이벤트 처리

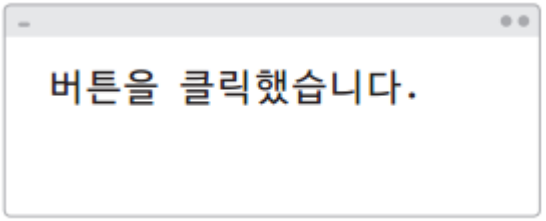
이벤트 구동 프로그래밍

- **이벤트의 개념과 처리 과정**
  - 윈도우 시스템에서 사용자의 움직임을 애플리케이션에 전달하는 일종의 신호
  - GUI 프로그램은 이벤트가 실행 흐름을 결정하는 이벤트 구동 방식
  - 이벤트 구동 프로그램의 이벤트 처리 과정

  - 이벤트 리스너는 발생한 이벤트를 처리하는 객체
  - 이벤트 핸들러는 이벤트를 처리하는 이벤트 리스너의 멤버 메서드

이벤트 구동 프로그래밍

- **이벤트 맛보기**
  - HelloEventDemo.java



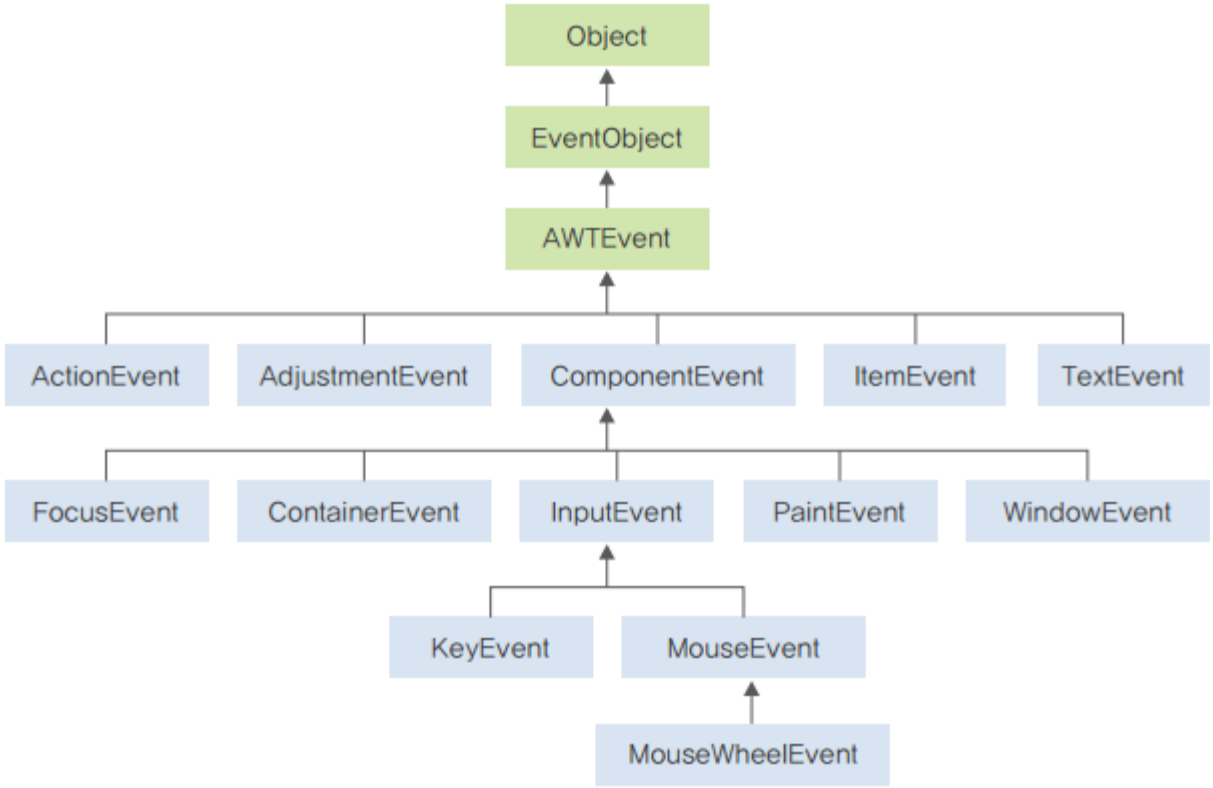
이벤트 구동 프로그래밍

- **이벤트 맛보기**
  - HelloEventDemo.java

```
3  import java.awt.event.*;
4  import javax.swing.*;
5
6  public class HelloEventDemo extends JFrame {
7      HelloEventDemo() {
8          setTitle("이벤트 맛보기");
9
10         ActionListener l = new ActionListener() {
11             public void actionPerformed(ActionEvent e) {
12                 System.out.println("버튼을 클릭했습니다.");
13             }
14         };
15         JButton b = new JButton("클릭");
16         b.addActionListener(l);
17         add(b);
18         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         setSize(260, 100);
20         setVisible(true);
21     }
22     public static void main(String[] args) {
23         new HelloEventDemo();
24     }
25 }
```

이벤트 클래스와 이벤트 리스너

- **이벤트 클래스 소개**
  - 이벤트 클래스는 이벤트 처리에 필요한 많은 정보를 제공. 예를 들어 사용자가 마우스를 클릭하면 클릭 여부, 클릭된 컴포넌트, 클릭된 위치 등과 같은 정보를 제공
  - 계층 구조 :



이벤트 클래스와 이벤트 리스너

- **이벤트 클래스의 종류**
  - 의미 이벤트
    - 버튼 클릭처럼 사용자가 의도하는 이벤트를 의미
    - 일부 스윙 컴포넌트에서만 발생
    - 예 : ActionEvent, AdjustmentEvent, ItemEvent, TextEvent 등
  - 저수준 이벤트
    - 의미 이벤트를 가능하게 하는 이벤트를 의미(의미 이벤트인 버튼 클릭은 마우스 이동, 마우스 누름, 마우스 놓기 등 여러 단계의 세부적인 이벤트로 구성)
    - 모든 스윙 컴포넌트에서 발생
    - 예 : ComponentEvent, ContainerEvent, FocusEvent, MouseEvent, KeyEvent 등

이벤트 클래스와 이벤트 리스너

- 주요 이벤트의 메서드와 상수

클래스	메서드 또는 상수	설명
EventObject	Object getSource( )	발생한 이벤트를 반환한다. 반환 타입이 Object 이므로 사용할 때 타입 변환이 필요하다.
AWTEvent	int getID( )	AWT 이벤트의 id 속성을 조사해서 반환한다.
	String paramString( )	이벤트의 상태를 문자열로 반환한다.
ItemEvent	static int DESELECTED	항목의 선택을 해제한다.
	static int ITEM_STATE_CHANGED	항목의 상태를 변경한다.
	static int SELECTED	항목을 선택한다.
	Object getItem( )	선택된 항목을 반환한다.
	int getStateChanged( )	변경된 상태를 반환한다.
WindowEvent	static int WINDOW_ACTIVATED	윈도우가 활성화된 상태이다.
	static int WINDOW_CLOSED	윈도우가 닫힌 상태이다.
	static int WINDOW_DEICONIZED	아이콘에서 윈도우로 변경된 상태이다.
	static int WINDOW_ICONIZED	윈도우가 아이콘으로 바뀐 상태이다.
	Window getWindow( )	이벤트가 발생한 윈도우를 반환한다.

이벤트 클래스와 이벤트 리스너

- **이벤트 리스너의 소개**
  - 이벤트 소스와 이벤트 리스너

이벤트 소스	이벤트	이벤트 리스너
버튼, 리스트, 메뉴 아이템, 텍스트 필드	ActionEvent	ActionListener
스크롤바	AdjustmentEvent	AdjustmentListener
체크 박스, 콤보 박스, 리스트	ItemEvent	ItemListener
컨테이너	ContainerEvent	ContainerListener
컴포넌트	ComponentEvent	ComponentListener
	FocusEvent	FocusListener
	KeyEvent	KeyListener
	MouseEvent	MouseListener, MouseMotionListener
윈도우	WindowEvent	WindowListener

이벤트 클래스와 이벤트 리스너

- **이벤트 리스너의 소개**
  - 주요 리스너 인터페이스와 추상 메서드

리스너 인터페이스	추상 메서드
ActionListener	void actionPerformed(ActionEvent)
ItemListener	void itemStateChanged(ItemEvent)
AdjustmentListener	void adjustmentValueChanged(AdjustmentEvent)
KeyListener	• void keyPressed(KeyEvent) • void keyReleased(KeyEvent) • void keyTyped(KeyEvent)
MouseListener	• void mousePressed(MouseEvent) • void mouseReleased(MouseEvent) • void mouseClicked(MouseEvent) • void mouseEntered(MouseEvent) • void mouseExited(MouseEvent)
MouseMotionListener	• void mouseDragged(MouseEvent) • void mouseMoved(MouseEvent)

이벤트 클래스와 이벤트 리스너

- **이벤트 리스너의 소개**
  - 이벤트 소스에 이벤트 리스너 등록

```
void addXXXListener(이벤트리스너객체);
```

컴포넌트에서 발생하는 이벤트 이름이다. ActionEvent면 XXX는 Action이며, MouseEvent면 XXX는 Mouse이다.

  - 이벤트 리스너는 매우 빠르게 처리되도록 가능한 짧게 작성될 필요
  - 이벤트 리스너가 하나의 스레드로 과다한 작업을 수행한다면 프로그램이 반응하지 않을 수도 있음
  - 이벤트 처리 시간이 길다면 별도의 스레드에 맡기는 것이 바람직

이벤트 클래스와 이벤트 리스너

- **이벤트 클래스와 주요 메서드**
  - ActionEvent

메서드	설명
String getActionCommand( )	액션과 관련된 명령어 문자열을 반환한다.
int getModifiers( )	액션이 발생할 때 눌린 변환키의 값을 반환한다.

  - KeyEvent

메서드	설명
char getKeyChar( )	키보드로 입력한 문자를 반환한다.
int getKeyCode( )	키보드로 입력한 문자의 코드 정수 값을 반환한다.

```
컴포넌트.requestFocus();
```

이벤트 클래스와 이벤트 리스너

- **이벤트 클래스와 주요 메서드**
  - MouseEvent

메서드	설명
int getButton( )	상태가 변경된 마우스 버튼을 반환한다.
int getClickCount( )	이벤트와 관련된 마우스의 클릭 횟수를 반환한다.
Point getLocationOnScreen( )	이벤트가 발생한 위치의 좌표를 반환한다.
static String getMouseModifiersText( )	마우스 버튼과 함께 누른 변환키의 텍스트를 반환한다.
int getX( )	이벤트가 발생할 때 마우스의 X 좌표를 반환한다.
int getY( )	이벤트가 발생할 때 마우스의 Y 좌표를 반환한다.

  - 마우스의 움직임을 추적할 때는 시스템에 상당한 부담을 주기 때문에 자바는 MouseListener 인터페이스와 별도로 MouseMotionListener 인터페이스로 구분해 제공

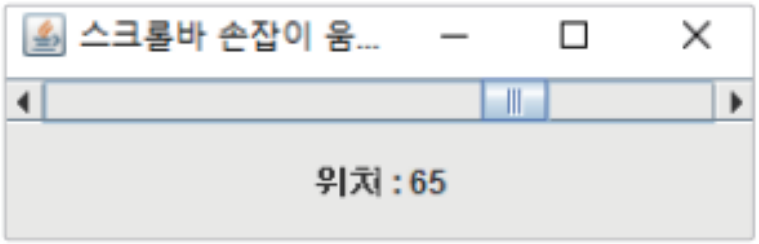
이벤트 클래스와 이벤트 리스너

- **이벤트 클래스와 주요 메서드**
  - AdjustmentEvent

메서드	설명
int getValue( )	이벤트의 현재 값을 반환한다.

이벤트 처리 응용

- **스크롤바 움직이기**
  - 예제 : AdjustmentEvent 처리
    AdjustmentListenerDemo.java



이벤트 처리 응용

- **스크롤바 움직이기**
  - 예제 : AdjustmentEvent 처리
    AdjustmentListenerDemo.java

```
 3 import javax.swing.*;
 4
 5 public class AdjustmentListenerDemo extends JFrame {
 6     AdjustmentListenerDemo() {
 7         setTitle("스크롤바 손잡이 움직이기");
 8
 9         JLabel label = new JLabel("", JLabel.CENTER);
10         JScrollBar bar = new JScrollBar(JScrollBar.HORIZONTAL);
11         bar.setValues(50, 10, 0, 100);
12         bar.addAdjustmentListener(e ->{
13             int v = e.getValue();
14             label.setText("위치 : " + v);
15         });
16         add("Center", label);
17         add("North", bar);
18
19         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         setSize(300, 100);
21         setVisible(true);
22     }
23
24     public static void main(String[] args) {
25         new AdjustmentListenerDemo();
26     }
27 }
```

어댑터 클래스

- **기초**
  - 개발자가 필요한 추상 메서드만 구현하면 되도록 리스너에 대응하는 어댑터 클래스를 제공
  - 어댑터 클래스는 리스너 인터페이스에 포함된 모든 추상 메서드를 빈 본체로 구현한 클래스에 불과
  - 예 :

```
public abstract class KeyAdapter implements KeyListener {
    void keyPressed(KeyEvent e) { }
    void keyReleased(KeyEvent e) { }
    void keyTyped(KeyEvent e) { }
}
```

본체는 비어 있다.

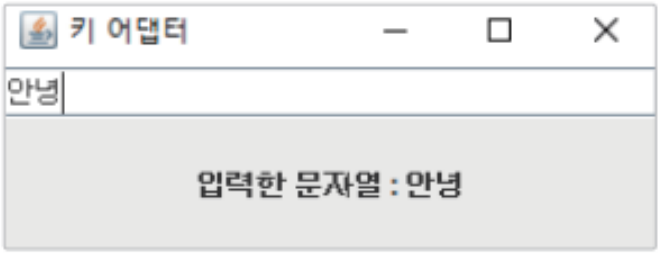
어댑터 클래스

- **기초**
  - 리스너 인터페이스와 대응하는 어댑터 클래스

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

어댑터 클래스

- **KeyAdapter 클래스**
  - 예제 : KeyAdapterDemo.java



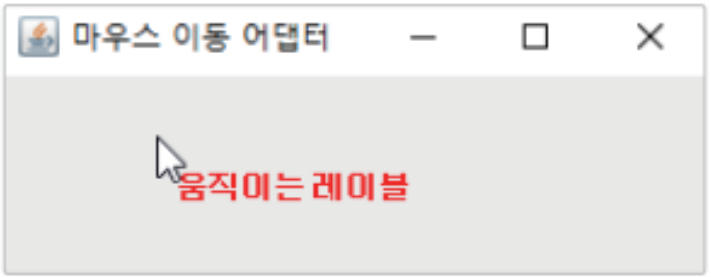
어댑터 클래스

- **KeyAdapter 클래스**
  - 예제 : KeyAdapterDemo.java

```
3  import java.awt.event.*;
4
5  import javax.swing.*;
6
7  public class KeyAdapterDemo extends JFrame {
8      public KeyAdapterDemo() {
9          setTitle("키 어댑터");
10         JLabel l = new JLabel("", JLabel.CENTER);
11         JTextField t = new JTextField(10);
12         add("North", t);
13         add("Center", l);
14         t.addKeyListener(new KeyAdapter() {
15             public void keyPressed(KeyEvent e) {
16                 if (e.getKeyCode() == KeyEvent.VK_ENTER) {
17                     l.setText("입력한 문자열 : " + t.getText());
18                 }
19             }
20         });
21
22         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23         setSize(300, 120);
24         setVisible(true);
25     }
26     public static void main(String[] args) {
27         new KeyAdapterDemo();
28     }
29 }
```

어댑터 클래스

- **MouseMotionAdapter 클래스**
  - 예제 : MouseMotionAdapterDemo.java



어댑터 클래스

- **MouseMotionAdapter 클래스**
  - 예제 : MouseMotionAdapterDemo.java

```
3  import java.awt.*;
4  import java.awt.event.*;
5  import javax.swing.*;
6  public class MouseMotionAdapterDemo extends JFrame {
7      MouseMotionAdapterDemo() {
8          setTitle("마우스 이동 어댑터");
9          JLabel label = new JLabel("움직이는 레이블");
10         label.setForeground(Color.RED);
11         add(label);
12         addMouseMotionListener(new MyMouseMotionAdapter(label));
13         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14         setSize(300, 120);
15         setVisible(true);
16     }
17     public static void main(String[] args) {
18         new MouseMotionAdapterDemo();
19     }
20 }
21 class MyMouseMotionAdapter extends MouseMotionAdapter {
22     JLabel label;
23     public MyMouseMotionAdapter(JLabel label) {
24         this.label = label;
25     }
26     public void mouseMoved(MouseEvent e) {
27         label.setLocation(e.getX(), e.getY() - 50);
28     }
29 }
```