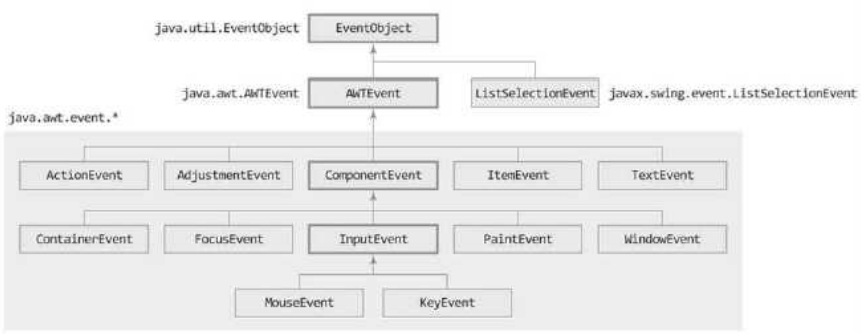
10차시 정리

```
// 이벤트 기반 프로그래밍

// 이벤트 종류
// 사용자의 입력 : 마우스 드래그, 마우스 클릭, 키보드 누름
// 센서로부터의 입력, 네트워크로부터 데어티 송수신
// 다른 응용프로그램이나 다른 스레드로부터의 메시지

// 이벤트의 발생에 의해 프로그램 흐름이 결정되는 방식
// 이벤트가 발생하면 이벤트를 처리하는 이벤트 리스너 실행
// 프로그램 내의 어떤 코드가 언제 실행될 지 이벤트 발생에 의해 전적으로 결정

// 반대되는 개념 : 배치 실행 - 프로그램의 개발자가 프로그램의 흐름을 결정하는 방식
// 현재의 개념 : 이벤트 - 대부분 약 90% 이벤트 기반으로 프로그램이 돌아간다.

// 이반트 기반 프로그램의 구조 - 이벤트 리스너 들의 집합

// 이벤트 처리 순서
// 이벤트 발생 : 마우스나 키보드의 움직임 혹은 입력
// 이벤트 객체 생성 : 현재 발생한 이벤트에 대한 정보를 가진 객체
// 이벤트 리스너 찾기
// 이벤트 리스너 호출 : 이벤트 객체가 리스너에 전달죔
// 이벤트 리스너 실행

// 자바의 이벤트 기반 GUI 응용프로그램 구성
// PC등 하드웨어 : Action(만져지는 것들 : 마우스, 모니터 등등)
// 운영체제
// 자바 가상 기계(JVM)
// 자바 응용프로그램(
// 이벤트 분배 스레드(ActionEvent 생성) -> 이벤트 리스너 -> JButton, JMenuItem, JTextField 등
// )
// 발생한 이벤트는 Action 이벤트
// 이벤트 소스는 JButton
// 이벤트 객체는 ActionEvent
// 이벤트 리스너는 이벤트 리스너4
```

// 이벤트 관련 용어
// 이벤트 소스 : 이벤트를 발생시킨 GUI 컴포넌트
// 이벤트 객체 : 발생한 이벤트에 대한 정보(이벤트 종류, 이벤트 소스, 화변 좌표, 마우스 버튼 종류)
// 이벤트 리스너 : 마우스, 키보드 등 이벤트를 처리하는 코드, 컴포넌트에 등록되어야 작동 가능
// 이벤트 분배 스레드
// -- 동작(자바 가상 기계로부터 이벤트의 발생을 통지 받음, 이벤트 소스와 이벤트 종류 결정,
// 적절한 이벤트 객체 생성, 이벤트 리스너를 찾아 호출)
// -- 무한 루프를 실행하는 스레드

// 이벤트 객체
// 이벤트가 발샐할 때, 발생한 이벤트에 관한 정보를 가진 객체
// 이벤트 리스너에 전달됨 : 이벤트 리스너 코드에서 이벤트가 발생한 상황을 파악할 수 있게 함

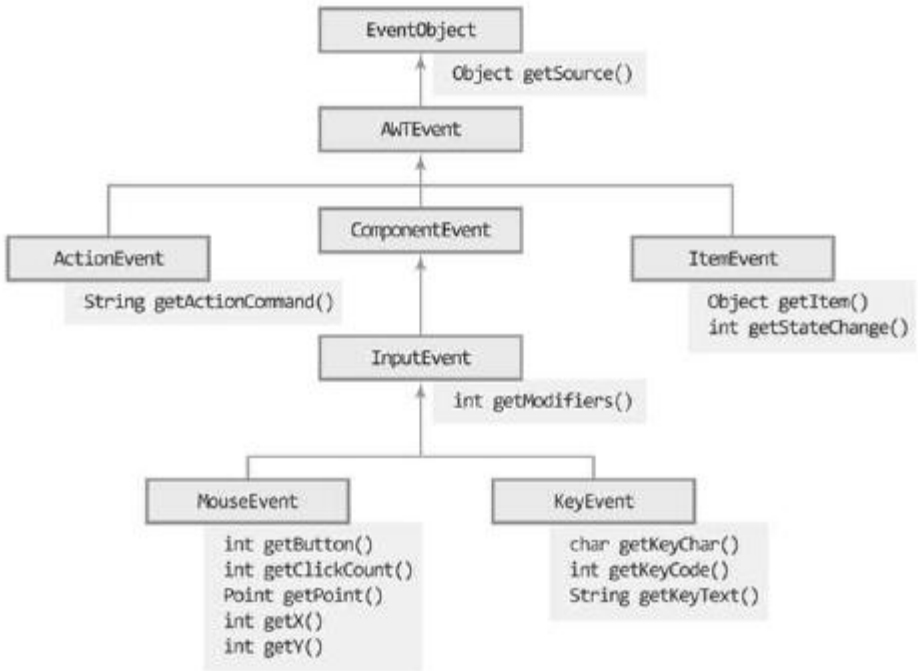
// 이벤트 객체의 종류



// 이벤트 객체에 포함된 정보
// 이벤트 종류, 이벤트 소스, 이벤트가 발생한 화면좌표 나 컴포넌트 내 좌표
// 클릭된 마우스 버튼 번호, 마우스 클릭 횟수, 눌린 키에 대한 코드값과 문자 값
// 체크박스, 라디오 버튼 등과 같은 컴포넌트에 이벤트가 발생하였다면 체크 상태

// 이벤트에 따라 조금씩 다른 정보 포함
// ActionEvent 객체 : 액션 문자열
// MouseEvent 객체 : 마우스의 우치 정보, 마우스 버튼, 함께 눌러진 키 정보 등
// ItemEvent 객체 : 아이템의 체트 상태

// 이벤트 소스 알아내기 - Object EventObject.getSource()
// 발생한 이벤트의 소스 컴포넌트 리턴
// Object 타입으로 리턴하므로 캐스팅하여 사용
// 모든 이벤트 객체에 대해 적용



// 이벤트 객체와 이벤트 소스

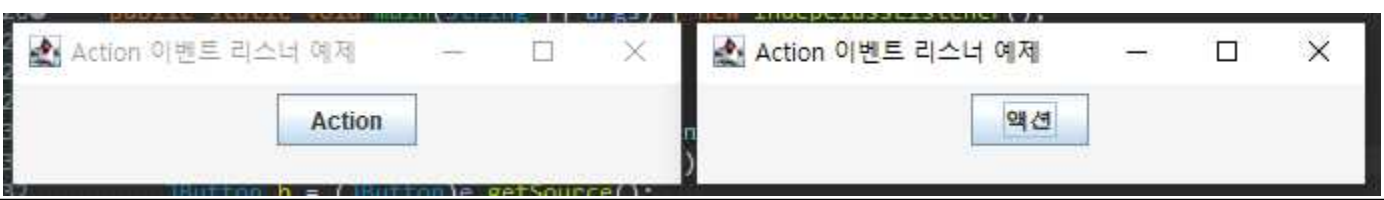
이벤트 객체	이벤트 소스	이벤트가 발생하는 경우
ActionEvent	JButton	마우스나 <Enter> 키로 버튼 선택
	JMenuItem	메뉴 아이템 선택
	JTextField	텍스트 입력 중 <Enter> 키 입력
ItemEvent	JCheckBox	체크박스의 선택 혹은 해제
	JRadioButton	라디오버튼의 선택 상태가 변할 때
	JCheckBoxMenuItem	체크박스 메뉴 아이템의 선택 혹은 해제
ListSelectionEvent	JList	리스트에서 선택된 아이템이 변경될 때
KeyEvent	Component	키가 눌려지거나 눌려진 키가 떼어질 때
MouseEvent	Component	마우스 버튼이 눌려지거나 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
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, 이동, 크기 변경 시
ContainerEvent	Container	Container에 컴포넌트의 추가 혹은 삭제

// 이벤트 리스너
// 이벤트를 처리하는 코드, 클래스로 작성, 실제 처리기
// JDK에서 이벤트 리스너 작성을 위한 인터페이스 제공 : 개발자가 리스너 인터페이스의 추상 메소드를 구현
// 예) ActionListener 인터페이스
// 예) interface ActionListener{ public void actionPerformed(ActionEvent e); }
// 예) actionPerformed를 구현 해야됨
// 예) MouseListener 인터페이스
// 예) interface MouseListener{ public void mousePressed(MouseEvent e); ...;}
// 예) mousePressed, ... 등의 메소드를 구현해야됨

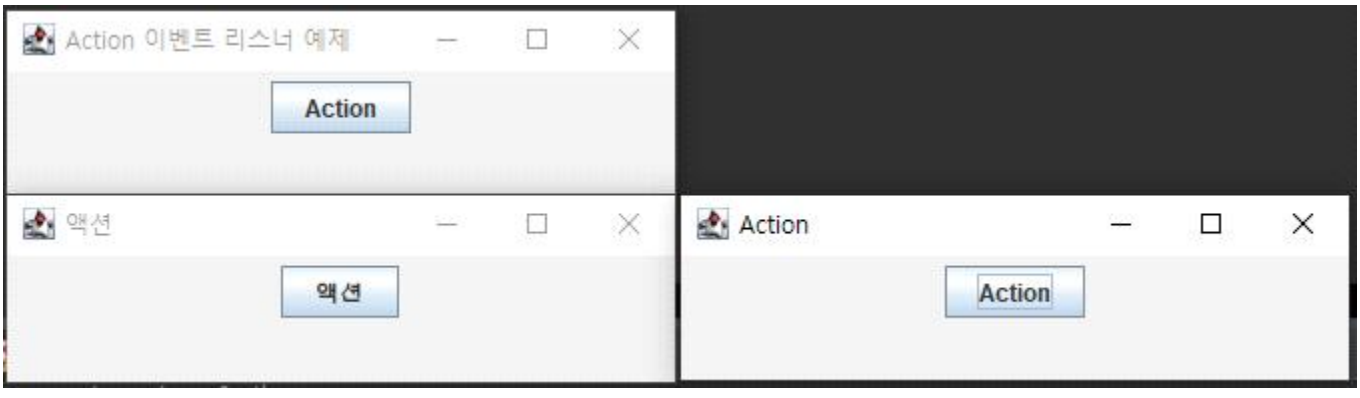
// 리스너 인터페이스와 메소드

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 눌려질 때
		void keyReleased(KeyEvent)	모든 키에 대해 눌려진 키가 떼어질 때
		void keyTyped(KeyEvent)	유니코드 키가 입력될 때
Mouse	MouseListener	void mousePressed(MouseEvent)	마우스 버튼이 눌려질 때
		void mouseReleased(MouseEvent)	눌려진 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
ListSelection	ListSelectionListener	void valueChanged(ListSelectionEvent)	JList에 선택된 아이템이 변경될 때
Window	WindowListener	void windowOpened(WindowEvent)	윈도우가 생성되어 처음으로 보이게 될 때
		void windowClosing(WindowEvent)	윈도우의 시스템 메뉴에서 윈도우 닫기를 시도할 때
		void windowIconified(WindowEvent)	윈도우가 아이콘화 될 때
		void windowDeiconified(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

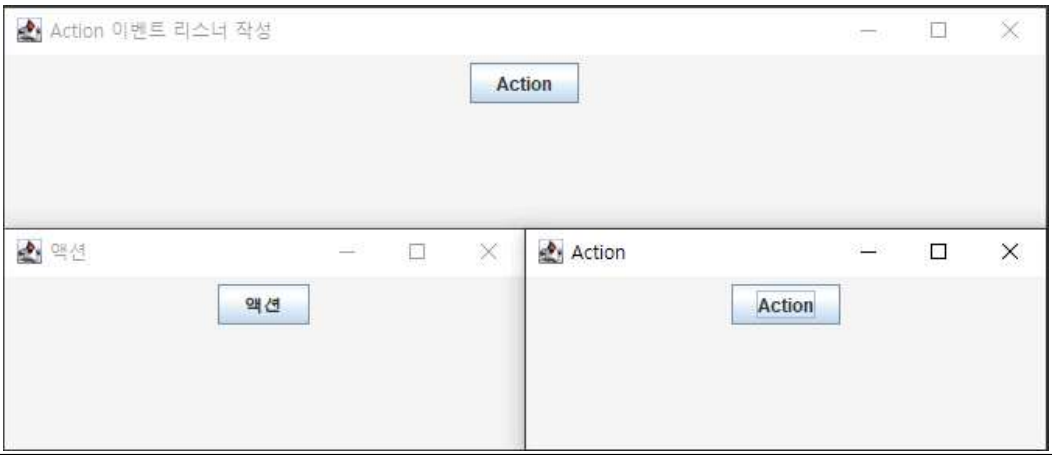
```
// 리스너 등록 메소드가 addXXXListener인 이유
// 컴포넌트는 다른 이벤트에 대한 리스너를 동시에 가질수 있다.
// JButton.addActionListener(); // Action 리스너
// JButton.addKeyListener(); // Key 리스너
// JButton.addFocusLitener(); // Focus 리스너
// 컴포넌트는 한 이벤트에 대해 여러 개의 리스너를 동시에 가질 수 있다.
// JButton.addActionListener(new MyButtonListener1());
// JButton.addActionListener(new MyButtonListener2());
// JButton.addActionListener(new MyButtonListener3());
// 이때, 리스너는 등록된 반대 순으로 모두 실행된다.

// 이벤트 리스너 작성 방법 - 3가지 방법
// 독립 클래스로 작성
// -- 이벤트 리스너를 완전한 클래스로 작성
// -- 이벤트 리스너를 여러 곳에서 사용할 때 적합
// 내부 클래스로 작성
// -- 클래스 안에 멤버처럼 클래스 작성
// -- 이벤트 리스너를 특정 클래스에서만 사용할 때 적합
// 익명 클래스로 작성
// -- 클래스의 이름 없이 간단히 리스너 작성
// -- 클래스 조차 만들 필요 없이 리스너 코드가 간단한 경우에 적합
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class IndepClassListener extends JFrame {
	public IndepClassListener() {
		setTitle("Action 이벤트 리스너 예제");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		Container c = getContentPane(); // c라는 객체를 찾고
		c.setLayout(new FlowLayout());

		JButton btn = new JButton("Action"); // 버튼 생성
		btn.addActionListener(new MyActionListener()); // Action 리스너 달기
		c.add(btn);

		setSize(350, 150);
		setVisible(true);
	}
	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");
	}
}
// 결과 : 버튼을 누르면 이름이 Action 과 액션 으로 바뀐다.
```



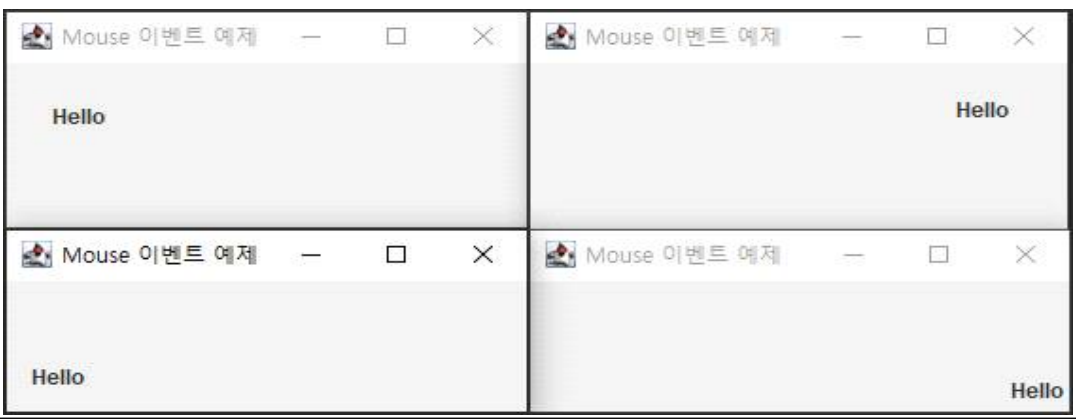
```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class InnerClassListener extends JFrame {
	public InnerClassListener() {
		setTitle("Action 이벤트 리스너 예제"); // 창 제목
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 창의 오른쪽 상단 - ㅁ x 표시

		Container c = getContentPane(); // 참조 객체 가져오기
		c.setLayout(new FlowLayout()); // 레이아웃 설정

		JButton btn = new JButton("Action"); // 버튼 추가
		btn.addActionListener(new MyActionListener()); // Action 리스너 설정
		c.add(btn);

		setSize(350, 150);
		setVisible(true);
	}
	private class MyActionListener implements ActionListener { // 내부 클래스로 Action 리스너 작성
		public void actionPerformed(ActionEvent e) {
			JButton b = (JButton)e.getSource(); // 버튼의 이름을 가져온다.
			if(b.getText().equals("Action")) // 버튼의 문자를 바꾼다.
				b.setText("액션");
			else
				b.setText("Action");
			// InnerClassListener의 멤버나 JFrame의 멤버를 호출할 수 있음
			InnerClassListener.this.setTitle(b.getText()); // 프레임 타이틀에 문자를 바꾼다.
		}
	}
	public static void main(String [] args) {
		new InnerClassListener();
	}
}


// 출력
```



```
// 익명 클래스로 이벤트 리스너 작성
// 익명 클래스란 : (클래스 정의 + 인스턴스 생성) 을 한번에 작성
// ActionListener를 구현하는 익명의 이벤트 리스너 작성
// 예) 정상
// 예) class MyActionListener implements ActionListener{
// 예)     public void actionPerformed(ActionEvent e){ ... }
// 예) }
// 예) b.addActionListener(new MyActionListener());
// 예) 익명의 이벤트 리스너 작성
// 예) b.addActionListener(new MyActionListener(){
// 예)     public void actionPerformed(ActionEvent e){ ... }
// 예) });

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

public class AnonymousClassListener extends JFrame {
	public AnonymousClassListener() {
		setTitle("Action 이벤트 리스너 작성");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		Container c = getContentPane();
		c.setLayout(new FlowLayout());

		JButton btn = new JButton("Action");
		c.add(btn);

		// 익명 클래스로 Action 리스너 작성
		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());
			}
		});
		setSize(350, 150);
		setVisible(true);
	}
	public static void main(String [] args) {
		new AnonymousClassListener();
	}
}
```



```
// 이벤트 리스너
// 이벤트를 처리하는 코드, 클래스로 작성, 실제 처리기
// JDK에서 이벤트 리스너 작성을 위한 인터페이스 제공 : 개발자가 리스너 인터페이스의 추상 메소드를 구현
// 예) ActionListener 인터페이스
// 예) interface ActionListener{ public void actionPerformed(ActionEvent e); }
// 예) actionPerformed를 구현 해야됨
// 예) MouseListener 인터페이스
// 예) interface MouseListener{ public void mousePressed(MouseEvent e); ...;}
// 예) mousePressed, ... 등의 메소드를 구현해야됨

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class MouseListenerEx extends JFrame {
	private JLabel la = new JLabel("Hello");

	public MouseListenerEx() {
		setTitle("Mouse 이벤트 예제");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container c = getContentPane();
		c.addMouseListener(new MyMouseListener());
		c.setLayout(null);

		la.setSize(50, 20);
		la.setLocation(30, 30);
		c.add(la);

		setSize(250, 250);
		setVisible(true);
	}
	class MyMouseListener implements MouseListener {
		public void mousePressed(MouseEvent e) {
			// 마우스 버튼이 눌러진 위치를 알아내어
			// la("hello" 문자열)를 그 위치로 옮긴다.
			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();
	}
}
```



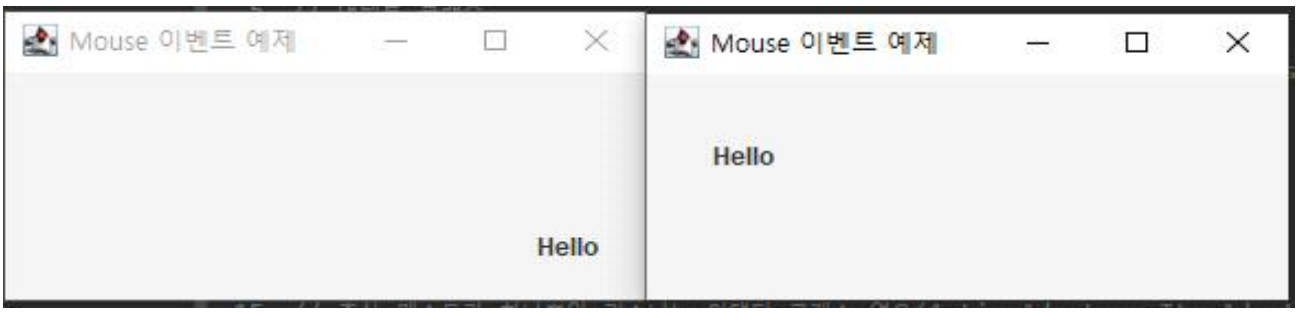
// 어댑터 클래스
// 이벤트 리스너 구현에 따른 부담 해소를 위해
// 리스너 작성시 추상 메소드들을 모두 구현해야 하는 부담 해소 위해

// 어댑트 클래스
// 리스너의 모든 메소드가 단순 리턴하도록 구현해 놓은 클래스
// 예) class MouseAdapter implements MouseListener, MouseMotionListener{
// 예) public void mousePressed(MouseEvent e){}
// 예) public void mouseReleased(MouseEvent e){}
// 예) public void mouseClicked(MouseEvent e){}
// 예) public void mouseEntered(MouseEvent e){}
// 예) public void mouseExited(MouseEvent e){}
// 예) public void mouseDragged(MouseEvent e){}
// 예) }
// 추상 메소드가 하나뿐인 리스너는 어댑터 클래스 없음(ActionAdapter, ItemAdapter 클래스)

// JDK에서 제공하는 어댑터 클래스

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

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

public class MouseAdapterEx extends JFrame {

	private JLabel la = new JLabel("Hello");

	public MouseAdapterEx() {

		setTitle("Mouse 이벤트 예제");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container c = getContentPane();

		c.addMouseListener(new MyMouseAdapter());

		c.setLayout(null);
		la.setSize(50, 20);
		la.setLocation(30, 30);
		c.add(la);

		setSize(250, 250);
		setVisible(true);
	}

	class MyMouseAdapter extends MouseAdapter {
		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();
	}
}


// 결과
```

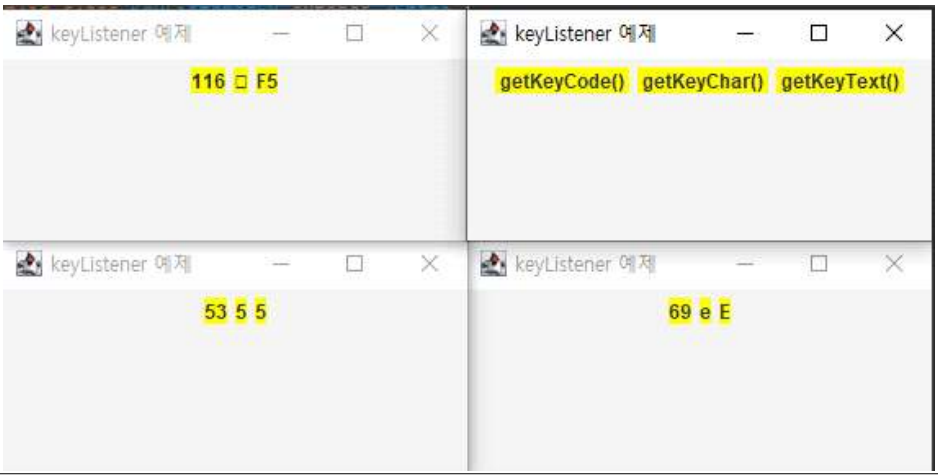


```
// key 이벤트와 포커스
// 키 입력 시 다음 세 경우에 key이벤트 발생
//  1. 키를 누르는 순간
//  2. 누른 키를 떼는 순간
//  3. 누른 키를 떼는 순간(유니코드 키의 경우에만)
// 키 이벤트를 받을 수 있는 조건 : 모든 컴포넌트 가능하지만, 현재 포커스를 가진 컴포넌트
// 포커스
// 컴포넌트나 응용프로그램이 키 이벤트를 독점하는 권한
// 컴포넌트에 포커스 설정 방법 : 다음 2라인의 코드 필요
// -- component.setFouseable(true); // component가 포커스를 받을 수 있도록 설정
// -- component.requestFouse(); // component에 포커스 강제 지정

// 컴포넌트에 포커스 주기
// 스윙 프레임이 만들어질 포커스를 주고자 하는 경우
// -- JFrame의 setVisible(true) 이후에 포커스를 주어야 함
// 예) setVisible(true);
// 예) component.setFocusable(true);
// 예) component.requestFocus();
// 마우스로 컴포넌츠를 클릭할 때 포커스 지정하는 방법
// 언제든지 필요할 때 포커스 줄 수 있음
// 예) component.addMouseListener(new MouseAdapter() {
// 예)     public void mouseClicked(MouseEvent e) {
// 예)             Component c = (Component)e.getSource();
// 예)             component.setFocusable(true);
// 예)             component.requestFocus();
// 예)     }
// 예) });

// keyListener의 메소드와 키
// keyListener의 3개 메소드(실행되는 순서 : 1, 3, 2)
// 1. void keyPressed(keyEvent e){}
// 2. void keyReleased(keyEvent e){}
// 3. void keyTyped(keyEvent e){}
// 컴포넌트에 키 이벤트 리스너 등록
// component.addKeyListener(myKeyListener);

// 가상키 : 가상키 코드는 KeyEvent 클래스에 상수로 선언
```

가상 키	설명	가상 키	설명
VK_0 ~ VK_9	0에서 9까지의 키, '0'~'9'까지의 유니코드 값과 동일	VK_LEFT	왼쪽 방향 키
VK_A ~ VK_Z	A에서 Z까지의 키, 'A'~'Z'까지의 유니코드 값과 동일	VK_RIGHT	오른쪽 방향 키
VK_F1 ~ VK_F24	<F1>~<F24>까지의 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

```
// 입력된 키 판별
// 키가 입력되면 키 정보를 가진 이벤트 객체 생성 : ketEvnet 객체
// 1. 키의 문자 코드(유니코드) 알아내기, char KeyEvnet.getKeyChar()
// -- 눌러진 키가 문자 키인 경우에만 작동하며 문자코드(유니코드) 리턴
// 2. 입력된 키의 가상키 값 알아내기, int KeyEvent.getKeyCode()
// -- 가상 키 값은 KeyEvent 클래스의 상수로 정의됨
// 3. 키 이름 문자열 리턴 String KeyEvent.getKeyText(int keyCode)
// -- Static 메소드로 매개변수 keyCode의 코드 값(가상 키)에 해당하는 키의 이름 문자열 리턴
```

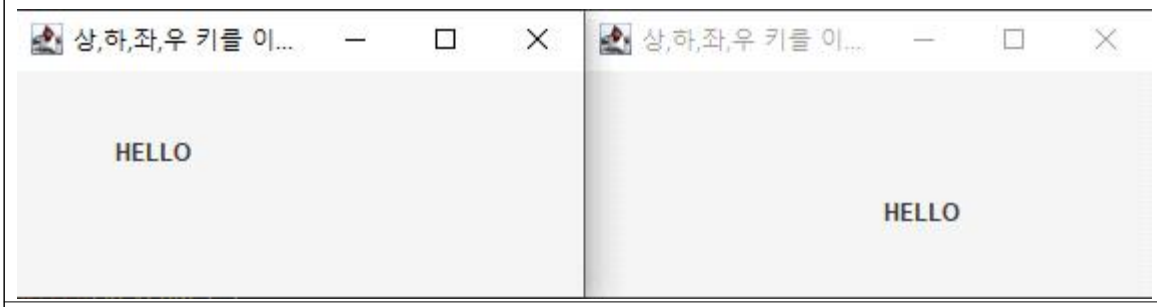
```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class KeyListenerEx extends JFrame {

	private JLabel [] keyMessage;

	public KeyListenerEx() {
		setTitle("keyListener 예제");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container c = getContentPane();
		c.setLayout(new FlowLayout());

		c.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++) {
			c.add(keyMessage[i]);
			// 컴포넌트의 바탕색이 보이도록 하기 위해서는 컴포넌트가 불투명하기 지정될 필요 있음
			keyMessage[i].setOpaque(true);
			keyMessage[i].setBackground(Color.YELLOW);
		}
		setSize(300,150);
		setVisible(true);

		c.setFocusable(true);
		c.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();
	}
}
```



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

public class KeyCodeEx extends JFrame {

	private JLabel la = new JLabel();
	Container c = getContentPane();

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

		c.addKeyListener(new MyKeyListener());

		c.add(la);
		setSize(300,150);
		setVisible(true);

		// 키 입력을 받을 수 있도록 포커스를 준다.
		c.setFocusable(true);
		c.requestFocus();
	}
	class MyKeyListener extends KeyAdapter {
		public void keyPressed(KeyEvent e) {

			int keycode = e.getKeyCode();

			la.setText(e.getKeyText(keycode));

			// % 키를 판별하기 위해 e.getKeyChar() 호출
			if(e.getKeyChar() == '%')
				c.setBackground(Color.YELLOW);
			// F1 키를 판별하기 위해 e.getKeyCode() 호출
			// KeyEvent.VK_F1 값과 비교
			else if(e.getKeyCode() == KeyEvent.VK_F1)
				c.setBackground(Color.GREEN);
		}
	}
	public static void main(String [] args) {
		new KeyCodeEx();
	}
}


// 출력
// %를 입력하면 색상이 바꿈
```

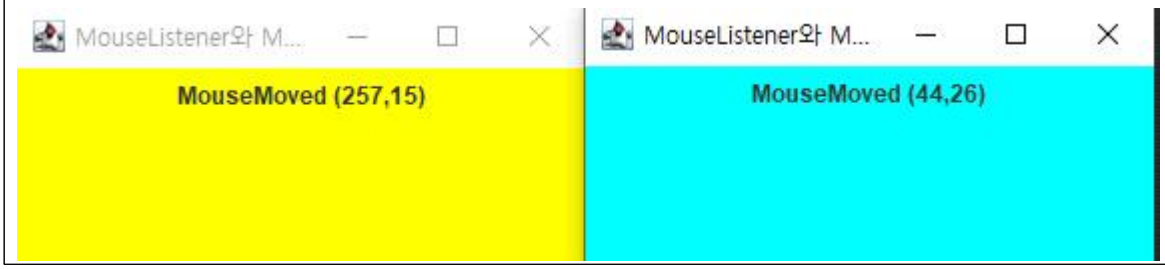
```
// 상하좌우 키로 텍스트 움직이기
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

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

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

		Container c = getContentPane();
		c.setLayout(null);
		c.addKeyListener(new MyKeyListener());

		la.setLocation(50,50);
		la.setSize(100,20);
		c.add(la);

		setSize(300,300);
		setVisible(true);

		c.setFocusable(true);
		c.requestFocus();

		c.addMouseListener(new MouseAdapter() {
			public void mouseClicked(MouseEvent e) {
				Component com = (Component)e.getSource();
				com.setFocusable(true);
				com.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();
	}
}
```

// 출력
// 상하좌우 키를 입력하면 10 픽셀씩 움직인다.




```
// 마우스 이벤트와 마우스 관련 리스너
// 마우스 이벤트(8가지 경우)
// ---------------------사진

// 마우스가 눌러진 위치에서 떼어지는 경우 메소드 호출 순서
// mousePressed(), mouseReleased(), mouseClicked()
// 마우스가 드래그될 때 호출되는 메소드 호출 순서
// mousePressed(), mouseDragged(), mouseDragged(),..., mouseDragged(), mouseReleased()

// 마우스 리너스 달기
// MouseListener의 5 개의 이벤트를 처리하는 경우
// mouseEntered(), mouseExited(), mousePressed(), mouseReleased(),mouseClicked()
// 마우스 리스너 등록
// component.addMouseListener(myMouseListener);

// MouseMotionListener의 이벤트도 함께 처리하고자 하는 경우
// mouseDragged(), mouseMoved()
// 마우스 모션 리스너 등록 필요
// component.addMouseMotionListener(myMouseMotionListener);

// MouseEvent 객체로부터 얻을 수 있는 정보
// 1. 마우스 포인터의 위치(int getX(), int getY(),Point getPoint())
// 2. 입력된 마우스 버튼(int getButton())
// 3. 마우스 클릭 횟수(int getClickCount())
```

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class MouseListenerAllEx extends JFrame {
	private JLabel la = new JLabel("No Mouse Event");

	public MouseListenerAllEx() {
		setTitle("MouseListener와 MouseMotionListener 예제");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		Container c = getContentPane();
		c.setLayout(new FlowLayout());

		MyMouseListener listener = new MyMouseListener();
		c.addMouseListener(listener);
		c.addMouseMotionListener(listener);

		c.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) {
			Component c = (Component)e.getSource();
			c.setBackground(Color.CYAN);
		}
		public void mouseExited(MouseEvent e) {
			Component c = (Component)e.getSource();
			c.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();
	}
}
// 출력
// 마우스를 올려두면 MouseMoved, 마우스를 클릭하면 MousePressed, 마우스를 드래그하면 MouseDragged로 표현한다.
// 커서를 창 밖으로 놓으면 노란색 창으로 바뀐다.
```



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

public class ClickAndDoubleClickEx extends JFrame {
	public ClickAndDoubleClickEx() {
		setTitle("Click and DoubleClick 예제");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container c = getContentPane();
		c.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);
				Component c = (Component)e.getSource();
				c.setBackground(new Color(r,b,g));
			}
		}
	}
	public static void main(String [] args) {
		new ClickAndDoubleClickEx();
	}
}

// 출력 : 더블클릭시 랜덤으로 색상 바뀜
```