

# 이벤트 처리

Mobile Software  
2019 Fall

# What to do next?

- 이벤트 처리
  - 이벤트 핸들러를 구현하는 3가지 방법
    - 내부 클래스
    - Anonymous class
    - Activity 클래스에서 직접 처리
- 이벤트 종류
  - Touch 이벤트
  - 화면 방향(orientation) 전환
- Basic widgets
  - Bitmap button
  - EditText
  - CheckBox
  - RadioButton
  - RatingBar

# 실습 준비

- 새 프로젝트 생성
  - Activity : **Empty Activity**
  - Application name : **EventHandler**
  - Minimum API level : **API 24** (Nougat)
  - Activity name : **MainActivity.kt** (자동 생성)
  - Layout name : **activity\_main.xml** (자동 생성)
- 자동 생성된 **activity\_main.xml**은 기본 layout으로 **ConstraintLayout** 이 지정되어 있음.
  - 필요할 경우 **ConstraintLayout** 대신 **LinearLayout** 등 다른 layout으로 변경
  - 자동으로 포함된 **TextView** 는 삭제

# 이벤트 처리

- **delegation** (위임) **model**

- 의미

- 발생한 이벤트를 view 객체에 전달
      - 이벤트 처리는 view 객체에게 전적으로 맡김(➔ 위임).
    - View 객체는
      - 어떤 이벤트가 발생하며,
      - 발생한 이벤트를 어느 메소드가 처리해야 하는지 알고 있음.

- 구현

- **Step 1 – View 객체가 이벤트 listener를 등록 (setOnXXXListener)**
    - **Step 2 – 해당 이벤트의 listener 를 정의**
      - listener 인터페이스를 상속받은 클래스 정의
      - 실제 이벤트를 처리하는 callback 메소드 구현

# 이벤트 핸들러 구현 방식

```
class MyClass
```

```
{
```

```
    class Listener implements OnClickListener {
```

```
        public void onClick(View v){
```

```
            ...
```

```
        }
```

```
    }
```

```
    ...
```

```
    Listener lis = new Listener();
```

```
    button.setOnClickListener(lis);
```

```
    ...
```

```
}
```

Listener 인터페이스를  
상속받은 구현 클래스

callback 메소드

Listener 객체 생성

View 객체에  
Listener 객체 등록(위임)

```
class MainActivity : AppCompatActivity() {
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_main)
```

```
        class BtnListener : View.OnClickListener {
```

```
            override fun onClick(v: View?) {
```

```
                Toast.makeText(applicationContext,
```

```
                    "버튼을 눌렀습니다",
```

```
                    Toast.LENGTH_SHORT).show()
```

```
            }
```

```
        }
```

```
        val lis = BtnListener()
```

```
        btnView.setOnClickListener(lis)
```

```
    }
```

```
}
```

# 이벤트 핸들러를 구현하는 3가지 방법

- Listener 인터페이스를 상속받는 내부 클래스 정의
  - formal method(실습1)
- **Anonymous class** (무명 클래스) : 실습2
  - 클래스 이름이 없음.
- Listener 인터페이스를 **Activity 클래스에서 직접 상속받음** : 실습3
  - 이벤트 핸들러가 Activity 클래스의 멤버 메소드

# 실습 1,2,3: Layout

```
<Button  
  android:id="@+id/btnView"  
  android:layout_width="wrap_content"  
  android:layout_height="wrap_content"  
  android:text="Press Me !"  
  android:textSize="24sp"  
  app:layout_constraintBottom_toBottomOf="parent"  
  app:layout_constraintLeft_toLeftOf="parent"  
  app:layout_constraintRight_toRightOf="parent"  
  app:layout_constraintTop_toTopOf="parent"  
  tools:textAllCaps="false" />
```

**android:onClick** 속성은 지정하지 않음

**tools:textAllCaps="false"**  
대소문자 구분을 위한 속성



# 실습 1: 내부 클래스

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        class BtnListener : View.OnClickListener {  
            override fun onClick(v: View?) {  
                Toast.makeText(applicationContext,  
                    "버튼을 눌렀습니다",  
                    Toast.LENGTH_SHORT).show()  
            }  
        }  
  
        val lis = BtnListener()  
        btnView.setOnClickListener(lis)  
    }  
}
```

클래스 상속    `class A : B()`            // B는 **class**  
구현 상속     `class A : B`             // B는 **interface**  
→ interface 는 추상 메소드(abstract method)를 갖고 있음.  
→ 구현 상속받은 클래스는 추상 메소드를 반드시 구현해야 함.



# Toast

```
Toast.makeText(applicationContext,  
    "버튼을 눌렀습니다",  
    Toast.LENGTH_SHORT).show()
```

**Toast** 는 **화면**에 메시지를 출력하고 사라짐(SnackBar와 유사)  
→ Toast.LENGTH\_SHORT, Toast.LENGTH\_LONG



화면 = *context* = Activity 컴포넌트를 구현하는 클래스



*applicationContext()* 메소드의 반환 값은 ?  
→ Activity 컴포넌트를 구현하는 클래스  
→ MainActivity → *this@MainActivity*

# 잠깐! 코드 제대로 이해하자!

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        class BtnListener : View.OnClickListener {  
            override fun onClick(v: View?) {  
                Toast.makeText(this,  
                    "버튼을 눌렀습니다",  
                    Toast.LENGTH_SHORT).show()  
            }  
        }  
    }  
}
```

*applicationContext* 대신  
*this* 로 쓰면 에러!

여기서 *this* 는 BtnListener 를 가리킴.  
BtnListener 는 UI를 구현하는 클래스가 아니다!!!

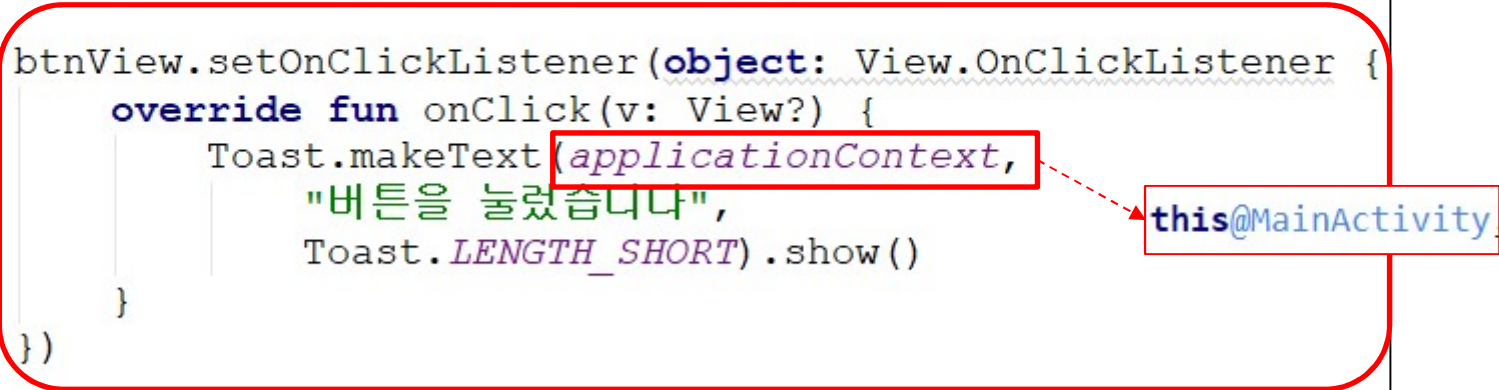
*this* 대신 **this@MainActivity** 와 같이  
UI를 구현하는 클래스 이름을 정확하게 적어야 함.

그러나, *applicationContext* 를 사용하는 것이 좋음.  
→ 항상 해당 UI 클래스를 찾아주기 때문

## 실습 2: Anonymous class

- 클래스 body는 정의하지만, 이름이 없는 클래스
  - 클래스 정의와 동시에 객체를 생성
  - 무명 클래스를 정의할 때는 **object** 키워드 사용

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        btnView.setOnClickListener(object: View.OnClickListener {  
            override fun onClick(v: View?) {  
                Toast.makeText(applicationContext,  
                    "버튼을 눌렀습니다",  
                    Toast.LENGTH_SHORT).show()  
            }  
        })  
    }  
}
```



# Kotlin에서 OnClickListener 인터페이스 구현(1/2)

```
btnView.setOnClickListener (object: View.OnClickListener {  
    override fun onClick(v: View?) {  
        Toast.makeText(applicationContext,  
            "버튼이 눌렸습니다",  
            Toast.LENGTH_SHORT).show()  
    }  
})
```

Lambda expression  
변환

```
btnView.setOnClickListener({  
    v -> Toast.makeText(applicationContext,  
        "버튼이 눌렸습니다",  
        Toast.LENGTH_SHORT).show()  
})
```

함수의 인자가 함수이므로  
괄호 밖으로 옮길 수 있다.

```
btnView.setOnClickListener(){  
    v -> Toast.makeText(applicationContext,  
        "버튼이 눌렸습니다",  
        Toast.LENGTH_SHORT).show()  
}
```

## Kotlin에서 OnClickListener 인터페이스 구현(2/2)

함수의 인자가 하나뿐이므로  
괄호를 없앨 수 있다.

```
btnView.setOnClickListener(){  
    v -> Toast.makeText(applicationContext,  
        "버튼이 눌렸습니다",  
        Toast.LENGTH_SHORT).show()  
}
```



함수의 인자를 사용하지 않  
으므로 함수의 왼쪽을  
없앨 수 있다.

```
btnView.setOnClickListener {  
    v -> Toast.makeText(applicationContext,  
        "버튼이 눌렸습니다",  
        Toast.LENGTH_SHORT).show()  
}
```



```
btnView.setOnClickListener {  
    Toast.makeText(applicationContext,  
        "버튼이 눌렸습니다",  
        Toast.LENGTH_SHORT).show()  
}
```

## 실습 2: Anonymous class (concise version)

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        btnView.setOnClickListener {  
            Toast.makeText(applicationContext,  
                "버튼이 눌렸습니다.",  
                Toast.LENGTH_SHORT).show()  
        }  
    }  
}
```

# 실습 3: Activity 클래스에서 직접 처리

```
class MainActivity : AppCompatActivity(), View.OnClickListener {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        btnView.setOnClickListener(this)  
    }  
  
    override fun onClick(v: View?) {  
        Toast.makeText(applicationContext,  
            "버튼을 눌렀습니다",  
            Toast.LENGTH_SHORT).show()  
    }  
}
```

Activity에서  
Listener  
인터페이스를  
직접 구현

Event가 발생하면 Activity  
클래스에서 직접 처리

Event handler는  
Activity의 메소드



# 잠깐! 최대한 간단하게 구현해보자!

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/button01"
    android:text="Press Me!"
    android:onClick="onClick"
    android:textAllCaps="false"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

JavaScript 에서  
사용하는 이벤트  
처리 방식

```
class MainActivity : AppCompatActivity() {
    fun onClick(v: View?) {
        Toast.makeText(this,
            "버튼이 눌러졌습니다",
            Toast.LENGTH_LONG).show();
    }

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

callback 메소드 **onClick**을  
코드에서 구현할 때

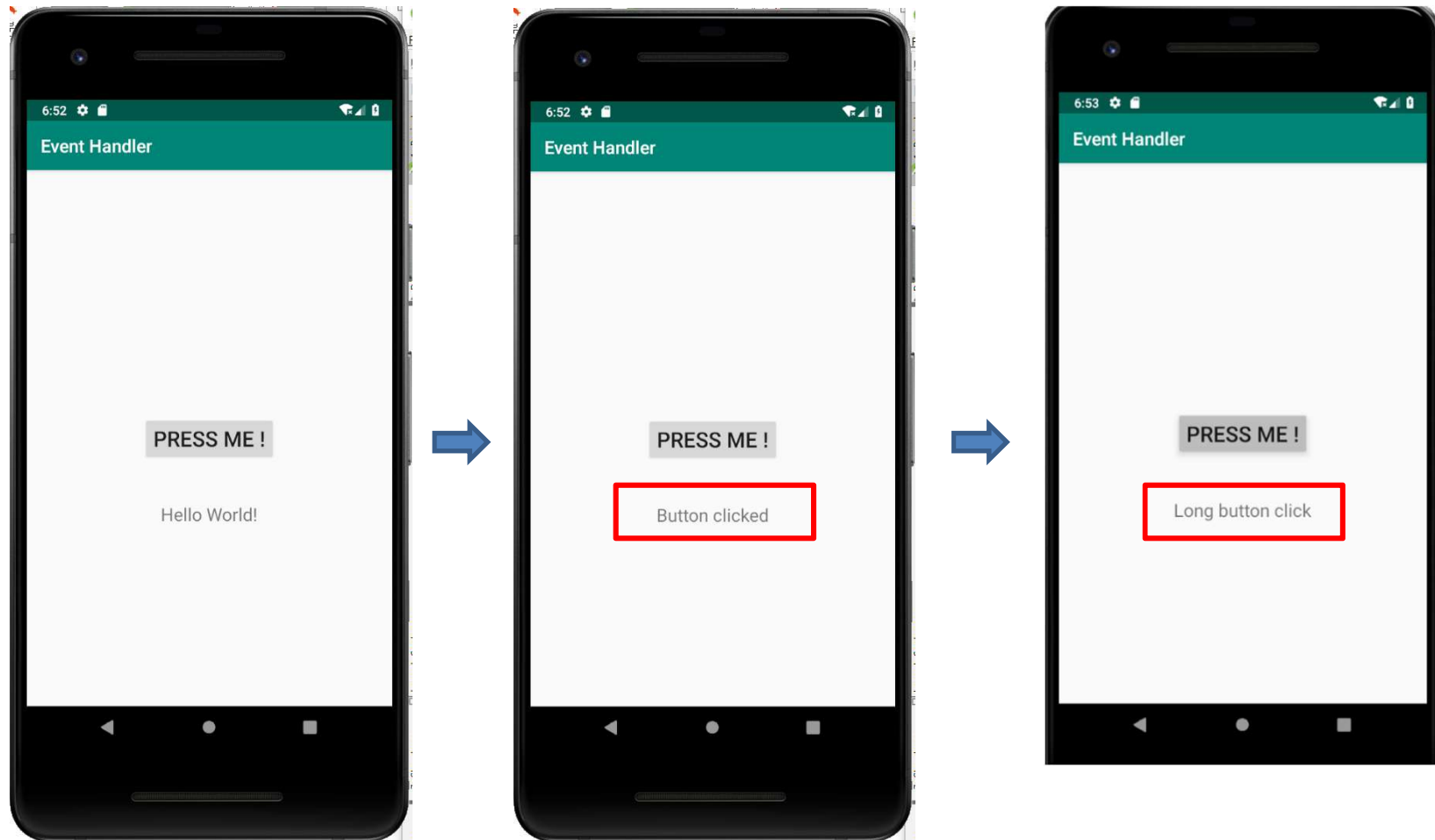
- **public** (생략 가능)
- 반환 값 없음
- parameter는 **View** 객체



# Listener 종류

리스너	콜백 메소드	설명
<u>View.OnClickListener</u>	<u>onClick()</u>	사용자가 어떤 항목을 터치하거나 내비게이션 키나 트랙볼로 항목으로 이동한 후에 엔터키를 눌러서 선택하면 호출된다.
<u>View.OnLongClickListener</u>	<u>onLongClick()</u>	사용자가 항목을 터치하여서 일정 시간 동안 그대로 누르고 있으면 발생한다.
<u>View.OnFocusChangeListener</u>	<u>onFocusChange()</u>	사용자가 하나의 항목에서 다른 항목으로 포커스를 이동할 때 호출된다.
<u>View.OnKeyListener</u>	<u>onKey()</u>	포커스를 가지고 있는 항목 위에서 키를 눌렀다가 놓았을 때 호출된다.
<u>View.OnTouchListener</u>	<u>onTouch()</u>	사용자가 터치 이벤트로 간주되는 동작을 한 경우에 호출된다.

## 실습 4: 한 개 view에 대한 2개 이벤트 처리



## 실습 4: 한 개 view에 대한 2개 이벤트 처리

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        btnView.setOnClickListener(object: View.OnClickListener {  
            override fun onClick(v: View?) {  
                statusText.text = "Button clicked"  
            }  
        })  
  
        btnView.setOnLongClickListener(object: View.OnLongClickListener {  
            override fun onLongClick(v: View?): Boolean {  
                statusText.text = "Long button click"  
                return false  
            }  
        })  
    }  
}
```

이벤트를 제대로 처리 못했으니  
등록된 다른 listener가 처리하라고 알림.

```
btnView.setOnClickListener {  
    statusText.text = "Button clicked"  
}  
  
btnView.setOnLongClickListener {  
    statusText.text = "Long button click"  
    false  
}
```

# What to do next?

- 이벤트 처리
  - 이벤트 핸들러를 구현하는 3가지 방법
    - 내부 클래스
    - Anonymous class
    - Activity 클래스에서 직접 처리
- **이벤트 종류**
  - **Touch 이벤트**
  - **화면 방향(orientation) 전환**
- Basic widgets
  - Bitmap button
  - EditText
  - CheckBox
  - RadioButton
  - RatingBar

# 이벤트 종류

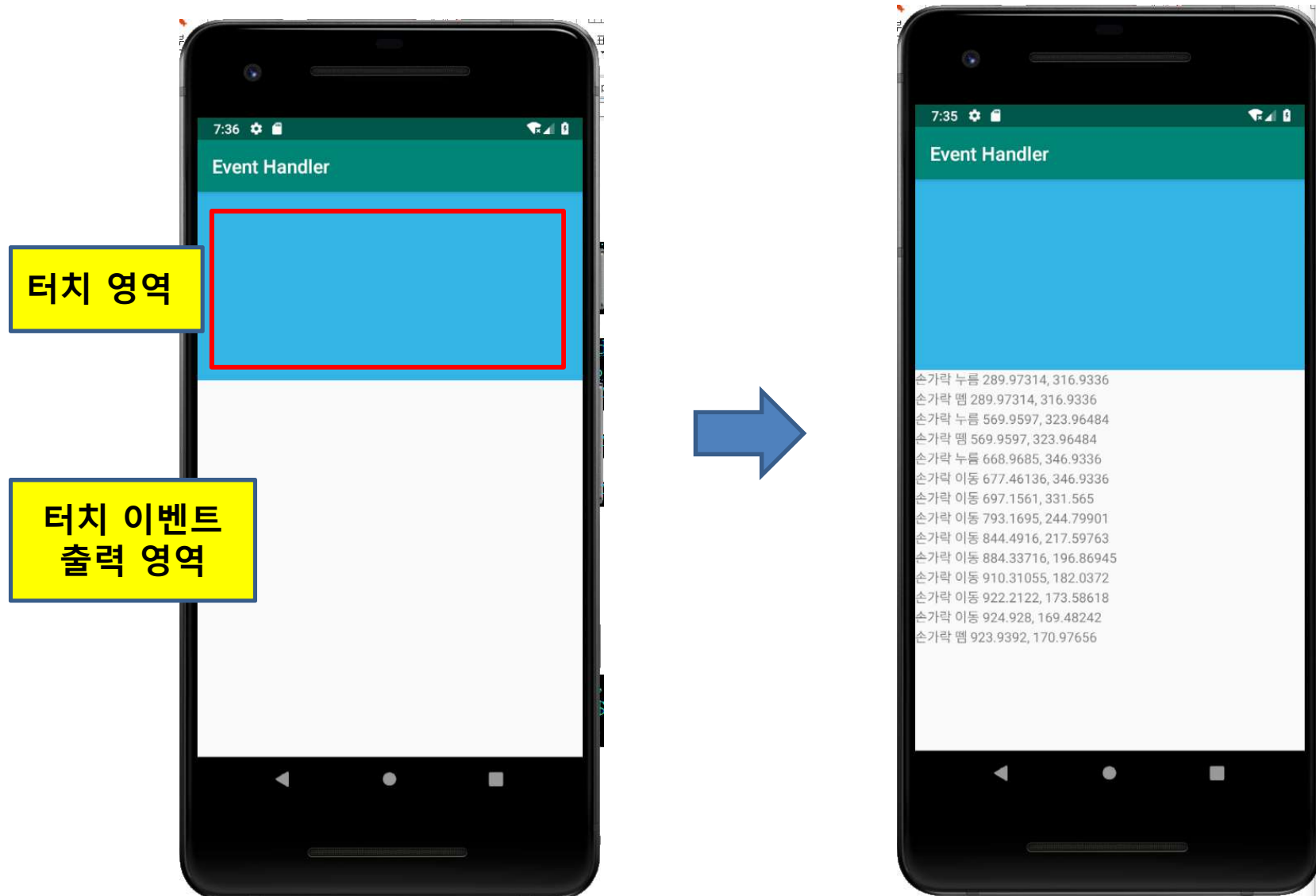
- **Touch 이벤트** : 화면을 손가락으로 touch 할 때 발생
- **Gesture 이벤트**
  - 위-아래로 scroll 할 때처럼 일정 패턴으로 구분되는 이벤트
    - Touch 이벤트 발생 → 움직임 확인 → gesture 이벤트 처리
  - callback 메소드
    - onDown, onShowPress
    - onSingleTapUp, onSingleTapConfirmed
    - onDoubleTap, onDoubleTapEvent
    - onScroll, **onFling (swipe 이벤트)**, onLongPress
- **Key 이벤트**
  - Keypad(**S/W key**)나 hardware button(**H/W key**)을 누를 때 발생
- **Focus 이벤트** : 해당 view가 입력 처리를 위해 focus를 갖음
- **화면 방향(orientation) 변경 이벤트**
  - 화면 방향이 가로(landscape)/세로(portrait)로 바뀔 때 발생

# Touch 이벤트

액션	설명
ACTION_DOWN	누르는 동작이 시작됨
ACTION_UP	누르고 있다가 떼어낼 때 발생함
ACTION_MOVE	누르는 도중에 움직임
ACTION_CANCEL	터치 동작이 취소됨
ACTION_OUTSIDE	터치가 현재의 위젯을 벗어남



# 실습 5: Touch Event



# 실습 5: Touch Event - Layout

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

    <View
        android:id="@+id/myView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@android:color/holo_blue_light"/>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2">

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"/>

    </ScrollView>
</LinearLayout>
```

Palette 창에서  
Widget > View

Palette 창에서  
Containers > ScrollView



# 실습 5: Touch Event

```
class MainActivity : AppCompatActivity() {  
    val sb = StringBuilder()  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        myView.setOnClickListener(object: View.OnClickListener {  
            override fun onTouch(v: View?, event: MotionEvent): Boolean {  
                handleTouch(event)  
                return true  
            }  
        })  
    }  
  
    private fun handleTouch(m: MotionEvent) {  
        val x = m.x  
        val y = m.y  
  
        when (m.action) {  
            MotionEvent.ACTION_DOWN -> sb.append("손가락 누름 $x, $y\n")  
            MotionEvent.ACTION_MOVE -> sb.append("손가락 이동 $x, $y\n")  
            MotionEvent.ACTION_UP -> sb.append("손가락 땀 $x, $y\n")  
            else -> sb.append("\n")  
        }  
        textView.text = sb.toString()  
    }  
}
```

이벤트 리스너 등록

Touch 이벤트  
발생 위치

Touch 이벤트 종류

소스코드 - 6~7쪽

## 실습 5-2: 소수점 둘째자리까지 출력

```
private fun handleTouch(m: MotionEvent) {  
    val x = m.x  
    val y = m.y  
  
    when (m.action) {  
        MotionEvent.ACTION_DOWN -> outMessage("손가락 누름", x, y)  
        MotionEvent.ACTION_MOVE -> outMessage("손가락 이동", x, y)  
        MotionEvent.ACTION_UP -> outMessage("손가락 땸", x, y)  
        else -> sb.append("\n")  
    }  
}  
  
private fun outMessage(msg: String, x1:Float, y1:Float) {  
    val df = DecimalFormat("#.##")  
    df.roundingMode = RoundingMode.CEILING  
    sb.append("$msg : ${df.format(x1)}, ${df.format(y1)} \n")  
    textView.text = sb.toString()  
}
```

Single touch 일 경우 m.x 사용

소수점 이하 2째 자리까지 출력

CEILING : 2.31 → 3, -1.1 → -1  
FLOOR : 2.31 → 2, -1.1 → -2

```
private fun outMessage(msg: String, x1:Float, y1:Float) {  
    sb.append("$msg : %.2f, %.2f \n".format(x1, y1))  
    textView.text = sb.toString()  
}
```

소스코드 - 7~8쪽

## 실습 5-3: safe call (1/2)

```
myView.setOnTouchListener(object: View.OnTouchListener {  
    override fun onTouch(v: View?, event: MotionEvent?): Boolean {  
        handleTouch(event)  
        return true  
    }  
})
```

event 가 null 값을 가질 수도 있는 경우

```
private fun handleTouch(m: MotionEvent?) {  
    val x:Float? = m?.x  
    val y:Float? = m?.y  
  
    when (m?.action) {  
        MotionEvent.ACTION_DOWN -> outMessage("손가락 누름", x, y)  
        MotionEvent.ACTION_MOVE -> outMessage("손가락 이동", x, y)  
        MotionEvent.ACTION_UP -> outMessage("손가락 땸", x, y)  
        else -> sb.append("\n")  
    }  
}  
  
private fun outMessage(msg: String, x1:Float?, y1:Float?) {  
    sb.append("$msg : %.2f, %.2f \n".format(x1, y1))  
    textView.text = sb.toString()  
}
```

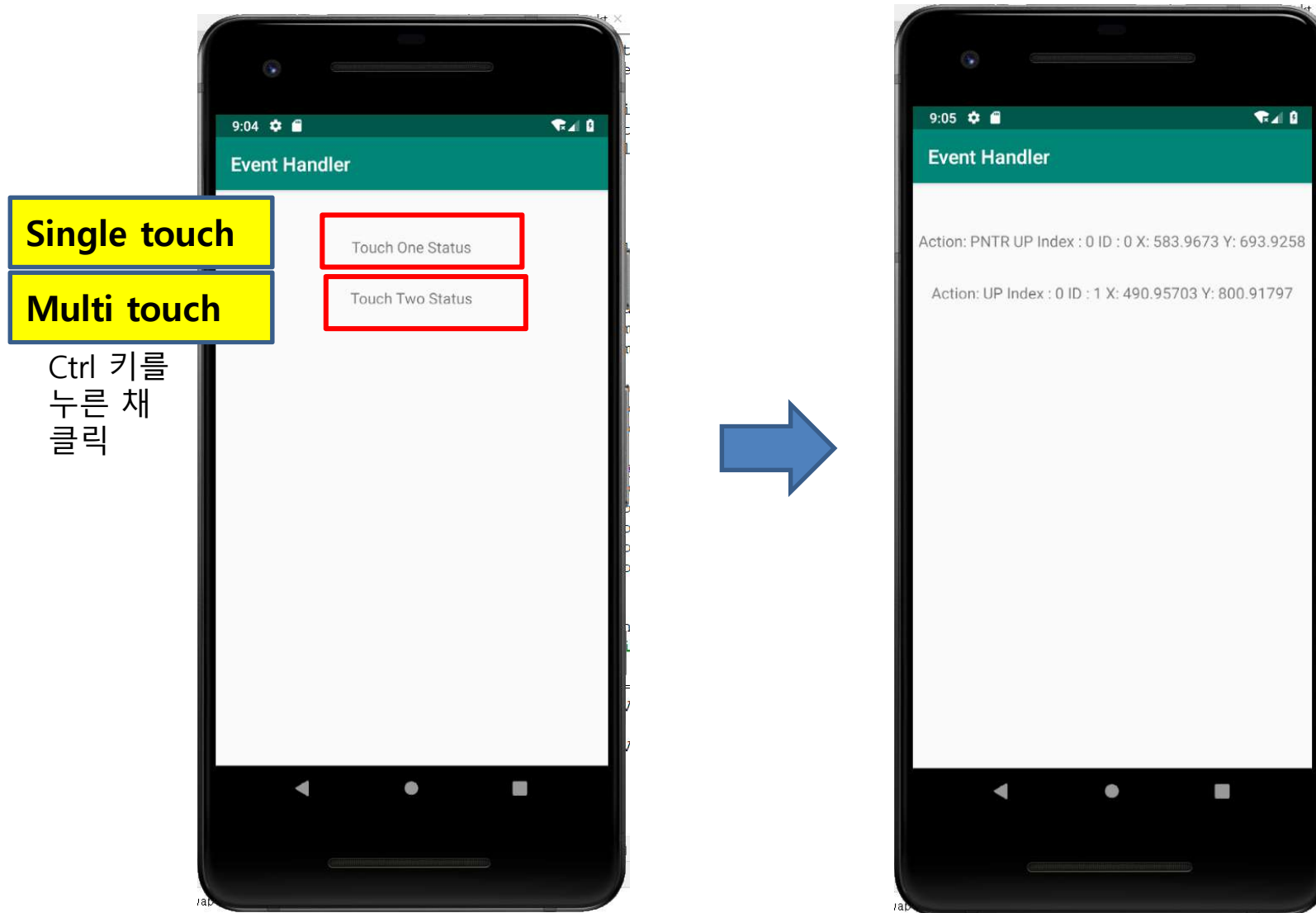
## 실습 5: safe call (2/2)

```
myView.setOnTouchListener(object: View.OnTouchListener {  
    override fun onTouch(v: View?, event: MotionEvent?): Boolean {  
        handleTouch(event)  
        return true  
    }  
}))
```

```
private fun handleTouch(m: MotionEvent?) {  
    if (m == null) return  
    val x:Float = m.x  
    val y:Float = m.y  
  
    when (m.action) {  
        MotionEvent.ACTION_DOWN -> outMessage("손가락 누름", x, y)  
        MotionEvent.ACTION_MOVE -> outMessage("손가락 이동", x, y)  
        MotionEvent.ACTION_UP -> outMessage("손가락 땜", x, y)  
        else -> sb.append("\n")  
    }  
}  
  
private fun outMessage(msg: String, x1:Float, y1:Float) {  
    sb.append("$msg : %.2f, %.2f \n".format(x1, y1))  
    textView.text = sb.toString()  
}
```

If 문을 사용해 m이 null 인 경우를 배제.

# 실습 6: Multi-Touch Event





# 실습 6: Multi-Touch Event - Layout

```
<android.support.constraint.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"
    android:id="@+id/activity_motion_event"
    tools:context=".MainActivity">
```

```
<TextView
    android:text="Touch One Status"
    android:id="@+id/textView"
    android:textSize="16sp"
    android:layout_marginTop="16dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
<TextView
    android:text="Touch Two Status"
    android:id="@+id/textView2"
    android:textSize="16sp"
    android:layout_marginTop="32dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

Touch One Status

Touch Two Status

# 실습 6: Multi-Touch Event

```
private fun handleTouch(m: MotionEvent) {  
    val pointerCount = m.pointerCount  
  
    for (i in 0 until pointerCount) {  
        val x = m.getX(i)  
        val y = m.getY(i)  
        val id = m.getPointerId(i)  
        val actionIndex = m.actionIndex  
        val actionString: String  
        when (m.action) {  
            MotionEvent.ACTION_DOWN -> actionString = "DOWN"  
            MotionEvent.ACTION_UP -> actionString = "UP"  
            MotionEvent.ACTION_POINTER_DOWN -> actionString = "PNTR DOWN"  
            MotionEvent.ACTION_POINTER_UP -> actionString = "PNTR UP"  
            MotionEvent.ACTION_MOVE -> actionString = "MOVE"  
            else -> actionString = ""  
        }  
        var message = "Action: $actionString, Index: $actionIndex, ID: $id"  
        if (id == 0) textView.text = outMessage(message, x, y)  
        else textView2.text = outMessage(message, x, y)  
    }  
}  
  
private fun outMessage(msg: String, x1: Float, y1: Float) =  
    "$msg : %.2f, %.2f \n".format(x1, y1)
```

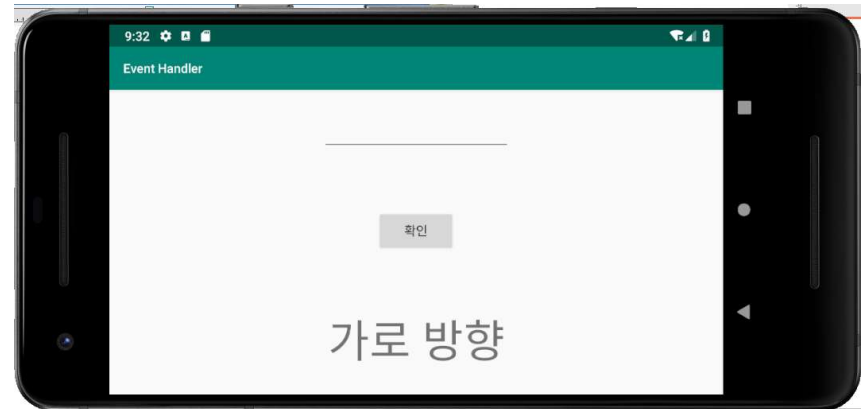
```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    activity_motion_event.setOnTouchListener{  
        , m: MotionEvent ->  
            handleTouch(m)  
        true  
    }  
}
```

Multi-touch 일 경우  
m.getX() 사용

Multi touch의 경우  
각 touch는  
pointer로 취급

각 point는  
Index로 참조되며,  
ID가 할당됨.

# 실습 7: 화면 방향 전환

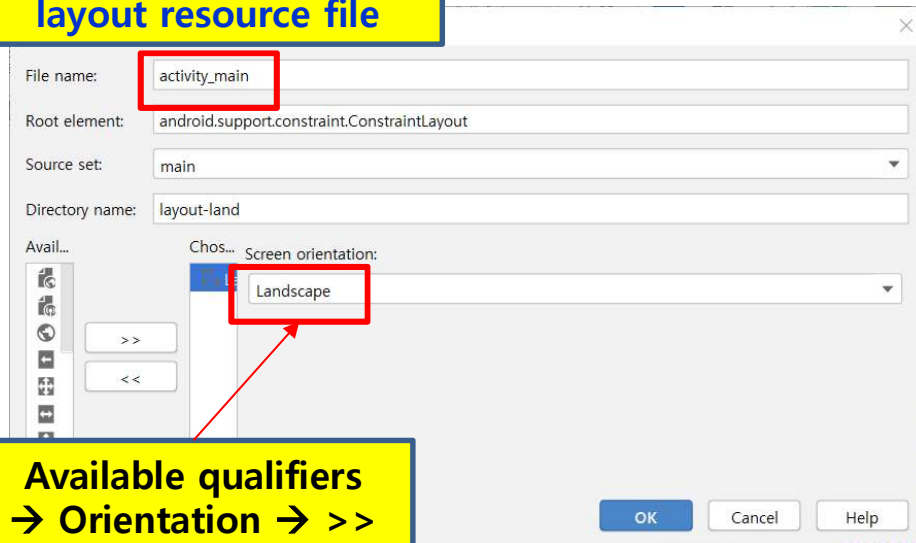


소스코드 - 12~14쪽

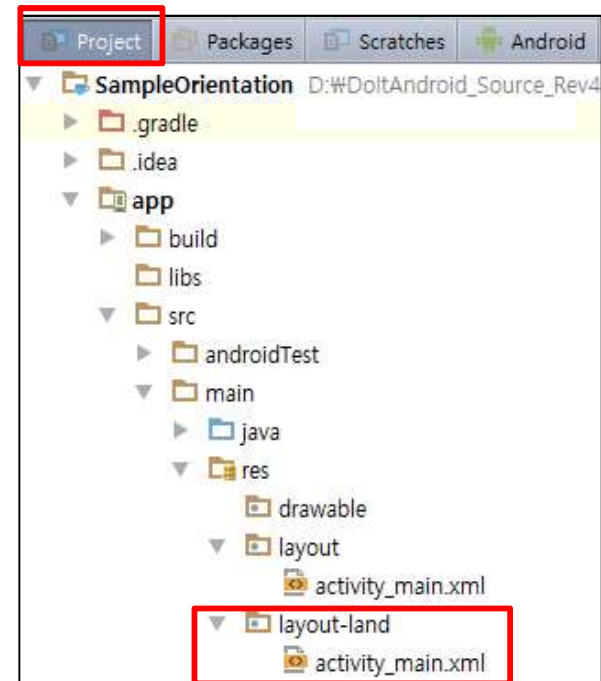
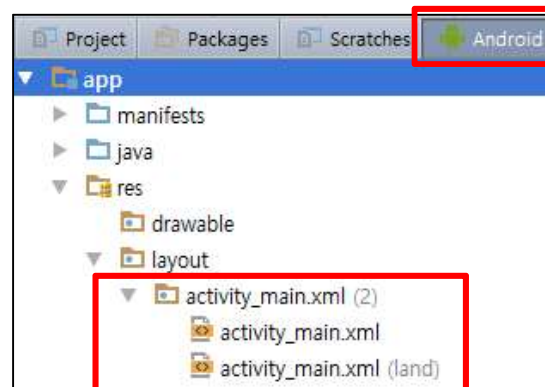


# 실습 7: landscape layout 추가

App > res > layout >  
오른쪽 버튼 > New >  
layout resource file

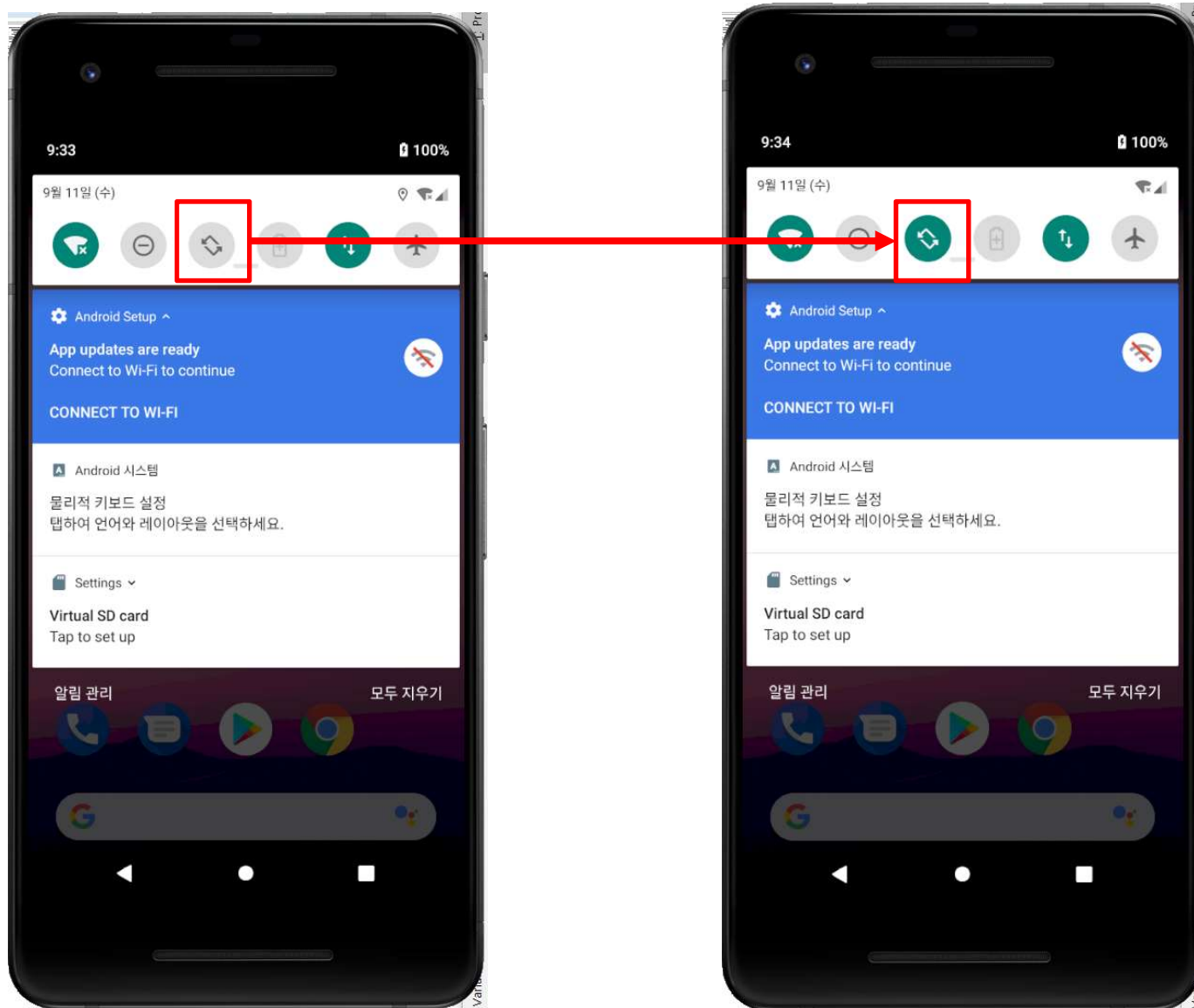


Available qualifiers  
→ Orientation → >>

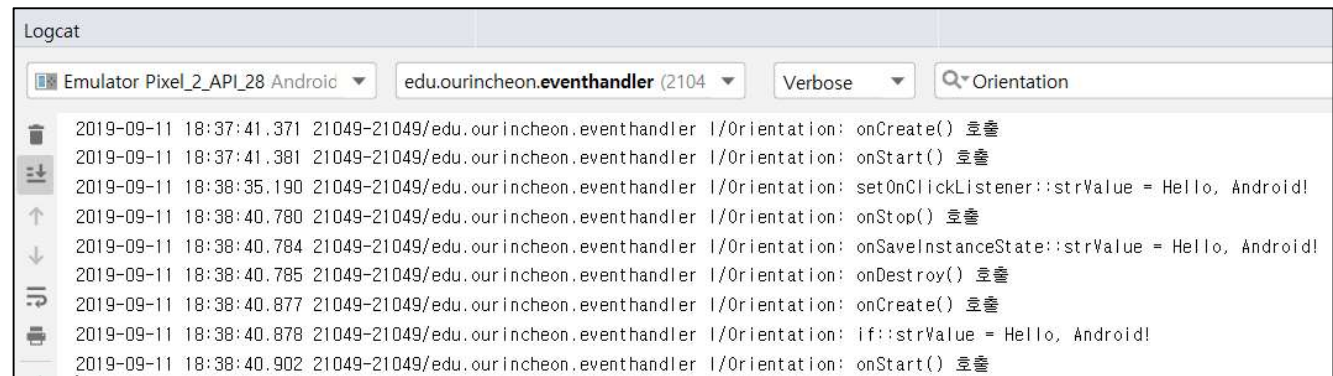
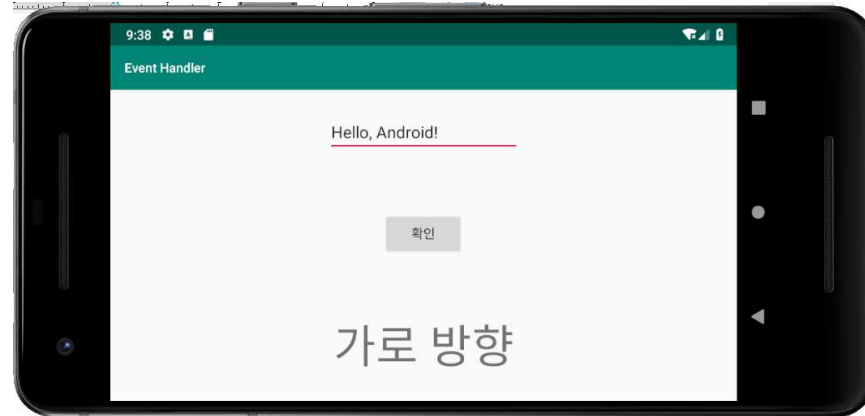
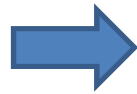
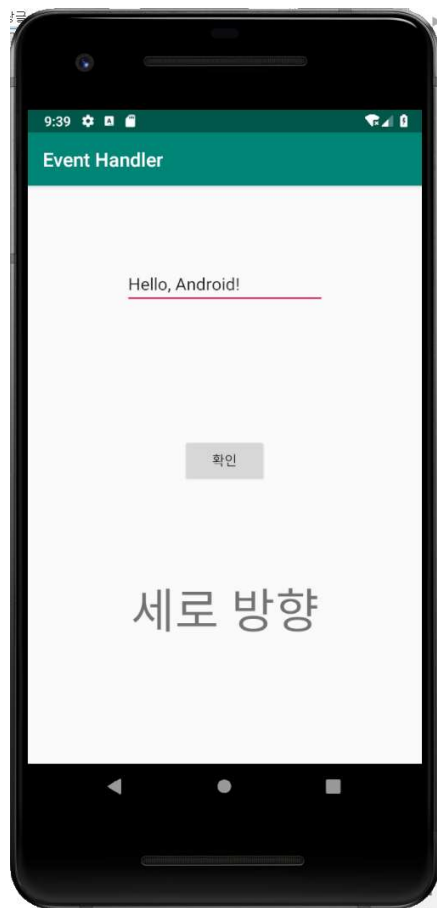


# 실습 7: 화면 전환 설정

↓  
화면 상단에서  
드래그



# 실습 7: 실행 결과



# 실습 7: 화면 방향 전환 (1/2)

```
class MainActivity : AppCompatActivity() {  
    val TAG = "Orientation"  
    var strValue = ""  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        Log.i(TAG, "onCreate() 호출")  
  
        button.setOnClickListener {  
            strValue = editText.text.toString()  
            Log.i(TAG, "setOnClickListener::strValue = $strValue")  
            Toast.makeText(applicationContext,  
                "입력 값 $strValue 저장", Toast.LENGTH_SHORT)  
                .show()  
        }  
  
        if (savedInstanceState != null) {  
            strValue = savedInstanceState.getString("strName")  
            Log.i(TAG, "savedInstanceState::strValue = $strValue")  
            Toast.makeText(this,  
                "저장된 값 $strValue 복원", Toast.LENGTH_SHORT)  
                .show()  
        }  
    }  
}
```

현재 어느 callback 메소드가 호출되고 있는가?

EditText 창에 입력된 값을 저장

이전 activity에  
저장되어 있던 값을 읽어 옴

## 실습 7: 화면 방향 전환 (2/2)

```
override fun onSaveInstanceState(outState: Bundle?) {  
    super.onSaveInstanceState(outState)  
    Log.i(TAG, "onSaveInstanceState::strValue = $strValue")  
    outState?.putString("strName", strValue)  
}  
  
override fun onStart() {  
    super.onStart()  
    Log.i(TAG, "onStart() 호출")  
}  
  
override fun onStop() {  
    super.onStop()  
    Log.i(TAG, "onStop() 호출")  
}  
  
override fun onDestroy() {  
    super.onDestroy()  
    Log.i(TAG, "onDestroy() 호출")  
}
```

현재 activity의 값을 저장  
**putString(이름, 값) → getString**

Activity의 life cycle

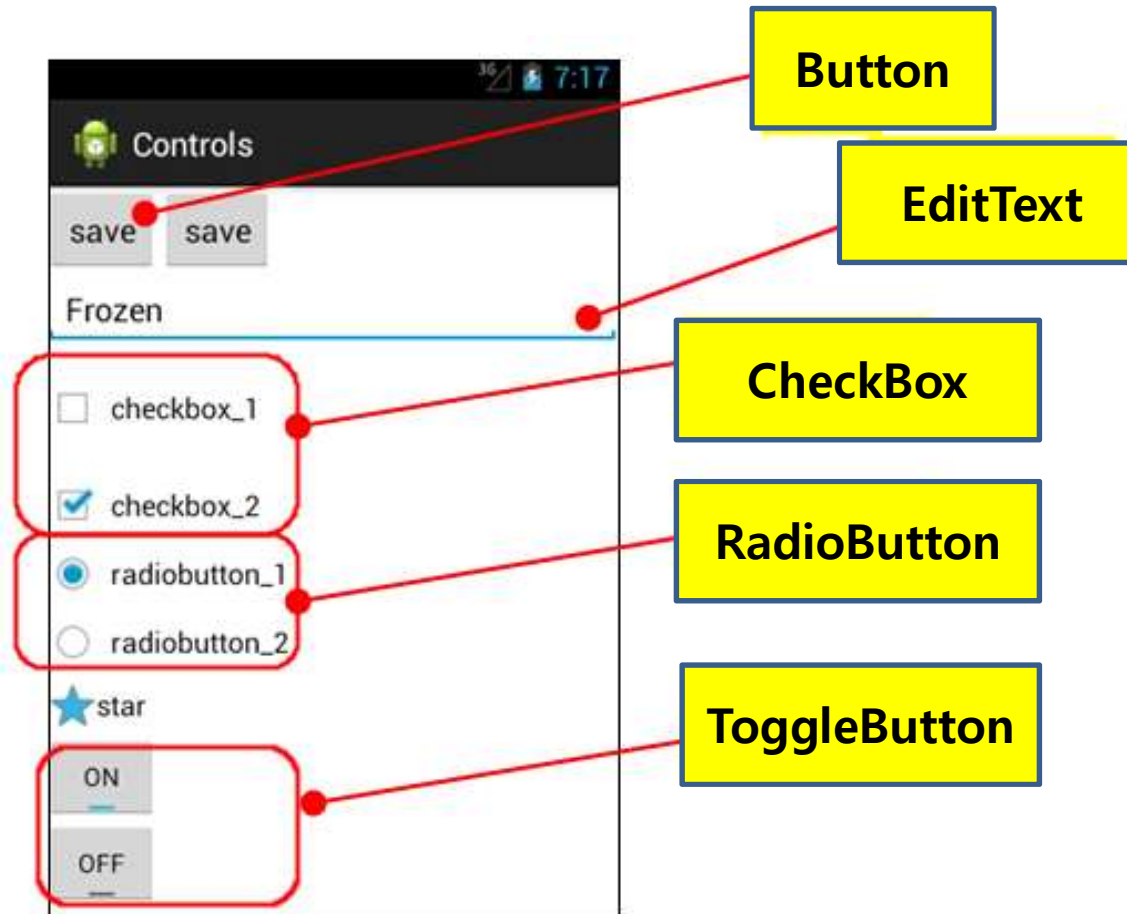
Activity의 life cycle

Activity의 life cycle

# What to do next?

- 이벤트 처리
  - 이벤트 핸들러를 구현하는 3가지 방법
    - 내부 클래스
    - Anonymous class
    - Activity 클래스에서 직접 처리
- 이벤트 종류
  - Touch 이벤트
  - 화면 방향(orientation) 전환
- **Basic widgets**
  - **Bitmap button**
  - **EditText**
  - **CheckBox**
  - **RadioButton**
  - **RatingBar**

# Basic widgets





# Button 과 ImageButton

```
<ImageButton  
    android:id="@+id/imageButton"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    app:srcCompat="@drawable/mybutton"/>
```



```
<Button  
    android:id="@+id/button2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:drawableLeft="@drawable/mybutton"  
    android:text="Press Me!"
```



## Button과 ImageButton의 차이점은?

**Button**은 **text** 속성을 지정할 수 있지만, **src** 속성은 지정할 수 없다.

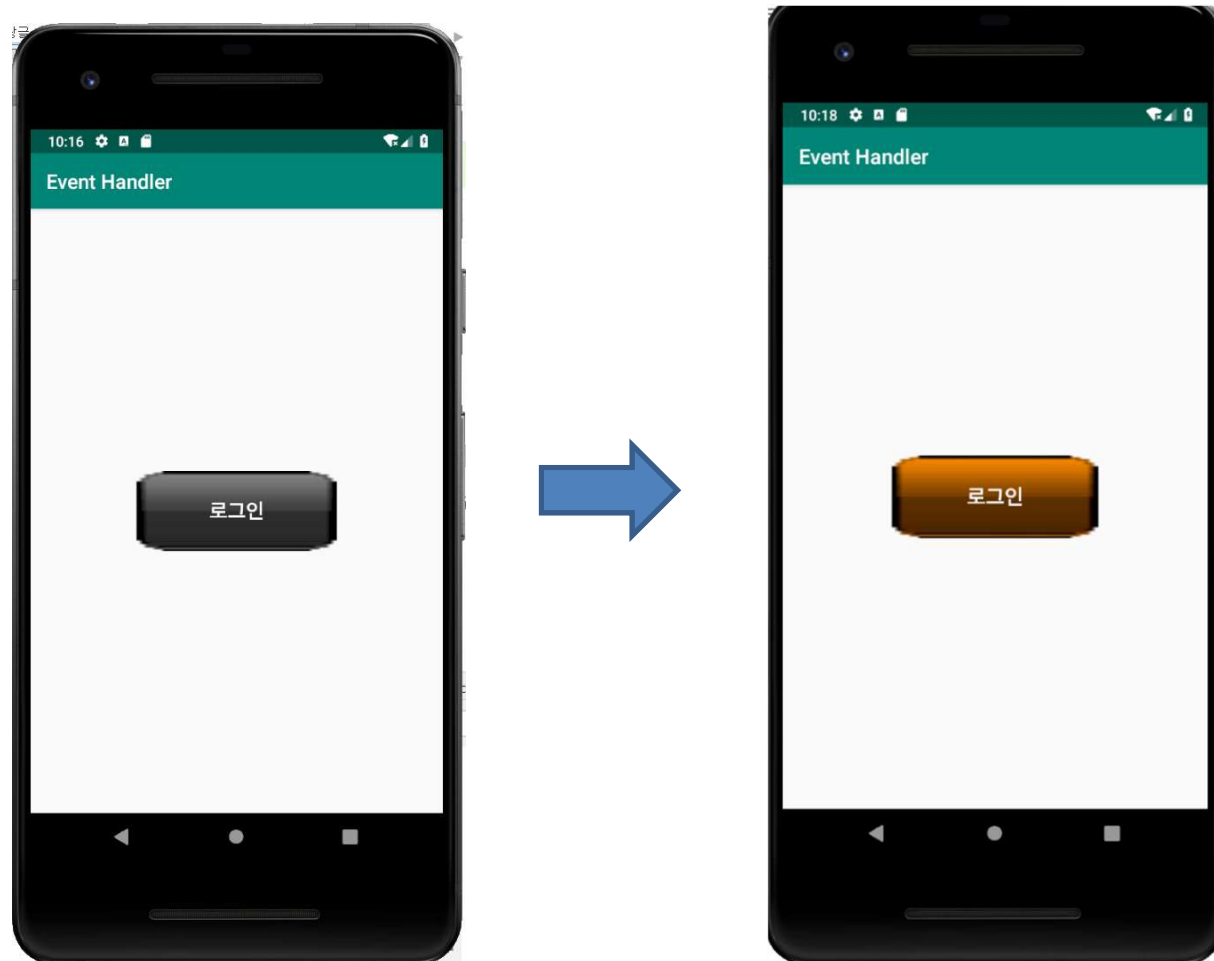
대신 **drawableLeft**, **drawableRight**, ... 속성으로 이미지를 나타낼 수 있다.

**ImageButton**은 **text** 속성을 지정할 수 없지만, **src** 속성을 지정할 수 있다.



## 실습 8 : 버튼 이미지 변경

- Button 이벤트에 따라 배경 색상이 바뀌도록 함



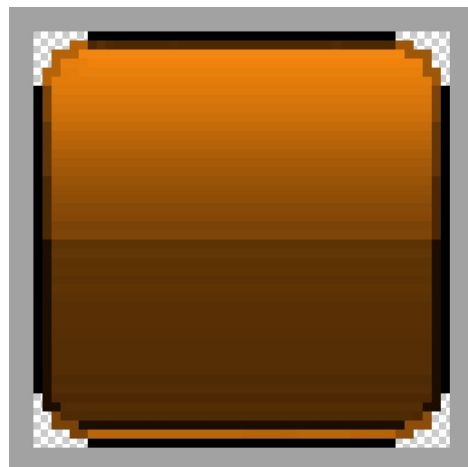
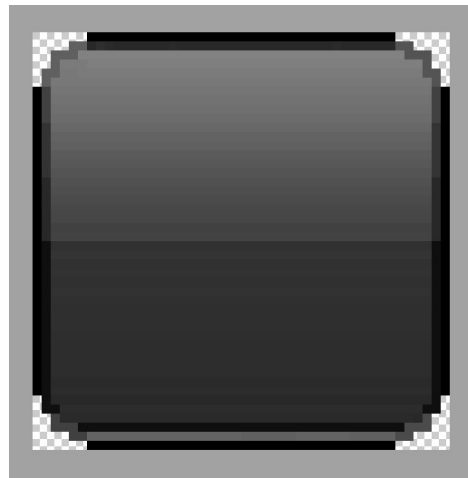
# 실습 8: background 버튼 이미지

res/drawable 폴더에 2개의 이미지 복사

2개의 kotlin 파일  
**BitmapButton.kt**  
**MainActivity.kt**

MainActivity.kt – 자동 생성된  
코드 그대로 사용

소스코드 참조 – 17쪽



9-patch 이미지

# 실습 8: 버튼 이미지 변경 (1/2)

BitmapButton.kt

```
class BitmapButton (context: Context) : AppCompatActivity(context) {  
    init {  
        AppCompatActivity(context)  
        initialize()  
    }  
  
    constructor(context: Context, attrs: AttributeSet) : this(context) {  
        AppCompatActivity(context, attrs)  
        initialize()  
    }  
  
    private fun initialize() {  
        setBackgroundResource(R.drawable.bitmap_button_normal_9)  
        text = "로그인"  
        setTextColor(Color.WHITE);  
        textSize = 20.0F;  
        typeface = Typeface.DEFAULT_BOLD;  
    }  
  
    override fun onTouchEvent(event: MotionEvent): Boolean {...}  
}
```

XML layout 에서  
이 객체를 사용하기 위해 필요

생성자의 2번째  
Parameter가  
AttributeSet

## 실습 8: 버튼 이미지 변경 (2/2)

BitmapButton.kt

```
override fun onTouchEvent(event: MotionEvent?): Boolean {  
    super.onTouchEvent(event)  
  
    when (event?.action) {  
        MotionEvent.ACTION_DOWN ->  
            setBackgroundResource(R.drawable.bitmap_button_clicked)  
        MotionEvent.ACTION_UP ->  
            setBackgroundResource(R.drawable.bitmap_button_normal)  
        else -> setBackgroundResource(R.drawable.bitmap_button_normal)  
    }  
    invalidate()  
  
    return true  
}
```

화면에 그릴  
내용이 변경되었다고 알림 →  
다시 그려 줄 것을 간접적으로 요청

# 실습 8: 버튼 이미지 변경 - Layout

activity\_main.xml

```
<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">

    <edu.ourincheon.eventhandler.BitmapButton
        android:id="@+id/button"
        android:layout_width="200dp"
        android:layout_height="80dp"
        android:layout_centerInParent="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

소스코드 - 17쪽

# 잠깐! onTouch와 onTouchEvent는 뭐가 다른가요?

## onTouch

```
public boolean onTouch(View v, MotionEvent event)
```

- 1- implement OnTouchListener
- 2- call setOnTouchListener() on the view you want to set catch the event
- 3- override onTouch() to handle the event.

onTouch() works on view, viewgroup, activity. Meaning you can use onTouch() inside view, viewgroup or activity.

## onTouchEvent

```
public boolean onTouchEvent(MotionEvent event)
```

but if you want to use onTouchEvent() you don't need to do step 1 & 2 above. just you need to override onTouchEvent().

**onTouchEvent() takes on one argument [ onTouchEvent(MotionEvent e) ]. Thus this can be used only inside the view that implements it or on the derived view.**

자료 인용 : <https://stackoverflow.com/questions/5002049/ontouchevent-vs-ontouch>

# android : **inputType**

- InputType 속성
  - **text**
  - **textEmailAddress**
  - **textUri** : URL 주소 입력
  - **number, phone, ...**
- 키보드 제어
  - **textCapSentences** : 문장의 첫 번째 글자를 대문자로
  - **textCapWords** : 각 단어의 첫 번째 글자를 대문자로
  - **textAutoCorrect** : 자동 완성
  - **textPassword**
  - **textMultiLine**



# EditText - **inputType** 속성(1/2)

```
<EditText  
    android:id="@+id/editTextBox03"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:inputType="textEmailAddress"  
    android:padding="10dp"  
    android:textSize="16dp" />
```



**android:inputType="textEmailAddress"**

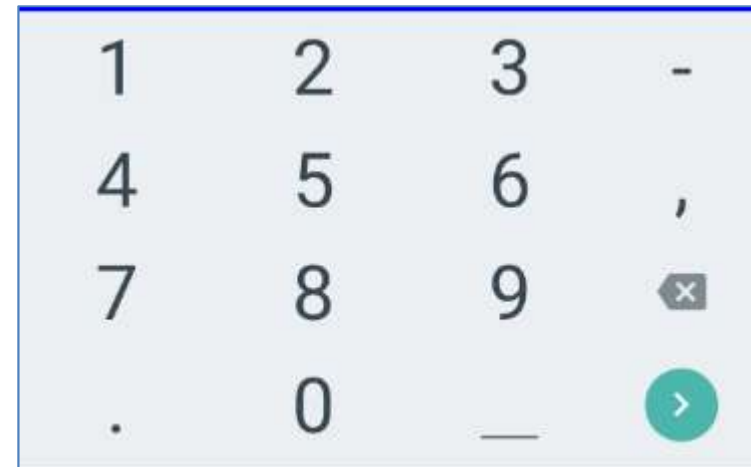


**android:inputType="phone"**

# EditText - **inputType** 속성(2/2)



**android:inputType**="text |  
textCapWords"



**android:inputType**="number |  
numberSigned | numberDecimal"

# imeOptions 이란?

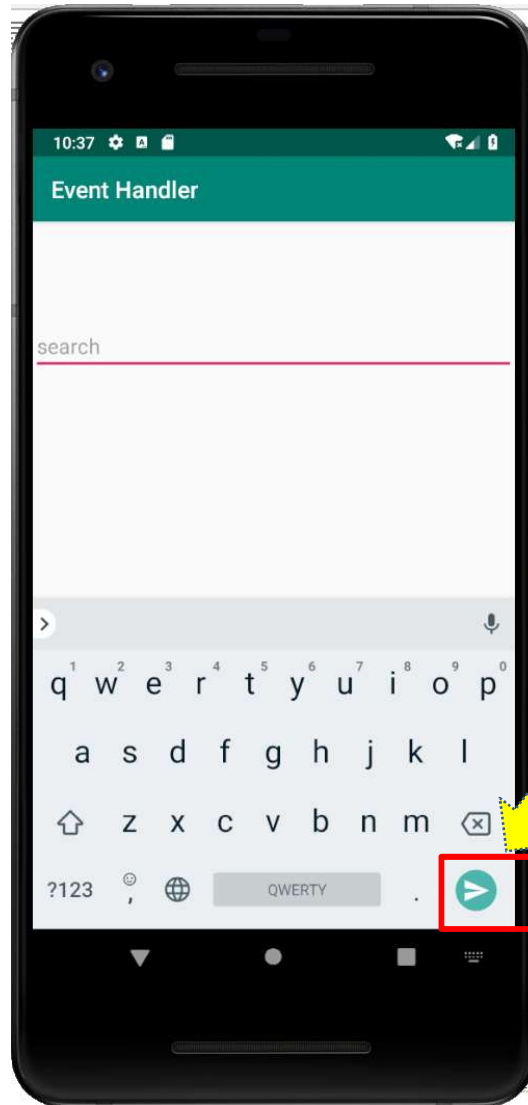
*IME(input method editor)*

EditText 창에 문자열 입력하려고 하면,  
화면 아래쪽에 keyboard가 나타남.

➔ 이 keyboard에서 Enter 키를 실행 목적에 맞게 바꿀 수 있는 것을 말함.



# 실습 9: EditText – imeOptions 속성



```
android:imeOptions="actionSend"
```

imeOptions 속성을  
**actionSend**로  
설정

소스코드 – 19쪽

# 실습 9: EditText

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        srchEditText.setOnEditorActionListener(  
            object: TextView.OnEditorActionListener {  
                override fun onEditorAction(v: TextView?,  
                    actionId: Int, event: KeyEvent?): Boolean {  
                    if (actionId == EditorInfo.IME_ACTION_SEND) {  
                        Toast.makeText(applicationContext,  
                            srchEditText.text,  
                            Toast.LENGTH_LONG).show()  
                        return true  
                    }  
                    return false  
                }  
            })  
    }  
}
```

소스코드 - 19~20쪽

## 실습 9: EditText – lambda version

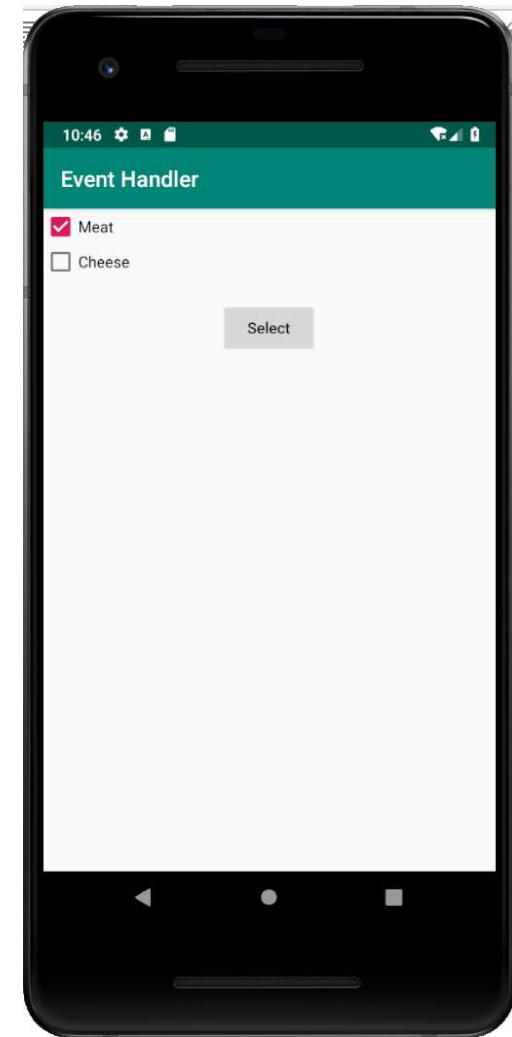
```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        srchEditText.setOnEditorActionListener { _, actionId, _ ->  
            if (actionId == EditorInfo.IME_ACTION_SEND) {  
                Toast.makeText(applicationContext,  
                    srchEditText.text,  
                    Toast.LENGTH_LONG).show()  
                true  
            }  
            false  
        }  
    }  
}
```

소스코드 – 20쪽

# 실습 10: CheckBox – Layout

```
<CheckBox
    android:id="@+id/checkBox_meat"
    android:text="Meat"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="left"
    android:onClick="onCheckBoxClicked"
    android:checked="true" />
<CheckBox
    android:id="@+id/checkBox_cheese"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="left"
    android:text="Cheese"
    android:onClick="onCheckBoxClicked" />
<Button
    android:id="@+id/btn_select"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:layout_gravity="center_horizontal"
    android:text="Select"
    android:textAllCaps="false"
    android:onClick="onSelected" />
```

소스코드 - 21쪽





# 실습 10: CheckBox

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    showMessage("meat", checkBox_meat.isChecked)  
    showMessage("cheese", checkBox_cheese.isChecked)  
}
```

```
fun onCheckBoxClicked(view: View) {  
    var checked = (view as CheckBox).isChecked  
  
    when (view.getId()) {  
        R.id.checkBox_meat -> showMessage("meat", checked)  
        R.id.checkBox_cheese -> showMessage("cheese", checked)  
    }  
}  
  
fun onSelected(view: View) {  
    val str = "$status_meat and $status_cheese"  
    Toast.makeText(this, str, Toast.LENGTH_LONG).show()  
}  
  
private fun showMessage(food: String, flag: Boolean) {  
    var status = StringBuilder()  
    status.append(food)  
    status.append(if (flag) " selected" else " unselected")  
  
    if (food == "meat") status_meat = status.toString()  
    else status_cheese = status.toString()  
  
    Toast.makeText(this, status.toString(),  
        Toast.LENGTH_LONG).show()  
}
```

```
private var status_meat = ""  
private var status_cheese = ""
```

소스코드 - 21~22쪽

# 실습 10: CheckBox – different code

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    status_meat.append("meat ")  
    status_cheese.append("cheese ")  
    showMessage(status_meat, checkBox_meat.isChecked)  
    showMessage(status_cheese, checkBox_cheese.isChecked)  
}
```

```
private var status_meat = StringBuilder()  
private var status_cheese = StringBuilder()
```

```
fun onCheckBoxClicked(view: View) {  
    var checked = (view as CheckBox).isChecked  
  
    when (view.id) {  
        R.id.checkBox_meat -> showMessage(status_meat, checked)  
        R.id.checkBox_cheese -> showMessage(status_cheese, checked)  
    }  
}  
  
fun onSelected(view: View) {  
    val str = "$status_meat and $status_cheese"  
    Toast.makeText(this, str, Toast.LENGTH_LONG).show()  
  
    // reset  
    status_meat.setLength(0)  
    status_cheese.setLength(0)  
    status_meat.append("meat ")  
    status_cheese.append("cheese ")  
}  
  
private fun showMessage(menu: StringBuilder, flag: Boolean) {  
    menu.append(if (flag) "selected" else "unselected")  
    Toast.makeText(this, menu, Toast.LENGTH_LONG).show()  
}
```

소스코드 – 22~23쪽

# 실습 11: RadioButton - Layout

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <RadioButton
        android:id="@+id/radio_red"
        android:text="Red"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:checked="true" />
    <RadioButton
        android:id="@+id/radio_blue"
        android:text="Blue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked" />
</RadioGroup>
```

소스코드 - 24쪽



# 실습 11: RadioButton

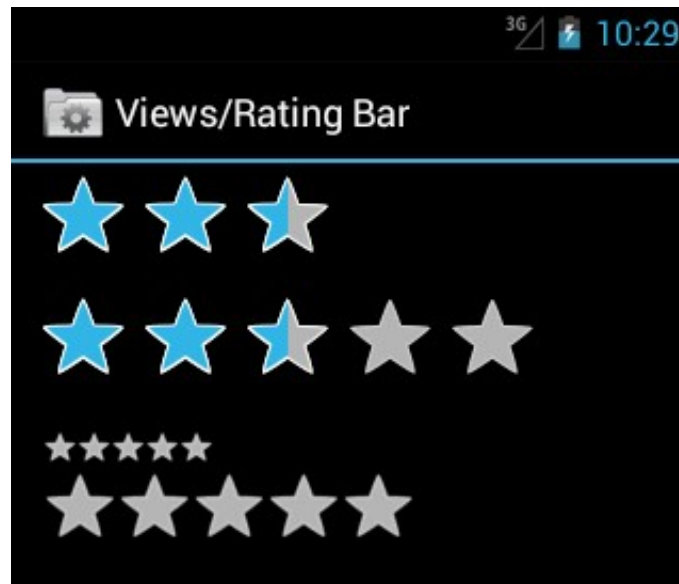
```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    if (radio_red.isChecked)  
        showToast("Red가 기본으로 선택되어 있습니다.")  
    else // if (radio_blue.isChecked)  
        showToast("Blue가 기본으로 선택되어 있습니다.")  
}  
  
fun onRadioButtonClicked(view: View) {  
    var sb = StringBuilder()  
    var str = ""  
  
    when (view.id) {  
        R.id.radio_red -> str = (view as RadioButton).text.toString()  
        R.id.radio_blue -> str = (view as RadioButton).text.toString()  
    }  
    sb.run {  
        append(str)  
        append("를 선택했군요!")  
    }  
    showToast(sb.toString())  
}  
  
private fun showToast(msg: String) {  
    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show()  
}
```

RadioGroup에 속한 버튼 중  
하나를 반드시 선택.  
→ 버튼의 checked  
상태를 확인할 필요가 없음.  
→ 어느 버튼에서  
이벤트가 발생했는지 확인.

소스코드 - 21~22쪽

# RatingBar

- 별을 사용하여 점수를 표시





# 실습 12: RatingBar - Layout

```
<?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">

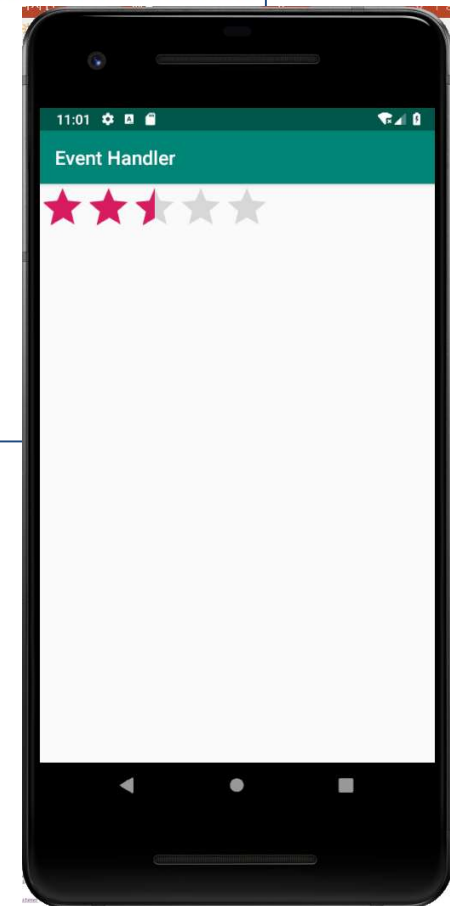
    <RatingBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ratingBar"
        android:numStars="5"
        android:stepSize="0.5"/>

</LinearLayout>
```

레이아웃 파일 : 소스코드 - 23쪽

android:layout\_width="wrap\_content"  
android:layout\_height="wrap\_content"

numStars가 제대로 나타나지 않을 경우  
viewGroup의 layout 속성을 조정



# 실습 12: RatingBar

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val ratingBarListener = RatingBar.OnRatingBarChangeListener {  
            ratingBar, rating, fromUser ->  
                Toast.makeText(applicationContext,  
                    "New rating: $rating", Toast.LENGTH_SHORT).show()  
        }  
  
        ratingBar.onRatingBarChangeListener = ratingBarListener  
    }  
}
```