Java프로그래밍

14강. AWT 이벤트 처리하기 (교재 13장)

컴퓨터과학과 김희천 교수

한국방송통신대학교

오늘의 학습목차

1. AWT 컨트롤 클래스 (교재 12장)
2. 이벤트
3. 이벤트 클래스와 이벤트 리스너
4. 이벤트 처리 방법
5. 이벤트 종류와 이벤트 처리

1. AWT 컨트롤 클래스 (교재 12장)

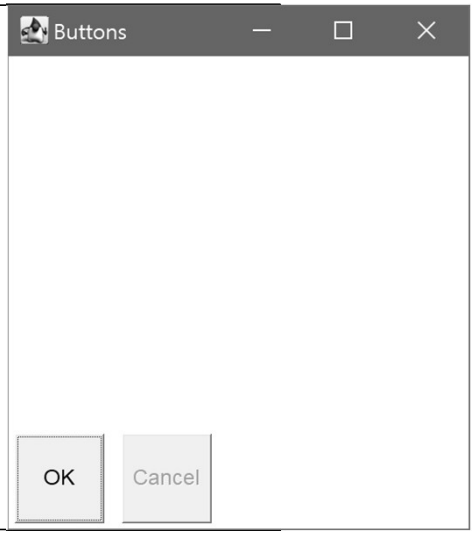
1. AWT 컨트롤 클래스

1) Button 클래스

- 제목(또는 이름)이 있는 버튼을 표현
- 버튼을 눌렀을 때 이벤트 처리에 의해 액션을 수행할 수 있음

생성자와 메소드

- Button( ), Button(String label)
- void setLabel(String label)
- String getLabel( )
- Component 클래스에서 상속받은 메소드
  - ✓ setVisible( ), setBounds( ), setEnabled( ) 등



2) Checkbox 클래스

- 체크 박스를 표현
  - ✓ 선택(true)/비선택(false) 또는 on/off 상태를 표현

생성자와 메소드

- Checkbox( ), Checkbox(String label, boolean state)
  - ✓ 제목과 상태를 지정할 수 있음
- Checkbox(String , boolean , CheckboxGroup)
  - ✓ 같은 그룹의 체크박스들은 라디오 버튼처럼 동작
  - ✓ 하나의 체크박스만이 on 상태가 될 수 있음
- void setLabel(String) String getLabel()
- void setState(booelan), Boolean getState()

3) Checkbox 예제

```
import java.awt.*;
public class CheckboxTest {
    public static void main(String[ ] args) {
        Frame f = new Frame("Checkbox");
        f.setLayout(new FlowLayout( ));
        f.add(new Checkbox("Whiskey"));
        f.add(new Checkbox("Beer"));

        CheckboxGroup group = new CheckboxGroup( );
        f.add(new Checkbox("Yes", false, group));
        f.add(new Checkbox("No", true, group));
        f.setSize(300, 80);
        f.setVisible(true);
    }
}
```

4) Choice 클래스

- 콤보 박스를 표현
  - ✓ 선택할 아이템이 많은 경우 사용
  - ✓ 하나를 선택할 수 있으며, 현재 선택된 것이 보임

메소드

- void addItem(String item)
- void insert(String item, int index)
  - ✓ 인덱스는 0부터 시작함
- String getItem(int index)
- int getSelectedIndex( )
- String getSelectedItem( )

5) List 클래스

- 하나 또는 여러 개의 아이템을 선택할 수 있게 함
  - ✓ 선택할 아이템이 매우 많을 때 사용
  - ✓ 기본은 단일 선택이나 다중 선택 가능, 스크롤 가능

생성자와 메소드

- List( ), List(int),List(int rows, boolean multipleMode)
  - ✓ 한 번에 보여줄 아이템의 개수와 다중 선택 여부를 지정
- void add(String item), void add(String item, int index)
- String getItem(int index)
- int getSelectedIndex( ), int[ ] getSelectedIndexes( )
- String getSelectedItem( ), String[ ] getSelectedItems( )

6) List 예제

```
import java.awt.*;
public class ListTest {
   public static void main(String[ ] args) {
        Frame f = new Frame("List");
        List l = new List( );

        l.add("Red");
        l.add("Green");
        l.add("Blue");
        l.add("Yello");
        l.add("Cyan");

        f.add(l);
        f.setSize(200, 100);
        f.setVisible(true);
   }
}
```

7) TextComponent 클래스

- 텍스트를 편집하거나 다루기 위한 컴포넌트
- TextArea와 TextField의 부모 클래스

메소드

- int getCaretPosition( ),
  void setCaretPosition(int position)
  - ✓ (편집 위치를 알려주는) 캐릿의 위치를 조회/설정
- String getSelectedText( )
- String getText( )
- void select(int start, int end)
- void setText(String t)

8) TextField 클래스

◆ 한 줄의 텍스트 편집을 위한 컴포넌트
  ✓ 여러 줄의 텍스트 편집에는 TextArea 클래스를 사용

생성자와 메소드

◆ TextField(String text, int columns)
◆ void setColumns(int cols), int getColumns( )
◆ void setEchoChar(char c), char getEchoChar( )
  ✓ 비밀번호 등을 입력할 때 echo 문자를 '*' 등으로 설정
◆ void setText(String text)

2. 이벤트

1) 이벤트(Event)

- 사용자가 GUI 컴포넌트를 사용하면서 발생시키는 행위
  - ✓ 마우스 클릭, 버튼 누름, 키보드 입력, 메뉴 선택 등
- 이벤트는 종류별로 클래스로 정의되어 있음
  - ✓ MouseEvent, ActionEvent, ItemEvent 등

java.awt.event

- AWT 컴포넌트에서 발생하는 다양한 이벤트를 처리하기 위한 인터페이스와 클래스 제공
  - ✓ xxxEvent, *xxxListener*, xxxAdapter

2) 이벤트 기반 프로그래밍

◆ 무한루프를 돌면서 사용자의 행위로 인한 이벤트를 청취하여 응답하는 형태로 작동하는 프로그래밍

컴포넌트와 이벤트 발생

◆ GUI 컴포넌트에서 여러 이벤트가 발생할 수 있음
  ✓ 이벤트 소스(Event Source)

◆ 컴포넌트마다 발생될 수 있는 이벤트가 정해져 있음
  ✓ Button의 경우 발생 가능한 이벤트
    • ActionEvent, ComponentEvent, FocusEvent, KeyEvent, MouseEvent
  ✓ ActionEvent가 발생할 수 있는 컴포넌트
    • Button, List, MenuItem, TextField

3) 이벤트 처리 방식

- 위임형 이벤트 처리 모델
  - ✓ 이벤트 리스너 객체에 이벤트 처리를 위임함
    - 컴포넌트(이벤트 소스)에 이벤트가 발생했을 때, 컴포넌트에 등록된 리스너 객체를 통해 이벤트 처리 코드를 실행함

방법

- 모든 컴포넌트는 이벤트 소스가 될 수 있음
- 이벤트 소스에 이벤트 리스너 객체를 등록하는 방법
  - ✓ 하나의 이벤트 소스에서 여러 다른 이벤트가 발생 가능하며, 이벤트 종류별로 각각 이벤트 처리를 등록해야 함
- 방법
  - ✓ 컴포넌트.add*XxxListener* (리스너객체);

3. 이벤트 클래스와 이벤트 리스너

4. 이벤트 처리 방법

1) 이벤트 클래스와 리스너 인터페이스

◆ 이벤트 클래스

✓ 이벤트는 클래스(xxxEvent)로 분류되어 있음

- 예 – MouseEvent는 마우스 클릭과 관련된 이벤트

✓ 이벤트 클래스는 종류에 맞는 정보와 메소드를 가짐

리스너 인터페이스

◆ 이벤트 리스너 인터페이스는 이벤트 처리를 위한 인터페이스

✓ 리스너 객체는 이벤트 처리를 위임 받은 객체

◆ 이벤트 클래스에 1대1로 대응되는 인터페이스

✓ 예: WindowEvent - *WindowListener*,
ItemEvent - *ItemListener*

2) 이벤트 리스너 인터페이스

- 리스너 인터페이스에는 개별 이벤트를 처리하기 위한 메소드가 하나 또는 여러 개 존재함
- 2개 이상의 메소드를 가지는 인터페이스는 상응하는 이벤트 어댑터 클래스가 존재함

이벤트 리스너 객체

- 이벤트 리스너 인터페이스를 구현한 클래스의 객체
  - ✓ 또는 이벤트 어댑터 클래스를 상속받은 클래스의 객체
- 이벤트 처리를 담당함
  - ✓ 적절한 메소드를 실행

3) 이벤트 클래스와 이벤트 발생

이벤트 클래스	이벤트가 발생하는 경우
ActionEvent	버튼 클릭, 리스트 항목을 더블 클릭, 메뉴 항목 선택, 텍스트필드에서 엔터키를 치는 경우(*ActionListener*)
AdjustmentEvent	스크롤바를 조작할 때
ComponentEvent	컴포넌트가 가려지거나 보일 때, 크기나 위치가 변할 때(*ComponentListener*)
ContainerEvent	컨테이너에 자식 컴포넌트가 추가되거나 삭제될 때
FocusEvent	입력 포커스를 얻거나 잃었을 때(*FocusListener*)
ItemEvent	체크박스, 체크메뉴 항목, 초이스 항목, 리스트 항목을 선택하거나 해제할 때(*ItemListener*)
KeyEvent	키보드가 누르거나 떼거나 타이핑할 때(*KeyListener*)
MouseEvent	마우스 클릭, 마우스 포인터가 컴포넌트 위로 올라오거나 나갈 때, 마우스를 누르거나 뗄 때(*MouseListener*)
	마우스가 움직이거나 드래깅 될 때(*MouseMotionListener*)
TextEvent	텍스트컴포넌트에서 텍스트에 변화가 생겼을 때(*TextListener*)
WindowEvent	윈도우의 시스템 버튼을 눌렀을 때(*WindowListener*)

4) 이벤트 리스너, 이벤트 어댑터와 메소드(1)

리스너 인터페이스와 어댑터	이벤트 처리를 위한 개별 메소드
ActionListener	actionPerformed(ActionEvent e)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)
ComponentListener ComponentAdapter	componentHidden(ComponentEvent e), componentMoved(ComponentEvent e), componentResized(ComponentEvent e), componentShown(ComponentEvent e)
ContainerListener ContainerAdapter	componentAdded(ContainerEvent e), componentRemoved(ContainerEvent e)
FocusListener FocusAdapter	focusGained(FocusEvent e), focusLost(FocusEvent e)
ItemListener	itemStateChanged(ItemEvent e)
KeyListener KeyAdpater	keyTyped(KeyEvent e), keyPressed(KeyEvent e), keyReleased(KeyEvent e)

4) 이벤트 리스너, 이벤트 어댑터와 메소드(2)

리스너 인터페이스와 어댑터	이벤트 처리를 위한 개별 메소드
MouseListener MouseAdapter	mousePressed(MouseEvent e), mouseReleased(MouseEvent e), mouseEntered(MouseEvent e), mouseExited(MouseEvent e), mouseClicked(MouseEvent e)
MouseMotionListener MouseAdapter	mouseMoved(MouseEvent e), mouseDragged(MouseEvent e)
TextListener	textValueChanged(TextEvent e)
WindowListener WindowAdapter	windowOpened(WindowEvent), windowClosing(WindowEvent), windowClosed(WindowEvent), windowIconified(WindowEvent), windowDeiconified(WindowEvent) windowActivated(WindowEvent), windowDeactivated(WindowEvent)

4. 이벤트 처리 방법

4. 이벤트 처리 방법

1) 이벤트의 등록과 처리 과정(1)

◆ 필요한 클래스의 정의

✓ 이벤트 소스와 처리할 이벤트 종류를 결정

- 예: Button에서 ActionEvent를 처리하고자 함

✓ 상응하는 리스너 인터페이스를 구현하는 클래스를 정의

- 여기서 이벤트 처리를 위한 개별 메소드를 구현함
- 개별 메소드가 이벤트를 처리하는 코드임
- 예: *ActionListener* 를 구현하는 클래스 A를 정의하고 actionPerformed( )를 구현

✓ 상응하는 어댑터 클래스가 존재하면 어댑터 클래스를 상속 받는 클래스를 정의해도 됨

- 불필요한 메소드의 구현을 생략할 수 있음

4. 이벤트 처리 방법

1) 이벤트의 등록과 처리 과정(2)

- 이벤트 등록
  - ✓ 리스너 객체를 생성하고 해당 이벤트 소스에 처리하고자 하는 이벤트를 등록함
    - 예: aButton.addActionListener(new A( ));

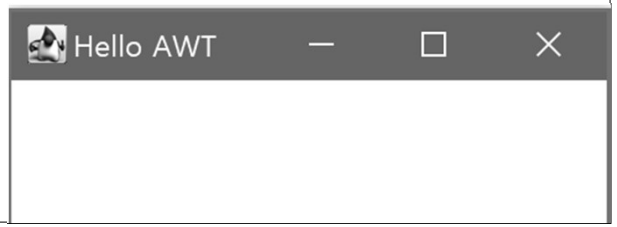
이벤트 처리

- 버튼을 누르면 ActionEvent가 발생함
- 등록되어 있는 이벤트 리스너 객체를 통해 void actionPerformed(ActionEvent ev)가 실행됨
  - ✓ 이벤트 객체가 인자로 전달됨

4. 이벤트 처리 방법

2) 이벤트 처리 예제(1)

```
class MyFrame extends Frame {
	public MyFrame(String title) {
		super(title);
		this.setSize(400, 300);
		this.setVisible(true);
		//이벤트 리스너 등록
		this.addWindowListener(new MyListener( ));
	}
	public void paint(Graphics g) {
		g.drawString("Hello AWT", 150, 150);
	}
}
public class WindowEventTest {
	public static void main(String args[ ]) {
		MyFrame myFrame = new MyFrame("Hello AWT");
	}
}
```



2) 이벤트 처리 예제(2)

```
import java.awt.*;
import java.awt.event.*;
//이벤트 리스너 구현
class MyListener implements WindowListener {
   public void windowClosing(WindowEvent ev) {
     System.exit(0);
   }
   public void windowActivated(WindowEvent ev) { }
   public void windowClosed(WindowEvent ev) { }
   public void windowDeactivated(WindowEvent ev) { }
   public void windowDeiconified(WindowEvent ev) { }
   public void windowIconified(WindowEvent ev) { }
   public void windowOpened(WindowEvent ev) { }
}
```

3) 어댑터 클래스 사용 예

```
import java.awt.*;
import java.awt.event.*;

//어댑터를 상속
class MyListener extends WindowAdapter {
   public void windowClosing(WindowEvent ev) {
     System.exit(0);
   }
}
```

5. 이벤트 종류와 이벤트 처리

5. 이벤트 종류와 이벤트 처리

1) ActionEvent와 *ActionListener*

- *ActionListener* 의 메소드
  - ✓ ActionEvent 발생은 명령의 실행을 의미함
  - ✓ void actionPerformed(ActionEvent ev)
    - 버튼을 클릭하는 경우, 메뉴 항목을 클릭하는 경우, TextField에서 엔터키를 치는 경우, 리스트의 항목을 더블 클릭하는 경우

- ActionEvent의 메소드
  - ✓ String getActionCommand( )
    - 명령의 이름을 리턴
  - ✓ Object getSource( )
    - 부모인 EventObject 클래스에서 상속된 메소드

2) WindowEvent와 *WindowListener*

- *WindowListener* 의 메소드
  - ✓ 윈도우의 상태 변화를 야기하는 경우
  - ✓ void windowActivated(WindowEvent ev)
  - ✓ void windowClosed(WindowEvent ev)
  - ✓ ……
- WindowEvent의 메소드
  - ✓ int getNewState( )
    - 0이면 정상 상태임
  - ✓ int getOldState( )
  - ✓ Window getWindow( )

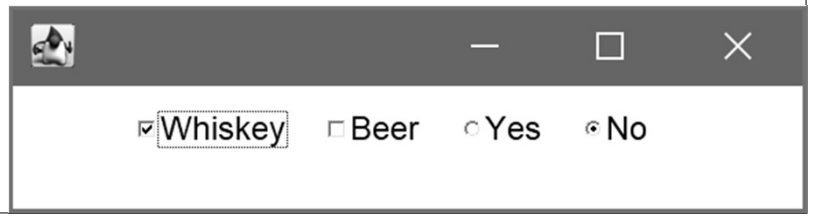
3) ItemEvent와 *ItemListener*

- *ItemListener* 의 메소드
  - ✓ void itemStateChanged(ItemEvent ev)
    - Checkbox, CheckboxMenuItem, Choice, List에서 아이템을 선택하거나 해제하는 경우

- ItemEvent의 메소드
  - ✓ Object getItem( )
    - 선택/해제한 항목을 리턴
  - ✓ int getStateChange( )
    - ItemEvent.*SELECTED*, ItemEvent.*DESELECTED*

4) ItemEvent 처리 예제

```
class MyListener implements ItemListener {
	public void itemStateChanged(ItemEvent ev) {
	String item = (String)ev.getItem( );
	System.out.print(item + "\t");
	if (ev.getStateChange( ) == ItemEvent.SELECTED)
		System.out.println("SELECTED");
	else
		System.out.println("DESELECTED");
	}
}
```



5) MouseEvent와 리스너(1)

- *MouseListener* 의 메소드
  - ✓ void mouseClicked(MouseEvent ev)
  - ✓ void mouseEntered(MouseEvent e)
  - ✓ void mouseExited(MouseEvent e)
  - ✓ void mousePressed(MouseEvent e)
  - ✓ void mouseReleased(MouseEvent e)

- *MouseMotionListener* 의 메소드
  - ✓ void mouseDragged(MouseEvent ev)
  - ✓ void mouseMoved(MouseEvent ev)

5) MouseEvent와 리스너(2)

- MouseEvent의 메소드
  - ✓ int getButton( )
    - 상태를 바꾼 마우스 버튼을 리턴
    - MouseEvent.*BUTTON1* , MouseEvent.*BUTTON2* 또는 MouseEvent.*BUTTON3* 을 리턴
  - ✓ int getClickCount( )
    - 클릭한 횟수를 리턴
  - ✓ Point getPoint( )
    - 마우스의 x, y 좌표를 리턴
  - ✓ int getX( ), int getY( )

6) MouseEvent 처리 예제

```
class MyMouseListener extends MouseAdapter {
     public void mouseClicked(MouseEvent ev) {
      Point p = ev.getPoint( );
      String btn = null;
      switch(ev.getButton( )) {
         case MouseEvent.BUTTON1: btn = "Left Button";
            break;
         case MouseEvent.BUTTON2: btn = "Middle Button";
            break;
         case MouseEvent.BUTTON3: btn = "Right Button";
            break;
      }
      System.out.println("Mouse " + btn+ " clicked : " + p);
     }
}
```


```
Mouse Left Button clicked : java.awt.Point[x=81,y=35]
Mouse Right Button clicked : java.awt.Point[x=135,y=57]
```

Java프로그래밍

다음시간안내

15강. JDBC 프로그래밍 (교재 14장)