3 w.

Custom View

Ivy

이벤트 처리

안드로이드 이벤트

- 사용자 액션에 대한 응답 → 입력 이벤트 (input event)
  - ex. 스크린과 상호작용
- 이벤트 큐(event queue)
  - FIFO
  - 이벤트 알림, 이벤트에 관한 정보
- 이벤트 처리
  - event listener
  - 특정 타입의 이벤트 응답처리

→ 버튼의 클릭 이벤트 처리 과정을 설명할 수 있나요?

주요 Event Listener와 Callback 메서드

Event Listener	역할	이벤트 처리
OnClickListener	View가 차지하는 영역을 사용자가 터치하면 발생하는 클릭 이벤트 감지	onClick
OnLongClickListener	View가 차지하는 영역을 사용자가 길게 터치하면 발생하는 클릭 이벤트 감지	onLongClick
OnTouchListener	단일/다중 터치, 제스처를 포함한 화면에서 발생하는 모든 접촉 감지	onTouch
OnFocusChangeListener	현재의 View에서 Focus가 변경되는 것을 감지	onFocusChange
OnKeyListener	device의 key가 눌러진 것을 감지	onKey

이벤트 소비

```
Interface definition for a callback to be invoked when a view is clicked.
public interface OnClickListener {
    Called when a view has been clicked.
    Params: v – The view that was clicked.
    void onClick(View v);
}
```

```
btnPress.setOnClickListener { it: View!
    textLearning.text = "Click"
}
btnPress.setOnLongClickListener { it: View!
    textLearning.text = "Long Click"
    return@setOnLongClickListener true
}
```

```
public interface OnLongClickListener {
    Called when a view has been clicked and held.
    Params: v – The view that was clicked and held.
    Returns: true if the callback consumed the long click, false otherwise.
    boolean onLongClick(View v);
}
```

onLongClick의 반환값은 어떤 것을 의미할까?

```
btnPress.setOnClickListener { it: View!
    textLearning.text = "Click"
}
btnPress.setOnLongClickListener { it: View!
    textLearning.text = "Long Click"
    return@setOnLongClickListener false
}
```

터치 이벤트

MotionEvent

- 터치 위치
- 액션 타입 식별

1) 단일 터치

2) 다중 터치

POINTER는 어떤 것을 의미할까?

다중 터치 정보 얻기

- 터치 이벤트 타입
- 터치 좌표
- 터치 개수

포인터

- 다중 터치가 발생할 때의 접촉 지점

```
containerSetting.setOnTouchListener { v, motionEvent ->
    handleTouch(motionEvent)
    return@setOnTouchListener true
}
```

- 포인터 인덱스 : 포인터 중, 현재 이벤트를 받고 있는 포인터
- 포인터 ID : 제스처 중인 손가락의 고유 ID

→ 더블탭 처리기를 만든다면?

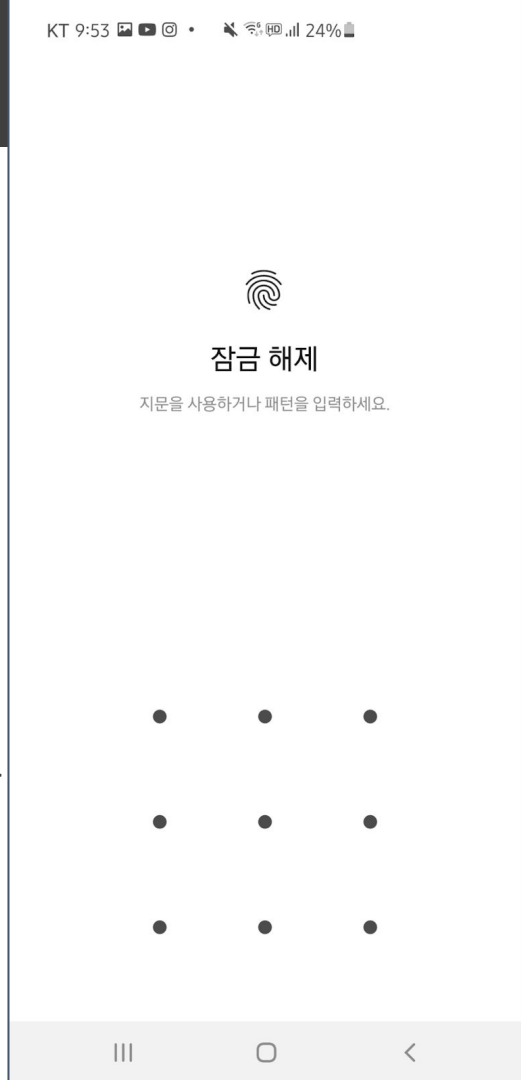
Custom View 생성

언제 Custom View 구현하는가

Android가 제공하는 View, ViewGroup

- Button, TextView, EditText 등
- ConstraintLayout, LinearLayout 등

1. 위젯, 레이아웃으로는 UI 요구사항을 만족시킬 수 없을 때
2. 더 정밀한 제어가 필요한 경우
   - 사용자 입력에 따라 제어가 필요한 경우 (컨트롤러, 화면 잠금
3. 여러 화면에서 사용되는 경우



절차

1. super class 선택하고, sub class를 만든다.
   a. 단순 커스텀뷰 : 자식 뷰 없이 단순한 구조를 갖는 경우
   b. 복합 커스텀뷰 : 여러 자식 뷰로 구성해야 하는 구조를 갖는 경우

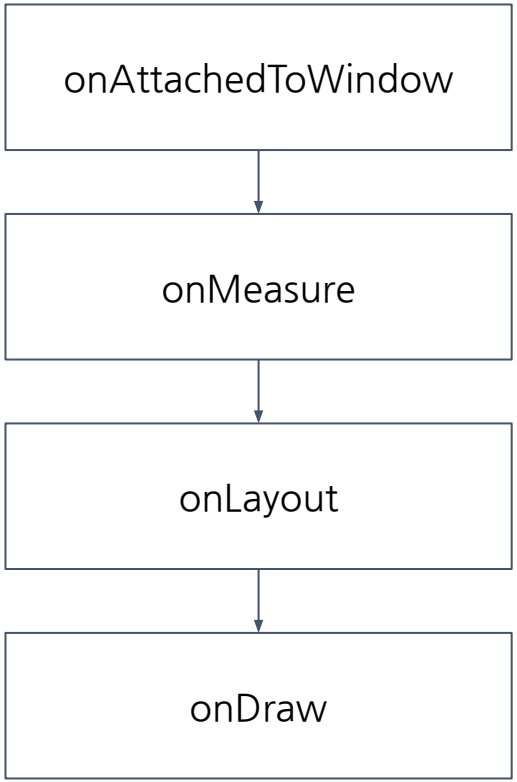
2. super class의 함수를 override 해서 커스터마이징한다.

단순 Custom View

UI Widget의 subclass 만들기

- UI widget(TextView, Button 등)의 subclass를 만든다
- UI widget이 제공하지 않는 View를 만들려면 View의 subclass를 생성한다

View의 주요 생명주기



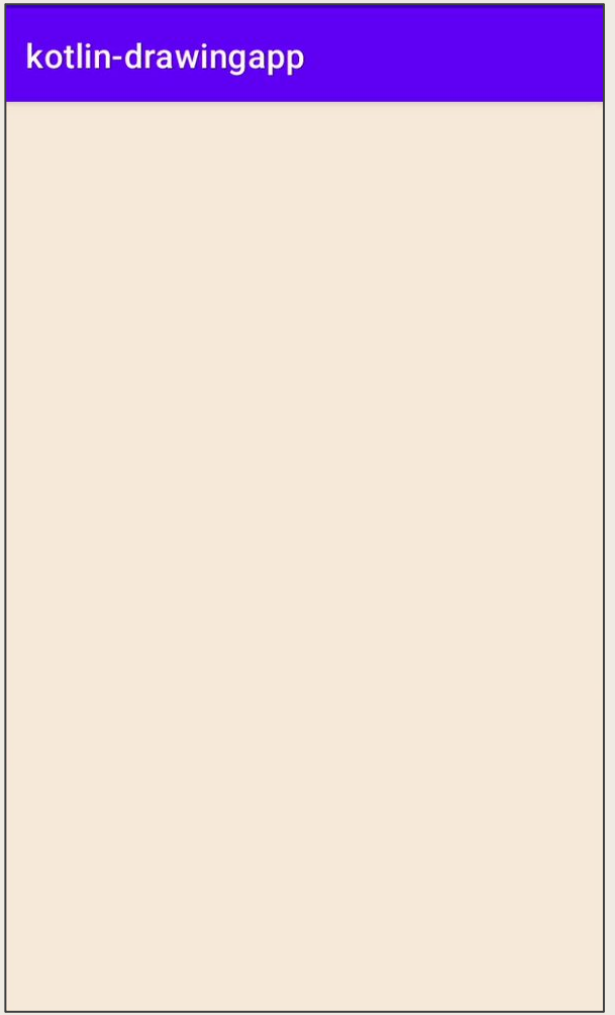
처음 view가 추가되었을 때 호출

View의 사이즈가 결정되면 호출

View의 사이즈, 위치 적용시 호출

Canvas를 사용해 View를 그릴 때 호출

드래그해서
사각형 그리는
커스텀뷰 만들어보기



onSizeChanged - View size 계산

- 처음 layout이 inflate 될 때 호출
- view size가 변경될 때 호출

```
override fun onSizeChanged(width: Int, height: Int, oldWidth: Int, oldHeight: Int) {
    radius = (min(width, height) / 2.0 * 0.8).toFloat()
}
```

onDraw - Canvas 사용해 View 그리기

- onDraw 메서드는 screen이 갱신될때마다 1초에도 여러번 호출된다. Visual glitches를 피하기 위해 onDraw에서는 최대한 작은 작업을 처리해야 한다.

  → **allocation은 garbage collection을 일으켜 버벅이는 현상 발생 가능**

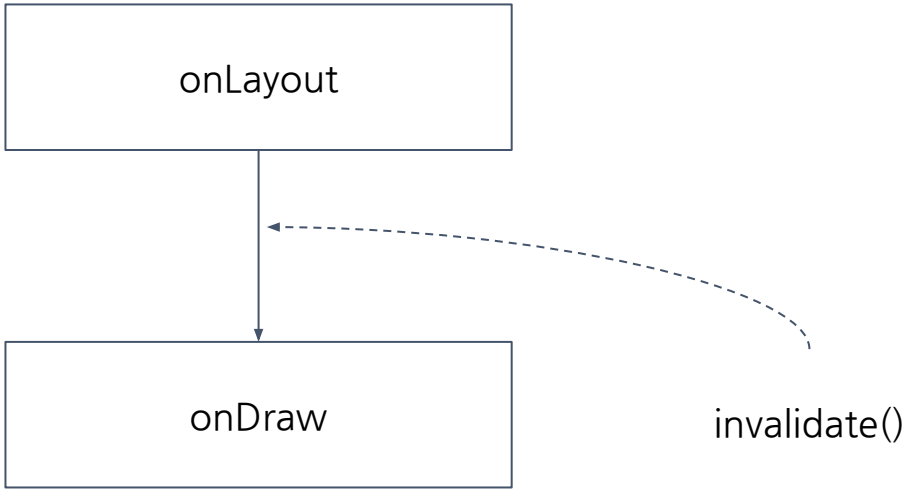
- drawText, drawRect, drawOval, drawArc, drawBitmap
- setTypeface
- setColor, setStyle

Canvas

- 2D drawing surface
- Paint 객체로 style과 color 설정
- 커스텀뷰를 생성하는 일반적인 패턴
  - + override onDraw, onSizeChanged

User Interaction 처리

- isClickable = true
- invalidate() : User Interaction에 따른 View의 변경이 필요한 경우, invalidate를 호출하면 onDraw를 호출해 view를 다시 그린다.

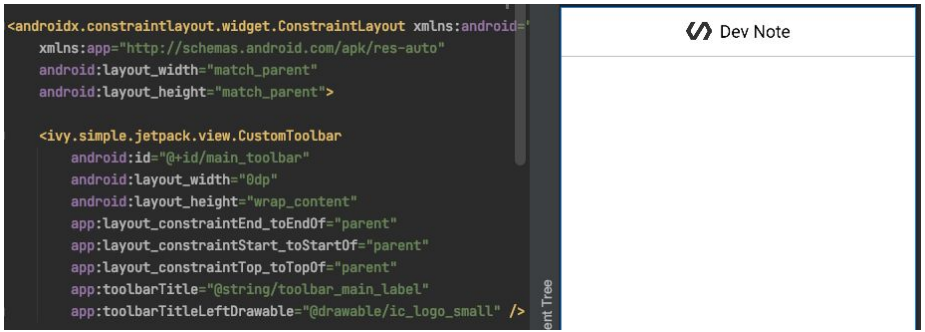


복합 Custom View

Layout의 subclass 만들기

[요구사항]

- Toolbar의 중앙에는 텍스트와 로고가 항상 함께 보여야 한다
- 우상단 메뉴는 화면마다 갯수가 다르고, margin 값도 다르다



[고려사항]

1. 적합한 Super class 를 선택합니다.

2. Super class의 주요 함수를 override 해서 커스터마이징하고, 의도한대로 동작하는지 확인합니다.

[최소조건]

Context, AttributeSet 를 생성자로 제공

→ 제공하지 않으면, Layout XML 파일로부터 인스턴스를 만들 수 없습니다.

Custom 속성 정의

values/attrs.xml

```xml
<resources>
    <declare-styleable name="CustomToolbar">
        <attr name="toolbarTitle" format="reference" />
        <attr name="toolbarTitleLeftDrawable" format="reference" />
    </declare-styleable>
</resources>
```

- Custom 속성 정의

```xml
<ivy.simple.jetpack.view.CustomToolbar
    android:id="@+id/main_toolbar"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:toolbarTitle="@string/toolbar_main_label"
    app:toolbarTitleLeftDrawable="@drawable/ic_logo_small" />
```

- XML 레이아웃에서 속성 값 지정

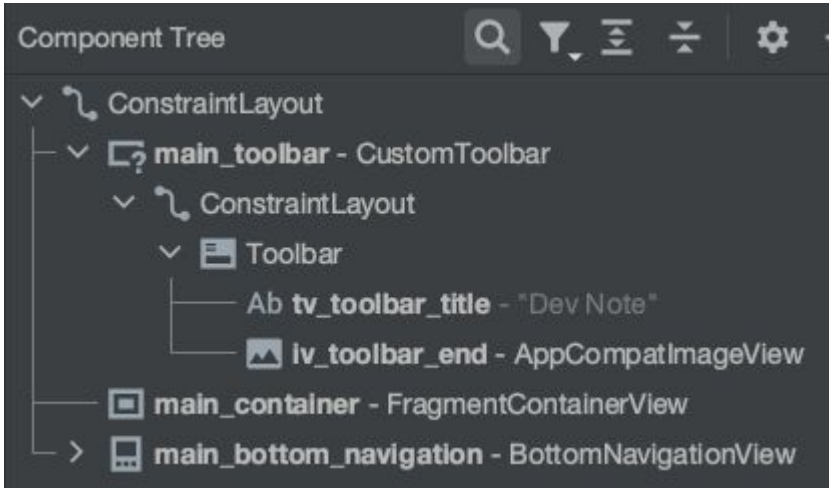
Custom 속성 적용

```
class CustomToolbar(context: Context, attrs: AttributeSet?) : ConstraintLayout(context, attrs) {

    private val title: TextView

    init {
        val view = inflate(context, R.layout.view_custom_toolbar,  root: this)
        title = view.findViewById(R.id.tv_toolbar_title)

        context.theme.obtainStyledAttributes(
            attrs,
            R.styleable.CustomToolbar,
             defStyleAttr: 0,  defStyleRes: 0
        ).apply {  this: TypedArray
            try {
                setTitle(
                    textRes = getResourceId(R.styleable.CustomToolbar_toolbarTitle,  defValue: 0),
                    drawableRes = getResourceId(
                        R.styleable.CustomToolbar_toolbarTitleLeftDrawable,
                         defValue: 0
                    )
                )
            } finally {
                recycle()
            }
        }

    }
```

```
private fun setTitle(textRes: Int, drawableRes: Int) {
    if (textRes != 0) {
        title.text = context.getString(textRes)
    }
    if (drawableRes != 0) {
        title.setCompoundDrawablesWithIntrinsicBounds(drawableRes,  top: 0,  right: 0,  bottom: 0)
    }
}
```

View 계층구조와 성능

계층 구조 분석



[성능 이슈]

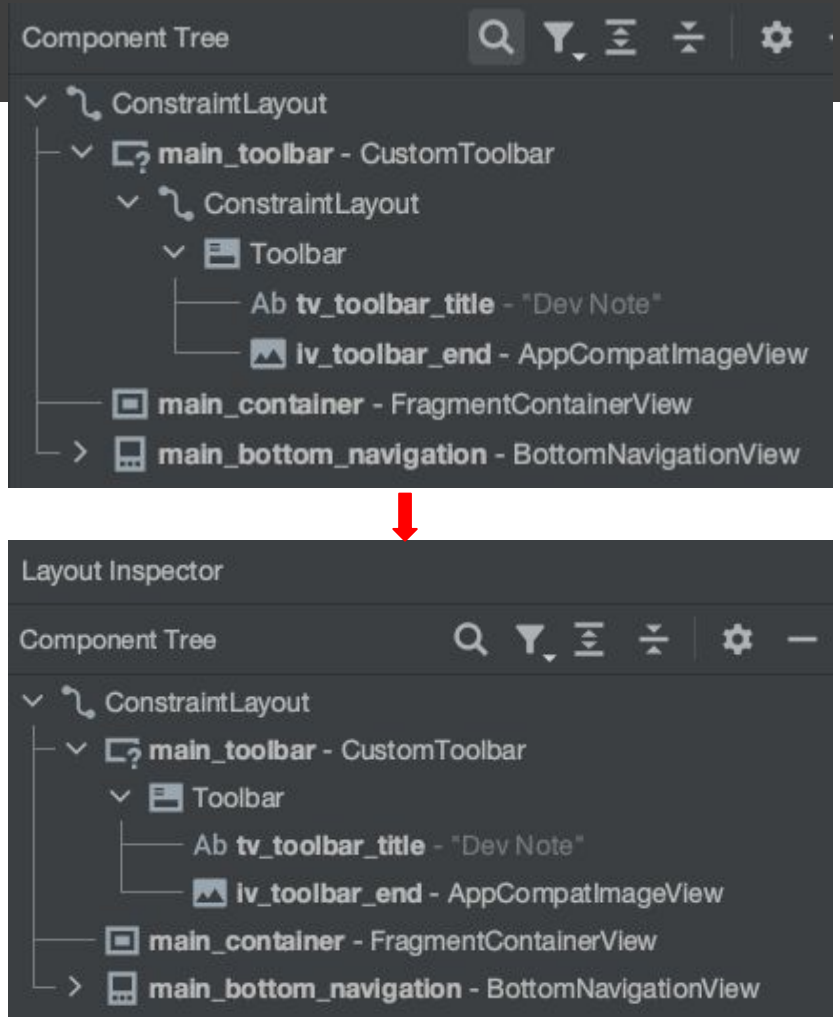
중첩된 Layout 객체는 Layout 렌더링 비용을 추가합니다.

[문제 발견]

CustomToolbar에서 ConstraintLayout을 재사용 가능한 레이아웃으로 사용했기 때문에 중첩된 ConstraintLayout 구조를 갖게됩니다.

중복된 ViewGroup 제거

```
<merge 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="wrap_content"
    tools:parentTag="androidx.constraintlayout.widget.ConstraintLayout">

    <androidx.appcompat.widget.Toolbar
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/tv_toolbar_title"
```



Summary

- 사용자 이벤트 처리 과정을 이해한다
- UI widget의 subclass를 만든다 : 단순, 복합 커스텀뷰
- onSizeChanged, onDraw와 같은 View의 method를 override 해서 원하는 appearance를 만든다.
- 성능 이슈 : Allocate space 와 변수 초기화 작업은 onDraw 외부에서 한다.
- XML layout에서 다른 UI widget 처럼 추가한다.
- Custom Attribute을 정의할 수 있다.

Resource

1. View 클래스 만들기
2. 성능과 뷰 계층 구조
3. Layout 계층 구조 최적화
4. Layout 재사용