이벤트 처리 (2)

Mobile Software

2022 Fall

What to do next?

- 이벤트 핸들러를 구현하는 3가지 방법
  - 중첩 클래스
  - Anonymous class
  - Activity 클래스에서 직접 상속
  - 실습 1: 다양한 단위 변환 방식 적용
  - 실습 2: 한 개의 view에 대한 2개의 이벤트 처리

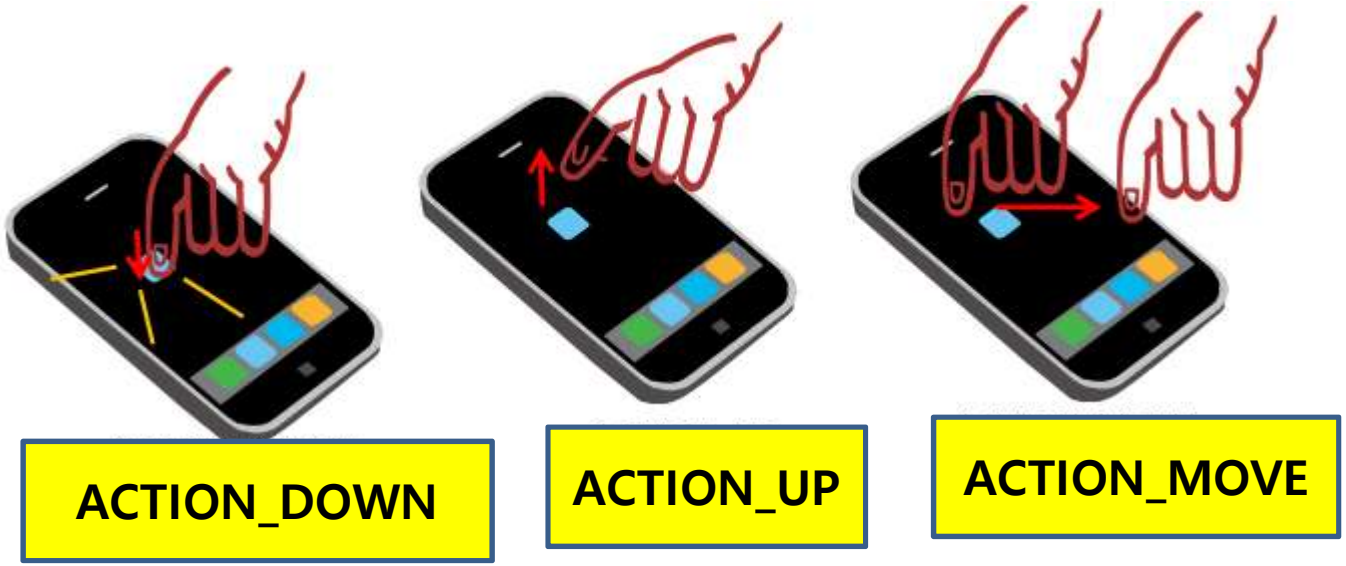
- **이벤트 종류**
  - **실습 3: single touch 이벤트**
  - 실습 4: multi-touch 이벤트

- **강의 노트에 포함된 코드: 5-2.소스코드.hwp**

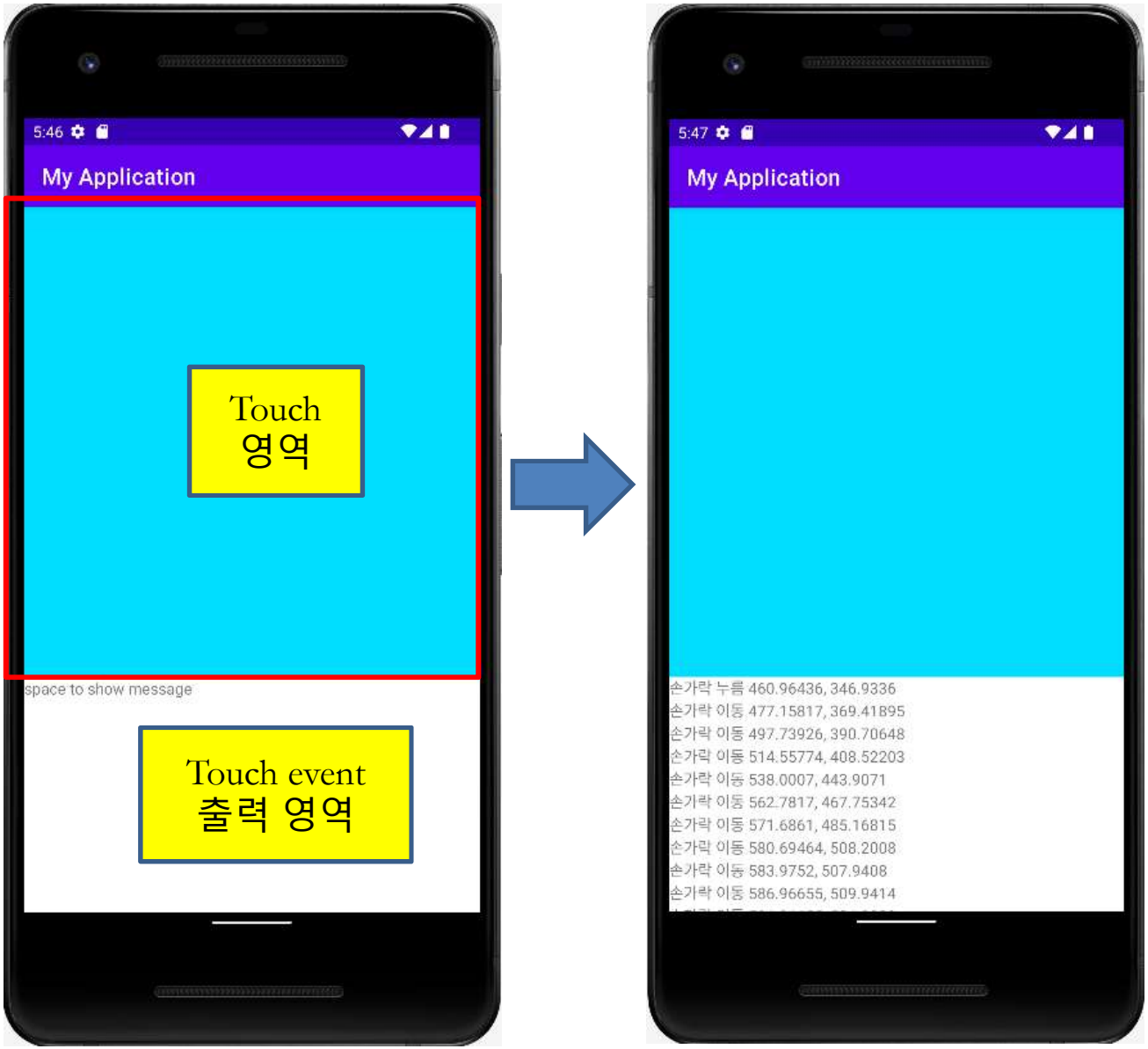
이벤트 종류

- **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

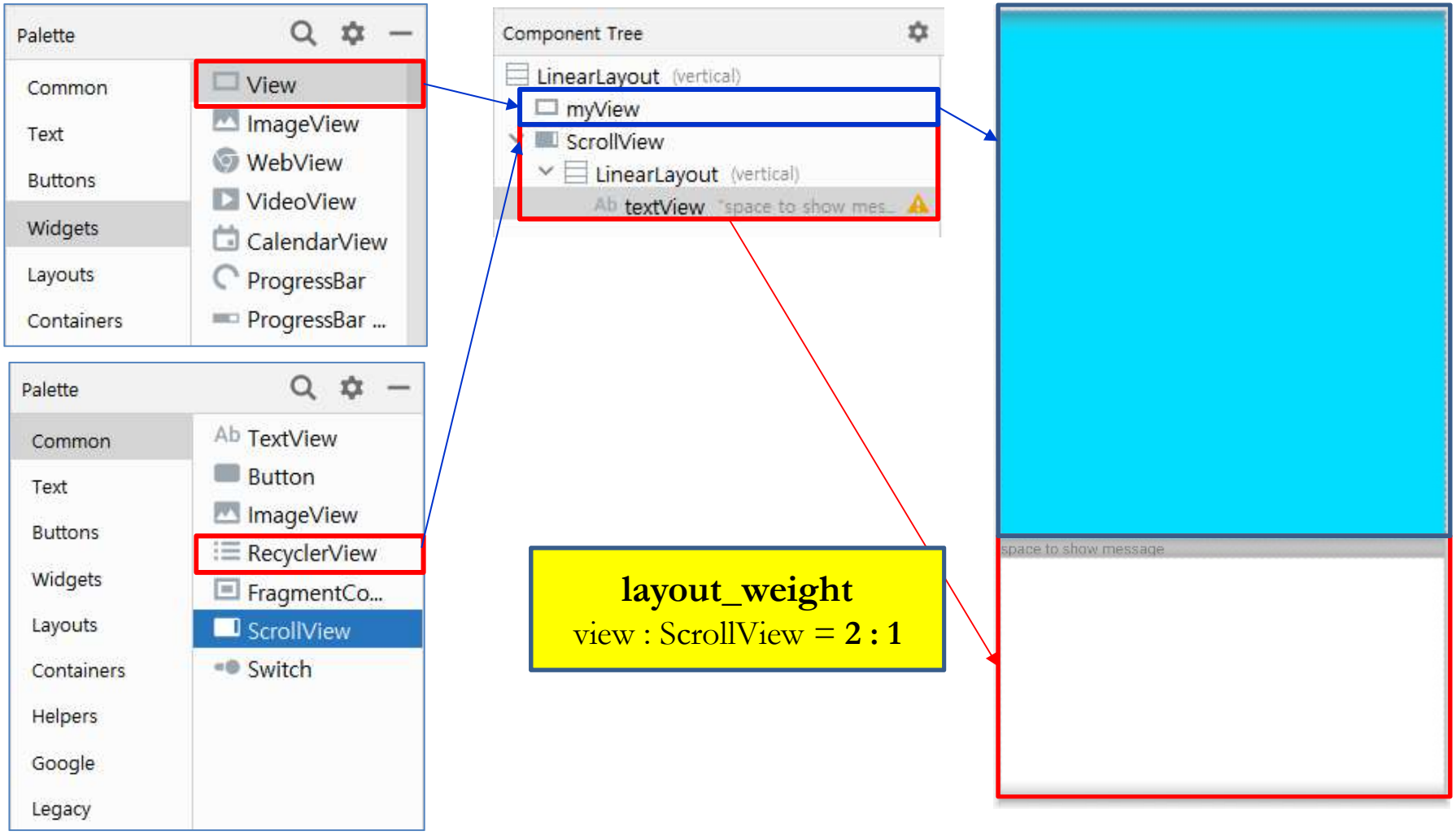
Touch 이벤트



실습 3: Single-Touch Event



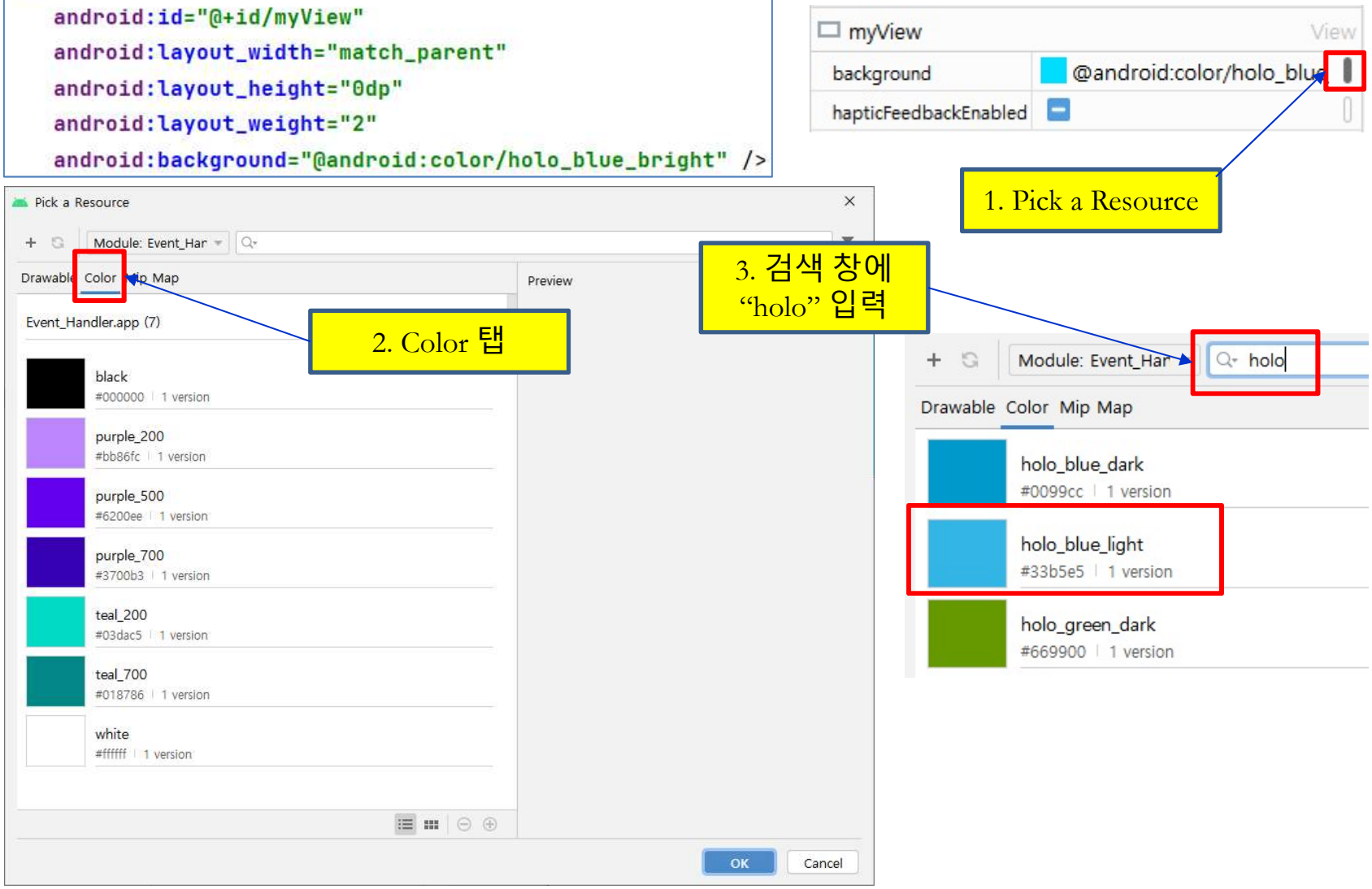
실습 3: Layout – View, ScrollView 추가



소스코드 – 1쪽

잠깐! background 색상 지정

```
<View
    android:id="@+id/myView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="2"
    android:background="@android:color/holo_blue_bright" />
```



실습 3: Activity 코드 구현 (1/2)

```
class MainActivity : AppCompatActivity() {
    private val sb = StringBuilder()

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

        val myView: View = findViewById(R.id.myView)
        myView.setOnTouchListener(object : View.OnTouchListener {
            override fun onTouch(v: View?, event: MotionEvent?): Boolean {
                handleTouch(event)
                return true
            }
        })
    }

    private fun handleTouch(m: MotionEvent?) {...}
}
```

```
myView.setOnTouchListener { v, event ->
    handleTouch(event)
    true  ^setOnTouchListener
}
```

소스코드 – 2쪽

실습 3: Activity 코드 구현 (2/2)

Touch 이벤트 발생 위치

Touch 이벤트 종류 구분

```
private fun handleTouch(m: MotionEvent?) {
    var x:Float? = m?.x
    var y:Float? = m?.y
    var textView: TextView = findViewById(R.id.textView)

    val sb = StringBuilder()
    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 뗌 $x, $y\n")
        else -> sb.append("\n")
    }
    textView.text = sb.toString()
}
```

소스코드 – 2쪽

잠깐! Safe call (1/2)

event 가 null 값을
가질 수도 있는 경우이면?

```
myView.setOnTouchListener { v: View?, event: MotionEvent? ->
    handleTouch(event)
    true   ^setOnTouchListener
}
```

```
private fun handleTouch(m: MotionEvent?) {
    var x:Float? = m?.x
    var y:Float? = m?.y
    var textView:TextView = findViewById(R.id.textView)

    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 뗌 $x, $y\n")
        else -> sb.append("\n")
    }
    textView.text = sb.toString()
}
```

Safe call: type 뒤에 ?을 붙여
전달된 파라미터가
Null 값을 가질 수 있음을 표시

객체 m을 사용하는 곳에는
safe call 표기를 모두 추가.

잠깐! Safe call (2/2)

event 가 null 값을
가질 수도 있는 경우이면?

```
myView.setOnTouchListener { v: View?, event: MotionEvent? ->
    handleTouch(event)
    true   ^setOnTouchListener
}
```

조금 더 효과적인 코딩 스타일은?

```
private fun handleTouch(m: MotionEvent?) {
    if (m == null) return
    var x:Float = m.x
    var y:Float = m.y
    var textView:TextView = findViewById(R.id.textView)

    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 뗌 $x, $y\n")
        else -> sb.append("\n")
    }
    textView.text = sb.toString()
}
```

entry point에서
m이 null 인 경우를 차단

소스코드 – 3쪽

Quiz 1: 아래 코드 실행 결과는?

```
class MainActivity : AppCompatActivity() {
    // private val sb = StringBuilder()

    override fun onCreate(savedInstanceState: Bundle?) {...}

    private fun handleTouch(m: MotionEvent?) {
        val sb = StringBuilder()

        if (m == null) return
        var x:Float? = m.x
        var y:Float? = m.y
        var textView: TextView = findViewById(R.id.textView)

        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 뗌 $x, $y\n")
            else -> sb.append("\n")
        }
        textView.text = sb.toString()
    }
}
```

실습 3: 출력 서식 지정 (1/2)

```
private fun handleTouch(m: MotionEvent?) {
    if (m == null) return
    var x:Float = m.x
    var 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 뗌", x, y)
        else -> outMessage("액션 없음", x, y)
    }
}

private fun outMessage(msg: String, x1:Float, y1:Float) {
    var textView:TextView = findViewById(R.id.textView)
    sb.append("%-10s: %.2f, %.2f \n".format(msg, x1, y1))
    textView.text = sb.toString()
}
```

```
손가락 누름   : 307.97, 227.93
손가락 이동   : 316.12, 233.12
손가락 이동   : 336.82, 252.88
손가락 이동   : 367.23, 277.71
손가락 이동   : 391.44, 297.94
손가락 이동   : 407.31, 309.30
손가락 이동   : 429.55, 322.58
손가락 뗌     : 428.96, 321.97
손가락 누름   : 495.97, 421.93
손가락 이동   : 498.97, 421.93
```

소수점 이하
2째 자리까지 출력

소스코드 – 4쪽

실습 3: 출력 서식 지정 (2/2)

```
when (m.action) {
    MotionEvent.ACTION_DOWN -> outMessage2("손가락 누름", x, y)
    MotionEvent.ACTION_MOVE -> outMessage2("손가락 이동", x, y)
    MotionEvent.ACTION_UP -> outMessage2("손가락 뗌", x, y)
    else -> outMessage2("액션 없음", x, y)
}
```

```
import java.math.RoundingMode
import java.text.DecimalFormat
```

```
private fun outMessage2(msg: String, x: Float, y: Float) {
    var textView:TextView = findViewById(R.id.textView)
    val df = DecimalFormat("#.##")
    df.roundingMode = RoundingMode.CEILING

    sb.append("$msg : ${df.format(x)}, ${df.format(y)} \n")
    textView.text = sb.toString()
}
```

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

소스코드 – 4쪽

What to do next?

- 이벤트 핸들러를 구현하는 3가지 방법
  - 중첩 클래스
  - Anonymous class
  - Activity 클래스에서 직접 상속
  - 실습 1: 다양한 단위 변환 방식 적용
  - 실습 2: 한 개의 view에 대한 2개의 이벤트 처리

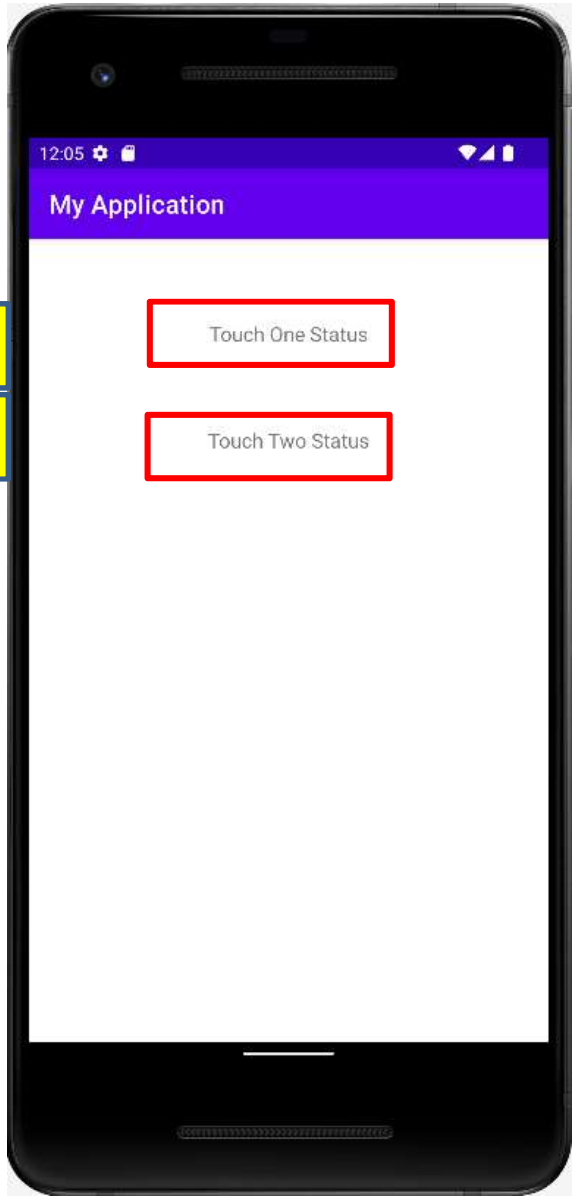
- **이벤트 종류**
  - 실습 3: single touch 이벤트
  - **실습 4: multi-touch 이벤트**

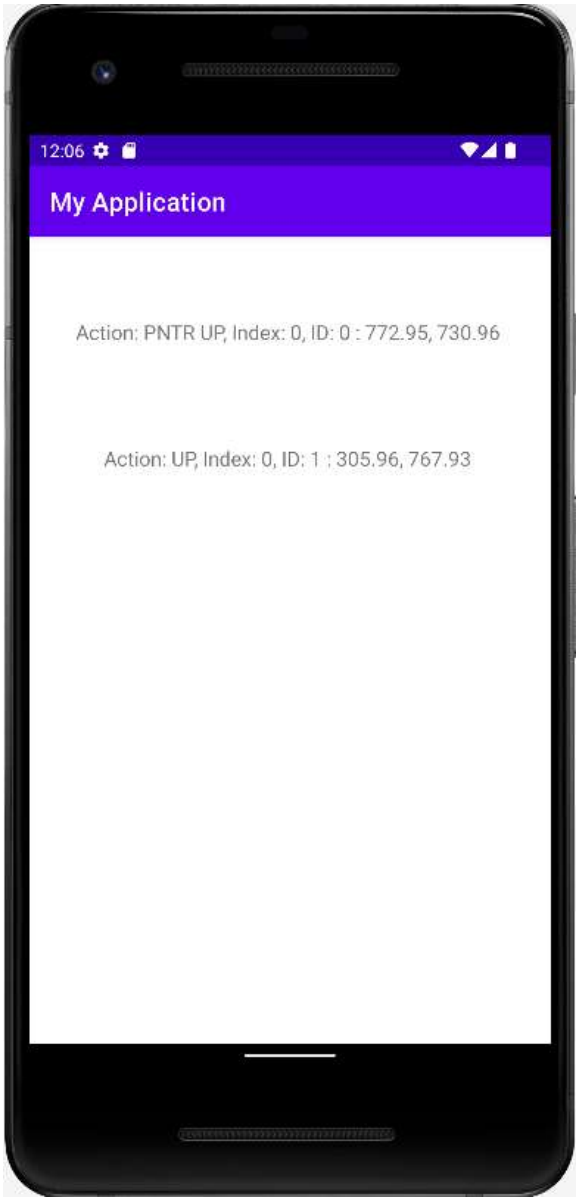
실습 4: Multi-Touch Event

Single touch

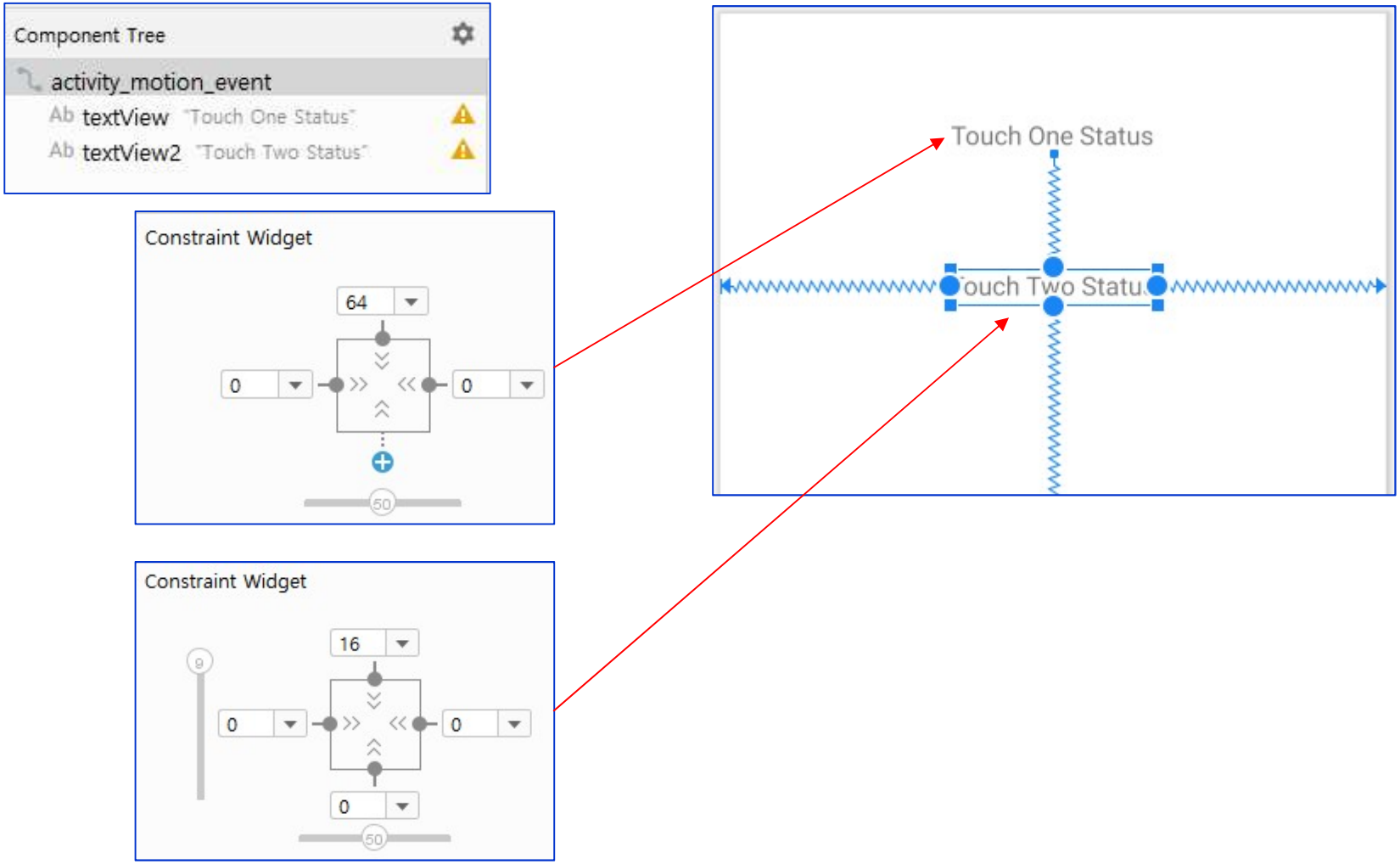
Multi touch

Ctrl 키를
누른 채
클릭





실습 4: Layout – 2개의 textView



소스코드 – 5쪽

실습 4: Activity 코드 구현 (1/2)

```
class MainActivity : AppCompatActivity() {

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

        val activityMotionEvent:ConstraintLayout
                = findViewById(R.id.activity_motion_event)
        activityMotionEvent.setOnTouchListener( object: View.OnTouchListener {
            override fun onTouch(v: View?, event: MotionEvent?): Boolean {
                handleTouch(event)
                return true
            }
        })
    }

    private fun handleTouch(m: MotionEvent?) {...}

    private fun outMessage(msg: String, x1: Float, y1: Float) =
        "%s : %.2f, %.2f \n".format(msg, x1, y1)

}
```

Root view인 ConstraintLayout의 공간에서 발생하는 이벤트를 처리

return 값이 하나 ➔ 수식 할당 형태로 표현

소스코드 – 5~6쪽

실습 4: Activity 코드 구현 (2/2)

```
private fun handleTouch(m: MotionEvent) {
    val pointerCount = m.pointerCount
    var textView:TextView = findViewById(R.id.textView)
    var textView2:TextView = findViewById(R.id.textView2)

    val actionIndex:Int = m.actionIndex
    for (i in 0 until pointerCount) {
        val x:Float = m.getX(i)
        val y:Float = m.getY(i)
        val id:Int = m.getPointerId(i)

        var actionString = when (m.action) {
            MotionEvent.ACTION_DOWN -> "DOWN"
            MotionEvent.ACTION_UP -> "UP"
            MotionEvent.ACTION_POINTER_DOWN -> "PNTR DOWN"
            MotionEvent.ACTION_POINTER_UP -> "PNTR UP"
            MotionEvent.ACTION_MOVE -> "MOVE"
            else -> ""
        }

        var message = "Index: $actionIndex, Action: $actionString, ID: $id"
        if (id == 0) textView.text = outMessage(message, x, y)
        if (id == 1) textView2.text = outMessage(message, x, y)
    }
}
```

각 pointe는 Index로 참조되며, ID가 할당됨.

Multi-touch 일 경우 m.getX() 사용

Multi touch의 경우 각 touch는 pointer로 취급

소스코드 – 5~6쪽