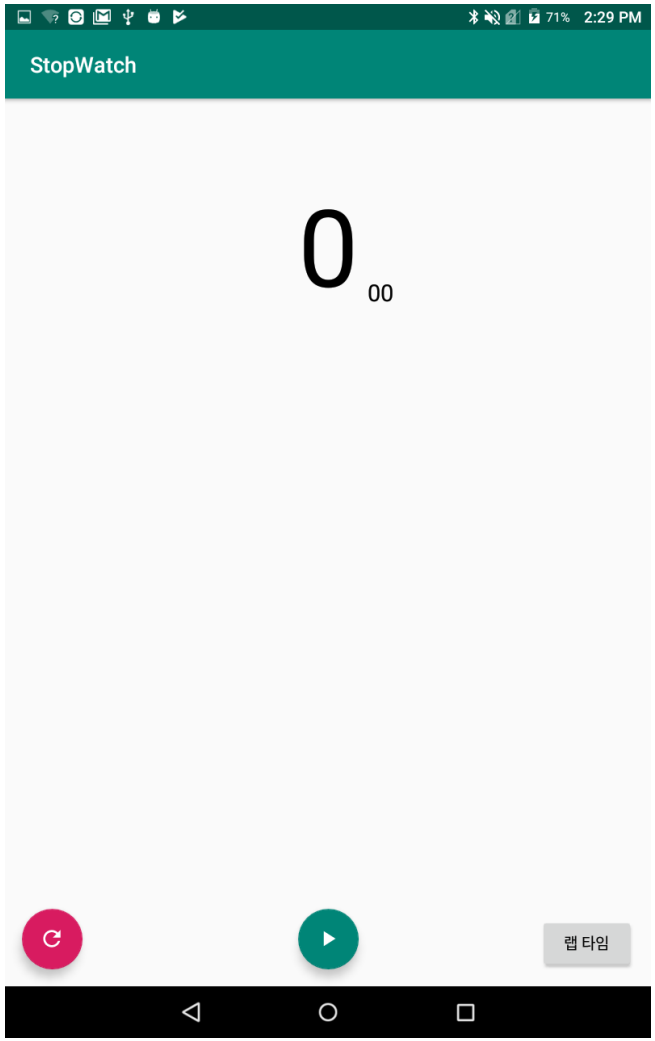
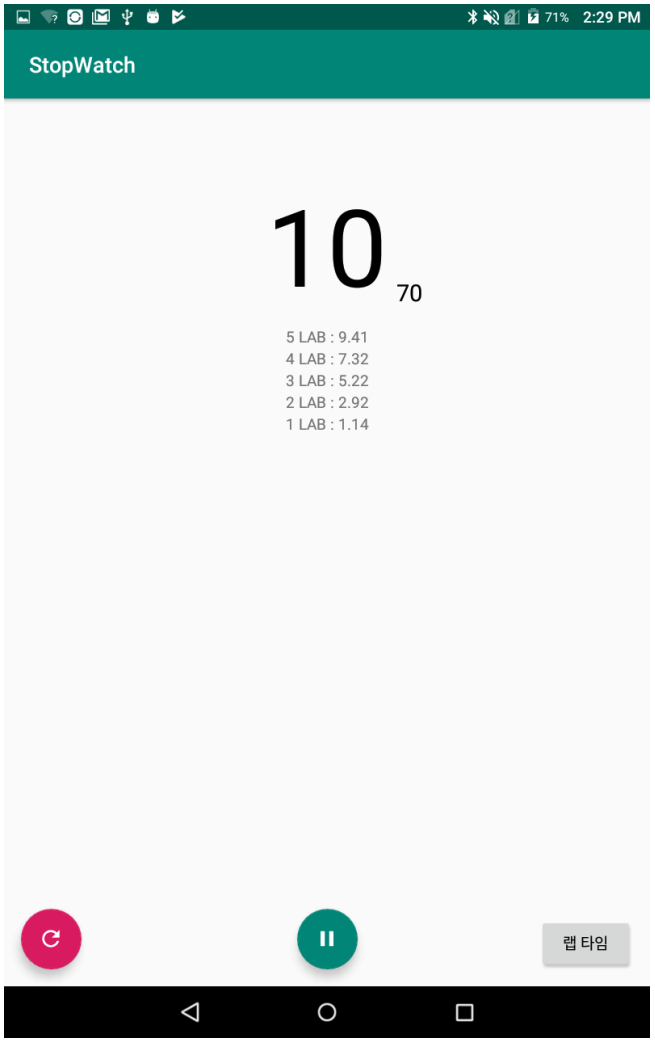
Kotlin을 이용한 Android 프로그래밍

스톱워치 앱 만들기

Contents

스톱워치 앱 만들기


71% 2:29 PM
StopWatch
0 00
랩 타임


71% 2:29 PM
StopWatch
10 70
5 LAB : 9.41
4 LAB : 7.32
3 LAB : 5.22
2 LAB : 2.92
1 LAB : 1.14
랩 타임

스톱워치 앱 만들기

- 프로젝트 명 : StopWatch
- 기능
  - 타이머의 시작, 일시정지, 초기화
  - 타이머 실행 중에 랩타임을 측정하여 표시
- 구성요소
  - Timer : 일정시간 간격으로 코드를 백그라운드 스레드에서 실행
  - runOnUiThread : 메인 스레드에서 UI를 갱신
  - ScrollView : 랩타임을 표시할 때 상하로 스크롤되는 뷰를 사용
  - FloatingActionButton : 머티리얼 디자인의 둥근 모양의 버튼
- 라이브러리 설정
  - 벡터 드로어블 하위 호환 설정 : 안드로이드 5.0 미안에서 벡터 드로어블을 지원
  - Design 라이브러리 : FloatingActionButton 등 머티리얼 디자인을 제공하는 라이브러리

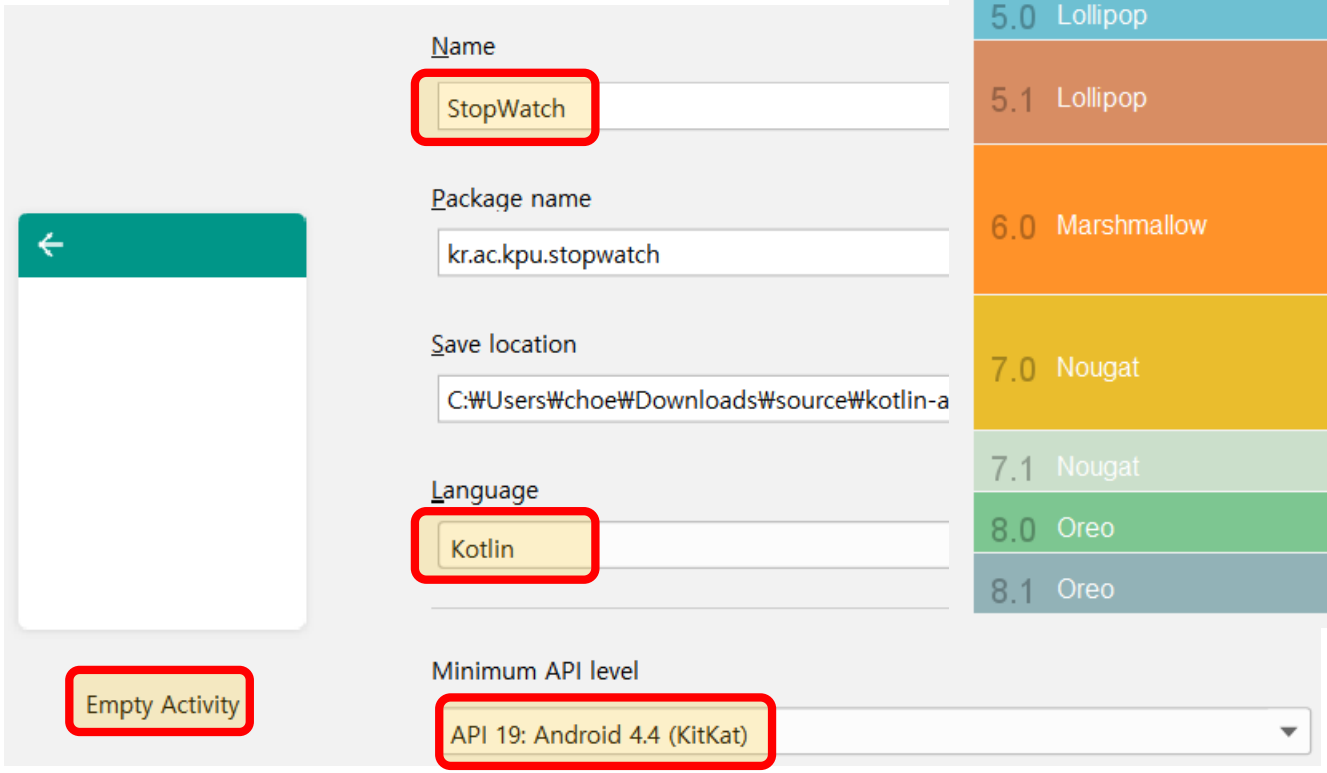
스톱워치 앱 만들기

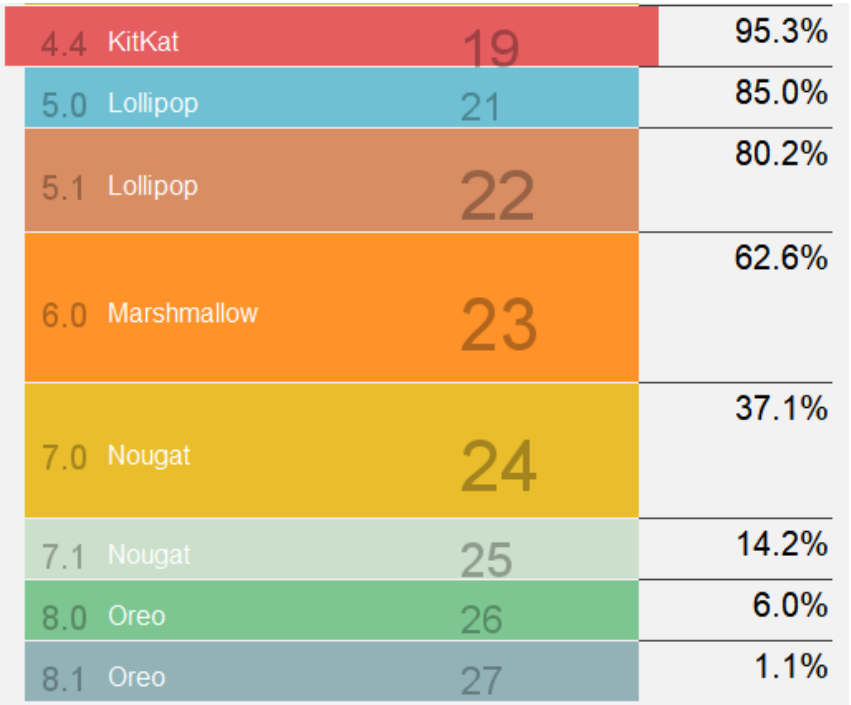
- 프로젝트 설계
  - 스톱워치 앱은 화면이 하나기 때문에 Anko 라이브러리를 사용하지 않음
  - FloatingActionButton이라는 둥근 형태의 버튼을 사용하기 때문에 벡터 드로어블 하위 호환성 처리가 필요
  - 스톱워치를 구현하기 위하여 빠르게 계산하면서 UI를 갱신
    - 각각 timer와 runOnUiThread 메서드로 구현
  - 랩타임을 누적하여 표시할 ScrollView에 동적으로 TextView를 추가
- 전체 구현 순서

1) 프로젝트 생성 및 안드로이드 설정
2) 화면 디자인
3) 타이머 구현
4) 랩 타임 기록

스톱워치 앱 만들기

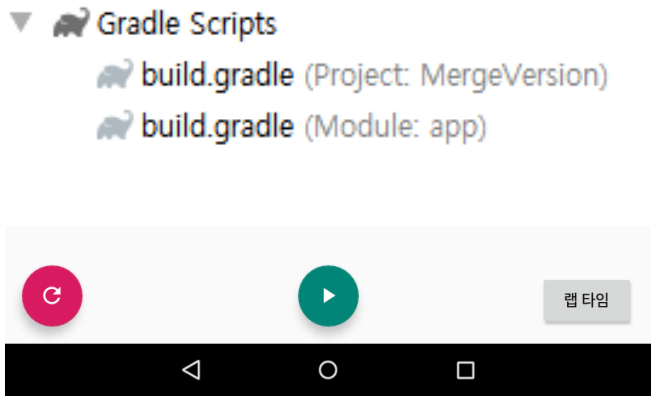
- 프로젝트 생성
  - 프로젝트 명 : StopWatch
  - minSdkVersion : 19
  - 기본 액티비티 :Empty Activity





스톱워치 앱 만들기

- 벡터 드로어블 하위호환성 설정
  - 화면에서 스톱워치를 FloatingActionButton으로 제어하고 이 버튼의 배경에 벡터 이미지를 사용할 예정
  - 안드로이드 5.0 미만의 기기에서도 벡터 이미지가 잘 표시되도록 모듈 수준의 그레이들 파일에 다음 코드를 추가한 후 싱크
    - vectorDrawables.useSupportLibrary = true

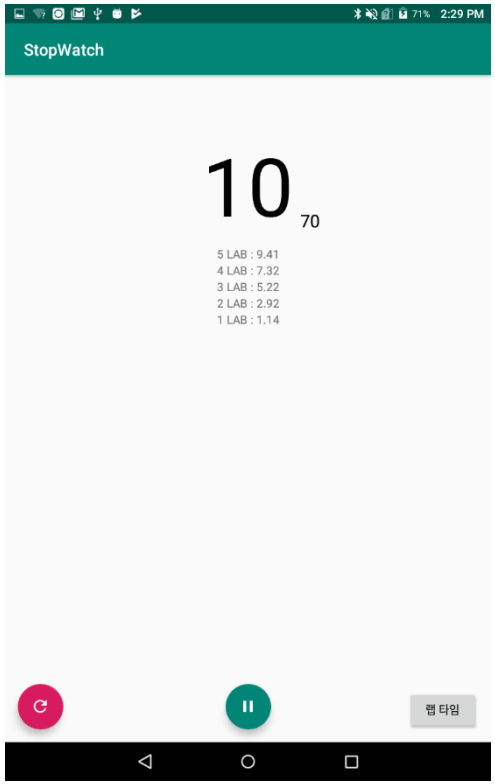



```
7   android {
8       compileSdkVersion 28
9       defaultConfig {
10          applicationId "kr.ac.kpu.bmicalculator"
11          minSdkVersion 19
12          targetSdkVersion 28
13          versionCode 1
14          versionName "1.0"
15          testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
16
17          vectorDrawables.useSupportLibrary = true
18      }
```

Gradle files have changed since last project sync. A project sync may be necessary fo. Sync Now

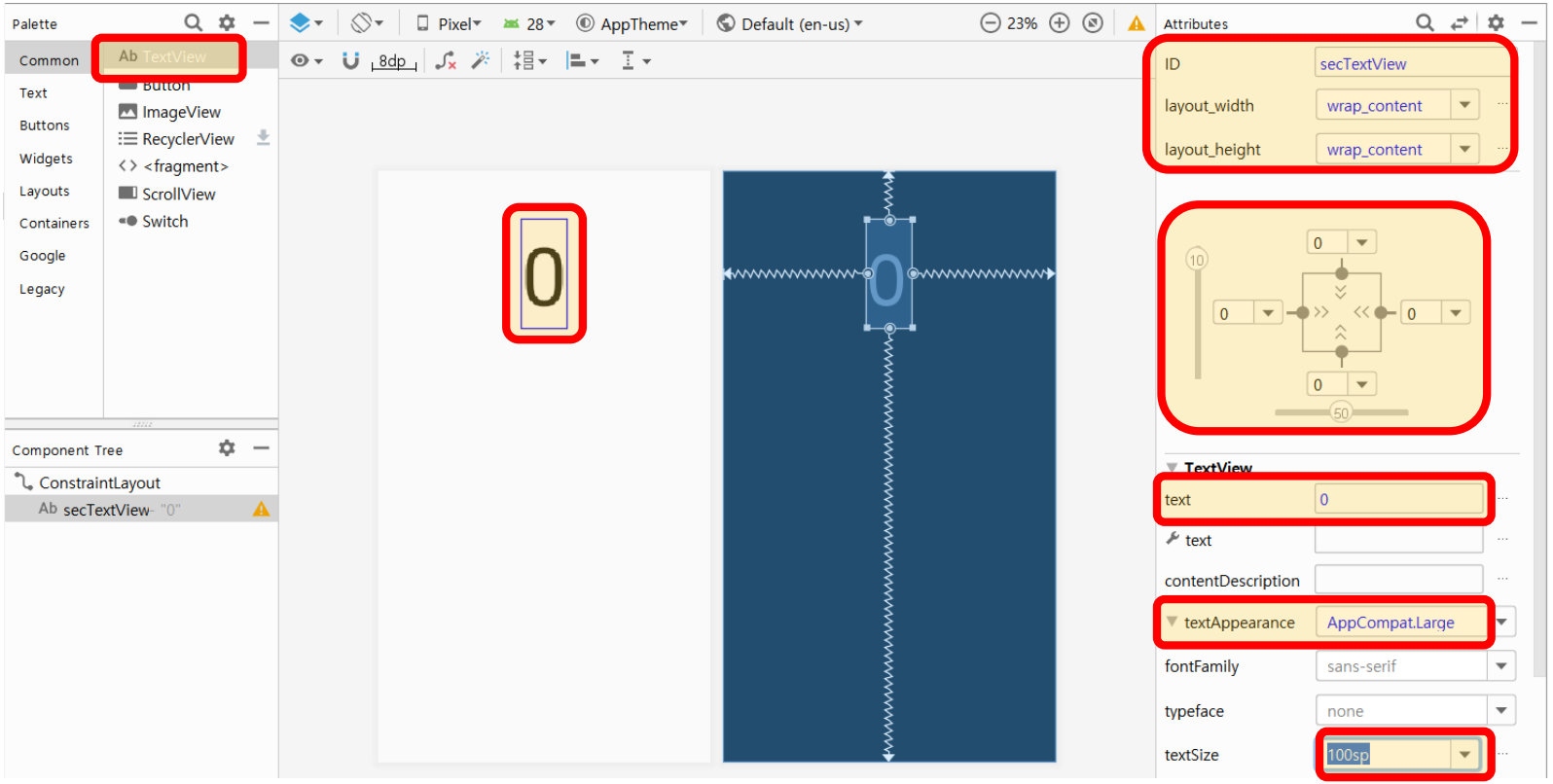
스톱워치 앱 만들기

- 화면 디자인
  - 시간을 표시하는 TextView 2개
  - 타이머를 시작 및 일시정지, 초기화하는 FloatingActionButton 2개
  - 랩 타임을 위한 버튼 1개
  - 앱 타임을 기록하고 표시할 ScrollView
- 디자인 순서
  - TextView 배치 후 정렬
  - 벡터 이미지 준비
  - FloatingActionButton 배치
  - 버튼 배치
  - ScrollView 배치



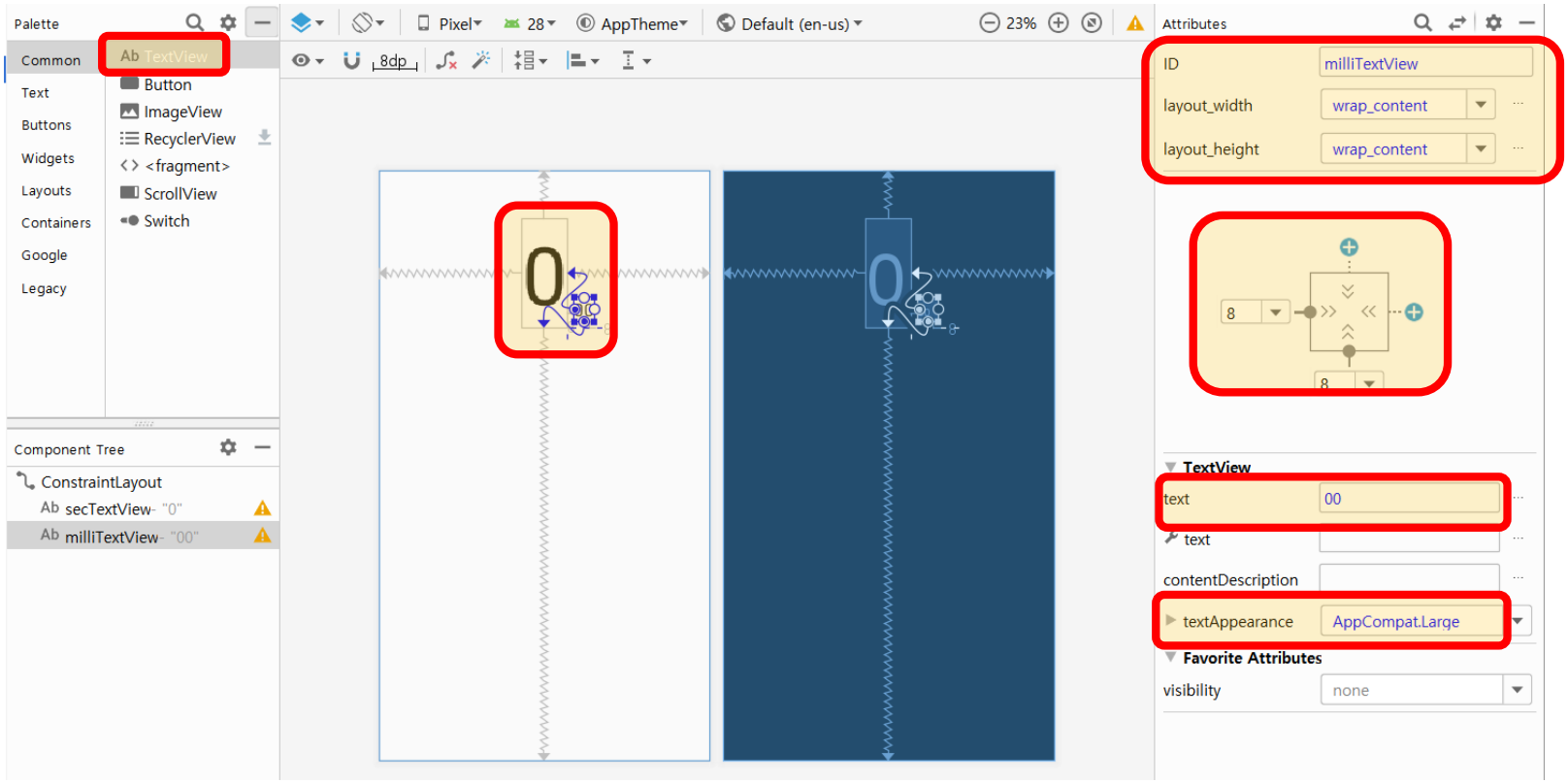
스톱워치 앱 만들기

- 시간을 표시하는 TextView 배치
  - Helloworld 텍스트뷰 삭제
  - 초를 표시할 텍스트 뷰 배치



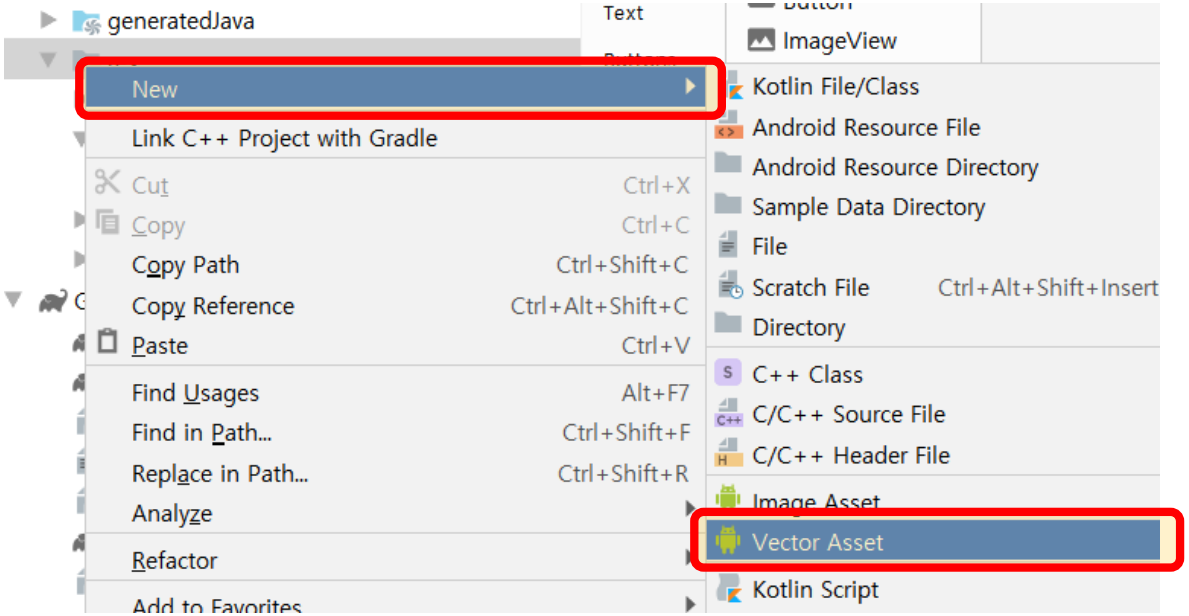
스톱워치 앱 만들기

- 시간을 표시하는 TextView 배치
  - 밀리초를 표시할 텍스트 뷰 배치



스톱워치 앱 만들기

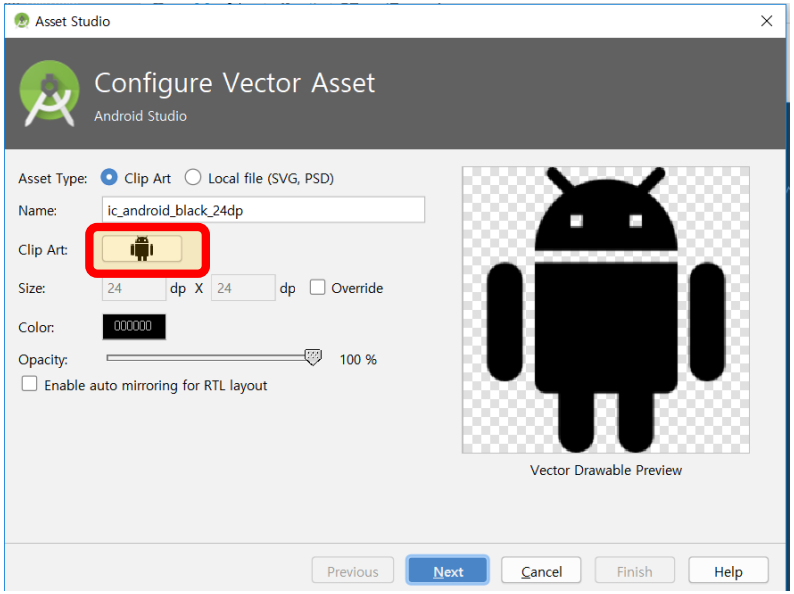
- 벡터 이미지 준비
  - 벡터 이미지는 3개가 필요 : 시작, 일시정지, 초기화
  - Asset Studio 실행
    - res에서 마우스 우 클릭 - New - Vector Asset

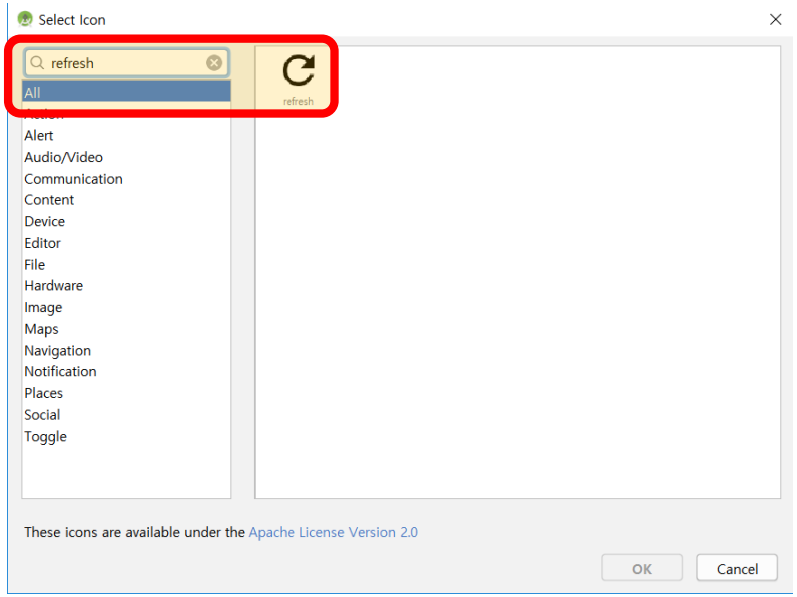


스톱워치 앱 만들기

벡터 이미지 준비

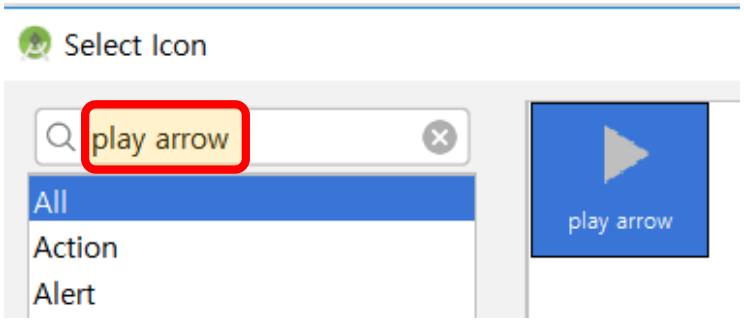
검색 기능을 이용하여 새로고침 이미지 선택

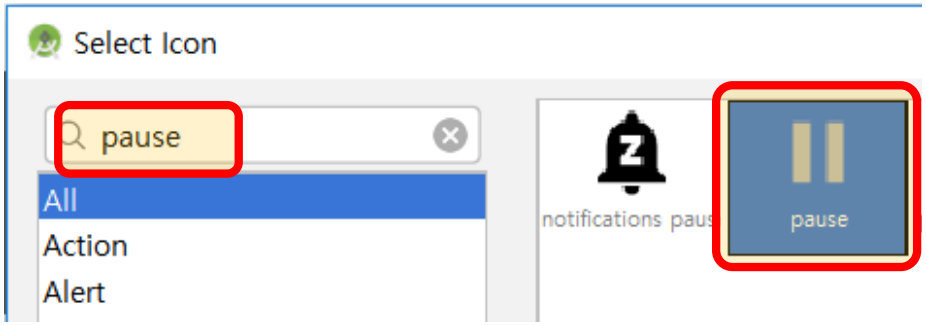




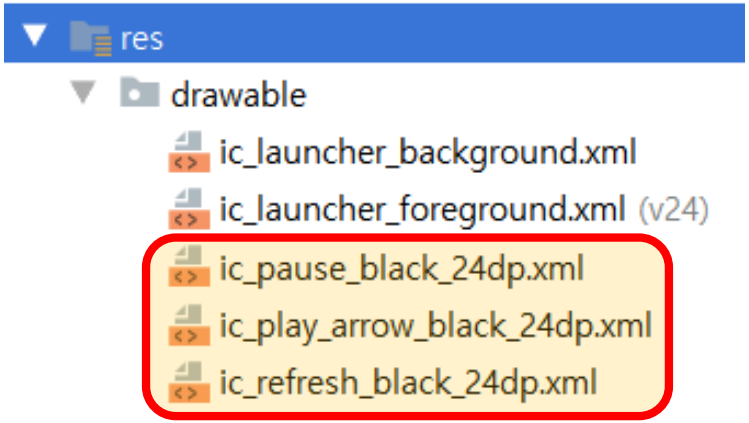
스톱워치 앱 만들기

- 벡터 이미지 준비
  - play arrow, pause 버튼 추가





  - drawable 확인



스톱워치 앱 만들기

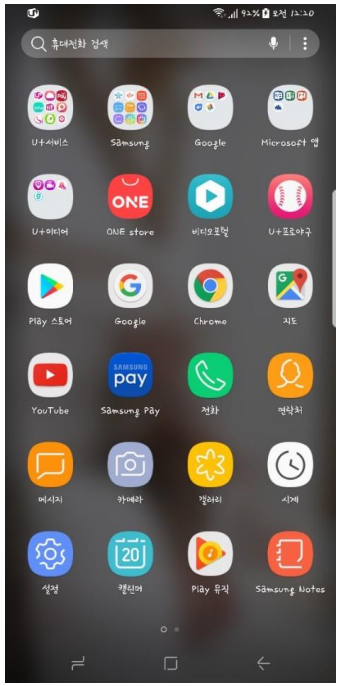
- FlaotingActionButton 개요
  - 구글의 머티리얼 디자인(Merterial Design)에서 자주 사용되는 둥근 버튼
    - 머티리얼 디자인은 안드로이드 5.0이상의 위젯 사용
  - 벡터 이미지를 사용하여 깔끔한 버튼을 표현하기에 적합

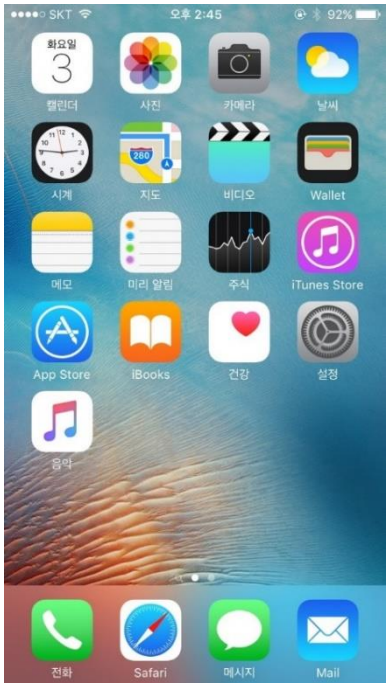
스톱워치 앱 만들기

- Merterial Design
  - 구글이 TV, 자동차, 모바일 등 다양한 분야의 기기를 생산하고 있기 때문에 안드로이드 OS를 기반으로 하는 디지털 디바이스의 종류는 점차 증가됨

스톱워치 앱 만들기

- Merterial Design
  - iOS에 비하여 안드로이드는 사용방식이나 디자인의 일관성이 낮음



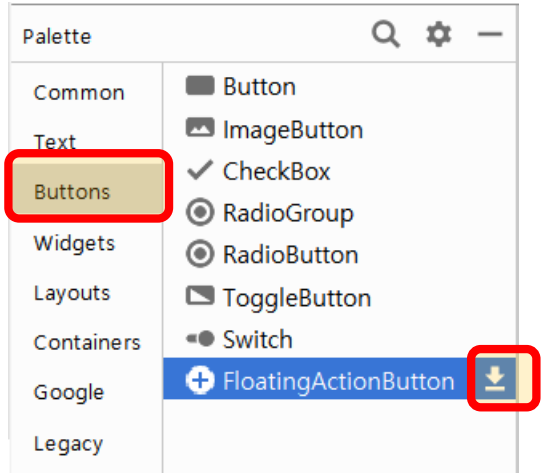


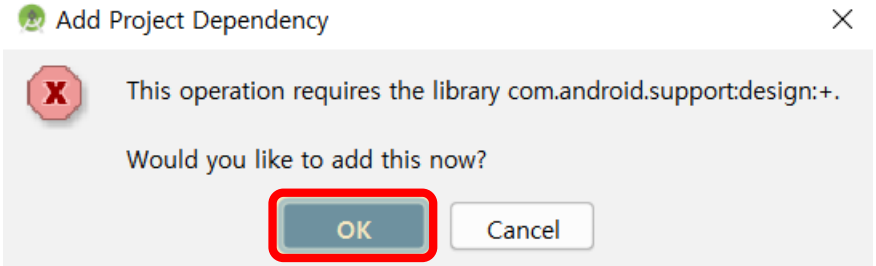
스톱워치 앱 만들기

- Merterial Design
  - 구글에서 안드로이드 기기 사용자들에게 일관된 경험을 느끼게 하기 위하여 발표한 디자인 가이드
    - 화면 디자인, 움직임, 상호작용 설계에 대한 디자인 가이드라인
    - 앱을 개발할 시 안드로이드 개발자 사이트에 있는 머티리얼 디자인 규격을 따라야 함
    - 제조사나 통신사 마다 UI/UX가 파편화 되어 있었으나 머티리얼 디자인을 통해 디자인의 일관성을 강제할 수 있음

스톱워치 앱 만들기

- 타이머 시작을 위한 FloatingActionButton 작성
  - 기본 제공되는 컴포넌트가 아니므로 다운로드 해야함





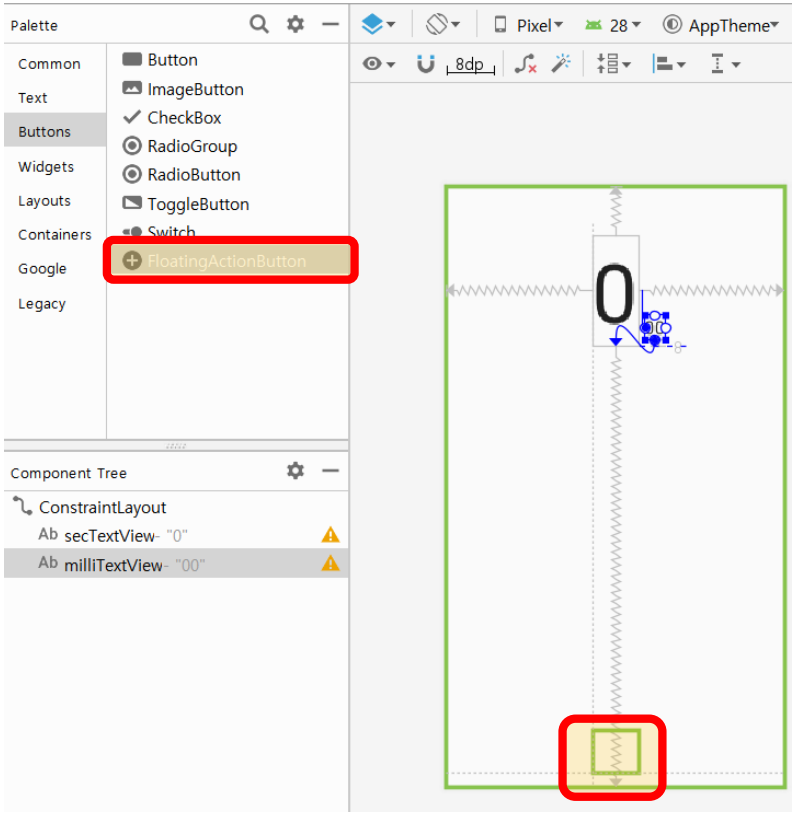
  - 다이어로그에서 OK를 클릭하면 자동으로 design 라이브러리를 추가하고 싱크

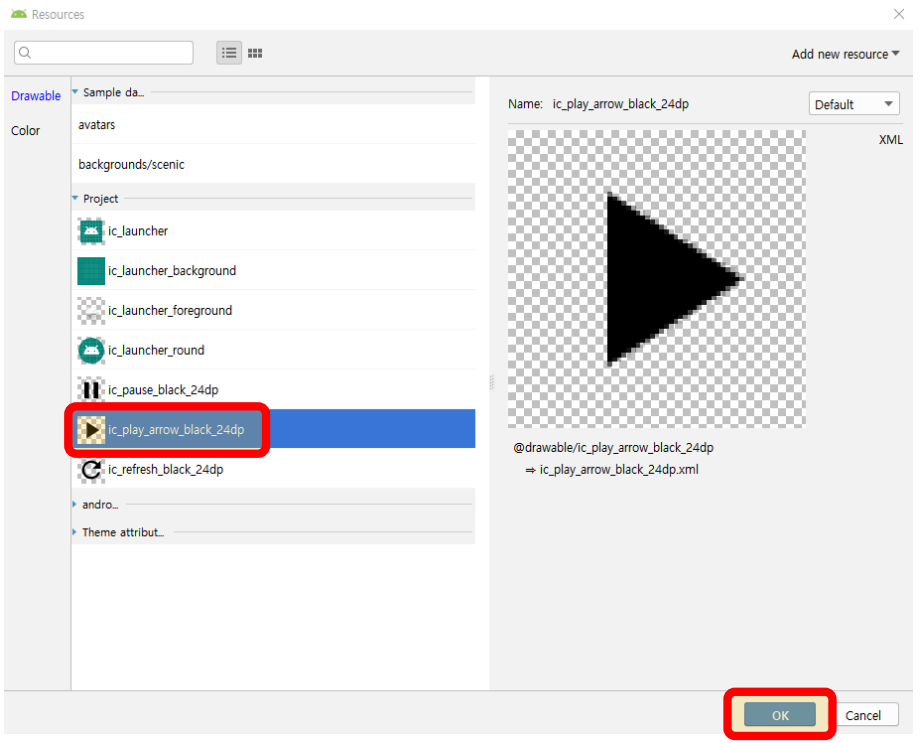
스톱워치 앱 만들기

- 타이머 시작을 위한 FloatingActionButton 작성
  - 라이브러리가 추가되었는지 확인

스톱워치 앱 만들기

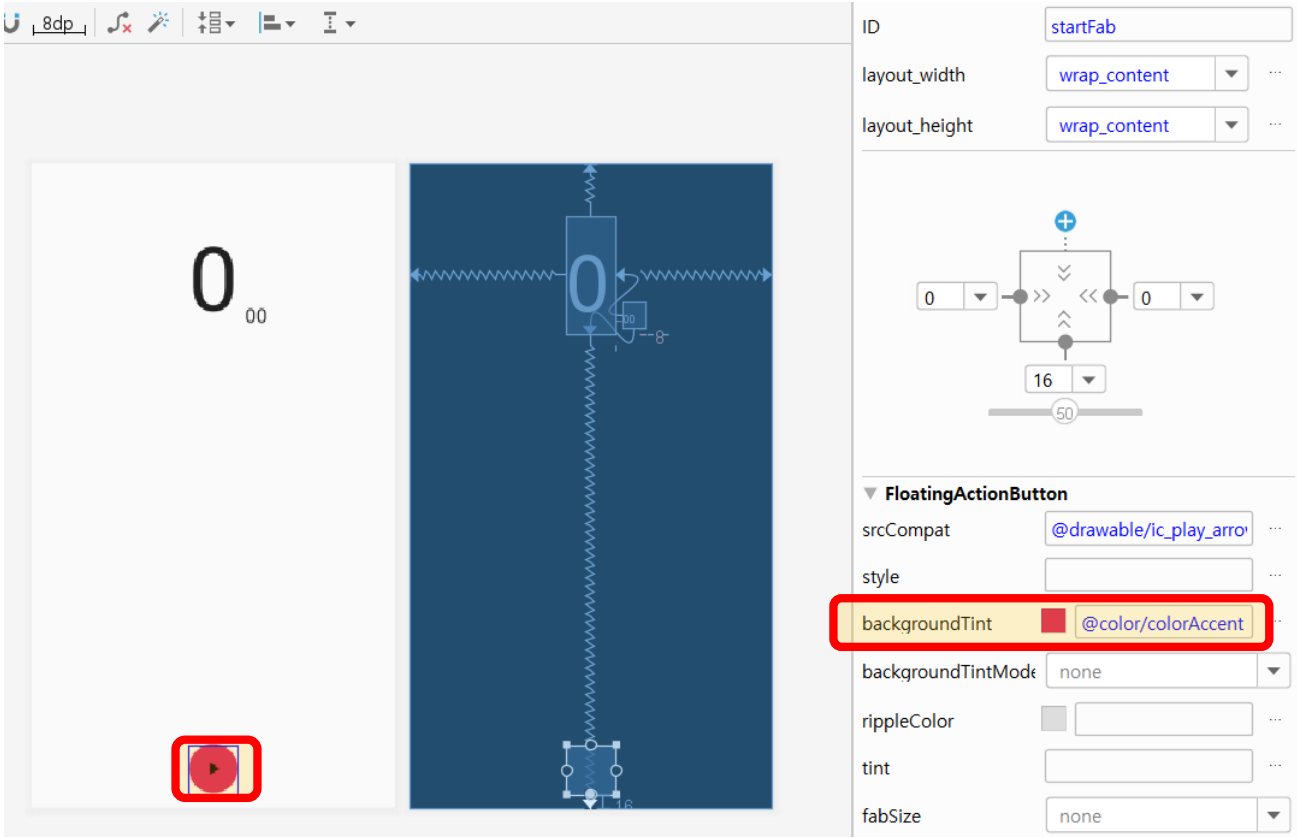
- 타이머 시작을 위한 FloatingActionButton 작성
  - 버튼을 화면의 하단에 추가한 후 이미지 선택





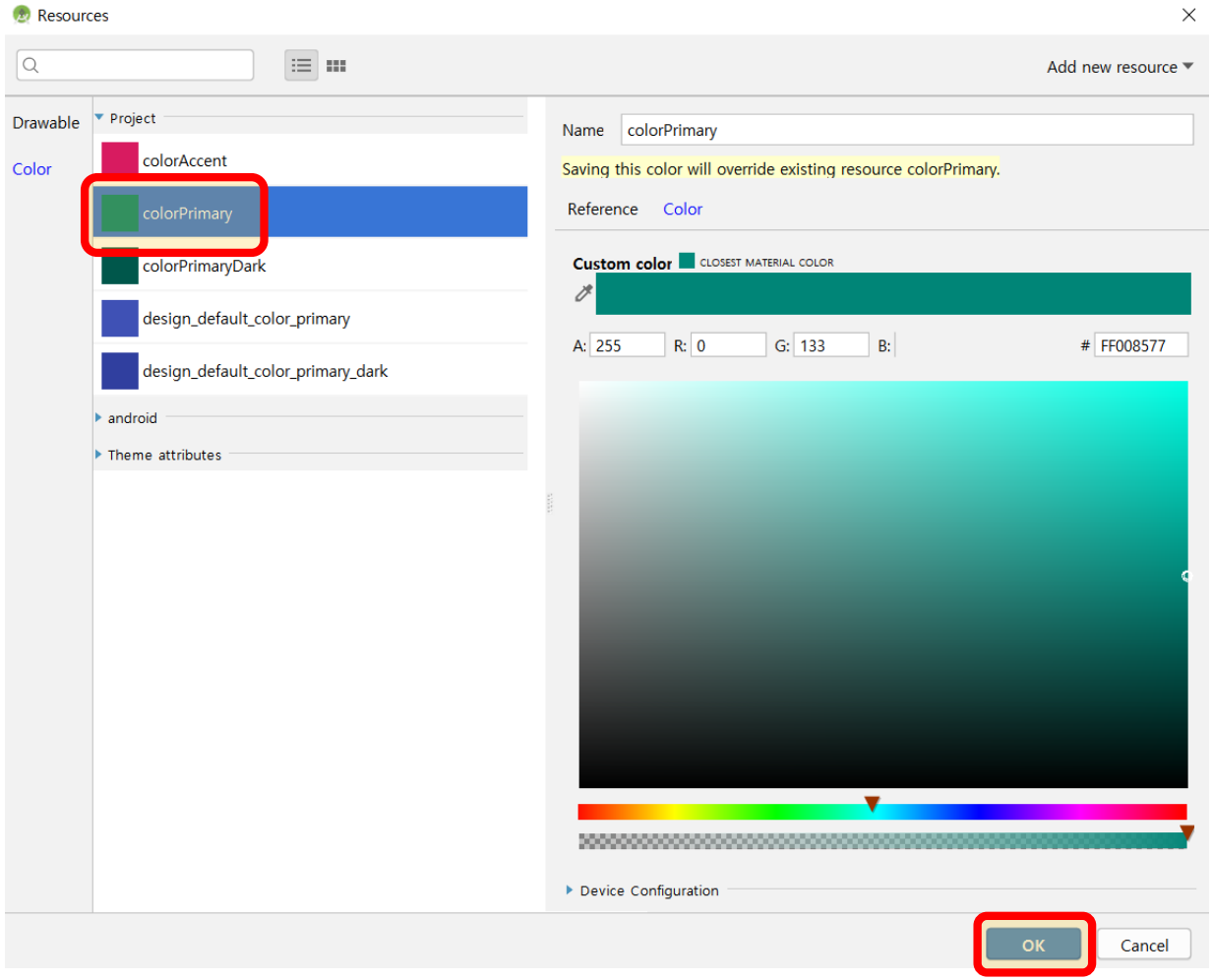
스톱워치 앱 만들기

- 타이머 시작을 위한 FloatingActionButton 작성
  - 버튼은 배경색이 기본 테마의 색상 중에 하나인 colorAccent로 자동 지정
  - 배경색을 변경하려면 속성 창에서 backgroundTint 값을 변경



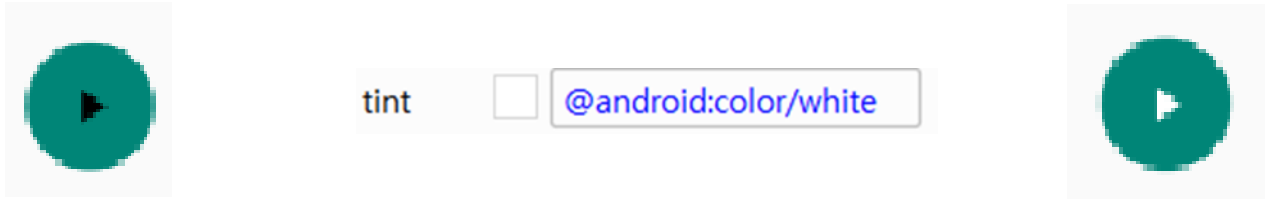
스톱워치 앱 만들기

- 타이머 시작을 위한 FloatingActionButton 작성
  - colorPrimary로 변경



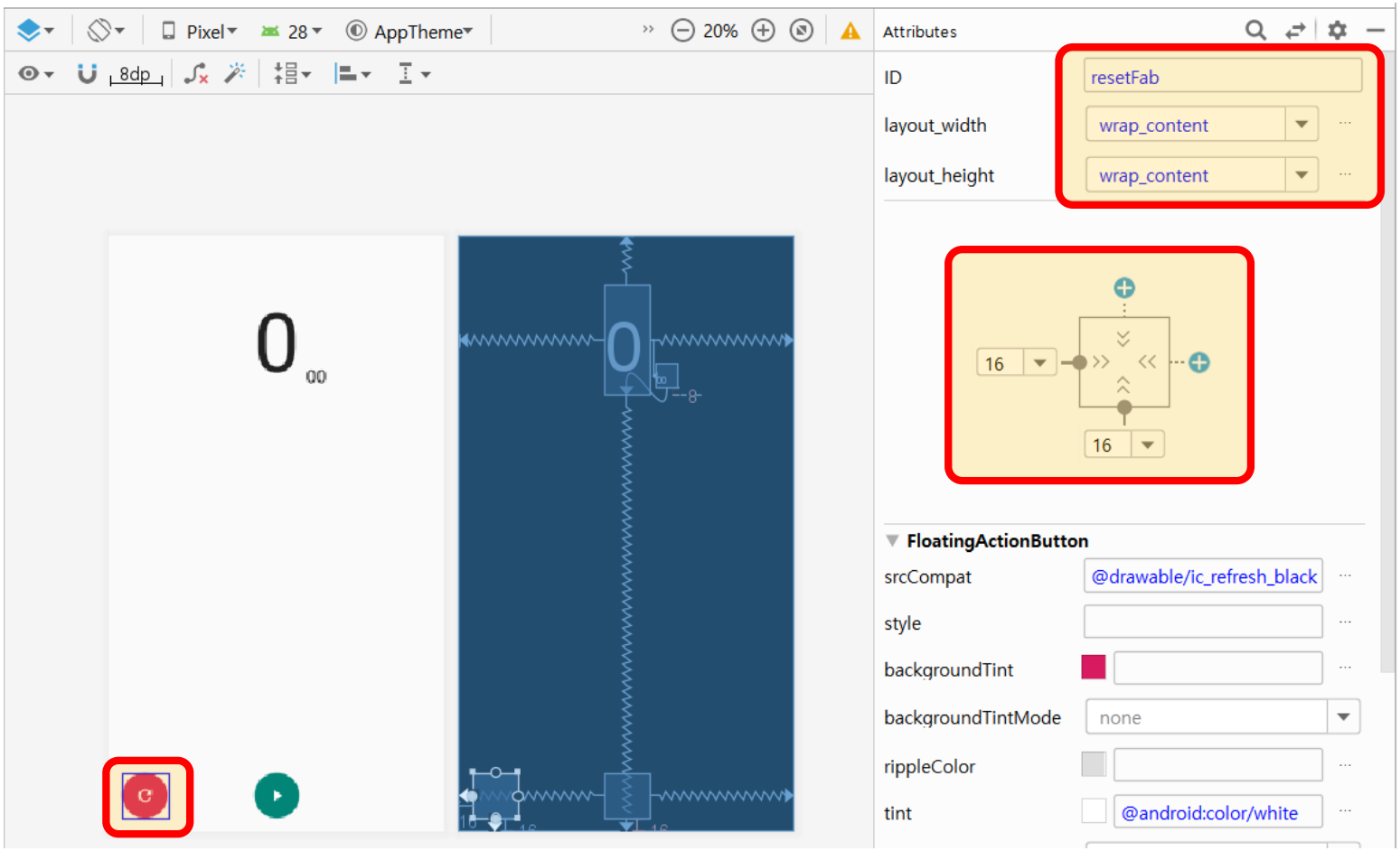
스톱워치 앱 만들기

- 타이머 시작을 위한 FloatingActionButton 작성
  - 배경색이 어둡고 벡터 이미지가 검정색이라 잘 보이지가 않기 때문에 tint 속성을 white로 변경



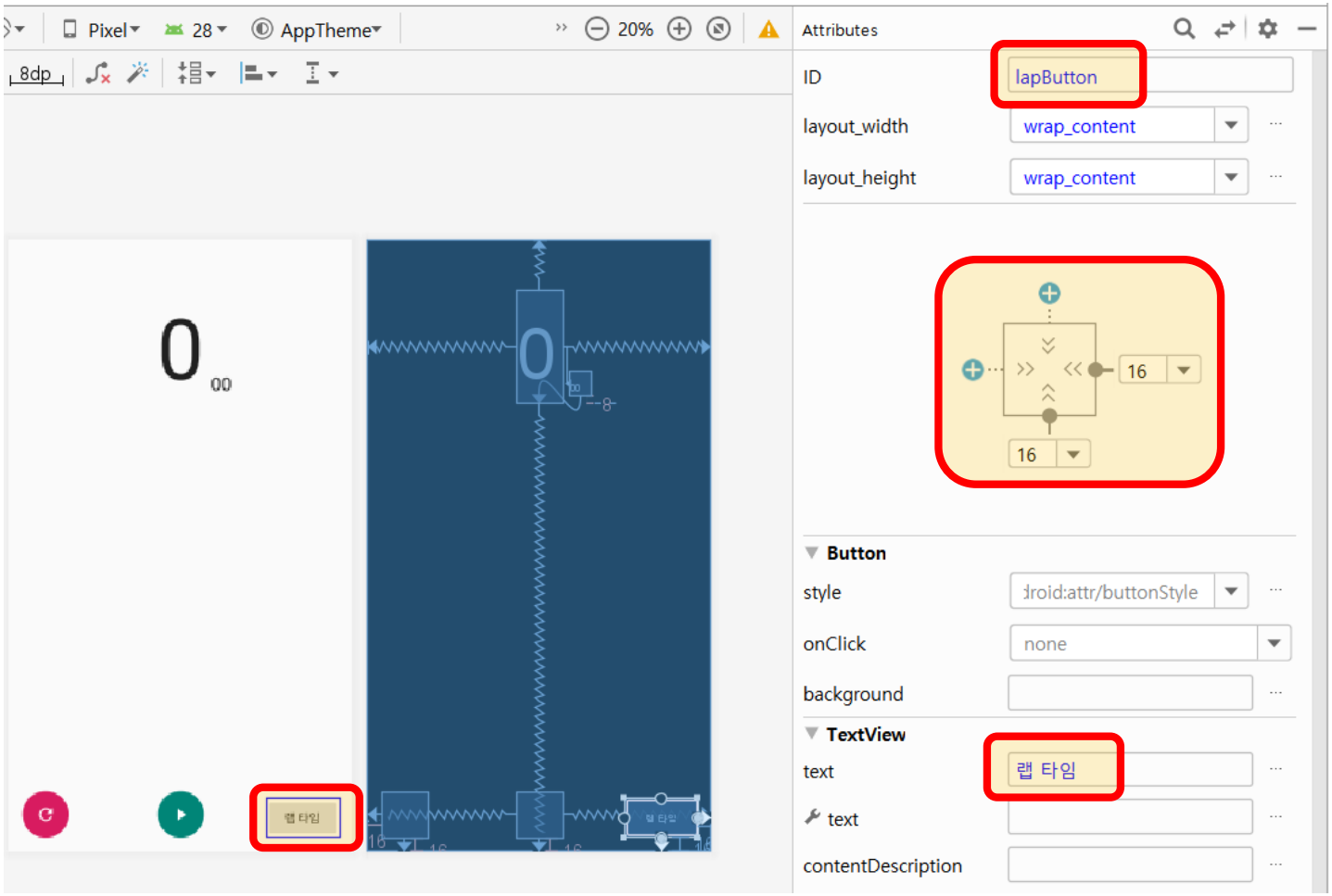
스톱워치 앱 만들기

- 타이머 초기화를 위한 FloatingActionButton 작성



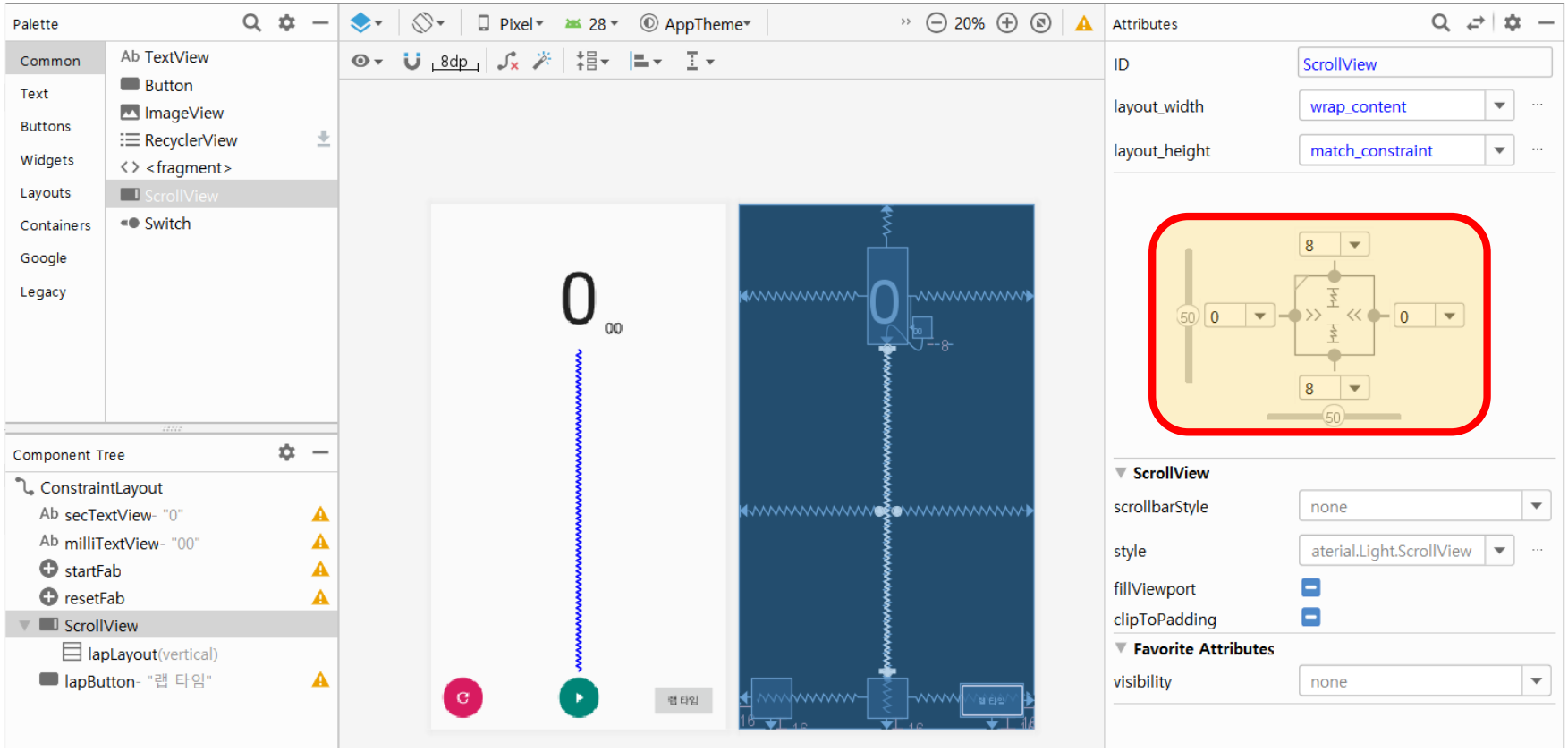
스톱워치 앱 만들기

- 랩 타임 기록을 위한 버튼 작성



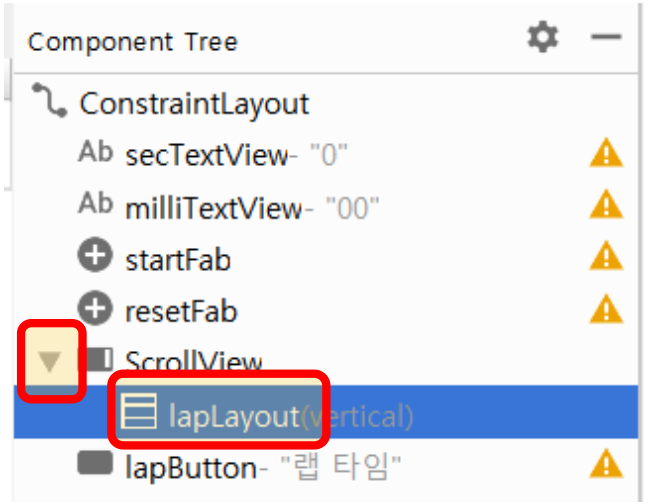
스톱워치 앱 만들기

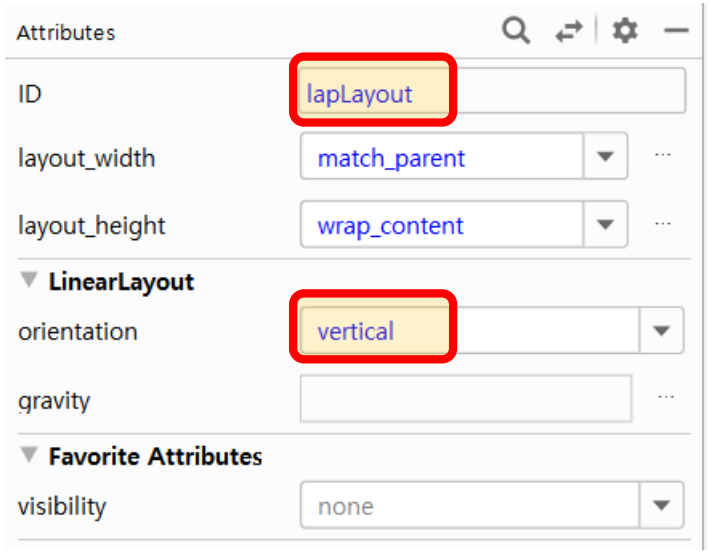
- 랩 타임 기록을 표시하는 ScrollView 배치
  - ScrollView내부에 랩 타임 버튼을 누를 때마다 시간이 차곡차곡 쌓이는 LinearLayout 구성
  - match_parent와 0dp는 같은 의미



스톱워치 앱 만들기

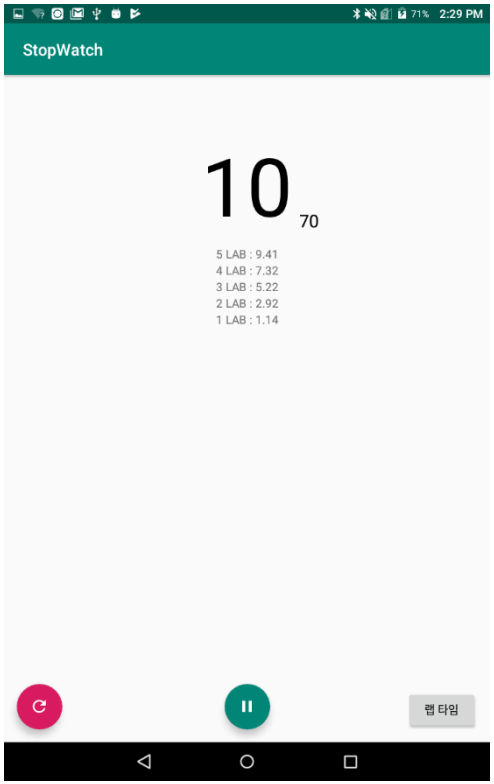
- 랩 타임 기록을 표시하는 ScrollView 배치
  - 스크롤 뷰 내부에 있는 리니어레이아웃 ID 수정





스톱워치 앱 만들기

- 타이머 구현
  - 기본 동작은 시작 버튼을 누르면 타이머가 동작하고 이미지는 일시정지로 교체
  - 다시 버튼을 누르면 타이머는 일시 정지되며 이미지는 시작 이미지로 교체됨
  - 구현 순서
    - Timer 구현
    - 타이머 시작
    - 타이머 일시정지
    - 버튼에 이벤트 연결



스톱워치 앱 만들기

- Timer 사용 방법
  - 안드로이드에는 UI를 조작하는 메인 스레드와 오래 걸리는 작업을 보이지 않는 곳에서 처리하는 워크 스레드가 존재하며 위 코드는 워커 스레드
    - 워커 스레드 코드

```
timer(period = 1000){ this: TimerTask

}
```

    - 메인 스레드 코드

```
timer(period = 1000){ this: TimerTask
    runOnUiThread{

    }
}
```

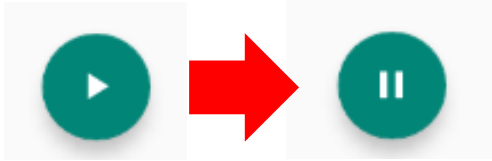
스톱워치 앱 만들기

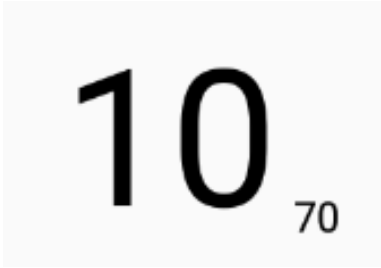
▶타이머 시작 구현

▶MainActivity.kt 파일에 타이머를 시작할 때 호출할 start()메서드를 작성

```
private var time = 0  //시간을 계산할 변수를 0으로 초기화
private var timerTask: Timer? = null //Timer 타입의 timerTask를 null을 허용하도록 선언(Timer 객체를 변수에 저장)
                                     //추후에 timer를 취소하려면 timer를 실행하고 반환되는 값을 저장할 필요가 있음
private fun start() {
    startFab.setImageResource(R.drawable.ic_pause_black_24dp)  //일시정지 이미지로 변경

    timerTask = timer(period = 10) { this: TimerTask  //0.01초마다 작업하는 timer
        time++ //time 변수를 1씩 증가
        val sec = time / 100
        val milli = time % 100
        runOnUiThread {   //UI 갱신
            secTextView.text = "$sec"
            milliTextView.text = "$milli"
        }
    }
}
```





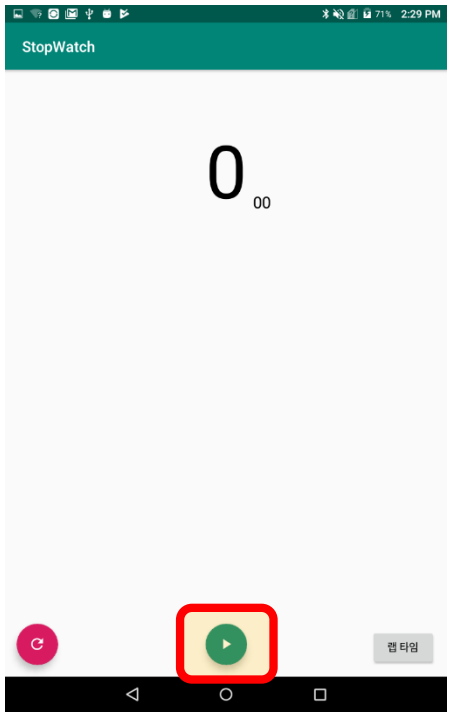
스톱워치 앱 만들기

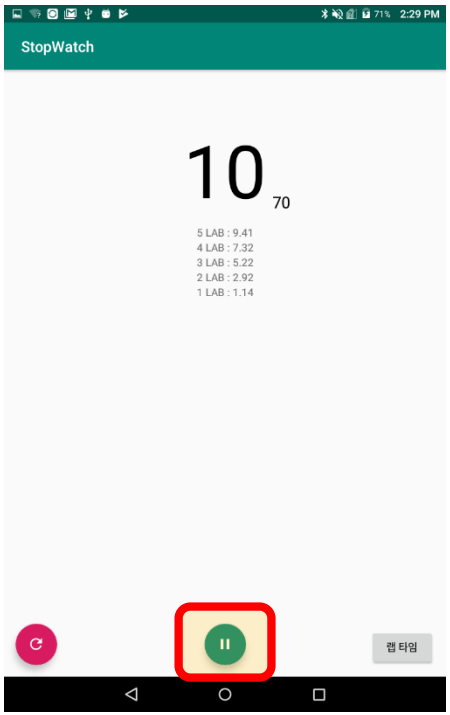
타이머 일시정지 구현

MainActivity.kt 파일에 일시정지 메서드인 pause()메서드를 작성

```
private fun pause() {
    startFab.setImageResource(R.drawable.ic_play_arrow_black_24dp) //시작 이미지로 교체


    timerTask?.cancel() //타이머 취소
}
```





스톱워치 앱 만들기

- 버튼에 이벤트 연결
  - onCreate()메소드 내부에 시작과 일시정지 이벤트를 구현하고 외부에 flag 역할을 할 isRunning 변수 선언

```
private var time = 0
private var timerTask: Timer? = null
private var isRunning = false

startFab.setOnClickListener { it: View!
    isRunning = !isRunning

    if (isRunning) {
        start()
    } else {
        pause()
    }
}
```

스톱워치 앱 만들기

- 랩 타임 기록하기
  - 100미터 달리기용 랩 타임을 기록하는 기능을 추가
  - 타이머의 초기화 구현
  - 구현 순서
    - 동적으로 리니어 레이아웃에 뷰 추가
    - 랩 타입 표시
    - 타이머 초기화 구현

스톱워치 앱 만들기

- 동적으로 LinearLayout에 뷰 추가하기
  - 스크롤 뷰의 내부에 있는 리니어 레이아웃은 수직(Vertical)으로 자식 뷰를 추가
  - 코틀린에서 addView()메소드를 이용하여 리니어 레이아웃 내부에 동적으로 뷰 추가
  - 리니어 레이아웃에 텍스트 뷰 추가하는 예제

```
val textView = TextView( context: this)
textView.text = "글자"
lapLayout.addView(textView)
```

    - 위 코드는 TextView 객체를 동적으로 생성하여 LinearLayout의 위에서 부터 아래로 쌓으면서 보여짐
      - 그러므로 최근 랩타임이 맨 위로 오도록 해야함
    - addView()의 두번째 인자에 index 번호 0을 넣어서 추가될 때 마다 항상 맨 위로 오도록 함
      - addView(textView, 0)
      - 인덱스 번호는 위젯을 포함한 번호이고 위젯 개수보다 많을 시 오류발생

스톱워치 앱 만들기

랩 타임 표시하기

- 랩 타임을 기록하고 화면에 표시하는 메서드인 recordLapTime()메서드 작성

```
private var lap = 1  //추가되는 랩 타임 번호

private fun recordLapTime() {
    val lapTime = this.time    //현재 시간 저장
    val textView = TextView( context: this) //텍스트 뷰 생성
    textView.text = "$lap LAB : ${lapTime / 100} .${lapTime % 100} "//1 LAB 5.35 처럼 출력되도록 설정

    // 맨 위에 랩타임 추가
    lapLayout.addView(textView,  index: 0) //리니어 레이아웃에 랩 타임 추가
    lap++ //랩 타임 번호 증가
}
```

스톱워치 앱 만들기

- 랩 타임 표시하기
  - 랩 타임 버튼에 이벤트를 연결하도록 onCreate()메소드 내부에 구현

```
lapButton.setOnClickListener { it: View!
    recordLapTime()
}
```

스톱워치 앱 만들기

타이머 초기화 구현

타이머를 초기화하는 reset()메서드를 구현

```
private fun reset() {
    timerTask?.cancel()   //실행중인 타이머 취소

    // 모든 변수 초기화    //초기화이기 때문에 모든 변수와 화면에 보이는 모든 정보를 초기화
    time = 0
    isRunning = false
    startFab.setImageResource(R.drawable.ic_play_arrow_black_24dp)
    secTextView.text = "0"
    milliTextView.text = "00"

    // 모든 랩타임을 제거
    lapLayout.removeAllViews()
    lap = 1
}
```

스톱워치 앱 만들기

- 타이머 초기화 구현
  - 초기화 버튼에 reset() 연결

```
resetFab.setOnClickListener { it: View!
    reset()
}
```

Q & A