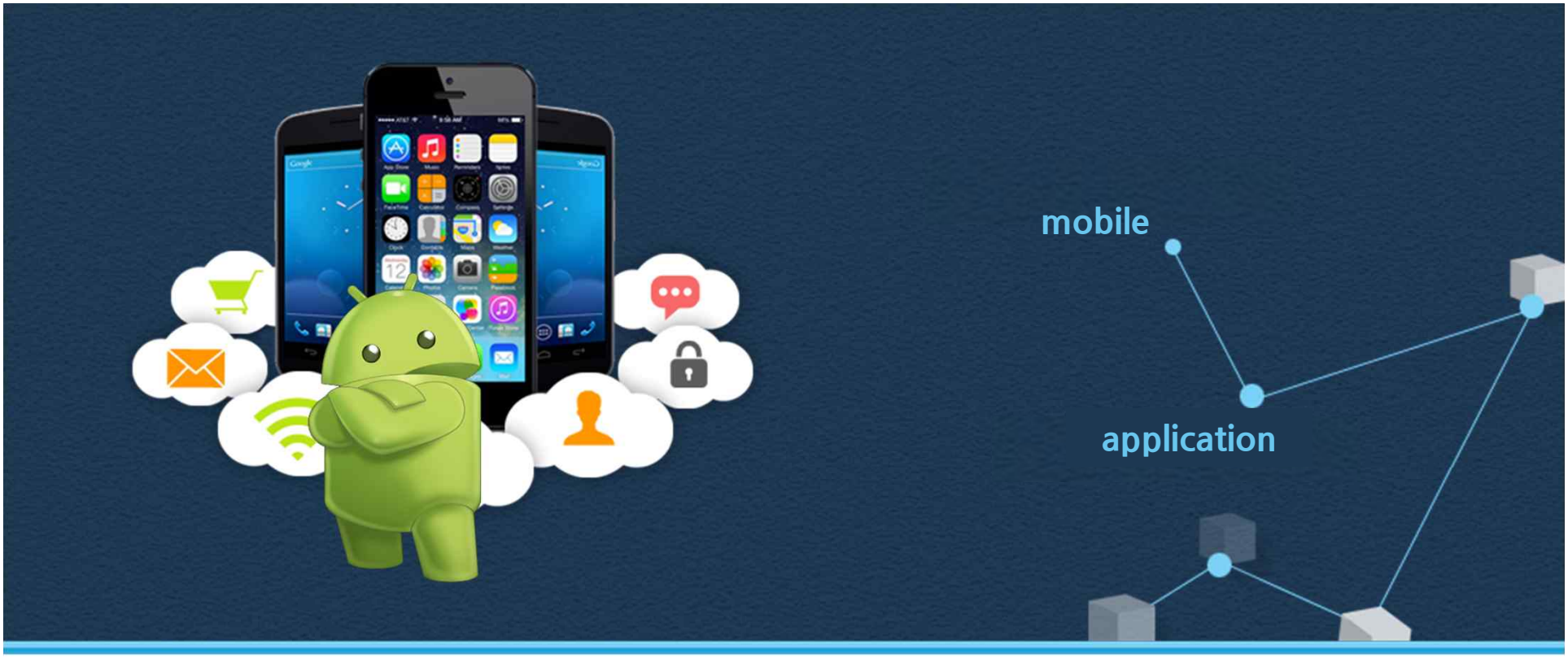


컨텍스트, 이벤트 처리

모바일게임프로그래밍

김지심 교수

학습목표

Context를 이해하고 활용할 수 있다.
터치 이벤트를 리뷰하고 실습에 활용할 수 있다.

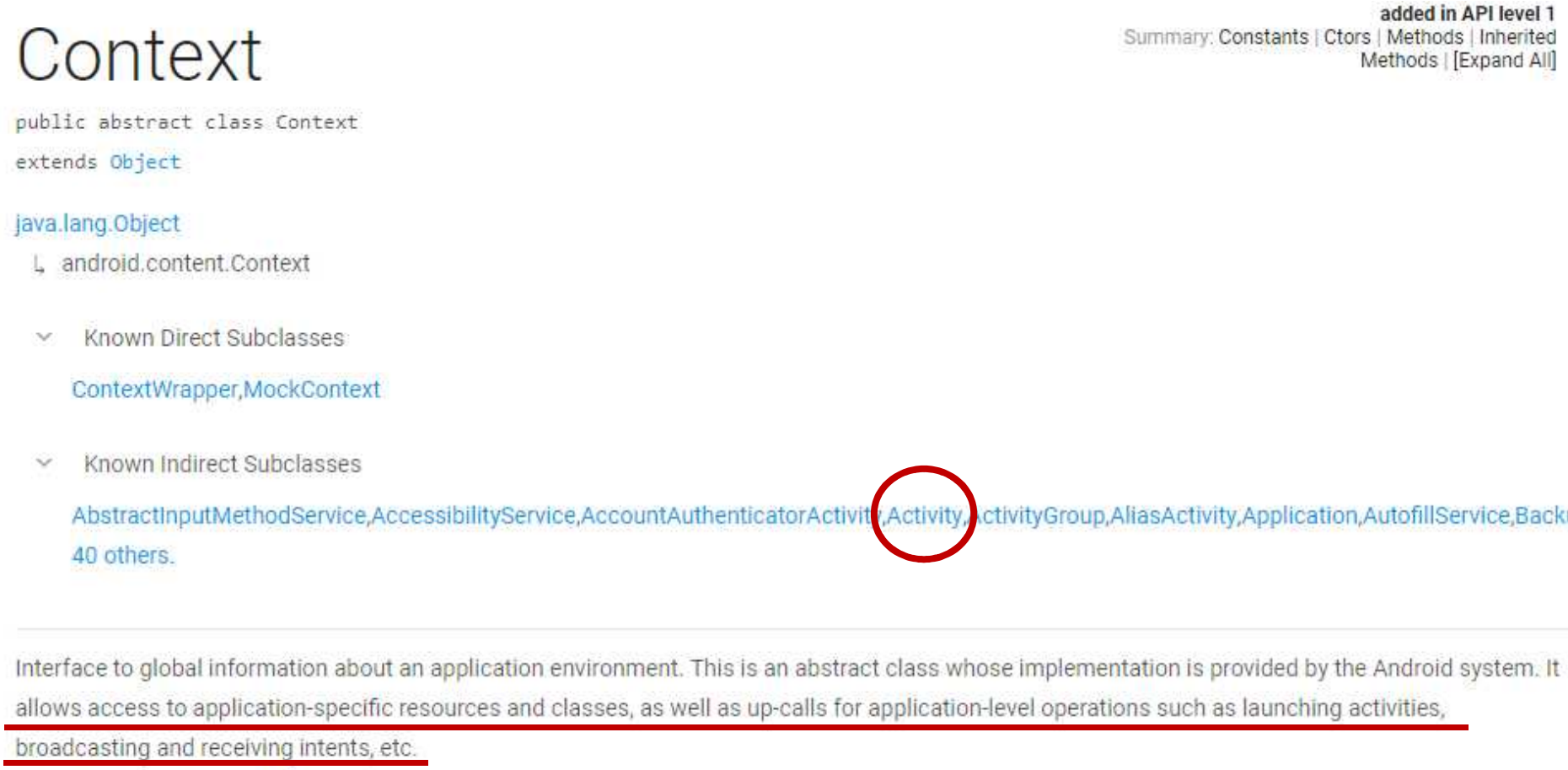
01

Context

Context

Context 계층도

Activity, Service 등의 부모



Context

added in API level 1

Summary: Constants | Ctors | Methods | Inherited Methods | [Expand All]

```
public abstract class Context
extends Object
```

java.lang.Object

↳ android.content.Context

Known Direct Subclasses

ContextWrapper,MockContext

Known Indirect Subclasses

AbstractInputMethodService,AccessibilityService,AccountAuthenticatorActivity,Activity,ActivityGroup,AliasActivity,Application,AutofillService,Backu

40 others.

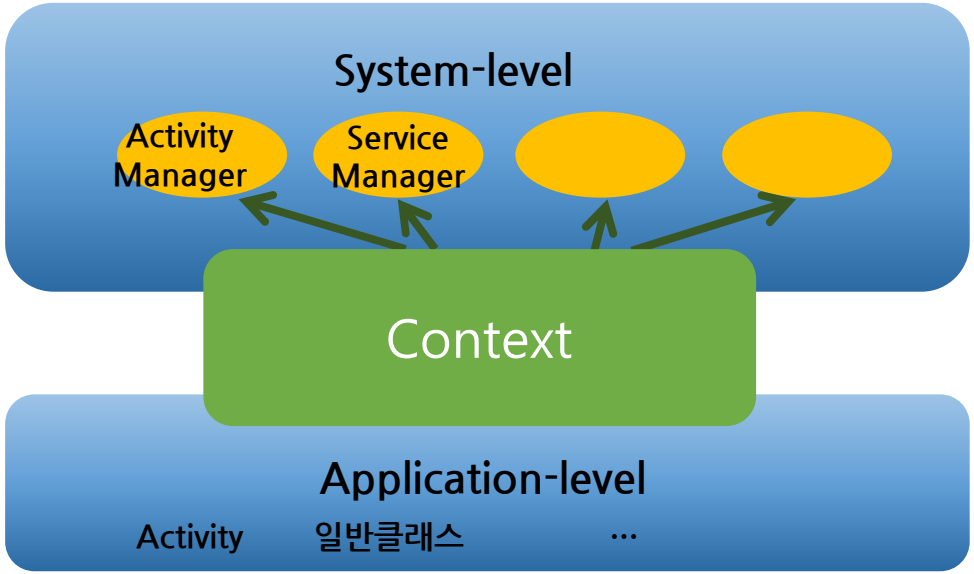
Interface to global information about an application environment. This is an abstract class whose implementation is provided by the Android system. It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc.

Context

Context란?

① 안드로이드 시스템의 글로벌 정보나 애플리케이션에 관한 정보에 접근하거나

② 안드로이드 시스템 서비스에서 제공하는 API를 호출할 때 사용하는 클래스



Context

Context 객체에서 제공하는 함수

①의 대표적 메소드: getResources(), getPackageName(), ...

②의 대표적 메소드: startActivity(), startService(), registerReceiver(), ...

> 결국, 앱 레벨에서 시스템 레벨의 기능 및 정보를 이용해야 할 때 Context 객체를 이용

Context

왜 Context를 통해야 할까?

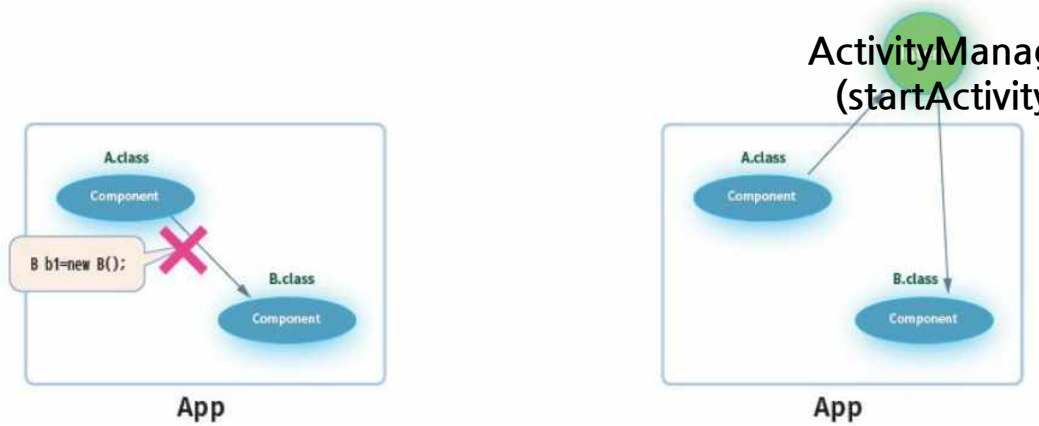
OS의 가장 중요한 역할 중 하나는 프로세스를 관리하는 것임.

만약 정확히 앱=프로세스라면 System의 함수들을 이용하여 직접 정보를 가져올 수 있으나,

안드로이드에서는 앱과 프로세스는 독립적임!

안드로이드 컴포넌트들은 매우 유연하게 관리됨!

메모리가 부족한 경우 프로세스가 강제 종료되거나,
혹은 앱에 관한 일부 정보만 관리하다가(ActivityManager) 이 정보들을 바탕으로 메모리 공간이 확보되면 새로운 프로세스가 실행되기도 함



> Context라는 매개체를 통해 앱 레벨에서 시스템에 접근

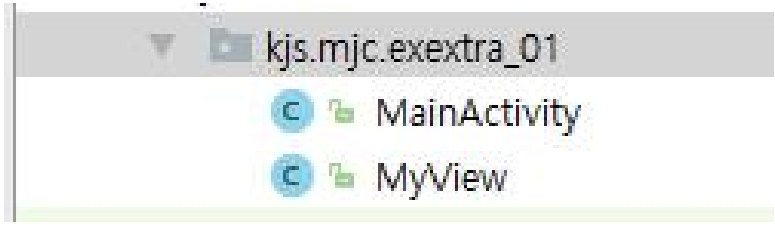
02

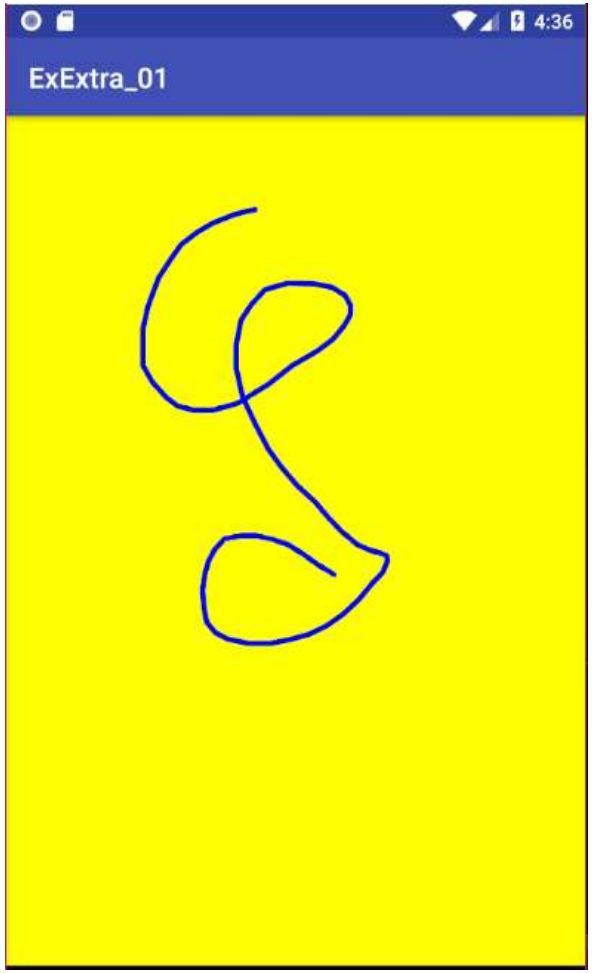
터치 이벤트 실습

기본적인 터치 이벤트

실습: ExDraw_01

터치하여 자유선 그리기





기본적인 터치 이벤트

실습: ExDraw_01

activity_main.xml은 없어도 됨

MyView.java

커스텀뷰 생성 및 그리기 처리

```
17  public class MyView extends View {
18
19      private Paint paint = new Paint();
20      Path path = new Path();
21
22      public MyView(Context context, AttributeSet attrs) {
23          super(context);
24          paint.setAntiAlias(true);
25          paint.setStrokeWidth(10f);
26          paint.setColor(Color.BLUE);
27          paint.setStyle(Paint.Style.STROKE);
28          paint.setStrokeJoin(Paint.Join.ROUND);
29          setBackgroundColor(Color.YELLOW);
30      }
31
32      @Override
33      public void onDraw(Canvas canvas) {
34          canvas.drawPath(path, paint);
35      }
```

기본적인 터치 이벤트

실습: ExDraw_01

MyView.java

```
37      @Override
38      public boolean onTouchEvent(MotionEvent event) {
39          float eventX = (int) event.getX();
40          float eventY = (int) event.getY();
41
42          switch (event.getAction()) {
43
44
45
46
47
48
49
50
51
52
53
54
55
56          return true;
57      }
58  }
```

완성하기!
터치의 MotionEvent를 받아서
상태값에 따라
path 그리기 관련 메소드를 사용

False를 리턴할 경우 후속 리스너에 이벤트를 전달하나,
이때에는 ACTION_MOVE, ACTION_UP이 제거되므로
터치 이벤트를 모두 처리해야 할 때에는 true를 리턴해야 함

기본적인 터치 이벤트

실습: ExDraw_01

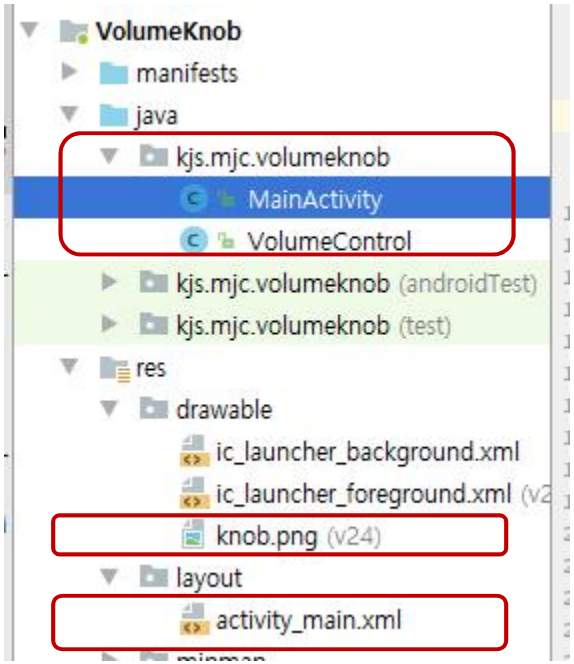
MainActivity.java

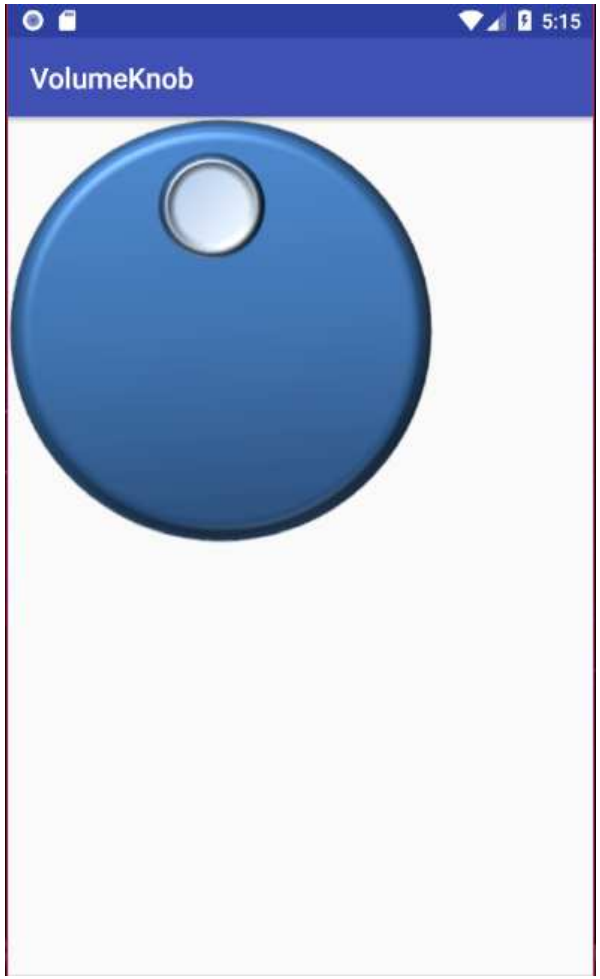
```
6   public class MainActivity extends AppCompatActivity {
7
8       @Override
9       protected void onCreate(Bundle savedInstanceState) {
10          super.onCreate(savedInstanceState);
11          MyView myView = new MyView(this, null);
12          setContentView(myView);
13      }
14  }
```

터치 이벤트 실습

실습: VolumeKnob

볼륨을 조절하는 Knob 만들기

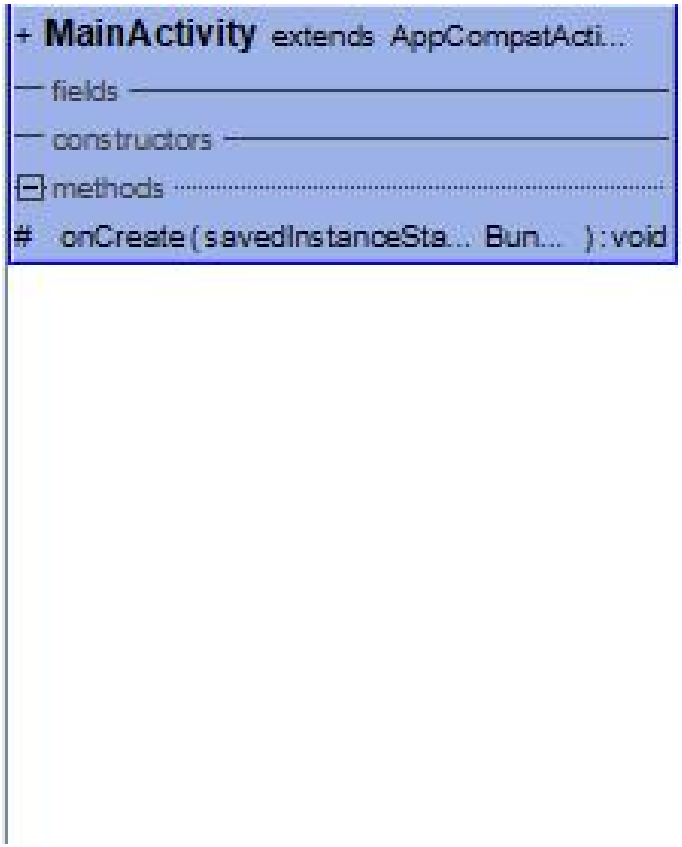


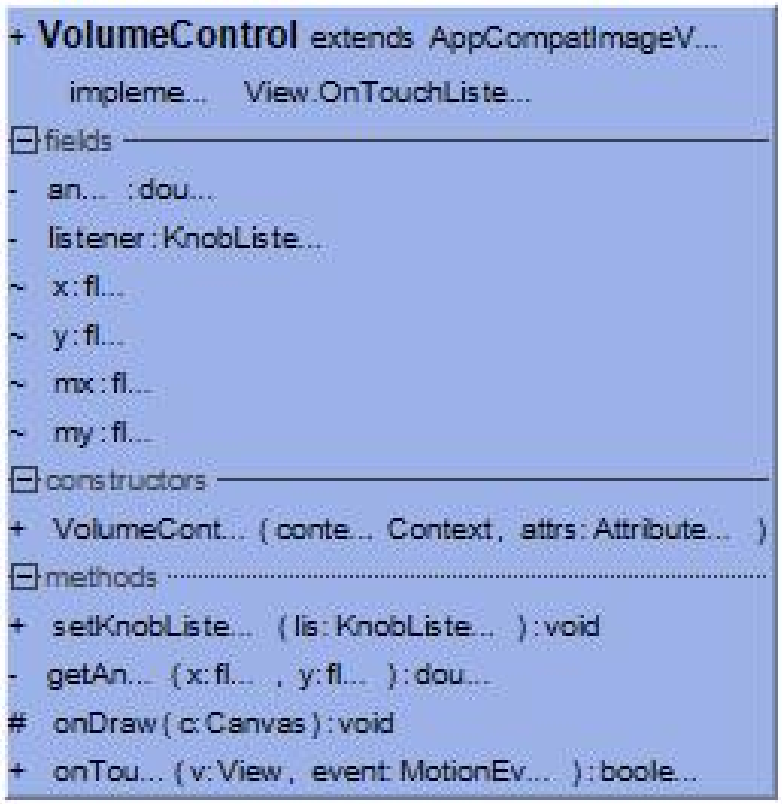


터치 이벤트 실습

실습: VolumeKnob

리스너를 구현하여(VolumeControl) 이를 Knob에 적용





터치 이벤트 실습

실습: VolumeKnob

VolmueControl.java

이미지뷰를 상속받고 터치리스너를 구현(implement)

```
15  public class VolumeControl extends AppCompatImageView implements View.OnTouchListener  {
16
17      private double angle = 0.0;
18      private KnobListener listener;
19      float x, y;
20      float mx, my;
21
22      public  VolumeControl(Context context, AttributeSet attrs)
23      {
24          super(context, attrs);
25          this.setImageResource(R.drawable.knob);
26          this.setOnTouchListener(this);
27      }
28
29      public interface KnobListener {
30          public void onChanged(double angle);
31      }
32
33      public void setKnobListener(KnobListener lis )
34      {
35          listener = lis;
36      }
37
38      private double getAngle(float x, float y)
39      {
40          mx = x - (getWidth() / 2.0f);
41          my = (getHeight() / 2.0f) - y;
42
43          double degree = Math.atan2(mx, my) * 180.0 / 3.141592;
44          return degree;
45      }
```

이제까지, 안드로이드 자바에서 제공하는 리스너를 이용해
익명 내부 클래스로 바로 원하는 뷰에 리스너를 셋팅한 것과 달리
개발자가 직접 원형으로 리스너를 정의

생성자 선언

리스너가 가지는 콜백 추상메소드인 onChanged 정의

원하는 뷰에 리스너를 설정하는
setKnobListener(setter) 정의

터치 이벤트 실습

실습: VolumeKnob

VolmueControl.java

사용자의 터치에 따라 Draw

```
47      @Override
48      protected void onDraw(Canvas canvas) {
49
50          canvas.rotate((float)angle, getWidth()/2, getHeight()/2); Invalidate 시 호출되어 회전
51          super.onDraw(canvas); //커스텀시 부모의 onDraw를 호출하여 ImageView의 속성을 이용하여 그리기 위해
52
53      }
54
55      @Override
56      public boolean onTouch(View v, MotionEvent event) {
57          x = event.getX(0);
58          y = event.getY(0);
59          angle = getAngle(x, y);
60          invalidate();
61          listener.onChanged(angle); //필요시 MainActivity에서 onChanged를 overriding하여 이벤트 처리
62
63          return true;
64      }
65
66  }
```

터치 이벤트 실습

실습: VolumeKnob

activity_main.xml

커스텀뷰 적용

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://s
    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">

    <kjs.mjc.volumeknob.VolumeControl
        android:id="@+id/volume"
        android:layout_width="300dp"
        android:layout_height="300dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>

</android.support.constraint.ConstraintLayout>
```

터치 이벤트 실습

실습: VolumeKnob

MainActivity.java

```
6   public class MainActivity extends AppCompatActivity {
7
8       @Override
9       protected void onCreate(Bundle savedInstanceState) {
10          super.onCreate(savedInstanceState);
11          setContentView(R.layout.activity_main);
12
13          VolumeControl view = findViewById(R.id.volume);
14          view.setKnobListener(new VolumeControl.KnobListener() {
15              @Override
16              public void onChanged(double angle) {
17
18                  if (angle > 0)
19                      ; //TODO
20                  else
21                      ; //TODO
22
23              }
24          });
25      }
26  }
27
```

개발자가 직접 만든 VolumeControl.KnobListener 리스너를 사용하여 Knob의 이벤트 처리

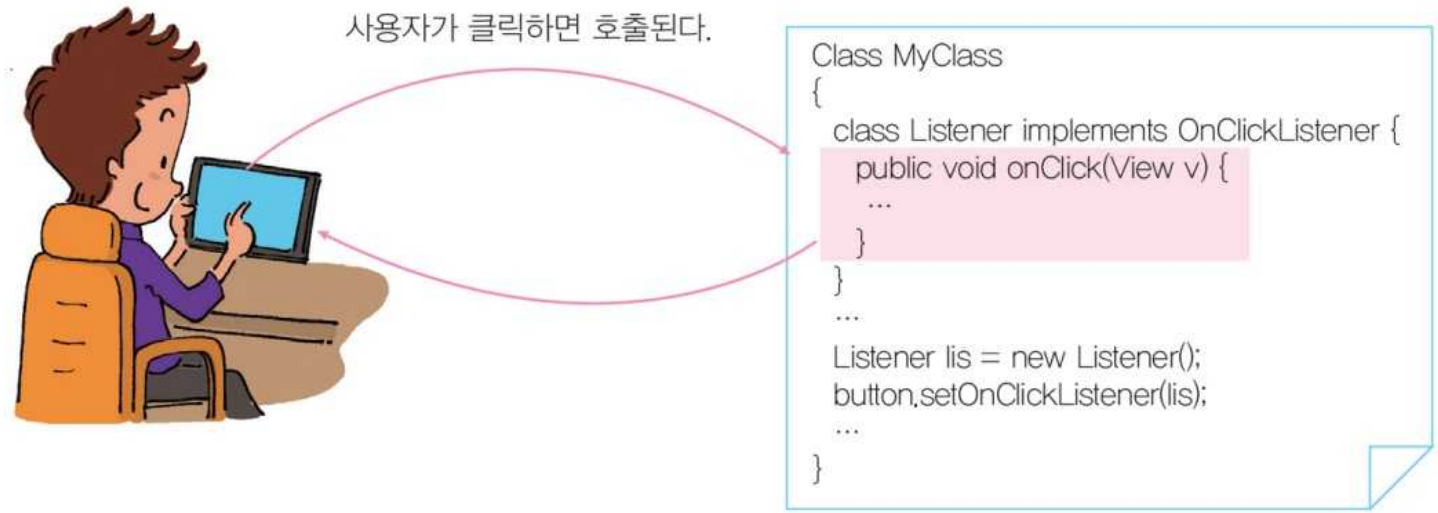
※ 제스처: https://material.io/guidelines/patterns/gestures.html

Review

리스너 이해

2. 프로젝트 구성 살펴보기

버튼에 붙은 리스너 객체가 이벤트를 처리함



2. 프로젝트 구성 살펴보기

리스너의 기본 정의

```
class MyClass
{
    class Listener implements OnClickListener {
        public void onClick(View v){
            ...
        }
    }
    ...
    Listener lis = new Listener();
    button.setOnClickListener(lis);
    ...
}
```

리스너 인터페이스를 구현한 클래스 정의

이벤트 리스너 객체 생성

버튼에 이벤트 리스너 객체를 등록

참고문헌

천인국(2015), 그림으로 쉽게 설명하는 안드로이드 프로그래밍(개정3판), 생능출판사.

강성윤(2017). 깡샘의 안드로이드 프로그래밍, 루비페이퍼.