

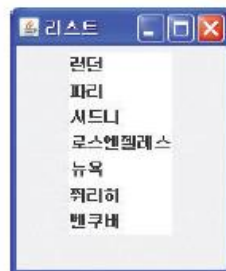
24. 스윙 컴포넌트 II

학습목표

- 이런 장에서 스윙이 제공하는 보다 강력한 컴포넌트에 대하여 살펴보자. 이번 장의 목표는 메뉴를 가지고 있고, 사용자로부터 텍스트를 입력받을 수 있으며 콤보 박스 등을 사용자가 주어진 목록 중에서 하나를 선택할 수 있는 애플리케이션 작성이다.

2. 리스트(List)

리스트(List)는 여러 개의 선택 항목 중에서 하나를 선택하기 위한 컴포넌트이다. 리스트는 한 줄에 하나씩 선택 항목을 나타내며 이 영역은 스크롤이 가능하다. 일반적으로 사용자는 마우스 클릭에 의하여 항목을 선택하며, 더블 클릭이나 엔터 키를 치면 액션 이벤트가 발생한다.



- 생성자

생성자	설명
<code>JList()</code>	비어 있는 리스트를 생성한다.
<code>JList(Object[] items)</code>	배열에 있는 값들을 가지고 선택 항목을 생성한다.
<code>JList(ListModel list)</code>	지정된 리스트 모델을 사용하는 리스트를 생성한다.
<code>JList(Vector[] items)</code>	벡터에 있는 값들을 가지고 선택 항목을 생성한다.




리스트를 생성하는 가장 일반적인 방법은 배열을 `JList`의 생성자에 전달하는 것이다. `JList`의 생성자를 이용하여서 리스트를 생성하여 보면 다음과 같다.

```
private String[] names = { "하나", "둘", "셋", "넷", "다섯", "여섯" };  
list = new JList(names);
```

위의 코드는 생성자에 문자열의 배열을 전달한다. 벡터나 `Object` 타입의 배열로부터도 초기화가 가능하다. 디폴트 리스트 모델은 변경이 불가능하다. 즉 리스트에 항목을 추가하거나 삭제할 수 없다. 만약 항목들이 개별적으로 변경될 수 있는 리스트를 생성하기 위해서는 리스트 모델을 `DefaultListModel`의 인스턴스로 변경하여야 한다. `setModel()`을 이용하여서 모델을 변경할 수 있다.

1) 선택 모드

리스트에서 항목을 선택할 수 있는 모드에는 다음의 3가지가 있다.

모드	그림	설명
단일 선택 (SINGLE_SELECTION)		한 번에 하나의 항목만이 선택된다. 사용자가 새로운 항목을 클릭하면 이전의 선택은 지워진다.
단일 구간 선택 (SINGLE_INTERVAL_SELECTION)		여러 개의 연속적인 항목들이 선택될 수 있다. 일정 범위의 항목을 선택하려면 첫 번째 항목을 클릭하고 쉬프트 키를 누른 채로 마지막 항목을 클릭한다.
다중 구간 선택 (MULTIPLE_INTERVAL_SELECTION)		디폴트로 항목들이 자유롭게 선택될 수 있다. 컨트롤 키를 누른 채로 마우스로 클릭을 하면 여러 개를 선택할 수 있다.

디폴트는 다중 구간 선택이다. 만약 선택 모드를 변경하고 싶으면 `setSelectionMode()` 메소드를 사용한다. 매개 변수로는 선택 모드를 나타내는 다음과 같은 값 중에서 하나를 주어야 한다.

`ListSelectionModel.SINGLE_SELECTION`
`ListSelectionModel.SINGLE_INTERVAL_SELECTION`
`ListSelectionModel.MULTIPLE_INTERVAL_SELECTION`

예를 들어서 단일 선택 모드로 변경하려면 다음과 같은 문장을 사용한다.

`list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);`

2) 이벤트 처리

리스트의 항목이 선택되면 리스트 선택 이벤트(ListSelectionEvent)를 발생한다. 이 이벤트는 리스트 선택 리스너를 가지고 처리할 수 있다. 리스트 선택 리스너에는 하나의 메소드만 구현되어 있으면 되는데 바로 `valueChanged()` 메소드이다.

```
private class ListListener implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent e) {
        if (list.getSelectedIndex() == -1) {
            //선택이 되지 않았음
        } else {
            //선택이 되었음
        }
    }
}
```

항목 인덱스의 값을 얻기 위해서는 `getSelectedIndex()`를, 항목 이름을 얻기 위해서는 `getSelectedItem()` 메소드를 사용한다. `getSelectedValue()`는 현재 선택된 항목을 반환한다. 예를 들어서 리스트가 문자열로 구성되어 있다면 다음과 같은 문장은 문자열 객체를 반환한다.

```
String name;  
name = (String) list.getSelectedValue();
```

여기서 `getSelectedValue()`가 실제로 반환하는 것은 `Object` 참조이다. 따라서 `String` 타입으로 형변환을 하여서 `name`에 저장하여야 한다.

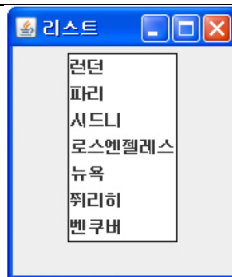
`getSelectedIndex()`는 선택된 항목의 인덱스를 반환한다. 만약 선택되지 않았다면 `-1`을 반환한다. 내부적으로 항목들은 숫자가 매겨져서 저장된다. 첫 번째 항목의 인덱스는 `0`이다. 이 인덱스를 이용하여서 항목들을 추출할 수 있다.

```
int index = list.getSelectedIndex();  
if( index != -1)  
String name = cities[index];
```

3) 리스트에 경계선 만들기

다른 컴포넌트들과 마찬가지로 리스트 주위에 경계선을 만들려면 `setBorder()` 메소드를 사용한다.

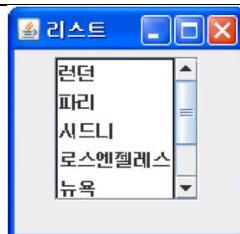
```
list.setBorder(BorderFactory.createLineBorder(Color.black, 1));
```



4) 리스트에 스크롤바 추가하기

리스트에 많은 항목들이 있는 경우에는 스크롤바를 설치하는 것이 보기 좋다. 리스트 컴포넌트는 자동적으로 스크롤바를 표시하지 않는다. 앞서서와 마찬가지로 스크롤 페인을 생성하고 여기에 리스트를 추가하면 된다. 다만 이 과정에서 화면에 표시되는 행수를 설정하는 것이 좋다.

```
list.setVisibleRowCount(10); // 화면에 표시되는 행수 설정  
JScrollPane scrollPane = new JScrollPane(list); // 스크롤 페인에 리스트 추가  
panel.add(scrollPane); // 스크롤 페인을 패널에 추가
```



5) 리스트에 항목 동적 추가/제거

리스트에 동적으로 항목을 추가하거나 제거하려면 리스트를 생성할 때 `ListModel`의 인스턴스를 생성하여 생성자의 매개 변수로 전달하여야 한다.

```
listModel = new DefaultListModel();
listModel.addElement("런던");
listModel.addElement("파리");
...
list = new JList(listModel);
```

동적으로 항목을 추가하거나 제거하려면 다음과 같은 코드를 사용한다.

```
listModel.remove(index);
...
listModel.insertElementAt("뉴욕", index);
```

(예제)

다음의 프로그램에서는 리스트, 레이블, 버튼을 생성하여 리스트에서 선택한 항목의 인덱스와 항목 이름이 레이블에 표시되도록 한다.

```
ListTest.java

import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import java.awt.event.*;
import java.awt.*;

public class ListTest extends JFrame {
    private JLabel label;
    private JTextField selected;
    private JPanel panel, listPanel;
    private JList list;
    private String[] cities = { "런던", "파리", "시드니", "로스엔젤레스", "뉴욕", "쥘리히", "벤쿠버" };

    public ListTest() {
        setTitle("리스트");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        listPanel = new JPanel();
        panel = new JPanel();
        list = new JList(cities);
        // 리스트에 경계선을 설정한다.
        list.setBorder(BorderFactory.createLineBorder(Color.black, 1));
        // 리스트에 스크롤바를 추가한다.
        JScrollPane scroller = new JScrollPane(list);
```

```

        scroller.setPreferredSize(new Dimension(200, 100));
        // 리스트의 선택 모드를 단일 선택 모드로 변경
        list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        list.addListSelectionListener(new ListListener());
        label = new JLabel("선택된 항목: ");
        selected = new JTextField(10);
        // 텍스트 필드를 편집 불가로 설정한다.
        selected.setEditable(false);

        listPanel.add(scroller);
        panel.add(label);
        panel.add(selected);
        add(listPanel, BorderLayout.NORTH);
        add(panel, BorderLayout.CENTER);
        setVisible(true);
    }

    private class ListListener implements ListSelectionListener {
        public void valueChanged(ListSelectionEvent e) {
            // 선택된 도시를 얻는다.
            String selection = (String) list.getSelectedValue();
            // 선택된 도시를 텍스트 필드에 기록한다.
            selected.setText(selection);
        }
    }

    public static void main(String[] args) {
        ListTest frame = new ListTest();
    }
}

```

3. 콤보박스

콤보 박스(combo box)도 여러 항목 중에서 하나를 선택하는데 사용할 수 있다. 콤보 박스는 텍스트 필드와 리스트의 결합이다. 사용자는 콤보 박스의 텍스트를 직접 입력할 수도 있고 리스트에서 선택할 수도 있다. 하지만 텍스트 필드를 편집 불가로 해놓은 경우에는 사용자는 리스트에서 선택만 할 수 있다.



1) 생성자

콤보 박스를 생성하기 위해서는 먼저 생성자 중에서 하나를 골라서 호출하여야 한다. 첫 번째 생성자는 비어 있는 콤보 박스를 생성한다.

```
JComboBox combo = new JComboBox();
```

여기에 항목을 추가하려면 addItem() 메소드를 사용한다.

```
combo.addItem("dog");  
combo.addItem("lion");  
combo.addItem("tiger");
```

다른 방법으로는 문자열 배열을 만든 후에 이 문자열 배열을 생성자의 매개 변수로 전달하는 방법도 있다.

```
String[] names = { "dog", "lion", "tiger"};  
JComboBox combo = new JComboBox(names);
```

벡터 객체를 사용할 수도 있고 컬렉션인 경우에는 toArray() 메소드를 호출하여서 컬렉션을 배열로 전환하여 매개 변수로 전달할 수도 있다.

```
JComboBox combo = new JComboBox(collection.toArray());
```

여기서 collection은 ArrayList와 같은 컬렉션을 참조하는 변수이다.

2) 메소드

생성자 또는 메소드	설명
<code>void addItem(Object)</code> <code>void insertItemAt(Object, int)</code>	콤보 박스에 지정된 객체를 지정된 위치에 삽입한다. 만약 위치가 지정되지 않으면 현재 위치의 앞에 삽입한다.
<code>Object getItemAt(int)</code> <code>Object getSelectedItem()</code>	콤보 박스의 메뉴에서 항목을 가져온다.
<code>void removeAllItems()</code> <code>void removeItemAt(int)</code> <code>void removeItem(Object)</code>	지정된 항목을 삭제한다.
<code>int getItemCount()</code>	항목의 개수를 반환한다.
<code>void setEditable(boolean)</code> <code>boolean isEditable()</code>	콤보 박스에 사용자가 입력할 수 있는지 여부를 설정한다.

위의 표는 콤보 박스에서 지원되는 메소드 중에서 많이 사용되는 메소드를 나타낸다. 콤보박스는 사용자가 편집할 수 없도록 설정되어 있다. 만약 사용자가 편집하기를 원한다면 setEditable(true)를 호출하면 된다. 콤보 박스에서 항목들을 제거하고 싶으면 remove() 계열 메소드 중의 하나를 사용하면 된다. 만약 제거하고 싶은 항목의 번호를 아는 경우에는 removeItemAt()를 사용하고, 객체를 아는 경우에는 removeItem()을 사용한다. 콤보 박스로부터 사용자가 선택한 항목을 가져오려면 getSelectedItem()을 사용한다. 이 메소드는 Object 타입으로 반환하므로 이것을 형변환하여서 사용하여야 한다.

3) 이벤트 처리

이벤트 리스너	설명
<code>void addActionListener(ActionListener)</code>	사용자가 콤보 박스에서 항목을 선택했을 경우, 또는 사용자가 엔터키를 눌렀을 때 액션 이벤트가 발생한다.
<code>void addItemListener(ItemListener)</code>	아이템 리스너를 추가한다. 리스너의 <code>itemStateChanged()</code> 메소드는 콤보 박스의 항목의 선택 상태가 변경되면 호출된다.

사용자가 콤보 박스에서 항목을 선택하면 이벤트가 발생된다. 보통은 이 이벤트는 무시하고 특정 버튼을 누르는 경우에만 선택된 항목을 가져오게 된다. 하지만 사용자가 항목을 선택하였을 경우, 피드백을 주고 싶은 경우에는 콤보 박스의 액션 이벤트를 처리할 수 있다.

```
private class ComboListener implements ActionListener{
    public void actionPerformed(ActionEvent e){
        if (e.getSource() == combo)
        {
            String s = (String)combo.getSelectedItem();
            if (s.equals("dog"))
                System.out.println("강아지가 선택되었습니다");
        }
    }
}
```

(예제)

```
ComboBoxTest.java

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

public class ComboBoxTest extends JFrame implements ActionListener {
    JLabel label;
    public ComboBoxTest() {
        setTitle("콤보 박스");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 200);
        String[] animals = { "dog", "lion", "tiger" };
        JComboBox animalList = new JComboBox(animals);
        animalList.setSelectedIndex(0); // 첫번째 항목 선택
        animalList.addActionListener(this);
        label = new JLabel();
        label.setHorizontalAlignment(JLabel.CENTER);
        changePicture(animals[animalList.getSelectedIndex()]);
    }
}
```

콤보박스생성

```

        add(animalList, BorderLayout.PAGE_START);
        add(label, BorderLayout.PAGE_END);
        setVisible(true);
    }
    // 콤보 박스의 항목이 선택되면 호출
    public void actionPerformed(ActionEvent e) {
        JComboBox cb = (JComboBox) e.getSource();
        String name = (String) cb.getSelectedItem();
        changePicture(name);
    }
    // 선택된 항목에 따라 적절한 이미지를 선택한다.
    protected void changePicture(String name) {
        ImageIcon icon = new ImageIcon(name + ".gif");
        label.setIcon(icon);
        if (icon != null) {
            label.setText(null);
        } else {
            label.setText("이미지가 발견되지 않았습니다.");
        }
    }
    public static void main(String[] args) {
        ComboBoxTest frame = new ComboBoxTest();
    }
}

```

4. 스피너

스피너(spinner)는 두 개의 화살표가 붙은 텍스트 필드이다. 일반적으로 텍스트 필드는 숫자를 포함하고 있고 사용자는 화살표를 클릭하여서 값들을 증가시키거나 감소시킬 수 있다. 그러나 다른 용도로도 얼마든지 스피너를 활용할 수 있다.



1) 생성자

스피너를 사용하기 위해서는 JSpinner 클래스의 객체를 생성하면 된다.

생성자	설명
JSpinner() JSpinner(SpinnerModel)	새로운 JSpinner 객체를 생성한다. 매개변수가 없으면 정수를 가지고 있는 SpinnerNumberModel이 가정되고 초기값은 0이다.
void setValue(java.lang.Object) Object getValue()	값을 설정하거나 읽는다.
Object getNextValue() Object getPreviousValue()	다음 값이나 이전 값을 반환한다.

생성자를 사용하여 컴포넌트 객체를 생성하기 전에 스피너 모델을 결정하여야 한다. 스윙API는 3가지의

스피너 모델을 제공한다. 이 스피너 모델을 생성자의 매개 변수로 전달한다. 만약 스피너 모델없이 생성자를 호출하면 SpinnerNumberModel이 지정된다.

① SpinnerListModel

스피너의 값들이 객체들의 배열이나 List 객체가 된다.

```
String[] items = { "소설", "잡지", "전공서적", "취미" };  
SpinnerListModel listModel = new SpinnerListModel(items);  
JSpinner spinner = new JSpinner(listModel);
```

② SpinnerNumberModel

스피너의 값들이 정수 혹은 실수 타입의 연속적인 숫자가 된다. 최대값과 최소값을 지정할 수 있으며 단계의 크기도 지정이 가능하다. 예를 들면 다음과 같이 생성할 수 있다.

```
// 초기값, 최대값, 최소값, 단계  
SpinnerModel numberModel = new SpinnerNumberModel(0, -10, +10, 1);  
JSpinner spinner = new JSpinner(numberModel);
```

③ SpinnerDateModel

스피너의 값들이 모두 Date 타입의 객체이다. 최소날짜와 최대날짜를 지정할 수 있으며 증가나 감소되는 필드(예를 들어서 Calendar.YEAR)를 지정할 수 있다.

```
// value, start, end 등은 모두 Date의 객체이다.  
SpinnerDateModel dateModel = new SpinnerDateModel(value, start, end, Calendar.YEAR);  
JSpinner spinner = new JSpinner(dateModel);
```

(예제)

```
SpinnerTest.java  
  
import javax.swing.*;  
import java.awt.*;  
import java.util.*;  
  
public class SpinnerTest extends JFrame {  
    public SpinnerTest() {  
        setTitle("스피너 테스트");  
        setSize(500, 100);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        JPanel panel = new JPanel();  
        JSpinner spinner;  
  
        // List Model 테스트  
        String[] items = { "소설", "잡지", "전공서적", "취미" };  
        SpinnerListModel listModel = new SpinnerListModel(items);
```

```

    spinner = new JSpinner(listModel);
    panel.add(spinner);

    // Number Model 테스트
    SpinnerModel numberModel = new SpinnerNumberModel(1, 0, 9, 1);
    spinner = new JSpinner(numberModel);
    panel.add(spinner);

    // Date Model 테스트
    Calendar calendar = Calendar.getInstance();
    Date value = calendar.getTime(); // 현재 날짜
    calendar.add(Calendar.YEAR, -50); // 현재값보다 -50년 증가
    Date start = calendar.getTime();
    calendar.add(Calendar.YEAR, 100); // 주의 +50하면 현재 날짜가 됨
    Date end = calendar.getTime();
    SpinnerDateModel dateModel = new SpinnerDateModel(value, start, end, Calendar.YEAR);
    spinner = new JSpinner(dateModel);
    // 날짜를 편집할 수 있는 편집기를 지정한다.
    spinner.setEditor(new JSpinner.DateEditor(spinner, "yyyy/MM/dd"));
    panel.add(spinner);
    add(panel);
    setVisible(true);
}

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

```

날짜(연도/월/일)를 편집할 수 있도록 지원하는 스피너의 내장된 편집기 생성자이다

setEditor()는 스피너의 편집기를 지정한다. 편집기란 값들을 사용자가 어떻게 변경할 수 있는지를 지정하는 것이

다. JSpinner.DateEditor()는 (연도/월/일)과 같이 한국식으로 날짜를 편집할 수 있도록 지원하는 스피너의 내장된 편집기 생성자이다.



5. 슬라이더

슬라이더(slider)는 사용자가 특정한 범위 안에서 하나의 값을 선택할 수 있는 컴포넌트이다. 슬라이더는 특히 사용자로부터 일정 범위 안의 수치값을 입력받을 때 편리하다.



1) 생성자

생성자	설명
<code>JSlider()</code>	0부터 100사이의 수평 슬라이더를 생성한다.
<code>JSlider(int min, int max)</code> <code>JSlider(int min, int max, int value)</code>	지정된 최소값과 최대값을 가지는 수평 슬라이더를 생성한다. value는 슬라이더의 초기값이다.

슬라이더를 생성하려면 `JSlider` 클래스의 생성자를 호출하여야 한다. 예를 들어서 0에서 100까지의 범위를 가지고 초기값이 50인 슬라이더는 다음과 같이 생성할 수 있다.

```
slider = new JSlider(0, 100, 50);
```

만약 매개 변수를 주지 않았더라도 디폴트로 0에서 100까지의 슬라이더가 생성된다. 슬라이더를 좀 더 사용하기 쉽게 하려면 장식을 덧붙인다. 예를 들어서 슬라이더에 눈금을 표시하려면 다음과 같이 한다.

```
slider.setMajorTickSpacing(10); // 큰 눈금 간격  
slider.setMinorTickSpacing(1); // 작은 눈금 간격  
slider.setPaintTicks(true); // 눈금을 표시한다  
slider.setPaintLabels(true); // 값을 레이블로 표시한다
```

2) 이벤트 처리

슬라이더의 값이 변경되었을 경우, 이벤트를 받고 싶으면 `Change` 리스너를 등록하면 된다.

```
slider.addChangeListener(this);
```

이벤트 리스너 안에서 슬라이더의 값을 읽기 위해서는 `getValue()` 메소드를 사용한다.

```
public void stateChanged(ChangeEvent e) {  
    JSlider source = (JSlider) e.getSource();  
    if (!source.getValueIsAdjusting()) {  
        int value = (int) source.getValue();  
        button.setSize(value * 10, value * 10);  
    }  
}
```

(예제)

다음 예제에서는 슬라이더를 움직이면 이미지의 크기가 변경된다.

```
SliderTest.java  
import java.awt.*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.applet.Applet;
```

```

import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class SliderTest extends JFrame implements ChangeListener {
    static final int INIT_VALUE = 15;
    private JButton buttonOK;
    private JSlider slider;
    private JButton button;

    public SliderTest() {
        JPanel panel;
        setTitle("슬라이더 테스트");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        panel = new JPanel();
        JLabel label = new JLabel("슬라이더를 움직여보세요", JLabel.CENTER);
        label.setAlignmentX(Component.CENTER_ALIGNMENT);
        panel.add(label);
        // 슬라이더를 생성하고 눈금을 붙인다. 이벤트 리스너를 등록한다.
        slider = new JSlider(0, 30, INIT_VALUE);
        slider.setMajorTickSpacing(10);    // 큰 눈금 간격
        slider.setMinorTickSpacing(1);    // 작은 눈금 간격
        slider.setPaintTicks(true);        // 눈금을 표시한다.
        slider.setPaintLabels(true);        // 값을 레이블로 표시한다.
        slider.addChangeListener(this);    // 이벤트 리스너를 붙인다.
        panel.add(slider);

        button = new JButton("");
        ImageIcon icon = new ImageIcon("dog.gif");
        button.setIcon(icon);
        button.setSize(INIT_VALUE * 10, INIT_VALUE * 10);
        panel.add(button);
        add(panel);

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

    public void stateChanged(ChangeEvent e) {
        JSlider source = (JSlider) e.getSource();
        if (!source.getValueIsAdjusting()) {

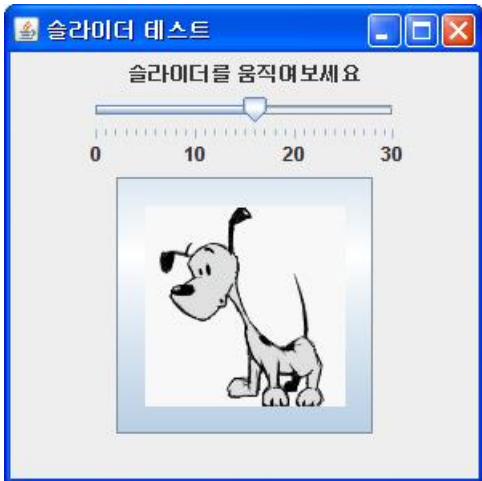
```

```

        int value = (int) source.getValue();
        button.setSize(value * 10, value * 10);
    }
}

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

```



6. 트리

트리(tree)는 계층적인 데이터를 표시하는데 사용되는 컴포넌트이다. 트리는 JTree 클래스로부터 생성된다. 사용 시 한 가지 주의점은 JTree는 단순히 데이터의 뷰만을 제공한다는 점이다. JTree 객체 안에는 데이터가 저장되지 않는다. 각각의 트리는 하나의 루트 노드(root node)를 가지고 있다. 다른 노드들은 루트 노드에서 파생된다. 노드는 자식 노드를 가질 수도 있고 가지지 않을 수도 있다. 사용자는 마우스를 클릭하여서 노드의 자식 노드를 보이게 할 수도 있고 보이지 않게 할 수도 있다. 프로그램에서는 이벤트를 사용하여서 노드의 확장 상태를 감지할 수 있다.



1) 트리 생성하기

생성자	설명
JTree()	비어있는 트리를 생성한다.
JTree(TreeNode root)	매개변수로 주어진 노드를 루트로 하여 트리를 생성한다.

참조 변수를 선언하고 JTree 클래스의 생성자를 호출하여서 객체를 생성한다.

```
JTree tree = new JTree(root);
```

화면에 나타나는 행의 수는 setVisibleRowCount()를 이용하여서 설정할 수 있다.

```
tree.setVisibleRowCount(10);
```

만약 사용자의 선택 모델을 바꾸려면 다음과 같이 한다.

```
tree.getSelectionModel().setSelectionModel(TreeSelectionModel.SINGLE_TREE_SELECTION);
```

그리고 트리는 매우 길 수 있기 때문에 스크롤 페인 속에 넣는 것이 일반적이다.

```
JScrollPane scroll = new JScrollPane(tree);
```

- JTree 클래스의 메소드

메소드	설명
<code>Object getLastSelectedPathComponent()</code>	현재 선택된 노드를 반환한다.
<code>TreeSelectionModel.getSelectionModel()</code>	트리의 선택 모델을 반환한다.
<code>void setVisibleRowCount(int count)</code>	화면에 보이는 행의 개수를 설정한다.

2) 노드 생성하기

트리가 생성되면 노드를 생성하여서 트리 안에 추가하여야 한다. 가장 간단한 방법은 `DefaultMutableTreeNode`를 사용하는 것이다. `DefaultMutableTreeNode` 클래스는 다음과 같은 생성자를 가진다.

생성자	설명
<code>DefaultMutableTreeNode()</code>	비어있는 트리 노드를 생성한다.
<code>DefaultMutableTreeNode(Object obj)</code>	지정된 객체를 가지고 트리 노드를 생성한다.

생성자를 이용하여서 다음과 같이, 필요한 노드들을 생성한다.

```
DefaultMutableTreeNode root = new DefaultMutableTreeNode("루트 노드");  
DefaultMutableTreeNode child1 = new DefaultMutableTreeNode("첫 번째 자식 노드");  
DefaultMutableTreeNode child2 = new DefaultMutableTreeNode("두 번째 자식 노드");
```

`add()` 메소드를 이용하여서 자식 노드를 부모 노드에 추가한다.

```
root.add(child1);  
root.add(child2);
```

- 메소드

메소드	설명
<code>void add(TreeNode child)</code>	자식 노드를 추가한다.
<code>TreeNode getFirstChild()</code>	현재 노드의 첫번째 자식 노드를 반환한다.
<code>DefaultMutableTreeNode getNextSibling()</code>	현재 노드의 다음 형제 노드를 반환한다.
<code>TreeNode getParent()</code>	부모 노드를 반환한다.

3) 선택된 노드를 가져오는 방법

트리에서 사용자가 선택한 노드를 가장 쉽게 가져오는 방법은 `getLastSelectedPathComponent()`을 호출하는 것이다. 이 메소드는 `Object` 타입을 반환하기 때문에 사용하기 전에 형변환하여야 한다.

```
DefaultMutableTreeNode node;  
node = (DefaultMutableTreeNode) tree.getLastSelectedPathComponent();
```

4) 이벤트 처리

만약 프로그램이 사용자가 노드를 선택할 때마다 어떤 작업을 해야 한다면, 선택 이벤트를 처리하는 리스너를 만들어주면 된다. 리스너는 다음과 같이 TreeSelectionListener를 구현하여야 한다.

```
public class TreeTest extends JFrame
{
    //단일 선택 모드로 변경
    tree.getSelectionModel().setSelectionMode
        (TreeSelectionMode.SINGLE_TREE_SELECTION);
    ...
    //선택이 변경되는지 관찰한다.
    tree.addTreeSelectionListener(new TreeListener());
    ...
    private class TreeListener implements TreeSelectionListener {
        public void valueChanged(TreeSelectionEvent e) {
            DefaultMutableTreeNode node = (DefaultMutableTreeNode)
                tree.getLastSelectedPathComponent();
            if (node == null)
                return;
            String s = (String) node.getUserObject();
            label.setText(s);
        }
    }
}
```

트리 안의 노드를 선택하면 호출된다.
현재는 노드의 이름을 얻어서 레이블의 텍스트를 변경한다.

여기서 노드의 이름을 가져오는 메소드는 getUserObject()이다.

(예제) 문서를 저장하고 있는 트리를 생성하고 사용자가 노드를 선택하면 노드의 텍스트를 레이블에 나타내는 프로그램

TreeTest.java
<pre>import javax.swing.*; import java.awt.event.*; import javax.swing.tree.*; import javax.swing.event.*; public class TreeTest extends JFrame { private JTree tree; private JLabel label; private DefaultTreeModel model; public TreeTest() { this.setSize(400, 300); this.setTitle("트리 테스트"); this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 트리의 노드 생성 DefaultMutableTreeNode root = new DefaultMutableTreeNode("나의 문서함"); DefaultMutableTreeNode child1 = new DefaultMutableTreeNode("편지"); DefaultMutableTreeNode child2 = new DefaultMutableTreeNode("음악"); DefaultMutableTreeNode child3 = new DefaultMutableTreeNode("한국");</pre>

```

DefaultMutableTreeNode child4 = new DefaultMutableTreeNode("전통");
DefaultMutableTreeNode child5 = new DefaultMutableTreeNode("힙팝");
DefaultMutableTreeNode child6 = new DefaultMutableTreeNode("외국");
DefaultMutableTreeNode child7 = new DefaultMutableTreeNode("소셜");
DefaultMutableTreeNode child8 = new DefaultMutableTreeNode("추리");
// 계층 관계 생성
root.add(child1);
root.add(child2);
root.add(child7);
child2.add(child3);
child2.add(child6);
child3.add(child4);
child3.add(child5);
child7.add(child8);
// 트리 생성
tree = new JTree(root);
tree.setVisibleRowCount(10);
tree.addTreeSelectionListener(new TreeListener());

JPanel panel = new JPanel();
JScrollPane scroll = new JScrollPane(tree);
panel.add(scroll);

label = new JLabel();
panel.add(label);
add(panel);
this.setVisible(true);
}
// 트리에서 노드가 생성되면 호출, 트리의 이름을 읽어서 레이블의 텍스트를 변경한다.
private class TreeListener implements TreeSelectionListener {
    public void valueChanged(TreeSelectionEvent e) {
        DefaultMutableTreeNode node = (DefaultMutableTreeNode) tree
            .getLastSelectedPathComponent();
        if (node == null)
            return;
        String s = (String) node.getUserObject();
        label.setText(s);
    }
}

public static void main(String[] args) {
    // 룩앤필을 윈도우로 변경한다.

```



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

new TreeTest();
}
```

7. 색상 선택기

애플리케이션에 따라서 사용자가 색상을 선택하도록 하는 경우도 많다. 이 때 간편하게 사용할 수 있는 클래스가 JColorChooser이다. 이 클래스는 색상 팔레트에서 사용자가 하나의 색상을 선택할 수 있게 한다.



색상 선택기는 두 부분으로 구성된다. 탭이 붙어있는 패널과 미리 보기 패널이다. 탭이 붙어있는 패널은 각기 다른 색상 모델을 지원한다. 미리보기 패널은 현재 선택된 색상을 다양한 방식으로 표시한다. 아래의 예제는 간단히 색상 선택기를 생성시키고 현재 선택된 색상을 이벤트 리스너에서 가져온다. JColorChooser 생성자는 최초 색상을 나타내는 Color 매개 변수를 받는다. 만약 초기 색상을 지정하지 않으면 색상 선택기는 Color.white를 표시한다.

```
JColorChooser colorChooser = new JColorChooser(Color.RED);
```

(예제)

```
ColorChooserTest.java

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

public class ColorChooserTest extends JFrame implements ChangeListener {
```

```

protected JColorChooser color;

public ColorChooserTest() {
    setTitle("색상 선택기 테스트");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    // 색상 선택기를 생성하고 이벤트 리스너를 붙인다.
    color = new JColorChooser();
    color.getSelectionModel().addChangeListener(this);
    color.setBorder(BorderFactory.createTitledBorder("색상 선택"));

    JPanel panel = new JPanel();
    panel.add(color);
    add(panel);
    pack();
    this.setVisible(true);
}

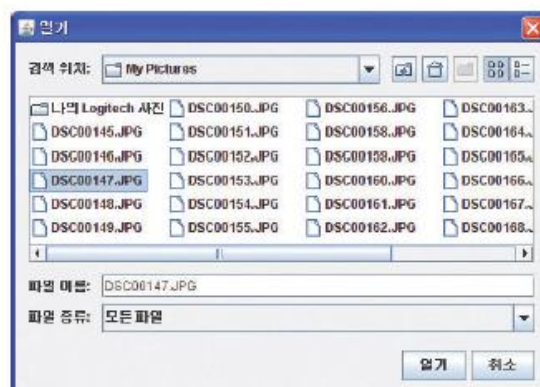
public void stateChanged(ChangeEvent e) {
    Color newColor = color.getColor();
}

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

```

2) 파일 선택기

파일 선택기는 파일 시스템을 탐색하여 파일이나 디렉토리를 선택하는 대화 상자를 제공한다. 파일 선택기를 표시하려면 JFileChooser 클래스를 사용한다. JFileChooser 클래스는 모달 대화 상자를 표시한다. JFileChooser 객체는 단지 파일을 선택하기 위한 GUI만을 제공한다. 선택된 파일을 열거나 쓰는 것은 사용자 프로그램 책임이다.



표준 열기 대화 상자를 만드는 것은 다음의 문장으로 한다.

```
//파일 선택기를 생성한다
final JFileChooser fc = new JFileChooser();

...

//버튼이 클릭되면 반환된다
int returnVal = fc.showOpenDialog(aComponent);
```

showOpenDialog() 메소드의 매개 변수로 대화 상자의 부모 컴포넌트를 전달한다. 부모 컴포넌트는 대화 상자의 위치에 영향을 준다. 디폴트로 파일 선택기는 사용자의 홈 디렉토리에 있는 모든 파일을 표시한다. 사용자는 setCurrentDirectory() 메소드를 사용하여 파일 선택기의 초기 디렉토리를 지정할 수 있다. showOpenDialog() 메소드 호출은 일반적으로 액션 리스너에 등장한다.

```
public void actionPerformed(ActionEvent e) {
    //"파일열기"라는 버튼의 액션 이벤트를 처리한다
    if (e.getSource() == openButton) {
        int returnVal = fc.showOpenDialog(FileChooserTest.this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            // 파일에 대한 처리를 한다
        } else {
            // 파일 선택이 사용자에게 의하여 취소되었음
        }
    }
    ...
}
```

showXXXDialog() 메소드는 사용자가 파일을 선택하였는지를 나타내는 정수를 반환한다. 일반적으로는 반환값이 APPROVE_OPTION 인지만을 검사하면 된다. 선택된 파일을 얻기 위해서는 getSelectedFile() 메소드를 호출한다. 이 메소드는 File의 인스턴스를 반환한다. 파일을 저장하는 대화상자는 다음과 같은 문장으로 생성할 수 있다.

```
int returnVal = fc.showSaveDialog(FileChooserTest.this);
```

(예제)

```
FileChooserTest.java

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

public class FileChooserTest extends JFrame implements ActionListener {
    JButton openButton, saveButton;
    JFileChooser fc;
```

```

public FileChooserTest() {
    setTitle("파일 선택기 테스트");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(300, 200);

    fc = new JFileChooser();

    JLabel label = new JLabel("파일 선택기 컴포넌트 테스트입니다.");
    openButton = new JButton("파일 오픈");
    openButton.addActionListener(this);

    saveButton = new JButton("파일 저장");
    saveButton.addActionListener(this);

    JPanel panel = new JPanel();
    panel.add(label);
    panel.add(openButton);
    panel.add(saveButton);
    add(panel);
    setVisible(true);
}

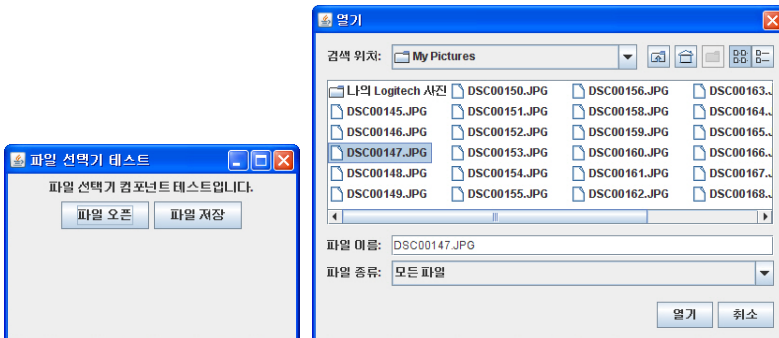
public void actionPerformed(ActionEvent e) {
    // "파일 오픈"버튼에 대한 액션 이벤트 처리
    if (e.getSource() == openButton) {
        int returnVal = fc.showOpenDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            // 실제 파일을 오픈한다.
        } else {
            // 사용자 취소
        }
        // "파일 저장"버튼에 대한 액션 이벤트 처리
    } else if (e.getSource() == saveButton) {
        int returnVal = fc.showSaveDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            // 실제 파일에 저장한다.
        } else {
            // 사용자 취소
        }
    }
}

```

```

    }
    public static void main(String[] args) {
        FileChooserTest frame = new FileChooserTest();
    }
}

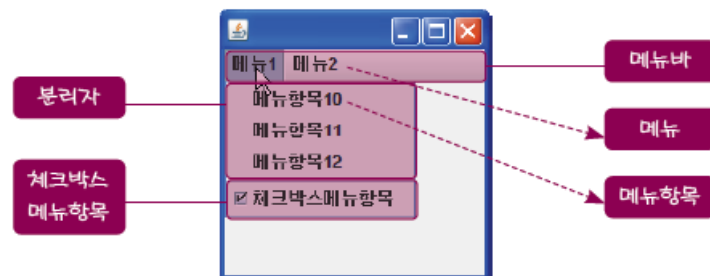
```



8. 메뉴

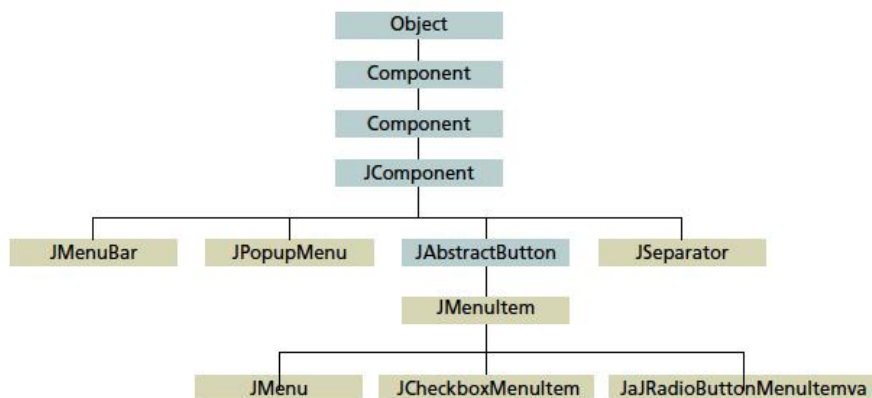
메뉴(menu)는 공간을 절약하면서 사용자가 여러 가지 옵션 중에서 하나를 선택하게 하는 방법이다. 메뉴는 다른 컴포넌트와는 달리 컨테이너 안에 배치되지 않는다. 메뉴는 메뉴바에 나타나거나 팝업 메뉴로만 나타난다. 메뉴바는 윈도우 상단에 위치하면서 여러 개의 메뉴를 가지고 있다. 팝업 메뉴는 사용자가 팝업이 가능한 컴포넌트 위에서 마우스 오른쪽 버튼을 누르면 그 위치에 등장한다.

메뉴 항목은 이미지나 텍스트를 가질 수 있다. 물론 폰트나 색상 등도 변경이 가능하다. 메뉴에 관련된 용어는 다음과 같다.



1) 메뉴 컴포넌트 계층도

메뉴 관련 클래스의 계층도는 다음과 같다.



계층도에서도 알 수 있듯이 메뉴는 간단하게는 버튼이다. 그 이유는 버튼과 마찬가지로 누르면 이벤트가 발생하기 때문이다. 하지만 버튼과는 다르게 이벤트 처리기에서 하위 메뉴들을 표시하게 된다. 메뉴바는 JMenuBar 클래스에 의하여 지원된다. 메뉴바는 프레임에만 부착될 수 있다. 패널에는 부착할 수 없다. 팝업 메뉴는 JPopupMenu 클래스로 표현되는데 마우스의 오른쪽 버튼 클릭과 같은 특정한 이벤트가 발생하면 화면에 보여지게 된다. 메뉴 항목은 JMenuItem 객체로 표현된다. 메뉴는 JMenu 객체에 의해 표현되는데 JMenuItem의 서브 클래스로 구현되기 때문에 메뉴 안의 메뉴, 즉 서브 메뉴를 쉽게 만들 수 있다. 각 메뉴 항목들은 체크 박스를 포함할 수 있는데 이것은 JCheckboxMenuItem 객체에 의하여 표현된다.

2) 메뉴 생성

기본적인 메뉴의 생성 순서는 다음과 같다.

```
// ❶ 메뉴바 관련 변수를 선언한다.
JMenuBar menuBar;      // 메뉴바
JMenu menu;            // 메뉴
JMenuItem menuItem;    // 메뉴 항목

// ❷ 메뉴바를 생성한다.
menuBar = new JMenuBar();

// ❸ 메뉴를 생성하여 메뉴바에 추가한다.
menu = new JMenu("메뉴1");
menuBar.add(menu);

// ❹ 메뉴 항목을 생성하여 메뉴에 추가한다.
menuItem = new JMenuItem("메뉴항목1", KeyEvent.VK_T);
menu.add(menuItem);

// ❺ 프레임에 메뉴바를 설정한다.
frame.setJMenuBar(mb);
```

Jframe에 대한 메뉴바를 설정하려면 setJMenuBar() 메소드를 사용하면 된다. JMenu를 JMenuBar에 추가하기 위해서는 add(JMenu) 메소드를 사용한다. 메뉴 항목을 메뉴에 추가하기 위해서는 add(JMenuItem) 메소드를 사용한다. 이상의 순서로 풀다운 메뉴(full down menu)가 생성된다. 또한, setMenuShortcut 클래스를 사용하여 각 메뉴 항목에 키보드 단축키(keyboard shoutcut)를 설정하는 것도 가능하다.

2) 이벤트 처리

사용자가 JMenuItem을 선택한 것을 감지하기 위해서는 액션 이벤트를 처리하여야 한다.

```
public class MenuTest ... implements ActionListener {

    ...

    public MenuTest() {

        ...

        // 각각의 메뉴 항목에 대하여 이벤트 처리기 등록
        menuItem.addActionListener(this);

    }

    public void actionPerformed(ActionEvent e) {
```

```
// 액션 이벤트 처리
```

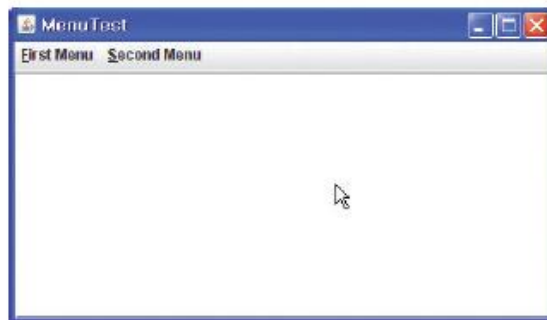
```
}
```

```
}
```

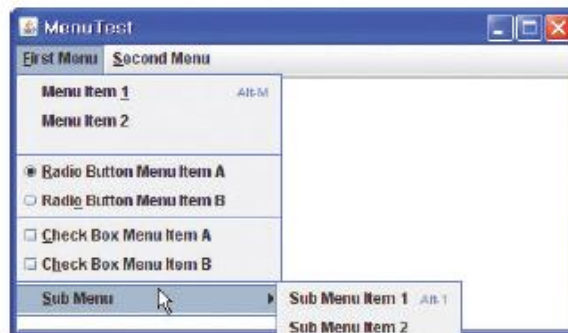
3) 키보드와의 관계

메뉴는 기억키(mnemonics)와 가속키(accelerators)의 두 가지 종류의 키보드 선택을 지원한다. 기억키는 키보드를 사용하여 메뉴 계층을 둘러보는데 이용된다. 반면에 가속키는 메뉴 계층 구조를 통하지 않고 직접 메뉴 항목을 선택할 수 있는 키보드 지름길을 제공한다. 기억키는 모든 사용자를 위한 것이고 가속키는 고급 사용자를 위한 것이다.

기억키는 이미 표시되어 있는 메뉴 항목을 선택되게 한다. 예를 들어서 다음의 프로그램에서 첫 번째 메뉴는 기억키 A를 가진다. 이것이 의미하는 바는 프로그램을 수행시킬 때 Alt + A 키를 누르면 첫 번째 메뉴가 표시된다는 것을 의미한다.



가속키는 메뉴 항목이 선택되는 키들의 조합이다. 예를 들어서 Alt + 1를 누르면 Sub Menu에서 첫 번째 메뉴 항목인 Sub Menu Item 1이 선택되게 한다. 단말 메뉴 항목만이 가속키를 가질 수 있다.



기억키는 `setMnemonic()` 메소드를 사용하여 지정할 수 있다. 가속키는 `setAccelerator()` 메소드를 사용한다.

```
// 기억키를 지정한다
```

```
menuItem.setMnemonic(KeyEvent.VK_T);
```

```
// 가속키를 설정한다
```

```
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_T, ActionEvent.ALT_MASK));
```

(예제)

프레임을 생성한 후에 메뉴바를 붙인다. 메뉴바에는 두 개의 메뉴를 생성하여 추가하고 첫 번째 메뉴에 다양한 항목들을 생성하여 추가하여 보자.

MenuTest.java

```

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

public class MenuTest extends JFrame implements ActionListener, ItemListener {
    private JMenuBar menuBar;
    private JMenu menu, submenu;
    private JMenuItem menuItem1, menuItem2;
    private JMenuItem sbMenuItem1, sbMenuItem2;
    private JRadioButtonMenuItem rbMenuItem1, rbMenuItem2;
    private JCheckBoxMenuItem cbMenuItem1, cbMenuItem2;

    public MenuTest() {
        // 메뉴바를 생성한다.
        menuBar = new JMenuBar();
        // 첫번째 메뉴를 생성
        menu = new JMenu("첫번째 메뉴");
        menu.setMnemonic(KeyEvent.VK_F);
        menuBar.add(menu);
        // 메뉴 항목들을 생성
        menuItem1 = new JMenuItem("메뉴 항목 1", KeyEvent.VK_1);
        menuItem1.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_M,
            ActionEvent.ALT_MASK));
        menuItem1.addActionListener(this);
        menu.add(menuItem1);

        ImageIcon icon = new ImageIcon("icon.gif");
        menuItem2 = new JMenuItem("메뉴 항목 2", icon);
        menu.add(menuItem2);

        // 라디오 버튼 메뉴 항목들을 생성하여 메뉴에 추가
        menu.addSeparator();
        ButtonGroup group = new ButtonGroup();

        rbMenuItem1 = new JRadioButtonMenuItem("라디오 버튼 메뉴 항목 1");
        rbMenuItem1.setSelected(true);
        group.add(rbMenuItem1);
        menu.add(rbMenuItem1);

        rbMenuItem2 = new JRadioButtonMenuItem("라디오 버튼 메뉴 항목 2");
        group.add(rbMenuItem2);
    }

```



```

        menu.add(rbMenuItem2);

        // 체크 박스 메뉴 항목들을 생성하여 메뉴에 추가
        menu.addSeparator();
        cbMenuItem1 = new JCheckBoxMenuItem("체크 박스 메뉴 항목 1");
        cbMenuItem1.addItemListener(this);
        menu.add(cbMenuItem1);

        cbMenuItem2 = new JCheckBoxMenuItem("체크 박스 메뉴 항목 2");
        menu.add(cbMenuItem2);

        // 서브 메뉴 생성
        menu.addSeparator();
        submenu = new JMenu("서브 메뉴");
        submenu.setMnemonic(KeyEvent.VK_S);

        sbMenuItem1 = new JMenuItem("서브 메뉴 항목 1");
        sbMenuItem1.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_1,
            ActionEvent.ALT_MASK));
        submenu.add(sbMenuItem1);

        sbMenuItem2 = new JMenuItem("서브 메뉴 항목 2");
        submenu.add(sbMenuItem2);
        menu.add(submenu);

        // 메뉴바의 두번째 메뉴를 작성한다.
        menu = new JMenu("두번째 메뉴");
        menu.setMnemonic(KeyEvent.VK_S);
        menuBar.add(menu);

        // 메뉴바를 프레임에 부착한다.
        setJMenuBar(menuBar);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == menuItem1) {
            JOptionPane.showMessageDialog(this, "메뉴 항목 1이 선택되었습니다.");
        }
    }
}

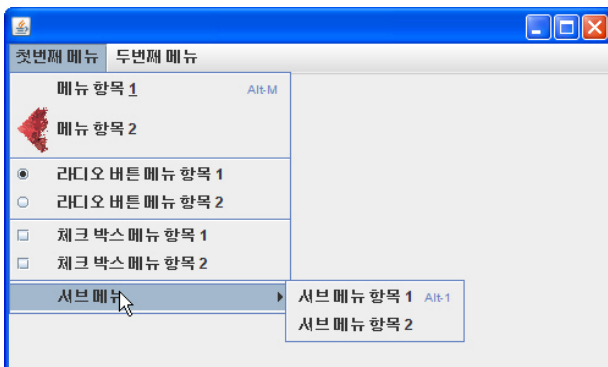
```

```

        public void itemStateChanged(ItemEvent e) {
            if (e.getSource() == cbMenuItem1) {
                JOptionPane.showMessageDialog(this, "체크 박스 메뉴 항목 1이 변경되었습니다.");
                // 체크 박스 메뉴 항목에 대한 처리
            }
        }

        public static void main(String args[]) {
            MenuTest f = new MenuTest();
            f.setSize(500, 200);
            f.setVisible(true);
        }
    }
}

```



4) 팝업 메뉴 예제

팝업 메뉴를 생성하기 위해서는, 팝업 메뉴와 연관되는 컴포넌트에 마우스 리스너를 등록하여야 한다. 마우스 리스너에서 사용자가 마우스를 누른 것을 감지하고 팝업 메뉴를 표시한다. 일반적으로 윈도우에서는 사용자가 마우스의 오른쪽 버튼을 눌렀다가 떼면 팝업 메뉴가 표시된다.

(예제)

PopupMenuTest.java

```

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

public class PopupMenuTest extends JFrame implements ActionListener {
    private JPopupMenu popup;
    private JMenuItem menuItem1, menuItem2;
    private JTextArea ta;

    public PopupMenuTest() {
        ta = new JTextArea(10, 20);
        add(ta);
    }
}

```

```

        // 팝업 메뉴를 생성한다.
        popup = new JPopupMenu();
        menulitem1 = new JMenuItem("탐색 메뉴");
        menulitem1.addActionListener(this);
        popup.add(menulitem1);
        menulitem2 = new JMenuItem("저장 메뉴");
        menulitem2.addActionListener(this);
        popup.add(menulitem2);

        // 팝업 메뉴를 생성하기 위하여 텍스트 영역에 마우스 리스너 등록
        ta.addMouseListener(new PopupListener());
        setVisible(true);
    }

    class PopupListener extends MouseAdapter {
        public void mousePressed(MouseEvent e) {
            showPopup(e);
        }
        public void mouseReleased(MouseEvent e) {
            showPopup(e);
        }
        private void showPopup(MouseEvent e) {
            if (e.isPopupTrigger()) {
                popup.show(e.getComponent(), e.getX(), e.getY());
            }
        }
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == menulitem1) {
            JOptionPane.showMessageDialog(this, "탐색이 선택되었습니다.");
        }
    }

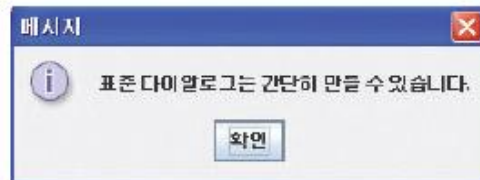
    public static void main(String args[]) {
        PopupMenuTest f = new PopupMenuTest();
        f.setTitle("팝업 메뉴 테스트");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(300, 200);
        f.setVisible(true);
    }
}

```

9. 대화 상자

대화 상자(dialog box)윈도우는 스윙 애플리케이션 윈도우와는 별도로 임시 정보를 나타내는데 사용되는 독립적인 서브 윈도우이다. 대부분의 대화 상자는 사용자에게 오류나 경고를 나타내는데 사용된다.

사용자의 편의를 위하여 몇 개의 클래스들은 아주 간단하게 직접 대화 상자 객체를 생성하여서 표시한다. 예를 들어서 JOptionPane은 주로 표준화된 간단한 대화 상자를 생성하고 ProgressMonitor는 진행 과정을 보여주는 간단한 대화 상자를 생성하여서 표시한다. 앞에서 학습한 JColorChooser와 JFileChooser 역시 표준적인 윈도우를 제공한다. 만약 맞춤형 대화 상자를 생성하려면 JDialog 클래스를 사용하면 된다.



이러한 간단한 대화 상자는 대개 한 줄이면 된다.

```
JOptionPane.showMessageDialog(frame, "표준 대화 상자는 간단히 만들 수 있습니다.");
```

대화 상자는 프레임에 의존적이다. 프레임이 삭제되면 대화 상자도 함께 삭제된다. 프레임이 아이콘화되면 대화 상자도 아이콘화된다. 이러한 특성은 AWT의 Dialog 클래스로부터 상속된 것이다.

1) 대화 상자의 종류

대화 상자에는 모달형과 비모달형의 두 가지 종류가 있다.

- 모달형(Modal Dialog)

대화 상자를 끝내지 않고서는 다른 작업을 할 수 없는 대화 상자. 대표적으로 JOptionPane이 모달형 대화상자이다.

- 비모달형(Non-Modal Dialog)

대화 상자를 끝내지 않고서도 사용자가 다른 윈도우를 사용하여 작업할 수 있다. 이러한 대화 상자를 생성하려면 JDialog 클래스를 직접 사용하여야 한다. 디폴트로 다이얼로그들은 비모달형이다. 모달형 또는 비모달형의 지정은 생성자에서도 할 수 있고 또는 setModal()메소드를 사용하여 지정할 수도 있다.

2) JOptionPane의 개요

JOptionPane을 사용하면 빠르게 다양한 종류의 대화 상자를 생성할 수 있다. JOptionPane은 표준적인 대화 상자를 위한 배치 관리, 아이콘 선택, 제목과 텍스트, 버튼의 텍스트를 변경할 수 있다.

JOptionPane은 많은 메소드들을 제공한다. 대부분의 간단한 모달형 대화 상자를 생성하기 위해서는 JOptionPane의 showXXXDialog() 메소드를 사용하면 된다. 이 중에서 showMessageDialog(), showOptionDialog(), showInputDialog()를 살펴보자.

① showMessageDialog()

버튼에 하나 있는 간단한 대화 상자를 생성하고 표시한다. 다음과 같이 메시지, 타이틀과 아이콘을 변경할 수 있다.

```
JOptionPane.showMessageDialog(this, "범위를 초과하였습니다.", "경고",  
JOptionPane.WARNING_MESSAGE);
```



아이콘은 다음 4개의 표준 아이콘(question, information, warning, error) 중에서 하나를 사용할 수 있고 사용자가 지정할 수도 있다.

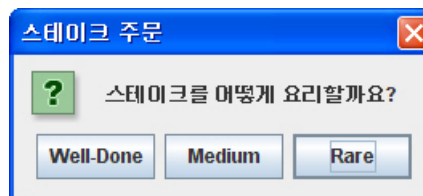


아이콘을 PLAIN_MESSAGE로 지정하면 아이콘이 사라진다.

② showOptionDialog()

버튼과 아이콘, 메시지, 제목을 이용하여서 모달형 대화 상자를 표시한다. 이 메소드를 이용하여서 표준 대화 상자의 버튼에 나타나는 텍스트를 변경할 수 있다.

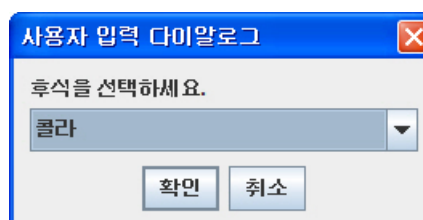
```
Object[] options = {"Well-Done", "Medium", "Rare"};
int n = JOptionPane.showOptionDialog( this, "스테이크를 어떻게 요리할까요?", "스테이크 주문",
    JOptionPane.YES_NO_CANCEL_OPTION,
    JOptionPane.QUESTION_MESSAGE, null,
    options, options[2]);
```



③ showInputDialog()

showInputDialog()를 이용하면 사용자로부터 문자열을 입력받을 수 있다. 이 메소드는 문자열을 가리키는 Object 참조값을 반환한다. 아래는 사용자가 3개의 문자열 중에서 하나를 선택하도록 하는 showInputDialog()의 예제이다.

```
Object[] dessert = {"우유", "콜라", "커피"};
String s=(String)JOptionPane.showInputDialog(frame, "후식을 선택하세요.",
    "사용자 입력 대화 상자", JOptionPane.PLAIN_MESSAGE,
    null, dessert, "ham");
```



3) 사용자 맞춤형 대화 상자

만약 위의 대화 상자 중에서 자신이 원하는 대화 상자가 없는 경우에는 JDialog 클래스를 이용하여서 직접 대화 상자를 작성할 수 있다.

생성자	설명
Dialog(Frame parent)	타이틀이 없는 Dialog을 생성한다.
Dialog(Frame parent, boolean modal)	타이틀이 없는 Dialog을 생성한다. modal이 true인 경우에는 Modal Dialog, false인 경우에는 Non-Modal Dialog이다.
Dialog(Frame parent, String title)	타이틀이 있는 Dialog을 생성한다.
Dialog(Frame parent, String title, boolean modal)	타이틀이 있는 Dialog을 생성한다. modal이 true인 경우에는 Modal Dialog, false인 경우에는 Non-Modal Dialog이다.

메소드	설명
show()	Dialog를 표시한다.
setModal(boolean b)	true로 지정하면 Modal이고, false이면 Non-Modal이다.
isModal()	Dialog가 Modal이면 true를 반환한다.
String getTitle()	Dialog의 Title을 반환한다.
String setTitle(String title)	Dialog의 Title을 설정한다.
boolean isResizable()	크기를 변경할 수 있는 Dialog이면 true를 반환한다.
setResizable(boolean b)	true이면 크기가 변경가능한 Dialog가 된다.

(예제) 버튼을 누르면 "ok"와 "cancel"버튼을 가지고 있는 모달형 대화상자가 나타나는 프로그램

DialogTest.java
<pre> import java.awt.*; import java.awt.event.*; import javax.swing.*; public class DialogTest extends JFrame implements ActionListener { JPanel p; JButton b1, b2; SimpleDialog aDialog; DialogTest() { p = new JPanel(); add(p, "Center"); b1 = new JButton("대화상자 생성"); </pre>

```

        b2 = new JButton("종료");
        b1.addActionListener(this);
        b2.addActionListener(this);
        p.add(b1);
        p.add(b2);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == b1) {
            if (aDialog == null) {
                // 대화 상자가 아직 생성되지 않았으면 생성한다.
                aDialog = new SimpleDialog(this, "Simple Dialog");
            }
            // 반드시 메소드를 호출하여야 화면에 표시한다.
            aDialog.setVisible(true);
            b1.requestFocus();
        }
        if (e.getSource() == b2) {
            System.exit(0);
        }
    }
    public static void main(String args[]) {
        DialogTest f = new DialogTest();
        f.setTitle("DialogTest");
        f.setSize(300, 100);
        f.setVisible(true); // (2)
    }
}

```

// Dialog 클래스로부터 상속을 받아서 SimpleDialog 클래스를 만든다.

```

class SimpleDialog extends JDialog implements ActionListener {
    JPanel p1, p2;
    JLabel l;
    JButton okButton;
    JButton cancelButton;

    SimpleDialog(Frame parent, String str) {
        // Dialog 클래스의 생성자를 호출하여 모달형 대화 상자를 만든다.
        super(parent, str, true);
        p1 = new JPanel();
        p1.setLayout(new BorderLayout());
        add(p1);
    }
}

```

```

        l = new JLabel("간단한 대화 상자입니다.", JLabel.CENTER);
        p1.add(l, "Center");
        p2 = new JPanel();
        p1.add(p2, "South");
        okButton = new JButton("OK");
        okButton.addActionListener(this);
        p2.add(okButton);
        cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(this);
        p2.add(cancelButton);
        setSize(300, 200);
        // 사용자가 윈도우를 닫으면 대화 상자를 제거한다.
        addWindowListener(new MyWinListener());
    }

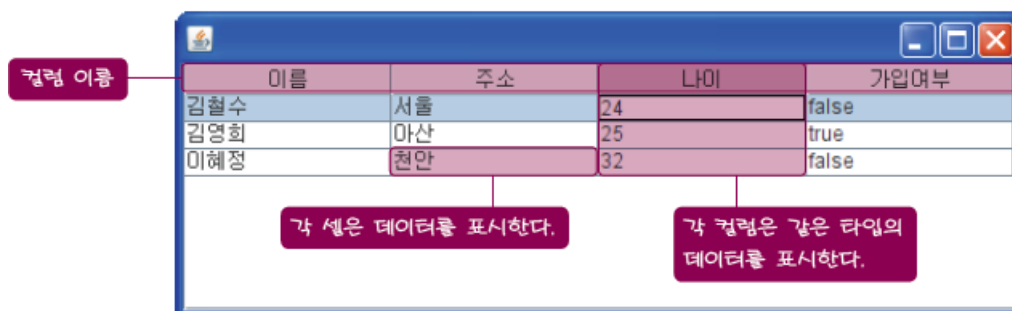
    public void actionPerformed(ActionEvent e) {
        dispose(); // 대화상자를 제거
    }

    class MyWinListener extends WindowAdapter { // 이벤트 처리
        public void windowClosing(WindowEvent e) {
            dispose(); // 대화상자를 제거
        }
    }
}

```

10. 테이블

테이블(table)은 스프레드 시트처럼 데이터를 테이블 형식으로 표시하고 사용자가 값을 편집할 수 있는 컴포넌트이다. JTable 클래스는 데이터를 저장하지는 않고 데이터를 보여주는 뷰를 제공한다.



1) 테이블 생성하기

먼저 테이블 컬럼의 이름을 String 배열에 정의하여야 한다.

```
String[] columnNames = {"이름", "주소", "나이", "가입여부"};
```


데이터는 일반적으로 2차원 Object 배열에 저장된다.

```
Object[][] data = {    {"김철수", "서울", new Integer(24), new Boolean(false)},
                      {"김영희", "아산", new Integer(25), new Boolean(true)},
                      {"이혜정", "천안", new Integer(32), new Boolean(false)}  };
```

테이블은 컬럼 이름과 데이터를 사용하여 생성된다.

```
JTable table = new JTable(data, columnNames);
```

2) 테이블을 컨테이너에 추가하기

테이블에는 보통 스크롤바가 붙어야 실용적이다. 스크롤바를 테이블에 추가하려면 테이블을 스크롤 페인 안에 추가하여야 한다.

```
JScrollPane scrollPane = new JScrollPane(table);
table.setFillViewportHeight(true);
```

먼저 JScrollPane 생성자를 호출한다. 매개 변수는 테이블을 참조하는 변수가 된다. 이 스크롤 페인은 테이블의 컨테이너가 된다. setFillViewportHeight()은 컨테이너의 전체 높이를 테이블이 전부 사용하게끔 설정한다.

3) 이벤트 처리

테이블 각 셀의 데이터가 변경되면 TableModelListener 리스너를 이용하여서 감지할 수 있다. 다음의 코드에서는 리스너를 통하여 변경된 셀의 데이터를 얻는다.

```
import javax.swing.event.*;
import javax.swing.table.TableModel;

public class TableTest extends JFrame implements TableModelListener {
    public TableTest() {
        ...
        final JTable table = new JTable(data, columnNames);
        table.getModel().addTableModelListener(this);
    }
    public void tableChanged(TableModelEvent e) {
        int row = e.getFirstRow();
        int column = e.getColumn();
        TableModel model = (TableModel)e.getSource();
        String columnName = model.getColumnName(column);
        Object data = model.getValueAt(row, column);
        // 데이터를 처리한다
        ...
    }
}
```

4) 정렬과 필터링

테이블의 정렬과 필터링은 정렬기 객체에 의하여 수행된다.

가장 쉬운 방법은 테이블의 `autoCreateRowSorter` 특성을 `true`로 설정하는 것이다.

```
JTable table = new JTable();  
table.setAutoCreateRowSorter(true);
```

행을 정렬하는 정렬키를 정의한다. 사용자가 컬럼 헤더를 클릭하면 정렬이 수행된다.

5) 콤보 박스를 셀 편집기로 사용하는 방법

테이블의 경우, 콤보 박스를 통하여 셀의 값을 변경하는 것이 필요한 경우도 있다. 이 경우에는 콤보 박스를 셀 편집기로 설정하면 된다.

```
TableColumn hobbyColumn = table.getColumnModel().getColumn(2);  
...  
JComboBox comboBox = new JComboBox();  
comboBox.addItem("독서");  
comboBox.addItem("골프");  
...  
// 특정 컬럼의 편집기를 콤보 박스로 설정한다  
hobbyColumn.setCellEditor(new DefaultCellEditor(comboBox));
```

6) 테이블을 프린터로 출력하기

JTable은 테이블을 출력하기 위한 간단한 API를 제공한다. 가장 간단한 방법은 `JTable.print()` 메소드를 호출하는 것이다.

```
try {  
    if (! table.print()) {  
        System.err.println("사용자가 인쇄를 취소하였음");  
    }  
} catch (java.awt.print.PrinterException e) {  
    System.err.format("출력 도중 오류 발생: %s\n", e.getMessage());  
}
```

(예제)

테이블을 이용하여서 회원 가입하는 폼을 작성하였다. 주소 입력시에는 콤보 박스를 사용하도록 하고 나이를 변경할 때는 이벤트 처리를 통하여 이상한 값이 입력되지 않도록 하였다. 컬럼 헤더를 클릭하면 자동으로 정렬이 수행되도록 설정하였다.

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

```

import javax.swing.table.TableModel;

public class SimpleTableTest extends JFrame implements TableModelListener {

    public SimpleTableTest() {
        String[] columnNames = { "이름", "주소", "나이", "가입여부" };
        Object[][] data = {
            { "김철수", "서울", new Integer(24), new Boolean(false) },
            { "김영희", "부산", new Integer(25), new Boolean(true) },
            { "이혜정", "천안", new Integer(32), new Boolean(false) } };

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500, 200);
        final JTable table = new JTable(data, columnNames);

        table.setPreferredScrollableViewportSize(new Dimension(500, 150));
        table.setFillsViewportHeight(true);

        // 셀이 변경되면 이벤트 처리
        table.getModel().addTableModelListener(this);
        // 컬럼 헤더를 클릭하면 자동 정렬
        table.setAutoCreateRowSorter(true);

        // 주소를 입력할 때 지정된 도시만 입력이 가능하도록 편집기 설정
        TableColumn cityColumn = table.getColumnModel().getColumn(1);
        JComboBox comboBox = new JComboBox();
        comboBox.addItem("서울");
        comboBox.addItem("부산");
        comboBox.addItem("광주");
        comboBox.addItem("대구");
        comboBox.addItem("대전");
        comboBox.addItem("천안");
        cityColumn.setCellEditor(new DefaultCellEditor(comboBox));

        // 스크롤 페인을 설정하고 테이블을 추가
        JScrollPane scrollPane = new JScrollPane(table);

        // 패널을 스크롤 페인에 추가
        add(scrollPane);
        setVisible(true);
    }
}

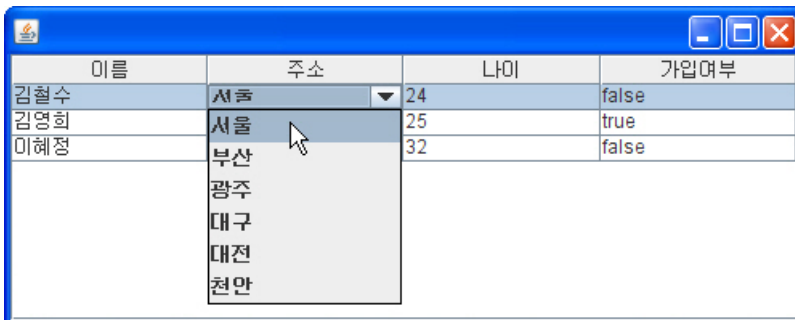
```

```

public void tableChanged(TableModelEvent e) {
    int row = e.getFirstRow();
    int column = e.getColumn();
    if (column == 2) {
        TableModel model = (TableModel) e.getSource();
        String columnName = model.getColumnName(column);
        Object data = model.getValueAt(row, column);
        String s = (String) data;
        if (Integer.parseInt(s) > 100)
            JOptionPane.showMessageDialog(this, "범위를 초과하였습니다: " + s, "경고",
                                           JOptionPane.WARNING_MESSAGE);
    }
}

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

```



이름	주소	나이	가입여부
김철수	서울	24	false
김영희	서울	25	true
미혜정	부산	32	false

9. 전화번호 프로그램 9단계

SWING 예제 모음

JLabel 클래스

GUI 프로그램에 대한 정보 또는 텍스트를 생성하기 위한 레이블을 생성하는 컴포넌트이다.

```
JLabel jl = new JLabel("연습");
```

JButton 클래스

버튼을 생성하는 컴포넌트이다

```
JButton jb = new JButton ("버튼");
```

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

public class JLabelJButtonTest extends JFrame {
    public JLabelJButtonTest(){
        super("레이블과 버튼연습");
        Container c = getContentPane();
        c.setLayout(new FlowLayout());

        JLabel jl = new JLabel("이것이 라벨이다");
        c.add(jl);

        JButton b1 = new JButton("버튼1");
        JButton b2 = new JButton("버튼2");
        JButton b3 = new JButton("버튼3");

        c.add(b1);
        c.add(b2);
        c.add(b3);

        setSize(400,250);
        setVisible(true);
    }

    public static void main(String[] args) {
        JLabelJButtonTest f = new JLabelJButtonTest();
        f.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

JTextField 클래스

JTextField는 문자열을 입력할 수 있는 컴포넌트이다.

```
JTextField jb = new JTextField ();
```

JPasswordField 클래스

JPasswordField는 화면에 입력한 문자를 '*'로 대치시켜 내용을 보호하는 컴포넌트이다.

```
JPasswordField jb = new JPasswordField();
```

JTextArea 클래스

JTextArea는 여러 줄의 문자열을 표시할 수 있는 컴포넌트이다. 스윙 컴포넌트인 JTextArea는 자동으로 스크롤바가 생기지 않는다. 스크롤바를 붙일 때는 JScrollPane클래스 객체를 사용해서 스크롤바를 붙여 준다.

```
JTextArea content = new JTextArea(3,20);
```

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

public class JComponentTest extends JFrame {
    JLabel label1, label2, la1, la2;
    JTextField id;
    JPasswordField passwd;
    JPanel idPanel, paPanel, loginPanel;
    JButton b1, b2;
    JTextArea content;
    public JComponentTest() //컴포넌트의 생성과 배치
    {
        super( "JComponent테스트" );

        Container c = getContentPane();
        c.setLayout( new FlowLayout() );
        EtchedBorder eborder = new EtchedBorder();
        Icon bug = new ImageIcon("test.jpg"); //이미지 파일 필요

        label1 = new JLabel( "아이디와 패스워드를 입력해 주세요" );

        label2 = new JLabel( bug);
        label2.setBorder(eborder);
        label2.setToolTipText("라벨2");
    }
}
```

```

c.add( label2 );
c.add( label1 );

idPanel = new JPanel();
la1 = new JLabel("아이디");
id = new JTextField(10);
idPanel.add( la1 );
idPanel.add( id );

paPanel = new JPanel();
la2 = new JLabel("패스워드");
passwd = new JPasswordField(10);
paPanel.add( la2 );
paPanel.add( passwd );

loginPanel = new JPanel();
b1 = new JButton("로그인");
b2 = new JButton("회원가입");
loginPanel.add( b1 );
loginPanel.add( b2 );

c.add(idPanel);
c.add(paPanel);
c.add(loginPanel);

content = new JTextArea(3,20);
JScrollPane s= new JScrollPane(content);
c.add(s);
setSize( 250, 350 );
setVisible(true);
}

public static void main( String args[] )
{
    JComponentTest la = new JComponentTest();
    la.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```

레이아웃 매니저 1 (Layout Manager)

복잡한 형태의 컴포넌트는 중간컨테이너인 JPanel을 사용하면 해결한다.

컴포넌트를 어떤 형태로 부착 시킬 것인가는 레이아웃 매니저를 사용해서 처리한다.

플로우 레이아웃(FlowLayout)

플로우 레이아웃은 컨테이너에 컴포넌트를 차례로 왼쪽에서 오른쪽으로 추가시킨다. 한 줄에 표시하기에 컴포넌트가 많으면 자연스럽게 다음 줄에 추가된다. 플로우 레이아웃은 기본적으로 컨테이너가 중앙 정렬 된다. 컨테이너중 AWT의 Applet과 , Panel과 스윙의 JPanel의 기본 레이아웃 매니저는 FlowLayout 이다.

보더 레이아웃(BorderLayout)

보더 레이아웃은 컨테이너에 컴포넌트를 부착시킬 위치가 이미 정해져 있는것으로 East/West/South/North/Center 이 다섯 부분에만 컴포넌트가 부착될 수 있다. JFrame과 JApplet의 기본 레이아웃 매니저는 BorderLayout이다.

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

public class BorderLayoutTest extends JFrame {
    public BorderLayoutTest() {
        super("BorderLayout 연습");
        Container c = getContentPane();
        c.add(new JButton("North"), BorderLayout.NORTH);
        c.add(new JButton("Center"), BorderLayout.CENTER);
        c.add(new JButton("South"), BorderLayout.SOUTH);
        c.add(new JButton("West"), BorderLayout.WEST);
        c.add(new JButton("East"), BorderLayout.EAST);
        setSize(300, 200);
        setVisible(true);
    }

    public static void main(String args[]) {
        BorderLayoutTest border = new BorderLayoutTest();
        border.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

그리드 레이아웃(GridLayout)

그리드 레이아웃은 컨테이너가 일률적으로 정해진 같은 크기로 컴포넌트를 추가시킨다. 행렬의 형태를 가지며, 왼쪽부터 열의 개수만큼 컴포넌트가 추가되며, 한행이 모두 추가되면 다음 행의 에 추가된다.


```

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

public class GridLayoutTest extends JFrame {
    String num[]={"7","8","9","*","4","5","6","/","1","2","3","+","0",".", "=", "-"};
    JButton b[];
    public GridLayoutTest() {
        super("GridLayout 연습");
        Container c = getContentPane();
        c.setLayout(new GridLayout(4,4));
        b= new JButton[16];
        for(int i=0;i<16;i++){
            b[i]=new JButton(num[i]);
            c.add(b[i]);
        }
        setSize(200, 200);
        setVisible(true);
    }

    public static void main(String args[]) {
        GridLayoutTest grid = new GridLayoutTest();
        grid.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

```

JCheckBox 클래스

체크박스는 JCheckBox클래스로 객체를 생성한다

```

JCheckBox c1 = new JCheckBox( "잠자기" );
JCheckBox c2 = new JCheckBox( "TV시청" );
JCheckBox c3 = new JCheckBox( "만화보기" );

```

JRadioButton 클래스

라디오버튼은 JRadioButton 클래스를 사용해서 객체를 생성한다. JRadioButton은 ButtonGroup을 이용해서 논리적 그룹화를 해야 한다.

```

JRadioButton r1 = new JRadioButton( "남자" ,true);
JRadioButton r2 = new JRadioButton( "여자" ,false);
ButtonGroup bg = new ButtonGroup();
bg.add(r1);
bg.add(r2);

```

JList 클래스

JList는 리스트박스를 생성하는 컴포넌트로, setVisibleRowCount(5)를 사용해서 표시하고 싶은 리스트의 개수를 표시한다. 단일 선택 또는 복수 선택은 setSelectionMode()메소드를 사용해서 한다.

```
JList jl = new JList(sports);
```

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

public class JComponentTest2 extends JFrame {
    JCheckBox c1, c2, c3;
    JPanel p,p1,p2;
    JRadioButton r1, r2;
    ButtonGroup bg;
    JComboBox com;
    JList jl;
    String major[]={"컴퓨터공학과", "기계공학과", "전자공학과", "제어계측학과", "산업공학과"};
    String sports[]={"농구", "야구", "축구", "배구", "격투기"};

    public JComponentTest2(){
        super( "JComponent연습" );

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

        p=new JPanel(new GridLayout(1,2));
        p1= new JPanel(new GridLayout(3,1));
        c1 = new JCheckBox( "잠자기" );
        c2 = new JCheckBox( "TV시청" );
        c3 = new JCheckBox( "만화보기" );
        p1.add( c1 );
        p1.add( c2 );
        p1.add( c3 );
        p.add(p1);

        p2= new JPanel(new GridLayout(2,1));
        r1 = new JRadioButton( "남자" ,true);
        r2 = new JRadioButton( "여자" ,false);
        p2.add( r1 );
        p2.add( r2 );
        p.add(p2);
    }
}
```

```

        bg = new ButtonGroup();
        bg.add(r1);
        bg.add(r2);

        c.add( p);

        com = new JComboBox(major);
        JScrollPane s= new JScrollPane(com);
        c.add( s );

        jl = new JList(sports );
        JScrollPane s1= new JScrollPane(jl);
        jl.setVisibleRowCount(4);
        c.add( s1 );

        setSize( 250, 300 );
        setVisible(true);
    }

    public static void main( String args[] )
    {
        JComponentTest2 jc = new JComponentTest2();
        jc.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

```

레이아웃 매니저 2 (Layout Manager)

카드 레이아웃(CardLayout)

카드 레이아웃(CardLayout)은 컴포넌트들을 카드를 여러장이 겹쳐있는 형태 배치한다. 각각의 카드에는 하나의 컴포넌트만을 넣을 수 있다.

```

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

public class CardLayoutTest extends JFrame implements ActionListener{
    CardLayout cardManager;
    JPanel deck;
    JButton b[];
    Container c;
    int k=0;
}

```

```

public CardLayoutTest(){
    super( "CardLayout 연습" );
    String face[]={"Hearts","Diamonds","Clubs","Spades"};
    String num[]={"Ace","Deuce","Three","Four","Five","Six","Seven","Eight","Nine","Ten","Jack","Queen","King"};
    b=new JButton[52];
    c = getContentPane();
    cardManager=new CardLayout();
    c.setLayout(cardManager);
    //JButton의 객체를 생성하고, 이벤트 리스너에 등록한다.
    //카드 매니저에 JButton의 객체를 추가한다.

    for(int i=0;i<face.length;i++){
        for(int j=0;j<num.length;j++){
            b[k]=new JButton(new ImageIcon("images/"+num[j]+"of"+face[i]+".gif"));
            b[k].addActionListener(this); //JButton의 객체를 ActionListener에 등록한다.
            c.add(b[k],"a"+i); // JButton의 객체를 카드 매니저에 등록한다.
            k++;
        }
    }
    setSize(50,150);
    setVisible(true);
}

public void actionPerformed(ActionEvent e) //이벤트를 처리하는 메소드
{
    cardManager.next(c);
}

public static void main(String args[])
{
    CardLayoutTest card = new CardLayoutTest();
    card.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```

박스 레이아웃(BoxLayout)

컴포넌트들이 왼쪽에서 오른쪽으로 수평 배치되거나, 위에서 아래로 수직 배치가 되도록 해주는 레이아웃관리자다.

박스 레이아웃은 컴포넌트들 간에 수직 혹은 수평공간을 확보하기 위해 Strut 과 Glue를 제공한다.

Strut은 컴포넌트가 부착되어있는 컨테이너의 크기가 변하더라도 절대 크기가 변하지 않고, Glue는 변한

다. Strut 과 Glue는 일종의 투명컴포넌트로서 자리만 차지하고, 보이지는 않는다

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

public class BoxLayoutTest extends JFrame
{
    JRadioButton r1,r2,r3,r4;

    public BoxLayoutTest(){
        super( "BoxLayout연습예제" );
        Container c = getContentPane();
        c.setLayout(new BorderLayout());

        Box left = Box.createVerticalBox();
        left.add(Box.createVerticalStrut(30));
        ButtonGroup radioGroup = new ButtonGroup();
        radioGroup.add(r1 = new JRadioButton("Red"));
        left.add(r1);
        left.add(Box.createVerticalStrut(30));
        radioGroup.add(r2 = new JRadioButton("Black"));
        left.add(r2);
        left.add(Box.createVerticalStrut(30));
        radioGroup.add(r3 = new JRadioButton("White"));
        left.add(r3);
        left.add(Box.createVerticalStrut(30));
        radioGroup.add(r4 = new JRadioButton("Yellow"));
        left.add(r4);
        left.add(Box.createGlue());

        JPanel leftPanel = new JPanel(new BorderLayout());
        leftPanel.setBorder(new TitledBorder( new EtchedBorder(),"Car Color"));
        leftPanel.add(left, BorderLayout.CENTER);

        Box right = Box.createVerticalBox();
        right.add(Box.createVerticalStrut(30));
        right.add(new JCheckBox("스타렉스"));
        right.add(Box.createVerticalStrut(30));
        right.add(new JCheckBox("소나타 EF"));
        right.add(Box.createVerticalStrut(30));
    }
}
```

```

right.add(new JCheckBox("투스카니"));
right.add(Box.createVerticalStrut(30));
right.add(new JCheckBox("산타페"));
right.add(Box.createGlue());

JPanel rightPanel = new JPanel(new BorderLayout());
rightPanel.setBorder(new TitledBorder(new EtchedBorder(), "Car Name"));
rightPanel.add(right, BorderLayout.CENTER);

Box center = Box.createHorizontalBox();
center.add(leftPanel);
center.add(rightPanel);
c.add(center, BorderLayout.CENTER);
setSize(300,300);
setVisible(true);
}

public static void main(String[] args) {
    BoxLayoutTest box = new BoxLayoutTest();
    box.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```

메뉴사용하기(JMenuBar, JMenu, JMenuItem)

메뉴(Menu)는 setMenuBar()메소드를 제공하는 컨테이너의 객체에만 사용할 수 있는데, 이것이 가능한 컨테이너는 최종 컨테이너로 사용되는 JFrame 과 JApplet이다.

메뉴를 정의해서 사용하기 위해서는 4단계의 과정이 필요하다.

1단계 JMenuBar생성해서, 3단계에서 생성한 JMenu를 JMenuBar에 추가시킨다.

JMenuBar bar = new JMenuBar();

2단계 setJMenuBar() 사용해서 컨테이너에서 메뉴바를 사용할 수 있게한다.

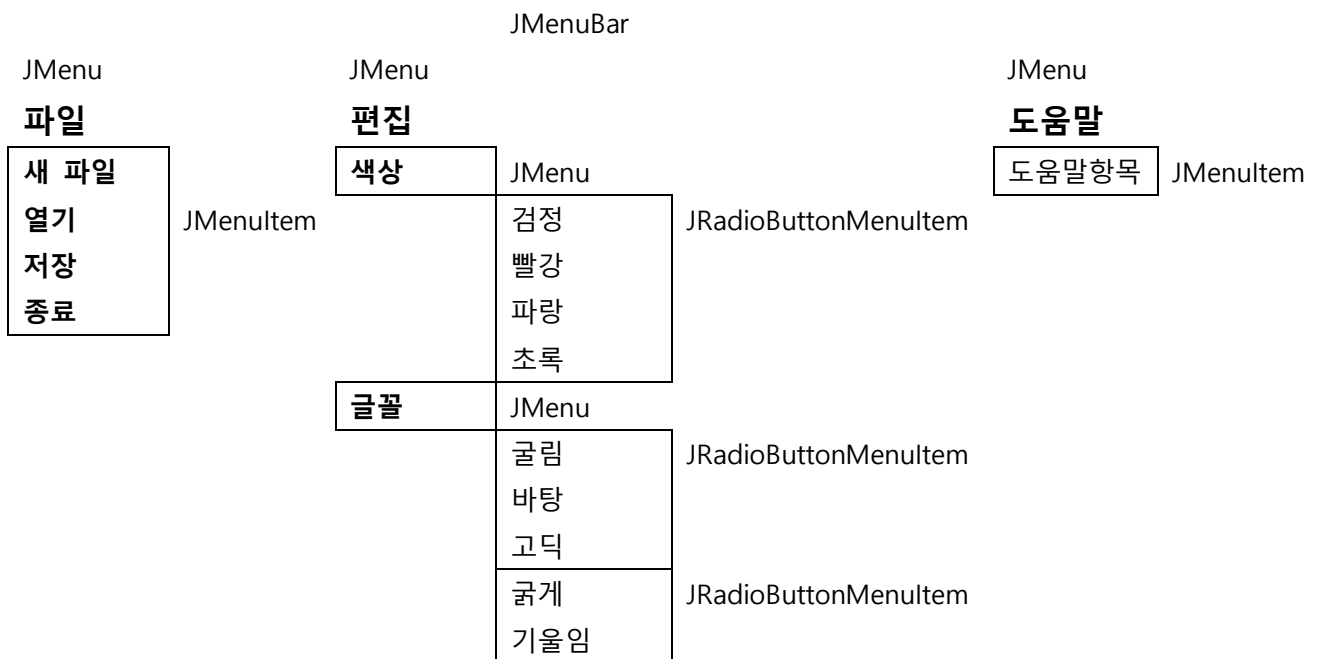
setJMenuBar(bar);

3단계 JMenu생성해서 유사한JMenuItem 그룹을 관리한다.

JMenu fileMenu = new JMenu("파일");

4단계 JMenuItem , JCheckBoxMenuItem 혹은 JRadioButtonMenuItem등의 가장 하위단계의 메뉴의 항목을 생성한다.

JMenuItem newItem = new JMenuItem("새파일");



```
import javax.swing.*;

public class JMenuTest extends JFrame {
    private JRadioButtonMenuItem colorItems[], fonts[];
    private JCheckBoxMenuItem styleItems[];
    private ButtonGroup fontGroup, colorGroup;
```

```

public JMenuTest()
{
    super( "JMenu연습예제" );

    JMenuBar bar = new JMenuBar(); //1단계
    setJMenuBar( bar );           //2단계
    //[파일] 메뉴 부분
    JMenu fileMenu = new JMenu( "파일(F)" ); //3단계
    fileMenu.setMnemonic( 'F' ); //단축키 : Alt + F

    JMenuItem newItem = new JMenuItem( "새파일(N)" ); //4단계
    newItem.setMnemonic( 'N' );
    fileMenu.add( newItem ); //JMenuItem를 JMenu에 부착

    JMenuItem openItem = new JMenuItem( "열기(O)" ); //4단계
    openItem.setMnemonic( 'O' );
    fileMenu.add( openItem ); //JMenuItem를 JMenu에 부착

    JMenuItem saveItem = new JMenuItem( "저장(S)" ); //4단계
    saveItem.setMnemonic( 'S' );
    fileMenu.add( saveItem ); //JMenuItem를 JMenu에 부착

    JMenuItem exitItem = new JMenuItem( "종료(X)" ); //4단계
    exitItem.setMnemonic( 'X' );
    fileMenu.add( exitItem ); //JMenuItem를 JMenu에 부착
    bar.add( fileMenu );      // JMenu를 JMenuBar에 부착

    //[편집]메뉴 부분
    JMenu formatMenu = new JMenu( "편집(E)" );
    formatMenu.setMnemonic( 'E' );

    //하위 메뉴인 [색상] 메뉴 부분
    String colors[] = { "검정", "파랑", "빨강", "초록" };
    JMenu colorMenu = new JMenu( "색상(C)" );
    colorMenu.setMnemonic( 'C' );
    colorItems = new JRadioButtonMenuItem[ colors.length ];
    colorGroup = new ButtonGroup();

    for ( int i = 0; i < colors.length; i++ ) {
        colorItems[ i ] =
            new JRadioButtonMenuItem( colors[ i ] );
    }
}

```



```

        colorMenu.add( colorItems[ i ] );
        colorGroup.add( colorItems[ i ] );
    }

    colorItems[ 0 ].setSelected( true ); //첫 번째 항목을 기본 선택 값으로 설정
    formatMenu.add( colorMenu );
    formatMenu.addSeparator(); //메뉴의 구분선

    //하위 메뉴인 [글꼴] 메뉴 부분
    String fontNames[] = { "굴림", "바탕", "고딕" };
    JMenu fontMenu = new JMenu( "글꼴(T)" );
    fontMenu.setMnemonic( 'T' );
    fonts = new JRadioButtonMenuItem[ fontNames.length ];
    fontGroup = new ButtonGroup();

    for ( int i = 0; i < fonts.length; i++ ) {
        fonts[ i ] =
            new JRadioButtonMenuItem( fontNames[ i ] );
        fontMenu.add( fonts[ i ] );
        fontGroup.add( fonts[ i ] );
    }

    fonts[ 0 ].setSelected( true );
    fontMenu.addSeparator();

    String styleNames[] = { "굵게", "기울임" };
    styleItems = new JCheckBoxMenuItem[ styleNames.length ];

    for ( int i = 0; i < styleNames.length; i++ ) {
        styleItems[ i ] = new JCheckBoxMenuItem( styleNames[ i ] );
        fontMenu.add( styleItems[ i ] );
    }

    formatMenu.add( fontMenu );
    bar.add( formatMenu ); // JMenu을 JMenuBar에 부착

    JMenu helpMenu = new JMenu( "도움말(H)" );
    helpMenu.setMnemonic( 'H' );

    JMenuItem helpItem = new JMenuItem( "도움말항목(H)" );
    helpItem.setMnemonic( 'L' );

```

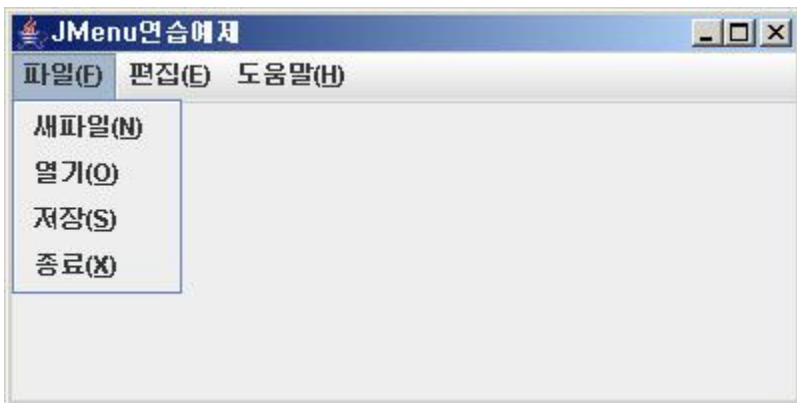
```

        helpMenu.add(helpItem);
        bar.add( helpMenu );    // JMenu을 JMenuBar에 부착

        setSize( 400, 200 );
        setVisible(true);
    }

    public static void main( String args[] )
    {
        JMenuTest jmenu = new JMenuTest();
        jmenu.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

```



팝업메뉴(JPopupMenu)

JPopupMenu는 GUI프로그램에서 제공되는 팝업메뉴를 생성한다. 팝업메뉴는 오른쪽 마우스버튼을 누르거나(mousePressed) 해제(mouseReleased)될 때 수행된다.

```
PopupMenu popupMenu = new JPopupMenu();
```

```

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

public class JPopupMenuTest extends JFrame {
    JRadioButtonMenuItem items[];
    public JPopupMenuTest()
    {
        super( "JPopupMenu연습예제" );

        final JPopupMenu popupMenu = new JPopupMenu();
        String colors[] = { "파랑", "초록", "빨강" };
        ButtonGroup colorGroup = new ButtonGroup();
    }
}

```



```

items = new JRadioButtonMenuItem[ 3 ];

for ( int i = 0; i < items.length; i++ ) {
    items[ i ] = new JRadioButtonMenuItem( colors[ i ] );
    popupMenu.add( items[ i ] );
    colorGroup.add( items[ i ] );
}

addMouseListener( // 마우스 이벤트를 리스너에 등록
    new MouseAdapter() {
        public void mousePressed( MouseEvent e ) //마우스를 누를 때
            { checkForTriggerEvent( e ); }

        public void mouseReleased( MouseEvent e ) //마우스를 놓을 때
            { checkForTriggerEvent( e ); }

        private void checkForTriggerEvent( MouseEvent e ){
            if ( e.isPopupTrigger() ) //마우스오른쪽버튼을 누르거나 해제할 때 발생
                popupMenu.show( e.getComponent(), e.getX(), e.getY() );
        }
    }
);

setSize( 300, 200 );
setVisible(true);
}

public static void main( String args[] )
{
    JPopupTest jpopup = new JPopupTest();
    jpopup.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```

인터널 프레임(JInternalFrame)

GUI프로그램에서는 보다 발전된 형태의 GUI를 제공하기위해서는 부모 윈도우안에 별도의 자식윈도우를 갖는MDI(Multiple Data Interface)를 사용한다.

자바에서는 JInternalFrame을 제공한다. JInternalFrame은 혼자 동작할 수 없고 반드시 JDesktopPane에 JInternalFrame을 추가시켜야 자식 프레임인 JInternalFrame이 일반프레임처럼 작동된다.

1. JDesktopPane클래스의 객체를 생성한다.

```
JDesktopPane desktop = new JDesktopPane();
```

2. JInternalFrame클래스의 객체생성한다.

```
JInternalFrame iframe = new JInternalFrame("내부프레임", true, true, true, true);
```

```
JInternalFrame(String title, boolean resize, boolean closable, boolean maximizable, boolean iconifiable);
```

3. JDesktopPane클래스의 객체에 JInternalFrame클래스의 객체를 추가한다.

```
desktop.add(iframe);
```

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

```
public class JInternalFrameTest extends JFrame{
```

```
    JInternalFrame iframe;
```

```
    JLabel i;
```

```
    ImageIcon icon;
```

```
    public JInternalFrameTest(){
```

```
        super("JInternalFrame 연습예제");
```

```
        JMenuBar bar = new JMenuBar();
```

```
        setJMenuBar( bar );
```

```
        JMenu addMenu = new JMenu( "추가" );
```

```
        JMenuItem newInFrame = new JMenuItem( "내부프레임 추가" );
```

```
        addMenu.add(newInFrame);
```

```
        bar.add(addMenu);
```

```
        final JDesktopPane desktop = new JDesktopPane(); // JDesktopPane클래스의 객체를 생성
        getContentPane().add(desktop, BorderLayout.CENTER);
```

```
        newInFrame.addActionListener( //[내부 프레임추가] 메뉴를 클릭하면 수행된다.
```

```
            new ActionListener(){
```

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

```
                    iframe = new JInternalFrame("내부프레임", true, true, true, true);
```

```
                    //JInternalFrame 클래스의 객체를 생성
```

```
                    icon = new ImageIcon("testImage.jpg");
```

```
                    i=new JLabel(icon);
```

```
                    iframe.getContentPane().add(i, BorderLayout.CENTER);
```

```
                    iframe.setSize(200,100);
```



```

        inframe.show();
        desktop.add(inframe);
        // JDesktopPane클래스의 객체에 JInternalFrame클래스의 개체를 추가한다.
    });
    setSize(300,200);
    setVisible(true);
}
public static void main(String args[]){
    JInternalFrameTest jinter = new JInternalFrameTest();
    jinter.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```

테이블(JTable)

JTable클래스는 Table형태로 자료를 표현하고자 할 때 사용된다.

```
JTable table = new JTable(data, title);
```

```

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

public class JTableTest extends JFrame{
    public JTableTest(){
        super("JTable 예제");
        String title[] = {"번호", "이름", "입사일"};
        String data[][] = {
            {"1", "이문세", "2000-03-10"},
            {"2", "김재동", "2001-10-07"},
            {"3", "홍길동", "2002-05-20"},
            {"4", "장동건", "2004-12-22"},
            {"5", "연개소문", "2005-07-05"},
            {"6", "대조영", "2006-04-11"},
            {"7", "연개소문", "2007-01-15"},
            {"8", "대조영", "2007-04-13"}
        };

        JTable table = new JTable(data, title);
        JScrollPane sp = new JScrollPane(table);
        getContentPane().add(sp, BorderLayout.CENTER);
    }
}

```

```

        setSize(300,150);
        setVisible(true);
    }

    public static void main(String args[]){
        JTableTest jtable = new JTableTest();
        jtable.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

```

번호	이름	입사일
1	이문세	2000-03-10
2	김재동	2001-10-07
3	홍길동	2002-05-20
4	장동건	2004-12-22
5	연개소문	2005-07-05
6	대조영	2006-04-11
7	연개소문	2007-04-11

탭판(JTabbedPane)

탭(Tab)을 자바에서 제공하는 것이 JTabbedPane 컴포넌트이다. JTabbedPane은 객체를 생성할 때 탭의 위치를 JTabbedPane.TOP , JTabbedPane.BOTTOM, JTabbedPane.LEFT, JTabbedPane.RIGHT로 지정할 수 있다.

탭판 작성

1. JTabbedPane 객체생성

```
JTabbedPane tab = new JTabbedPane(JTabbedPane.TOP); // JTabbedPane 객체 생성
```

2. JTabbedPane 객체에 컴포넌트 추가

```
Tab.addTab("인사관리(기본)", one);
```

```
Tab.addTab("인사관리(세부)", two);
```

```

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

public class JTabbedPaneTest extends JFrame{
    JTabbedPane tab;
    JTable1 j1;
    JTable2 j2;
    public JTabbedPaneTest(){
        super("JTabbedPane 연습예제");
        tab = new JTabbedPane(JTabbedPane.TOP);
        // tab = new JTabbedPane(JTabbedPane.BOTTOM);
        // tab = new JTabbedPane(JTabbedPane.LEFT);
    }
}

```

```

        // tab = new JTabbedPane(JTabbedPane.RIGHT);

        JPanel one = new JPanel();
        j1 = new JTable1();
        JPanel two = new JPanel();
        j2 = new JTable2();

        one.add(j1);
        two.add(j2);

        tab.addTab("인사관리(기본)", one);
        tab.addTab("인사관리(세부)", two);

        getContentPane().add(tab, BorderLayout.CENTER);
        setSize(500,200);
        setVisible(true);
    }

    public static void main(String args[]){
        JTabbedPaneTest jt= new JTabbedPaneTest();
        jt.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

class JTable1 extends JPanel{
    public JTable1(){
        String title[] = {"번호", "이름", "입사일"};
        String data[][] = {
            {"1","김개동", "2000-03-10"},
            {"2","김은옥", "2001-10-07"},
            {"3","홍길동", "2002-05-20"}
        };

        JTable table = new JTable(data, title);
        JScrollPane sp = new JScrollPane(table);
        add(sp);
    }
}

class JTable2 extends JPanel {

```

```

public JTable2(){
    String title[] = {"번호", "부서", "직급"};
    String data[][] = {
        {"1","기획부", "과장"},
        {"2","홍보부", "대리"},
        {"3","개발부", "대리"}
    };

    JTable table = new JTable(data, title);
    JScrollPane sp = new JScrollPane(table);
    add(sp);
}
}

```



룩앤필

룩앤필(Look-and-Feel)이란 GUI프로그램의 컴포넌트들이 프로그램이 수행되는 해당 플랫폼에 따라 그 모양이 약간씩 다르게 보이는 것을 말한다. 기존의 AWT에서는 룩앤필을 무시했었다. 그결과로 여러 가지 기능성문제가 발생하여서 Swing에서는 플랫폼에 따른 룩앤필(Look-and-Feel)을 정의 가능하도록 해서 이런 문제를 제거하였다.

```

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

public class LookAndFeelTest extends JFrame {
    private String strings[] = { "메탈(자바기본)", "모티브(UNIX)", "윈도우즈" };
    private UIManager.LookAndFeelInfo looks[];
    private JRadioButton radio[];
    private ButtonGroup group;
    private JButton button;
    private JLabel label;
}

```



```

public LookAndFeelTest()
{
    super("Look-and-Feel 확인예제");
    Container c = getContentPane();

    JPanel northPanel = new JPanel();
    northPanel.setLayout( new GridLayout( 3, 1, 0, 5 ) );
    label = new JLabel("메탈(자바기본) look-and-feel", SwingConstants.CENTER );
    northPanel.add( label );
    button = new JButton("컴포넌트의 모양 - JButton");
    northPanel.add( button );

    c.add( northPanel, BorderLayout.CENTER );

    JPanel southPanel = new JPanel();
    radio = new JRadioButton[ strings.length ];
    group = new ButtonGroup();
    ItemHandler handler = new ItemHandler();
    southPanel.setLayout( new GridLayout( 1, radio.length ) );

    for ( int i = 0; i < radio.length; i++ ) {
        radio[ i ] = new JRadioButton( strings[ i ] );
        radio[ i ].addItemListener( handler );
        group.add( radio[ i ] );
        southPanel.add( radio[ i ] );
    }

    c.add( southPanel, BorderLayout.SOUTH );
    looks = UIManager.getInstalledLookAndFeels();
    setSize( 350, 200 );
    setVisible(true);
    radio[ 0 ].setSelected( true );
}

private void changeTheLookAndFeel( int value ) //룩앤필을 변경하는 메소드
{
    try {
        UIManager.setLookAndFeel(looks[ value ].getClassName() ); //룩앤필 변경
        SwingUtilities.updateComponentTreeUI( this ); //룩앤필에 따른 컴포넌트 변경
    }
}

```

```

        catch ( Exception e ) {
            e.printStackTrace();
        }
    }

    public static void main( String args[] )
    {
        LookAndFeelTest dx = new LookAndFeelTest();
        dx.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private class ItemHandler implements ItemListener {
        public void itemStateChanged( ItemEvent e ) //라디오 버튼을 선택할 때마다 호출된다.
        {
            for ( int i = 0; i < radio.length; i++ )
                if ( radio[ i ].isSelected() ) {
                    label.setText( strings[ i ] + " look-and-feel" );
                    changeTheLookAndFeel( i ); //선택한 라디오 버튼에 따라 해당 값을 가지고
                                                // changeTheLookAndFeel() 메소드 호출
                }
            }
        }
    }
}

```



ActionEvent

주로 버튼을 클릭하거나 텍스트 필드에서 enter키를 누를 때 또는 메뉴를 선택할 때 발생한다.

ActionEvent는 **ActionListener**인터페이스를 implements받아 사용하는데 ActionListener가 가지고 있는 메소드인 **actionPerformed(ActionEvent e)**메소드를 반드시 오버라이딩 해야 한다.

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

public class ActionEventTest extends JFrame implements ActionListener {
    JLabel la1;
    JTextField in;
    JPanel inputPanel;
    JButton send;
    JTextArea content;

    public ActionEventTest()
    {
        super( "텍스트 필트와 버튼에서 ActionEvent 연습" );
        Container c = getContentPane();
        c.setLayout( new BorderLayout() );

        inputPanel = new JPanel();
        la1 = new JLabel("내용");
        in = new JTextField(20);
        in.addActionListener(this); //이벤트가 발생하면 actionPerformed()메소드를 수행
        send = new JButton("전송");
        send.addActionListener(this); //이벤트가 발생하면 actionPerformed()메소드를 수행
        inputPanel.add( la1 );
        inputPanel.add( in );
        inputPanel.add( send );

        content = new JTextArea(5,30);
        JScrollPane s= new JScrollPane(content);
        c.add(inputPanel, BorderLayout.SOUTH);
        c.add(s, BorderLayout.CENTER);
        setSize( 350, 200 );
        setVisible(true);
        in.requestFocus();
    }

    public void actionPerformed(ActionEvent e){
```

```

        if(e.getSource()== in || e.getSource()== send){
            //이벤트가 발생한 컴포넌트가 in(텍스트필드) 또는 send(보내기단추)이면 수행
            content.append(in.getText()+"\n");
            in.setText("");
        }
    }
}
public static void main( String args[] )
{
    ActionEventTest action = new ActionEventTest();
    action.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```

ItemEvent

체크 박스나 라디오 버튼의 아이템을 선택하거나 해제할 때 또는 콤보 박스에서 아이템을 선택할 때 ItemEvent가 발생한다. ItemEvent는 **ItemListener** 인터페이스를 implements 받아 사용하는데 반드시 ItemListener가 가지고 있는 **itemStateChanged(ItemEvent e)** 메소드를 오버라이딩해야 한다.

```

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

public class ItemEventTest1 extends JFrame {
    JTextField text;
    JCheckBox bold, italic;
    JPanel p;

    public ItemEventTest1()
    {
        super( "체크박스에서 ItemEvent 연습" );

        Container c = getContentPane();
        c.setLayout(new GridLayout(2,1));

        text = new JTextField( "JAVA 연습" );
        text.setFont( new Font( "굴림", Font.PLAIN, 18 ) );
        c.add( text );

        p = new JPanel();
        bold = new JCheckBox( "진하게" );
        p.add( bold );
    }
}

```

```

italic = new JCheckBox( "기울임" );
p.add( italic );
c.add(p);

CheckBoxHandler handler = new CheckBoxHandler(); //이벤트를 처리하는 객체 handler 생성
bold.addItemListener( handler ); //ItemListener에 등록
italic.addItemListener( handler ); //ItemListener에 등록

setSize( 250, 100 );
setVisible(true);
}

private class CheckBoxHandler implements ItemListener { //이벤트를 처리하는 클래스
    int valBold = Font.PLAIN;
    int valItalic = Font.PLAIN;

    public void itemStateChanged( ItemEvent e ) //ItemEvent 를 처리하는 메소드
    {
        if ( e.getSource() == bold ) //진하게 체크박스에서 이벤트 발생시
            if ( e.getStateChange() == ItemEvent.SELECTED ) //선택시
                valBold = Font.BOLD;
            else //해제시
                valBold = Font.PLAIN;

        if ( e.getSource() == italic ) //기울림 체크박스에서 이벤트 발생시
            if ( e.getStateChange() == ItemEvent.SELECTED ) //선택시
                valItalic = Font.ITALIC;
            else //해제시
                valItalic = Font.PLAIN;

        text.setFont(
            new Font( "굴림", valBold + valItalic, 18 ) );
    }
}

public static void main( String args[] )
{
    ItemEventTest1 item = new ItemEventTest1();
    item.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```



MouseEvent

마우스 버튼을 누르거나 마우스를 이동할 때는 MouseEvent가 발생한다.

MouseEvent를 사용하는 두 개의 리스너는 MouseListener와 MouseMotionListener이다.

MouseListener는 마우스 버튼을 클릭하거나 해제하는 것과 관련된 리스너이고, MouseMotionListener는 마우스의 이동과 관련된 리스너이다. 따라서 MouseEvent가 발생할 때는 마우스 버튼을 누를 때와 이동할 때에 따라 각각 MouseListener와 MouseMotionListener에 리스너를 등록해야 한다.

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

public class MouseEventTest extends JFrame implements MouseListener {
    JLabel statusBar;

    public MouseEventTest(){
        super( "MouseEvent에서 MouseListener 사용" );

        statusBar = new JLabel();
        getContentPane().add(statusBar,BorderLayout.SOUTH );

        addMouseListener( this );

        setSize( 300, 150 );
        setVisible(true);
    }

    public void mouseClicked( MouseEvent e ){ //마우스 버튼을 클릭했을 때
        statusBar.setText( "[" + e.getX() + ", " + e.getY() + "] 지점에서 마우스 Click" );
    }

    public void mousePressed( MouseEvent e ){ //마우스 버튼을 눌렀을 때
        statusBar.setText( "[" + e.getX() + ", " + e.getY() + "] 지점에서 마우스 Press" );
    }

    public void mouseReleased( MouseEvent e ){ //마우스 버튼을 놓았을 때
        statusBar.setText( "[" + e.getX() + ", " + e.getY() + "] 지점에서 마우스 Release" );
    }

    public void mouseEntered( MouseEvent e ){ //프레임 안으로 마우스 커서가 들어갈 때
```

```

        statusBar.setText( "마우스 포인터가 해당 윈도우안에 들어옴" );
    }
    public void mouseExited( MouseEvent e ){    //프레임 안의 마우스 커서가 프레임 밖으로 나올 때
        statusBar.setText( "마우스 포인터가 해당 윈도우밖으로 나감" );
    }
    public static void main( String args[]){
        MouseEventTest mouse = new MouseEventTest();
        mouse.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}

```



```

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

public class MouseMotionEventTest extends JFrame{
    private int xValue = 0, yValue = 0;

    public MouseMotionEventTest(){
        super( "MouseEvent에서 MouseMotionListener 사용" );

        addMouseMotionListener( //MouseMotionListener 등록
            new MouseMotionAdapter() {
                public void mouseDragged( MouseEvent e ) //마우스를 드래그하는 이벤트 발생시
                {
                    xValue = e.getX();
                    yValue = e.getY();
                    repaint();
                }
            }
        );

        setSize( 500, 400 );
        setVisible(true);
    }
}

```

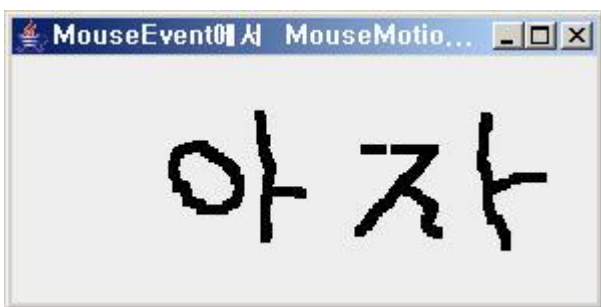
```

}

public void paint( Graphics g ){
    g.fillRect( xValue, yValue, 5, 5 ); //g.fillRect(x좌표, y좌표, width, height)
}

public static void main( String args[] ){
    MouseMotionEventTest mouse = new MouseMotionEventTest();
    mouse.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```



WindowEvent

WindowEvent는 윈도우 창을 닫거나 열 때, 크기를 변경할 때 발생하는 이벤트로 WindowListener 인터페이스를 implements 받기보다는 WindowAdapter클래스를 상속받아 필요한 메소드를 오버라이딩하는 구조로 많이 사용된다.

WindowAdapter 클래스를 상속받아서 windowClosing(WindowEvent e)메소드만 오버라이딩하는 예

```

LabelButtonTest f = new LabelButtonTest();
f.addWindowListener(
    new WindowAdapter() {
        public void windowClosing(WindowEvent e) { // 닫기 단추를 활성화하는 메소드
            System.exit(0);
        }
    }
);

```

ListSelectionEvent

ListSelectionEvent는 JList의 Item을 선택할 때 발생한다. ListSelectionListener를 implements받아 valueChanged(ListSelectionEvent e) 메소드를 오버라이딩해서 사용한다.

해당 컴포넌트를 ListSelectionListener에 등록하면 리스트에서 아이템을 마우스로 클릭할 때마다 valueChanged(ListSelectionEvent e) 메소드가 실행된다.

```
import java.awt.*;
```



```

import javax.swing.*.*;
import javax.swing.event.*;

public class ListSelectionEventTest extends JFrame {
    JList list;
    JLabel label;

    String imageNames[] = { "blue.jpg", "black.jpg", "red.jpg", "green.jpg", "pink.jpg" };
    String listNames[] = { "blue", "black", "red", "green", "pink" };
    Icon icons[];

    public ListSelectionEventTest(){
        super( "JList에서 ListSelectionEvent 연습" );

        Container c = getContentPane();
        c.setLayout( new GridLayout(1,2) );

        list = new JList( listNames );
        list.setVisibleRowCount( 5 );

        list.setSelectionMode( ListSelectionModel.SINGLE_SELECTION );

        c.add( new JScrollPane( list ) );
        icons = new ImageIcon[imageNames.length];
        for(int i=0;i<icons.length;i++){
            icons[i]=new ImageIcon( imageNames[i] );
        }

        list.addListSelectionListener(
            new ListSelectionListener() {
                public void valueChanged(ListSelectionEvent e ) //JList클래스의 객체list의 목록 클릭시
                {
                    label.setIcon(
                        icons[ list.getSelectedIndex() ] );
                }
            }
        );
        label = new JLabel(icons[0]);
        c.add(label);
        setSize( 350, 150 );
        setVisible(true);
    }
}

```



```

public static void main( String args[] ){
    ListSelectionEventTest list = new ListSelectionEventTest();
    list.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```

KeyEvent

KeyEvent는 키보드 키를 누르거나 놓을 때마다 발생한다. KeyListener를 implements받아 KeyPressed(KeyEvent e), keyReleased(KeyEvent e), KeyType(KeyEvent e)를 오버라이딩해야한다. 또는 KeyAdapter를 상속받아 필요한 메소드만 오버라이딩해도 된다..

```

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

public class KeyEventTest extends JFrame implements KeyListener {
    private String key;
    private int x;
    private JTextArea textArea;

    public KeyEventTest(){
        super( "JFrame에서 KeyEvent 연습" );

        textArea = new JTextArea( 10, 15 );
        textArea.setText( "키보드의 키를 누르면 키이름과 키값이 표시" );
        textArea.setEnabled( false );

        addKeyListener( this );

        getContentPane().add( textArea );

        setSize( 350, 100 );
        setVisible(true);
    }

    public void keyPressed( KeyEvent e ){ // 키를 눌렀을 때 실행된다.
        key = "Key pressed: " + e.getKeyText( e.getKeyCode() );
        x = e.getKeyCode();
        textArea.setText( key + " : " + x + "\n" );
    }

    public void keyReleased( KeyEvent e ){ //키를 놓았을 때 실행된다.
    }

    public void keyTyped( KeyEvent e ){ //키를 눌렀다가 놓았을 때 실행된다.
    }
}

```

```

public static void main( String args[] ){
    KeyEventTest key = new KeyEventTest();
    key.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```



ChangeEvent

ChangeEvent는 JSlider와 같은 컴포넌트에서 값이 변화될 때 실행된다. ChangeEvent를 제대로 제어하기 위해서는 ChangeListener인터페이스를 implements 받아 stateChanged(ChangeEvent e) 메소드를 오버라이딩하여 사용해야 한다. 이때 ChangeEvent와 stateChanged(ChangeEvent e)메소드를 연결하려면 ChangeListener를 등록해야 한다.

```

//예제07-23 ChangeEventTest.java
package ch07;
import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;

class Circle extends JPanel { //원을 그리는 클래스
    private int diameter = 20; //원의 지름 20

    public void paintComponent( Graphics g ){ //paint()메소드와 마찬가지로 그리는 메소드 자동 호출
        super.paintComponent( g );
        g.fillOval( 10, 10, diameter, diameter ); //g.fillOval(x좌표, y좌표, width, height);
    }
    public void setDiameter( int diameter ){
        this.diameter = ( diameter >= 0 ? diameter : 20 ); //원의 지름의 기본값 20
        repaint(); //paintComponent( Graphics g )호출
    }
}

public class ChangeEventTest extends JFrame { //JSlider로 원의 크기를 제어하는 클래스
    private JSlider slider;
    private Circle circlePanel;
}

```

```

public ChangeEventTest()
{
    super( "ChangeEvent 연습" );
    circlePanel = new Circle();

    slider = new JSlider( SwingConstants.HORIZONTAL, 0, 100, 10 ); //(위치, 시작, 끝, 처음값)
    slider.setMajorTickSpacing( 10 );
    slider.setPaintTicks( true );
    slider.addChangeListener(
        new ChangeListener() {
            //JSlider 클래스의 객체 slider의 값이 변화할 때마다 불려진다.
            public void stateChanged( ChangeEvent e ){
                circlePanel.setDiameter( slider.getValue());
            }
        }
    );
    Container c = getContentPane();
    c.add( slider, BorderLayout.SOUTH );
    c.add( circlePanel, BorderLayout.CENTER );
    setSize(230, 250);
    setVisible(true);
}

public static void main(String args[]){
    ChangeEventTest change = new ChangeEventTest();
    change.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}

```

