이벤트 구동 프로그래밍

이벤트의 개념

윈도우 시스템에서 사용자의 움직임을 애플리케이션에 전달하는 일종의 신호이다.
GUI 프로그램은 이벤트가 실행 흐름을 결정하는 이벤트 구동 방식이다.
이벤트 리스너는 발생한 이벤트를 처리하는 객체이다.
이벤트 핸들러는 이벤트를 처리하는 이벤트 리스너의 멤버 메소드이다.

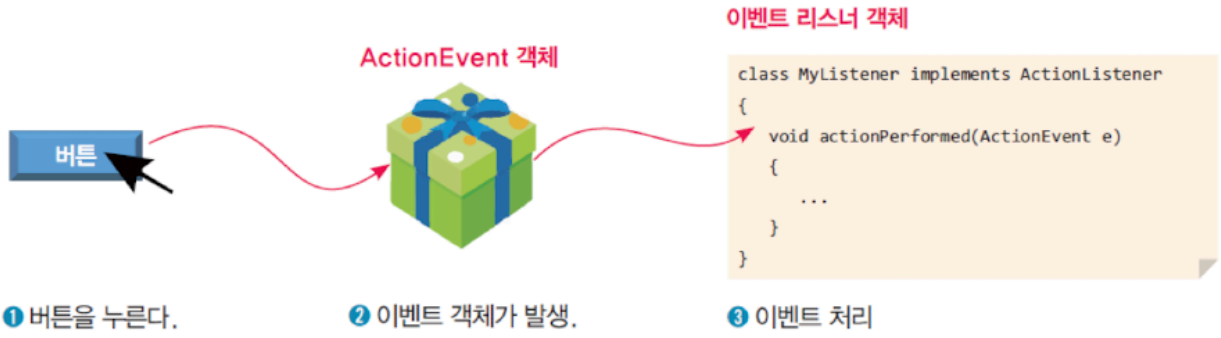
- 이벤트 구동 프로그램의 이벤트 처리과정

이벤트 처리 과정

1. 이벤트 리스너(하나의 객체 메소드)를 작성한다.
   어떤 클래스가 이벤트 리스너가 되기 위해서는 리스너 인터페이스를 구현해야 된다.

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

액션 이벤트가 발생하면 호출된다.



2. 이벤트 리스너를 컴포넌트에 등록한다.
   이벤트 리스너를 컴포넌트에 등록하는 단계이다.

```
public class MyFrame extends JFrame {        // 프레임을 상속하여서 MyFrame 선언
   ...
   public MyFrame()    // 생성자에서 컴포넌트를 생성하고 추가한다.
   {
      button = new JButton("동작");            // 버튼 생성
      button.addActionListener(new MyListener());
      ...
   }
}
```

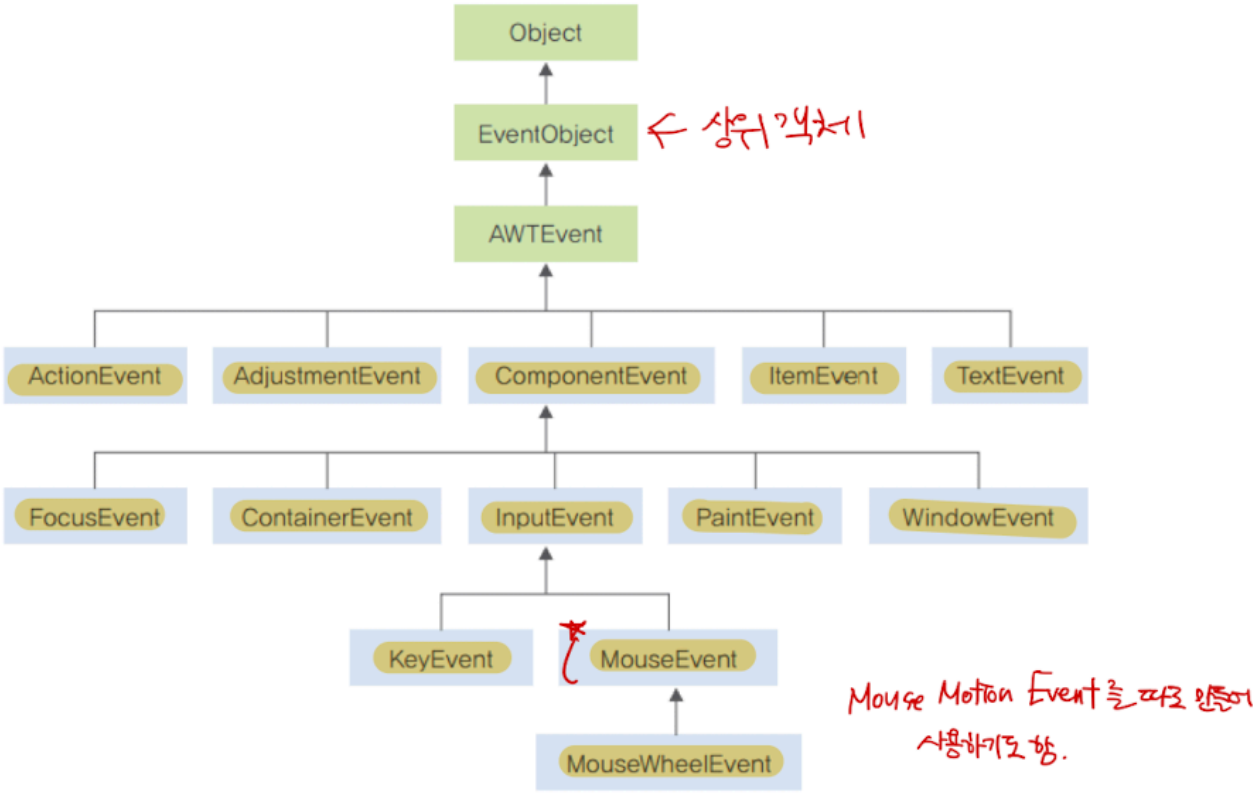
이벤트 리스너 객체를 new를 이용하여서 생성하고, 버튼에 이벤트 리스너 객체를 등록한다.

이벤트 클래스와 이벤트 리스너

이벤트 클래스 소개

이벤트 클래스는 이벤트 처리에 필요한 많은 정보를 제공한다.
예를 들어 사용자가 마우스를 클릭하면 클릭 여부, 클릭된 컴포넌트, 클릭된 위치 등과 같은 정보를 제공한다.

- 계층 구조



이벤트 클래스 종류

- 의미 이벤트

버튼 클릭처럼 사용자가 의도하는 이벤트를 의미한다.
일부 스윙 컴포넌트에서만 발생한다.
ex) ActionEvent, AdjustmentEvent, ItemEvent, TextEvent 등이 있다.

- 저수준 이벤트

의미 이벤트를 가능하게 하는 이벤트를 의미한다.(의미 이벤트인 버튼 클릭은 마우스 이동, 마우스 누름, 마우스 놓기 등 여러 단계의 세부적인 이벤트로 구성한다.)
모든 스윙 컴포넌트에서 발생한다.
ex) ComponentEvent, ContainerEvent, FocusEvent, MouseEvent, KeyEvent 등이 있다.

주요 이벤트의 메소드와 상수

클래스	메서드 또는 상수	설명
EventObject	Object getSource( )	발생한 이벤트를 반환한다. 반환 타입이 Object이므로 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 리스너 소개

이벤트 소스에 이벤트 리스너를 등록한다.

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

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

이벤트 리스너는 매우 빠르게 처리되도록 가능한 짧게 작성될 필요가 있다.
이벤트 리스너가 하나의 스레드로 과다한 작업을 수행한다면 프로그램이 반응하지 않을 수도 있다.
이벤트 처리 시간이 길다면 별도의 스레드에 맡기는 것이 바람직하다.

- 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

- 주요 리스너 인터페이스와 추상 메소드

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 주요 메소드

- ActionEvent

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

- KeyEvent

```
char getKeyChar()           // 키보드로 입력한 문자를 반환한다.
int getKeyCode()            // 키보드로 입력한 문자의 코드 정수 값을 반환한다.
```

- MouseEvent

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

```
int getButton()                         // 상태가 변경된 마우스 버튼을 반환한다.
int getClickCount()                     // 이벤트와 관련된 마우스의 클릭 횟수를 반환한다.
Point getLocationOnScreen()             // 이벤트가 발생한 위치의 좌표를 반환한다.
static String getMouseModifiers Text()  // 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.
```

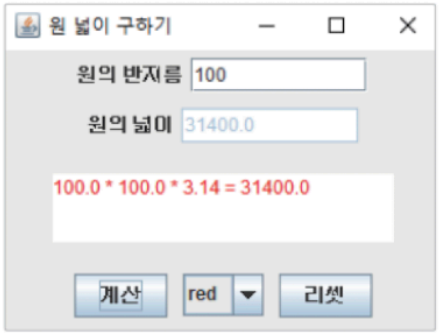
- AdjustmentEvent

```
int getValue()              // 이벤트의 현재 값을 반환한다.
```

이벤트 처리 응용

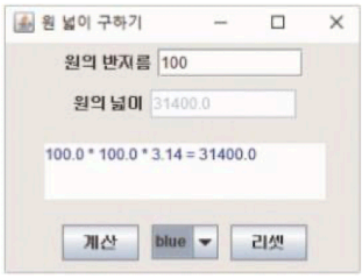
원의 넓이 구하기
이벤트 처리가 없는 경우

```java
ActionListener listener1 = e -> {
    if (e.getSource() == cal) {
        if (t1.getText().isEmpty())
            area.setText("반지름을 입력하세요.!!!");
        else {
            String s = t1.getText();
            double radius = Double.parseDouble(s);
            double result = radius * radius * 3.14;
            t2.setText("" + result);
            area.setText(radius + " * " + radius + " * 3.14 = " + result);
        }
    } else {
        t1.setText("");
        t2.setText("");
        area.setText("");
    }
};

cal.addActionListener(listener1);
reset.addActionListener(listener1);
```

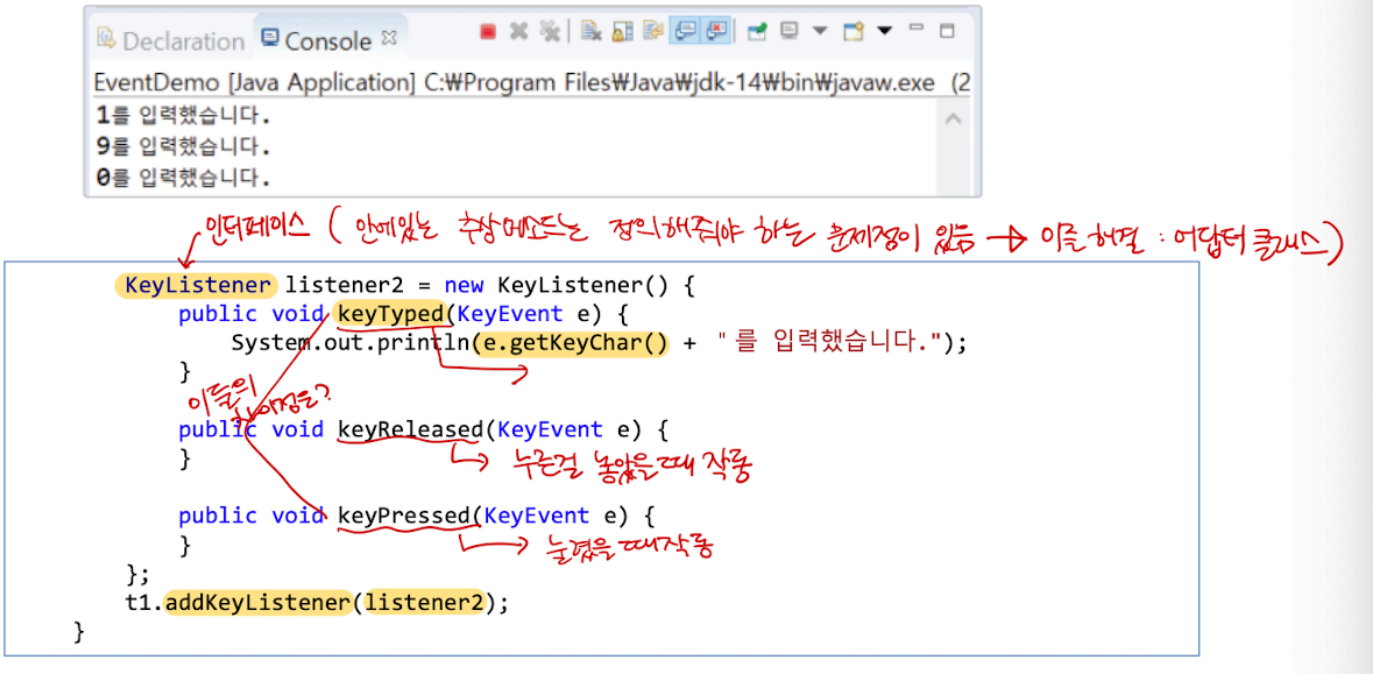


ItemEvent 처리

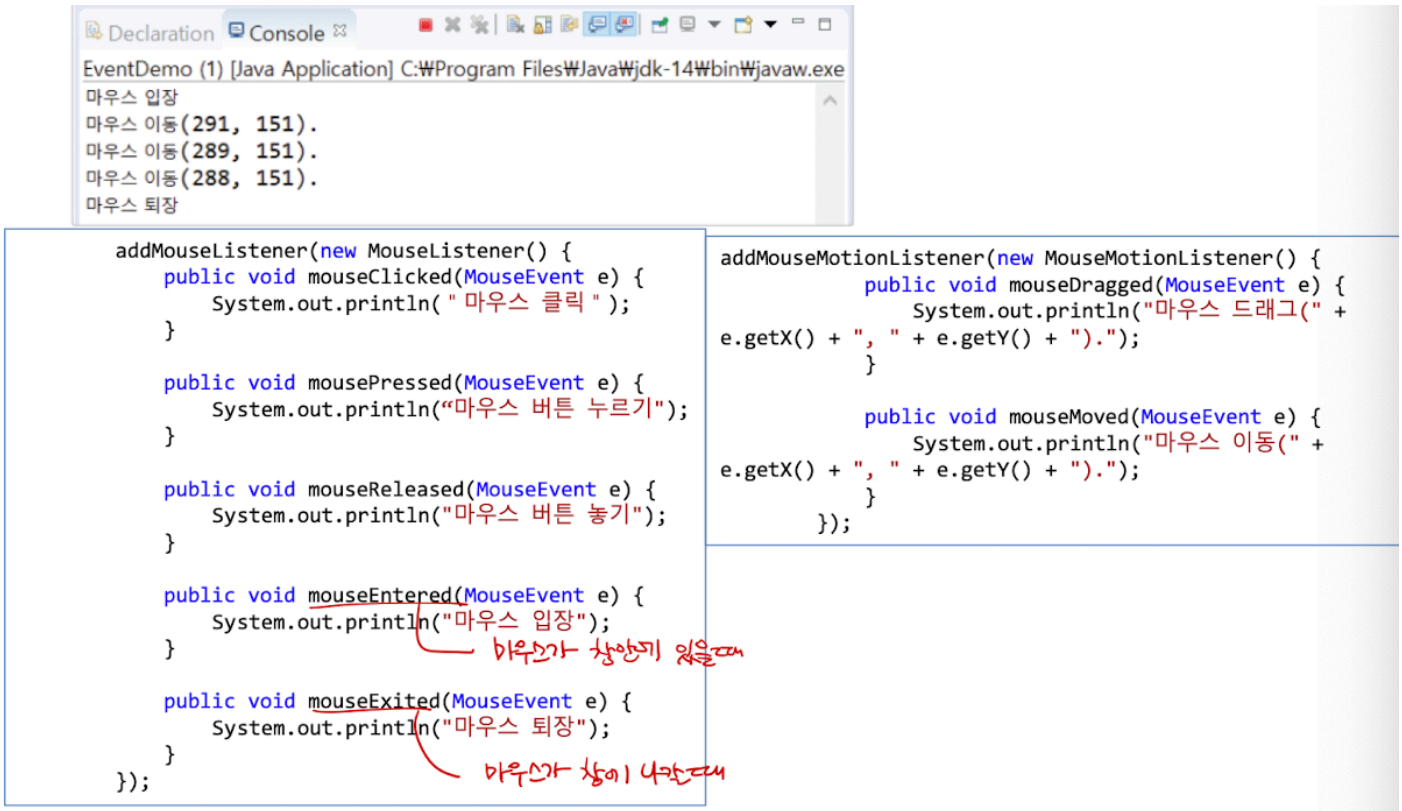
```java
cb.addItemListener(e -> {
    int index = ((JComboBox) cb).getSelectedIndex();
    if (index == 0)
        area.setForeground(Color.RED);
    else
        area.setForeground(Color.BLUE);
});
```



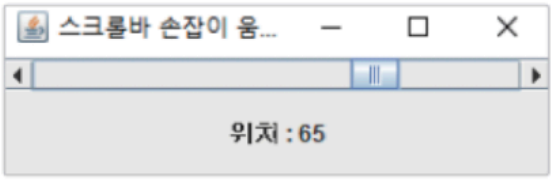
keyEvent 처리



MouseEvent처리



스크롤바 움직이기
AdjustmentEvent 처리




```
AdjustmentListenerDemo() {
    setTitle( " 스크롤바 손잡이 움직이기");

    JLabel label = new JLabel("", JLabel.CENTER);

    JScrollBar bar = new JScrollBar(JScrollBar.HORIZONTAL);
    bar.setValues(50, 10, 0, 100);
    bar.addAdjustmentListener(e -> {
        int v = e.getValue();
        label.setText( " 위치 : " + v);
    });

    add("Center", label);
    add("North", bar);

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(300, 100);
    setVisible(true);
}

public static void main(String[] args) {
    new AdjustmentListenerDemo();
}
```

어댑터 클래스

기초

인터페이스의 약점을 해결하기 위해 인터페이스를 위한 클래스이다.
개발자가 필요한 추상 메소드만 구현하면 되도록 리스너에 대응하는 어댑터 클래스를 제공한다.
어댑터 클래스는 리스너 인터페이스에 포함된 모든 추상 메소드를 빈 본체로 구현한 클래스에 불과하다.

- 예시

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

본체는 비어 있다.

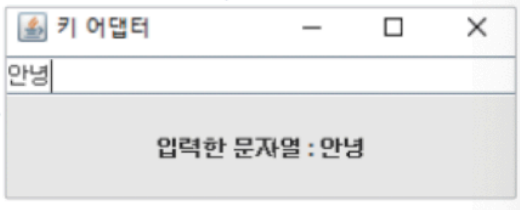
- 리스너 인터페이스와 대응하는 어댑터 클래스

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

- KeyAdapter 클래스 예시

```
public KeyAdapterDemo() {
    setTitle("키 어댑터");

    Jlabel l = new Jlabel(" ", Jlabel.CENTER);
    JTextField t = new JTextField(10);

    add("North", t);
    add("Center", l);

    t.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
            if (e.getKeyCode() == KeyEvent.VK_ENTER) {
                l.setText("입력한 문자열 : " + t.getText());
            }
        }
    });

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(300, 120);
    setVisible(true);
}

public static void main(String[] args) {
    new KeyAdapterDemo();
}
```

어댑터 클래스 사용해서 pressed만 사용해도 오류안남. 상수임. (alt, ctrl 등 있음.)



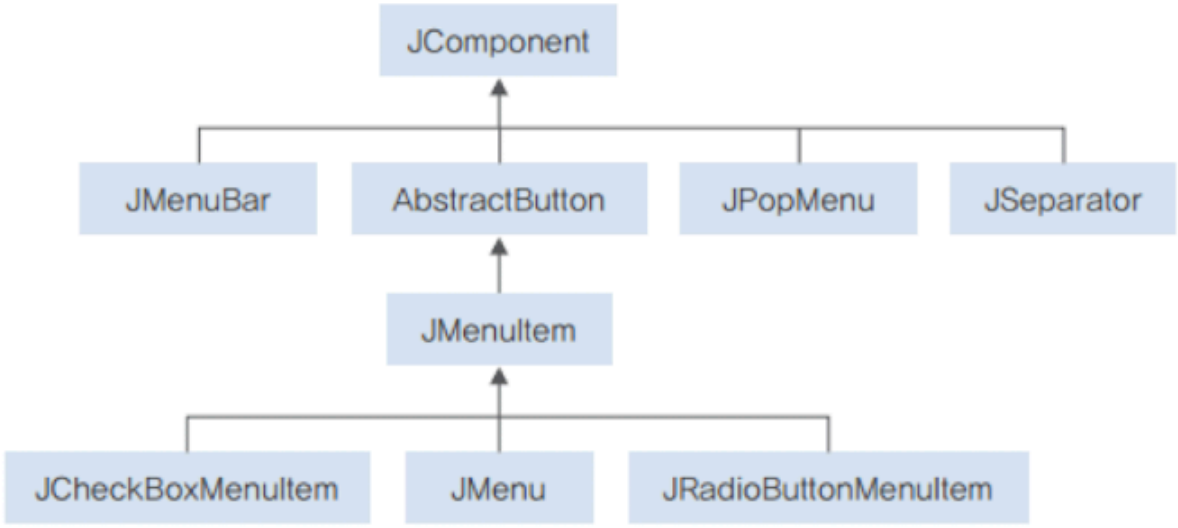
- MouseMotionAdapter 클래스 예시

```java
public class MouseMotionAdapterDemo extends JFrame {
    MouseMotionAdapterDemo() {
        setTitle("마우스 이동 어댑터");

        JLabel label = new JLabel("움직이는 레이블");
        label.setForeground(Color.RED);
        add(label);

        addMouseMotionListener(new MyMouseMotionAdapter(label));

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 120);
        setVisible(true);
    }

    public static void main(String[] args) {
        new MouseMotionAdapterDemo();
    }
}

class MyMouseMotionAdapter extends MouseMotionAdapter {
    JLabel label;

    public MyMouseMotionAdapter(JLabel label) {
        this.label = label;
    }

    public void mouseMoved(MouseEvent e) {
        label.setLocation(e.getX(), e.getY() - 50);
    }
}
```

이벤트와 메뉴

메뉴의 구성

메뉴의 계층 구조



메뉴 구성 순서

1. 메뉴 바 생성

```
JMenuBar mb = JMenuBar();
```

2. 메뉴바에 메뉴 추가

```
JMenu menu = new JMenu("File");
mb.add(menu);
```

3. 메뉴에 메뉴 항목 추가

```
JMenuItem item = new JMenuItem("New File");
menu.add(item);
```

4. 프레임에 메뉴바 부착

```
frame.setJMenuBar(mb);
```

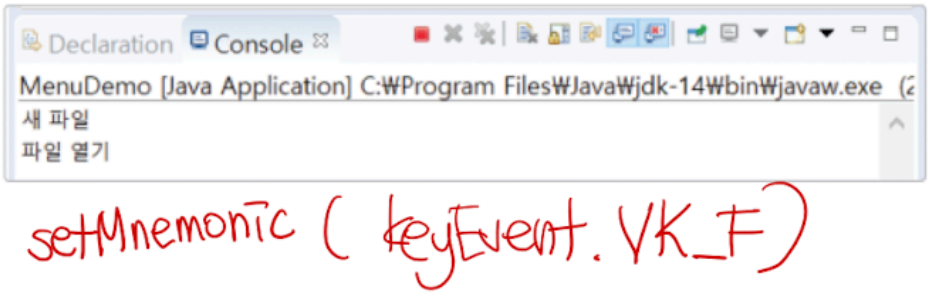
전형적인 메뉴 애플리케이션

```
public class MenuDemo extends JFrame implements ActionListener {
  MenuDemo() {
     setTitle("메뉴 구성하기");
     setSize(250, 170);
     makeMenu();
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     setVisible(true);
  }

  void makeMenu() {
     ...
  }

  public static void main(String[] args) {
     new MenuDemo();
  }

  public void actionPerformed(ActionEvent e) {
     ...
  }
}
```

메뉴 항목을 선택할 때 ActionEvent를 처리하

메뉴 만들기와 관련된 코드를 포함한다.

응용



setMnemonic (keyEvent.VK_F)