Power Java

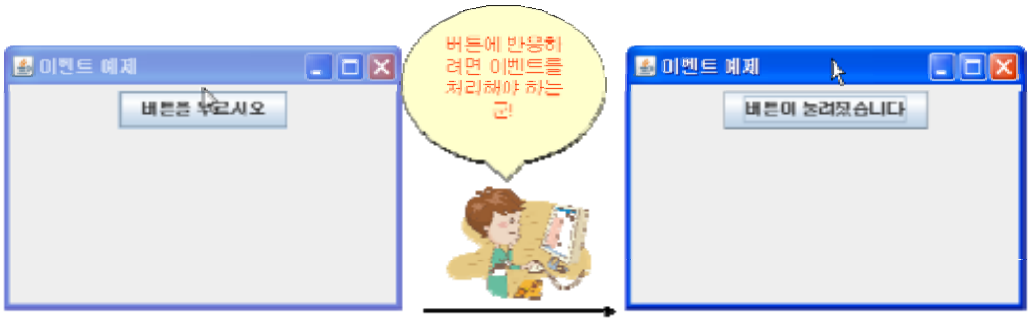
제18장 이벤트 처리

이번 장에서 학습할 내용

- 이벤트 처리의 개요
- 이벤트
- 액션 이벤트
- Key, Mouse, MouseMotion
- 어댑터 클래스

이번 장의 목표

- 버튼을 누르면 버튼의 텍스트가 변경되게 한다.



이벤트 처리 과정

(1) 버튼을 누른다. (2) 이벤트 객체가 발생. (3) 이벤트 처리

이벤트 처리 과정

(1) 이벤트를 발생하는 컴포넌트를 생성하여야 한다.

```
public class MyFrame extends JFrame { // 프레임을 상속하여서 MyFrame 선언
    private JButton button;    // 버튼을 참조하는 변수를 선언
    ...
    public MyFrame() // 생성자에서 컴포넌트를 생성하고 추가한다.
    {
            JPanel panel = new JPanel();        // 패널 생성
            button = new JButton("동작");       // 버튼 생성
            panel.add(button);                  // 버튼을 패널에 추가
            add(panel);                         // 패널을 프레임에 추가
            ...
    }
    ...
}
```


이벤트 처리 과정

(2) 이벤트 리스너 클래스를 작성한다.

```
class MyListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
            ...  // Action 이벤트를 처리하는 코드가 여기에 들어간다.
    }
}
```

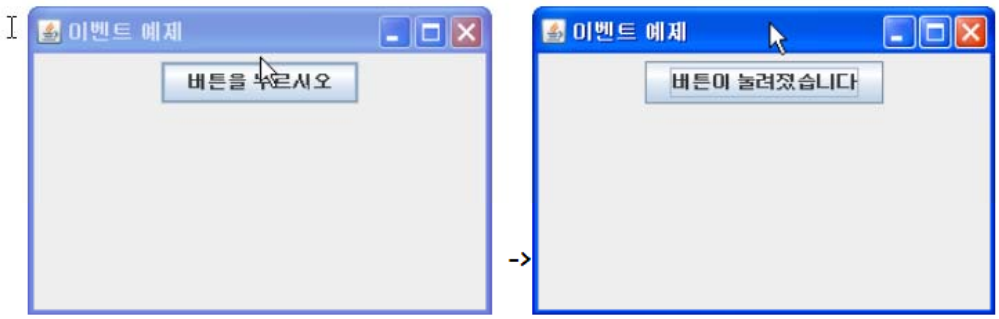
이벤트 처리 과정

(3) 이벤트 리스너를 이벤트 소스에 등록한다.

```
public class MyFrame extends JFrame { // 프레임을 상속하여서 MyFrame 선언
    ...
    public MyFrame()// 생성자에서 컴포넌트를 생성하고 추가한다.
    {
            JPanel panel = new JPanel();        // 패널 생성
            button = new JButton("동작");       // 버튼 생성
            button.addActionListener(new MyListener());        // 이벤트 리스너 등록
            panel.add(button);                  // 버튼을 패널에 추가
            add(panel);                         // 패널을 프레임에 추가
            ...
    }
    ...
}
```

이벤트 리스너를
컴포넌트에 붙인다.

프로그램 결과



이벤트 처리기를 어디에...

이벤트 처리기의 위치

- (1) 별도의 클래스로 이벤트 처리기를 작성: 앞의 예제
- (2) 내부 클래스로 이벤트 처리기를 작성
- (3) 프레임 클래스가 이벤트 처리기도 함께 구현

내부 클래스 사용 방법

- 만약 MyListener라는 클래스를 별도의 클래스로 하면 MyFrame 안의 멤버 변수들을 쉽게 사용할 수 없다.
- 일반적으로 MyListener 클래스를 내부 클래스로 만든다.

내부 클래스를 사용하는 버전

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

public class MyFrame extends JFrame {
	private JButton button;

	public MyFrame() {
		this.setSize(300, 200);
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setTitle("이벤트 예제");
		JPanel panel = new JPanel();
		button = new JButton("버튼을 누르시오");
		button.addActionListener(new MyListener());
		panel.add(button);
		this.add(panel);
		this.setVisible(true);
	}
```


내부 클래스를 사용하는 버전

```
	private class MyListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			if (e.getSource() == button) {
				button.setText("버튼이 눌려졌습니다");
			}
		}
	}
}
public class MyFrameTest {
	public static void main(String[] args) {
		MyFrame t = new MyFrame();
	}
}
```

내부 클래스

MyFrame에서 이벤트도 처리하는 방법

- 더 많이 사용되는 방법은 MyFrame 클래스가 JFrame을 상속받으면서 동시에 ActionListener 인터페이스도 구현하는 경우이다.

MyFrame이 이벤트도 처리

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

class MyFrame extends JFrame implements ActionListener {
    private JButton button;

    public MyFrame() {
        this.setSize(300, 200);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("이벤트 예제");
        JPanel panel = new JPanel();
        button = new JButton("버튼을 누르시오");
        button.addActionListener(this);
        panel.add(button);
        this.add(panel);
        this.setVisible(true);
    }
```

이벤트도 처리

MyFrame이 이벤트도 처리

```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == button) {
            button.setText("버튼이 눌려졌습니다");
        }
    }
}

public class MyFrameTest1 {
    public static void main(String[] args) {
        MyFrame t = new MyFrame();
    }
}
```


중간 점검 문제

1. 사용자가 버튼을 누르면 무엇이 발생하는가?
2. 이벤트를 처리하는 클래스를 무엇이라고 하는가?
3. 내부 클래스를 사용하는 장점은 무엇인가?

이벤트의 분류 #1

저수준 이벤트: Mouse, MouseMotion, Key, Component, Container, Focus, Window

의미적 이벤트: Action, Adjustment, Document, Item, Text

그림 18-3 이벤트의 종류

이벤트의 분류 #2

- 스윙 컴포넌트에 의하여 지원되는 이벤트는 크게 두 가지의 카테고리로 나누어진다.

이벤트
- 모든 컴포넌트가 지원하는 이벤트
- 일부 컴포넌트가 지원하는 이벤트

모든 컴포넌트가 지원하는 이벤트

이벤트 종류	설명
Component	컴포넌트의 크기나 위치가 변경되었을 경우 발생
Focus	키보드 입력을 받을 수 있는 상태가 되었을 때, 혹은 그반대의 경우에 발생
Container	컴포넌트가 컨테이너에 추가되거나 삭제될 때 발생
Key	사용자가 키를 눌렀을 때 키보드 포커스를 가지고 있는 객체에서 발생
Mouse	마우스 버튼이 클릭되었을 때, 또는 마우스가 객체의 영역으로 들어오거나 나갈 때 발생
MouseMotion	마우스가 움직였을 때 발생
MouseWheel	컴포넌트 위에서 마우스 휠을 움직이는 경우 발생
Window	윈도우에 어떤 변화가 있을 때 발생(열림, 닫힘, 아이콘화등)

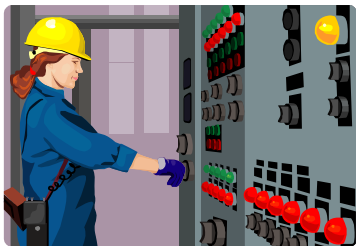
일부 컴포넌트가 지원하는 이벤트

이벤트 종류	설 명
Action	사용자가 어떤 동작을 하는 경우에 발생
Caret	텍스트 삽입점이 이동하거나 텍스트 선택이 변경되었을 경우 발생
Change	일반적으로 객체의 상태가 변경되었을 경우 발생
Document	문서의 상태가 변경되는 경우 발생
Item	선택 가능한 컴포넌트에서 사용자가 선택을 하였을 때 발생
ListSelection	리스트나 테이블에서 선택 부분이 변경되었을 경우에 발생

컴포넌트	이벤트						
	Action	Caret	Change	Document	Item	ListSelection	Window
버튼(button)	V		V		V		
체크박스(check box)	V		V		V		
색상 선택 (color chooser)			V				
콤보 박스(combo box)	V				V		
다이알로그(dialog)							V
파일 선택 (file chooser)	V						
프레임(frame)							V
리스트(list)						V	
메뉴 항목(menu item)	V		V		V		
진행바(progress bar)			V				
라디오 버튼 (radio button)	V		V		V		
슬라이더(slider)			V				
스피너(spinner)			V				
테이블(table)						V	
텍스트 영역(text area)		V		V			
텍스트 필드(text field)	V	V		V			

Action 이벤트

- 사용자가 버튼을 클릭하는 경우
- 사용자가 메뉴 항목을 선택하는 경우
- 사용자가 텍스트 필드에서 엔터키를 누르는 경우



액션 이벤트 예제

```
import javax.swing.*;
import java.awt.Color;
import java.awt.event.*;

class MyFrame extends JFrame {
	private JButton button1;
	private JButton button2;
	private JPanel panel;

	public MyFrame() {
		this.setSize(300, 200);
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setTitle("이벤트 예제");
		panel = new JPanel();
		button1 = new JButton("노란색");
		button1.addActionListener(new MyListener());
		panel.add(button1);
		button2 = new JButton("핑크색");
		button2.addActionListener(new MyListener());
		panel.add(button2);
		this.add(panel);
		this.setVisible(true);
	}
```

액션 이벤트 예제

```
	private class MyListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			if (e.getSource() == button1) {
				panel.setBackground(Color.YELLOW);
			} else if (e.getSource() == button2) {
				panel.setBackground(Color.PINK);
			}
		}

	}
}
public class MyFrameTest2 {
	public static void main(String[] args) {
		MyFrame t = new MyFrame();
	}
}
```

결과 화면

실행결과

이벤트 발생원의 식별

- getSource()메소드를 이용하여 이벤트를 발생시킨 객체를 식별한다.
- getId() 메소드를 이용하여 이벤트의 타입을 식별한다.
- getActionCommand()메소드를 이용하여 이벤트를 발생시킨 컴포넌트 이름을 식별한다.

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource () == button1){
            ...
    }
}
```

Key 이벤트

- KeyListener 인터페이스 구현

KeyListener 인터페이스

메소드	설 명
keyTyped(KeyEvent e)	사용자가 글자를 입력했을 경우에 호출
keyPressed(KeyEvent e)	사용자가 키를 눌렀을 경우에 호출
keyReleased(KeyEvent e)	사용자가 키에서 손을 떼었을 경우에 호출

키보드 이벤트 예제

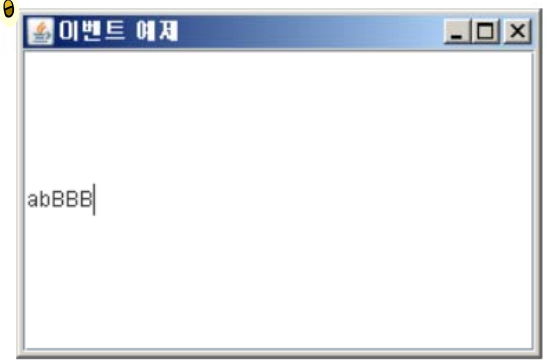
```
import javax.swing.*;

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

class MyFrame extends JFrame implements KeyListener { // (1)

    public MyFrame() {
        setTitle("이벤트 예제");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JTextField tf = new JTextField(20);
        tf.addKeyListener(this); // (2)

        add(tf);
        setVisible(true);

    }
```

키보드 이벤트 처리기를 붙인다.

액션 이벤트 예제

```
    public void keyTyped(KeyEvent e) { // (3)
        display(e, "KeyTyped ");
    }

    public void keyPressed(KeyEvent e) {
        display(e, "KeyPressed ");
    }

    public void keyReleased(KeyEvent e) {
        display(e, "Key Pressed ");
    }

    protected void display(KeyEvent e, String s) {
        char c = e.getKeyChar();
        int keyCode = e.getKeyCode();
        String modifiers = e.isAltDown() + " " + e.isControlDown() + " "
                + e.isShiftDown();
        System.out.println(s + " " + c + " " + keyCode + " " + modifiers);
    }
}
```


실행 결과




```
KeyPressed   a 65 false false false
KeyTyped   a 0 false false false
Key Pressed   a 65 false false false
KeyPressed   b 66 false false false
KeyTyped   b 0 false false false
Key Pressed   b 66 false false false
```

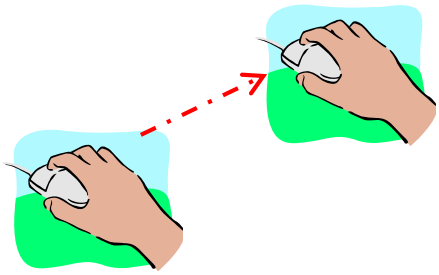
Mouse 이벤트

MouseListener 인터페이스

메소드	설 명
mouseClicked(MouseEvent e)	사용자가 컴포넌트를 마우스로 클릭한 경우에 호출된다.
mouseEntered(MouseEvent e)	마우스 커서가 컴포넌트의 경계안으로 커서가 들어가면 호출된다.
mouseExited(MouseEvent e)	마우스 커서가 컴포넌트의 경계밖으로 커서가 나가면 호출된다.
mousePressed(MouseEvent e)	마우스가 컴포넌트위에서 눌려지면 호출된다.
mouseReleased(MouseEvent e)	마우스가 컴포넌트위에서 떼어지면 호출된다.

MouseMotion 이벤트

메소드	설 명
mouseDragged(MouseEvent e)	마우스 버튼을 누른 채로 마우스를 이동하는 경우에 호출된다.
mouseMoved(MouseEvent e)	마우스 버튼을 누르지 않고 마우스를 이동하는 경우에 호출된다.



마우스 이벤트 예제

```
import javax.swing.*;

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

class MyFrame extends JFrame implements MouseListener, MouseMotionListener {

    public MyFrame() {
        setTitle("Mouse Event");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panel = new JPanel();
        panel.addMouseListener(this);
        panel.addMouseMotionListener(this);
        add(panel);
        setVisible(true);

    }
```

마우스 이벤트 처리기를 붙인다.

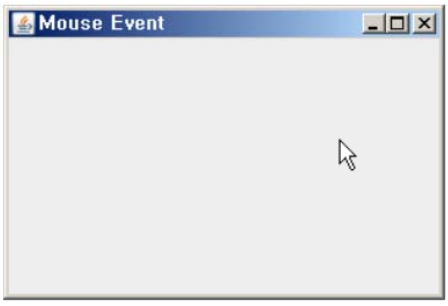
마우스 이벤트 예제

```
    public void mousePressed(MouseEvent e) {
        display("Mouse pressed (# of clicks: " + e.getClickCount() + ")", e);
    }

    public void mouseReleased(MouseEvent e) {
        display("Mouse released (# of clicks: " + e.getClickCount() + ")", e);
    }

    public void mouseEntered(MouseEvent e) {
        display("Mouse entered", e);
    }

    public void mouseExited(MouseEvent e) {
        display("Mouse exited", e);
    }

    public void mouseClicked(MouseEvent e) {
        display("Mouse clicked (# of clicks: " + e.getClickCount() + ")", e);
    }
```

마우스 이벤트 예제

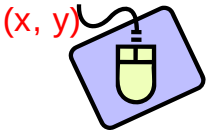
```
    protected void display(String s, MouseEvent e) {
        System.out.println(s + " X=" + e.getX() + " Y=" + e.getY());
    }

    public void mouseDragged(MouseEvent e) {
        display("Mouse dragged", e);
    }

    public void mouseMoved(MouseEvent e) {
        display("Mouse moved", e);

    }
}

public class MyFrameTest5 {
    public static void main(String[] args) {
        MyFrame f=new MyFrame();
    }
}
```


실행 결과




```
...
Mouse moved X=139 Y=78
Mouse pressed (# of clicks: 1) X=139 Y=78
Mouse released (# of clicks: 1) X=139 Y=78
Mouse clicked (# of clicks: 1) X=139 Y=78
Mouse moved X=139 Y=77
Mouse moved X=141 Y=77
```

마우스의 좌표를 얻으려면?



MouseEvent 클래스

메소드	설명
int getClickCount()	빠른 연속적인 클릭의 횟수를 반환한다. 예를 들어 2이면 더블 클릭을 의미한다.
int getX() int getY() Point getPoint()	이벤트가 발생했을 당시의 (x,y) 위치를 반환한다. 위치는 컴포넌트에 상대적이다.
int getXOnScreen() int getYOnScreen() int getLocationOnScreen()	절대 좌표 값 (x,y)을 반환한다. 이들 좌표값은 가상 화면에 상대적이다.
int getButton()	어떤 마우스 버튼의 상태가 변경되었는지를 반환한다. NOBUTTON, BUTTON1, BUTTON2, BUTTON3 중의 하나이다.
boolean isPopupTrigger()	마우스 이벤트가 팝업 메뉴를 나타나게 하면 true를 반환한다.
String getMouseModifiersText(int)	이벤트 도중의 수식키와 마우스 버튼을 기술하는 설명문을 반환한다.

어댑터 클래스

- 인터페이스의 경우, 모든 메소드를 구현하여야 한다.
- 어댑터 클래스(Adaptor Class)를 사용하면 원하는 메소드 만을 구현하는 것이 가능해진다

인터페이스	어댑터 클래스
ComponentListener	ComponentAdapter
ContainerListener	ContainerAdapter
FocusListener	FocusAdater
KeyListener	KeyAdapter
MouseListener	MouseAdapter
MouseMotionListener	MouseMotionAdapter
WindowListener	WindowAdapter

리스너를 사용하는 경우

```
public class KeyEventTest implements KeyListener {
        ...
        p.addKeyListener(this);
        ...
        public void keyTyped(KeyEvent e){
                if( e.getKeyChar() == 'x' ){
                        ...
                }
        }
        public void keyPressed(KeyEvent e){
        }
        public void keyReleased(KeyEvent e){
        }
}
```


어댑터를 사용하는 경우

```
public class KeyEventTest extends KeyAdaptor {
        public void keyTyped(KeyEvent e){
                if( e.getKeyChar() == 'x' ){
                        ...
                }
        }
}
```

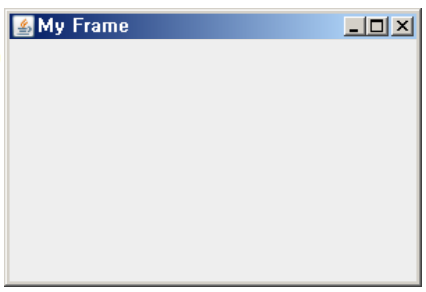
마우스 이벤트 예제

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.JFrame;

class MyFrame extends JFrame {
        public MyFrame() {
                setSize(300, 200);
                setTitle("My Frame");
                addWindowListener(new MyWindowAdaptor());
                setVisible(true);
        }

        class MyWindowAdaptor extends WindowAdapter {
                public void windowClosing(WindowEvent e) {
                        System.exit(0);
                }
        }
}
```

내부 클래스 정의

마우스 이벤트 예제

```
public class MyFrameTest6 {
        public static void main(String args[]) {
                MyFrame w = new MyFrame();
        }
}
```



Q & A