제 10 장 자바의 이벤트 처리

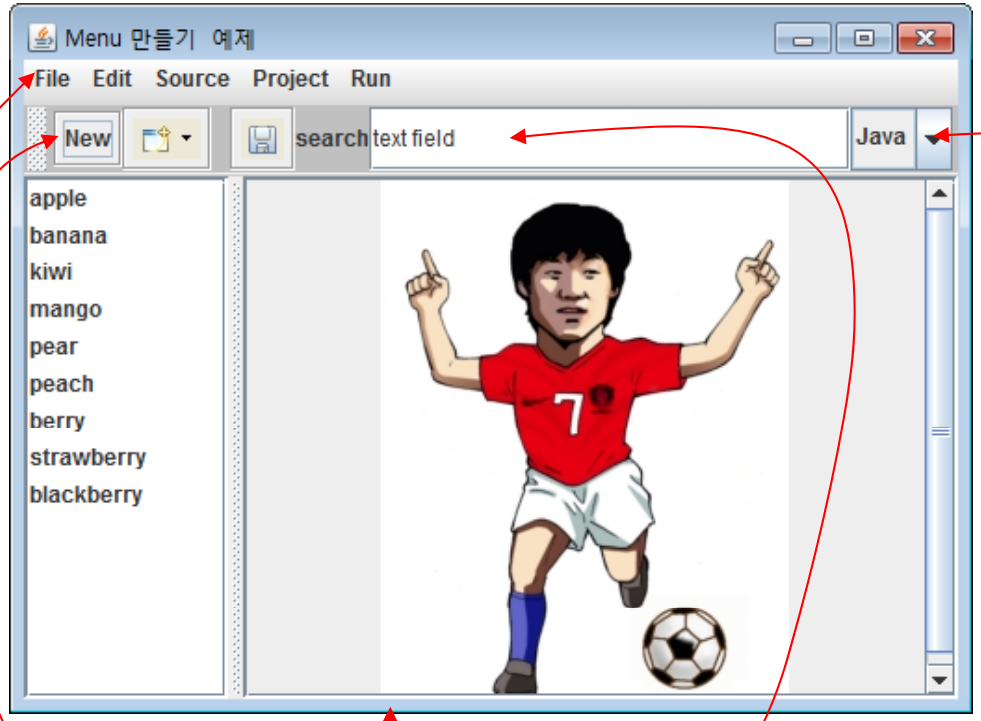
GUI 응용 프로그램 => 이벤트 기반 프로그래밍

- 이벤트 기반 프로그래밍(Event Driven Programming)
  - 이벤트의 발생에 의해 프로그램 흐름이 결정되는 방식
    - 이벤트가 발생하면 이벤트를 처리하는 루틴(이벤트 리스너) 실행
    - 프로그램 내의 어떤 코드가 언제 실행될 지 아무도 모름, 이벤트의 발생에 의해 전적으로 결정
  - 반대되는 개념 : 배치 실행(batch programming)
    - 프로그램의 개발자가 프로그램의 흐름을 결정하는 방식

- 이벤트 종류
  - 사용자의 입력 : 마우스 드래그, 마우스 클릭, 키보드 누름 등
  - 센서의 입력, 네트워크로부터 데이터 송수신
  - 다른 응용프로그램이나 다른 스레드로부터의 메시지

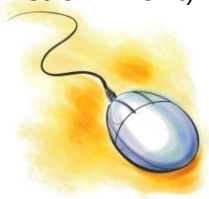
- 이벤트 기반 프로그램의 구조
  - 프로그램에서 처리하고자 하는 이벤트의 **이벤트 처리 리스너 들의 집합**

이벤트의 실제 예



마우스로 메뉴 선택
(Mouse Event, Action Event)

마우스로 버튼 클릭
(Mouse Event, Action Event)



마우스로 윈도우 크기 조절
(Mouse Event, Container Event)

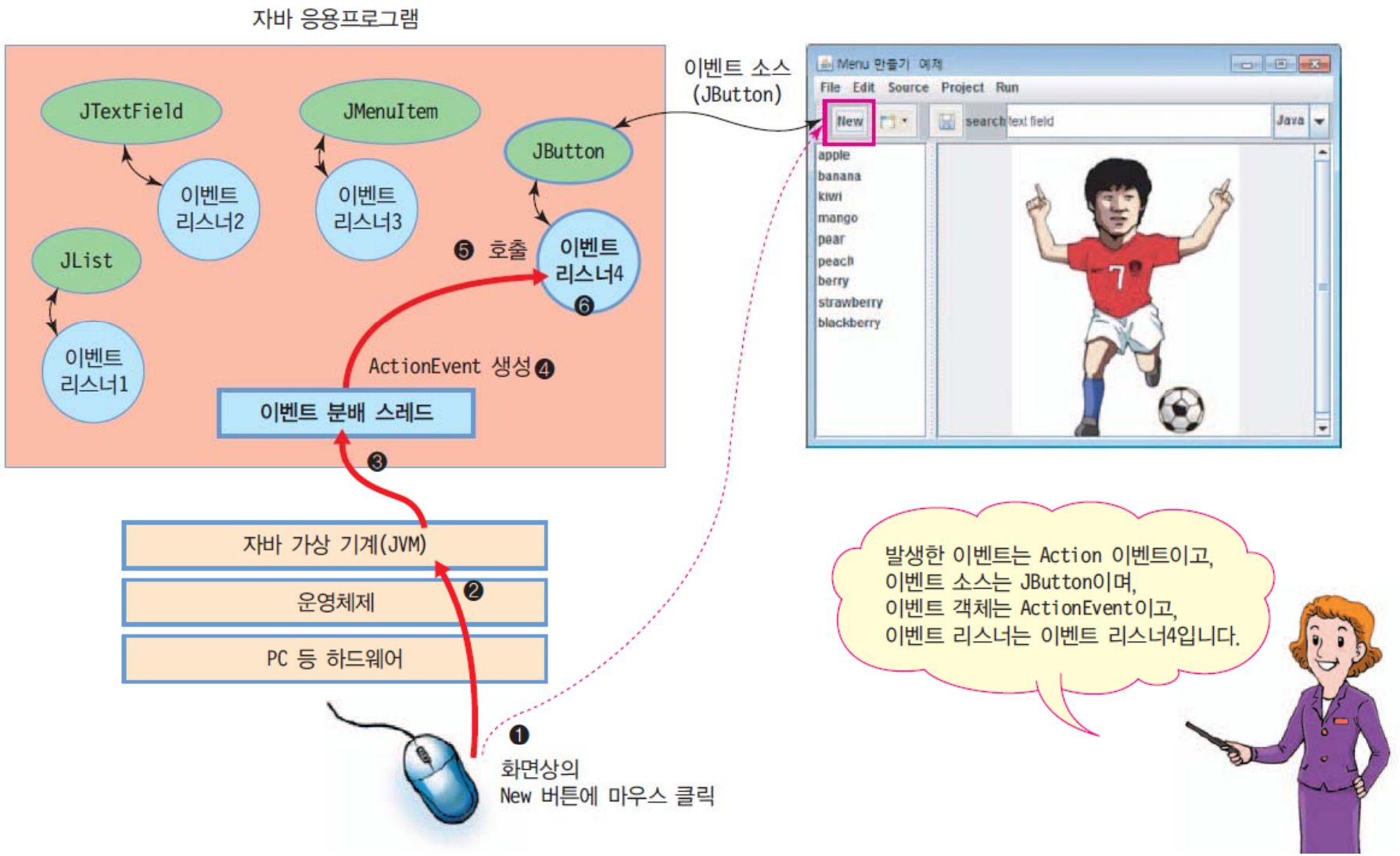
키보드 입력
(Key Event)

키보드로 메뉴 선택
(Key Event, Action Event)

GUI 응용 프로그램 => 이벤트 기반 프로그래밍

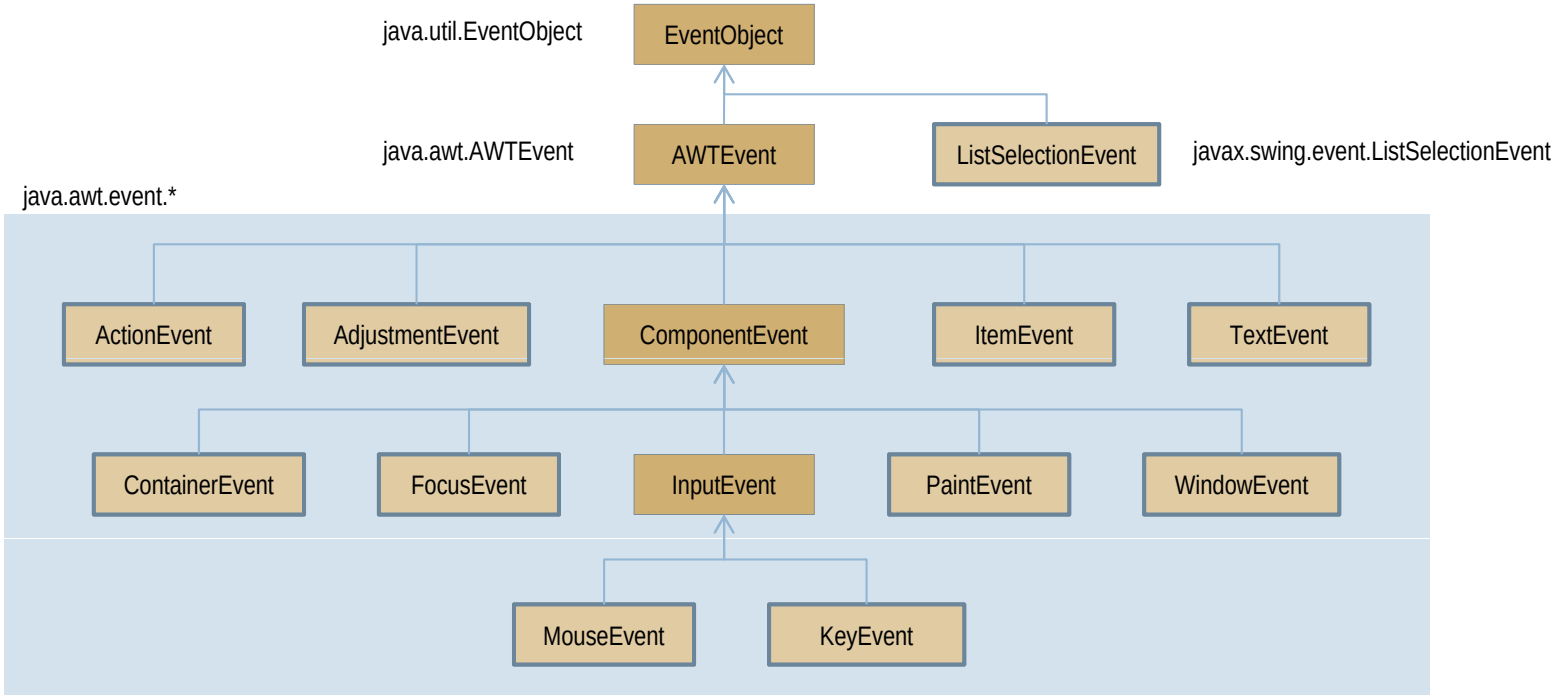
- 이벤트 관련 요소
  - 이벤트 소스(Event Source)
    - 이벤트를 발생시킨 GUI Component
  - 이벤트 객체
    - 이벤트 발생시 이벤트 종류, 이벤트 소스, 이벤트가 발생한 화면의 좌표, 마우스 버튼 종류, 눌려진 키의 코드값 등 이벤트에 대한 여러 속성값을 가진 객체
  - 이벤트 리스너(Event Listener)
    - 이벤트를 처리하는 코드로서 컴포넌트에 등록되어야 작동 가능함
  - 이벤트 분배 스레드
    - 이벤트 기반 프로그래밍의 핵심 요소로서 무한 루프를 실행하는 스레드
    - JVM으로부터 이벤트 발생을 통지 받으면 이벤트 소소와 이벤트 종류를 결정하고 이에 따라 이벤트 객체 생성, 이벤트 리스너 호출

자바의 이벤트 기반 GUI 응용프로그램 구성



이벤트 객체

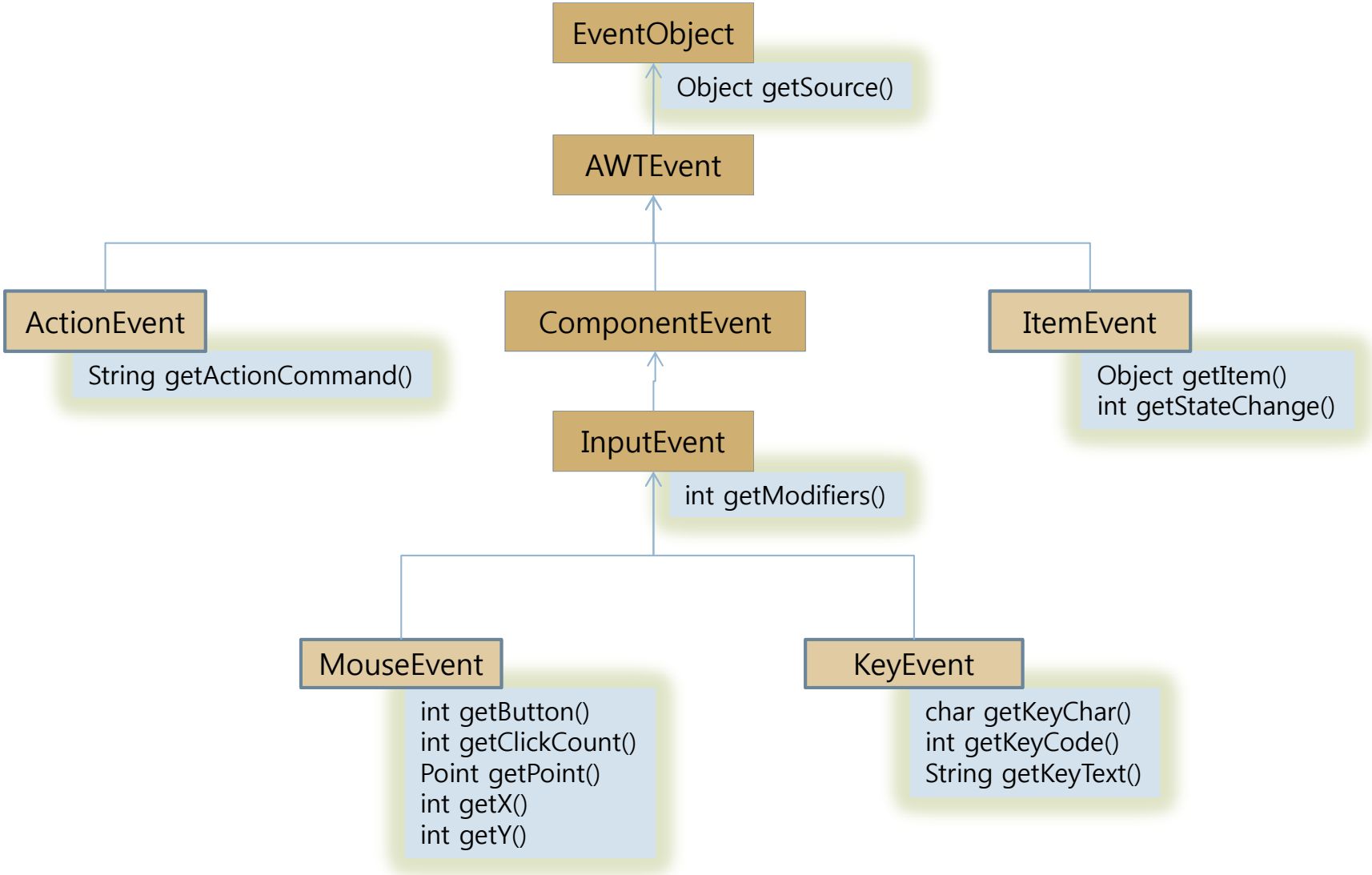
- 이벤트 객체란?
  - 이벤트가 발생할 때, 발생한 이벤트에 관한 정보를 가진 객체
    - 이벤트를 처리하는 응용프로그램에 다음 패키지가 포함되어야 함
      - *import java.awt.event.*;*
      - *import javax.swing.event.*;*
  - 이벤트 리스너에 전달됨
    - 이벤트 리스너 코드에서 이벤트가 발생한 상황을 파악할 수 있게 함
- 이벤트 객체의 종류



이벤트 객체에 포함된 정보

- 이벤트 객체가 포함하는 정보
  - 이벤트 종류
  - 이벤트 소스
  - 이벤트가 발생한 화면 좌표
  - 이벤트가 발생한 컴포넌트 내 좌표
  - 버튼이나 메뉴 아이템에 이벤트가 발생한 경우 버튼이나 메뉴 아이템의 문자열
  - 클릭된 마우스 버튼 번호
  - 마우스의 클릭 횟수
  - 키가 눌러졌다면 키의 코드 값과 문자 값
  - 체크박스, 라디오버튼 등과 같은 컴포넌트에 이벤트가 발생하였다면 체크 상태
- 이벤트에 따라 조금씩 다른 정보 포함
  - ActionEvent 객체 : 액션 문자열
  - MouseEvent 객체 : 마우스의 위치 정보, 마우스 버튼, 함께 눌러진 키 정보 등
  - ItemEvent 객체 : 아이템의 체크 상태
- 이벤트 소스 알아 내기
  - Object EventObject.getSource()
    - 발생한 이벤트의 소스 컴포넌트의 레퍼런스를 리턴
    - Object 타입으로 리턴하므로 캐스팅하여 사용
    - 모든 이벤트 객체에 대해 적용

이벤트 객체의 메소드


EventObject
Object getSource()
AWTEvent
ActionEvent
String getActionCommand()
ComponentEvent
ItemEvent
Object getItem()
int getStateChange()
InputEvent
int getModifiers()
MouseEvent
int getButton()
int getClickCount()
Point getPoint()
int getX()
int getY()
KeyEvent
char getKeyChar()
int getKeyCode()
String getKeyText()

이벤트 객체와 이벤트 소스

이벤트 객체	이벤트 소스	이벤트가 발생하는 경우
ActionEvent	JButton	마우스나 키로 버튼 선택
	JList	리스트 아이템을 더블클릭하여 리스트 아이템 선택
	JMenuItem	메뉴 아이템 선택
	JTextField	텍스트 입력 중 〈Enter〉 키 입력
ItemEvent	JCheckBox	체크박스의 선택 혹은 해제
	JCheckBoxMenuItem	체크박스 메뉴 아이템의 선택 혹은 해제
	JList	리스트 아이템 선택
KeyEvent	Component	키가 눌러지거나 눌러진 키가 떼어질 때
MouseEvent	Component	마우스 버튼이 눌러지거나 떼어질 때, 마우스 버튼이 클릭될 때, 컴포넌트 위에 마우스가 올라갈 때, 올라간 마우스가 내려올 때, 마우스가 드래그될 때, 마우스가 단순히 움직일 때
FocusEvent	Component	컴포넌트가 포커스를 받거나 잃을 때
TextEvent	TextField	텍스트 변경
	TextArea	텍스트 변경
WindowEvent	Window	Window를 상속받는 모든 컴포넌트에 대해 윈도우 활성화, 비활성화, 아이콘화, 아이콘에서 복구, 윈도우 열기, 윈도우 닫기, 윈도우 종료
AdjustmentEvent	JScrollBar	스크롤바를 움직일 때
ComponentEvent	Component	컴포넌트가 사라지거나, 나타나거나, 이동하거나, 크기가 변경될 때
ContainerEvent	Container	Container에 컴포넌트의 추가 혹은 삭제

이벤트 리스너(Event Listener)

- 이벤트 리스너란?
  - 이벤트를 처리하는 코드
  - 클래스로 작성
- JDK에서 이벤트 리스너 작성을 위한 인터페이스(interface) 제공
  - 개발자가 리스너 인터페이스의 추상 메소드 구현
    - 이벤트가 발생하면 자바 플랫폼은 리스너 인터페이스의 추상 메소드 호출
  - 예) ActionListener 인터페이스

```
interface ActionListener { // 아래 메소드를 개발자가 구현해야 함
   public void actionPerformed(ActionEvent e); // Action 이벤트 발생시 호출됨
}
```

  - 예) MouseListener 인터페이스

```
interface MouseListener { // 아래의 5개 메소드를 개발자가 구현해야 함
   public void mousePressed(MouseEvent e); // 마우스 버튼이 눌러지는 순간 호출
   public void mouseReleased(MouseEvent e); // 눌러진 마우스 버튼이 떼어지는 순간 호출
   public void mouseClicked(MouseEvent e); // 마우스가 클릭되는 순간 호출
   public void mouseEntered(MouseEvent e); // 마우스가 컴포넌트 위에 올라가는 순간 호출
   public void mouseExited(MouseEvent e); // 마우스가 컴포넌트 위에서 내려오는 순간 호출
}
```

이벤트 리스너 등록

- 이벤트 리스너 등록
  - 이벤트를 받아 처리하고자 하는 컴포넌트에 이벤트 리스너 등록
- 이벤트 리스너 등록 메소드
  - Component.addXXXListener(listener)
    - xxx : 이벤트 명
    - listener : 이벤트 리스너 객체
    - 예) addMouseListener(), addActionListener(), addFocusListener() 등
- 이벤트 리스너가 등록된 컴포넌트에만 이벤트 전달
  - 이벤트 리스너가 등록된 컴포넌트만 이벤트 리스너 코드 작동

리스너 인터페이스와 메소드

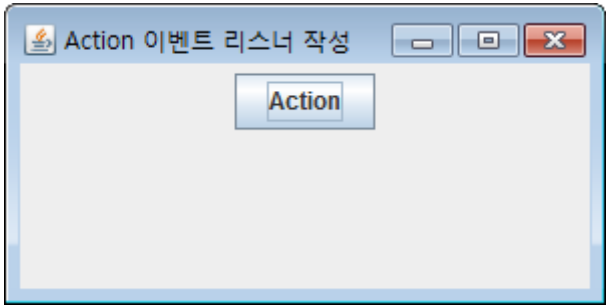
이벤트 종류	리스너 인터페이스	리스너의 추상 메소드	메소드가 호출되는 경우
Action	ActionListener	void actionPerformed(ActionEvent)	Action 이벤트가 발생하는 경우
Item	ItemListener	void itemStateChanged(ItemEvent)	Item 이벤트가 발생하는 경우
Key	KeyListener	void keyPressed(KeyEvent)	모든 키에 대해 키가 눌러질 때
		void keyReleased(KeyEvent)	모든 키에 대해 눌러진 키가 떼어질 때
		void keyTyped(KeyEvent)	유니코드 키가 입력될 때
Mouse	MouseListener	void mousePressed(MouseEvent)	마우스 버튼이 눌러질 때
		void mouseReleased(MouseEvent)	눌러진 마우스 버튼이 떼어질 때
		void mouseClicked(MouseEvent)	마우스 버튼이 클릭될 때
		void mouseEntered(MouseEvent)	마우스가 컴포넌트 위에 올라올 때
		void mouseExited(MouseEvent)	컴포넌트 위에 올라온 마우스가 컴포넌트를 벗어날 때
Mouse	MouseMotionListener	void mouseDragged(MouseEvent)	마우스를 컴포넌트 위에서 드래그할 때
		void mouseMoved(MouseEvent)	마우스가 컴포넌트 위에서 움직일 때
Focus	FocusListener	void focusGained(FocusEvent)	컴포넌트가 포커스를 받을 때
		void focusLost(FocusEvent)	컴포넌트가 포커스를 잃을 때
Text	TextListener	void textValueChanged(TextEvent)	텍스트가 변경될 때
Window	WindowListener	void windowOpened(WindowEvent)	윈도우가 생성되어 처음으로 보이게 될 때
		void windowClosing(WindowEvent)	윈도우의 시스템 메뉴에서 윈도우 닫기를 시도할 때
		void windowIconfied(WindowEvent)	윈도우가 아이콘화될 때
		void windowDeiconfied(WindowEvent)	아이콘 상태에서 원래 상태로 복귀할 때
		void windowClosed(WindowEvent)	윈도우가 닫혔을 때
		void windowActivated(WindowEvent)	윈도우가 활성화될 때
		void windowDeactivated(WindowEvent)	윈도우가 비활성화될 때
Adjustment	AdjustmentListener	void adjustmentValueChanged(AdjustmentEvent)	스크롤바를 움직일 때
Component	ComponentListener	void componentHidden(ComponentEvent)	컴포넌트가 보이지 않는 상태로 될 때
		void componentShown(ComponentEvent)	컴포넌트가 보이는 상태로 될 때
		void componentResized(ComponentEvent)	컴포넌트의 크기가 변경될 때
		void componentMoved(ComponentEvent)	컴포넌트의 위치가 변경될 때
Container	ContainerListener	void componentAdded(ContainerEvent)	컴포넌트가 컨테이너에 추가될 때
		void componentRemoved(ContainerEvent)	컴포넌트가 컨테이너에서 삭제될 때

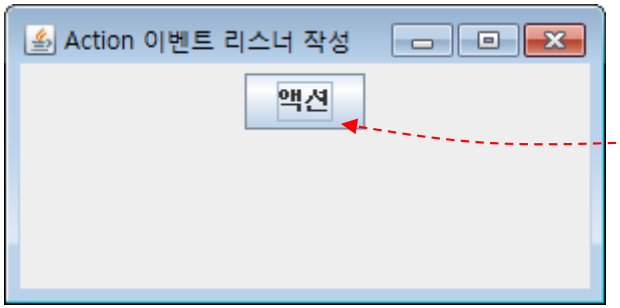
이벤트 리스너 작성 예


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

public class ListenerSample extends JFrame {
  ListenerSample () {
    setTitle("Action 이벤트 리스너 작성");
    setLayout(new FlowLayout());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JButton btn = new JButton("Action");
    MyActionListener listener = new MyActionListener ();
    btn.addActionListener(listener );
    add(btn);
    setSize(300,150);
    setVisible(true);
  }
  public static void main(String [] args) {
    new ListenerSample ();
  }
}
class MyActionListener implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    JButton b = (JButton)e.getSource();
    if(b.getText().equals("Action"))
      b.setText("액션");
    else
      b.setText("Action);
  }
}
```

Mouse 이벤트 리스너 객체 생성
Mouse 이벤트 리스너 등록

이벤트 리스너 등록



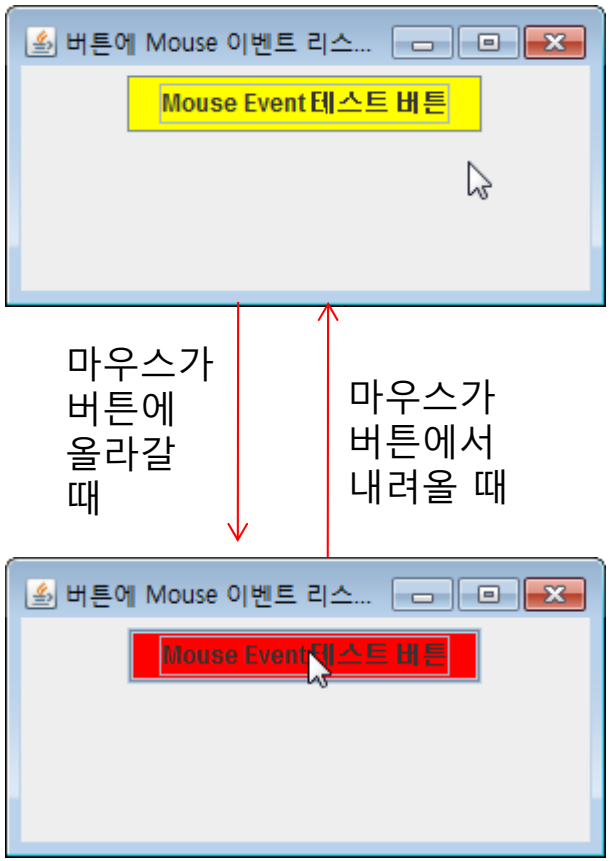


버튼의 문자열 변경

이벤트 리스너 구현

예제 10-1 : 버튼이 Mouse 이벤트를 처리하는 예제

초기 상태




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

public class ListenerMouseEx extends JFrame {
    ListenerMouseEx() {
        setTitle("버튼에 Mouse 이벤트 리스너 작성");
        setLayout(new FlowLayout());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton btn = new JButton("Mouse Event 테스트 버튼");
        btn.setBackground(Color.YELLOW);
        MyMouseListener listener = new MyMouseListener();
        btn.addMouseListener(listener);
        add(btn);
        setSize(300,150);
        setVisible(true);
    }
    public static void main(String [] args) {
        new ListenerMouseEx();
    }
}

class MyMouseListener implements MouseListener {
    public void mouseEntered(MouseEvent e) {
        JButton btn = (JButton)e.getSource();
        btn.setBackground(Color.RED);
    }
    public void mouseExited(MouseEvent e) {
        JButton btn = (JButton)e.getSource();
        btn.setBackground(Color.YELLOW);
    }
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
}
```

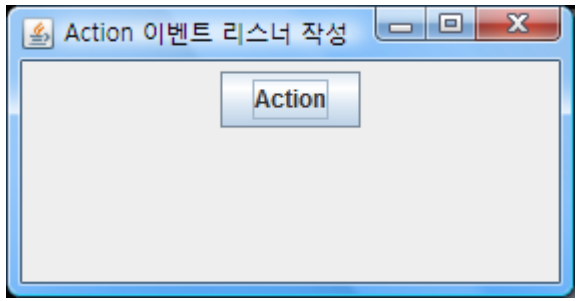
Tip : 리스너 등록 메소드가 addXXXListener인 이유?

- 컴포넌트는 다른 이벤트에 대한 리스너를 동시에 가질 수 있다.
  - JButton.addActionListener(); // Action 리스너
  - JButton.addKeyListener(); // Key 리스너
  - JButton.addFocusListener(); // Focus 리스너
  - ~~Jbutton.addTextListener(); // Test 리스너 (X)~~
- 컴포넌트는 한 이벤트에 대해 여러 개의 리스너를 동시에 가질 수 있다.
  - JButton.addActionListener(new MyButtonListener1());
  - JButton.addActionListener(new MyButtonListener2());
  - JButton.addActionListener(new MyButtonListener3());
  - 이때, 리스너는 등록된 반대 순으로 모두 실행된다.

이벤트 리스너 작성 방법

- 3 가지 방법
  - 독립 클래스로 작성
    - 이벤트 리스너를 완전한 클래스로 작성
    - 이벤트 리스너를 여러 곳에서 사용할 때 적합
  - 내부 클래스(inner class)로 작성
    - 클래스 안에 멤버처럼 클래스 작성
    - 이벤트 리스너를 특정 클래스에서만 사용할 때 적합
  - 익명 클래스(anonymous class)로 작성
    - 클래스의 이름 없이 간단히 리스너 작성
    - 클래스 조차 만들 필요 없이 리스너 코드가 간단한 경우에 적합

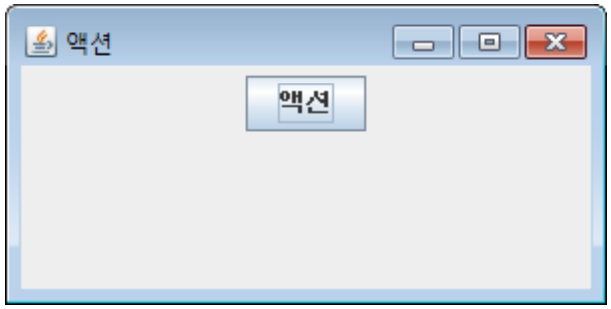
독립 클래스로 리스너 작성

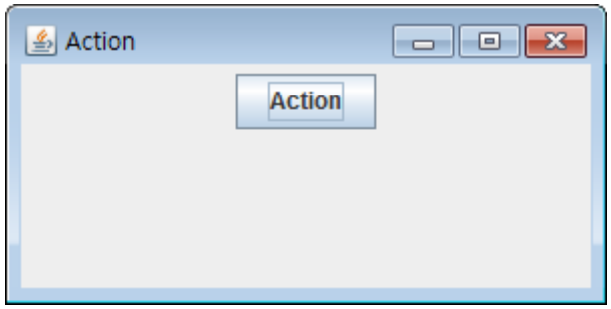


- 독립된 클래스로 Action 이벤트 리스너 작성
- 이 클래스를 별도의 MyActionListener.java파일로 작성하여도 됨

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

public class IndepClassListener extends JFrame {
   IndepClassListener() {
      setTitle("Action 이벤트 리스너 작성");
      setLayout(new FlowLayout());
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setSize(300,150);
      setVisible(true);
      JButton btn = new JButton("Action");
      MyActionListener listener = new MyActionListener();
      btn.addActionListener(listener);
      add(btn);
   }
   public static void main(String [] args) {
      new IndepClassListener();
   }
}
class MyActionListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      JButton b = (JButton)e.getSource();
      if(b.getText().equals("Action"))
         b.setText("액션");
      else
         b.setText("Action");
   }
}
```

내부 클래스로 리스너 작성





- Action 이벤트 리스너를 내부 클래스로 작성
- private으로 선언하여 InnerClassListener의 외부에서 사용할 수 없게 함

- 리스너에서 InnerClassListener의 멤버에 대한 접근 용이

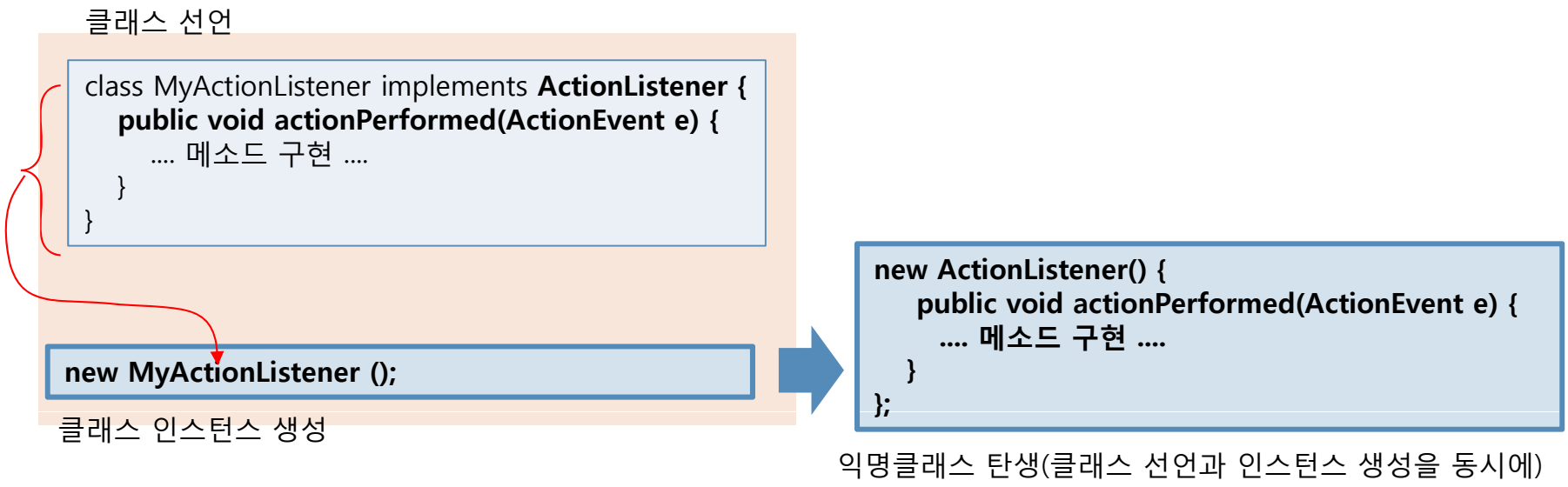
```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class InnerClassListener extends JFrame {
    InnerClassListener() {
        setTitle("Action 이벤트 리스너 작성");
        setLayout(new FlowLayout());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300,300);
        setVisible(true);
        JButton btn = new JButton("Action");
        btn.addActionListener(new MyActionListener());
        add(btn);
    }
    private class MyActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            JButton b = (JButton)e.getSource();
            if(b.getText().equals("Action"))
                b.setText("액션");
            else
                b.setText("Action");

            // InnerClassListener의 멤버나 JFrame의 멤버를 호출할 수 있음
            setTitle(b.getText()); // JFrame.setTitle() 호출
        }
    }
    public static void main(String [] args) {
        new InnerClassListener();
    }
}
```

익명 클래스로 이벤트 리스너 작성

- 익명 클래스란?
  - (클래스 정의 + 인스턴스 생성)을 한번에 작성

```
new 익명클래스의수퍼클래스이름(생성자의 인자들) {
        ..................
        클래스 정의
        ..................
};
```

  - ActionListener를 구현하는 익명의 이벤트 리스너 작성 예



익명 클래스로 이벤트 리스너 작성

익명 클래스로 다시 작성된 결과

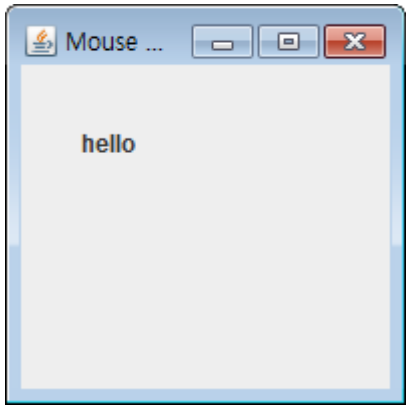
```
btn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JButton b = (JButton)e.getSource();
        if(b.getText().equals("Action"))
            b.setText("액션");
        else
            b.setText("Action");
            // AnonymousClassListener의 멤버나
            // JFrame의 멤버를 호출할 수 있음
            setTitle(b.getText());
    }
}
);
```

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

public class AnonymousClassListener extends JFrame {
    AnonymousClassListener() {
        setTitle("Action 이벤트 리스너 작성");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());
        setSize(300,300);
        setVisible(true);
        JButton btn = new JButton("Action");
        add(btn);
        btn.addActionListener(new MyActionListener()  );
    }
    private class MyActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            JButton b = (JButton)e.getSource();
            if(b.getText().equals("Action"))
                b.setText("액션");
            else
                b.setText("Action");

            // InnerClassListener의 멤버나 JFrame 멤버 호출 가능
            setTitle(b.getText());
        }
    }
    public static void main(String [] args) {
        new AnonymousClassListener ();
    }
}
```

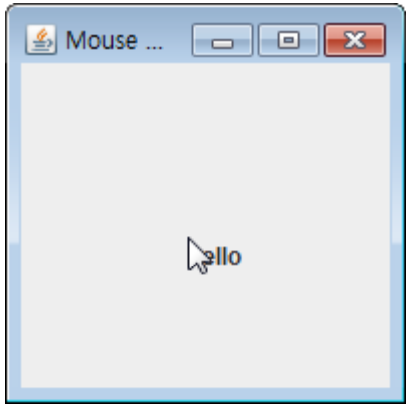
- 간단한 리스너의 경우 익명 클래스 사용 추천.
- 메소드의 개수가 1, 2개인 리스너(ActionListener, ItemListener)에 대해 주로 사용

예제 10-2 : 마우스로 문자열 이동시키기

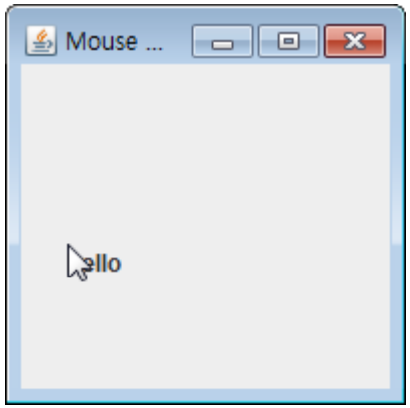
프레임상의 임의의 위치에 마우스 버튼을 누르면 마우스 포인트가 있는 위치로 "hello" 문자열을 이동시키는 스윙 응용프로그램을 작성하라



초기화면



마우스 다른 곳에 클릭한 경우



마우스 다른 곳에 클릭한 경우

- 마우스 버튼을 누르면 마우스가 있는 위치로 "hello" 문자열을 이동시킨다.
- 이벤트와 리스너 : MouseEvent와 MouseListener
- 이벤트 소스 : JPanel
- 구현할 리스너의 메소드 : mousePressed()
- "hello" 문자열 : JLabel 컴포넌트 이용
- 컨텐트 팬 : JPanel로 교체, 배치관리자를 null로 설정

예제 10-2의 소스

마우스 버튼이 눌러진 위치를 알아내어 la("hello" 문자열)를 그 위치로 옮긴다.

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

public class MouseListenerEx extends JFrame {
    JLabel la;

    MouseListenerEx() {
        setTitle("Mouse 이벤트 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel contentPane = new JPanel();
        setContentPane(contentPane);
        setLayout(null);
        contentPane.addMouseListener(new MyMouseListener());

        la = new JLabel("hello");
        la.setSize(50, 20);
        la.setLocation(30, 30);
        contentPane.add(la);

        setSize(200,200);
        setVisible(true);
    }
```

```
    class MyMouseListener implements MouseListener {
        public void mousePressed(MouseEvent e) {
            int x = e.getX();
            int y = e.getY();
            la.setLocation(x, y);
        }

        public void mouseReleased(MouseEvent e) {}
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
    }

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

어댑터(Adapter) 클래스

- 이벤트 리스너 구현에 따른 부담
  - 리스너의 추상 메소드들을 모두 구현해야하는 부담
    - 마우스 리스너에서 마우스가 눌러지는 경우(mousePressed())만 처리하고자 하는 경우에도 나머지 4 개의 메소드를 모두 구현해야 하는 부담
- 어댑터 클래스
  - JDK에서 제공
  - 리스너의 모든 메소드가 단순 리턴하도록 구현해 놓은 클래스
    - MouseAdapter 예

```
class MouseAdapter implements MouseListener {
   public void mousePressed(MouseEvent e) {}
   public void mouseReleased(MouseEvent e) {}
   public void mouseClicked(MouseEvent e) {}
   public void mouseEntered(MouseEvent e) {}
   public void mouseExited(MouseEvent e) {}
}
```

  - 추상 메소드가 하나뿐인 리스너는 어댑터 없음
    - ActionAdapter, ItemAdapter 클래스는 존재하지 않음

JDK에서 제공하는 어댑터 클래스

리스너 인터페이스	대응하는 어댑터 클래스
ActionListener	없음
ItemListener	없음
KeyListener	KeyAdapter
MouseListener	MouseAdapter
MouseMotionListener	MouseMotionAdapter 혹은 MouseAdapter
FocusListener	FocusAdapter
TextListener	없음
WindowListener	WindowAdapter
AdjustmentListener	없음
ComponentListener	ComponentAdapter
ContainerListener	ContainerAdapter

어댑터 사용 예


```
JLabel la;
JPanel contentPane  = new JPanel();
contentPane.addMouseListener(new MyMouseListener());
.....

class MyMouseListener implements MouseListener {
    public void mousePressed(MouseEvent e) {
        int x = e.getX();
        int y = e.getY();
        la.setLocation(x, y);
    }
    public void mouseReleased(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
}
```

MouseListener를 이용한 경우

```
JLabel la;
JPanel contentPane = new JPanel();
contentPane.addMouseListener(new MyMouseAdapter());
.....

class MyMouseAdapter extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        int x = e.getX();
        int y = e.getY();
        la.setLocation(x, y);
    }
}
```

MouseAdapter를 이용한 경우

예제 10-3: MouseAdapter 사용하기

MouseAdapter를 이용하여 예제 10-2를 수정하라.

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

public class MouseAdapterEx extends JFrame {
    JPanel contentPane = new JPanel();
    JLabel la;

    MouseAdapterEx() {
        setTitle("Mouse 이벤트 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(contentPane);
        setLayout(null);
        contentPane.addMouseListener(new MyMouseAdapter());
        la = new JLabel("hello");
        la.setSize(50, 20);
        la.setLocation(30, 30);
        contentPane.add(la);
        setSize(200,200);
        setVisible(true);
    }
```

```
class MyMouseAdapter extends MouseApater {
        public void mousePressed(MouseEvent e) {
            int x = e.getX();
            int y = e.getY();
            la.setLocation(x, y);
        }
    }
    public static void main(String [] args) {
        new MouseAdapterEx();
    }
}
```

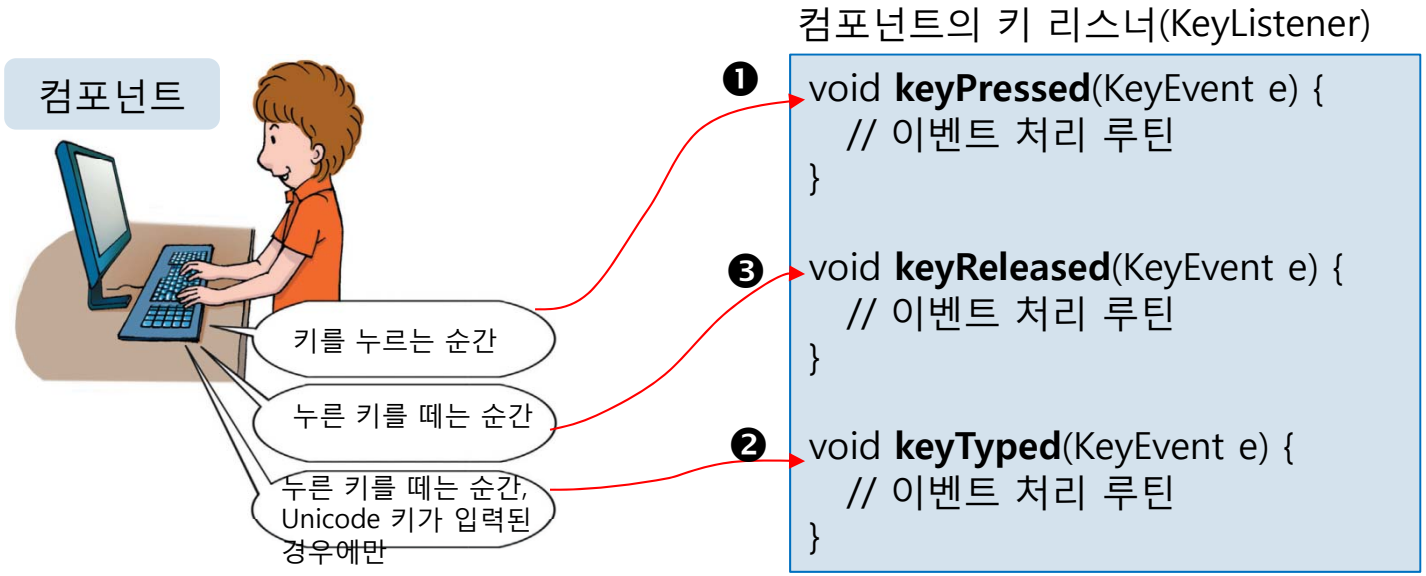
```
    class MyMouseListener implements MouseListener {
        public void mousePressed(MouseEvent e) {
            int x = e.getX();
            int y = e.getY();
            la.setLocation(x, y);
        }

        public void mouseReleased(MouseEvent e) {}
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
    }
    public static void main(String [] args) {
        new MouseListenerEx();
    }
}
```

Key 이벤트와 포커스

- 키 입력 시, 다음 세 경우에 Key 이벤트 발생
  - 키를 누르는 순간
  - 누른 키를 떼는 순간
  - 누른 키를 떼는 순간(Unicode 키의 경우에만)
- 키 이벤트를 받을 수 있는 조건
  - 모든 컴포넌트 가능하지만, 현재 포커스(focus)를 가진 컴포넌트
- 포커스(focus)
  - 컴포넌트나 응용프로그램이 키 이벤트를 독점하는 권한
  - 컴포넌트에 포커스 설정 방법
    - component.requestFocus(); // component가 키 이벤트를 받을 수 있게 함

KeyListener의 메소드와 키

- KeyListener의 3 개 메소드



KeyListener의 메소드가 실행되는 순서 ❶ ❷ ❸

- 컴포넌트에 키 이벤트 리스너 등록

```
component.addKeyListener(myKeyListener);
```

유니코드(Unicode)

- 유니코드 키의 특징
  - 국제 산업 표준
  - 전 세계의 문자를 컴퓨터에서 일관되게 표현하기 위한 코드 체계
  - 문자들에 대해서만 코드 값 정의
    - A~Z, a~z, 0~9, !, @, & 등
  - 문자 키가 아닌 경우에는 통일된 키 코드 값 없음
    - <Function> 키, <Home> 키, <Up> 키,<Delete> 키, <Control> 키, <Shift> 키, <Alt> 등
    - 플랫폼에 따라 키 코드 값이 다를 수 있음
- 유니코드 키가 아닌 경우
  - keyPressed(), keyReleased() 만 호출됨
- 유니코드 키가 입력되는 경우
  - keyPressed(), keyTyped(), keyReleased() 가 순서대로 호출

입력된 키 판별

- KeyEvent 객체
  - 키가 입력되면 입력된 키 정보를 가진 이벤트 객체 생성 : KeyEvent 객체
    - KeyEvent 객체가 리스너에 전달됨
- KeyEvent 객체의 메소드로 입력된 키 판별
  - Unicode 키의 문자 값 판별, char KeyEvent.getKeyChar()
    - 눌러진 키에 해당하는 문자 값 리턴
    - 눌러진 키가 Unicode 문자 키인 경우에만 의미 있음
  - Unicode 문자뿐 아니라 모든 키 판별, int KeyEvent.getKeyCode()
    - 눌러진 키에 대한 정수형 키 코드 값 리턴
    - Unicode 문자에 관계 없이, Function 키, Modifier(shift) 키, Control 키, Action 키 등 모든 키에 대해 키 코드 값 리턴
    - 운영체제나 하드웨어에 따라 키 셋은 서로 다름
    - 입력된 키를 판별하기 위해 가상키(Virtual Key) 값과 비교하여야 함
      - 가상 키 값은 KeyEvent 클래스의 상수로 정의됨
- 키 이름 문자열 리턴 String KeyEvent.getKeyText(int keyCode)
  - static 메소드
  - keyCode의 코드 값에 해당하는 키 이름 문자열 리턴
    - F1 키의 경우 "F1", Shift 키의 경우 "SHIFT" 등의 문자열 리턴

가상 키(Virtual Key)

- 가상 키는 KeyEvent 클래스에 상수로 선언
- 가상 키의 일부분 사례

가상 키	설명	가상 키	설명
VK_0 ~ VK_9	0에서 9까지의 키, '0' ~ '9' 까지의 ASCII 값과 동일	VK_LEFT	왼쪽 방향 키
VK_A ~ VK_Z	A에서 Z까지의 키, 'A' ~ 'Z' 까지의 ASCII 값과 동일	VK_RIGHT	오른쪽 방향 키
VK_F1 ~ VK_F24	〈Function〉 키. F1~F24까지의 키 코드	VK_UP	〈Up〉 키
VK_HOME	〈Home〉 키	VK_DOWN	〈Down〉 키
VK_END	〈End〉 키	VK_CONTROL	〈Control〉 키
VK_PGUP	〈Page Up〉 키	VK_SHIFT	〈Shift〉 키
VK_PGDN	〈Page Down〉 키	VK_ALT	〈Alt〉 키
VK_UNDEFINED	입력된 키의 코드 값을 알 수 없음	VK_TAB	〈Tab〉 키

KeyListener의 메소드와 키

a 키를 누르는 순간

```
void keyPressed(KeyEvent e) {
    char keyChar = e.getKeyChar();
    int keyCode = e.getKeyCode();
}
```

keyChar	키 a의 유니코드 값('a')
keyCode	VK_A

〈F5〉 키를 누르는 순간

```
void keyPressed(KeyEvent e) {
    char keyChar = e.getKeyChar();
    int keyCode = e.getKeyCode();
}
```

keyChar	CHAR_UNDEFINED
keyCode	VK_F5

w 키를 떼는 순간

```
void keyTyped(KeyEvent e) {
    char keyChar = e.getKeyChar();
    int keyCode = e.getKeyCode();
}
```

keyChar	키 w의 유니코드 값('w')
keyCode	VK_UNDEFINED

〈F5〉 키를 떼는 순간

```
void keyTyped(KeyEvent e) {
    char keyChar = e.getKeyChar();
    int keyCode = e.getKeyCode();
}
```

keyChar	
keyCode	

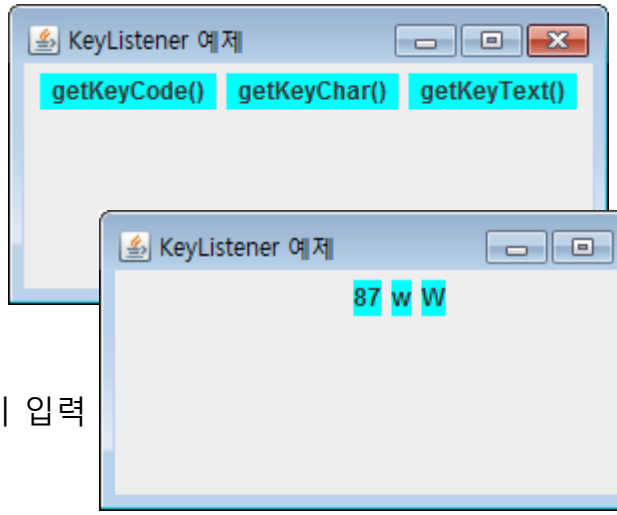
KeyEvent와 KeyListener의 활용 : getKeyCode(), getKeyChar(), getKeyText() 사용

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

public class KeyListenerEx extends JFrame {
    JPanel contentPane = new JPanel();
    JLabel [] keyMessage;

    KeyListenerEx() {
        setTitle("KeyListener 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setContentPane(contentPane);
        contentPane.addKeyListener(new MyKeyListener());

        keyMessage = new JLabel [3];
        keyMessage[0] = new JLabel("   getKeyCode()   ");
        keyMessage[1] = new JLabel("   getKeyChar()   ");
        keyMessage[2] = new JLabel("   getKeyText()   ");

        for(int i=0; i<keyMessage.length; i++) {
            contentPane.add(keyMessage[i]);
            keyMessage[i].setOpaque(true);
            keyMessage[i].setBackground(Color.CYAN);
        }
        setSize(300,150);
        setVisible(true);
        contentPane.requestFocus();
    }

class MyKeyListener extends KeyAdapter {
        public void keyPressed(KeyEvent e) {
            int keyCode = e.getKeyCode();
            char keyChar = e.getKeyChar();
            keyMessage[0].setText(Integer.toString(keyCode));

    keyMessage[1].setText(Character.toString(keyChar));
            keyMessage[2].setText(e.getKeyText(keyCode));
        }
    }

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

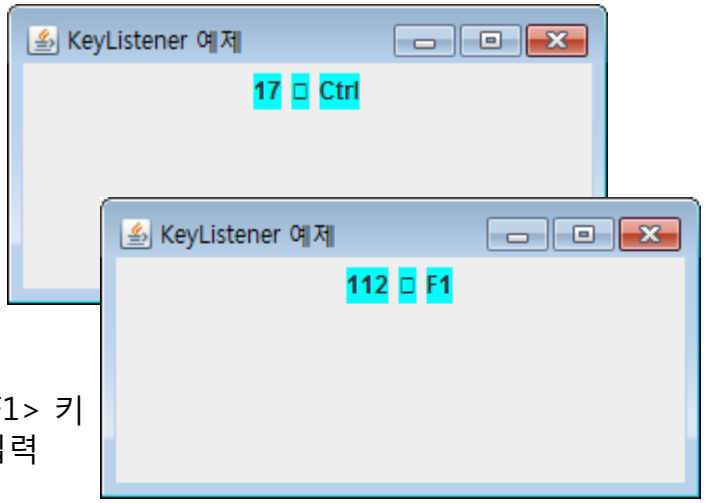
• JComponent 컴포넌트에 바탕색을 지정하기 위해서는 사전에 컴포넌트가 불투명함을 지정하여야 한다.

실행 결과

초기화면

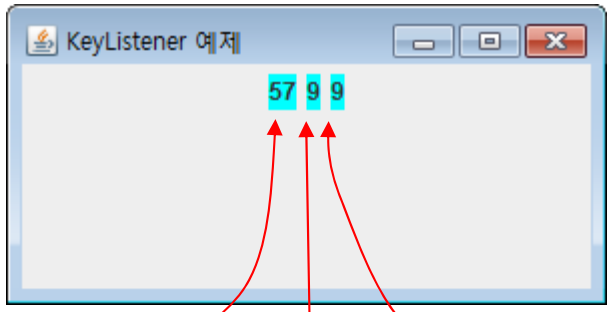


<Control> 키 입력



w 키 입력

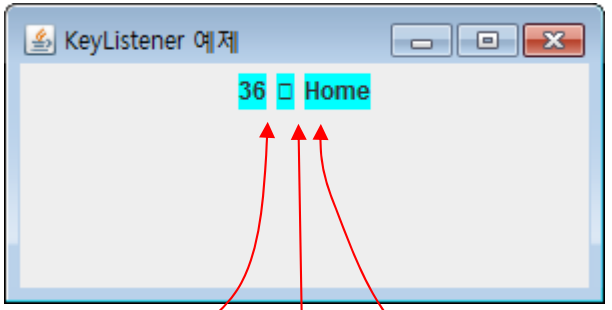
<F1> 키 입력



키 9의 키코드

키 9의 유니코드 문자

키 9의 이름 문자열



<Home> 키 코드

<Home>키에 대응하는 문자 없음

<Home> 키의 이름 문자열

예제 10-4 : F1 키를 입력받으면 바탕을 초록색으로, % 키를 입력받으면 바탕을 노란색으로 변경


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

public class KeyCodeEx extends JFrame {
    JPanel contentPane = new JPanel();
    JLabel la = new JLabel();

    KeyCodeEx() {
        setTitle("Key Code 예제 : F1키:초록색, % 키 노란색");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setContentPane(contentPane);
        contentPane.addKeyListener(new MyKeyListener());
        contentPane.add(la);
        setSize(300,150);
        setVisible(true);
        contentPane.requestFocus();
    }
```

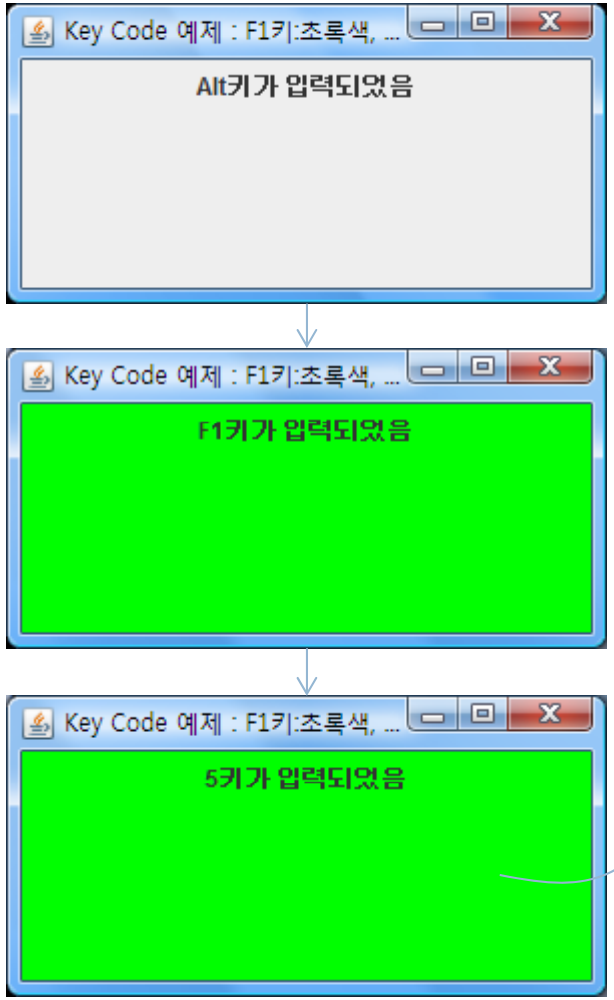
- JPanel이 키 입력을 받을 수 있도록 포커스를 준다.

```
    class MyKeyListener extends KeyAdapter {
        public void keyPressed(KeyEvent e) {
            la.setText(e.getKeyText(e.getKeyCode()));
            if(e.getKeyChar() == '%')
                contentPane.setBackground(Color.YELLOW);
            else if(e.getKeyCode() == KeyEvent.VK_F1)
                contentPane.setBackground(Color.GREEN);
        }
    }
    public static void main(String [] args) {
        new KeyCodeEx();
    }
}
```

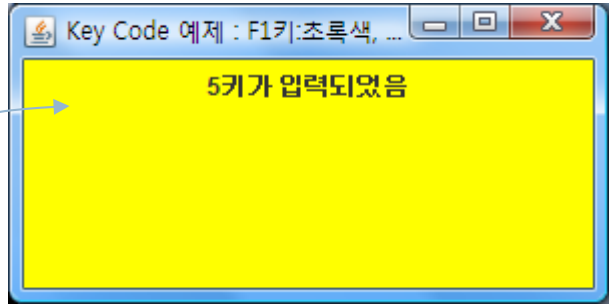
- % 키를 판별하기 위해 e.getKeyChar() 이용
- '%' 문자와 비교

- F1 키를 판별하기 위해 e.getKeyCode() 이용
- KeyEvent.VK_F1 값과 비교

예제 10-4 실행



키 5 를 누른 경우로서
노란색 배경으로 변하지 않는다.

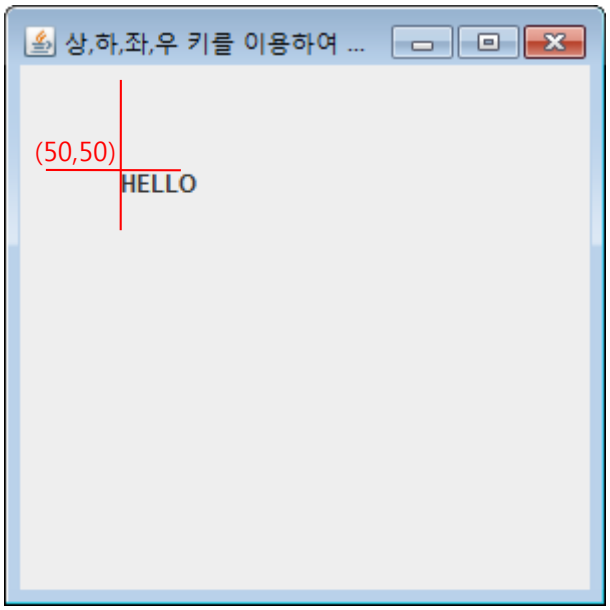


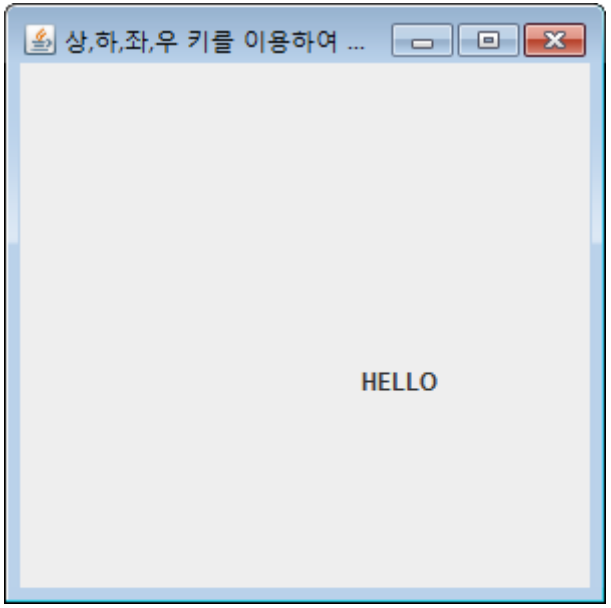
- %(Shift+5) 키가 입력된 경우로 배경이 노란색으로 변경되었다.
- % 는 Shift키+5키이므로 최종적으로는 5 키에 대한 문자열이 출력

예제 10-5 : 상,하,좌,우 키로 "HELLO"문자열 움직이기

상, 하, 좌, 우 키를 이용하여 아래 그림과 같이 "HELLO" 문자열을 화면에서 움직이는 응용프로그램을 작성하라.

"HELLO" 문자열은 JLabel 컴포넌트로 생성하여 컨텐트팬에 부착하고 상, 하, 좌, 우 키를 움직이면 키 방향으로 한 번에 10픽셀씩 움직인다. 이를 위해 컨텐트팬의 배치관리자를 삭제하여야 한다. 아래 그림은 초기 "HELLO" 문자열을 (50, 50) 위치에 출력하고 키를 입력함에 따라 키 방향으로 문자열이 움직이는 화면이다.





상,하,좌,우 키를 움직이면 한 번에 10픽셀씩 "HELLO" 텍스트는 상,하,좌,우로 이동한다.
이 텍스트는 프레임의 영역을 벗어나서 움직일 수 있다.

예제 소스: 상,하,좌,우 키로 텍스트 움직이기


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

public class FlyingTextEx extends JFrame {
    JPanel contentPane = new JPanel();
    JLabel la = new JLabel("HELLO");
    final int FLYING_UNIT = 10;

    FlyingTextEx() {
        setTitle("상,하,좌,우 키를 이용하여 텍스트 움직이기");
        setDefaultCloseOperation(
                JFrame.EXIT_ON_CLOSE);

        setContentPane(contentPane);
        contentPane.setLayout(null);
        contentPane.addKeyListener(new MyKeyListener());
        la.setLocation(50,50);
        la.setSize(100,20);
        contentPane.add(la);
        setSize(300,300);
        setVisible(true);
        contentPane .requestFocus();
    }
```

```
    class MyKeyListener extends KeyAdapter {
        public void keyPressed(KeyEvent e) {
            int keyCode = e.getKeyCode();
            switch(keyCode) {
                case KeyEvent.VK_UP:
                    la.setLocation(la.getX(), la.getY()-FLYING_UNIT);
                    break;
                case KeyEvent.VK_DOWN:
                    la.setLocation(la.getX(), la.getY()+FLYING_UNIT);
                    break;
                case KeyEvent.VK_LEFT:
                    la.setLocation(la.getX()-FLYING_UNIT, la.getY());
                    break;
                case KeyEvent.VK_RIGHT:
                    la.setLocation(la.getX()+FLYING_UNIT, la.getY());
                    break;
            }
        }
    }

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

MouseEvent와 MouseListener,MouseMotionListener

- Mouse 이벤트
  - 사용자의 마우스 조작에 따라 발생하는 이벤트

Mouse 이벤트가 발생하는 경우	리스너의 메소드	리스너
마우스가 컴포넌트 위에 올라갈 때	void mouseEntered(MouseEvent e)	MouseListener
마우스가 컴포넌트에서 내려올 때	void mouseExited(MouseEvent e)	MouseListener
마우스 버튼이 눌러졌을 때	void mousePressed(MouseEvent e)	MouseListener
눌러진 버튼이 떼어질 때	void mouseReleased(MouseEvent e)	MouseListener
마우스로 컴포넌트를 클릭하였을 때	void mouseClicked(MouseEvent e)	MouseListener
마우스가 드래그되는 동안	void mouseDragged(MouseEvent e)	MouseMotionListener
마우스가 움직이는 동안	void mouseMoved(MouseEvent e)	MouseMotionListener

  - 마우스가 클릭되어 드래그되는 동안 호출되는 메소드 순서

mousePressed(), mouseDragged(), mouseReleased(), mouseClicked()

MouseEvent로부터 얻을 수 있는 정보

- 마우스 포인터의 위치
  - int getX(), int getY(),
  - Point getPoint()

```
public void mousePressed(MouseEvent e) {
   int x = e.getX();
   int y = e.getY();
}
```

- 입력된 마우스 버튼
  - short getButton()

```
public void mousePressed(MouseEvent e) {
   if(e.getButton() == MouseEvent.BUTTON1)
      System.out.println("Left Button Pressed");
}
```

- 마우스 클릭 횟수
  - int getClickCount()

```
public void mouseClicked(MouseEvent e) {
   if(e.getClickCount() == 2) {
      // 더블클릭을 처리하는 루틴
   }
}
```

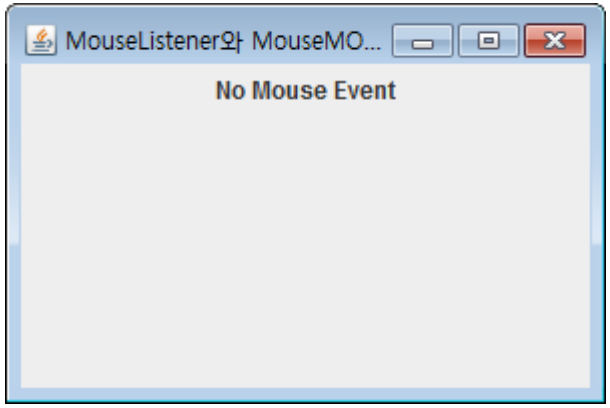
- 팝업 메뉴 클릭
  - boolean isPopupTrigger()

MouseListener와 MouseMotionListener 사용

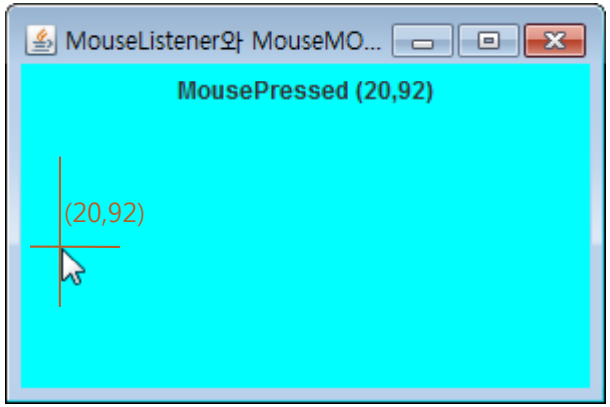
```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class MouseListenerAllEx extends JFrame {
    JPanel contentPane = new JPanel();
    JLabel la;

    MouseListenerAllEx() {
        setTitle("MouseListener와 MouseMOtionListener 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(contentPane);
        contentPane.addMouseListener(
                    new MyMouseListener());
        contentPane.addMouseMotionListener(
                    new MyMouseListener());
        la = new JLabel("No Mouse Event");
        contentPane.add(la);
        setSize(300,200);
        setVisible(true);
    }
```

```
    class MyMouseListener implements MouseListener,
                                     MouseMotionListener {
        public void mousePressed(MouseEvent e) {
            la.setText("MousePressed ("+e.getX()+","+e.getY()+")");
        }
        public void mouseReleased(MouseEvent e) {
            la.setText("MouseReleased("+e.getX()+","+e.getY()+")");
        }
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {
            JPanel p = (JPanel)e.getSource();
            p.setBackground(Color.CYAN);
        }
        public void mouseExited(MouseEvent e) {
            JPanel p = (JPanel)e.getSource();
            p.setBackground(Color.YELLOW);
        }
        public void mouseDragged(MouseEvent e) {
            la.setText("MouseDragged ("+e.getX()+","+e.getY()+")");
        }
        public void mouseMoved(MouseEvent e) {
            la.setText("MouseMoved ("+e.getX()+","+e.getY()+")");
        }
    }

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

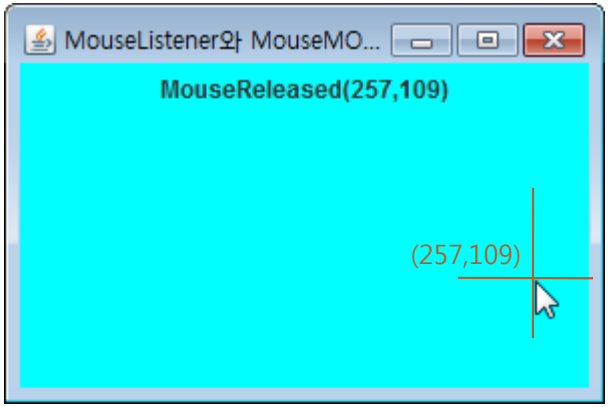
실행: MouseListener와 MouseMotionListener 사용



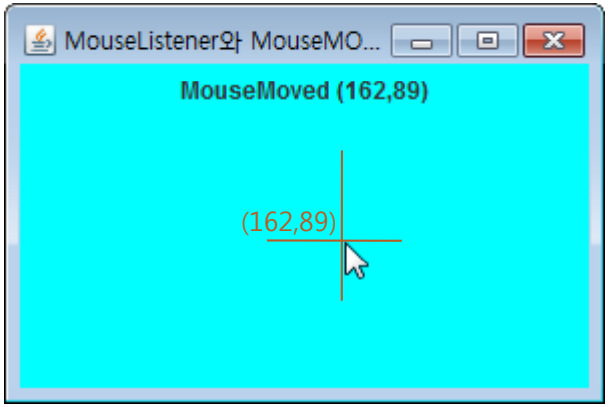
초기화면



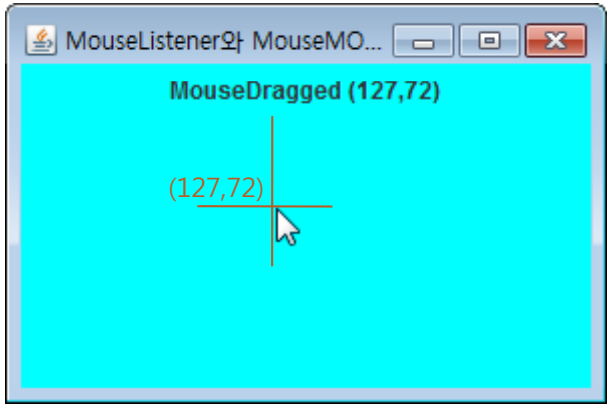
mouseEntered()에 의해 배경색 변경.
마우스 버튼이 눌러진 순간



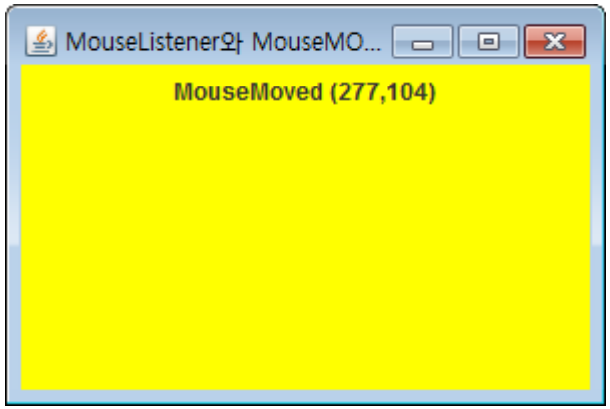
눌러진 마우스 버튼이 떼어진 순간



마우스가 패널 위에 이동하는 동안



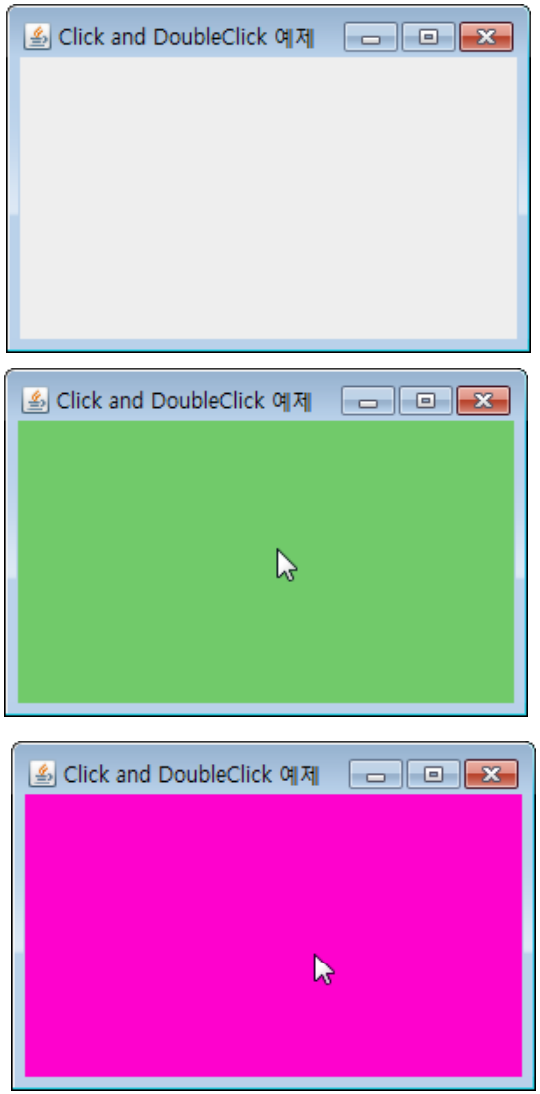
마우스가 패널 위에 드래깅하는 동안



마우스가 패널 바깥으로 나가면
mouseExited()에 의해 배경색 변경

예제 10-6 : 더블클릭 시 컨텐트팬의 배경색 변경

더블클릭할 때마다 컨텐트팬의 배경색을 랜덤하게 변경한다.




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

public class ClickAndDoubleClickEx extends JFrame {
   JPanel contentPane = new JPanel();
   ClickAndDoubleClickEx() {
      setTitle("Click and DoubleClick 예제");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setContentPane(contentPane);
      contentPane.addMouseListener(new MyMouseListener());
      setSize(300,200);
      setVisible(true);
   }

   class MyMouseListener extends MouseAdapter {
      public void mouseClicked(MouseEvent e) {
         if(e.getClickCount() == 2) {
            int r = (int)(Math.random()*256);
            int g = (int)(Math.random()*256);
            int b = (int)(Math.random()*256);

            JPanel p = (JPanel)e.getSource();
            p.setBackground(new Color(r,b,g));
         }
      }
   }

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