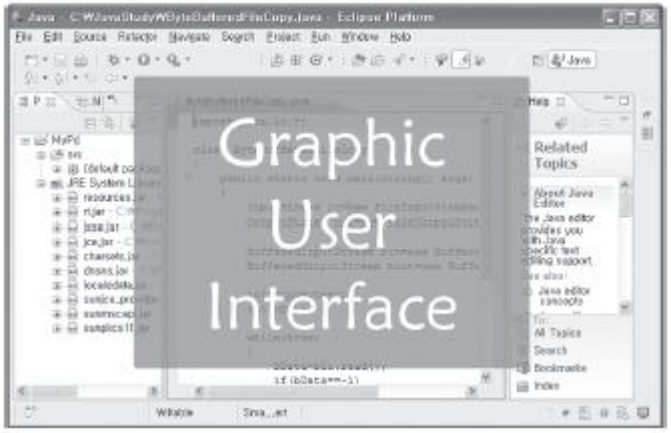
Chapter 25. Swing 컴포넌트와 이벤트 핸들링

나는 정말 JAVA를 공부한 적이 없다고

25-1. Swing을 시작하기에 앞서

■ Swing에 대한 여러가지 이야기



√ UI와 기능의 구현을 구분하는 형태로 소프트웨어의 개발방법은 발전해 나갈 것이다.

√ AWT와 Swing은 UI와 기능의 구현을 구분한 모델은 아니다. 하지만, 이에 대한 학습을 통해서 UI 구현에 대한 전반적인 이해를 갖출 수 있다.

√ Swing을 사용하지 않더라도 Swing의 학습은 다른 프레임워크의 이해에 많은 도움이 된다.

√ AWT와 Swing을 각각 별도로 공부할 필요는 없다. 둘 중 하나를 선택에서 UI 구현에 대한 경험을 갖는 것에 의미를 두자!

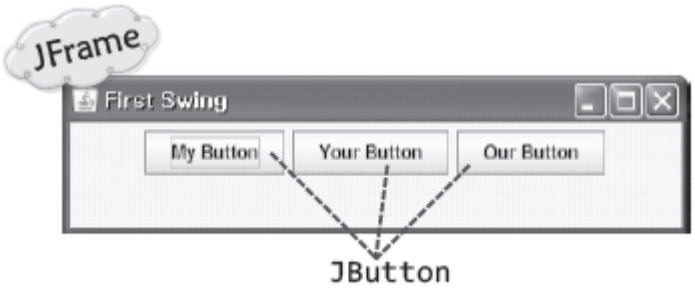
나는 정말 JAVA를 공부한 적이 없다고

25-2. Swing 컴포넌트와 이벤트 핸들링

Swing 프로그래밍에 대한 이해

나 정말 JAVA를 공부한 적 없는데

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

class FirstSwing
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("First Swing");
        frm.setBounds(120, 120, 400, 100);
        frm.setLayout(new FlowLayout());

        JButton btn1=new JButton("My Button");
        JButton btn2=new JButton("Your Button");
        JButton btn3=new JButton("Our Button");

        frm.add(btn1);
        frm.add(btn2);
        frm.add(btn3);
        frm.setVisible(true);
    }
}
```



골격이 되는 창은 **JFrame**의 인스턴스

JFrame이 상속하는 클래스중 하나는 **java.awt.Container** 클래스! 이 클래스를 상속하는 컴포넌트는 다른 Swing 컴포넌트를 얹을 수 있다.
Container 클래스는 배치와 관련된 클래스이다. 따라서 **setLayout** 메소드 역시 Container 클래스의 메소드이다.

FlowLayout 인스턴스가 장착되어 컴포넌트의 배치를 관리하게 된다. 단순히 add 메소드의 호출을 통해서 컴포넌트를 얹으면 FlowLayout 인스턴스가 적절히 배치한다.

setVisible 메소드 호출을 통해서 그간 작성한 JFrame을 눈에 보이게 한다.

Swing과 AWT의 비교

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

class FirstSwing
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("First Swing");
        frm.setBounds(120, 120, 400, 100);
        frm.setLayout(new FlowLayout());

        JButton btn1=new JButton("My Button");
        JButton btn2=new JButton("Your Button");
        JButton btn3=new JButton("Our Button");

        frm.add(btn1);
        frm.add(btn2);
        frm.add(btn3);
        frm.setVisible(true);
    }
}
```

Swing 코드

```
import java.awt.*;

class FirstAWT
{
    public static void main(String[] args)
    {
        Frame frm=new Frame("First Swing");
        frm.setBounds(120, 120, 400, 100);
        frm.setLayout(new FlowLayout());

        Button btn1=new Button("My Button");
        Button btn2=new Button("Your Button");
        Button btn3=new Button("Our Button");

        frm.add(btn1);
        frm.add(btn2);
        frm.add(btn3);
        frm.setVisible(true);
    }
}
```

AWT 코드

JFrame을 대신해서 Frame 사용! JButton을 대신해서 Button 사용! FlowLayout은 양쪽에 모두 사용! 단, Swing 컴포넌트는 운영체제에 상관없이 보이는 모습이 동일하지만 AWT 컴포넌트는 운영체제에 따라서 보이는 모습에 차이가 있다.

exit 메소드

```
class FirstAWTExitEvent
{
    public static void main(String[] args)
    {
        Frame frm=new Frame("First Swing");
        frm.setBounds(120, 120, 400, 100);
        frm.setLayout(new FlowLayout());

        WindowListener listen=new WindowAdapter()
        {
            public void windowClosing(WindowEvent ev)
            {
                System.exit(0);     // 프로그램의 종료 명령 메소드
            }
        };

        frm.addWindowListener(listen);

        Button btn1=new Button("My Button");
        Button btn2=new Button("Your Button");
        Button btn3=new Button("Our Button");

        frm.add(btn1);
        frm.add(btn2);
        frm.add(btn3);
        frm.setVisible(true);
    }
}
```

GUI 창의 우측 상단의 x 버튼을 눌러도 프로그램은 종료되지 않는다. x 버튼이 눌렸을 때 프로그램이 종료되기를 원한다면 그에 따른 이벤트 핸들링이 필요하다!

exit 메소드의 전달 값

나 정말 JAVA를 공부한 적 없는데

```
class RunningProcess
{
    public static void main(String[] args) throws IOException, InterruptedException
    {
        Process proc=Runtime.getRuntime().exec("java FirstAWTExitEvent");
        proc.waitFor();

        if(proc.exitValue()==0)
            System.out.println("잘 종료되었군!");
        else
            System.out.println("무엇인가 문제가 있어!");
    }
}
```

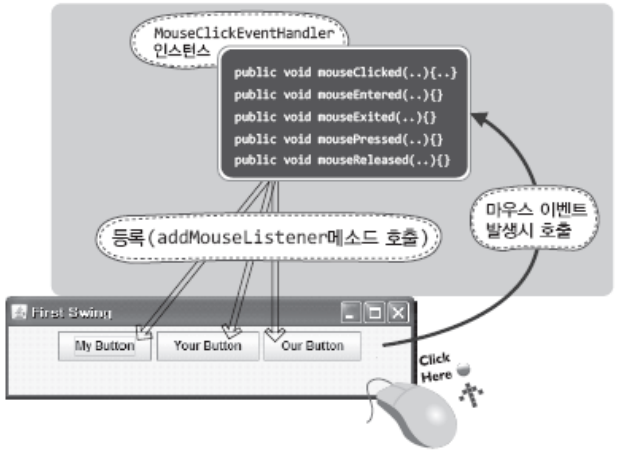
- waitFor　　프로그램이 종료되기를 기다린다.
- exitValue　　프로그램의 종료 값(exit value)를 얻는다.

exec 메소드를 통해서 다른 자바 프로그램의 실행을 명령할 수 있다. 그리고 이러한 상황에서 실행을 명령한 자바 프로그램이 종료 시 전달한 값, 예를 들어서 exit 함수 호출 시 전달한 값을 얻을 수 있다.

이벤트 리스너에 대한 간단한 소개1

```
class MouseEventHandler implements MouseListener
{
    /* 마우스 버튼이 클릭되었을 때(눌렸다 풀렸을 때) 호출됩니다. */
    public void mouseClicked(MouseEvent e)
    {
        JButton button=(JButton)e.getComponent();
        button.setText("Clicked");
        System.out.println("Clicked Button"+e.getButton());
        System.out.println("마우스 버튼 눌렸다 풀림");
    }
    /* 마우스 커서가 버튼 위에 올라가면 호출됩니다. */
    public void mouseEntered(MouseEvent e)
    {
        System.out.println("커서 버튼 위 진입");
    }
    /* 마우스 커서가 버튼을 빠져나가면 호출됩니다. */
    public void mouseExited(MouseEvent e)
    {
        System.out.println("커서 버튼 위 탈출");
    }
    /* 마우스 버튼이 눌리는 순간 호출됩니다. */
    public void mousePressed(MouseEvent e)
    {
        System.out.println("마우스 버튼 눌림");
    }
    /* 마우스 버튼이 풀리는 순간 호출됩니다. */
    public void mouseReleased(MouseEvent e)
    {
        System.out.println("마우스 버튼 풀림");
    }
}
```

```
class EventHandler
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("First Swing");
        frm.setBounds(120, 120, 400, 100);
        frm.setLayout(new FlowLayout());
        JButton btn1=new JButton("My Button");
        MouseListener listener=new MouseEventHandler();
        btn1.addMouseListener(listener);
        JButton btn2=new JButton("Your Button");
        btn2.addMouseListener(listener);
        JButton btn3=new JButton("Our Button");
        btn3.addMouseListener(listener);
        frm.add(btn1);
        frm.add(btn2);
        frm.add(btn3);
        frm.setVisible(true);
    }
}
```



이벤트 리스너에 대한 간단한 소개2

나 정말 JAVA를 공부한 적 없는데

동일 위치에서 버튼이 눌렸다 놓일 때 호출되는 메소드

```
public void mouseClicked(MouseEvent e)
{
    JButton button=(JButton)e.getComponent();
    button.setText("Clicked");
    System.out.println("clicked Button"+e.getButton());
    System.out.println("마우스 버튼 눌렸다 풀림");
}
```

MouseEvent 인스턴스를 통해서 얻을 수 있는 정보들

• 이벤트가 발생한 위치정보	getX, getY 메소드
• 이벤트가 발생한 인스턴스의 참조 값	getComponent 메소드
• 이벤트를 발생시킨 마우스 버튼의 종류	getButton 메소드

버튼이 눌린 이후
mouseClicked 메소드 호출을 통해서 발생한 상태의 변화

나는 정말 JAVA를 공부한 적이 없다고

25-3. 레이아웃 매니저

FlowLayout 배치 관리자

- 왼쪽에서 오른쪽으로 배치한다.
- 중앙으로 정렬해가며 배치한다.
- 한 줄에 모든 컴포넌트를 배치할 수 없을 때에는 다음 줄에 이어서 배치를 한다.

배치의 기준

```
class FlowLayoutManager
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("FlowLayout Test");
        frm.setBounds(120, 120, 400, 200);
        frm.setLayout(new FlowLayout());

        frm.add(new JButton("Hi!"));
        frm.add(new JButton("I like Swing"));
        frm.add(new JButton("I am a button"));

        frm.add(new LargeButton("Hi!"));
        frm.add(new LargeButton("I like Swing"));
        frm.add(new LargeButton("I am a button"));
        frm.setVisible(true);
    }
}
```

```
class LargeButton extends JButton
{
    public LargeButton(String str)
    {
        super(str);
    }
    public Dimension getPreferredSize()
    {
        Dimension largeBtmSz
            =new Dimension(
                    super.getPreferredSize().width+30,
                    super.getPreferredSize().height+15
                );
        return largeBtmSz;
    }
}
```

getPreferredSize 메소드는 JButton 클래스가 상속하는 상위 클래스에 정의된 메소드로써, GUI창에 그려질 컴포넌트의 적절한 크기정보를 반환한다. 그리고 FlowLayout 배치관리자는 이 메소드를 호출해서 반환되는 값을 참조하여 컴포넌트를 배치한다.

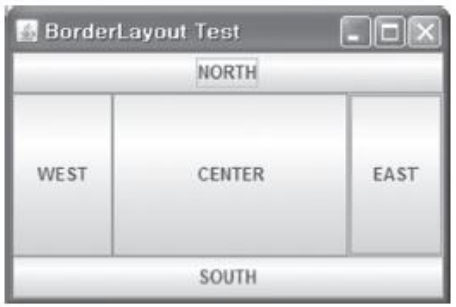
Dimension 클래스의 두 인스턴스 변수는 width와 height이며, 직접 접근이 가능하다.

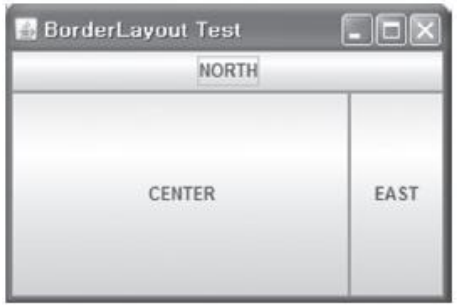
BorderLayout 배치 관리자

```
BorderLayout.NORTH      // 북(North)
BorderLayout.WEST       // 서(West)
BorderLayout.CENTER     // 중앙(Center)
BorderLayout.EAST       // 동(East)
BorderLayout.SOUTH      // 남(South)
```

배치의 기준

```
class BorderLayoutManager
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("BorderLayout Test");
        frm.setBounds(120, 120, 300, 200);
        frm.setLayout(new BorderLayout());
        frm.add(new JButton("EAST"), BorderLayout.EAST);
        frm.add(new JButton("WEST"), BorderLayout.WEST);
        frm.add(new JButton("SOUTH"), BorderLayout.SOUTH);
        frm.add(new JButton("NORTH"), BorderLayout.NORTH);
        frm.add(new JButton("CENTER"), BorderLayout.CENTER);
        frm.setVisible(true);
    }
}
```





SOUTH, WEST 채우지 않았을 때의 실행결과

중앙을 제외한 나머지 영역이 비게 되면, 그만큼 다른 영역의 컴포넌트에 의해 빈 영역이 채워짐!

GridLayout 배치 관리자

```
class GridLayoutManager
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("GridLayout Test");
        frm.setBounds(120, 120, 300, 200);
        frm.setLayout(new GridLayout(3, 2));

        frm.add(new JButton("One")); frm.add(new JButton("Two"));
        frm.add(new JButton("Three")); frm.add(new JButton("Four"));
        frm.add(new JButton("Five")); frm.add(new JButton("Six"));
        frm.setVisible(true);
    }
}
```

세로 3, 가로 2로 분할

올리는 대로 왼쪽에서 오른쪽으로,
위에서 아래로

```
public GridLayout(int rows, int cols, int hgap, int vgap)
```

위의 생성자를 통해서 세로와 가로의 분할정보 뿐만 아니라, 세로와 가로의 컴포넌트간 간격도 지정할 수 있다.

hgap과 vgap에 2를 전달했을때의
실행결과

■ 하나의 JFrame에 둘 이상의 배치 관리자 적용!

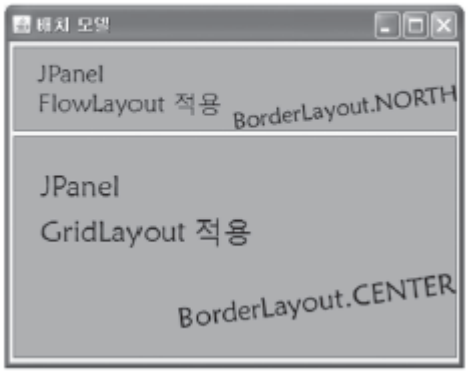
나 정말 JAVA를 공부한 적 없는데

JPanel 컴포넌트

√ Jpanel은 눈에 보이는 성격의 컴포넌트가 아니다.

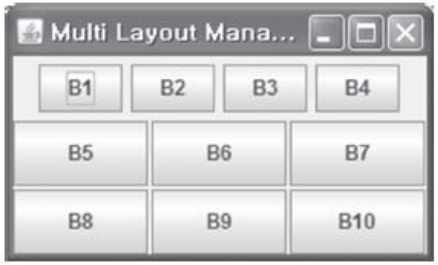
√ JFrame처럼 다른 컴포넌트를 얹을 수 있고, 또 배치 관리자의 지정도 가능하다!

JFrame에는
BorderLayout 적용



적용의 예

JFrame에 배치 관리자 지정해서 두 개의 JPanel을 올린다.
각각의 Jpanel에 각각의 배치 관리자를 별도로 지정한다.




```
class MultiLayoutManager
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("Multi Layout Manager");
        frm.setBounds(120, 120, 250, 150);
        frm.setLayout(new BorderLayout());

        JButton btm1=new JButton("B1");
        JButton btm2=new JButton("B2");
        JButton btm3=new JButton("B3");
        JButton btm4=new JButton("B4");
        JButton btm5=new JButton("B5");
        JButton btm6=new JButton("B6");
        JButton btm7=new JButton("B7");
        JButton btm8=new JButton("B8");
        JButton btm9=new JButton("B9");
        JButton btm10=new JButton("B10");

        JPanel panel1=new JPanel();
        panel1.setLayout(new FlowLayout());
        panel1.add(btm1); panel1.add(btm2);
        panel1.add(btm3); panel1.add(btm4);

        JPanel panel2=new JPanel();
        panel2.setLayout(new GridLayout(2, 3, 2, 2));
        panel2.add(btm5); panel2.add(btm6);
        panel2.add(btm7); panel2.add(btm8);
        panel2.add(btm9); panel2.add(btm10);

        frm.add(panel1, BorderLayout.NORTH);
        frm.add(panel2, BorderLayout.CENTER);

        frm.setVisible(true);
    }
}
```

나는 정말 JAVA를 공부한 적이 없다고

25-4. 이벤트와 이벤트 리스너

■ 이벤트의 종류와 그에 따른 이벤트 리스너

• MouseEvent	MouseListener	마우스 관련 이벤트
• MouseEvent	MouseMotionListener	마우스 움직임 관련 이벤트
• TextEvent	TextListener	텍스트 관련 컴포넌트의 문자 편집 이벤트
• ItemEvent	ItemListener	선택 관련 이벤트
• AdjustmentEvent	AdjustmentListener	스크롤 바 이벤트
• WindowEvent	WindowListener	GUI 프레임 창 관련 이벤트
• ActionEvent	ActionListener	컴포넌트 별 특정 행위 관련 이벤트

ActionEvent가 발생하는 상황은 컴포넌트에 따라 결정된다.
JButton의 경우 JButton이 눌렸을때 ActionEvent가 발생한다.

ActionEvent가 발생하는 상황은 컴포넌트에 따라 결정된다.
JButton의 경우 JButton이 눌렸을때 ActionEvent가 발생한다.

setEnable(false)에 의해서 컴포넌트가 비활성화되면, 눌리지 않기 때문에 ActionEvent가 발생하지 않는다. 그러나 MouseEvent는 발생한다.

```
class MouseEventHandler implements MouseListener
{
    public void mouseClicked(MouseEvent e)
    {
        System.out.println("마우스 버튼 눌렸다 풀림");
    }
    public void mouseEntered(MouseEvent e) { }
    public void mouseExited(MouseEvent e) { }
    public void mousePressed(MouseEvent e) { }
    public void mouseReleased(MouseEvent e) { }
}

class JButtonMouseEvent
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("JButton Disable");
        frm.setBounds(120, 120, 250, 150);
        frm.setLayout(new FlowLayout());
        MouseListener mouseHandler=new MouseEventHandler();
        JButton btn1=new JButton("Button One");
        btn1.addMouseListener(mouseHandler);
        JButton btn2=new JButton("Button Two");
        btn2.addMouseListener(mouseHandler);
        frm.add(btn1);
        frm.add(btn2);
        btn1.setEnabled(false);
        frm.setVisible(true);
    }
}
```

ActionEvent

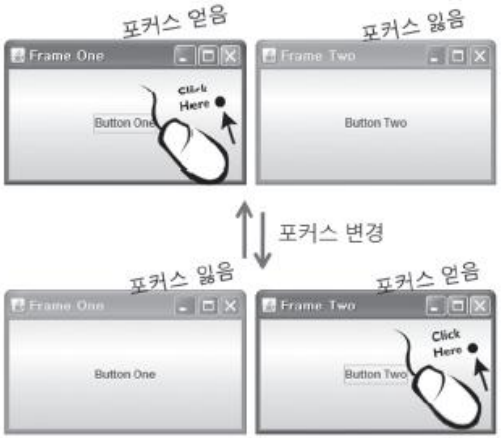
```
class ActionEventHandler implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        System.out.println(e.getActionCommand());
    }
}
class JButtonActionEvent
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("JButton Disable");
        frm.setBounds(120, 120, 250, 150);
        frm.setLayout(new FlowLayout());

        JButton btn1=new JButton("Button One");
        JButton btn2=new JButton("Button Two");

        ActionListener actionHandler=new ActionEventHandler();
        btn1.addActionListener(actionHandler);
        btn2.addActionListener(actionHandler);

        frm.add(btn1);
        frm.add(btn2);
        btn1.setEnabled(false);

        frm.setVisible(true);
    }
}
```

버튼의 눌림에 대해서는 ActionEvent를 기반으로 이벤트를 처리하는 것이 타당하다.

WindowEvent 1

```
class JFrameWindowEvent
{
    public static void main(String[] args)
    {
        JFrame frmOne=new JFrame("Frame One");
        JFrame frmTwo=new JFrame("Frame Two");

        frmOne.setBounds(120, 120, 250, 150);
        frmTwo.setBounds(380, 120, 250, 150);

        frmOne.addWindowListener(new WindowEventHandler("Frame One"));
        frmTwo.addWindowListener(new WindowEventHandler("Frame Two"));

        frmOne.add(new JButton("Button One"));
        frmTwo.add(new JButton("Button Two"));
        frmOne.setVisible(true);
        frmTwo.setVisible(true);
    }
}
```

```
class WindowEventHandler implements WindowListener
{
    String frameInfo;

    public WindowEventHandler(String info)
    {. . .
    public void windowActivated(WindowEvent e)
    {. . .
    public void windowClosed(WindowEvent e)
    {. . .
    public void windowClosing(WindowEvent e)
    {. . .
    public void windowDeactivated(WindowEvent e)
    {. . .
    public void windowDeiconified(WindowEvent e)
    {. . .
    public void windowIconified(WindowEvent e)
    {. . .
    public void windowOpened(WindowEvent e)
    {
        System.out.println(frameInfo+" windowOpened");
    }
}
```



- public void windowOpened(WindowEvent e)

GUI 창이 등장하면서 호출

- public void windowActivated(WindowEvent e)
- public void windowDeactivated(WindowEvent e)

GUI 창의 활성화! 비활성화!

WindowEvent 2

나는 정말 JAVA를 공부한 적이 없다고

- public void windowIconified(WindowEvent e)

  우측 상단의 최소화 버튼이 눌릴 때

- public void windowDeiconified(WindowEvent e)

  다시 복원 되었을 때

- public void windowClosing(WindowEvent e)

  우 상단의 X 버튼이 눌리면 호출

- public void windowClosed(WindowEvent e)

  창이 소멸되면서 호출

X 버튼이 눌리면 호출

```
public void windowClosing(WindowEvent e)
{
    JFrame frm=(JFrame)e.getWindow();
    frm.dispose();   windowClosed 함수의 호출로 이어짐
    System.out.println(frameInfo+" windowClosing");
}
```

X 버튼이 눌려서 창이 보이지 않는다고 해서 창이 소멸된 것은 아니다!

프로그램을 종료시키는 또 다른 방법

나 정말 JAVA를 공부한 적 없는데

```
class SetDefaultCloseOperation
{
    public static void main(String[] args) throws Exception
    {
        JFrame frmOne=new JFrame("Frame One");
        JFrame frmTwo=new JFrame("Frame Two");

        frmOne.setBounds(120, 120, 250, 150);
        frmTwo.setBounds(380, 120, 250, 150);

        frmOne.add(new JButton("Button One"));
        frmTwo.add(new JButton("Button Two"));

        frmOne.setDefaultCloseOperation(
                WindowConstants.DISPOSE_ON_CLOSE);
        frmTwo.setDefaultCloseOperation(
                WindowConstants.DISPOSE_ON_CLOSE);

        frmOne.setVisible(true);
        frmTwo.setVisible(true);
    }
}
```

setDefaultCloseOperation의 인자로 WindowConstant.DISPOSE_ON_CLOSE가 전달되면 X 버튼 눌릴 때 dispose 메소드 호출과 동일한 결과를 보인다.

MouseListener & MouseMotionListener

나는 정말 JAVA를 공부한 적이 없다네

MouseEvent Listener

- √ MouseListener — 마우스 관련 이벤트
- √ MouseMotionListener — 마우스 움직임 관련 이벤트

```
class MouseMotionHandler implements MouseMotionListener
{
   public void mouseDragged(MouseEvent e)
   {
       System.out.printf("Drag [%d %d] \n", e.getX(), e.getY());
   }
   public void mouseMoved(MouseEvent e)
   {
       System.out.printf("Move [%d %d] \n", e.getX(), e.getY());
   }
}
class MouseMotionEvent
{
    public static void main(String[] args) throws Exception
    {
        JFrame frm=new JFrame("Mouse Motion");
        frm.setBounds(120, 120, 250, 150);
        frm.addMouseMotionListener(new MouseMotionHandler());

        frm.setVisible(true);
        frm.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    }
}
```

마우스 버튼이 눌린 상태에서의 움직임

Adapter 클래스

```
class MouseEventHandler implements MouseListener
{
    public void mouseClicked(MouseEvent e)
    {
        System.out.println("마우스 버튼 눌렸다 풀림");
        . . . .
    }
    public void mouseEntered(MouseEvent e) { }
    public void mouseExited(MouseEvent e) { }
    public void mousePressed(MouseEvent e) { }
    public void mouseReleased(MouseEvent e) { }
}
```

• MouseListener	MouseAdapter
• MouseMotionListener	MouseMotionAdapter
• TextListener	어댑터 클래스 없음
• ItemListener	어댑터 클래스 없음
• AdjustmentListener	어댑터 클래스 없음
• WindowListener	WindowAdapter
• ActionListener	어댑터 클래스 없음

Adapter 클래스 기반의 구현

```
class MouseEventHandler extends MouseAdapter
{
    public void mouseClicked(MouseEvent e)
    {
        System.out.println("마우스 버튼 눌렸다 풀림");
        . . . .
    }
}
```

인터페이스의 모든 메소드를 빈 상태로 구현해 놓은 것이 Adapter 클래스이다.

Anonymous(익명) 클래스의 활용

나는 정말 JAVA를 공부한 적이 없다

```
class AdapterAnonymousHandling
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("Mouse Motion");
        frm.setBounds(120, 120, 250, 150);
        frm.addMouseListener(
            new MouseAdapter()
            {
                public void mouseClicked(MouseEvent e)
                {
                    System.out.println("마우스 버튼 눌림");
                }
            }
        );

        frm.setVisible(true);
        frm.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    }
}
```

Adapter 클래스의 메소드를 하나만 정의하고자 하는 경우에는 Anonymous 클래스의 형태로 정의하는 것도 의미가 있다.

나는 정말 JAVA를 공부한 적이 없다

25-5. 다양한 Swing 컴포넌트

JLabel & JTextField

```
class PWHandler implements ActionListener
{
    JTextField id;
    JPasswordField pw;
    public PWHandler(JTextField id, JPasswordField pw)
    {
        this.id=id;
        this.pw=pw;
    }
    public void actionPerformed(ActionEvent e)
    {
        System.out.println("ID : "+id.getText());
        System.out.println("Password : "+new String(pw.getPassword()));
        id.setText("");
        pw.setText("");
    }
}
```

JTextField와 JPasswordField의 텍스트상에서 엔터 키가 입력되면 ActionEvent가 발생

JLabel은 문자열 출력을 위한 컴포넌트

JTextField는 문자열 출력을 위한 컴포넌트

JPasswordField는 문자열을 가리며 출력

```
class JLabelAndJTextField
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("JLabel & JTextField");
        frm.setBounds(120, 120, 180, 80);
        frm.setLayout(new GridLayout(2, 2));
        JLabel idLabel=new JLabel("ID ", SwingConstants.RIGHT);
        JTextField idText=new JTextField(10);
        JLabel pwLabel=new JLabel("Password ", SwingConstants.RIGHT);
        JPasswordField pwText=new JPasswordField(10);
        pwText.setEchoChar('*');
        pwText.addActionListener(new PWHandler(idText, pwText));
        frm.add(idLabel); frm.add(idText);
        frm.add(pwLabel); frm.add(pwText);
        frm.setVisible(true);
        frm.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    }
}
```

JTextArea

나는 정말 JAVA를 공부한 적이 없다

```
class ButtonTextHandler implements ActionListener
{
    JTextArea textArea;
    public ButtonTextHandler(JTextArea area)
    {
        textArea=area;
    }
    public void actionPerformed(ActionEvent e)
    {
        textArea.setText("모두 지웠습니다. \n");
        textArea.append("원하는 내용 입력하세요. \n");
    }
}
```

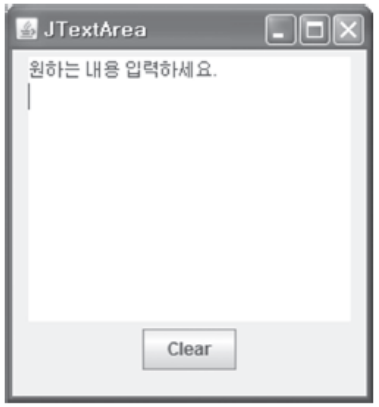
JTextArea는 여러 줄의 문자열 입력을 위한 컴포넌트

```
class JTextAreaSimpleModel
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("JTextArea");
        frm.setBounds(120, 120, 250, 270);
        frm.setLayout(new FlowLayout());
        JTextArea textArea=new JTextArea(10, 20);
        textArea.append("원하는 내용 입력하세요. \n");
        textArea.setCaretPosition(textArea.getText().length());
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        JButton btn=new JButton("Clear");
        btn.addActionListener(new ButtonTextHandler(textArea));
        frm.add(textArea); frm.add(btn);
        frm.setVisible(true);
        frm.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    }
}
```

세로 10, 가로 20의 JTextArea 생성

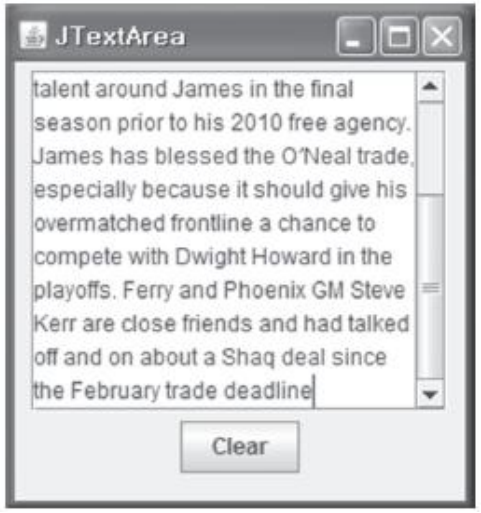
커서 다음 행으로 이동

자동 줄 바꿈
단어 단위 줄 바꿈



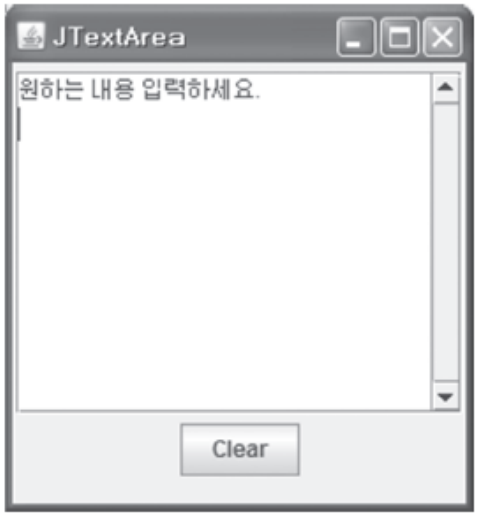
JScrollPane

```
JScrollPane simpleScroll=new JScrollPane(textArea);
frm.add(simpleScroll); frm.add(btn);

frm.setVisible(true);
frm.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
```



스크롤 바가 필요한 시점에 생성됨

```
JScrollPane simpleScroll=
    new JScrollPane(textArea,
        ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,     // 세로 스크롤 정책
        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER     // 가로 스크롤 정책
    );
```



VERTICAL_SCROLLBAR_ALWAYS은 계속해서 스크롤 바를 표시함을 의미

HORIZONTAL_SCROLLBAR_NEVER는 스크롤 바를 표시하지 말라는 의미

JCheckBox & JRadioButton


```
class CheckBoxHandler implements ItemListener
{
    JRadioButton btn1;
    JRadioButton btn2;
    JRadioButton btn3;
    public CheckBoxHandler(JRadioButton b1, JRadioButton b2, JRadioButton b3)
    {
        btn1=b1;
        btn2=b2;
        btn3=b3;
    }
    public void itemStateChanged(ItemEvent e)
    {
        if(e.getStateChange()==ItemEvent.SELECTED)
        {
            btn1.setEnabled(true);
            btn2.setEnabled(true);
            btn3.setEnabled(true);
        }
        else    /* ItemEvent.DESELECTED */
        {
            btn1.setEnabled(false);
            btn2.setEnabled(false);
            btn3.setEnabled(false);
        }
    }
}
```

```
class JCheckBoxAndJRadioButton
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("Choice Component");
        frm.setBounds(120, 120, 200, 200);
        frm.setLayout(new GridLayout(0, 1));   // 가로는 1, 세로는 자유롭게
        JCheckBox checkBox=new JCheckBox("Are you a programmer");
        JRadioButton rbtn1= new JRadioButton("I like C");
        JRadioButton rbtn2= new JRadioButton("I like C++");
        JRadioButton rbtn3= new JRadioButton("I like Java", true);
        ButtonGroup bGroup=new ButtonGroup();
        bGroup.add(rbtn1); bGroup.add(rbtn2); bGroup.add(rbtn3);
        checkBox.addItemListener(new CheckBoxHandler(rbtn1, rbtn2, rbtn3));
        frm.add(checkBox); frm.add(rbtn1); frm.add(rbtn2); frm.add(rbtn3);
        rbtn1.setEnabled(false);
        rbtn2.setEnabled(false);
        rbtn3.setEnabled(false);
        . . . .
        frm.setVisible(true);
        frm.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    }
}
```

JRadioButton 이벤트 핸들링

```
rbtn1.addItemListener(
    new ItemListener() {
        public void itemStateChanged(ItemEvent e)
        {
            if(e.getStateChange()==ItemEvent.SELECTED)
                System.out.println("I like C too");
        }
    }
);
rbtn2.addItemListener(
    new ItemListener() {
        public void itemStateChanged(ItemEvent e)
        {
            if(e.getStateChange()==ItemEvent.SELECTED)
                System.out.println("I like C++ too");
        }
    }
);
rbtn3.addItemListener(
    new ItemListener() {
        public void itemStateChanged(ItemEvent e)
        {
            if(e.getStateChange()==ItemEvent.SELECTED)
                System.out.println("I like Java too");
        }
    }
);
```

Border

```
JRadioButton rbtn1= new JRadioButton("I like C");
JRadioButton rbtn2= new JRadioButton("I like C++");
JRadioButton rbtn3= new JRadioButton("I like Java", true);
ButtonGroup bGroup=new ButtonGroup();
bGroup.add(rbtn1); bGroup.add(rbtn2); bGroup.add(rbtn3);

Border rbtnBorder=BorderFactory.createEtchedBorder();
rbtnBorder=BorderFactory.createTitledBorder(rbtnBorder, "Language");

JPanel rbtnBorderPanel=new JPanel();
rbtnBorderPanel.setLayout(new GridLayout(0, 1));
rbtnBorderPanel.setBorder(rbtnBorder);

rbtnBorderPanel.add(rbtn1);
rbtnBorderPanel.add(rbtn2);
rbtnBorderPanel.add(rbtn3);
```

JComboBox1

나는 정말 JAVA를 공부한 적이 없다네

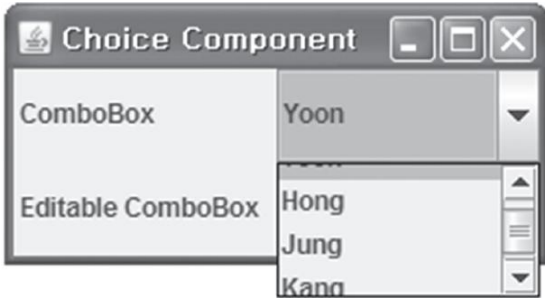
```
class MyFriend
{
    String name;
    int age;
    public MyFriend(String name, int age)
    {
        this.name=name;
        this.age=age;
    }
    public String toString() { return name; }
    public void showFriendInfo()
    {
        System.out.println("name : "+ name);
        System.out.println("age : "+ age);
    }
}
```

```
class ChoiceHandler implements ItemListener
{
    public void itemStateChanged(ItemEvent e)
    {
        if(e.getStateChange()==ItemEvent.SELECTED)
        {
            System.out.println("Selected... ");
            ((MyFriend)e.getItem()).showFriendInfo();
        }
        else
        {
            System.out.println("Deselected... ");
            ((MyFriend)e.getItem()).showFriendInfo();
        }
    }
}
class TextChangedHandler implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if((e.getActionCommand()).compareTo("comboBoxEdited")==0)
            System.out.println("ComboBox Edited");
        else /*comboBoxChanged */
            System.out.println("ComboBox Changed");
    }
}
```

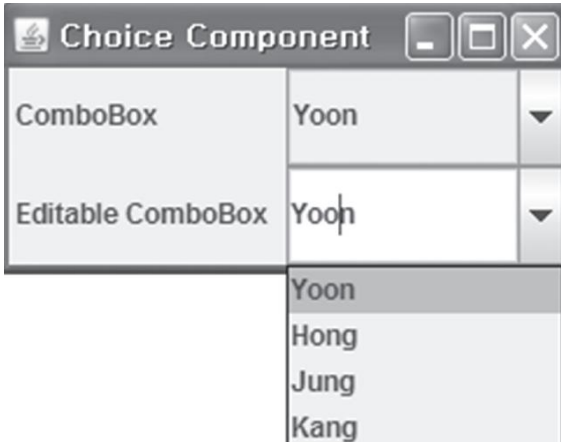
ItemEvent는 선택 받은 컴포넌트를 대상으로도 선택을 잃은 컴포넌트를 대상으로도 발생한다.

JComboBox2

```
class JComboBoxModel
{
    public static void main(String[] args)
    {
        JFrame frm=new JFrame("Choice Component");
        frm.setBounds(120, 120, 250, 120);
        frm.setLayout(new GridLayout(0, 2));

        Vector<MyFriend> friend=new Vector<MyFriend>();
        friend.add(new MyFriend("Yoon", 22));
        friend.add(new MyFriend("Hong", 23));
        friend.add(new MyFriend("Jung", 24));
        friend.add(new MyFriend("Kang", 25));

        JLabel label1=new JLabel(" ComboBox");
        JComboBox cmbBox1=new JComboBox(friend);
        cmbBox1.setMaximumRowCount(3);
        cmbBox1.addItemListener(new ChoiceHandler());

        JLabel label2=new JLabel(" Editable ComboBox");
        JComboBox cmbBox2=new JComboBox(friend);
        cmbBox2.setEditable(true);
        cmbBox2.addActionListener(new TextChangedHandler());

        frm.add(label1); frm.add(cmbBox1);
        frm.add(label2); frm.add(cmbBox2);

        frm.setVisible(true);
        frm.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    }
}
```

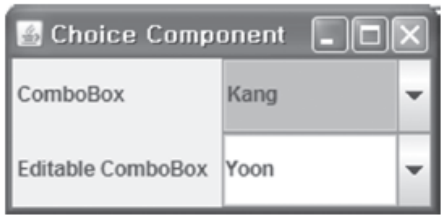


setMaximumRowCount 호출결과



setMaximumRowCount 호출하지 않은 결과

JComboBox3



Yoon에서 Kang을 선택 두 번의 ItemEvent 발생!

```
Deselected...
name : Yoon
age : 22
Selected...
name : Kang
age : 25
```

Yoon을 Jung으로 변경 ActionEvent 발생

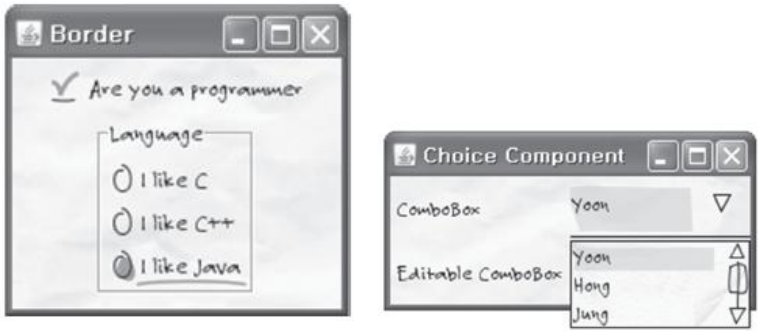
```
ComboBox Changed
```

입력 후 엔터! 두 번의 ActionEvent 발생

```
ComboBox Changed
ComboBox Edited
```

Look And Feel

```
try
{
    UIManager.setLookAndFeel(
        "com.sun.java.swing.plaf.motif.MotifLookAndFeel"
    );
}
catch(Exception e)
{
    e.printStackTrace();
}
```




```
try
{
    UIManager.setLookAndFeel(
        "net.sourceforge.napkinlaf.NapkinLookAndFeel"
    );
}
catch(Exception e)
{
    e.printStackTrace();
}
```

Swing에서는 Look And Feel의 변경을 통해서 모든 컴포넌트의 View를 일괄 변경할 수 있다.

나는 정말 JAVA를 공부한 적이 없다