6. 사용자 이벤트 처리하기

1. 터치와 키 이벤트
2. 뷰 이벤트
3. 시계 앱의 스톱워치 기능 만들기

1. 터치와 키 이벤트

터치 이벤트

- 터치 이벤트의 콜백 함수인 onTouchEvent()를 선언
- 매개변수는 MotionEvent 객체이며, 이 객체에 터치의 종류와 발생 지점(좌푯값)이 담김

• 터치 이벤트 처리

```
class MainActivity : AppCompatActivity() {
    (... 생략 ...)
    override fun onTouchEvent(event: MotionEvent?): Boolean {
        return super.onTouchEvent(event)
    }
}
```

1. 터치와 키 이벤트

- **터치 이벤트의 종류**
  - ACTION_DOWN: 화면을 손가락으로 누른 순간의 이벤트
  - ACTION_UP: 화면에서 손가락을 떼는 순간의 이벤트
  - ACTION_MOVE: 화면을 손가락으로 누른 채로 이동하는 순간의 이벤트

• 터치 이벤트 처리

```
override fun onTouchEvent(event: MotionEvent?): Boolean {
    when (event?.action) {
        MotionEvent.ACTION_DOWN -> {
            Log.d("kkang", "Touch down event")
        }
        MotionEvent.ACTION_UP -> {
            Log.d("kkang", "Touch up event")
        }
    }
    return super.onTouchEvent(event)
}
```

1. 터치와 키 이벤트

터치 이벤트

- 터치 이벤트 발생 좌표 얻기
- onTouchEvent() 함수의 매개변수인 MotionEvent 객체로 획득
  - x: 이벤트가 발생한 뷰의 X 좌표
  - y: 이벤트가 발생한 뷰의 Y 좌표
  - rawX: 화면의 X 좌표
  - rawY: 화면의 Y 좌표

• 터치 이벤트가 발생한 좌표 얻기

```
override fun onTouchEvent(event: MotionEvent?): Boolean {
    when (event?.action) {
        MotionEvent.ACTION_DOWN -> {
            Log.d("kkang",
                "Touch down event x: ${event.x}, rawX: ${event.rawX}")
        }
    }
    return super.onTouchEvent(event)
}
```

1. 터치와 키 이벤트

- **키 이벤트**
  - 사용자가 폰의 키를 누르는 순간에 발생
  - 콜백 함수
    - onKeyDown: 키를 누른 순간의 이벤트
    - onKeyUp: 키를 떼는 순간의 이벤트
    - onKeyLongPress: 키를 오래 누르는 순간의 이벤트
  - Focusable
    - view가 keyEvent를 받기 위해서 포커서 설정

```
android:focusable="true"
```

• 키 이벤트 처리

```
class MainActivity2 : AppCompatActivity() {
    (... 생략 ...)
    override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {
        Log.d("kkang", "onKeyDown")
        return super.onKeyDown(keyCode, event)
    }
    override fun onKeyUp(keyCode: Int, event: KeyEvent?): Boolean {
        Log.d("kkang", "onKeyUp")
        return super.onKeyUp(keyCode, event)
    }
}
```

1. 터치와 키 이벤트

- 콜백 함수 매개변수 : 첫 번째 매개변수는 키의 코드이며 이 값으로 사용자가 어떤 키를 눌렀는지 식별

• 어떤 키를 눌렀는지 식별

```
override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {
    when (keyCode) {
        KeyEvent.KEYCODE_0 -> Log.d("kkang", "0 키를 눌렀네요")
        KeyEvent.KEYCODE_A -> Log.d("kkang", "A 키를 눌렀네요")
    }
    return super.onKeyDown(keyCode, event)
}
```

1. 터치와 키 이벤트

- 키 이벤트가 발생하는 키는 폰에서 제공하는 소프트 키보드의 키를 의미하지 않습니다.
- 안드로이드 시스템 버튼도 키로 취급하므로 이 버튼의 이벤트를 처리
- 뒤로가기 버튼 이벤트에는 앞에서 살펴본 onKeyDown()이나 onKeyUp() 함수를 이용할 수도 있지만 onBackPressed() 함수를 이용할 수도 있다.

• 뒤로가기 버튼과 볼륨 조절 버튼의 이벤트 처리

```
override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {
    when (keyCode) {
        KeyEvent.KEYCODE_BACK -> Log.d("kkang", "BACK Button을 눌렀네요")
        KeyEvent.KEYCODE_VOLUME_UP -> Log.d("kkang", "Volume Up 키를 눌렀네요")
        KeyEvent.KEYCODE_VOLUME_UP -> Log.d("kkang", "Volume Down 키를 눌렀네요")
    }
    return super.onKeyDown(keyCode, event)
}
```

• 뒤로가기 버튼의 이벤트 처리

```
override fun onBackPressed() {
    Log.d("kkang", "Back Button을 눌렀네요")
}
```

2. 뷰 이벤트

뷰 이벤트의 처리 구조

- 뷰 이벤트 처리 : 이벤트 소스와 이벤트 핸들러로 역할이 나뉘며 이 둘을 리스너로 연결해야 이벤트를 처리할 수 있다.
  - 이벤트 소스: 이벤트가 발생한 객체
  - 이벤트 핸들러: 이벤트 발생 시 실행할 로직이 구현된 객체
  - 리스너: 이벤트 소스와 이벤트 핸들러를 연결해 주는 함수

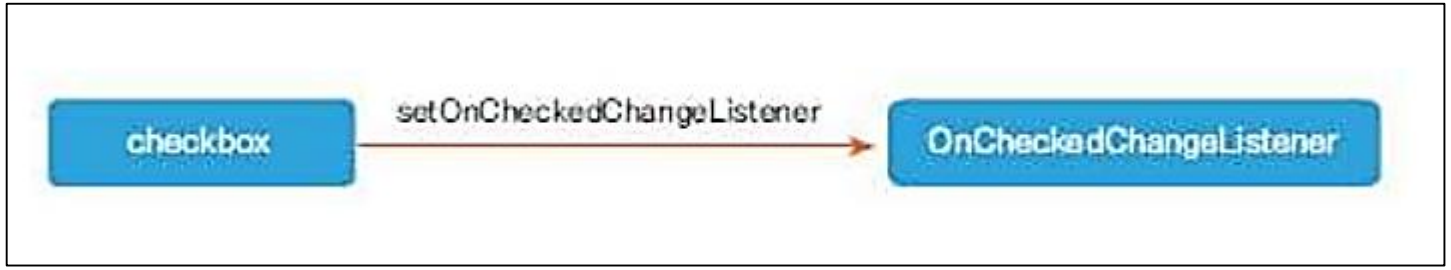
2. 뷰 이벤트

- 이벤트 처리 : 익명의 리스너 사용

• 체크박스 이벤트 처리

이벤트 소스 / 리스너(이벤트 핸들러 등록) / 이벤트 핸들러

```
binding.checkbox.setOnCheckedChangeListener(object: CompoundButton.OnCheckedChangeLis-
tener {
    override fun onCheckedChanged(p0: CompoundButton?, p1: Boolean) {
        Log.d("kkang", "체크박스 클릭")
    }
})
```



2. 뷰 이벤트

- 액티비티에 이벤트 리스너 인터페이스 구현하기

• 액티비티에서 인터페이스를 구현한 예

```
class MainActivity3 : AppCompatActivity(), CompoundButton.OnCheckedChangeListener {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding = ActivityMain3Binding.inflate(layoutInflater)
        setContentView(binding.root)
        binding.checkbox.setOnCheckedChangeListener(this)
    }
    override fun onCheckedChanged(p0: CompoundButton?, p1: Boolean) {
        Log.d("kkang", "체크박스 클릭")
    }
}
```

2. 뷰 이벤트

- 이벤트 리스터 클래스 작성

• 이벤트 핸들러를 별도의 클래스로 만든 예

```
class MyEventHandler : CompoundButton.OnCheckedChangeListener {
    override fun onCheckedChanged(p0: CompoundButton?, p1: Boolean) {
        Log.d("kkang", "체크박스 클릭")
    }
}
class MainActivity3 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding = ActivityMain3Binding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.checkbox.setOnCheckedChangeListener(MyEventHandler())
    }
}
```

2. 뷰 이벤트

- **SAM(Single Abstract Method) 기법으로 구현 :** 하나의 추상메소드만 제공하는 이벤트 리스너인 경우 : lambda 식 제공

• SAM 기법으로 구현한 예

```
class MainActivity3 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val binding = ActivityMain3Binding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.checkbox.setOnCheckedChangeListener {
            compoundButton, b ->
            Log.d("kkang", "체크박스 클릭")
        }
    }
}
```

2. 뷰 이벤트

▪ 클릭과 롱클릭 이벤트 처리

- ClickEvent, LongClickEvent는 뷰의 최상위 클래스인 View에 정의된 이벤트
  - open fun setOnClickListener(l: View.OnClickListener?): Unit
  - open fun setOnLongClickListener(l: View.OnLongClickListener?): Unit

• 버튼의 클릭, 롱클릭 이벤트 처리

```
binding.button.setOnClickListener {
    Log.d("kkang", "클릭 이벤트")
}

binding.button.setOnLongClickListener {
    Log.d("kkang", "롱클릭 이벤트")
    true
}
```

실습 : 시계 앱의 스톱워치 기능 만들기

- **1단계. 새 모듈 생성하기**
  - Ch8_Event라는 이름으로 새로운 모듈을 작성

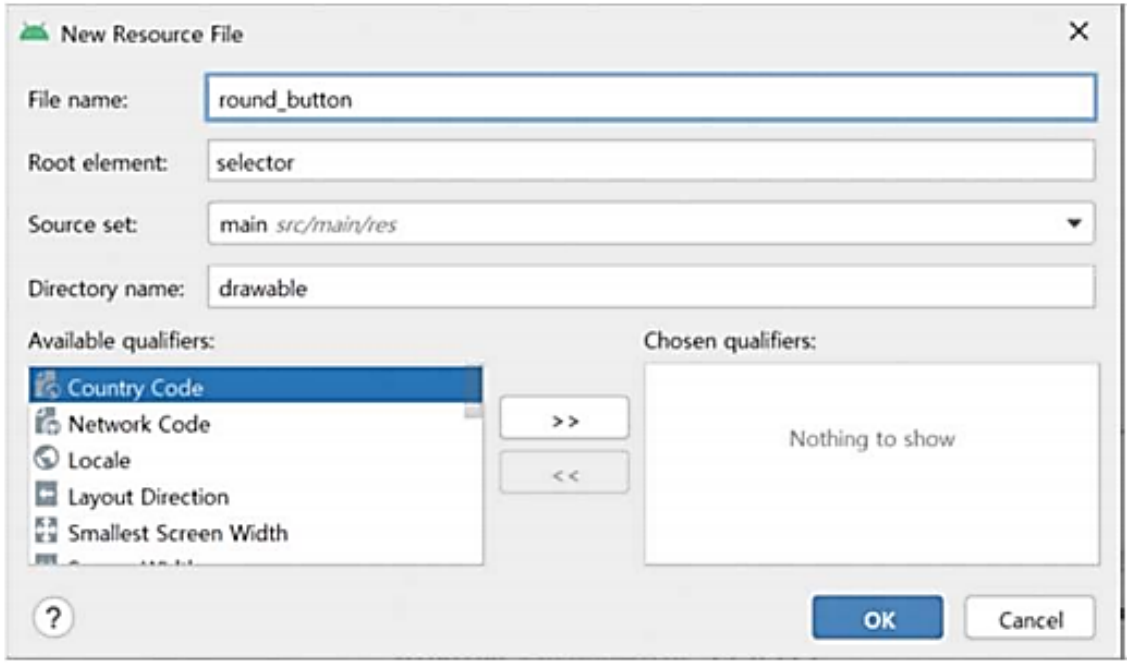
- **2단계. 그래들 설정하기**

Do it! • build.gradle (Module: AndroidLab.ch8_event)

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

3단계. 둥근 버튼 만들기

- [res → drawable] 디렉터리를 마우스 오른쪽 버튼으로 눌러 [New → Drawable Resource File] 메뉴를 선택
- round_button.xml 파일



4단계. 앱 화면 구성하기

- activity_main.xml 파일

실습 : 시계 앱의 스톱워치 기능 만들기

- **5단계. 메인 액티비티 작성하기**
  - MainActivity.kt 파일
- **6단계. 앱 실행하기**