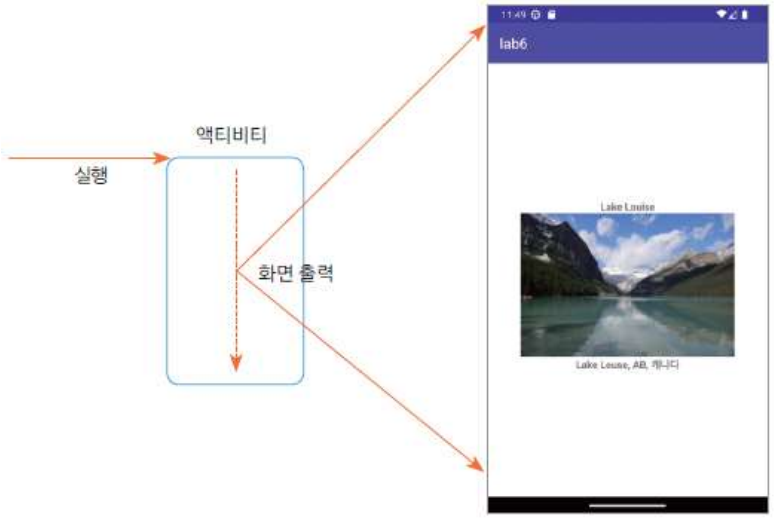
02-1. UI Architecture

UI Programming

UI 기본 구조

액티비티-뷰 구조

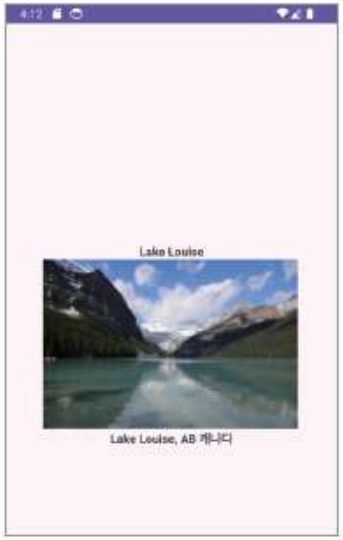
- 화면을 출력하는 컴포넌트는 액티비티
- 화면에 내용을 표시하려면 뷰 클래스를 이용



UI 기본 구조

액티비티 코드로 화면 구성하기

- 화면을 구성하는 뷰 클래스를 액티비티 코드에서 직접 생성




```
// 이름 문자열 출력 TextView 생성
val name = TextView(this).apply {
    typeface = Typeface.DEFAULT_BOLD
    text = "Lake Louise"
}
// 이미지 출력 ImageView 생성
val image = ImageView(this).also {
    it.setImageDrawable(ContextCompat.getDrawable(this, R.drawable.lake_1))
}
// 주소 문자열 출력 TextView 생성
val address = TextView(this).apply {
    typeface = Typeface.DEFAULT_BOLD
    text = "Lake Louise, AB, 캐나다"
}
val layout = LinearLayout(this).apply {
    orientation = LinearLayout.VERTICAL
    gravity = Gravity.CENTER
    // LinearLayout 객체에 TextView, ImageView, TextView 객체 추가
    addView(name, WRAP_CONTENT, WRAP_CONTENT)
    addView(image, WRAP_CONTENT, WRAP_CONTENT)
    addView(address, WRAP_CONTENT, WRAP_CONTENT)
}
// LinearLayout 객체를 화면에 출력
setContentView(layout)
```

UI 기본 구조

레이아웃 XML로 화면 구성하기

- 뷰를 XML의 태그로 명시해 화면을 구성하는 방법

• 액티비티에 레이아웃 XML 명시

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

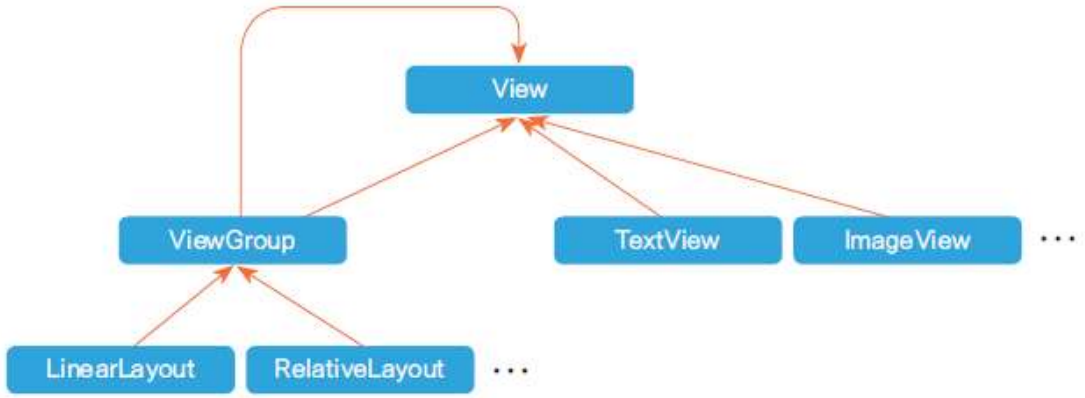
• 레이아웃 XML에서 화면 구성하기(activity_main.xml)

```
<?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>
```

UI 기본 구조

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
  - TextView: 특정 UI를 출력할 목적으로 사용하는 클래스.



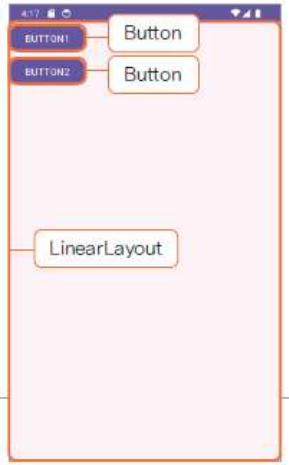
UI 기본 구조

- 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>
```

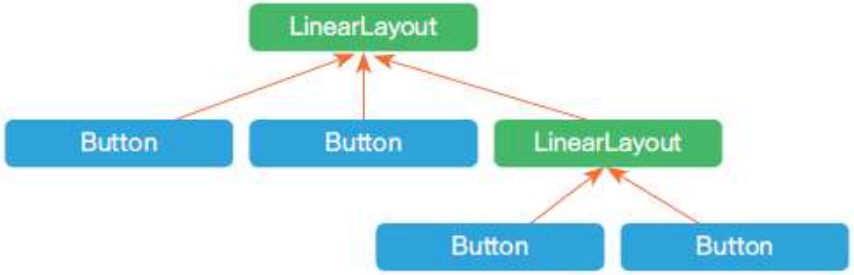
▶ 실행 결과



UI 기본 구조

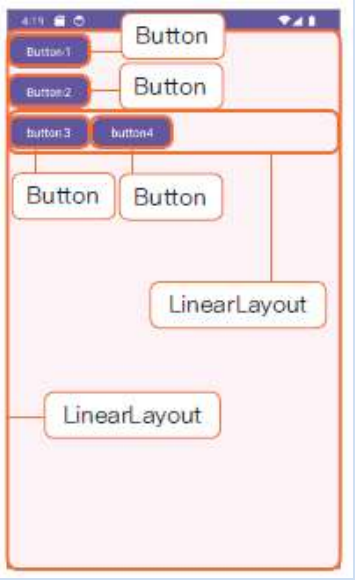
레이아웃 중첩

- 뷰의 계층 구조는 레이아웃 객체를 중첩해서 복잡하게 구성 가능



• 레이아웃 중첩

```
<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"
```



뷰 기초 속성

- id 속성
  - 뷰의 식별자 속성, 필수 속성은 아니며 필요할 때 추가
  - 지정한 id 값은 R.java 파일에 등록
  - XML에서 등록한 id 값을 매개변수로 하여 findViewById( ) 함수로 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)
// id값으로 뷰 객체 획득
val textView1: TextView = findViewById(R.id.text1)
```

액티비티 화면 출력(뷰 객체 생성)

• 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)
```

뷰 기초 속성

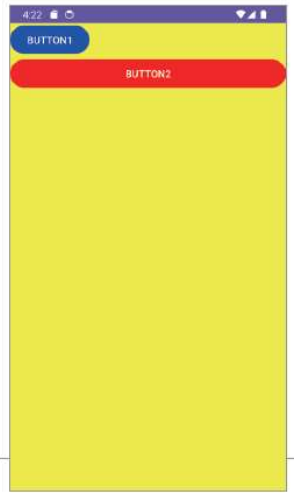
뷰의 크기를 지정하는 방법

- 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>
```

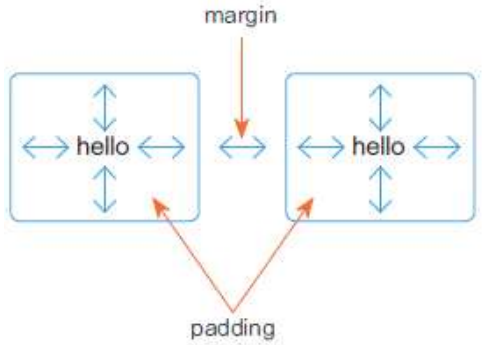
▶ 실행 결과



뷰 기초 속성

뷰의 간격 설정

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



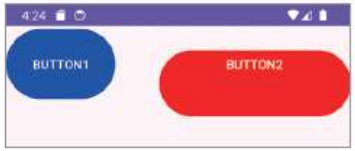
뷰 기초 속성

뷰의 간격 설정

• 뷰의 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



뷰 기초 속성

뷰의 표시 여부 설정

- 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" />
```

뷰 기초 속성

뷰의 표시 여부 설정

- 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
}
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare