLines 속성
 - 처음부터 여러 줄 입력 크기로 나오게 하는 속성이 android:lines
 - maxLines 은 처음에는 한 줄 입력 크기로 출력되다 지정한 크기까지 늘어남
- android:inputType 속성
 - 글을 입력할 때 올라오는 키보드를 지정하는 속성
 - android:inputType="phone"

```
<EditText  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:inputType="phone" />
```



3. 기본적인 뷰 살펴보기

■ 에디트 텍스트

속성값	설명		
none	입력 유형을 지정하지 않은 상태. 모든 문자 입력 가능하며 줄바꿈 가능		
text	문자열 한 줄 입력	textVisiblePassword	textPassword와 같으며 입력한 문자 표시
textCapCharacters	대문자 입력 모드	number	숫자 입력 모드
textCapWords	각 단어의 첫 글자 입력 시 키보드가 자동으로 대문자 입력 모드	numberSigned	number와 같으며 부호 키인 마이너스(-) 입력 가능
textCapSentences	각 문단의 첫 글자 입력 시 키보드가 자동으로 대문자 입력 모드	numberDecimal	number와 같으며 소숫점 입력 가능
textMultiLine	여러 줄 입력 가능	numberPassword	숫자 키만 입력 가능. 입력한 문자는 점으로 표시
textNoSuggestions	단어 입력 시 키보드의 추천 단어를 보여 주지 않음	phone	전화번호 입력 모드
textUri	URL 입력 모드		
textEmailAddress	이메일 주소 입력 모드		
textPassword	비밀번호 입력 모드로 입력한 문자를 점으로 표시. 키보드는 영문자와 숫자, 특수 키만 표시		

4. 뷰 바인딩

■ 뷰 바인딩

- 레이아웃 XML 파일에 선언한 뷰 객체를 코드에서 쉽게 이용하는 방법
- 액티비티에서 findViewById() 함수를 이용하지 않고 레이아웃 XML 파일에 등록된 뷰 객체를 쉽게 사용할 수는 방법 제공

• 그래들 파일에 뷰 바인딩 설정

```
android {  
    (... 생략 ...)  
    viewBinding {  
        enabled = true  
    }  
}
```

4. 뷰 바인딩

■ 뷰 바인딩

- 레이아웃 XML 파일에 등록된 뷰 객체를 포함하는 클래스가 자동으로 만들어짐
- 자동으로 만들어지는 클래스의 이름은 레이아웃 XML 파일명을 따름.
- 글자를 대문자로 하고 밑줄(_)은 빼고 뒤에 오는 단어를 대문자로 만든 후 'Binding'을 추가
 - activity_main.xml → ActivityMainBinding
 - item_main.xml → ItemMainBinding
- 자동으로 만들어진 클래스의 inflate() 함수를 호출하면 바인딩 객체를 얻을 수 있음
- 액티비티 화면 출력은 setContentView() 함수에 binding.root를 전달하면 됨.

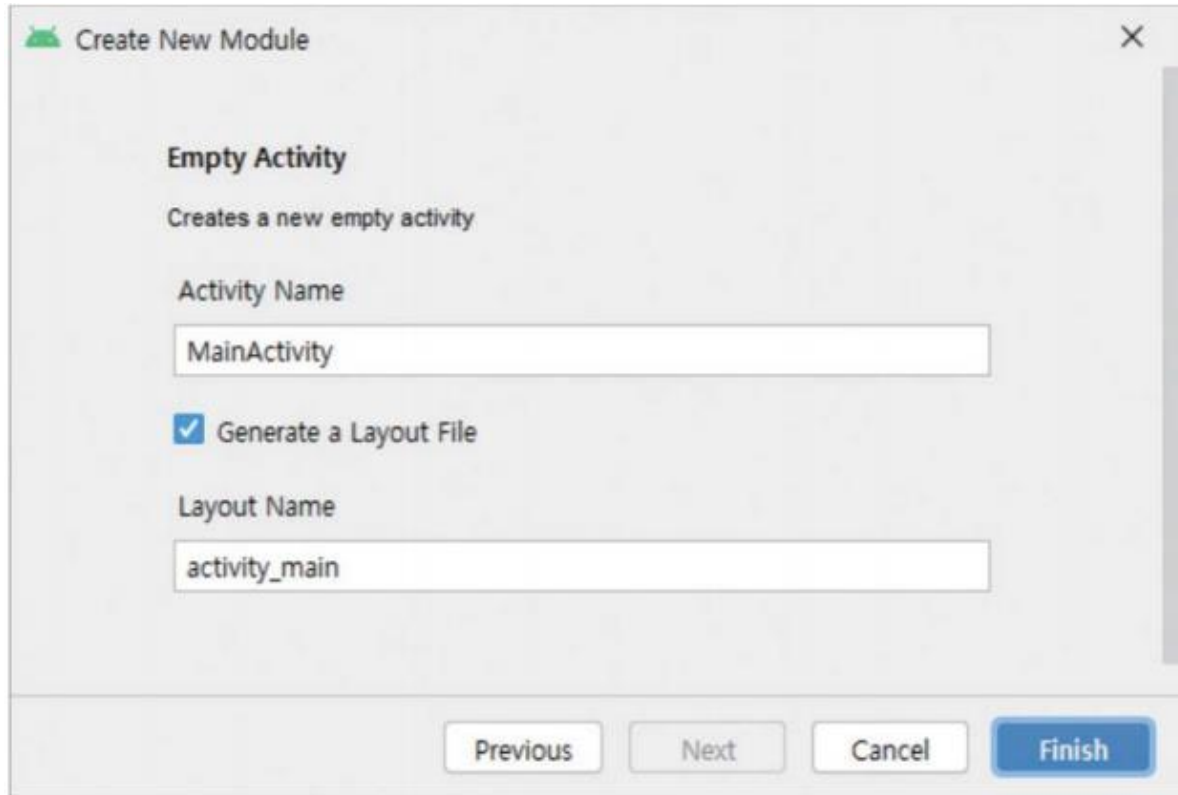
```
// 바인딩 객체 획득
val binding = ActivityMainBinding.inflate(layoutInflater)
// 액티비티 화면 출력
setContentView(binding.root)

// 뷰 객체 이용
binding.visibleBtn.setOnClickListener {
    binding.targetView.visibility = View.VISIBLE
}
```

실습 : 카카오톡 비밀번호 확인 화면 만들기

■ 1단계. 새 모듈 만들기

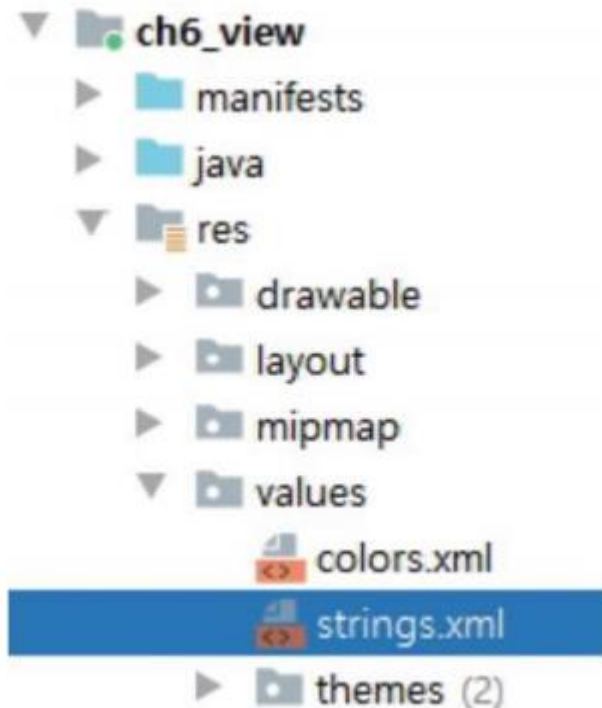
- [File → New → New Module] 메뉴
- Application/Library name 부분에Ch6_View라고입력



실습 : 카카오톡 비밀번호 확인 화면 만들기

■ 2단계. 문자열 리소스 등록하기

- res/values/strings.xml 파일



■ 3단계. 레이아웃 XML 파일 작성

- activity_main.xml 파일

■ 4단계. 앱 실행

