이벤트 처리

성신여자대학교 컴퓨터공학과
우종정 교수

이벤트 처리

- GUI 운영체제는 이벤트 구동(event driven) 방식으로 동작
- 사용자와 상호 작용하려면 이벤트를 매끄럽게 처리하는 기술 필요
- 안드로이드에는 이벤트를 처리하는 다양한 방법이 있다.
- 팀 작업을 위해 모든 방법을 숙지해야 하며 자바의 고급 문법이 사용된다.

이벤트 핸들러

콜백 메서드

- 상호 작용 주체가 View이므로 이벤트 콜백은 주로 View가 제공
  - boolean onTouchEvent (MotionEvent event)
  - boolean onKeyDown (int keyCode, KeyEvent event)
  - boolean onKeyUp (int keyCode, KeyEvent event)
  - boolean onTrackballEvent (MotionEvent event)

- 장점 : 간단하고 직관적

- 문제점 및 한계
  - 콜백 메서드 재정의를 위해 반드시 상속을 받아야 한다.
  - 모든 이벤트에 대해 콜백이 정의되어 있지 않다.

- 예제: HandleEvent1

이벤트 핸들러

리스너(listener)

- 이벤트를 처리하기 위한 인터페이스
- View 클래스의 내부 인터페이스로 정의되어 있음.
- 대응하는 이벤트를 받는 단 하나의 메서드가 선언되어 있음

- 예 :
  - OnTouchListener : boolean onTouch (View v, MotionEvent event)
  - OnKeyListener : boolean onKey (View v, int keyCode, KeyEvent event)
  - OnClickListener : void onClick (View v)
  - OnLongClickListener : boolean onLongClick (View v)
  - OnFocusChangeListener : void onFocusChange (View v, boolean hasFocus)

- 통상 특정 뷰에 대해 정의되는 것이므로 외부에 선언할 필요 없이 액티비티 클래스에 포함된 내부 클래스로 선언

이벤트 핸들러

리스너

- 인터페이스 구현, 리스너 객체 생성, 이벤트와 연결 과정을 거쳐 사용

- 리스너 등록 메서드 : set리스너
  - void setOnTouchListener (View.OnTouchListener l)
  - void setOnKeyListener (View.OnKeyListener l)
  - void setOnClickListener (View.OnClickListener l)
  - void setOnLongClickListener (View.OnLongClickListener l)
  - void setOnFocusChangeListener (View.OnFocusChangeListener l)

- 예제: HandleEvent2 (인터페이스 구현 객체 생성)

- 리스너는 여러 뷰에 의해 공유될 수 있으므로 어떤 객체에서 발생한 이벤트인지 View 타입 인수로 전달

이벤트 핸들러

리스너 구현 객체

- 인터페이스는 어떤 클래스에서도 구현 가능
  - 액티비티가 구현 : 코드는 짧아지지만 구조적으로 바람직하지 못하며 뷰의 독립성이 떨어진다.
  - 뷰가 구현 : 인터페이스 구현을 위해 상속을 받아야 하는 문제가 있다.
  - 두 방법은 특정한 조건이 만족될 때만 사용하는 것이 바람직하다.

- 액티비티가 리스너를 구현: HandleEvent3

- 뷰가 리스너를 구현: HandleEvent4

이벤트 핸들러

■ 익명 내부 클래스 사용

- 리스너 구현 클래스는 딱 하나의 객체만 생성하므로 굳이 이름을 줄 필요가 없다.

- 인터페이스를 구현하면서 동시에 리스너 객체를 생성한다.

class MyClass implements Interface {
메서드 구현
}
MyClass obj = new MyClass();
	Interface obj = new Interface() {
	메서드 구현
	};

■ 일반적인 코드	■ 축약된 코드

- 예제: HandleEvent5

이벤트 핸들러

익명 내부 객체

- 리스너 객체는 등록할 때 딱 한번만 사용하며 두 개 이상의 객체를 생성하지 않는다.
- 등록 메서드의 인수 목록에서 리스너 객체를 생성한다.

Class obj = new Class(); Method(obj);	Method(new Class());
■ 이름있는 객체 사용	■ 임시 객체 사용

- 예제: HandleEvent6

- 람다식 사용
  - vw.setOnTouchListener((v, event) -> { 이벤트 처리 코드 });

이벤트 핸들러

외부 변수 접근

- 내부 클래스는 외부 클래스의 멤버를 참조할 수 있다.
- 핸들러로 전달할 값을 멤버로 선언하는 것은 구조적으로 바람직하지 않다.
- 지역 변수를 핸들러로 전달할 때는 사실상 final

- 예제 : 레이아웃, 액티비티

여러 가지 이벤트

터치 입력

- 손가락으로 화면을 누를 때 발생한다.
  - boolean onTouchEvent (MotionEvent event)
  - boolean onTouch (View v, MotionEvent event)
- 콜백에 비해 리스너는 이벤트 대상 뷰를 인수로 전달받는다.
- getAction : 화면에서 어떤 동작을 했는지를 전달한다.

동작	설명
ACTION_DOWN	화면을 눌렀다.
ACTION_MOVE	누른채로 움직였다.
ACTION_UP	화면에서 손가락을 뗐다.

- getX, getY : 터치한 좌표

여러 가지 이벤트

키보드 입력

- 키보드 또는 화면 키보드를 눌렀을 때 발생한다.
  - boolean onKeyDown (int keyCode, KeyEvent event)
  - boolean onKey (View v, int keyCode, KeyEvent event)
- 인수로 키코드와 액션이 전달된다.

상수	설명
KEYCODE_DPAD_LEFT	왼쪽 이동키
KEYCODE_DPAD_RIGHT	오른쪽 이동키
KEYCODE_DPAD_UP	위쪽 이동키
KEYCODE_DPAD_DOWN	아래쪽 이동키
KEYCODE_DPAD_CENTER	이동키 중앙의 버튼
KEYCODE_A	알파벳 A. 알파벳 B부터는 KEYCODE_B 식이다.
KEYCODE_0	숫자 0. 1부터는 KEYCODE_1 식이다.
KEYCODE_CALL	통화
KEYCODE_ENDCALL	통화 종료
KEYCODE_HOME	홈
KEYCODE_BACK	뒤로
KEYCODE_VOLUME_UP	볼륨 증가 버튼
KEYCODE_VOLUME_DOWN	볼륨 감소 버튼

동작	설명
ACTION_DOWN	키를 눌렀다.
ACTION_UP	키를 뗐다.
ACTION_MULTIPLE	같은 키를 여러 번 눌렀다.

여러 가지 이벤트

클릭

- View.OnClickListener 리스너로 이벤트를 받아야 한다. 클릭하면 다음 메서드가 호출된다.
  - void onClick (View v)

- 비슷한 코드가 반복되면 통합하는 것이 리팩토링의 기본
- 비슷한 기능의 핸들러를 하나로 통합할 경우 임시 객체를 사용할 수 없으며 이름이 있어야 한다.
- 공유 핸들러는 리스너의 getId 메서드로 클릭된 버튼을 알아낸다.
- XML 리소스의 onClick 속성으로 핸들러 메서드를 지정할 수 있다.

- 예제: Fruit, layout
- 예제: Fruit2, layout

여러 가지 이벤트

롱클릭

- 누른 채로 일정 시간을 대기할 때 발생한다.
- 데스크탑의 마우스 우측 버튼 클릭에 해당하며 주로 컨텍스트 메뉴를 여는 용도로 사용
- 리스너 등록 메서드와 이벤트 핸들러
  - void setOnLongClickListener (View.OnLongClickListener l)
  - boolean onLongClick (View v)

- 클릭과 유사하되 반환 값으로 클릭 이벤트 발생 여부를 통제한다. 반환 값은 프로그램의 논리에 따라 달라진다.
  - true를 반환하면 처리를 끝낸다.
  - false를 반환하면 처리가 덜 끝난 것으로 알고 클릭 이벤트가 추가로 발생한다.

- 예제: activity, layout