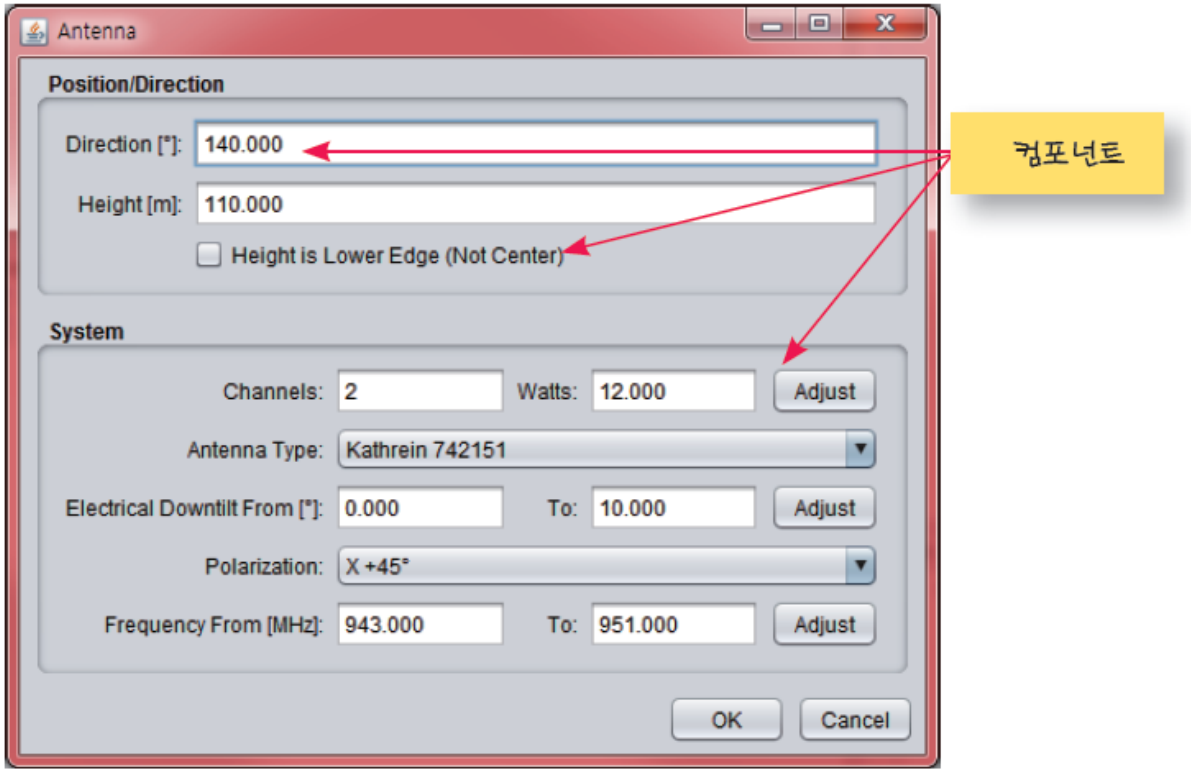
그래픽 사용자 인터페이스
이벤트 객체, 리스너

514760
2026년 봄학기
6/3/2026
박경신

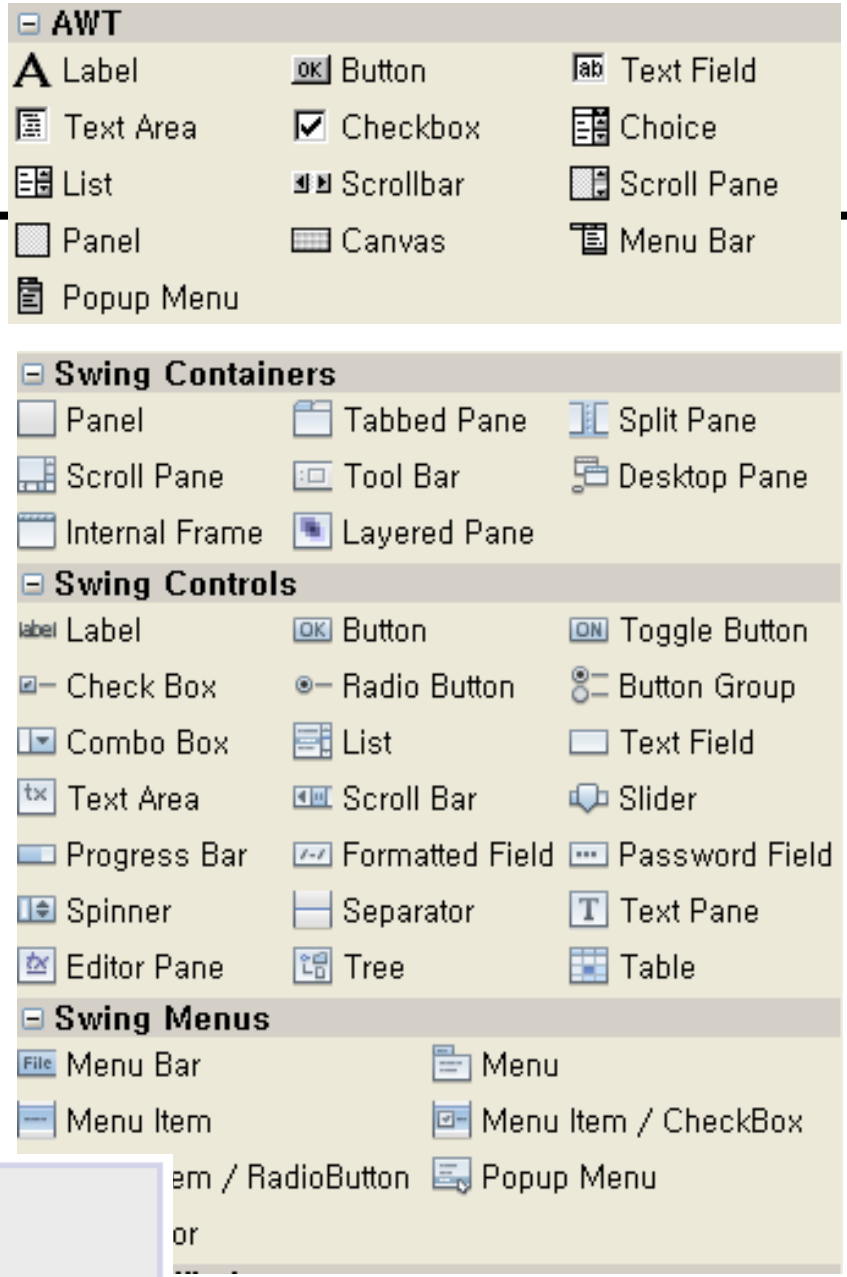
그래픽 사용자 인터페이스

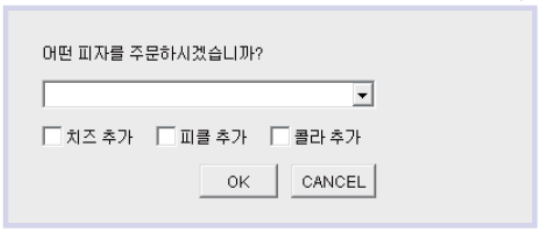
- 그래픽 사용자 인터페이스(Graphical User Interface, 간단히 GUI)는 컴포넌트들로 구성된다.

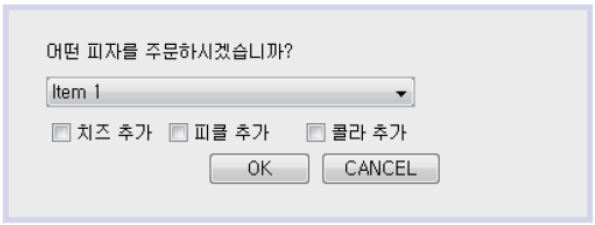


자바에서 GUI의 종류

- AWT(Abatract Windows Toolkit)
  - 운영 체제가 제공하는 자원을 이용하여서 컴포넌트를 생성한다.
- SWING
  - 스윙 컴포넌트가 자바로 작성되어 있기 때문에 어떤 플랫폼에서도 일관된 화면을 보여줄 수 있다.



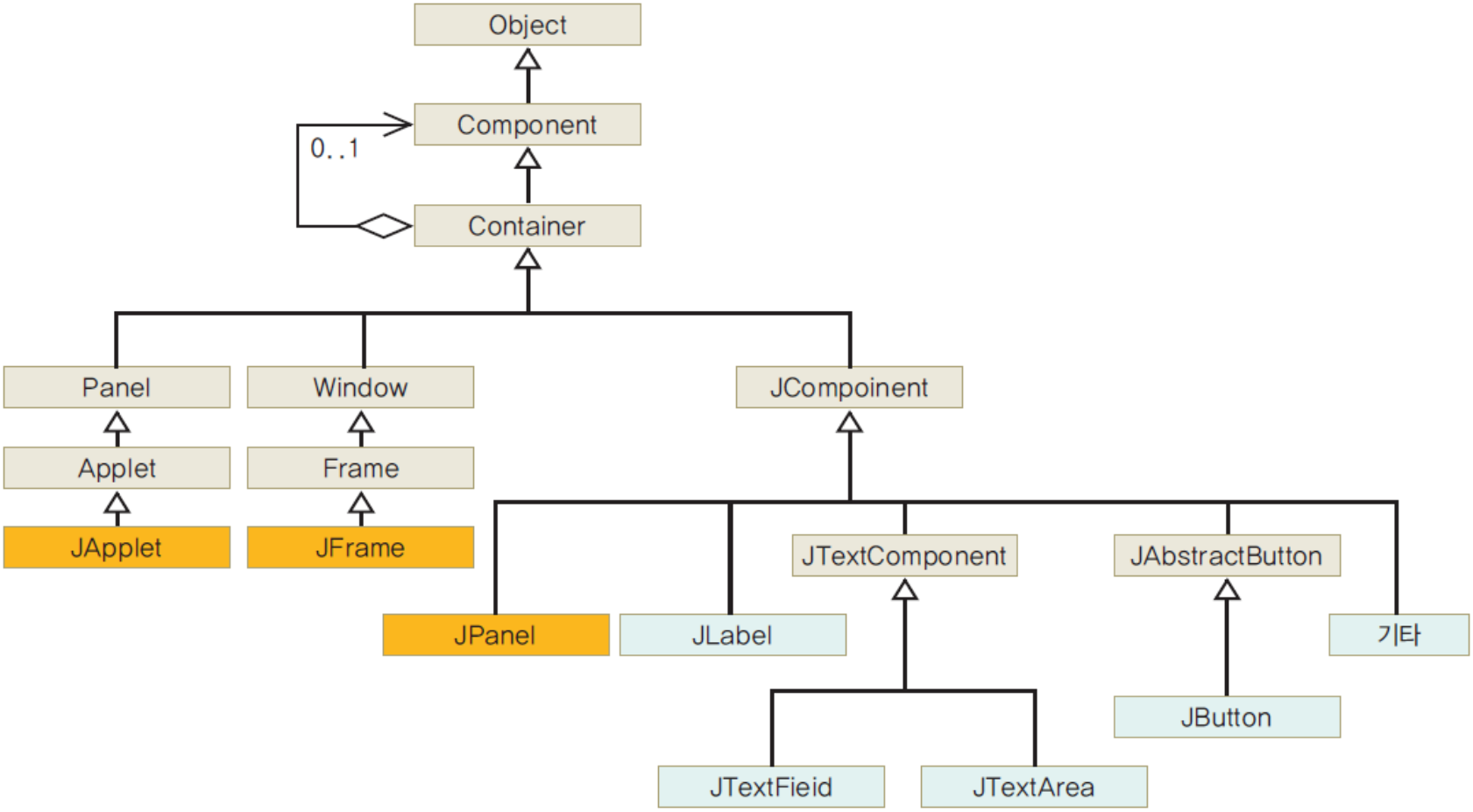




AWT와 SWING

컴포넌트	AWT 버전	스윙 버전
버튼	Button	JButton
레이블	Label	JLabel
리스트	List	JList
…		
패스워드필드	없음	JPasswordField
슬라이더	없음	JSlider

SWING 클래스 계층구조



SWING의 특징

- 스윙 GUI 컴포넌트
  - 형식화된 텍스트 입력이나 패스워드 필드 동작과 같은 복잡한 기능들이 제공
- 자바 2D API
  - 그림이나 이미지, 애니메이션 기능을 제공
  - 교체가능한 룩앤필(Look-and-Feel) 지원
- 데이터 전송
  - 자르기, 복사, 붙이기, 드래그앤 드롭 등의 데이터 전송 기능 제공
  - 되돌리기(undo)와 되풀이(redo) 기능을 손쉽게 제공

스윙 패키지

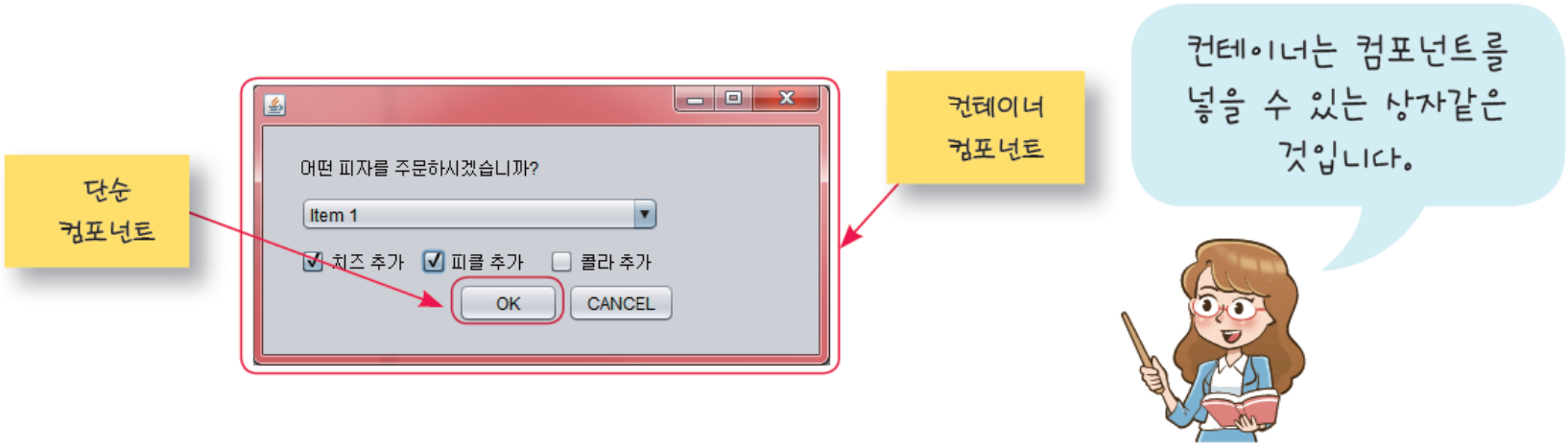
javax.accessibility	javax.swing.plaf	javax.swing.text
javax.swing	javax.swing.plaf.basic	javax.swing.text.html
javax.swing.border	javax.swing.plaf.metal	javax.swing.text.html.parser
javax.swing.colorchooser	javax.swing.plaf.multi	javax.swing.text.rtf
javax.swing.event	javax.swing.plaf.synth	javax.swing.tree
javax.swing.filechooser	javax.swing.table	javax.swing.undo

하지만 대부분의 프로그램은 스윙 API 중에서 아주 작은 부분 집합만을 사용한다. 따라서 대부분의 경우 다음의 두 가지의 패키지만을 포함하면 된다.

* javax.swing
* javax.swing.event

컨테이너와 컴포넌트

- **기본 컴포넌트**
  - JButton, JLabel, JCheckbox, JChoice, JList, JMenu, JTextField, JScrollbar, JTextArea, JCanvas 등이 여기에 속한다.
- **컨테이너 컴포넌트**
  - 다른 컴포넌트를 안에 포함할 수 있는 컴포넌트로서 JFrame, JDialog, JApplet, JPanel, JScrollPane 등이 여기에 속한다.



컨테이너의 종류

- **최상위 컨테이너**
  - 절대 다른 컨테이너 안에 포함될 수 없는 컨테이너를 의미한다. 프레임(JFrame), 다이알로그(JDialog), 애플릿(JApplet) 등이 여기에 해당된다.

JApplet

JDialog

JFrame

- **일반 컨테이너**
  - 다른 컨테이너 안에 포함될 수 있는 컨테이너로 패널(JPanel), 스크롤 페인(JScrollPane) 등을 의미한다.

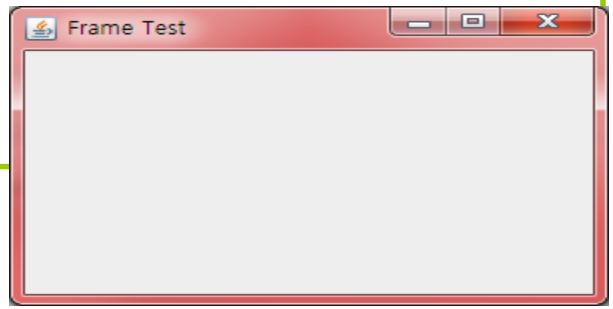
JPanel

JScrollPane

프레임 생성 #1

```
import javax.swing.*;

public class FrameTest {

    public static void main(String[] args) {

        JFrame f = new JFrame("Frame Test");

        f.setSize(300, 200);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setVisible(true);
    }

}
```

JFrame의 객체 생성



프레임 생성 #2

```
import javax.swing.*;

public class MyFrame extends JFrame {
        public MyFrame() {
                setSize(300, 200);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setTitle("MyFrame");
                setVisible(true);
        }
}
```

Jframe을 상속하여 MyFrame을 정의

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

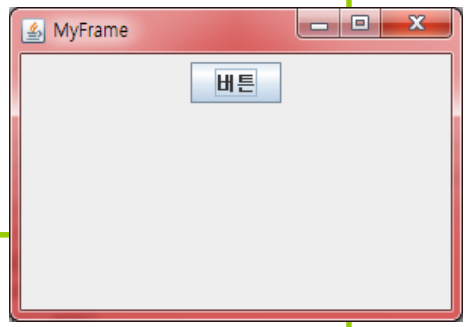
MyFrame의 객체 생성

컴포넌트 생성과 추가

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

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

		setLayout(new FlowLayout());
		 JButton button = new JButton("버튼");
		 this.add(button);
		setVisible(true);
	}
}
```

배치 관리자 설정!

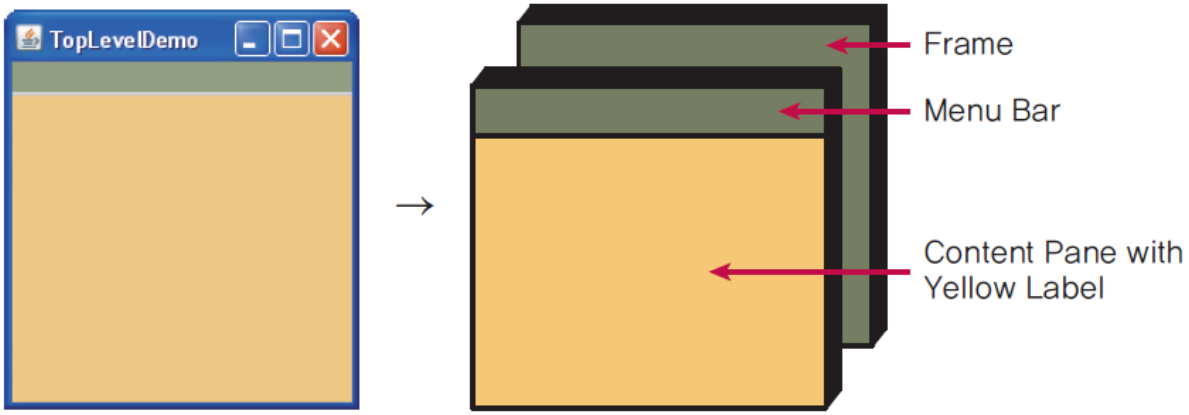
컴포넌트 생성 및 추가




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

JFrame 클래스

- 컨테이너는 컴포넌트들을 트리(tree) 형태로 저장한다. 최상위 컨테이너는 이 트리의 루트 노드가 된다.
- 최상위 컨테이너는 내부에 콘텐트 페인(content pane)을 가지고 있다. 여기에 화면에 보이는 컴포넌트를 저장한다.
- 최상위 컨테이너에는 메뉴바를 추가할 수 있다.

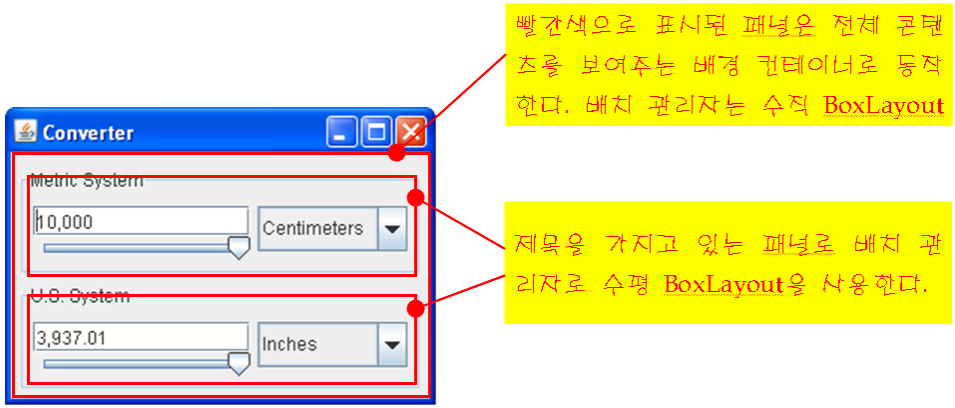


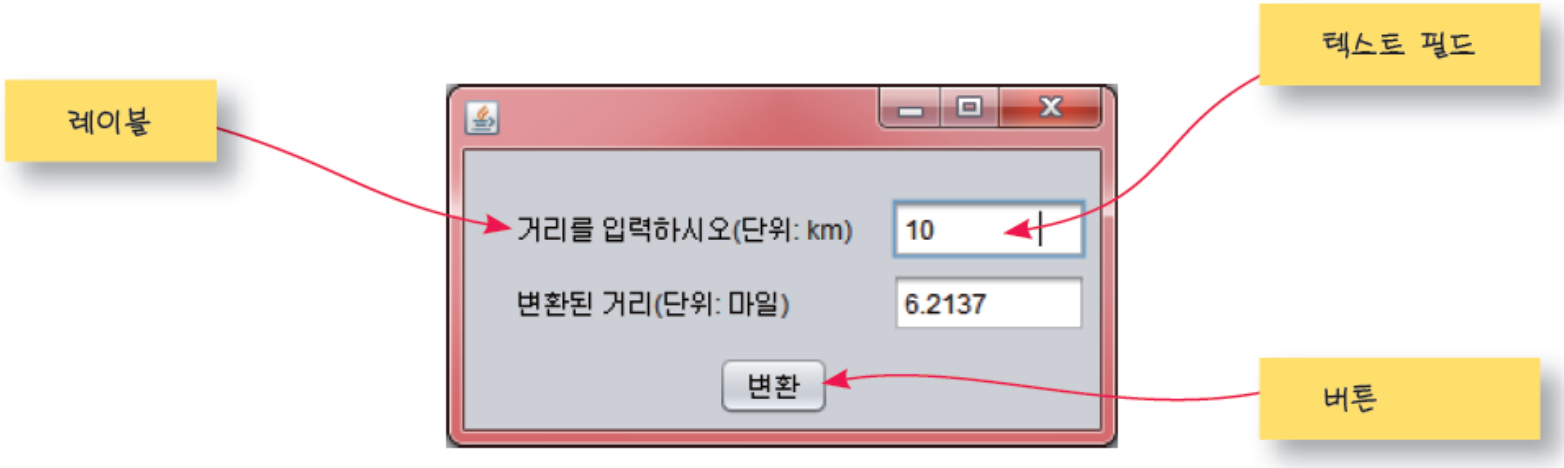
JFrame 클래스 주요 메소드

- *setLocation(x, y) , setBounds(x, y, width, height), setSize(width, height)*
  - 프레임의 위치와 크기를 설정한다.
- *setIconImage(IconImage)*
  - 윈도우 시스템에 타이틀 바, 태스크 스위처에 표시할 아이콘을 알려준다.
- *setTitle()*
  - 타이틀 바의 제목을 변경한다.
- *setResizable(boolean)*
  - 사용자가 크기를 조절할 수 있는지를 설정한다.

기본 컴포넌트

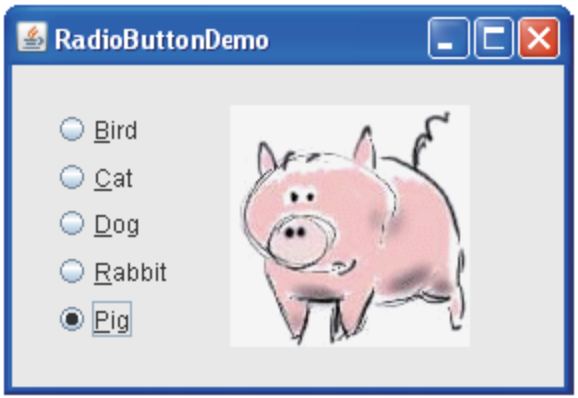
- 패널(panel)
- 레이블(label)
- 버튼(button)
- 텍스트 필드(text field)

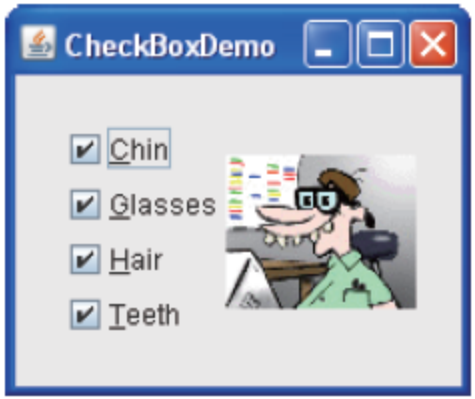




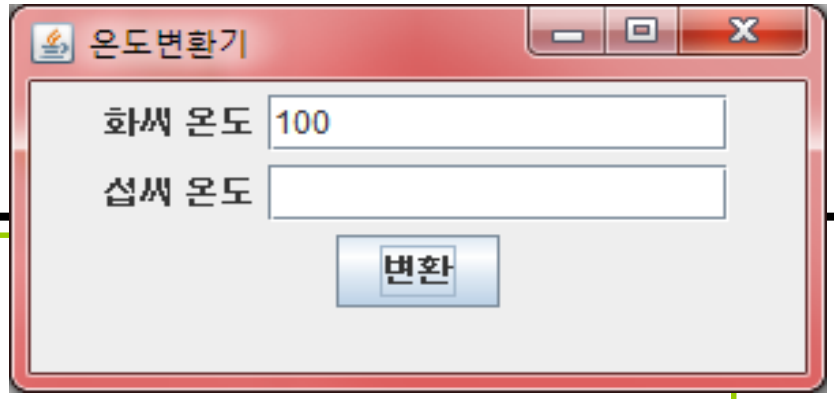
버튼의 종류

- JButton - 가장 일반적인 버튼
- JCheckBox - 체크박스 버튼
- JRadioButton - 라디오 버튼은 그룹 중 하나의 버튼만 체크할 수 있다.





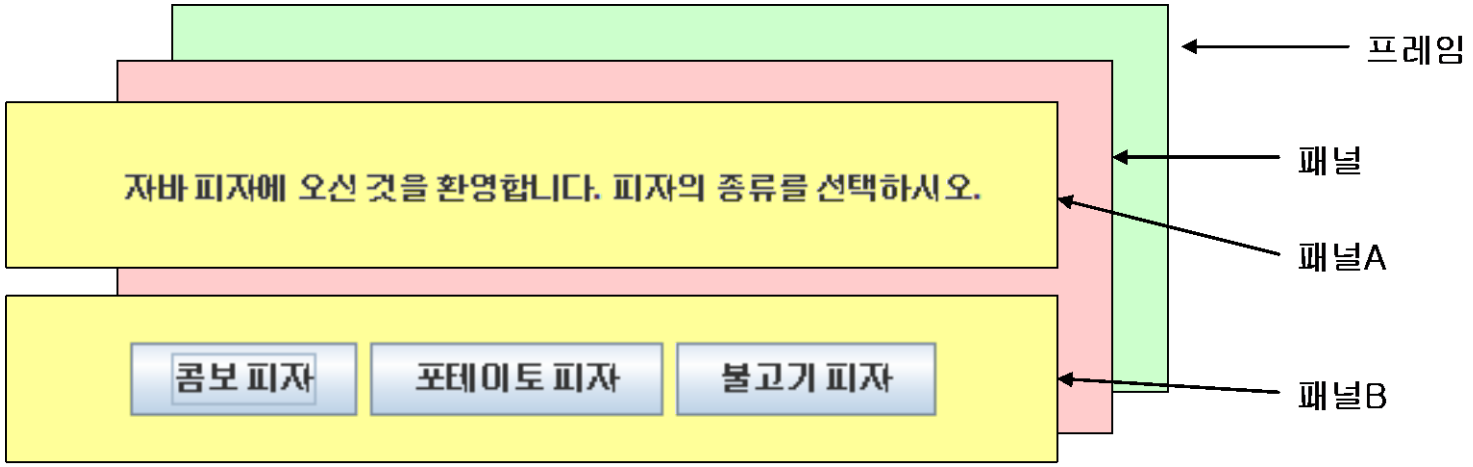
LAB: 온도 변환기

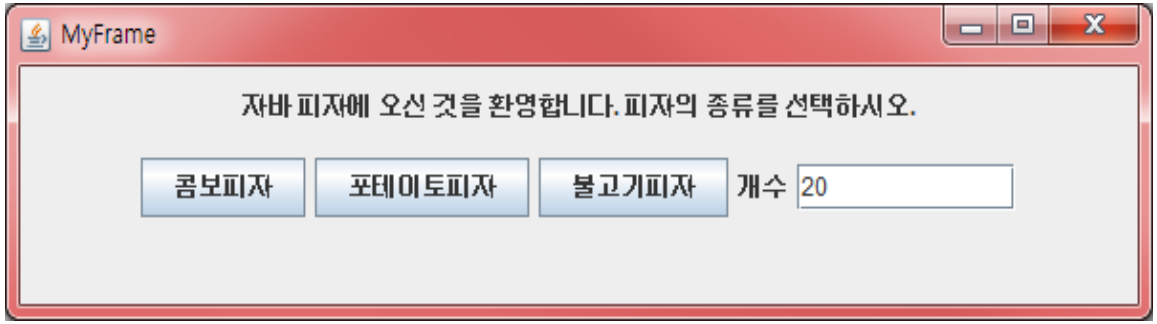



```
public class TemperatureConverter {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        JPanel panel = new JPanel();
        f.add(panel);
        JLabel label1 = new JLabel("화씨 온도");  JLabel label2 = new JLabel("섭씨 온도");
        JTextField field1 = new JTextField(15);  JTextField field2 = new JTextField(15);
        JButton button = new JButton("변환");
        panel.add(label1);  panel.add(field1);
        panel.add(label2);  panel.add(field2); panel.add(button);
        f.setSize(300, 150);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setTitle("온도변환기");
        f.setVisible(true);
    }
}
```

LAB: 피자 주문 화면

- 패널 안에 다른 패널이 포함될 수 있다. 이것을 이용하여서 다음 그림처럼 프로그램의 화면을 디자인하라.

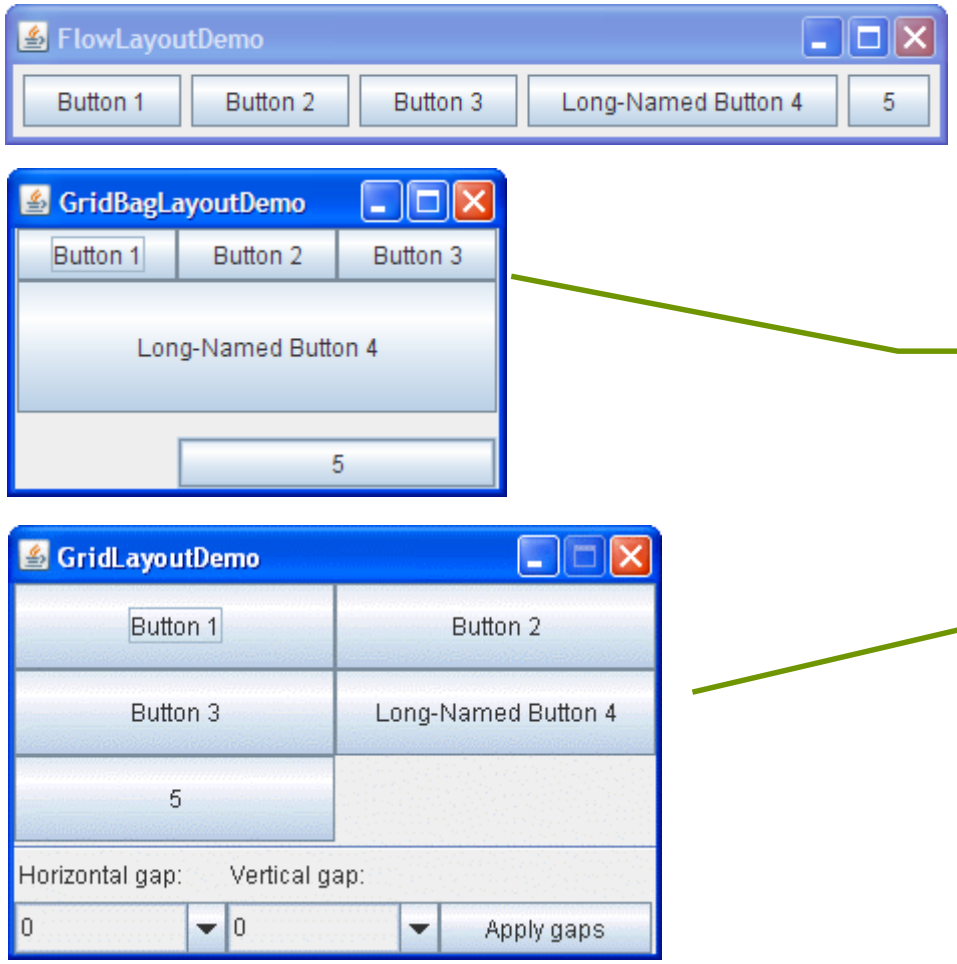




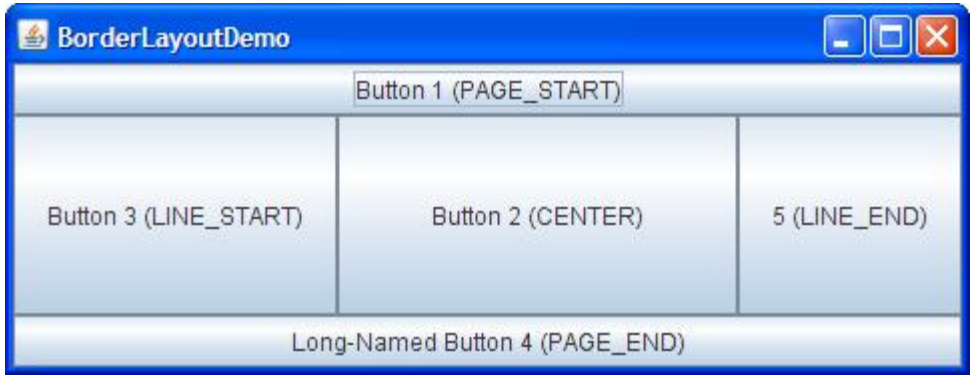
```
public class MyFrame extends JFrame {
    public MyFrame() {
        setSize(600, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("MyFrame");
        JPanel panel = new JPanel();
        JPanel panelA = new JPanel();
        JPanel panelB = new JPanel();
        JLabel label1 = new JLabel("자바 피자에 오신 것을 환영합니다. 피자의 종류를 선택
하시오.");
        panelA.add(label1);
        JButton button1 = new JButton("콤보피자");
        JButton button2 = new JButton("포테이토피자");
        JButton button3 = new JButton("불고기피자");
        panelB.add(button1);
        panelB.add(button2);
        panelB.add(button3);
        JLabel label2 = new JLabel("개수");
        JTextField field1 = new JTextField(10);
        panelB.add(label2);
        panelB.add(field1);
        panel.add(panelA);
        panel.add(panelB);
        add(panel);
        setVisible(true);
    }
}
```

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

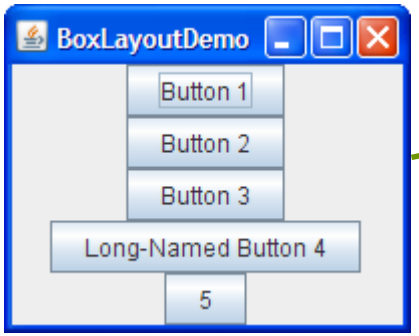
배치 관리자의 종류


FlowLayoutDemo
Button 1
Button 2
Button 3
Long-Named Button 4
5
GridBagLayoutDemo
Button 1
Button 2
Button 3
Long-Named Button 4
5
GridLayoutDemo
Button 1
Button 2
Button 3
Long-Named Button 4
5
Horizontal gap:
Vertical gap:
0
0
Apply gaps
FlowLayout
GridBagLayout
GridLayout

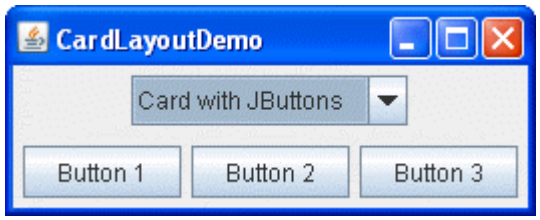
배치 관리자의 종류

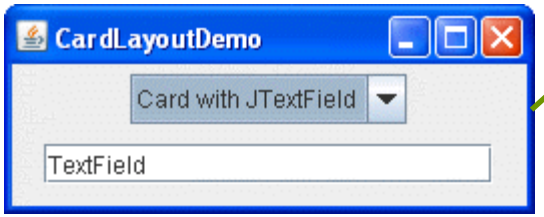


BorderLayout



BoxLayout





CardLayout

배치 관리자의 설정

1. 생성자를 이용하는 방법

```
JPanel panel = new JPanel(new BorderLayout());
```

2. setLayout() 메소드 이용

```
panel.setLayout(new FlowLayout());
```

- 프로그래머가 컴포넌트의 크기와 힌트를 배치 관리자에게 주고 싶은 경우에는 setMinimumSize(), setPreferredSize(), setMaximumSize() 메소드를 사용
  - button.setMaximumSize(**new** Dimension(300, 200)); // 최대 크기 힌트
  - button.setAlignmentX(JComponent.*CENTER_ALIGNMENT*); // 중앙 정렬 힌트

BorderLayout 클래스

생성자 또는 메소드	설명
BorderLayout(int hgap, int vgap)	컴포넌트 사이의 수평 간격 hgap과 수직 간격 vgap을 을 가지는 BorderLayout 객체 생성
setHgap(int)	컴포넌트 사이의 수평 간격 설정(단위는 픽셀)
setVgap(int)	컴포넌트 사이의 수직 간격 설정

GridLayout 클래스

생성자	설명
GridLayout(int rows, int cols)	rows 행과 cols 열을 가지는 GridLayout 객체를 생성한다. 만약 rows나 cols가 0이면 필요한 만큼의 행이나 열이 만들어진다.
GridLayout(int rows, int cols, int hgap, int vgap)	rows 행과 cols 열을 가지는 GridLayout 객체를 생성한다. hgap과 vgap은 컴포넌트 사이의 수평 간격과 수직 간격으로 단위는 픽셀이다.

절대 위치로 배치하기

1. 배치 관리자를 null로 설정한다.

   setLayout(**null**);

2. add() 메소드를 사용하여 컴포넌트를 컨테이너에 추가한다.

   Button b = Button("Absolute Position Button");

   add(b);

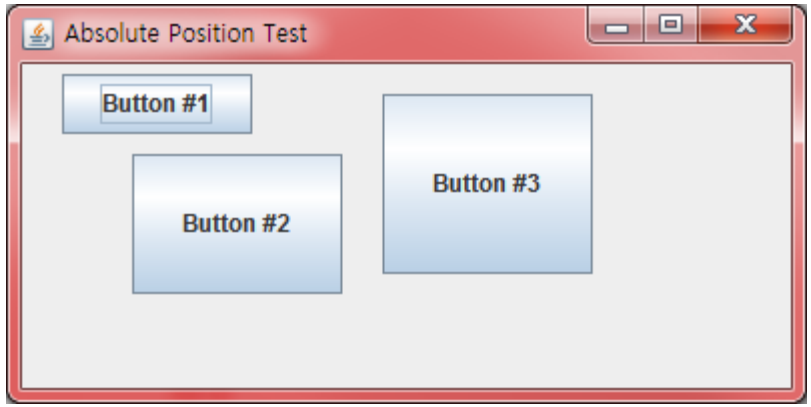
3. setBounds() 메소드를 사용하여 절대 위치와 크기를 지정한다.

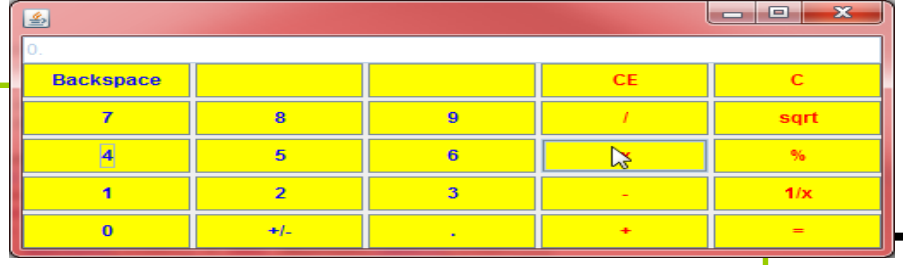
   b.setBounds(x, y, w, h);

4. 컴포넌트의 repaint() 메소드를 호출한다.

   b.repaint();

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class MyFrame extends JFrame {
	JButton b1;
	private JButton b2, b3;
	public MyFrame() {
		setTitle("Absolute Position Test");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(300, 200);
		JPanel p = new JPanel();
		p.setLayout(null);
		b1 = new JButton("Button #1");		p.add(b1);
		b2 = new JButton("Button #2");		p.add(b2);
		b3 = new JButton("Button #3");		p.add(b3);
		b1.setBounds(20, 5, 95, 30);
		b2.setBounds(55, 45, 105, 70);
		b3.setBounds(180, 15, 105, 90);
		add(p);
		setVisible(true);
	}
}
public class AbsoluteTest {
	public static void main(String args[]) {
		MyFrame f=new MyFrame();
	}
}
```






```
public class Calculator extends JFrame {
  private JPanel panel; private JTextField tField;
  private JButton[] buttons;
   private String[] labels = { "Backspace", "", "", "CE", "C",
   "7", "8", "9", "/", "sqrt", "4", "5", "6", "x", "%","1", "2", "3", "-", "1/x", "0", "+/-", ".", "+", "=" };
   public Calculator() {
      tField = new JTextField(35);        panel = new JPanel();
      tField.setText("0.");     tField.setEnabled(false);     panel.setLayout(new GridLayout(5, 5, 3, 3));
      buttons = new JButton[25];
      int index = 0;
      for (int rows = 0; rows < 5; rows++) {
                for (int cols = 0; cols < 5; cols++) {
                                buttons[index] = new JButton(labels[index]);
                                if( cols >= 3 )
                                                buttons[index].setForeground(Color.red);
                                else
                                                buttons[index].setForeground(Color.blue);
                                buttons[index].setBackground(Color.yellow);
                                panel.add(buttons[index]);
                                index++;
                }
       }
       add(tField, BorderLayout.NORTH);   add(panel, BorderLayout.CENTER);
       setVisible(true);
        pack();
   }
  public static void main(String args[]) {
                Calculator s = new Calculator();
   }
}
```

이벤트-구동 프로그래밍

- **이벤트-구동 프로그래밍(event-driven programming):**
  - 프로그램의 실행이 이벤트의 발생에 의하여 결정되는 방식

이벤트 처리 과정

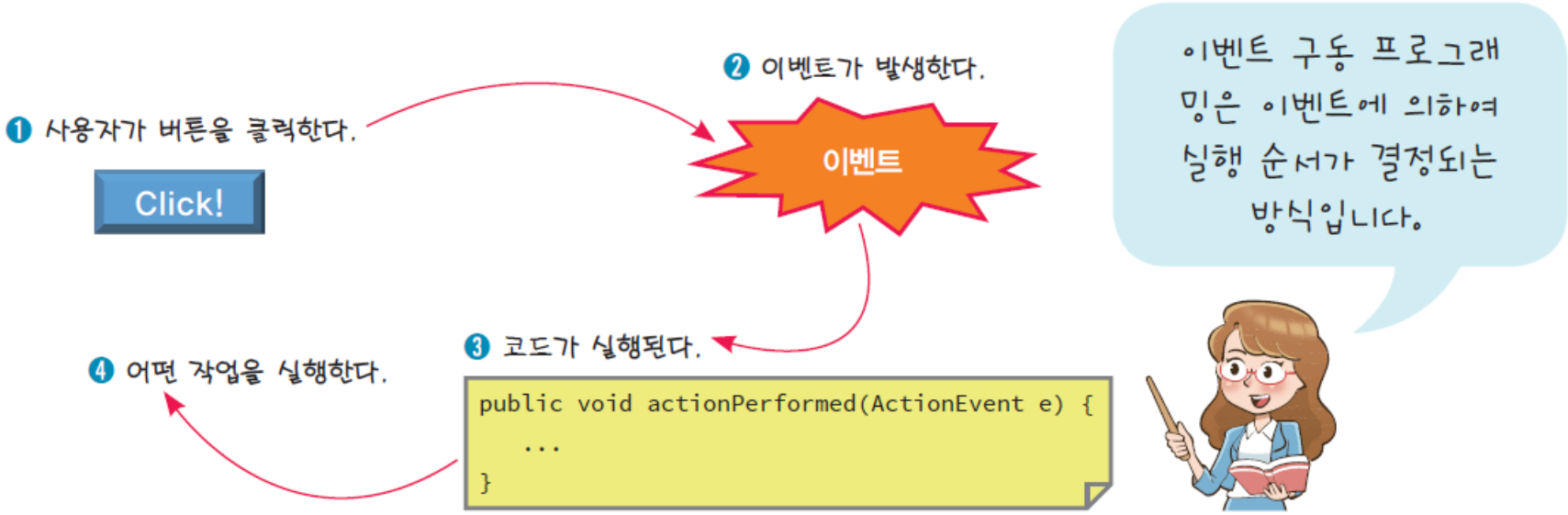
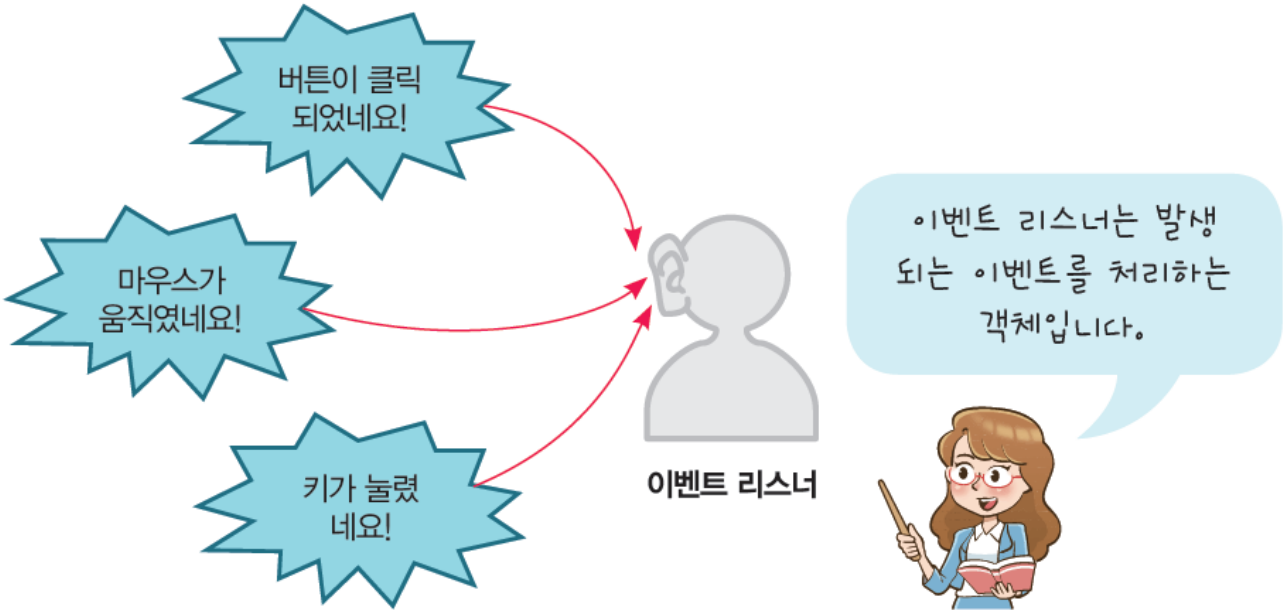


그림 10-1 • 이벤트 처리의 절차

이벤트 리스너

- 발생된 이벤트 객체에 반응하여서 이벤트를 처리하는 객체를 이벤트 리스너(event listener)라고 한다.



이벤트 처리 과정

1. 이벤트 리스너 클래스를 작성한다.

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

2. 이벤트 리스너를 이벤트 소스에 등록한다.

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

이벤트 리스너를 컴포넌트에 붙인다.

이벤트 객체

- EventObject 클래스를 상속받는다.
- Ex: MouseEvent 클래스

java.lang.Object
  java.util.EventObject
    java.awt.AWTEvent
      java.awt.event.ComponentEvent
        java.awt.event.InputEvent
          java.awt.event.MouseEvent

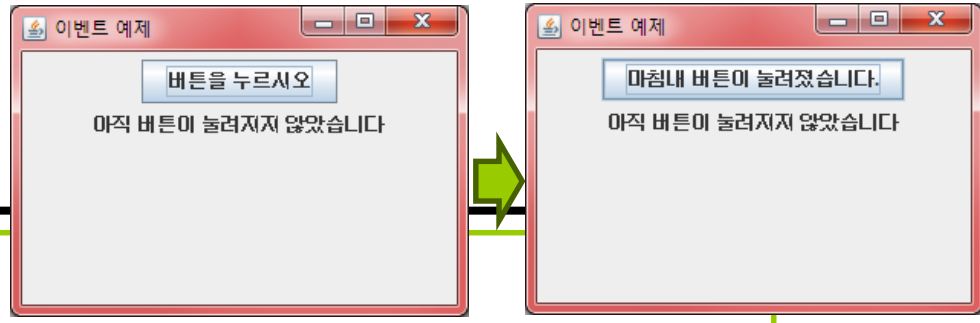
- 이벤트를 발생시킨 이벤트 소스 등의 여러 가지 정보를 제공한다.

```
public void actionPerformed(ActionEvent e) {
        button = (JButton)e.getSource();
                ...
}
```

이벤트 처리기 작성 방법

1. 독립적인 클래스로 이벤트 처리기를 작성
2. 내부 클래스로 이벤트 처리기를 작성
3. 프레임 클래스에 이벤트 처리를 구현
4. 무명 클래스를 사용하는 방법
5. 람다식을 이용하는 방법

독립적인 클래스 작성




```
import javax.swing.*;
import java.awt.event.*;
class MyListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
            JButton button = (JButton) e.getSource();
            button.setText("마침내 버튼이 눌려졌습니다.");
   }
}
class MyFrame extends JFrame {
   private JButton button; private JLabel label;
   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");
            label = new JLabel("아직 버튼이 눌려지지 않았습니다");
            button.addActionListener(new MyListener());
            panel.add(button);
            panel.add(label);
            this.add(panel);
            this.setVisible(true);
   }
}
```

```
public class ActionEventTest1 {
      public static void main(String[] args) {
            MyFrame t = new MyFrame();
      }
}
```

내부 클래스 방법

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

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

내부 클래스 label 에 접근할 수 있다.

MyFrame에서 이벤트도 처리하는 방법

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

```
...
class MyFrame extends JFrame implements ActionListener {
        ...
        public MyFrame() {
                ...
                button = new JButton("버튼을 누르시오");
                label = new JLabel("아직 버튼이 눌려지지 않았습니다");
                button.addActionListener(this);
                ...
        }
        public void actionPerformed(ActionEvent e) {
                if (e.getSource() == button) {
                        label.setText("마침내 버튼이 눌려졌습니다.");
                }
        }
}
...
```

이벤트도 처리

무명 클래스를 사용하는 방법

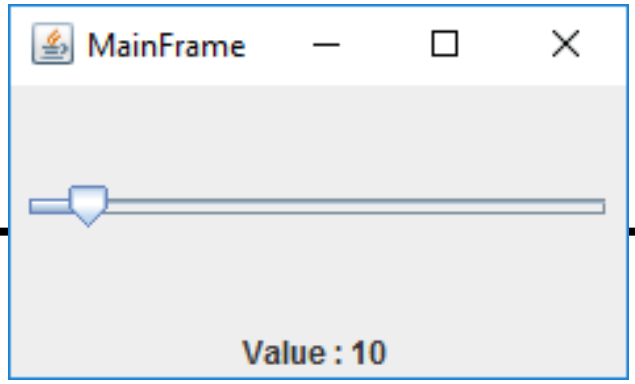
```
class MyFrame extends JFrame {
        ...
        public MyFrame() {
                ...
                button = new JButton("버튼을 누르시오");
                button.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                if (e.getSource() == button) {
                                        label.setText("마침내 버튼이
                                                눌려졌습니다.");
                                }
                        }
                });
                ...
        }
}
```

안드로이드에서 많이 사용된다!

람다식을 이용하는 방법

```
import javax.swing.*;
class MyFrame extends JFrame {
        private JButton button;
        private JLabel label;
        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");
                label = new JLabel("아직 버튼이 눌려지지 않았습니다");
                button.addActionListener(e -> {
                        label.setText("마침내 버튼이 눌려졌습니다.");
                });
                panel.add(button);
                panel.add(label);
                this.add(panel);
                this.setVisible(true);
        }
}
```

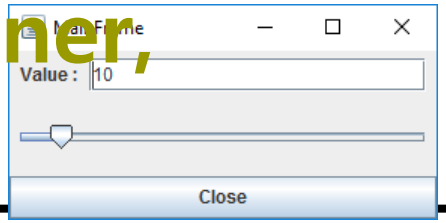
Slider 예제



- JLabel, JSlider를 이용한 슬라이더 값을 레이블에 출력하는 간단한 GUI를 구현한다.

```
public class MainFrame extends JFrame {
    JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 100, 10);
    JLabel label = new JLabel("Value : ", JLabel.CENTER);

    slider.addChangeListener(new ChangeListener() {
                public void stateChanged(ChangeEvent e) {
                    label.setText("Value : " +
((JSlider)e.getSource()).getValue());
                }
            });
}
```

Slider ChangeListener, KeyListener, ActionListener 예제



- JLabel, JSlider, JTextField, JButton를 이용한 슬라이더 값을 텍스트필드에 출력하는, 버튼으로 종료하는 간단한 GUI를 구현한다.
  - *public class MainFrame extends JFrame implements ChangeListener, KeyListener, ActionListener {*

```
JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 100, 10);
JLabel label = new JLabel("Value : ", JLabel.CENTER);
JTextField textfield = new JTextField("10", 20);
JButton button = new JButton("Close");
public void stateChanged(ChangeEvent e) { ..}
public void keyTyped(KeyEvent e) {    .. }
public void keyPressed(KeyEvent e) { .. }
public void keyReleased(KeyEvent e) { ..}
public void actionPerformed(ActionEvent e) { .. }
}
```

Slider ChangeListener, KeyListener, ActionListener 예제

```
public void stateChanged(ChangeEvent e) {
        textfield.setText("" + ((JSlider)e.getSource()).getValue()); // int -> String
}
public void keyTyped(KeyEvent e) {   }
public void keyPressed(KeyEvent e) {   }
public void keyReleased(KeyEvent e) {
        int key = e.getKeyCode();
        if (key == KeyEvent.VK_ENTER) {
                Toolkit.getDefaultToolkit().beep();
                System.out.println("ENTER pressed");
                JTextField textfield = (JTextField) e.getSource();
                int value = Integer.parseUnsignedInt(textfield.getText()); // String -> in
                slider.setValue(value);
        }
}
```

이벤트의 분류

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

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

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

저수준 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)

의미적 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

이벤트 발생원의 식별

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

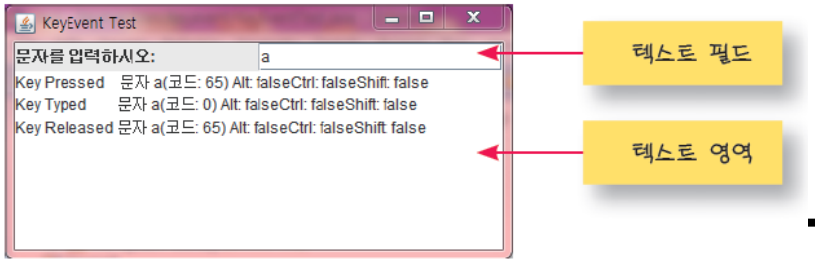
```
public void actionPerformed(ActionEvent e) {
    if (e.getSource () == button1){
        …
    }
}
```

Key 이벤트

- KeyListener 인터페이스 구현

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

Key 이벤트 예제



- 키보드에서 문자가 입력되면 문자 코드와 키코드, ALT나 SHIFT 키의 상태를 텍스트 영역에 출력한다.

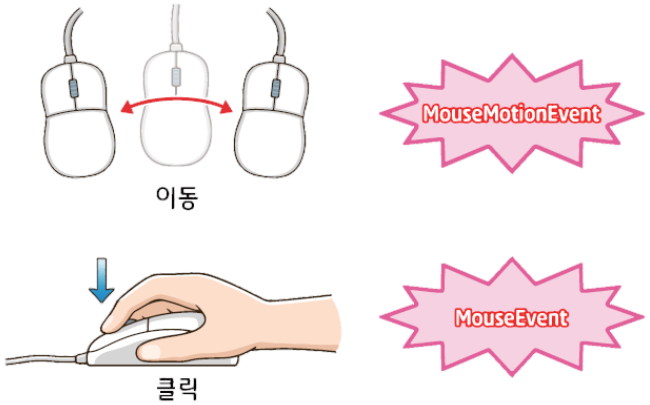
```
public class KeyEventTest extends JFrame implements KeyListener {
        private JPanel panel;  private JTextField field; private JTextArea area;
        public KeyEventTest() {
                panel = new JPanel(new GridLayout(0, 2));
                panel.add(new JLabel("문자를 입력하시오: "));
                field = new JTextField(10);
                panel.add(field);
                area = new JTextArea(3, 30);
                add(panel, BorderLayout.NORTH);
                add(area, BorderLayout.CENTER);
                field.addKeyListener(this);
                setTitle("KeyEvent Test");
                setSize(400, 200);
                setVisible(true);
        }
        public static void main(String[] args) {
                new KeyEventTest();
        }
```

Key 이벤트 예제

```
        public void keyTyped(KeyEvent e) {
                display(e, "Key Typed ");
        }
        public void keyPressed(KeyEvent e) {
                display(e, "Key Pressed ");
        }
        public void keyReleased(KeyEvent e) {
                display(e, "Key Released ");
        }
        protected void display(KeyEvent e, String s) {
                char c = e.getKeyChar();
                int keyCode = e.getKeyCode();
                String modifiers = "Alt: " + e.isAltDown() + "Ctrl: "
                                + e.isControlDown() + "Shift: " + e.isShiftDown();
                area.append("" + s + "문자 " + c + "(코드: " + keyCode + ") " + modifiers
                                + "₩n");
        }
}
```

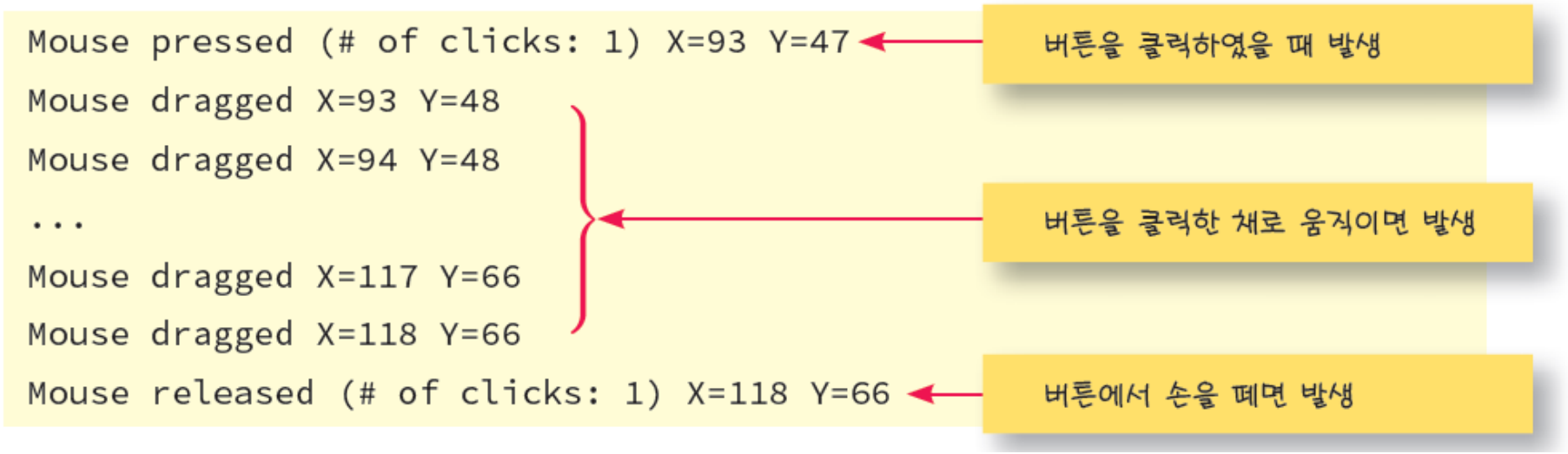
Mouse 이벤트

메소드	설 명
mouseClicked(MouseEvent e)	사용자가 컴포넌트를 클릭한 경우에 호출된다.
mouseEntered(MouseEvent e)	마우스 커서가 컴포넌트로 들어가면 호출된다.
mouseExited(MouseEvent e)	마우스 커서가 컴포넌트에서 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 드래그하면 호출된다.
mouseMoved(MouseEvent e)	마우스가 클릭되지 않고 이동하는 경우에 호출된다.



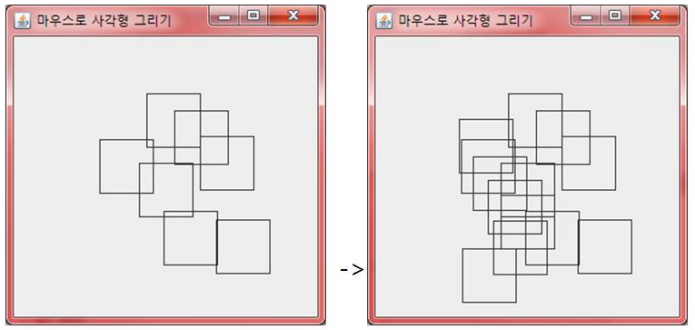
Mouse 이벤트 예제

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

마우스 이벤트 처리기를 붙인다.

```
        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);
        }
        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();
        }
}
```

LAB: 사각형 그리기



- 마우스를 클릭할 때마다 사각형이 화면에 그려지는 예제를 작성하여 보자.

```
class Rectangle {
        int x, y, w, h;
}
class MyPanel extends JPanel implements MouseListener {
        BufferedImage img = null;
        int img_x = 0, img_y = 0;
        Rectangle[] array = new Rectangle[100];
        int index = 0;
        public MyPanel() {
                this.addMouseListener(this);
        }
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                for (Rectangle r : array)
                        if (r != null)
                                g.drawRect(r.x, r.y, r.w, r.h);
        }
```

```
    @Override
    public void mousePressed(MouseEvent e) {
            if (index > 100) return;
            array[index] = new Rectangle();
            array[index].x = e.getX();
            array[index].y = e.getY();
            array[index].w = 50;
            array[index].h = 50;
            index++;
            repaint();
    }
    @Override
    public void mouseReleased(MouseEvent e) { }
    @Override
    public void mouseClicked(MouseEvent e) { }
    @Override
    public void mouseEntered(MouseEvent e) { }
    @Override
    public void mouseExited(MouseEvent e) { }
}
```

```
public class MouseEventTest extends JFrame {
    public MouseEventTest() {
        setSize(300, 300);
        setTitle("마우스로 사각형 그리기");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add(new MyPanel());
        setVisible(true);
    }
    public static void main(String[] args) {
        MouseEventTest s = new MouseEventTest();
    }
}
```

Adapter 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

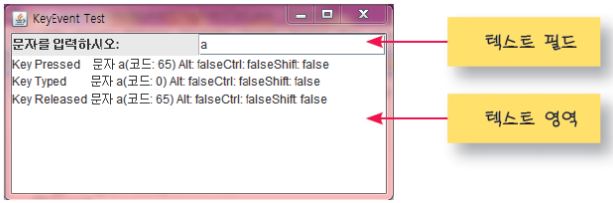
Listener를 사용하는 경우

```
public class MyClass implements MouseListener {
        public MyClass() {
                // ...
                someObject.addMouseListener(this);
        }
        public void mousePressed(MouseEvent e) {        }
        public void mouseReleased(MouseEvent e) {       }
        public void mouseEntered(MouseEvent e) {        }
        public void mouseExited(MouseEvent e) { }
        public void mouseClicked(MouseEvent e) {
                // ...
                // ...
        }
}
```

Adapter를 사용하는 경우

```
public class MyClass extends MouseAdapter {
        public MyClass() {
                // ...
                someObject.addMouseListener(this);
        }
        public void mouseClicked(MouseEvent e) {
                // ...
        }
}
```

Key 예제 KeyAdapter로 수정




```
public class KeyAdapterTest extends JFrame {
        private JPanel panel;  private JTextField field; private JTextArea area;
        public KeyAdapterTest() {
                panel = new JPanel(new GridLayout(0, 2));
                panel.add(new JLabel("문자를 입력하시오: "));
                field = new JTextField(10);
                panel.add(field);
                area = new JTextArea(3, 30);
                add(panel, BorderLayout.NORTH);
                add(area, BorderLayout.CENTER);
                field.addKeyListener(new KeyAdapter() {
                        public void keyPressed(KeyEvent e) { // keyPressed만 넣고 싶을때
                                display(e, "Key Pressed ");
                        }
                });
                setTitle("KeyAdapterTest");
                setSize(400, 200);
                setVisible(true);
        }
        public static void main(String[] args) {
                new KeyAdapterTest();
        }
}
```