4. 윈도우 프로그래밍

이 장에서는 GUI 프로그래밍 방법 중에서 AWT에 대하여 언급하고 있습니다. AWT에 비해 더욱 강력한 기능을 가진 Swing 컴포넌트 또는 Java FX를 이용하여 윈도우 애플리케이션을 만들 수도 있습니다. 그러나 여러분은 이 장을 통해서 기본적인 GUI 프로그램의 동작 원리를 익힐 수 있고 GUI 애플리케이션을 만들 수 있습니다. 자바로 만든 GUI 애플리케이션은 윈도우 운영체제뿐만 아니라 리눅스나 유닉스 운영체제에서도 실행이 가능하다는 점을 기억해주시기 바랍니다.

주요 내용입니다.

- Button, Label, Checkbox, Chlice, List, Scrollbar, Canvas
- TextField, TextArea
- Panel, Window, Frame, Dialog, FileDialog
- 레이아웃 관리자
- 메뉴
- 색상과 글꼴 변경

4.1. AWT 컴포넌트

AWT(Abstract Window Toolkit)는 GUI와 관련된 클래스들의 묶음입니다. AWT는 GUI와 관련된 모든 클래스를 말하지는 않지만 최근 많이 사용하는 스윙(SWING)도 여기에 포함됩니다. 다시 말해 AWT는 기본적인 GUI와 관련된 클래스의 묶음을 의미합니다. 자바 AWT의 기본 구성요소는 컴포넌트와 컨테이너입니다. 컴포넌트는 일반적으로 GUI에서 버튼(Button)이나 레이블(Label), 또는 텍스트 필드(TextField)와 같이 화면에 해당 구성 요소들을 보이게 나타내는 것이며, 이러한 컴포넌트들은 컨테이너에 포함되어 화면에 출력 나타나게 됩니다. 컨테이너에는 하나 이상의 컴포넌트를 포함할 수 있으며, 컴포넌트뿐만 아니라 다른 컨테이너들도 포함 할 수 있습니다.
AWT 프로그램에서도 모든 컴포넌트들이 객체단위로 움직입니다. 객체를 생성하듯이 컴포넌트들을 생성하고, 참조연산자(.)를 통해서 컴포넌트들의 상태를 변경시킬 수 있습니다.

다음의 예는 AWT를 이용한 간단한 GUI프로그램입니다.

exam/java/chapter11/awt/SimpleWindow.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class SimpleWindow {
6:     private Frame f;
7:
8:     public SimpleWindow() {
9:        f = new Frame("자바 윈도우");
10:    }
11:
12:    public void launchFrame() {
13:       f.setSize(300, 200);
14:       f.setVisible(true);
15:    }
16:
17:    public static void main (String[] args) {
18:       SimpleWindow sw = new SimpleWindow();
19:       sw.launchFrame();
20:    }//end main()
21:
22: }//end class
```

6라인에서 Frame클래스의 객체변수를 선언하였습니다. Frame클래스는 경계선과 타이틀바를 갖는 윈도우를 나타낼 수 있습니다.

```
8:     public SimpleWindow() {
9:        f = new Frame("자바 윈도우");
```

```
10:     }
```

생성자를 만들고 그 안에 Frame클래스의 객체를 생성합니다. Window클래스를 사용하지 않고 Frame클래스를 사용한 것은 Window클래스는 경계선과 타이틀 바를 갖지 않는 네모난 창에 불과하지만, Frame클래스는 Window클래스의 하위 클래스로서 프레임과 타이틀 바를 갖는 윈도우를 생성하기 때문입니다. 경계선과 타이틀 바가 없는 윈도우를 만들고 싶으면 Window클래스를 이용할 수 있습니다.

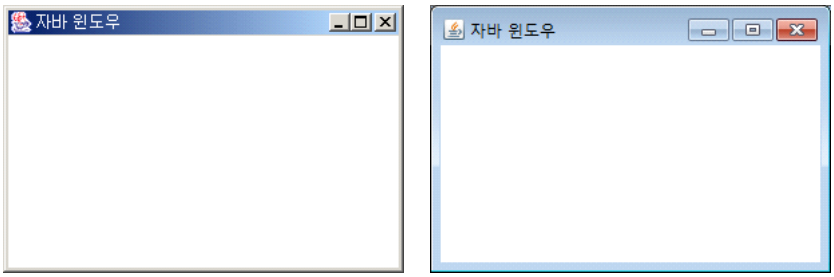
```
12:     public void launchFrame() {
13:         f.setSize(300, 200);
14:         f.setVisible(true);
15:     }
```

객체 생성과 관련된 부분은 생성자를 이용하였고, 이 메서드에서는 그 이외의 것을 기술하였습니다.
13라인의 setSize() 메서드는 컴포넌트 객체의 크기를 픽셀단위로 지정합니다. 여기서는 프레임의 크기를 가로 300픽셀, 세로 200픽셀로 설정하였습니다.
14라인의 setVisible() 메서드는 주어진 논리형 변수 값이 false면 해당 컴포넌트를 화면에 나타내지 않고, true이면 화면에 나타내 줍니다. 만약 이 라인이 없으면 실행은 되지만 화면에는 아무 것도 나타나지 않을 것입니다.

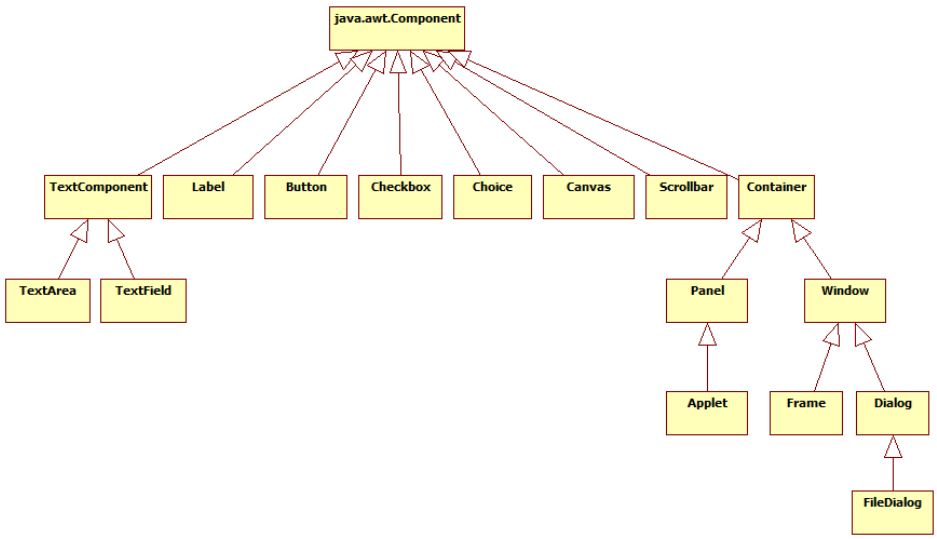
```
17:     public static void main (String[] args) {
18:         SimpleWindow sw = new SimpleWindow();
19:         sw.launchFrame();
20:     }
```

main() 메서드입니다. 객체생성 후 launchFrame() 메서드를 호출하여 윈도우의 크기를 조절하고 화면에 윈도우를 나타냅니다. launchFrame() 메서드 안의 내용을 생성자 안에 기술해도 실행에는 지장이 없습니다. 그러나, 생성자 안에서 객체 생성 이외의 작업을 진행하도록 프로그래밍 하는 방법은 권장할만한 기법은 아닙니다.

다음은 앞의 프로그램을 실행시켰을 때 나타나는 화면입니다. 물론 이벤트를 다루지 않았기 때문에 윈도우의 종료버튼을 눌러도 프로그램이 종료되지 않습니다.(만일 콘솔 창에서 프로그램을 실행하였다면 Ctrl+C를 누르면 프로그램이 강제종료 됩니다.) 그리고 실행 결과는 아래 그림처럼 플랫폼(윈도우 2003(좌) 윈도우 7(우))에 따라 다르게 나타날 수 있습니다.



다음은 자바 컴포넌트 계층 구조를 나타낸 것입니다. 자바에서 사용되는 모든 컴포넌트들을 나타내진 않았습니다. 자주 사용되는 몇 가지 컴포넌트들만 나타냈습니다.



자바의 GUI와 관련된 모든 컴포넌트들은 Component 클래스를 상속받습니다. 그래서 컴포넌트라고 부릅니다. 특성에 따라 컴포넌트들을 분류하면 다음과 같이 3가지로 분류할 수 있습니다.

- **기본 컴포넌트** : Button, Label, Checkbox, Choice, List, Scrollbar, Canvas
- **텍스트 컴포넌트** : TextField, TextArea
- **컨테이너 컴포넌트** : Panel, Applet, Window, Frame, Dialog, FileDialog

컴포넌트 클래스가 제공하는 주요 메서드들을 살펴보면 다음과 같습니다.

- Rectangle getBounds() : 컴포넌트의 테두리(바운드) 정보를 반환합니다.
- Rectangle getBounds(Rectangle r) : 컴포넌트의 바운드 정보를 주어진 Rectangle 객체 r에 저장하고 이 객체를 반환합니다.
- int getX() : 컴포넌트의 x 축 시작좌표 값을 반환합니다.
- int getY() : 컴포넌트의 y축 시작좌표 값을 반환합니다.
- Point getLocation() : 컴포넌트의 현재 위치 좌표를 반환합니다.

◦ Point getLocation(Point p) : 컴포넌트의 현재 위치 좌표를 주어진 Point객체 p에 저장하고 이 객체를 반환합니다.
◦ Point getLocationOnScreen() : 컴포넌트의 화면상에서의 현재 좌표위치를 반환합니다.
◦ int getWidth() : 컴포넌트의 가로(폭)값을 반환합니다.
◦ int getHeight() : 컴포넌트의 세로(높이)값을 반환합니다.
◦ Dimension getSize() : 컴포넌트의 크기를 반환합니다.
◦ Dimension getSize(Dimension d) : 컴포넌트의 크기를 얻어 Dimension객체 d에 저장하고 이 객체를 반환합니다.
◦ void setLocation(int x, int y) : 컴포넌트를 새로운 위치로 옮깁니다.
◦ void setLocation(Point p) : 컴포넌트를 새로운 위치로 옮깁니다.
◦ void setSize(Dimension d) : 컴포넌트의 폭과 높이를 각각 d.width와 d.height로 설정합니다.
◦ void setSize(int width, int height) : 컴포넌트의 폭과 높이를 각각 width와 height로 설정합니다.
◦ void setBounds(int x, int y, int width, int height) : 컴포넌트를 주어진 위치로 옮기고 크기를 변경합니다.
◦ void setBounds(Rectangle r) : 컴포넌트를 주어진 위치로 옮기고 크기를 변경합니다.
◦ void setName(String name) : 컴포넌트의 이름을 설정합니다.
◦ Dimension getMaximumSize() : 컴포넌트의 최대 크기를 반환합니다.
◦ Dimension getMinimumSize() : 컴포넌트의 최소 크기를 반환합니다.
◦ Dimension getPreferredSize() : 컴포넌트의 적당한 크기를 반환합니다.
◦ Component getComponentAt(int x, int y) : 주어진 좌표를 포함하고 있는 컴포넌트를 반환합니다.
◦ Component getComponentAt(Point p) : 주어진 좌표를 포함하고 있는 컴포넌트를 반환합니다.
◦ boolean contains(int x, int y) : 주어진 위치 좌표를 포함하고 있는 컴포넌트가 있는지의 여부를 반환합니다.
◦ boolean contains(Point p) : 주어진 위치좌표를 포함하고 있는 컴포넌트가 있는지의 여부를 반환합니다.
◦ boolean isDisplayable() : 이 컴포넌트가 디스플레이 가능한 지를 반환합니다.
◦ boolean isShowing() : 이 컴포넌트가 보이고 있는지를 반환합니다.
◦ boolean isValid() : 이 컴포넌트가 유효한지를 반환합니다.
◦ boolean isVisible() : 컴포넌트가 보일 수 있는지를 반환합니다.
◦ void setVisible(boolean b) : 컴포넌트를 보이거나 보이지 않게 설정합니다.
◦ void invalidate() : 컴포넌트를 무효화합니다.
◦ void validate() : 이 컴포넌트가 유효한 레이아웃을 갖도록 만듭니다.

4.1.1. 기본 컴포넌트

다음 표는 기본 컴포넌트들의 이름과 기능을 간단하게 나타낸 것입니다. 이들 클래스들은 모두 Component의 하위 클래스들입니다. 그러므로 Component 클래스의 메서드들을 상속받아 사용할 수 있습니다

컴포넌트 이름	컴포넌트 기능
Button	버튼 생성
Label	고정된 문자열 표시
Checkbox	체크박스나 라디오 박스 생성
Choice	드롭다운 메뉴 생성
List	리스트 메뉴 생성
Scrollbar	스크롤바 생성
Canvas	그래픽객체를 그릴 때 사용

4.1.1.1. Button

java.awt.Button 컴포넌트 클래스는 버튼을 생성합니다. 객체를 생성한 후 반드시 컨테이너 클래스의 add() 메서드를 사용하여 컨테이너에 추가해야 합니다.

Button 클래스가 제공하는 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

생성자

- public Button() : 레이블 없는 버튼을 생성합니다.
- public Button(String label) : 주어진 레이블의 버튼을 생성합니다.

메서드

- public String getLabel() : 버튼의 레이블을 반환합니다.
- public void setLabel(String label) : 버튼의 레이블을 주어진 문자열로 설정합니다.

다음은 코드는 Button 컴포넌트 예입니다.

exam/java/chapter11/awt/ButtonExample.java

```
1:  package exam.java.chapter11.awt;
2:
```

```
3:  import java.awt.*;
4:
5:  public class ButtonExample {
6:
7:     private Frame f;
8:     private Button b1, b2;
9:
10:    public ButtonExample() {
11:       f = new Frame("Button Example");
12:       b1 = new Button("버튼 1");
13:       b2 = new Button("버튼 2");
14:    }
15:
16:    public void launchFrame() {
17:       f.setLayout(new FlowLayout());
18:       b1.setBackground(Color.yellow);
19:       b1.setForeground(Color.blue);
20:       b2.setEnabled(false);
21:       f.add(b1);
22:       f.add(b2);
23:       f.setSize(300, 100);
24:       f.setVisible(true);
25:    }
26:
27:    public static void main(String[] args) {
28:       ButtonExample be = new ButtonExample();
29:       be.launchFrame();
30:    }
31: }
```

```
8:     private Button b1, b2;
```

버튼객체를 선언합니다.

```
12:       b1 = new Button("버튼 1");
13:       b2 = new Button("버튼 2");
```

생성자 내에서 각각 버튼의 이름이 “버튼 1”, “버튼 2”인 버튼 객체 두개를 생성합니다.

```
17:       f.setLayout(new FlowLayout());
```

레이아웃 관리자를 FlowLayout으로 지정한 것입니다. 버튼의 모양을 잘 나타내어 이해가 쉽도록 하기 위해 추가한 것입니다. 레이아웃 관리자는 컴포넌트의 배치를 다루는 것으로 자세한 내용은 다음 장에서 설명하기로 하겠습니다.

```
20:       b2.setEnabled(false);
```

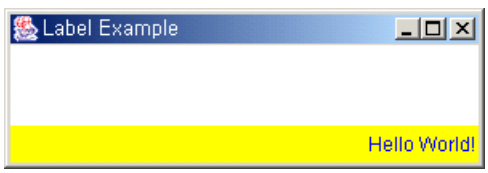
setEnabled() 인자 값을 false로 지정하면 버튼이 입력에 반응하지 않는 비활성 상태가 됩니다. 이 상태에서는 이벤트를 받을 수 없습니다.

```
21:         f.add(myButton);
22:         f.add(yourButton);
```

두 개의 버튼객체를 프레임 객체 참조변수인 f에 add() 메서드를 사용하여 추가합니다.

4.1.1.2. Label

java.awt.Label 컴포넌트 클래스는 고정된 텍스트를 표현하는데 사용합니다. 레이블 객체를 생성한 후 컨테이너 객체의 add() 메서드를 이용하여 컨테이너에 추가시킵니다.



Label 클래스가 제공하는 객체 생성자와 주요 메서드는 다음과 같습니다.

속성

- static int CENTER : 가운데 정렬
- static int LEFT : 왼쪽 정렬
- static int RIGHT : 오른쪽 정렬

생성자

- Label() : 레이블을 생성합니다.
- Label(String text) : 주어진 이름의 레이블을 생성하고, 기본적으로 왼쪽 정렬 상태를 갖습니다.
- Label(String text, int alignment) : 주어진 이름의 레이블을 생성하고, 주어진 정렬방식으로 정렬합니다. alignment 값은 Label.LEFT, Label.CENTER, Label.RIGHT중 하나를 사용할 수 있습니다.

메서드

- int getAlignment() : 현재 정렬 방식을 반환합니다.
- void setAlignment(int alignment) : 정렬 방식을 설정합니다.

◦ String getText() : 레이블의 텍스트를 반환합니다.
◦ void setText(String text) : 레이블의 텍스트를 주어진 텍스트로 설정합니다.

다음은 Label컴포넌트를 나타내는 예입니다.

exam/java/chapter11/awt/LabelExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class LabelExample {
6:
7:     private Frame f;
8:     private Label myLabel;
9:
10:    public LabelExample() {
11:       f = new Frame("Label Example");
12:       myLabel = new Label("Hello World!", Label.RIGHT);
13:    }
14:
15:    public void launchFrame() {
16:       myLabel.setBackground(Color.yellow);
17:       myLabel.setForeground(Color.blue);
18:       f.add(myLabel, BorderLayout.SOUTH);
19:       f.setSize(300, 100);
20:       f.setVisible(true);
21:    }
22:
23:    public static void main(String[] args) {
24:       LabelExample le = new LabelExample();
25:       le.launchFrame();
26:    }
27: }
```

```
8:     private Label myLabel;
```

레이블 객체를 선언합니다.

```
12:       myLabel = new Label("Hello World!", Label.RIGHT);
```

"Hello World!" 문자열을 오른쪽 정렬방식을 가진 레이블 객체로 생성합니다.

```
16:       myLabel.setBackground(Color.red);
17:       myLabel.setForeground(Color.blue);
```

레이블의 색상을 지정하는 코드입니다. setBackground()는 배경색(여기서는 레이블의 배

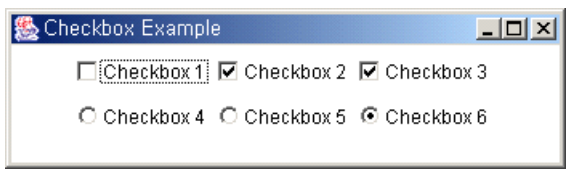
경색)을, setForeground() 전경색(여기서는 글자의 색)을 지정합니다.

```
18:         f.add(myLabel, BorderLayout.SOUTH);
```

객체 myLabel을 add() 메서드를 이용하여 컨테이너 아래쪽(SOUTH)에 추가시킵니다. 위치는 동(EAST), 서(WEST), 남(SOUTH), 북(NORTH), 중앙(CENTER)이 있습니다.

4.1.1.3. Checkbox와 CheckboxGroup

java.awt.Checkbox는 다양한 옵션을 선택하는데 사용됩니다. 체크박스는 "on" 또는 "off" 두 가지 중 한 가지 상태를 나타낼 수 있으며, 체크박스를 누르면 설정상태가 변하며 처리할 동작이 일어납니다. 이와 비슷하게 항목을 선택할 수 있는 컴포넌트에는 Choice, List, Menu 컴포넌트 등이 있습니다.



앞의 그림처럼 체크박스에는 두 종류가 있습니다. 먼저, 위 줄에 있는 세 개의 체크박스는 서로 독립적으로 선택될 수 있기 때문에 동시에 모두 선택할 수도 있고, 하나도 선택하지 않을 수도 있습니다. 아래 세 개의 체크박스는 그룹으로 관리되므로 그룹에 속한 체크박스 중 하나만 선택 할 수 있습니다. 이들 두 그룹의 체크박스는 외형상으로도 차이가 있는데, 위 줄의 체크박스는 일반적인 체크박스 형태인 반면 아래 줄의 체크박스들은 라디오 버튼의 형태를 갖고 있습니다. 여러 개의 체크박스들 중에서 하나만 선택하게 하는 라디오 버튼 형식으로 나타나게 하려면 체크박스 객체를 생성할 때에 체크박스그룹(CheckboxGroup) 속성을 지정해주면 됩니다.

Checkbox 클래스가 제공하는 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

- Checkbox() : 레이블이 없는 체크박스를 생성합니다.
- Checkbox(String label) : 주어진 레이블을 갖는 체크박스를 생성합니다.
- Checkbox(String label, boolean state) : 주어진 레이블을 갖는 체크박스를 생성하며, 체크박스의 초기

선택 여부를 설정합니다.

- Checkbox(String label, boolean state, CheckboxGroup group) : 주어진 레이블을 갖는 체크박스를 생성하며, 체크박스의 초기 선택 여부를 설정하고, 체크박스 그룹을 설정합니다.
- Checkbox(String label, CheckboxGroup group, boolean state) : 주어진 레이블의 체크박스를 생성하며, 체크박스의 초기 선택 여부를 설정하고, 체크박스 그룹을 설정합니다.

메서드

- String getLabel() : 체크박스의 레이블을 반환합니다.
- void setLabel(String label) : 체크박스의 레이블을 설정합니다.
- boolean getState() : 체크박스의 상태가 "on" 또는 "off" 상태인지를 반환합니다.
- void setState(boolean state) : 체크박스의 상태를 설정합니다.(true이면 "on")
- CheckboxGroup getCheckboxGroup() : 설정된 체크박스 그룹을 반환합니다.
- void setCheckboxGroup(CheckboxGroup g) : 체크박스 그룹을 설정합니다.

CheckboxGroup 클래스가 제공하는 객체 생성자와 주요 메서드는 다음과 같다.

생성자

- CheckboxGroup() : 체크박스 그룹 객체를 생성합니다.

메서드

- Checkbox getSelectedCheckbox() : 체크박스 그룹 중에서 현재 선택된 체크박스를 반환합니다.
- void setCurrent(Checkbox box) : setSelectedCheckbox(Checkbox)으로 바뀌었습니다.
- void setSelectedCheckbox(Checkbox box) : 현재 주어진 체크박스가 선택되도록 설정합니다.

다음 코드는 앞에서 설명한 두 가지 형태의 체크박스에 대한 예입니다.

exam/java/chapter11/awt/CheckboxExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class CheckboxExample {
6:     private Frame f;
7:     private Checkbox       check1, check2, check3;
8:     private Checkbox       radio1, radio2, radio3;
9:     private CheckboxGroup group1;
10:
11:    public CheckboxExample() {
12:       f = new Frame("Checkbox Example");
13:       check1 = new Checkbox("Checkbox 1");
14:       check2 = new Checkbox("Checkbox 2", true);
15:       check3 = new Checkbox("Checkbox 3");
16:       group1 = new CheckboxGroup();
```

```
17:        radio1 = new Checkbox("Checkbox 4", group1, false);
18:        radio2 = new Checkbox("Checkbox 5", group1, false);
19:        radio3 = new Checkbox("Checkbox 6", group1, true);
20:     }
21:     public void launchFrame() {
22:        f.setLayout(new FlowLayout());
23:        check3.setState(true);
24:        f.add(check1);  f.add(check2);  f.add(check3);
25:        f.add(radio1);  f.add(radio2);  f.add(radio3);
26:        f.setSize(350, 100);
27:        f.setVisible(true);
28:     }
29:
30:     public static void main(String[] args) {
31:        CheckboxExample ce = new CheckboxExample();
32:        ce.launchFrame();
33:     }
34: }
```

```
7:     private Checkbox check1, check2, check3;
8:     private Checkbox radio1, radio2, radio3;
```

7라인과 8라인에서 Checkbox 클래스를 이용하여 체크박스 객체와 라디오버튼 객체를 선언하였습니다.

```
9:     private CheckboxGroup group1
```

9라인에서 라디오버튼을 만들기 위해 체크박스 그룹객체를 선언하였습니다.

```
13:        check1 = new Checkbox("Checkbox 1");
14:        check2 = new Checkbox("Checkbox 2", true);
15:        check3 = new Checkbox("Checkbox 3");
```

13라인은 체크박스에 이름 부여합니다. 14라인은 체크박스 이름을 부여함과 동시에 초기 선택 값으로 true를 주었습니다. true를 지정하면 초기값이 선택된 상태로 나타납니다.

```
16:        group1 = new CheckboxGroup();
```

라디오버튼을 만들기 위해 7라인에서 선언된 체크박스그룹 객체를 생성하였습니다.

```
17:        radio1 = new Checkbox("Checkbox 4", group1, false);
18:        radio2 = new Checkbox("Checkbox 5", group1, false);
19:        radio3 = new Checkbox("Checkbox 6", group1, true);
```

체크박스를 체크박스그룹으로 생성하였습니다. 이렇게 하면 라디오버튼으로 나타나게 됩니다.

```
24:         check3.setState(true);
```

setState() 메서드의 인자값을 true로 지정하면 체크박스를 선택된 상태로 나타낼 수 있습니다. 선택 해제를 하려면 인자값을 false로 지정하면 됩니다.

```
25:         f.add(check1);  f.add(check2);  f.add(check3);
```

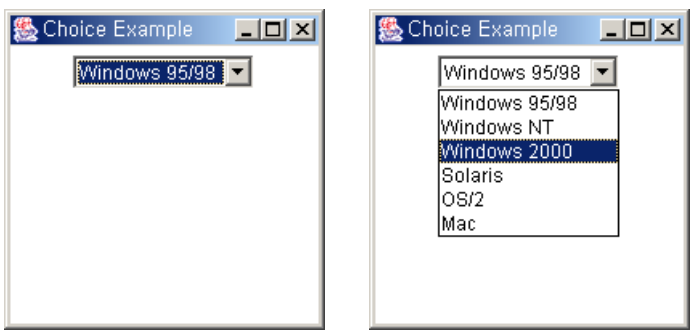
체크박스를 add() 메서드를 이용하여 프레임 컨테이너에 추가시킵니다.

```
25:         f.add(radio1);  f.add(radio2);  f.add(radio3);
```

라디오버튼을 add() 메서드를 이용하여 컨테이너에 추가시킵니다.

4.1.1.4. Choice

java.awt.Choice 컴포넌트 클래스는 다음 그림과 같이 드롭다운(drop-down) 리스트를 제공해주는 컴포넌트입니다. 이 컴포넌트는 제한된 공간에 여러 개의 선택사항들을 나타낼 때 유용합니다.



Choice 클래스의 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

◦ Choice() : 선택 메뉴를 생성합니다.

메서드

- void add(String item) : 아이템을 추가합니다.
- void addItem(String item) : 아이템을 추가합니다. JDK1.2버전에서 자주 보던 것입니다. Choice클래스에서는 사용되지만 다른 대부분의 컴포넌트 클래스에서는 사용하지 않습니다.
- int getItemCount() : 아이템의 개수를 반환합니다.
- String getItem(int index) : 주어진 인덱스에 해당하는 아이템을 반환합니다.
- int getSelectedIndex() : 선택된 아이템의 인덱스를 반환합니다.
- String getSelectedItem() : 선택된 아이템의 이름을 반환합니다.
- void insert(String item, int index) : 주어진 이름의 아이템을 주어진 인덱스에 추가합니다.
- void remove(int position) : 주어진 위치의 아이템을 제거합니다.
- void remove(String item) : 주어진 이름의 아이템을 제거합니다.
- void removeAll() : 선택 메뉴에 있는 모든 아이템을 제거합니다.
- void select(int pos) : 주어진 위치의 아이템이 선택되도록 합니다.
- void select(String str) : 주어진 이름의 아이템이 선택되도록 합니다.

다음 프로그램은 Choice 컴포넌트를 사용하는 예를 보인 것입니다.

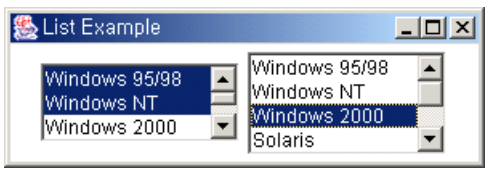
exam/java/chapter11/awt/ChoiceExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class ChoiceExample {
6:
7:     private Frame f;
8:     private Choice myChoice;
9:
10:    public ChoiceExample() {
11:       f = new Frame("Choice Example");
12:       myChoice = new Choice();
13:    }
14:
15:    public void launchFrame() {
16:       f.setLayout(new FlowLayout());
17:       myChoice.add("Windows 95/98");
18:       myChoice.add("Windows NT");
19:       myChoice.add("Windows 2000");
20:       myChoice.add("Solaris");
21:       myChoice.add("OS/2");
22:       myChoice.add("Mac");
23:       f.add(myChoice);
24:       f.setSize(200, 200);
25:       f.setVisible(true);
26:    }
27:    public static void main (String[] args) {
28:       ChoiceExample ce = new ChoiceExample();
```

```
29:         ce.launchFrame();
30:     }
31: }
```

4.1.1.5. List

java.awt.List 컴포넌트 클래스는 다수의 선택항목이 목록으로 보이고 사용자가 이를 선택하고자 할 때 사용합니다. 원하는 아이템을 클릭하여 선택하고, 더블 클릭하거나 엔터키로 액션 이벤트를 발생시킬 수 있습니다. List 컴포넌트는 동시에 여러 개를 선택할 수도 있고 하나의 아이템만 선택할 수도 있습니다.



그림에서 왼쪽리스트는 동시에 여러 개의 아이템을 선택할 수 있고, 오른쪽리스트는 하나의 아이템만 선택할 수 있는 리스트 컴포넌트입니다.

List 클래스가 제공하는 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

- List() : 스크롤 가능한 리스트 컴포넌트를 생성합니다. 기본 값으로 4개 항목이 최초 목록 개수로 설정됩니다.
- List(int rows) : 주어진 개수만큼의 줄을 보이는 스크롤 가능한 리스트 컴포넌트를 생성합니다.
- List(int rows, boolean multipleMode) : 주어진 개수만큼의 줄을 보이게 하는 스크롤 가능한 리스트 컴포넌트를 생성하면서, 다수의 아이템을 동시에 선택가능하게 할 것인지의 여부를 설정합니다. multipleMode 값이 true 이면 아이템을 여러 개 선택할 수 있습니다.

메서드

- void add(String item) : 주어진 이름의 아이템을 추가합니다.
- void add(String item, int index) : 주어진 이름의 아이템을 해당 인덱스에 추가합니다.
- void deselect(int index) : 주어진 인덱스의 아이템을 선택 해제합니다.
- String getItem(int index) : 주어진 인덱스의 아이템을 반환합니다.
- int getItemCount() : 리스트 내의 아이템 개수를 반환합니다.
- String[] getItems() : 리스트 내의 아이템 배열을 반환합니다.
- int getRows() : 리스트에서 보이는 아이템의 개수를 반환합니다.

- int getSelectedIndex() : 선택된 아이템의 개수를 반환합니다.
- int[] getSelectedIndexes() : 선택된 아이템의 인덱스 배열을 반환합니다.
- String getSelectedItem() : 선택된 아이템의 배열을 반환합니다.
- String[] getSelectedItems() : 선택된 아이템의 배열을 반환합니다.
- Object[] getSelectedObjects() : 선택된 아이템을 Object 객체 배열로 반환합니다.
- int getVisibleIndex() : makeVisible() 메서드에 의해 마지막으로 보인 아이템의 인덱스를 반환합니다.
- boolean isIndexSelected(int index) : 인덱스에 해당하는 아이템이 선택되었는지 여부를 반환합니다.
- boolean isMultipleMode() : 여러 개의 아이템이 선택가능한지 여부를 반환합니다.
- void makeVisible(int index) : 주어진 인덱스에 해당하는 아이템을 보이게 합니다.
- void remove(int position) : 주어진 위치의 아이템을 제거합니다.
- void remove(String item) : 주어진 이름의 아이템을 제거합니다.
- void removeAll() : 모든 아이템을 제거합니다.
- void replaceItem(String newValue, int index) : 주어진 인덱스에 해당하는 아이템을 새로운 이름으로 변경합니다.
- void select(int index) : 인덱스에 해당하는 아이템을 선택합니다.
- void setMultipleMode(boolean b) : 여러 개의 아이템을 선택 가능하도록 설정합니다.

다음 프로그램은 List컴포넌트를 사용하는 예를 보인 것입니다.

exam/java/chapter11/awt/ListExample.java

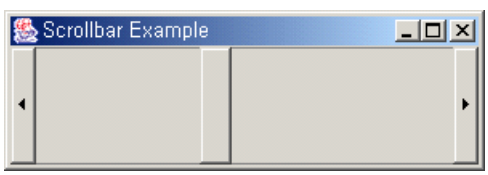
```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class ListExample {
6:     private Frame f;
7:     private List myList, yourList;
8:
9:     public ListExample() {
10:       f = new Frame("List Example");
11:       myList = new List(3, true);
12:       yourList = new List();
13:    }
14:
15:    public void launchFrame() {
16:       f.setLayout(new FlowLayout());
17:       myList.add("Windows 95/98");
18:       myList.add("Windows NT");
19:       myList.add("Windows 2000");
20:       myList.add("Solaris");
21:       myList.add("OS/2");
22:       myList.add("Machintosh");
23:       myList.add("MS-DOS");
24:       yourList.add("Windows 95/98");
25:       yourList.add("Windows NT");
26:       yourList.add("Windows 2000");
```

```
27:         yourList.add("Solaris");
28:         yourList.add("OS/2");
29:         yourList.add("Machintosh");
30:         yourList.add("MS-DOS");
31:
32:         f.add(myList);
33:         f.add(yourList);
34:         f.setSize(300, 100);
35:         f.setVisible(true);
36:     }
37:
38:     public static void main (String[] args) {
39:         ListExample le = new ListExample();
40:         le.launchFrame();
41:     }
42: }
```

11라인은 리스트 아이템이 3줄씩 보이게 하면서 여러 개의 아이템을 동시에 선택 가능(true)하게 하는 객체를 지정합니다. 생성자의 두 번째 인자인 true가 동시에 여러 개 선택이 가능하게 하는 것입니다.
12라인은 스크롤 가능한 리스트 컴포넌트를 생성합니다.
17라인에서 30라인까지는 주어진 이름의 아이템을 각각 해당 리스트에 추가합니다.

4.1.1.6. Scrollbar

java.awt.Scrollbar 컴포넌트 클래스는 연속적인 값을 선택하거나, 다른 컴포넌트의 옆에 붙어서 수평, 수직 스크롤바와 같이 실제 보이는 영역을 지정하는 역할을 합니다.



Scrollbar 클래스가 제공하는 객체 생성자와 주요 메서드는 다음과 같습니다.

속성

- static int HORIZONTAL : 수평 스크롤바 속성입니다.
- static int VERTICAL : 수직 스크롤바 속성입니다.

생성자

- Scrollbar() : 스크롤바를 생성합니다.
- Scrollbar(int orientation) : 주어진 값에 따라 수평/수직 스크롤바를 생성합니다.
- Scrollbar(int orientation, int value, int visible, int minimum, int maximum) : 주어진 스크롤바의 속성값에 해당하는 스크롤바를 생성합니다. 각 값은 다음과 같은 의미를 갖습니다.
  - int orientation : 스크롤바가 수평이면 Scrollbar.HORIZONTAL으로, 스크롤바가 수직이면 Scrollbar.VERTICAL으로 설정해줍니다.
  - int value : 스크롤바의 초기값을 나타냅니다. 일반적인 초기값은 0입니다.
  - int visible : 스크롤 가능한 영역의 보이는 부분에 대한 픽셀 단위의 크기를 나타냅니다.
  - int minimum : 스크롤바가 가질 수 있는 최소값을 나타냅니다. 스크롤 영역을 설정하기 위한 스크롤바의 경우에는 0이 됩니다.
  - int maximum : 스크롤바가 가질 수 있는 최대값을 나타낸다. 스크롤 영역을 설정하기 위한 스크롤바의 경우 픽셀 단위의 width 또는 height 값이 됩니다.

메서드

- int getBlockIncrement() : 스크롤바의 블록 증가량을 반환합니다.
- int getMaximum() : 스크롤바의 최대값을 반환합니다.
- int getMinimum() : 스크롤바의 최소값을 반환합니다.
- int getOrientation() : 스크롤바의 방향성 값을 반환합니다.
- int getUnitIncrement() : 스크롤바의 단위 증가량을 반환합니다.
- int getValue() : 스크롤바의 현재 값을 반환합니다.
- int getVisibleAmount() : 스크롤바의 보이는 부분의 양을 반환합니다.
- void setBlockIncrement(int v) : 스크롤바의 블록 증가량을 설정합니다.
- void setMaximum(int newMaximum) : 스크롤바의 최대값을 설정합니다.
- void setMinimum(int newMinimum) : 스크롤바의 최소값을 설정합니다.
- void setOrientation(int orientation) : 스크롤바의 방향성을 설정합니다.
- void setUnitIncrement(int v) : 스크롤바의 단위 증가량을 설정합니다.
- void setValue(int newValue) : 스크롤바의 현재 값을 설정합니다.
- void setValues(int value, int visible, int minimum, int maximum) : 스크롤바의 속성 값을 설정합니다.
- void setVisibleAmount(int newAmount) : 스크롤바의 보이는 부분의 양을 설정합니다.

다음 프로그램은 Scrollbar컴포넌트를 사용하는 예를 보인 것입니다.

exam/java/chapter11/awt/ScrollbarExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class ScrollbarExample {
6:     private Frame f;
7:     private Scrollbar mySlider;
8:
```

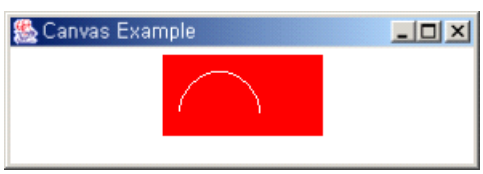
```
9:      public ScrollbarExample() {
10:        f = new Frame("Scrollbar Example");
11:        mySlider = new Scrollbar(Scrollbar.HORIZONTAL,100,20,0,255);
12:     }
13:
14:     public void launchFrame() {
15:        f.add(mySlider);
16:        f.setSize(300, 100);
17:        f.setVisible(true);
18:     }
19:
20:     public static void main(String[] args) {
21:        ScrollbarExample se = new ScrollbarExample();
22:        se.launchFrame();
23:     }
24: }
```

```
11:        mySlider = new Scrollbar(Scrollbar.HORIZONTAL,100,20,0,255);
```

최소값 0, 최대값 255, 초기값 100, 스크롤바 포인터(bubble)의 크기가 20픽셀인 수평 스크롤 바를 생성합니다.

4.1.1.7. Canvas

java.awt.Canvas 컴포넌트 클래스는 특정한 모습이 없으며, 그림을 그리는 등 영상처리 작업을 하는데 유용한 컴포넌트입니다.



다음 프로그램은 Canvas컴포넌트의 사용 예를 보인 것입니다. 이 예제는 실제 Canvas 클래스의 사용 예를 보여주기에는 충분하지 않습니다. 지금은 캔바스라는 것이 있다는 것과 이곳에 그림을 그릴 수 있다는 정도만 알아두시기 바랍니다. 윈도우 화면에 그림을 그리기 위해서 Canvas 클래스를 상속받아 새로운 클래스를 작성해야 합니다. 그리고 그 클래스에서 paint(Graphics g) 메서드를 재정의 해야 합니다.

exam/java/chapter11/awt/CanvasExample.java

```
1:  package exam.java.chapter11.awt;
2:
```

```
3:  import java.awt.*;
4:
5:  public class CanvasExample {
6:     private Frame f;
7:     private Canvas myCanvas;
8:
9:     public CanvasExample() {
10:       f = new Frame("Canvas Example");
11:       myCanvas = new MyCanvas();
12:    }
13:
14:    public void launchFrame() {
15:       f.setLayout(new FlowLayout());
16:       myCanvas.setBackground(Color.red);
17:       myCanvas.setSize(100, 50);
18:
19:       f.add(myCanvas);
20:       f.setSize(300, 100);
21:       f.setVisible(true);
22:    }
23:
24:    public static void main(String[] args) {
25:       CanvasExample ce = new CanvasExample();
26:       ce.launchFrame();
27:    }
28: }
29:
30: class MyCanvas extends Canvas {
31:    public void paint(Graphics g) {
32:       g.setColor(Color.WHITE);
33:       g.drawArc(10, 10, 50, 50, 0, 180);
34:    }
35: }
```

4.1.2. 텍스트 컴포넌트

텍스트 컴포넌트에는 TextField와 TextArea가 있습니다.

컴포넌트 이름	컴포넌트 기능
TextField	한 줄 문자입력
TextArea	여러 줄 문자입력

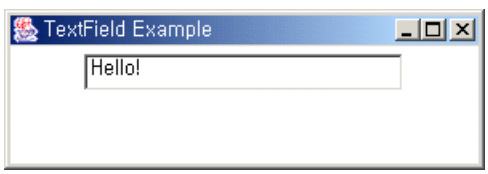
텍스트 컴포넌트의 하위 클래스에는 문자열을 한 줄 입력받을 수 있는 TextField와 여러

줄 입력받을 수 있는 TextArea가 있습니다. 이 두 클래스에서 공통으로 사용할 수 있는 TextComponent 클래스가 제공하는 주요 메서드들을 살펴보면 다음과 같습니다.

- int getCaretPosition() : 텍스트 삽입 캐럿(I자 형태)의 위치를 반환합니다.
- String getSelectedText() : 선택 영역의 텍스트를 반환합니다.
- int getSelectionEnd() : 선택된 텍스트의 끝 위치를 반환합니다.
- int getSelectionStart() : 선택된 텍스트의 시작 위치를 반환합니다.
- String getText() : 현재의 텍스트를 반환합니다.
- boolean isEditable() : 편집 가능한 상태인지를 반환합니다.
- void select(int selectionStart, int selectionEnd) : 시작 위치부터 끝 위치까지 선택합니다.
- void selectAll() : 모든 텍스트를 선택합니다.
- void setCaretPosition(int position) : 삽입 위치(캐럿)를 설정합니다.
- void setEditable(boolean b) : 편집 상태를 설정합니다.
- void setSelectionEnd(int selectionEnd) : 선택 영역의 끝 위치를 설정합니다.
- void setSelectionStart(int selectionStart) : 선택 영역의 시작 위치를 설정합니다.
- void setText(String t) : 현재 텍스트를 설정합니다.

4.1.2.1. TextField

java.awt.TextField 컴포넌트 클래스는 한 줄 내에서 사용자의 문자 입력을 받습니다. java.awt.TextComponent가 TextField 클래스의 상위 클래스이며, TextComponent 클래스는 텍스트 입력과 관련된 메서드를 처리해 주는 기능을 가지고 있습니다.



TextField 클래스의 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

- TextField() : 텍스트 필드를 생성합니다.

◦ TextField(int columns) : 주어진 열의 크기(개수)를 갖는 텍스트 필드를 생성합니다.
◦ TextField(String text) : 주어진 텍스트를 갖는 텍스트 필드를 생성합니다.
◦ TextField(String text, int columns) : 주어진 텍스트와 열의 크기(개수)를 갖는 텍스트 필드를 생성합니다.

메서드

◦ boolean echoCharIsSet() : 에코 문자가 설정되어 있는지 여부를 반환합니다. 패스워드를 입력받을 때 이 컴포넌트를 이용합니다. 패스워드 대신 화면상에 나타날 문자를 에코 문자라고 합니다.
◦ int getColumns() : 열의 크기(개수)를 반환합니다.
◦ char getEchoChar() : 에코 문자를 반환합니다.
◦ void setColumns(int columns) : 텍스트 필드의 크기인 열의 크기(개수)를 설정합니다.
◦ void setEchoChar(char c) : 에코 문자를 설정합니다.
◦ void setText(String t) : 텍스트를 설정합니다.

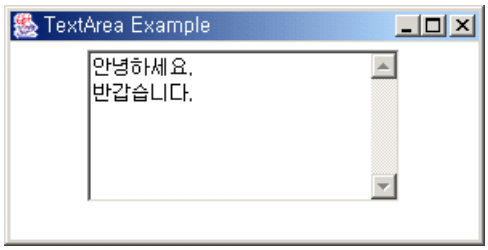
다음 프로그램은 TextField컴포넌트를 사용하는 예를 보인 것입니다.

exam/java/chapter11/awt/TextFieldExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class TextFieldExample {
6:     private Frame f;
7:     private TextField myTextField;
8:
9:     public TextFieldExample() {
10:       f = new Frame("TextField Example");
11:       myTextField = new TextField("Hello!", 25);
12:    }
13:
14:    public void launchFrame() {
15:       f.setLayout(new FlowLayout());
16:       f.add(myTextField);
17:       f.setSize(300, 100);
18:       f.setVisible(true);
19:    }
20:
21:    public static void main(String[] args) {
22:       TextFieldExample te = new TextFieldExample();
23:       te.launchFrame();
24:    }
25: }
```

4.1.2.2. TextArea

java.awt.TextArea 컴포넌트 클래스는 여러 줄의 문자입력을 받을 때 사용합니다.



TextArea 클래스의 객체 생성자와 주요 메서드는 다음과 같습니다.

속성

- static int SCROLLBARS_BOTH : 수직/수평 스크롤바를 생성하고 보여주는 속성입니다.
- static int SCROLLBARS_HORIZONTAL_ONLY : 수평 스크롤바를 생성하고 보여주는 속성입니다.
- static int SCROLLBARS_VERTICAL_ONLY : 수직 스크롤바를 생성하고 보여주는 속성입니다.
- static int SCROLLBARS_NONE : 스크롤바를 생성하지 않는 속성입니다.

생성자

- TextArea() : 텍스트영역을 생성합니다.
- TextArea(int rows, int columns) : 주어진 행과 열을 갖는 텍스트 영역을 생성합니다.
- TextArea(String text) : 주어진 텍스트를 갖는 텍스트 영역을 생성합니다.
- TextArea(String text, int rows, int columns) : 주어진 행, 열, 텍스트를 갖는 텍스트영역을 생성합니다.
- TextArea(String text, int rows, int columns, int scrollbars) : 주어진 행, 열, 텍스트, 그리고 스크롤바를 갖는 텍스트 영역을 생성합니다.

메서드

- void append(String str) : 문자열을 텍스트 영역에 추가합니다.
- int getColumns() : 열의 크기(개수)를 반환합니다.
- int getRows() : 행의 크기(개수)를 반환합니다.
- void insert(String str, int pos) : 주어진 위치(라인)에 문자열을 삽입합니다.
- void replaceRange(String str, int start, int end) : 시작 위치부터 끝 위치 사이의 문자열을 주어진 문자열로 대체합니다.
- void setColumns(int columns) : 열의 크기(개수)를 설정합니다.
- void setRows(int rows) : 행의 크기(개수)를 설정합니다.

다음 프로그램은 TextArea컴포넌트를 사용하는 예를 보인 것입니다.

exam/java/chapter11/awt/TextAreaExample.java

```
1:  package exam.java.chapter11.awt;
2:
```

```
3:  import java.awt.*;
4:
5:  public class TextAreaExample {
6:     private Frame f;
7:     private TextArea myTextArea;
8:
9:     public TextAreaExample() {
10:       f = new Frame("TextArea Example");
11:       myTextArea = new TextArea("안녕하세요.\n반갑습니다.", 5, 25);
12:    }
13:
14:    public void launchFrame() {
15:       f.setLayout(new FlowLayout());
16:       f.add(myTextArea);
17:       f.setSize(300, 150);
18:       f.setVisible(true);
19:    }
20:
21:    public static void main(String[] args) {
22:       TextAreaExample te = new TextAreaExample();
23:       te.launchFrame();
24:    }
25: }
```

4.1.3. 컨테이너 컴포넌트

컨테이너(Container) 컴포넌트는 자신의 영역 안에 다른 컴포넌트를 포함할 수 있는 컴포넌트 입니다. 혼자서는 특별한 동작을 할 수 없으며, 다른 컴포넌트를 포함할 때만 의미가 있습니다. 컨테이너 클래스도 컴포넌트 클래스의 하위 클래스이기 때문에 그 자체도 컴포넌트로 취급되며, 다른 컨테이너 내에 포함될 수 있습니다.
컴포넌트는 컨테이너에 포함되지 않으면 독자적으로 화면에 나타낼 수가 없습니다. 컨테이너에 컴포넌트를 포함시키기 위해서는 컨테이너 클래스의 add() 메서드를 이용해야 합니다.

컨테이너 클래스가 제공하는 주요 메서드는 다음과 같습니다.

- public Component add(Component comp) : 주어진 컴포넌트를 컴포넌트의 끝에 추가합니다.
- public void setLayout(LayoutManager mgr) : 레이아웃 관리자를 설정합니다.

컨테이너 컴포넌트의 종류와 기능은 다음과 같습니다.

컴포넌트 이름	컴포넌트 기능
Panel	컴포넌트 배치
Applet	애플릿 생성
Window	윈도우 생성(경계선과 타이틀바가 없음)
Frame	경계선과 타이틀바를 갖는 윈도우 생성
Dialog	대화상자 생성
FileDialog	파일 대화상자 생성

4.1.3.1. Panel

java.awt.Panel 컨테이너 클래스는 컨테이너의 하위 클래스로서 독립적인 모양이 없고 독립적인 창으로 사용할 수는 없습니다. 그러나 패널 컴포넌트는 다른 컴포넌트를 포함하거나 배치시키는데 사용합니다. 그리고 패널이 포함하고 있는 컴포넌트에 대한 이벤트를 처리할 수 있는 특별한 기능을 가지고 있습니다.

다음 프로그램은 서로 다른 Panel객체를 이용하여 버튼 컴포넌트를 배치시키는 예를 나타낸 것입니다.

exam/java/chapter11/awt/PanelExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class PanelExample {
6:     private Frame f;
7:     private Panel p1, p2;
8:     private Button b1, b2, b3, b4, b5, b6;
9:
10:    public PanelExample() {
11:       f = new Frame("Panel Example");
12:       p1 = new Panel();
13:       p2 = new Panel();
14:       b1 = new Button("Button 1");
15:       b2 = new Button("Button 2");
16:       b3 = new Button("Button 3");
17:       b4 = new Button("Button 4");
18:       b5 = new Button("Button 5");
19:       b6 = new Button("Button 6");
20:    }
21:
22:    public void launchFrame() {
```

```
23:         p1.setBackground(Color.yellow);
24:         p1.add(b1);
25:         p1.add(b2);
26:         p1.add(b3);
27:
28:         p2.setBackground(Color.red);
29:         p2.add(b4);
30:         p2.add(b5);
31:         p2.add(b6);
32:
33:         f.add(p1, BorderLayout.EAST);
34:         f.add(p2, BorderLayout.CENTER);
35:         f.setSize(300, 150);
36:         f.setVisible(true);
37:     }
38:
39:     public static void main(String[] args) {
40:         PanelExample pe = new PanelExample();
41:         pe.launchFrame();
42:     }
43: }
```

```
7:         Panel p1, p2;
```

Panel 클래스로부터 새로운 패널 객체 2개(p1, p2)를 정의하였습니다.

```
12:         p1 = new Panel();
13:         p2 = new Panel();
```

5라인에서 선언한 패널 객체 p1과 p2에 패널 객체를 생성 하였습니다.

```
23:         p1.setBackground(Color.yellow);
```

패널 객체 p1의 바탕색을 노란색으로 지정합니다. setBackground()는 배경 색을 지정하는데 사용됩니다.

```
24:         p1.add(b1);
```

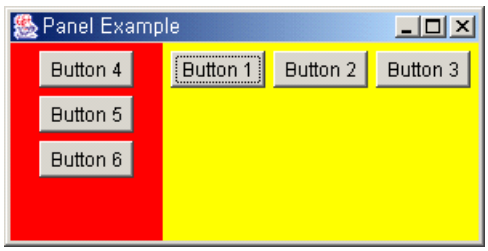
버튼 객체를 add() 메서드를 이용하여 패널 객체 p1에 부착시킵니다.

```
33:         f.add(p1, BorderLayout.EAST);
```

패널객체 p1을 프레임에 추가합니다. add() 메서드의 인자 값 중에서 BorderLayout.EAST는 BorderLayout 클래스에 선언되어 있는 상수입니다. 이는 Border

레이아웃 관리자를 따르는 컨테이너 컴포넌트의 오른쪽에 객체를 배치하라는 뜻입니다. 레이아웃 관리자는 컴포넌트들을 화면에 배치하는 용도로 사용되며, 다음 장에서 자세히 설명하기로 하겠습니다.

다음은 앞의 프로그램을 실행한 결과 화면입니다. 패널을 이용하면 컴포넌트의 배치를 좀 더 쉽고 다양하게 할 수 있습니다.



4.1.3.2. Window

java.awt.Window 컨테이너 클래스는 윈도우가 가져야 할 기본적인 기능을 제공하는 클래스로서, Dialog 와 Frame 클래스를 하위 클래스로 갖고 있습니다. Window 클래스는 경계선과 타이틀 바가 없는 윈도우를 생성합니다. 또한 Window 클래스를 상속받아 하위 클래스를 정의하면 독립된 윈도우로 동작할 수 있습니다. 실제 윈도우를 생성할 때는 Window 클래스를 사용하지 않고 Window의 하위 클래스인 Frame 클래스를 이용합니다.

Window 클래스의 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

- Window(Frame owner) : 주어진 프레임에 속하는 새로운 윈도우를 보이게 합니다.
- Window(Window owner) : 주어진 윈도우에 속하는 새로운 윈도우를 보이게 합니다.

메서드

- void dispose() : 윈도우에 속한 컴포넌트가 사용하는 모든 네이티브 스크린 자원의 할당을 해제합니다.
- Component getFocusOwner() : 윈도우가 활성화되어 있을 때, 현재 포커스를 가지고 있는 윈도우의 자식 컴포넌트를 반환합니다.
- InputContext getInputContext() : 윈도우의 입력 문장을 반환합니다.
- Locale getLocale() : 윈도우에 연결되어 있는 로케일 객체(Locale Object)를 반환합니다.
- Window[] getOwnedWindows() : 이 윈도우에 속한 모든 윈도우를 포함하는 배열을 반환합니다.
- Window getOwner() : 윈도우의 부모윈도우를 반환합니다.

- Toolkit getToolkit() : 프레임의 툴킷을 반환합니다.
- String getWarningString() : 윈도우에 출력될 경고 문자열을 반환합니다.
- boolean isShowing() : 이 윈도우가 스크린 상에 보이고 있는지를 반환합니다.
- void pack() : 윈도우를 적당한 크기로 조정하고 자신에 속한 컴포넌트를 배치합니다.
- void show() : 윈도우가 보이도록 합니다.
- void toBack() : 윈도우를 뒤로 보냅니다.
- void toFront() : 윈도우를 앞으로 가져옵니다.

다음 프로그램은 Frame을 이용하여 윈도우를 만들고 그 위에 Window객체를 이용하여 윈도우를 나타내는 예입니다.

exam/java/chapter11/awt/WindowExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class WindowExample extends Frame {
6:
7:     private Window myWindow;
8:
9:     public WindowExample() {
10:       myWindow = new Window(this);
11:       myWindow.setLayout(new FlowLayout());
12:       myWindow.add(new Label("New Window"));
13:    }
14:
15:    public void launchWindow() {
16:       myWindow.setLocation(150,150);
17:       myWindow.setBackground(Color.cyan);
18:       myWindow.setSize(250, 150);
19:       myWindow.show();
20:    }
21:
22:    public static void main(String[] args) {
23:       WindowExample we = new WindowExample();
24:
25:       we.setLocation(100,100);
26:       we.setSize(250, 150);
27:       we.setVisible(true);
28:
29:       we.launchWindow();
30:    }
31: }
```

```
7:     private Window myWindow;
```

Window 클래스로부터 새로운 윈도우 객체를 정의합니다.

```
10:         myWindow = new Window(this);
```

윈도우 객체를 생성합니다.

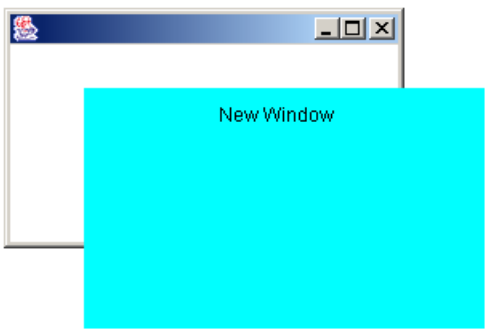
```
11:         myWindow.setLayout(new FlowLayout());
```

생성된 윈도우 객체의 레이아웃 관리자를 지정합니다. 레이아웃 관리자에 대해서는 다음 장에서 설명하겠습니다.

```
12:         myWindow.add(new Label("New Window"));
```

생성된 윈도우 객체에 "New Window"라는 문자열을 가진 레이블을 생성하여 포함시킵니다.

다음 그림은 앞의 프로그램을 실행시켰을 때 나타나는 결과화면입니다.



4.1.3.3. Frame

java.awt.Frame 클래스는 Window의 하위 클래스로서 경계선과 타이틀 바를 갖는 윈도우를 만들 수 있게 합니다. 앞에서 예를 든 컴포넌트 프로그램은 모두 Frame 클래스를 상속하여 나타낸 것입니다. 이로 미루어 알 수 있는 것은 윈도우를 이용하는 모든 프로그램은 적어도 하나의 프레임을 필요로 한다는 것입니다.

프레임 클래스의 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

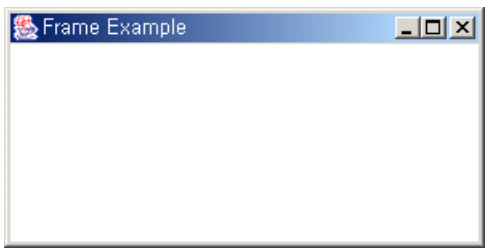
생성자

- Frame() : 새로운 프레임을 생성합니다. 초기값은 보이지 않게 설정됩니다.
- Frame(String title) : 주어진 이름을 갖는 새로운 프레임을 생성합니다. 초기값은 보이지 않게 설정됩니다.

메서드

- static Frame[] getFrames() : 생성된 모든 프레임을 포함하는 배열을 반환합니다.
- Image getIconImage() : 프레임이 최소화 아이콘 상태일 때, 표시될 이미지를 반환합니다.
- void setIconImage(Image image) : 프레임이 최소화 아이콘 상태일 때, 표시할 이미지를 설정합니다.
- MenuBar getMenuBar() : 프레임 메뉴바를 반환합니다.
- void setMenuBar(MenuBar mb) : 주어진 메뉴 바로 프레임의 메뉴바를 설정합니다.
- int getState() : 프레임 상태를 반환합니다.
- void setState(int state) : 프레임 상태를 설정합니다.
- String getTitle() : 프레임 제목을 반환합니다.
- void setTitle(String title) : 주어진 문자열로 프레임 제목을 설정합니다.
- boolean isResizable() : 사용자에 의해 프레임의 크기가 변경가능한지를 반환합니다.
- void setResizable(boolean resizable) : 사용자에 의해 프레임의 크기가 변경가능한지를 설정합니다.
- void remove(MenuComponent m) : 주어진 메뉴바를 프레임에서 제거합니다.

다음 그림은 Frame 클래스를 이용하여 윈도우를 나타낸 것입니다.



다음 프로그램은 Frame을 이용하여 윈도우를 나타내는 예입니다.

exam/java/chapter11/awt/FrameExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class FrameExample {
6:     private Frame f;
7:
8:     public FrameExample() {
9:        f = new Frame("Frame Example");
10:    }
11:
```

```
12:     public void launchFrame() {
13:         f.setSize(300, 150);
14:         f.setVisible(true);
15:     }
16:
17:     public static void main(String[] args) {
18:         FrameExample fe = new FrameExample();
19:         fe.launchFrame();
20:     }
21: }
```

```
9:          f = new Frame("Frame Example");
```

새로운 윈도우 객체를 생성합니다.

```
13:         f.setSize(300, 150);
14:         f.setVisible(true);
```

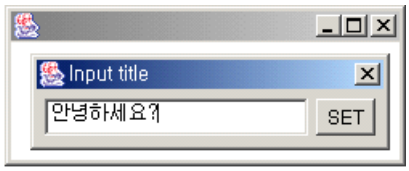
13라인은 윈도우의 크기를 가로 300픽셀, 세로 150픽셀로 설정합니다. 그리고 14라인은 윈도우를 화면에 보이게 합니다. 14라인이 생략되거나 setVisible() 메서드의 인자가 false로 지정되면 화면에서 사라집니다. 화면에 보이지 않는다고 해서 객체가 제거되는 것은 아닙니다. 화면에 안보일 뿐입니다.

4.1.3.4. Dialog

java.awt.Dialog는 프레임 클래스와 함께 Window의 하위 클래스로서 FileDialog를 하위 클래스로 두고 있으며, AWT프로그램에서 대화상자를 생성하도록 합니다. 대화상자는 윈도우에 종속되기 때문에 그 윈도우가 닫히면 대화상자도 함께 닫히게 되고, 윈도우가 최소화되면 대화상자는 사라지게 됩니다. 대화상자는 non-modal 또는 modalless를 기본값으로 갖는데, non-modal(또는 modalless)은 대화상자가 나타나 있을 때도 윈도우에서 다른 작업을 할 수 있는 상태를 말하며, 그 반대의 경우 즉, 대화상자가 닫힐 때까지 윈도우에서 대화상자 이외의 다른 작업을 할 수 없는 상태를 모달(modal)이라고 합니다.
애플릿은 프레임이나 윈도우 클래스를 상속받지 않고 패널 클래스를 상속받으므로 윈도우를 가질 수 없기 때문에 대화상자를 사용할 수도 없습니다. 그러나 애플릿이 자신의 윈도우(또는 프레임)를 가지면 그 윈도우에 종속된 대화상자를 가질 수 있습니다.

다음 그림은 프레임 윈도우의 제목표시줄에 Dialog 클래스를 이용하여 제목을 지정하는 프로그램의 결과입니다. 프로그램 소스코드는 생성자와 메서드에 대한 설명 뒤에 있습니다. 프로그램은 두 개의 클래스를 작성해야하며 TitleBox.java를 먼저 작성한 다음 DialogExample.java를 작성해야 합니다. 실행은 DialogExample.java를 컴파일하면 만들어지는 DialogExample.class파일을 이용합니다. 물론 두 개의 프로그램을 하나의 java파

일로 만들어도 가능하지만 이럴 경우 파일명을 main() 메서드가 있는 클래스를 파일명으로 해야 합니다.



프로그램을 처음 실행시켰을 때에는 위와 같은 화면이 나타납니다. 나타난 다이얼로그에 문자열을 입력하고 SET버튼을 클릭하면 아래와 같이 윈도우의 타이틀 바가 변경되는 것을 볼 수 있습니다.

Dialog 클래스의 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

- Dialog(Dialog owner), Dialog(Frame owner) : 이름이 없고, 주어진 프레임 또는 대화상자에 종속되고, non-modal인 대화상자를 생성합니다. 초기값은 보이지 않도록 설정됩니다.
- Dialog(Frame owner, boolean modal) : 이름이 없고, 주어진 프레임에 종속되고, 주어진 modal(true이면 modal, false이면 non-modal)의 대화상자를 생성하며 초기값은 보이지 않도록 설정됩니다.
- Dialog(Dialog owner, String title) : 주어진 이름을 갖고, 주어진 대화상자에 종속되고, non-modal인 대화상자를 생성합니다. 초기값은 보이지 않도록 설정됩니다.
- Dialog(Frame owner, String title) : 주어진 이름을 갖고, 주어진 프레임 에 종속되고, non-modal인 대화상자를 생성합니다. 초기값은 보이지 않도록 설정됩니다.
- Dialog(Dialog owner, String title, boolean modal) : 주어진 이름으로 대화상자에 종속되며 modal을 갖는 대화상자를 생성합니다. 초기값은 보이지 않도록 설정됩니다.
- Dialog(Frame owner, String title, boolean modal) : 주어진 이름으로 프레임에 종속되며 modal을 갖는 대화상자를 생성합니다. 초기값은 보이지 않도록 설정됩니다.

메서드

- String getTitle() : 대화상자의 제목을 반환합니다.
- boolean isModal() : 대화상자가 modal인지 여부를 반환합니다.
- boolean isResizable() : 대화상자가 임의로 크기 변화가 가능한지 여부를 반환합니다.
- void setModal(boolean b) : 대화상자의 modal을 주어진 값으로 설정합니다.

◦ void setResizable(boolean resizable) : 대화상자의 크기변화 여부를 주어진 값으로 설정합니다.
◦ void setTitle(String title) : 대화상자의 제목을 설정합니다.
◦ void show() : 대화상자를 보이지 않게 합니다.

다음은 다이얼로그 클래스를 만든 예입니다.

exam/java/chapter11/awt/TitleDialog.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class TitleDialog extends Dialog implements ActionListener{
7:
8:     private TextField field;
9:     private Button setButton;
10:    private Frame parent;
11:
12:    public TitleDialog(Frame f, String title) {
13:       super(f, title, false);
14:
15:       parent = f;
16:       setLayout(new FlowLayout());
17:
18:       field = new TextField(20);
19:       setButton = new Button("SET");
20:       setButton.addActionListener(this);
21:
22:       add(field);
23:       add(setButton);
24:
25:       pack();
26:    }
27:
28:    public void actionPerformed(ActionEvent e) {
29:       if( e.getActionCommand().equals("SET") ) {
30:          parent.setTitle(field.getText());
31:       }
32:       field.selectAll();
33:       setVisible(false);
34:    }
35: }
```

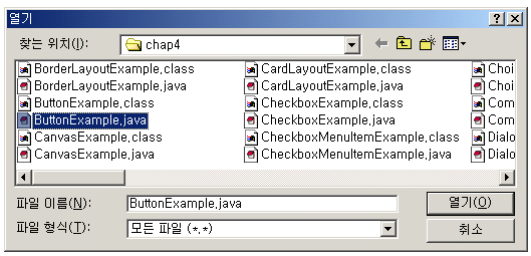
앞의 프로그램은 길이 20인 텍스트 필드와 버튼을 가진 대화상자를 나타냅니다. 텍스트 필드에 특정 문자열을 입력하고 "SET" 단추를 누르면 그 내용이 윈도우의 제목표시줄에 출력됩니다. 이 프로그램 하나만으로는 실행이 되지 않고, 다음에 나와 있는 DialogExample.java 프로그램 안에서 앞 클래스 파일의 인스턴스를 생성하여 실행시켜야 합니다. 즉 실행은 다음 프로그램을 이용합니다.

exam/java/chapter11/awt/DialogExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class DialogExample {
6:     private Frame f;
7:     private TitleDialog myTitleDialog;
8:
9:     public DialogExample() {
10:       f = new Frame();
11:       myTitleDialog = new TitleDialog(f, "Input title");
12:    }
13:
14:    public void launchFrame() {
15:       f.setLayout(new FlowLayout());
16:       myTitleBox.show();
17:
18:       f.setSize(250, 100);
19:       f.setVisible(true);
20:    }
21:
22:    public static void main(String[] args) {
23:       DialogExample de = new DialogExample();
24:       de.launchFrame();
25:    }
26: }
```

4.1.3.5. FileDialog

java.awt.FileDialog 클래스는 Dialog의 하위 클래스로 읽거나 쓸 파일을 선택하기 쉽게 도와줍니다. 윈도우 응용프로그램에서 파일을 읽거나 쓰려고 할 때 나타나는 대화상자와 같을 역할을 합니다.



FileDialog 클래스의 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

필드

- static int LOAD : 읽을 파일을 선택하기 위한 파일대화상자 다이얼로그 속성
- static int SAVE : 저장할 파일을 선택하기 위한 파일대화상자 다이얼로그 속성

생성자

- FileDialog(Frame parent) : 주어진 프레임에 속한 파일적재용(loading a file) 대화상자를 생성합니다.
- FileDialog(Frame parent, String title) : 주어진 프레임에 속하고 주어진 이름을 갖는 파일적재용(loading a file) 대화상자를 생성합니다.
- FileDialog(Frame parent, String title, int mode) : 주어진 프레임에 속하고 주어진 이름을 갖고 주어진 모드(loading/saving)의 파일 대화상자를 생성합니다.

메서드

- String getDirectory() : 파일 대화상자의 디렉터리를 반환합니다.
- String getFile() : 파일 대화상자에서 선택한 파일을 반환합니다.
- FilenameFilter getFilenameFilter() : 파일 대화상자의 파일 이름 필터를 반환합니다.
- int getMode() : 파일 대화상자의 용도가 읽기용(loading)인지 쓰기용(saving)인지를 반환합니다.
- void setDirectory(String dir) : 파일 대화상자의 디렉터리를 설정합니다.
- void setFile(String file) : 파일 대화상자의 파일을 설정합니다.
- void setFilenameFilter(FilenameFilter filter) : 파일 대화상자의 파일 이름 필터를 설정합니다.
- void setMode(int mode) : 파일 대화상자의 용도(mode)를 설정합니다. 필드로 선언된 LOAD 또는 SAVE를 인수 값으로 사용할 수 있습니다.

다음 프로그램은 FileDialog를 이용하여 선택한 파일명을 출력하는 예 입니다.

exam/java/chapter11/awt/FileDialogExample.java

```
1:  package exam.java.chapter11.awt;
2:
3:  import java.awt.*;
4:
5:  public class FileDialogExample{
6:
7:     private Frame f;
8:     private FileDialog loadDialog;
9:
10:    public FileDialogExample() {
11:       f = new Frame("FileDialog Example");
12:       loadDialog = new FileDialog(f, "열기", FileDialog.LOAD);
13:    }
14:
15:    public void launchFrame() {
16:       loadDialog.setVisible(true);
17:       System.out.println("File <" + loadDialog.getFile() + ">is Selected");
18:       f.pack();
19:       f.setVisible(true);
```

```
20:     }
21:
22:     public static void main(String[] args) {
23:        FileDialogExample fde = new FileDialogExample();
24:        fde.launchFrame();
25:     }
26: }
```

```
8:      private FileDialog loadDialog;
```

새로운 파일대화상자 객체를 정의합니다.

```
12:        loadDialog = new FileDialog(f, "열기", FileDialog.LOAD);
```

프레임 f에 속하면서 제목 표시줄에 "열기"이라는 문자열을 표시하고, 파일 불러오기 대화상자(FileDialog.LOAD)를 생성합니다. 저장하기 대화상자를 생성하기 위해서는 FileDialog.SAVE를 이용하면 됩니다.

```
16:        loadDialog.setVisible(true);
```

파일대화상자를 화면에 나타나게 합니다.

```
17:        System.out.println("File <" + loadDialog.getFile() + ">is Selected");
```

선택한 파일명을 출력합니다.

4.1.3.6. Applet

java.awt.Applet 컨테이너 클래스는 Panel클래스의 하위 클래스로 애플릿 프로그램을 만드는데 사용합니다. 애플릿이란 웹브라우저에서 실행되는 자바 프로그램입니다.

4.1.4. 레이아웃 관리자

레이아웃 관리자(Layout Manager)는 컨테이너 내에서 컴포넌트들의 배치를 결정하는데 사용합니다. 일반적으로 레이아웃 관리자는 컨테이너 내에서 컴포넌트들의 크기와 위치를 결정하는 일을 하기 때문에, 프로그래머가 컴포넌트들의 크기와 위치 등을 setLocation() 이나 setSize() 또는 setBounds() 등의 메서드를 이용하여 직접 설정해도 레이아웃 관리자는 이들 설정된 값을 무시할 수도 있습니다.

특별히 컴포넌트들의 크기와 위치를 직접 결정해야 할 경우라면 다음과 같이 컨테이너의 메서드를 이용하여 레이아웃 관리자를 사용할 수 없게 만들 수 있습니다.

```
setLayout(null);
```

이렇게 지정한 후 컴포넌트의 위치와 크기를 설정하려면 해당 컴포넌트에 대해 setLocation(), setSize() 또는 setBounds() 등의 메서드를 이용해야 하지만 시스템과 글꼴 크기의 차이 때문에 플랫폼과 관계없는 레이아웃이 만들어질 수 있습니다.

AWT에서 사용할 수 있는 레이아웃 관리자의 종류는 다음과 같습니다.

- FlowLayout - Panel 클래스의 기본 레이아웃 관리자이며, 컨테이너의 컴포넌트들을 문서의 단어들처럼 왼쪽에서 오른쪽으로, 그리고 위에서 아래로 배열합니다. 이 클래스는 한 행에 컴포넌트를 최대한 채워 넣은 후 그 다음 행으로 넘어갑니다.

- BorderLayout - Window, Dialog, Frame클래스의 기본 레이아웃 관리자이며, "NORTH", "SOUTH", "EAST", "WEST", "CENTER" 등의 이름으로 컨테이너에 추가된 컴포넌트를 배열합니다. 지정된 컴포넌트는 컨테이너의 위, 아래, 오른쪽, 왼쪽 그리고 중앙에 배열됩니다.

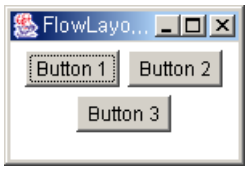
- GridLayout - 컨테이너를 지정된 수의 행과 열로 나눈 다음 그 행과 열에 왼쪽에서 오른쪽, 위에서 아래로 컴포넌트를 배치시킵니다. 컴포넌트들의 크기가 일정하게 나타납니다.

- CardLayout - 컴포넌트의 크기를 컨테이너 크기와 같게 한 번에 한 컴포넌트만 나타나게 합니다.

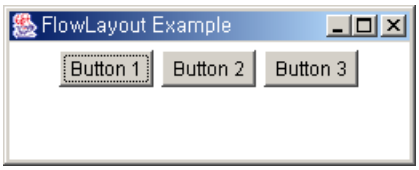
- GridBagLayout - java.awt 패키지에서 가장 복잡하면서도 가장 강력한 레이아웃 관리자입니다. 이 클래스는 컨테이너를 눈금모양의 행과 열로 나눈 다음 눈금 안에 넣고 필요하면 눈금의 크기를 조절하여 컴포넌트가 서로 겹쳐지지 않게 할 수 있습니다.

4.1.5. FlowLayout

FlowLayout 관리자는 Panel의 기본 레이아웃 관리자로서 컨테이너 내의 컴포넌트를 왼쪽에서 오른쪽, 위에서 아래로 차례대로 배치합니다. 즉, 문서 편집기에서 텍스트를 입력할 때 왼쪽에서 오른쪽으로 글이 입력되고, 한 줄이 다 차면 다음 줄로 넘어가는 방식처럼 컴포넌트를 차례대로 왼쪽에서 오른쪽을 배치하고, 한 줄이 넘어가면 새 줄을 시작하는 방식입니다.



크기 변경 전



크기 변경 후

다른 레이아웃 관리자와 달리 FlowLayout은 자신이 관리하는 컴포넌트의 크기를 제한하지 않고 컴포넌트의 크기가 최적이 되도록 합니다. 또 관리하는 영역의 크기가 변경되면 포함된 컴포넌트의 배치도 재설정 됩니다. 옵션을 사용하면 정렬상태를 왼쪽정렬이나 오른쪽정렬 방식으로 배열할 수도 있습니다. 또한 Inset을 지정하여 컴포넌트 사이의 테두리 영역을 더 크게 만들 수도 있습니다.

다음은 setLayout() 메서드를 이용하여 FlowLayout 관리자를 생성하는 예제 코드를 나타낸 것입니다.

```
setLayout(new FlowLayout(FlowLayout.RIGHT, 20, 40));
setLayout(new FlowLayout(FlowLayout.LEFT));
setLayout(new FlowLayout());
```

매개변수 FlowLayout.RIGHT는 컴포넌트를 오른쪽정렬 방식으로 설정하고, FlowLayout.LEFT는 왼쪽정렬방식으로 설정합니다. 매개변수 20과 40은 컴포넌트들 사이의 좌/우 여백(20)과 상/하 여백(40)을 의미합니다. 여백을 지정하지 않으면 기본 간격이 5픽셀로 설정됩니다.

FlowLayout 클래스의 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

필드

- static int LEFT : FlowLayout 관리자의 왼쪽정렬 속성
- static int CENTER : FlowLayout 관리자의 가운데정렬 속성
- static int RIGHT : FlowLayout 관리자의 오른쪽정렬 속성

◦ static int LEADING : FlowLayout 관리자의 리딩에지(leading edge) 정렬 속성
◦ static int TRAILING : FlowLayout 관리자의 트레일링에지(trailing edge) 정렬 속성

생성자

◦ FlowLayout() : 디폴트로 중앙정렬을 하고, 수평과 수직 간격이 5인 FlowLayout 관리자를 생성합니다.
◦ FlowLayout(int align) : 주어진 정렬을 하고, 수평과 수직 간격이 5인 FlowLayout 관리자를 생성합니다.
◦ FlowLayout(int align, int hgap, int vgap) : 주어진 정렬을 하고, 수평과 수직 간격이 각각 hgap과 vgap인 FlowLayout 관리자를 생성합니다.

메서드

◦ void addLayoutComponent(String name, Component comp) : 주어진 이름으로 컴포넌트를 추가합니다.
◦ void removeLayoutComponent(Component comp) : 컴포넌트를 제거합니다.
◦ int getAlignment() : 정렬 방식의 값을 반환합니다.
◦ void setAlignment(int align) : 정렬 방식을 설정합니다.
◦ void layoutContainer(Container target) : 컨테이너에 대한 레이아웃을 설정합니다.
◦ int getHgap() : 수평 간격을 반환합니다.
◦ int getVgap() : 수직 간격을 반환합니다.
◦ void setHgap(int hgap) : 수평 간격을 설정합니다.
◦ void setVgap(int vgap) : 수직 간격을 설정합니다.

다음 프로그램은 FlowLayout관리자를 지정한 Frame에 버튼을 배열하는 예입니다.

exam/java/chapter11/layout/FlowLayoutExample.java

```
1:  package exam.java.chapter11.layout;
2:
3:  import java.awt.*;
4:
5:  public class FlowLayoutExample {
6:     private Frame f;
7:
8:     public FlowLayoutExample() {
9:        f = new Frame("FlowLayout Example");
10:    }
11:
12:    public void launchFrame() {
13:       f.setLayout( new FlowLayout() );
14:       f.add(new Button("Button 1"));
15:       f.add(new Button("Button 2"));
16:       f.add(new Button("Button 3"));
17:       f.setSize(150, 100);
18:       f.setTitle("FlowLayout");
19:       f.setVisible(true);
20:    }
```

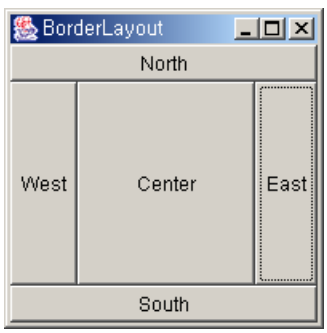
```
21:    public static void main(String[] args) {
22:       FlowLayoutExample fl = new FlowLayoutExample();
23:       fl.launchFrame();
24:    }
25: }
```

```
13:       f.setLayout( new FlowLayout() );
```

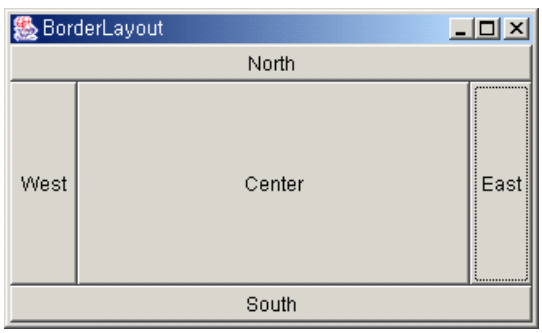
setLayout() 메서드를 이용하여 프레임의 레이아웃 관리자를 기본 정렬상태인 가운데 정렬, 그리고 좌/우, 상/하, 간격 5인 FlowLayout으로 지정합니다. setLayout() 메서드의 파라미터는 레이아웃 객체를 필요로 합니다.

4.1.6. BorderLayout

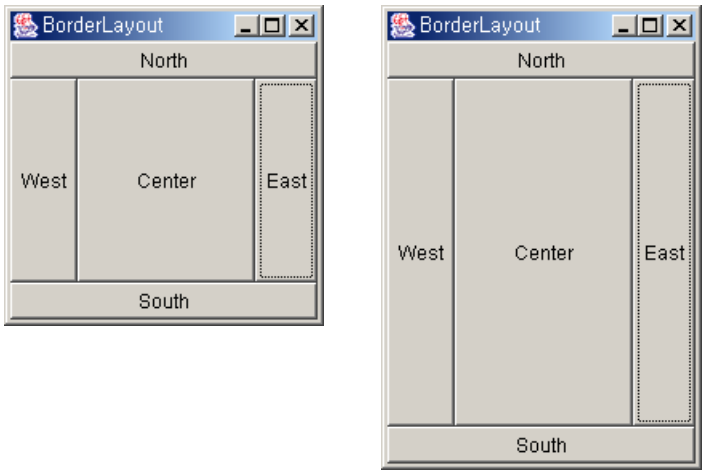
BorderLayout 관리자는 Window, Frame, Dialog의 기본 레이아웃 관리자입니다. 컨테이너에 포함된 컴포넌트는 North, South, East, West 및 Center 중에서 하나의 영역에 배치될 수 있습니다. North는 위를 차지하며, East는 오른쪽을 차지하고, Center영역은 North, South, East, West 등의 영역이 채워진 후에 남는 영역 전체입니다. 이들 영역에는 각각 하나의 컴포넌트만 추가할 수 있으며, 하나 이상의 컴포넌트를 추가해도 하나만 나타나게 됩니다. 만약 한 영역을 사용하지 않으면 그 영역의 최적크기는 0×0이 됩니다. 따라서 컴포넌트를 하나도 넣지 않으면 가운데 영역은 배경으로 나타나고, 주위영역은 0(Zero) 상태로 줄어들어 보이지 않게 됩니다.



크기 변경 전



가로방향 크기 변경 후



크기 변경 전 세로방향 크기 변경 후

BorderLayout 관리자를 사용할 때는 "North", "South", "East", "West", "Center" 등과 같은 문자열의 영역이름을 이용하여 위치를 지정할 수 있습니다. 이러한 영역은 미리 정의된 상수인 BorderLayout.NORTH, BorderLayout.SOUTH, BorderLayout.EAST, BorderLayout.WEST, BorderLayout.CENTER 등을 이용 할 수도 있습니다.

BorderLayout 클래스의 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

생성자

- BorderLayout() : 컴포넌트간의 수평간격과 수직간격이 0인 Border 레이아웃 관리자를 생성합니다.
- BorderLayout(int hgap, int vgap) : 컴포넌트간의 수평간격(hgap)과 수직간격(vgap)을 갖는 Border 레이아웃 관리자를 생성합니다.

메서드

- int getHgap() : 컴포넌트간의 수평간격(hgap)을 반환합니다.
- int getVgap() : 컴포넌트간의 수직간격(vgap)을 반환합니다.
- void setHgap(int hgap) : 컴포넌트간의 수평간격(hgap)을 설정합니다.
- void setVgap(int vgap) : 컴포넌트간의 수직간격(vgap)을 설정합니다.

다음 프로그램은 BorderLayout을 만드는 예입니다. 실행 후 창의 크기를 변경하여 BorderLayout 관리자가 어떻게 반응하는지 살펴보기 바랍니다.

exam/java/chapter11/layout/BorderLayoutExample.java

```
1:  package exam.java.chapter11.layout;
2:
3:  import java.awt.*;
4:
```

```
5:  public class BorderLayoutExample {
6:     private Frame f;
7:
8:     public BorderLayoutExample() {
9:        f = new Frame("BorderLayout Example");
10:    }
11:
12:    public void launchFrame() {
13:    // f.setLayout(new BorderLayout());
14:
15:       f.add(new Button("East"), BorderLayout.EAST);
16:       f.add(new Button("West"), BorderLayout.WEST);
17:       f.add(new Button("North"), BorderLayout.NORTH);
18:       f.add(new Button("Center"), BorderLayout.CENTER);
19:       f.add(new Button("South"), BorderLayout.SOUTH);
20:
21:       f.setSize(200, 200);
22:       f.setTitle("BorderLayout");
23:       f.setVisible(true);
24:    }
25:
26:    public static void main(String[] args) {
27:       BorderLayoutExample ble = new BorderLayoutExample();
28:       ble.launchFrame();
29:    }
30: }
```

```
13:    // setLayout(new BorderLayout());
```

주석처리하여 레이아웃관리자를 지정하지 않았는데 그 이유는 프레임 클래스의 기본 레이아웃관리자가 BorderLayout관리자이기 때문입니다. 즉, 이 예는 13라인이 주석 처리되건 그렇지 않건 상관없이 Border 레이아웃 관리자로 지정된다는 의미입니다.

```
15:       f.add(new Button("East"), BorderLayout.EAST);
```

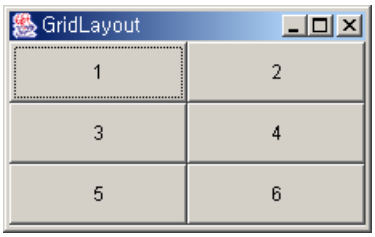
프레임에 버튼을 추가합니다. 위치는 프레임의 오른쪽에 추가합니다. BerderLayout.EAST 대신에 "East"문자열을 이용할 수도 있지만 권장하지는 않습니다.

4.1.7. GridLayout

GridLayout 관리자는 컨테이너의 영역을 주어진 행과 열 크기로 나누고, 같은 크기의 셀에 컴포넌트를 배치시키는 레이아웃 관리자입니다. 예를 들어 new GridLayout(3, 2)으로 Grid 레이아웃을 만들면 아래 그림처럼 여섯 개의 셀(3개의 행과 2개의 열)이 나타납니다.

이 때 모든 셀의 폭과 높이는 동일하게 지정됩니다. 나누어진 셀의 수보다 더 많거나 적게 컴포넌트를 삽입하면 행의 수는 고정되고, 열 수가 변하게 됩니다.

크기 변경 전



크기 변경 후

BorderLayout관리자와 마찬가지로 컨테이너 크기를 변경하면 컴포넌트의 상대적 위치는 변함이 없고, 컴포넌트의 크기만 변경됩니다.

GridLayout 클래스의 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

- GridLayout() : 하나의 행을 갖고 각 컴포넌트가 하나의 열을 차지하도록 Grid 레이아웃관리자를 생성합니다.
- GridLayout(int rows, int cols) : 주어진 행과 열을 갖는 Grid 레이아웃 관리자를 생성합니다.
- GridLayout(int rows, int cols, int hgap, int vgap) : 주어진 행과 열을 갖고 수평과 수직 간격이 각각 hgap과 vgap인 Grid 레이아웃 관리자를 생성합니다.

메서드

- void addLayoutComponent(String name, Component comp) : 주어진 이름으로 컴포넌트를 추가합니다.
- int getColumns() : 열의 크기를 반환합니다.
- int getHgap() : 수평간격을 반환합니다.
- int getRows() : 행의 크기를 반환합니다.
- int getVgap() : 수직간격을 반환합니다.
- void layoutContainer(Container parent) : 컨테이너를 레이아웃 합니다.
- void removeLayoutComponent(Component comp) : 주어진 컴포넌트를 제거합니다.
- void setColumns(int cols) : 열의 크기를 설정합니다.
- void setHgap(int hgap) : 수평간격을 설정합니다.
- void setRows(int rows) : 행의 크기를 설정합니다.
- void setVgap(int vgap) : 수직간격을 설정합니다.

다음 프로그램은 앞 그림의 GridLayout 관리자 사용 예입니다.

exam/java/chapter11/layout/GridLayoutExample.java

```
1:  package exam.java.chapter11.layout;
2:
```

```
3:  import java.awt.*;
4:
5:  public class GridLayoutExample {
6:     private Frame f;
7:
8:     public GridLayoutExample() {
9:        f = new Frame("GridLayout Example");
10:    }
11:
12:    public void launchFrame() {
13:       f.setLayout( new GridLayout(3, 2) );
14:
15:       f.add(new Button("1"));
16:       f.add(new Button("2"));
17:       f.add(new Button("3"));
18:       f.add(new Button("4"));
19:       f.add(new Button("5"));
20:       f.add(new Button("6"));
21:
22:       f.pack();
23:       f.setTitle("GridLayout");
24:       f.setVisible(true);
25:    }
26:
27:    public static void main(String[] args) {
28:       GridLayoutExample gle = new GridLayoutExample();
29:       gle.launchFrame();
30:    }
31: }
```

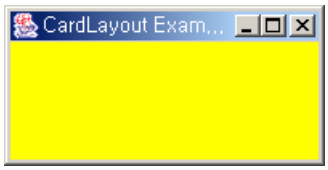
```
13:       f.setLayout( new GridLayout(3, 2) );
```

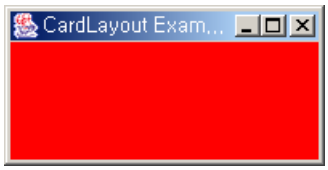
13라인에서는 3행 2열의 Grid 레이아웃을 설정합니다.

22라인의 pack() 메서드 호출은 프레임의 크기를 그 프레임 내에 포함된 모든 컴포넌트의 최적 크기와 일치되게 설정합니다. 따라서 프레임이 적절한 크기로 자동 설정됩니다.

4.1.8. CardLayout

인터페이스를 한 번에 한 장씩 나타나는 카드처럼 다룰 수 있습니다. 즉, 여러 장의 카드를 겹쳐 배치하는 식으로 관리하여 컨테이너가 포함한 컴포넌트를 한 장씩 볼 수 있게 합니다. 윈도우에서 탭 대화상자(Tabbed Dialog)를 나타낼 때 유용합니다. 여러 장으로 겹쳐진 경우 맨 처음 등록된 컴포넌트를 디폴트로 보여줍니다.





CardLayout 클래스의 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

생성자

- CardLayout() : 컴포넌트 간의 수평간격과 수직간격이 0인 카드레이아웃 관리자를 생성합니다.
- CardLayout(int hgap, int vgap) : 컴포넌트간의 수평간격이 hgap이고, 수직간격이 vgap인 카드레이아웃 관리자를 생성합니다.

메서드

- void addLayoutComponent(Component comp, Object constraints) : 주어진 컴포넌트를 카드레이아웃 관리자의 내부 이름테이블에 삽입합니다.
- void first(Container parent) : 주어진 컨테이너에 포함된 첫 번째 카드를 보여줍니다.
- void last(Container parent) : 주어진 컨테이너에 포함된 마지막 카드를 보여줍니다.
- void next(Container parent) : 주어진 컨테이너에 포함된 다음카드를 보여줍니다.
- void previous(Container parent) : 주어진 컨테이너에 포함된 이전카드를 보여줍니다.
- void show(Container parent, String name) : 주어진 컨테이너에 포함된 컴포넌트 중 주어진 이름을 가진 컴포넌트로 전환하여 보여줍니다.
- void removeLayoutComponent(Component comp) : 주어진 컴포넌트를 제거합니다.
- void layoutContainer(Container parent) : 이 카드 레이아웃 관리자를 이용하여 주어진 컨테이너에 포함된 컴포넌트를 레이아웃 시킵니다.
- int getHgap() : 컴포넌트간의 수평간격(hgap)을 반환합니다.
- int getVgap() : 컴포넌트간의 수직간격(vgap)을 반환합니다.
- void setHgap(int hgap) : 컴포넌트간의 수평간격(hgap)을 설정합니다.
- void setVgap(int vgap) : 컴포넌트간의 수직간격(vgap)을 설정합니다.

다음 프로그램은 앞의 그림처럼 마우스를 클릭 할 때마다 두 개의 Panel을 번갈아 보여주는 예제입니다.

exam/java/chapter11/layout/CardLayoutExample.java

```
1:  package exam.java.chapter11.layout;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class CardLayoutExample implements MouseListener {
7:     private Frame f;
8:     private Panel cardPanel, p1, p2;
```

```
9:     private CardLayout myCardLayout;
10:
11:    public CardLayoutExample() {
12:       f = new Frame("CardLayout Example");
13:       myCardLayout = new CardLayout();
14:       cardPanel = new Panel();
15:       p1 = new Panel();
16:       p2 = new Panel();
17:    }
18:
19:    public void launchFrame() {
20:       cardPanel.setLayout(myCardLayout);
21:
22:       p1.setBackground(Color.yellow);
23:       p2.setBackground(Color.red);
24:       p1.addMouseListener(this);
25:       p2.addMouseListener(this);
26:
27:       cardPanel.add(p1, "First");
28:       cardPanel.add(p2, "Second");
29:
30:       f.add(cardPanel);
31:       f.setSize(200, 100);
32:       f.setVisible(true);
33:    }
34:
35:    public static void main(String[] args) {
36:       CardLayoutExample cle = new CardLayoutExample();
37:       cle.launchFrame();
38:    }
39:    public void mousePressed(MouseEvent me) {
40:       myCardLayout.next(cardPanel);
41:    }
42:    public void mouseClicked(MouseEvent me) {}
43:    public void mouseReleased(MouseEvent me) {}
44:    public void mouseEntered(MouseEvent me) {}
45:    public void mouseExited(MouseEvent me) {}
46: }
```

```
6:  public class CardLayoutExample implements MouseListener {
```

마우스 이벤트를 사용하기 위해 MouseListener 인터페이스를 구현하였습니다. 이벤트에 대한 자세한 내용은 뒤에서 자세히 다루기로 하겠습니다.

```
8:     private Panel cardPanel, p1, p2;
9:     private CardLayout myCardLayout;
```

패널 객체와 CardLayout객체를 선언합니다. 선언된 패널 객체들 중에서 cardPanel객체는 또 다른 패널 객체인 p1과 p2객체를 포함시킬 패널입니다.

```
13:        myCardLayout = new CardLayout();
```

9라인에서 선언한 CardLayout 객체의 인스턴스를 생성합니다.

```
14:        cardPanel = new Panel();
15:        p1 = new Panel();
16:        p2 = new Panel();
```

6라인에서 선언한 Panel의 인스턴스를 생성합니다.

```
20:        cardPanel.setLayout(myCardLayout);
```

CardLayout객체의 인스턴스인 myCardLayout을 이용하여 cardPanel객체의 레이아웃 관리자를 CardLayout으로 설정합니다.

```
22:        p1.setBackground(Color.yellow);
23:        p2.setBackground(Color.red);
```

Panel객체 p1의 배경색은 노란색으로, p2의 배경색은 빨간색으로 설정합니다.

```
24:        p1.addMouseListener(this);
25:        p2.addMouseListener(this);
```

Panel객체 p1과 p2에 마우스 이벤트 처리 객체를 등록시켜 p1과 p2를 이벤트 발생시키는 객체로 만듭니다.

```
26:        cardPanel.add(p1, "First");
27:        cardPanel.add(p2, "Second");
```

Panel객체 p1과 p2를 6라인에서 선언된 Panel객체인 cardPanel에 부착합니다.

```
30:        f.add(cardPanel);
```

cardPanel객체를 프레임에 부착합니다.

```
39:    public void mousePressed(MouseEvent me) {
40:        myCardLayout.next(cardPanel);
41:    }
```

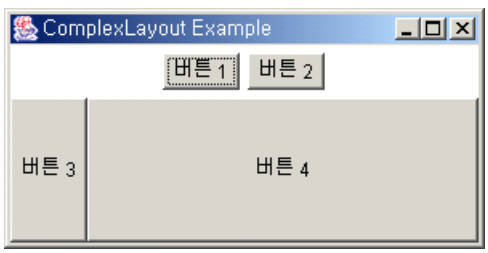
이벤트 처리 루틴을 구현한 코드입니다. 마우스가 눌려질 때마다 호출되는 메서드입니다. next()메서드는 다음 컴포넌트를 보여줍니다.

```
42:     public void mouseClicked(MouseEvent me) {}
```

42라인부터 45라인까지 메서드들은 이벤트 처리에 사용하지 않더라도 모두 구현해야 합니다.

4.1.9. 복합 레이아웃(Complex Layout)

프레임과 패널은 AWT에서 필수 컨테이너라고 할 수 있습니다. 프레임은 제목, 테두리, 크기조절용 모서리 등이 있는 "최상위 레벨" 윈도우입니다. 윈도우의 모양과 동작 방식은 사용하는 플랫폼에 따라 다르며, setLayout() 메서드를 사용하여 레이아웃 관리자를 지정하지 않으면 BorderLayout이 프레임의 기본 레이아웃 관리자가 됩니다. 일반적으로 GUI 환경으로 프로그램을 작성할 때는 적어도 하나의 프레임을 사용하게 되지만 하나의 코드 내에서 여러 프레임을 사용할 수도 있습니다. 프레임의 상위 클래스인 Window도 BorderLayout을 기본 레이아웃 관리자로 지정하고 있습니다.
패널에서는 setLayout() 메서드를 명시적으로 사용하지 않으면 FlowLayout 관리자가 사용됩니다. 패널은 독립적인 모양도 없고 독립적인 창으로 사용할 수도 없지만, 컴포넌트(버튼, 레이블 등) 뿐만 아니라 다른 컨테이너(패널, 프레임 등)를 포함할 수 있습니다. 패널을 컨테이너에 넣으면 독립적으로 다른 레이아웃 관리자를 사용할 수 있습니다. 패널 클래스의 하위 클래스인 Applet도 FlowLayout을 기본 관리자로 지정하고 있습니다.



그림에서 "버튼 1"과 "버튼 2"가 있는 패널은 FlowLayout관리자가 적용되었고, "버튼 3"과 "버튼 4"가 있는 패널에는 BorderLayout관리자가 적용되었습니다. 소스코드는 다음과 같습니다.

exam/java/chapter11/layout/ComplexLayoutExample.java

```
1:  package exam.java.chapter11.layout;
2:
3:  import java.awt.*;
4:
5:  public class ComplexLayoutExample {
```

```
6:     private Frame f;
7:     private Panel p1, p2;
8:     private Button b1, b2, b3, b4;
9:
10:    public ComplexLayoutExample() {
11:       f = new Frame("ComplexLayout Example");
12:       p1 = new Panel();
13:       p2 = new Panel();
14:       b1 = new Button("버튼 1");
15:       b2 = new Button("버튼 2");
16:       b3 = new Button("버튼 3");
17:       b4 = new Button("버튼 4");
18:    }
19:
20:    public void launchFrame() {
21:       p1.setLayout(new FlowLayout());
22:       p2.setLayout(new BorderLayout());
23:
24:       p1.add(b1);
25:       p1.add(b2);
26:       p2.add(b3, BorderLayout.WEST);
27:       p2.add(b4, BorderLayout.CENTER);
28:
29:       f.add(p1, BorderLayout.NORTH);
30:       f.add(p2, BorderLayout.CENTER);
31:       f.setSize(300, 150);
32:       f.setVisible(true);
33:    }
34:
35:    public static void main(String[] args) {
36:       ComplexLayoutExample cl = new ComplexLayoutExample();
37:       cl.launchFrame();
38:    }
39: }
```

```
12:       Panel p1 = new Panel();
13:       Panel p2 = new Panel();
```

두 개의 패널 객체를 생성합니다.

```
21:       p1.setLayout(new FlowLayout());
22:       p2.setLayout(new BorderLayout());
```

12라인과 13라인에서 생성된 패널객체 p1을 FlowLayout 관리자로 지정하고, p2에는 BorderLayout 관리자로 지정합니다.

```
24:       p1.add(b1);
```

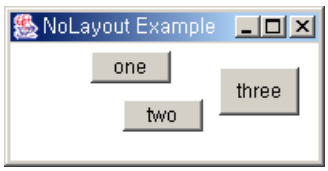
13라인에서 만들어진 버튼 객체를 패널 p1에 포함시킵니다. p1의 레이아웃은 FlowLayout 입니다.

```
26:         p2.add(b3, BorderLayout.WEST);
```

18라인에서 만들어진 버튼 객체를 p2에 포함시킵니다. p2패널의 레이아웃 관리자가 BorderLayout이므로 add() 메서드에 위치를 명시해 줍니다.

4.1.10. 레이아웃 관리자를 사용하지 않는 레이아웃

원하는 곳에 직접 컴포넌트를 배치하려면 먼저 레이아웃을 사용하지 않아야 하므로 레이아웃 관리자를 null(setLayout(null))로 설정합니다. 그리고 각 컴포넌트의 크기와 위치 결정은 setSize(), setLocation() 메서드 또는 setBounds() 메서드 등을 이용하여 결정합니다. 레이아웃 관리자를 지정하지 않을 경우 플랫폼에 따라 의도한 것과 다르게 나타날 수 있으므로 주의해야 합니다.



컴포넌트의 크기와 위치에 관련된 주요 메서드를 살펴보면 다음과 같습니다.

- void setLocation(int x, int y) : 컴포넌트를 새로운 위치로 옮깁니다.
- void setLocation(Point p) : 컴포넌트를 새로운 위치로 옮깁니다.
- void setSize(Dimension d) : 컴포넌트의 폭과 높이를 각각 d.width와 d.height로 설정합니다.
- void setSize(int width, int height) : 컴포넌트의 폭과 높이를 각각 width와 height로 설정합니다.
- void setBounds(int x, int y, int width, int height) : 컴포넌트를 주어진 위치로 옮기고 크기를 변경합니다.
- void setBounds(Rectangle r) : 컴포넌트를 주어진 위치로 옮기고 크기를 변경합니다.

다음 프로그램은 앞의 NoLayoutExample을 출력하기 위한 예를 보인 것으로 레이아웃 관리자를 사용하지 않고, 컴포넌트의 위치를 직접 설정해 주는 방법을 보여주는 예입니다.

exam/java/chapter11/layout/NoLayoutExample.java

```
1:  package exam.java.chapter11.layout;
2:
3:  import java.awt.*;
4:
5:  public class NoLayoutExample {
6:     private Frame f;
```

```
7:     private Button b1, b2, b3;
8:     private Insets insets;
9:
10:    public NoLayoutExample() {
11:       f = new Frame("NoLayout Example");
12:       b1 = new Button("one");
13:       b2 = new Button("two");
14:       b3 = new Button("three");
15:    }
16:
17:    public void launchFrame() {
18:       f.setLayout(null);
19:       f.setSize(200, 100);
20:       f.setVisible(true);
21:       insets = f.getInsets();
22:       b1.setBounds(50 + insets.left, 5 + insets.top, 50, 20);
23:       b2.setLocation(new  Point(70 + insets.left, 35 + insets.top));
24:       b2.setSize(new Dimension(50, 20));
25:       b3.setLocation(130 + insets.left, 15 + insets.top);
26:       b3.setSize(50, 30);
27:       f.add(b1);  f.add(b2);  f.add(b3);
28:    }
29:
30:    public static void main(String[] args) {
31:       NoLayoutExample nt = new NoLayoutExample();
32:       nt.launchFrame();
33:    }
34: }
```

```
8:     private Insets insets;
```

Insets 객체를 이용해 프레임 경계선의 두께를 구하기 위해 객체를 선언합니다.

```
18:       f.setLayout(null);
```

레이아웃 관리자를 사용하지 않도록 합니다.

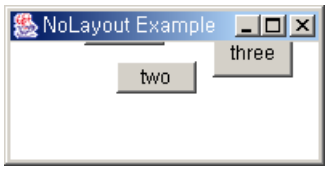
```
19:       f.setSize(200, 100);
```

프레임 윈도우의 크기를 가로 200픽셀 세로 100픽셀로 지정합니다.

```
20:       f.setVisible(true);
```

자바에서 테두리를 구하려면 먼저 컴포넌트가 보이는(visible) 상태여야 합니다. 그렇지 않으면 테두리(inset) 값은 모두 0이 됩니다. 그러므로 이 라인을 20라인 이후로 옮겨 놓고

실행시키면 다음과 같은 결과를 얻을 수 있습니다.




```
21:         insets = getInsets();
```

프레임의 경계선 두께를 얻습니다.

```
22:         b1.setBounds(50 + insets.left, 5 + insets.top, 50, 20);
```

setBound()는 컴포넌트의 크기와 위치를 동시에 지정할 수 있으므로 가로 50픽셀, 세로 20픽셀로 크기를 지정하고, 위치는 원점(0, 0)에서 (50, 5)에 컴포넌트의 왼쪽 상단 모서리를 배치합니다.

```
23:         b2.setLocation(new  Point(70 + insets.left, 35 + insets.top));
```

setLocation() 메서드는 컴포넌트의 위치를 지정해 주는 메서드로 좌표(70, 35)에 배치합니다. 여기서는 Point클래스(Point클래스는 x좌표와 y좌표를 가지고 있습니다.)를 이용하여 위치를 지정했지만 23라인처럼 Point클래스를 사용하지 않아도 됩니다.

```
24:         b2.setSize(new Dimension(50, 20));
```

setSize() 메서드는 컴포넌트의 크기를 지정해 주는 메서드입니다. 가로 50픽셀, 세로 20픽셀 크기의 컴포넌트를 생성합니다. 여기서는 Dimension 클래스를 이용했지만 24라인처럼 Dimension 클래스를 사용하지 않아도 됩니다.

```
27:         f.add(b1);  f.add(b2);  f.add(b3);
```

버튼 컴포넌트를 프레임에 붙입니다.

Inset 클래스

• 컨테이너에서 테두리를 나타내는 클래스로 레이아웃을 지정할 때 유용합니다. 즉, 컨테이너 영역 중에서 실제 컴포넌트를 표현하거나 작업할 수 있는 영역은 inset이 가리키는 테두리 영역이라 할 수 있습니다.

Inset 클래스는 다음과 같은 변수와 메서드를 제공합니다.

변수

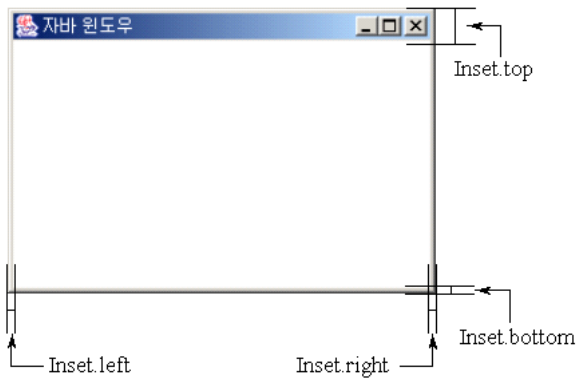
◦ int bottom : 테두리 아래쪽크기를 나타냅니다.
◦ int left : 테두리 왼쪽크기를 나타냅니다.
◦ int right : 테두리 오른쪽크기를 나타냅니다.
◦ int top : 테두리 위쪽크기를 나타냅니다.

생성자

◦ Insets(int top, int left, int bottom, int right) : 주어진 크기의 inset을 생성합니다.

메서드

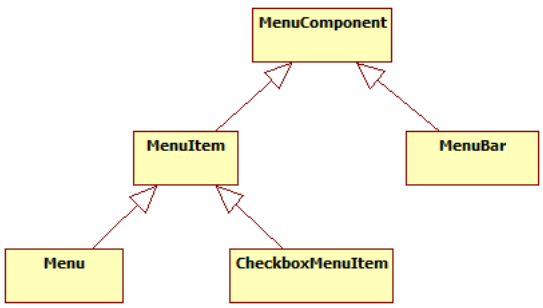
◦ String toString() : inset 객체를 문자열로 표현하여 반환합니다.



4.2. 메뉴(Menu)

AWT에서 메뉴를 구성하기 위해 몇 개의 클래스에서 제공하고 있습니다. 메뉴와 관련된 클래스들을 이용하여 자바 애플리케이션 또는 애플릿에서 원하는 형태의 메뉴를 구성할 수 있습니다. 윈도우는 몇 개의 메뉴를 갖고 각 메뉴는 다시 하나 이상의 메뉴, 메뉴 아이템, 메뉴 분리자 등을 갖습니다.
메뉴는 다른 컴포넌트(Button, TextField 등)처럼 흔히 사용하는 컨테이너에 추가할 수 없을 뿐만 아니라 레이아웃 관리자를 사용할 수도 없으며, 메뉴 컨테이너(Menu container)에만 추가할 수 있습니다. 이때 사용하는 메서드가 setMenuBar()이며, setMenuBar()를 이용하여 메뉴 표시줄을 만들면 메뉴 "트리"가 만들어집니다. 다시 메뉴에는 메뉴 항목뿐만 아니라 또 다른 메뉴가 추가될 수 있습니다.

다음은 AWT에서 제공되는 메뉴 컴포넌트 클래스의 계층 구조를 나타낸 것입니다.



4.2.1. MenuComponent

메뉴 클래스들 중에서 가장 상위 클래스인 MenuComponent는 메뉴를 제공하기 위한 기본적인 기능을 정의하며 실제 클래스 자체를 사용하지는 않습니다. 다른 메뉴 관련 클래스는 이 클래스를 상속받는 하위클래스들 입니다.

MenuComponent 클래스가 제공하는 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

생성자

◦ MenuComponent() : MenuComponent 객체를 생성합니다.

메서드

◦ Font getFont() : 메뉴 컴포넌트에 사용된 폰트를 반환합니다.

◦ String getName() : 메뉴 컴포넌트의 이름을 반환합니다.
◦ MenuContainer getParent() : 메뉴 컴포넌트의 부모 컨테이너를 반환합니다.
◦ void setFont(Font f) : 메뉴 컴포넌트가 사용할 폰트를 설정합니다.
◦ void setName(String name) : 주어진 문자열로 메뉴 컴포넌트의 이름을 설정합니다.

4.2.2. MenuBar

MenuBar 컴포넌트는 수평 메뉴로서 프레임 객체에만 추가할 수 있습니다. 메뉴바 컴포넌트에서는 발생하는 모든 이벤트가 정상적인 메뉴 동작의 일부로 처리되기 때문에 리스너(Listener)를 지원하지 않습니다. 메뉴바 컴포넌트에 setHelpMenu(Menu) 메서드를 이용하면 [Help]메뉴를 지정할 수 있습니다. 메뉴바는 프레임에 메뉴들을 추가하기 위한 기본적으로 필요로 하는 요소입니다. 메뉴바는 그 자체로는 아무 의미가 없으며 메뉴가 추가되었을 때에 그 메뉴들을 나타내는데 사용합니다.

MenuBar 클래스가 제공하는 객체 생성자와 메서드를 살펴보면 다음과 같습니다.

생성자

◦ MenuBar() : 메뉴바를 생성합니다.

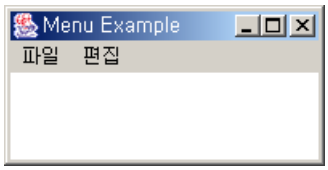
메서드

◦ Menu add(Menu m) : 메뉴바에 주어진 메뉴를 추가합니다.
◦ void deleteShortcut(MenuShortcut s) : 주어진 메뉴단축키를 삭제합니다.
◦ Menu getHelpMenu() : 메뉴바의 도움말 메뉴를 반환합니다.
◦ Menu getMenu(int i) : 주어진 번호의 메뉴를 반환합니다.
◦ int getMenuCount() : 메뉴바에 있는 메뉴의 개수를 반환합니다.
◦ MenuItem getShortcutMenuItem(MenuShortcut s) : 주어진 메뉴 단축키에 연결된 메뉴 아이템을 반환합니다.
◦ void remove(int index) : 주어진 인덱스에 해당하는 메뉴를 제거합니다.
◦ void remove(MenuComponent m) : 주어진 메뉴 컴포넌트를 제거합니다.
◦ void setHelpMenu(Menu m) : 도움말 메뉴를 설정합니다.
◦ Enumeration shortcuts() : 메뉴바에 등록된 메뉴단축키를 반환합니다.

4.2.3. Menu

Menu 컴포넌트는 기본적인 풀다운 메뉴를 만드는데 사용하며, MenuItem클래스의 하위 클래스이기 때문에 메뉴를 다른 메뉴에 추가하면 새로운 메뉴를 생성할 수 있습니다. 즉,

MenuBar에 추가하여 메뉴를 나타내거나, 다른 Menu에 추가하여 서브 메뉴를 구성할 수 있습니다. 다음 그림은 두 개의 Menu객체를 추가한 결과화면입니다.



Menu 클래스의 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

생성자

- Menu() : 레이블이 없는 메뉴를 생성합니다.
- Menu(String label) : 주어진 레이블을 갖는 메뉴를 생성합니다.
- Menu(String label, boolean tearOff) : 주어진 레이블을 갖고, 메뉴가 tear-off 될 수 있는지 여부를 설정하여 메뉴를 생성합니다.

메서드

- MenuItem add(MenuItem mi) : 메뉴 아이템을 추가합니다.
- void add(String label) : 주어진 이름의 메뉴 아이템을 추가합니다.
- void addSeparator() : 메뉴 분리자를 추가합니다.
- MenuItem getItem(int index) : 주어진 인덱스에 해당하는 메뉴 아이템을 반환합니다.
- int getItemCount() : 메뉴에 추가된 메뉴 아이템의 개수를 반환합니다.
- void insert(MenuItem menuitem, int index) : 주어진 인덱스에 메뉴 아이템을 추가합니다.
- void insert(String label, int index) : 주어진 인덱스에 주어진 이름의 메뉴 아이템을 추가합니다.
- void insertSeparator(int index) : 주어진 인덱스에 메뉴 분리자를 추가합니다.
- boolean isTearOff() : 이 메뉴가 tear-off 메뉴인지를 반환합니다.
- void remove(int index) : 주어진 인덱스에 해당하는 메뉴 아이템을 제거합니다.
- void remove(MenuComponent item) : 주어진 메뉴 아이템을 제거합니다.
- void removeAll() : 등록된 모든 메뉴 아이템을 제거합니다.

다음 프로그램은 위의 그림과 같이 윈도우의 프레임에 메뉴를 붙이는 예제입니다.

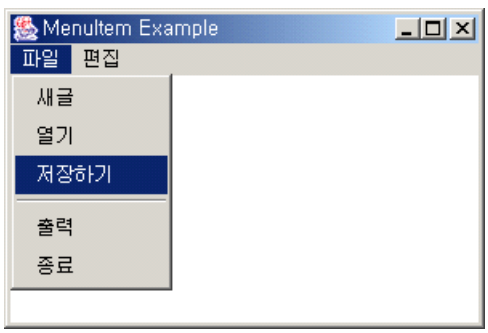
exam/java/chapter11/menu/MenuExample.java

```
1:  package exam.java.chapter11.menu;
2:
3:  import java.awt.*;
4:
5:  public class MenuExample {
6:     private Frame f;
7:     private MenuBar mb;
8:     private Menu fileMenu, editMenu;
9:
```

```
10:     public MenuExample() {
11:         f = new Frame("Menu Example");
12:         mb = new MenuBar();
13:         fileMenu = new Menu("파일");
14:         editMenu = new Menu("편집");
15:     }
16:
17:     public void launchFrame() {
18:         f.setMenuBar(mb);
19:         mb.add(fileMenu);
20:         mb.add(editMenu);
21:         f.setSize(200, 100);
22:         f.setVisible(true);
23:     }
24:
25:     public static void main (String[] args) {
26:         MenuExample mt = new MenuExample();
27:         mt.launchFrame();
28:     }
29: }
```

4.2.4. MenuItem

MenuItem 클래스는 Menu에 추가된 각 아이템을 만드는 클래스입니다.



MenuItem 클래스의 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

- MenuItem() : 레이블이 없고, 메뉴단축키가 없는 메뉴 아이템을 생성합니다.
- MenuItem(String label) : 주어진 레이블을 갖고, 연결된 메뉴단축키가 없는 아이템을 생성합니다.
- MenuItem(String label, MenuShortcut s) : 주어진 레이블을 갖고, 주어진 메뉴단축키가 연결된 메뉴 아이템을 생성합니다.

메서드

- void deleteShortcut() : 연결된 메뉴단축키를 제거합니다.
- String getLabel() : 메뉴 아이템의 레이블을 반환합니다.
- MenuShortcut getShortcut() : 메뉴 아이템에 연결된 메뉴단축키 객체를 반환합니다.
- boolean isEnabled() : 아이템이 활성화되어 있는지를 반환합니다.
- void setEnabled(boolean b) : 메뉴 아이템을 활성화 또는 비활성화 되도록 설정합니다.
- void setLabel(String label) : 메뉴 아이템의 레이블을 설정합니다.
- void setShortcut(MenuShortcut s) : 메뉴 아이템의 메뉴단축키 객체를 설정합니다.

다음 프로그램은 앞의 그림과 같은 결과를 얻도록 메뉴에 메뉴항목을 붙이는 예제입니다. 앞의 예제파일인 MenuExample.java에 몇 라인만 추가하면 결과를 확인할 수 있습니다.

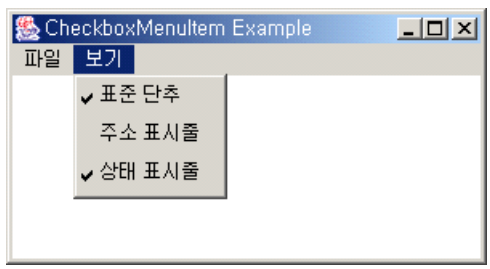
exam/java/chapter11/menu/MenuItemExample.java

```
1:  package exam.java.chapter11.menu;
2:
3:  import java.awt.*;
4:
5:  public class MenuItemExample {
6:     private Frame f;
7:     private MenuBar mb;
8:     private Menu fileMenu, editMenu;
9:     private MenuItem newItem, openItem, saveItem, printItem, quitItem;
10:
11:    public MenuItemExample() {
12:       f = new Frame("Menu Example");
13:       mb = new MenuBar();
14:       fileMenu = new Menu("파일");
15:       editMenu = new Menu("편집");
16:       newItem = new MenuItem("새글");
17:       openItem = new MenuItem("열기");
18:       saveItem = new MenuItem("저장하기");
19:       printItem = new MenuItem("출력");
20:       quitItem = new MenuItem("종료");
21:    }
22:
23:    public void launchFrame() {
24:       f.setMenuBar(mb);
25:       mb.add(fileMenu);
26:       fileMenu.add(newItem);
27:       fileMenu.add(openItem);
28:       fileMenu.add(saveItem);
29:       fileMenu.addSeparator();
30:       fileMenu.add(printItem);
31:       fileMenu.add(quitItem);
32:
33:       mb.add(editMenu);
```

```
34:         f.setSize(300, 200);
35:         f.setVisible(true);
36:     }
37:
38:     public static void main (String[] args) {
39:         MenuItemExample mt = new MenuItemExample();
40:         mt.launchFrame();
41:     }
42: }
```

4.2.5. CheckboxMenuItem

CheckboxMenuItem는 체크박스를 포함하는 메뉴 아이템을 만드는데 사용하며 선택 항목은 메뉴에 나열됩니다. CheckboxMenuItem은 ItemListener 인터페이스를 통해 제어가 이루어지기 때문에 상태가 바뀌면 itemStateChanged() 메서드가 호출됩니다.



CheckboxMenuItem의 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

생성자

- CheckboxMenuItem() : 레이블이 없는 체크박스 메뉴 아이템을 생성합니다.
- CheckboxMenuItem(String label) : 주어진 레이블을 갖는 체크박스 메뉴 아이템을 생성합니다.
- CheckboxMenuItem(String label, boolean state) : 주어진 레이블과 선택 상태를 갖는 체크박스 메뉴 아이템을 생성합니다.

메서드

- boolean getState() : 체크박스 메뉴 아이템의 선택 상태를 반환합니다.
- void setState(boolean b) : 체크박스 메뉴 아이템을 주어진 상태로 설정합니다.

다음 프로그램은 CheckboxMenuItem을 구현한 예입니다.
exam/java/chapter11/menu/CheckboxMenuItemExample.java

```
1:  package exam.java.chapter11.menu;
2:
3:  import java.awt.*;
4:
5:  public class CheckboxMenuItemExample {
6:     private Frame f;
7:     private MenuBar mb;
8:     private Menu fileMenu, viewMenu;
9:     private CheckboxMenuItem stdTool, addrTool, statusTool;
10:
11:    public CheckboxMenuItemExample() {
12:       f = new Frame("CheckboxMenuItem Example");
13:       mb = new MenuBar();
14:       fileMenu = new Menu("파일");
15:       viewMenu = new Menu("보기");
16:       stdTool = new CheckboxMenuItem("표준 단추");
17:       addrTool = new CheckboxMenuItem("주소 표시줄");
18:       statusTool = new CheckboxMenuItem("상태 표시줄");
19:    }
20:
21:    public void launchFrame() {
22:       f.setMenuBar(mb);
23:       mb.add(fileMenu);
24:       mb.add(viewMenu);
25:       viewMenu.add(stdTool);
26:       viewMenu.add(addrTool);
27:       viewMenu.add(statusTool);
28:       f.setSize(300, 160);
29:       f.setVisible(true);
30:    }
31:
32:    public static void main (String[] args) {
33:       CheckboxMenuItemExample ct = new CheckboxMenuItemExample();
34:       ct.launchFrame();
35:    }
36: }
```

4.2.6. 팝업메뉴(PopupMenu)

팝업메뉴(PopupMenu)는 독립형 메뉴를 만드는데 사용하며 메뉴(Menu)나 메뉴항목(MenuItem)을 추가할 수 있습니다. 이것은 컨테이너에 일반적인 컴포넌트를 넣는 것과는 달리 반드시 상위 컴포넌트에 붙여야 합니다. 팝업메뉴를 화면에 나타나게 하려면 show() 메서드를 호출해야 하고, 팝업메뉴가 나타날 x좌표와, y좌표의 위치를 지정해야 합니다. 위치지정은 동작을 시작시키는 컴포넌트를 이용합니다.

PopupMenu가 제공하는 객체 생성자와 주요 메서드는 다음과 같습니다.

생성자

- PopupMenu() : 팝업메뉴를 생성합니다.
- PopupMenu(String label) : 주어진 이름을 갖는 팝업메뉴를 생성합니다.

메서드

- void show(Component origin, int x, int y) : 팝업 메뉴를 주어진 컴포넌트의 해당 위치에 나타나게 합니다.

다음 프로그램은 팝업메뉴를 만드는 예제입니다. 마우스 오른쪽 버튼을 누르면 팝업 메뉴가 나타나게 하기 위해 이벤트 처리를 하였습니다. 이벤트에 대한 자세한 내용은 뒤에서 설명하기로 하겠습니다.

exam/java/chapter11/menu/PopupMenuExample.java

```
1:  package exam.java.chapter11.menu;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class PopupMenuExample
7:        extends MouseAdapter
8:        implements ActionListener {
9:
10:    private Frame f;
11:    private PopupMenu popupMenu;
12:    private Menu editMenu;
13:    private MenuItem cancelItem, redoItem, quitItem;
14:    private MenuItem copyItem, pasteItem;
15:
16:    public PopupMenuExample() {
17:        f = new Frame("PopupMenu Example");
18:        popupMenu = new PopupMenu();
19:        editMenu = new Menu("편집");
20:        cancelItem = new MenuItem("실행 취소");
21:        redoItem = new MenuItem("재 실행");
22:        copyItem = new MenuItem("복사하기");
23:        pasteItem = new MenuItem("붙여넣기");
```

```
24:        quitItem = new MenuItem("종료");
25:     }
26:
27:     public void launchFrame() {
28:        popupMenu.add(cancelItem);
29:        redoItem.setEnabled(false);
30:        popupMenu.add(redoItem);
31:        popupMenu.addSeparator();
32:        editMenu.add(copyItem);
33:        editMenu.add(pasteItem);
34:        popupMenu.add(editMenu);
35:        popupMenu.addSeparator();
36:        popupMenu.add(quitItem);
37:        f.add(popupMenu);
38:        f.setSize(250, 160);
39:
40:        copyItem.addActionListener(this);
41:        pasteItem.addActionListener(this);
42:
43:        f.addMouseListener(this);
44:        f.addWindowListener( new WindowAdapter() {
45:           public void windowClosing(WindowEvent e) {
46:              System.exit(0);
47:           }
48:        } );
49:        f.setVisible(true);
50:     }
51:
52:     public static void main(String[] args) {
53:        PopupMenuExample pme = new PopupMenuExample();
54:        pme.launchFrame();
55:     }
56:
57:     public void mouseClicked(MouseEvent e) {
58:        if(e.getModifiers() == MouseEvent.BUTTON3_MASK) {
59:           popupMenu.show(f, e.getX(), e.getY());
60:        }
61:     }
62:
63:     public void actionPerformed(ActionEvent e) {
64:        if(e.getSource().equals(copyItem)) {
65:           System.out.println("Selected copy item");
66:        } else if(e.getSource().equals(pasteItem)) {
67:           System.out.println("Selected paste item");
68:        }
69:     }
70: }
```

Menu 클래스를 상속받아 메뉴를 생성하므로 사용법은 메뉴 컴포넌트에서와 같습니다.

```
18:         popupMenu = new PopupMenu();
```

PopupMenu객체를 생성합니다.

```
42:         f.add(popupMenu);
```

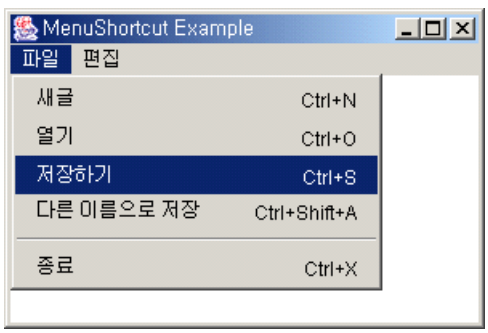
프레임에 팝업메뉴를 추가합니다.

```
57:     public void mouseClicked(MouseEvent e) {
58:         if(e.getModifiers() == MouseEvent.BUTTON3_MASK) {
59:             popupMenu.show(f, e.getX(), e.getY());
60:         }
61:     }
```

마우스가 클릭되었고, 클릭된 마우스의 버튼이 오른쪽 버튼(MouseEvent.BUTTOM3_MASK)일 경우에만 팝업메뉴를 이벤트가 발생한 위치에 보이게 합니다.

4.2.7. MenuShortcut

MenuShortcut 클래스는 메뉴에 단축키 기능을 제공합니다. 단축키를 등록하려면 해당키의 문자를 직접 입력할 수도 있고, 'KeyEvent.VK_'와 키 값을 입력하면 되는데 만약, 'A' 키라면 KeyEvent.VK_A라고 기술합니다. 이렇게 하면, CTRL 키와 조합되어 메뉴단축키를 생성합니다. SHIFT 키와 함께 누르는 단축키로 지정하려면 MenuShortcut 생성자의 useShiftModifier 값을 true로 설정합니다.



MenuShortcut 클래스의 객체 생성자와 주요 메서드를 살펴보면 다음과 같습니다.

생성자

◦ MenuShortcut(int key) : 주어진 키에 대한 메뉴단축키 객체를 생성합니다.

◦ MenuShortcut(int key, boolean useShiftModifier) : 주어진 키에 대한 메뉴단축키 객체를 생성합니다. Shift키를 함께 사용할 것인지 여부를 결정할 수 있습니다.

메서드

◦ int getKey() : 키 코드 값을 반환합니다.
◦ boolean usesShiftModifier() : 메뉴단축키 객체가 Shift키를 사용할지 여부를 설정합니다.

다음 프로그램은 MenuShortcut 클래스를 익히기 위한 예제입니다. 여기서는 이벤트 처리를 하지 않았으므로 실제 단축키를 눌러도 아무 반응도 나타나지 않습니다.

exam/java/chapter11/menu/MenuShortcutExample.java

```
1:  package exam.java.chapter11.menu;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class MenuShortcutExample {
7:     private Frame f;
8:     private MenuBar mb;
9:     private Menu fileMenu, editMenu;
10:    private MenuItem newItem, openItem, saveItem, saveAsItem, quitItem;
11:    private MenuShortcut saveAsShortcut = new MenuShortcut('A', true);
12:
13:    public MenuShortcutExample() {
14:       f = new Frame("MenuShortcut Example");
15:       mb = new MenuBar();
16:
17:       fileMenu = new Menu("파일");
18:       editMenu = new Menu("편집");
19:       newItem = new MenuItem("새글", new MenuShortcut('N'));
20:       openItem = new MenuItem("열기");
21:       saveItem = new MenuItem("저장하기", new MenuShortcut(KeyEvent.VK_S));
22:       saveAsItem = new MenuItem("다른 이름으로 저장", saveAsShortcut);
23:       quitItem = new MenuItem("종료", new MenuShortcut('X'));
24:    }
25:
26:    public void launchFrame() {
27:       f.setMenuBar(mb);
28:
29:       mb.add(fileMenu);
30:       mb.add(editMenu);
31:
32:       fileMenu.add(newItem);
33:       openItem.setShortcut(new  MenuShortcut('O'));
34:       fileMenu.add(openItem);
35:       fileMenu.add(saveItem);
36:       fileMenu.add(saveAsItem);
37:
```

```
38:         fileMenu.addSeparator();
39:
40:         fileMenu.add(quitItem);
41:
42:         f.setSize(300, 200);
43:         f.setVisible(true);
44:     }
45:
46:     public static void main (String[] args) {
47:         MenuShortcutExample mt = new MenuShortcutExample();
48:         mt.launchFrame();
49:     }
50: }
```

```
19:         newItem = new MenuItem("새글", new MenuShortcut('N'));
```

생성자를 사용하여 Ctrl+N을 단축키로 갖는 메뉴항목(MenuItem)을 생성합니다.

```
21:         saveITem = new MenuItem("저장하기", new MenuShortcut(KeyEvent.VK_S));
```

java.awt.event.KeyEvent 클래스의 필드(VK_S)를 사용하여 Ctrl+S를 단축키로 지정합니다. KeyEvent 클래스의 필드를 사용하려면 java.awt.event 패키지를 import해야 합니다.

```
33:         openItem.setShortcut(new  MenuShortcut('O'));
```

setShortcut() 메서드를 이용하여 Ctrl+O를 단축키로 설정합니다.

```
11:         MenuShortcut saveAsShortcut = new MenuShortcut('A', true);
```

단축키로 Shift키와 함께 사용하는 MenuShortcut 객체를 생성합니다. 단축키는 Ctrl+Shift+A를 갖게 됩니다.

```
22:         saveAsItem = new MenuItem("다른 이름으로 저장", saveAsShortcut);
```

11라인에서 생성된 MenuShortcut객체를 이용하여 MenuItem의 단축키를 지정합니다.

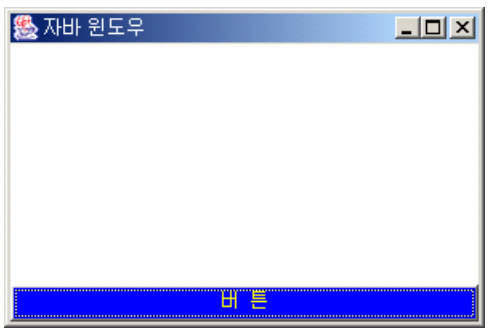
4.3. 색상(Color)과 글꼴(Font)

이 부분에서는 AWT가 제공하는 색상과 글꼴의 사용법 및 기능에 대해 설명하겠습니다.

4.3.1. 색상(Color)

색을 변경하려면 setForeground(Color c)메서드와 setBackground(Color c) 메서드를 이용합니다. 이 메서드는 모두 java.awt.Color 클래스의 인스턴스를 인수로 갖습니다. Color 클래스의 인스턴스는 자주 사용되는 색에 대해서는 상수로 지정된 Color.red(Color.RED), Color.blue(Color.BLUE) 등의 static 멤버변수를 사용할 수도 있고, Color 클래스의 생성자를 통해 R(빨강), G(초록), B(파랑)에 0~255사이의 값을 주어 고유한 색을 만들 수도 있습니다.

```
public Color(int red, int green, int blue)
```



다음 프로그램은 위 그림에서처럼 버튼의 배경색과 전경색(글자 색)을 바꾸는 예입니다.

exam/java/chapter11/color/ColorExample.java

```
1:  package exam.java.chapter11.color;
2:
3:  import java.awt.*;
4:
5:  public class ColorExample {
6:
7:      private Frame f;
8:      private Button btn;
9:
10:     public ColorExample() {
11:         f = new Frame("자바 윈도우");
```

```
12:         btn = new Button("버  튼");
13:     }
14:
15:     public void launchFrame() {
16:         f.add(btn, BorderLayout.SOUTH);
17:         f.setBackground(Color.CYAN);
18:         f.setCursor(Cursor.WAIT_CURSOR);
19:
20:         btn.setBackground(Color.blue);
21:
22:         Color yellow = new Color(255, 255, 0);
23:         btn.setForeground(yellow);
24:
25:         f.setSize(300, 200);
26:         f.setVisible(true);
27:     }
28:
29:     public static void main(String[] args) {
30:         ColorExample win = new ColorExample();
31:         win.launchFrame();
32:     }
33: }
```

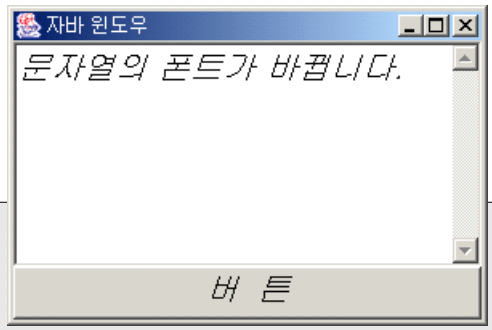
4.3.2. 글꼴(Font)

AWT에서 문자를 표시하는데 사용하는 글꼴은 setFont(Font f) 메서드를 이용하여 지정할 수 있습니다. setFont() 메서드는 java.awt.Font 클래스의 인스턴스를 인자로 갖습니다. Color 클래스와는 달리 글꼴을 직접 지정하는 상수는 없지만, 글꼴이름, 스타일, 포인트 크기 등을 지정할 수 있습니다.

```
public Font(String name, int style, int size)
```

◦ name에는 Dialog, Helvetica, TimesRoman, Curier 등의 글꼴 명을 사용합니다.
◦ 스타일 상수는 int값으로 다음 중 하나를 갖습니다.
  Font.BOLD
  Font.ITALIC
  Font.PLAIN
  Font.BOLD + Font.ITALIC

◦ 폰트 크기의 기본 값은 10 입니다.



다음 프로그램은 앞의 그림과 같은 글꼴을 출력시키는 예제입니다.

exam/java/chapter11/font/FontExample.java

```
1:  package exam.java.chapter11.font;
2:
3:  import java.awt.*;
4:
5:  public class FontExample {
6:
7:     private Frame f;
8:     private Button btn;
9:     private TextArea output;
10:
11:    public FontExample() {
12:       f = new Frame("자바 윈도우");
13:       btn = new Button("버  튼");
14:       output = new TextArea();
15:    }
16:
17:    public void launchFrame() {
18:       f.add(btn, BorderLayout.SOUTH);
19:       f.add(output, BorderLayout.CENTER);
20:
21:       Font myFont = new Font("Curier", Font.ITALIC, 20);
22:
23:       btn.setFont(myFont);
24:       output.setFont(myFont);
25:
26:       f.setSize(300, 200);
27:       f.setVisible(true);
28:    }
29:
30:    public static void main(String[] args) {
31:       FontExample win = new FontExample();
32:       win.launchFrame();
33:       win.output.setText("Curier, ITALIC, 20");
34:    }
35: }
```

4.4. 요점 정리

1. AWT

멤버변수로 선언하고, 생성자 안에서 객체 생성하고, add() 메서드 호출하여 컨테이너에 위젯 추가함
java.awt.*

2. 레이아웃 관리자

컴포넌트의 배치를 담당
    setLayout(레이아웃객체) 메서드를 통해 레이아웃 지정
대표적인 클래스
    FlowLayout
    BorderLayout
    GridLayout
    BoxLayout
여러 개의 패널(Panel)을 이용해 한 화면에 다수의 레이아웃을 사용하여 나타냄

3. Menu

MenuBar
Menu
MenuItem
CheckBoxMenuItem

4. 색상과 글꼴

색상
    상수 값 : Color.red(Color.RED), Color.blue(Color.BLUE) 등의 static 멤버변수
    public Color(int red, int green, int blue) : Color 클래스의 생성자를 통해 R(빨강), G(초록), B(파랑)에 0~255사이의 값
폰트
    setFont(Font f) 메서드를 이용해 지정
    public Font(String name, int style, int size)
        name : 글꼴 이름
        style : Font.BOLD, Font.ITALIC, Font.PLAIN, Font.BOLD + Font.ITALIC 중 하나
        size : 폰트 크기, 디폴트 10

5. 이벤트 프로그래밍

이벤트란 사용자가 버튼을 클릭 하거나 특정키를 누르는 행위 등 사용자 인터페이스를 통한 행위나, 어떤 객체가 활동하여 발생하는 모든 행위를 의미합니다. 이벤트는 AWT 컴포넌트에 대한 동작의 결과를 나타냅니다. 예를 들면, 버튼 위에서 마우스를 누르면 마우스 이벤트의 발생 원인이 되는 것입니다. 이벤트가 발생하면 동작 대상인 컴포넌트(버튼, 텍스트 필드 등)가 그 이벤트를 받게 되는데 이때 받은 이벤트를 처리할 객체가 필요합니다. 지금의 위임형 이벤트 모델에서 이벤트를 받아 처리하는 객체를 이벤트 핸들러(Event Handler)라고 합니다. 이 장에서는 GUI 프로그래밍에서 이벤트 처리 방법에 대해 설명합니다.

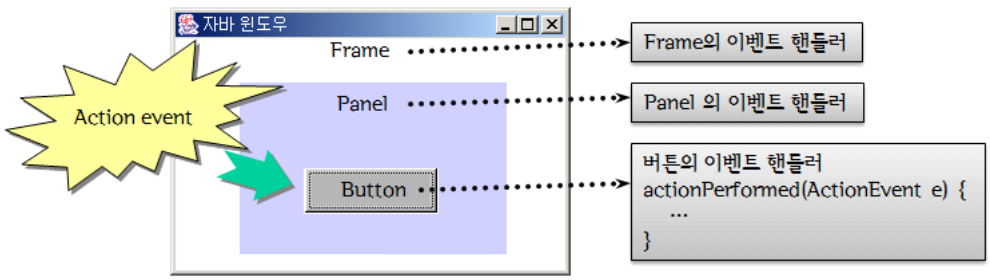
주요 내용입니다.

- 이벤트 모델
- 이벤트와 이벤트 리스너
- ActionEvent
- AdjustmentEvent
- ComponentEvent
- ContainerEvent
- FocusEvent
- KeyEvent
- MouseEvent
- ItemEvent
- TextEvent
- WindowEvent
- MouseWheelEvent

5.1. 이벤트와 이벤트 리스너

5.1.1. 이벤트 모델

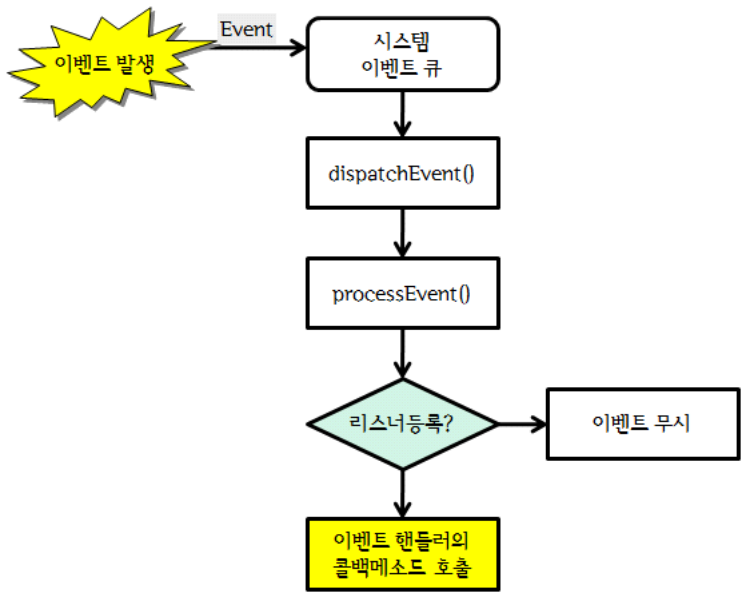
JDK 1.0의 계층형 이벤트 모델[6] 문제점을 해결하기 위해 JDK 1.1에서는 새로운 모델을 채택하였는데, 바로 위임형 이벤트 모델(delegation event model) 방식입니다. 이 모델은 컴포넌트를 기반으로 한 자바 빈즈(Beans)와의 호환을 위해 자바 빈즈에 적용된 방식을 자바에서도 채택한 것입니다.
위임형 이벤트모델 방식은 이벤트 발생 객체가 이벤트를 생성하여 이벤트 처리객체(혹은 이벤트 리스너)에게 위임하는 방식입니다. 이벤트 발생 객체는 각 이벤트 처리 객체를 명시하여 등록하고 이벤트 처리객체는 리스너 인터페이스를 구현하여 이벤트 처리를 책임지게 됩니다.



위 그림에서 보는 바와 같이 하나의 윈도우에 Frame, Panel, Button 객체가 포함되어 있을 때 Frame에서 발생한 이벤트를 처리하기 위한 클래스, Panel에서 발생한 이벤트를 처리하기 위한 클래스, 그리고 Button의 이벤트를 처리하기 위한 클래스를 통해 이벤트를 처리하도록 합니다. 이러한 클래스를 "이벤트 핸들러"라고 부릅니다. 이벤트 핸들러 클래스를 작성하기 위해서는 이벤트 타입에 따라 연관되어 있는 인터페이스를 구현합니다. 이러한 인터페이스를 리스너(Listener) 인터페이스라고 부르며 이들 리스너 인터페이스에 선언된 메서드들은 해당 이벤트가 발생되면 JVM에 의해 자동으로 호출[7]됩니다.

6) 상속 이벤트모델 또는 drill-up/drill-down 방식이라고도 하며 JDK 1.02버전 모델로 지금은 권장하지 않는 방법입니다. 계층형 이벤트 모델은 포함(containment)이라는 개념에 기초한 것입니다. 컴포넌트에 이벤트가 발생하면 시스템 안에서 이벤트 객체를 생성시킨 다음 프로그램의 최하위 컨테이너로부터 최상위 컴포넌트까지 이벤트를 전달합니다. 이렇게 전달된 이벤트는 컴포넌트에 의해 사용 여부가 결정되고 처리가 되면 true를 반환합니다. 만약 false를 반환하면 이벤트가 처리되지 않고 컨테이너에게 전달됩니다.
7) 프로그램 내에서 메서드가 호출되지 않고 JVM에 의해 자동을 호출되는 메서드들을 콜백(Callback)메서드라고 부릅니다. 콜백메서드들의 대표적인 예가 이벤트 리스터 인터페이스의 메서드입니다.

다음 그림은 자바에서 이벤트가 발생했을 경우 내부적으로 어떻게 처리되는지 보여주고 있습니다. 이벤트가 발생되면 발생된 이벤트 객체들은 시스템 이벤트 큐에 저장됩니다. 그리고 dispatchEvent() 메서드에 의해 해당 컴포넌트 또는 하위 컴포넌트에 이벤트를 발송하고, processEvent()를 호출하여 발생한 이벤트를 처리하도록 합니다. processEvent() 메서드는 이벤트를 처리하기 위한 이벤트 핸들러가 등록되어 있는 지 알아본 다음 이벤트 핸들러가 등록되어 있다면 등록된 이벤트 핸들러를 호출해 줘서 이벤트 처리를 하게 됩니다.



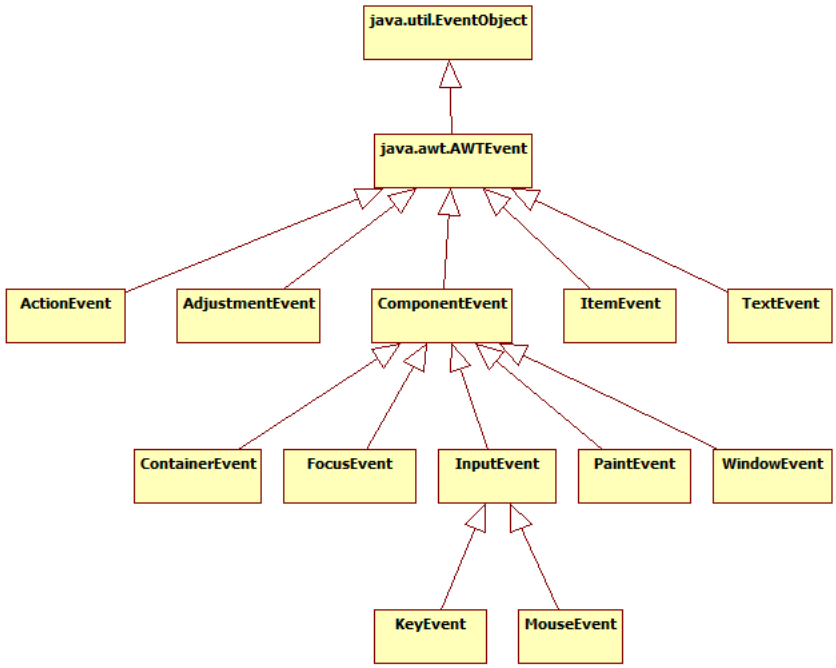
개발자는 이벤트 처리를 하기 위해서 핸들러 클래스를 정의한 다음 컴포넌트에 핸들러를 등록하는 코드를 포함시켜야 합니다. 좀 더 자세한 내용은 이벤트 클래스 계층 구조를 살펴본 다음 설명하겠습니다.

5.1.2. 이벤트 클래스 계층 구조

자바에서는 다양한 이벤트 타입을 지원하며 이벤트 리스너 인터페이스와 연관되어 있습니다. 자바에서는 이벤트 자체도 클래스이며, 모든 이벤트는 계층을 이루고 있습니다. 이벤트 객체는 java.util.EventObject에서 상속되며, java.util.EventObject 가 모든 이벤트 클래스들의 최상위 클래스입니다. 이 클래스는 getSource() 메서드를 갖고 있는데, 개발자는 핸들러 클래스에서 이벤트를 발생시킨 객체(Event Object)를 얻으려면 getSource() 메서드를 사용할 수 있습니다.

다음은 자바에서 제공하는 이벤트 클래스의 계층 구조를 나타낸 것입니다. 그러나 모든 클래스들을 나타낸 것은 아니므로 자세한 계층 구조에 대해서 알고 싶으면 API 문서를 참고

하세요.



EventObject의 하위클래스인 java.awt.AWTEvent 클래스는 위임형 이벤트 모델의 최상위 클래스입니다. 이 클래스는 이벤트의 정확한 특성을 정의한 정수형 이벤트 ID값을 갖습니다. 이 이벤트 ID값은 키를 누르는 이벤트가 발생하였을 때 어떤 키인지 구분할 수 있도록 상수로 표현한 것입니다. 프로그래머는 이벤트 핸들러 클래스에서 java.awt.AWTEvent 클래스 안의 getID() 메서드를 이용하여 이벤트 ID값을 얻을 수 있습니다.

5.1.3. 저수준 이벤트와 고수준 이벤트

AWTEvent 클래스를 상속받은 이벤트 타입은 저 수준(Low Level) 이벤트 클래스와 고수준(시멘틱; Semantic) 이벤트 클래스로 구분됩니다. 저 수준 이벤트란 키 누름, 마우스 누름, 마우스 이동, 포커스 변화 등의 컴포넌트에서 발생하는 사용자 입력이나 시스템 레벨의 이벤트를 말하며, 고수준 이벤트는 더 높은 준의 이벤트로 특정 컴포넌트에 의해서 만들어집니다.

다음은 이벤트 클래스들을 저수준 이벤트와 고주순 이벤트로 분류한 것입니다. 하지만 모두 동일한 처리 기법을 사용하므로 등급을 구별하는 것은 의미가 없을 수 있습니다. 그러나 고주순 이벤트가 발생할 경우에는 저수준 이벤트도 같이 발생하기 때문에 저수준 이벤

트를 처리하기 위한 핸들러 클래스와 고주순 이벤트를 처리하기 위한 이벤트 핸들러를 작성할 경우에는 주의해야 합니다.

	이벤트 이름	이벤트가 발생할 경우
저수준 이벤트	ComponentEvent	컴포넌트 크기변경, 이동 등 컴포넌트에 변화가 있을 때
	ContainerEvent	컴포넌트를 추가하거나 삭제할 때
	FocusEvent	컴포넌트가 포커스를 잃거나 얻었을 때
	KeyEvent	키보드의 키를 누르거나 놓을 때
	MouseEvent	마우스를 누르거나 뗄 때, 누를 때, 마우스 포인터가 컴포넌트 영역으로 들어가거나 밖으로 나왔을 때
		마우스 포인터를 이동하거나 끌(drag) 때
	WindowEvent	윈도우가 열리거나 닫힐 때, 아이콘을 표시하거나 복구할 때

	이벤트 이름	이벤트가 발생할 경우
고주순 이벤트	ActionEvent	컴포넌트 특유의 동작이 일어날 때 - Button : 클릭 - List : 아이템 선택 - MenuItem : 클릭하거나 Enter키를 눌러 선택했을 때 - TextField : Enter키를 누를 때
	AdjustmentEvent	스크롤 바 값이 변경되었을 때
	ItemEvent	아이템을 갖는 컴포넌트의 아이템을 선택하였을 때 - Choice : 아이템이 선택되었을 때 - List : 아이템을 누르거나 방향키로 선택했을 때 - Checkbox : 체크박스가 체크되거나 해제될 때 - CheckboxMenuItem : 체크박스 메뉴 아이템이 체크되거나 해제될 때
	TextEvent	TextField나 TextArea 컴포넌트의 텍스트를 변경할 때

5.1.4. EventListener

이벤트 리스너(Event Listener)는 특정 이벤트를 처리하는 인터페이스로, 프로그래머가 사용자 이벤트에 응답할 프로그램 고유의 코드를 구현하는 곳입니다. 이벤트 전달은 처리할 이벤트를 등록시켜 리스너로 전달할 준비가 끝난 다음 이벤트가 발생하면 객체화하여 상태를 저장하고, 리스너 인터페이스에 정의된 응답 메서드에 상태객체를 넘김으로써 이벤트를 처리하게 됩니다. 따라서 이벤트 리스너는 가장 중요한 역할을 하고 있는 것입니다.

이벤트 리스너는 "*이벤트이름*+Listener"의 형태로 java.awt.event 패키지에 포함되어 있습

니다. 예를 들어 ActionEvent의 리스너 인터페이스 이름은 ActionListener가 되고, WindowEvent의 리스너 인터페이스 이름은 WindowListener가 됩니다.

리스너를 이용하여 컴포넌트로부터 이벤트 클래스를 받기 위해 리스너를 등록하려면 컴포넌트에 "add+*이벤트이름*+Listener()"의 형식으로 리스너를 등록할 수 있는데, 예를 들면 ActionEvent 객체를 등록시키려면 addActionListener() 메서드를 이용합니다.

등록된 리스너를 제거하려면 "remove+*이벤트이름*+Listener" 메서드를 이용합니다. 컴포넌트는 여러 리스너를 가질 수 있지만 등록된 순서에 따라 처리되지는 않습니다. 뒤에서 설명하겠지만 하나의 컴포넌트가 여러 리스너를 가졌을 경우의 실행 순서는 이벤트의 종류에 따라 달라집니다.

다음 코드는 "종료" 버튼을 누르면 윈도우가 종료되는 프로그램입니다. ActionListener 인터페이스를 이용하여 구현하였습니다.

exam/java/chapter12/event/EventExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class EventExample implements ActionListener {
7:     private Frame f;
8:     private Button btn;
9:
10:    public EventExample() {
11:       f = new Frame("리스너 이벤트 처리 예제");
12:       btn = new Button("종료");
13:    }
14:
15:    public void launchFrame() {
16:       btn.addActionListener(this);
17:       f.add(btn, "South");
18:       f.setSize(300, 200);
19:       f.setVisible(true);
20:    }
21:
22:    public static void main(String[] args) {
23:       EventExample lt = new EventExample();
24:       lt.launchFrame();
25:    }
26:
27:    public void actionPerformed(ActionEvent evt) {
28:       String str = evt.getActionCommand();
29:       if(str.equals("종료"))
30:          System.exit(0);
```

```
31:     }
32: }
```

```
3:  public class EventExample implements ActionListener {
```

Frame클래스는 윈도우 창을 만들기 위해 상속받고, 액션 이벤트(ActionEvent)를 사용하기 위해 Listener 인터페이스(ActionListener)를 구현하였습니다.

```
16:         btn.addActionListener(this);
```

버튼을 누를 때 이벤트가 발생하도록 버튼 객체 btn에 액션 리스너를 등록합니다.

```
27:     public void actionPerformed(ActionEvent evt) {
```

실제 이벤트가 수행될 내용을 구현한 코드입니다. actionPerformed() 메서드는 ActionListener 인터페이스 내에 이미 정의가 되어있습니다.

```
28:         String str = evt.getActionCommand();
```

이벤트가 발생하는 객체의 레이블(여기에서는 버튼의 title)을 받아서 스트링형 변수에 저장합니다.

30라인의 System.exit()는 프로그램을 종료시킵니다. 메서드의 인자 값으로 0을 주면 윈도우를 정상적으로 종료시킵니다.

이벤트 핸들러 클래스를 작성하기 위해 이벤트 리스너를 이용할 경우 실제 이벤트 처리를 하지 않는 메서드들도 구현해 주어야 합니다. 이벤트를 처리할 메서드는 모두 public으로 선언해야하며, 리턴 타입이 void형임을 주의하세요. 예를 들면 마우스 이벤트 처리하기 위한 이벤트 핸들러를 MouseListener 인터페이스를 구현하여 작성할 경우 마우스 클릭 이벤트만 감지하여 처리하는 프로그램을 작성하더라도 다른 이벤트와 관련된 콜백메서드를 모두 구현해야 한다는 것입니다.
다음 프로그램은 앞의 윈도우 종료를 MouseListener 인터페이스를 이용하여 구현한 것입니다.

exam/java/chapter12/event/MouseListenerExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class MouseListenerExample implements MouseListener {
7:      private Frame f;
```

```
8:     private Button btn;
9:
10:    public MouseListenerExample() {
11:       f = new Frame("마우스 이벤트 처리 예제")
12:       btn = new Button("종료");
13:    }
14:
15:    public void launchFrame() {
16:       btn.addMouseListener(this);
17:       f.add(btn, "South");
18:       f.setSize(300, 200);
19:       f.setVisible(true);
20:    }
21:
22:    public static void main(String[] args) {
23:       MouseListenerExample mt = new MouseListenerExample();
24:       mt.launchFrame();
25:    }
26:
27:    public void mouseClicked(MouseEvent ae) {
28:       System.exit(0);
29:    }
30:    public void mouseEntered (MouseEvent evt) { }
31:    public void mouseExited (MouseEvent evt) { }
32:    public void mousePressed (MouseEvent evt) { }
33:    public void mouseReleased (MouseEvent evt) { }
34: }
```

6라인 클래스 선언부에서 마우스 이벤트를 사용하기 위해서 MouseListener 인터페이스를 implements 하였습니다. 16라인은 버튼 객체에 마우스 이벤트 리스너를 등록시킵니다.

```
27:    public void mouseClicked(MouseEvent ae) {
28:       System.exit(0);
29:    }
```

마우스를 누를 때 수행될 내용입니다. System.exit(0)은 프로그램을 강제 종료시키는 명령입니다.

```
30:    public void mouseEntered (MouseEvent evt) { }
31:    public void mouseExited (MouseEvent evt) { }
32:    public void mousePressed (MouseEvent evt) { }
33:    public void mouseReleased (MouseEvent evt) { }
```

30라인부터 33라인 까지는 이벤트에 사용하지 않는 메서드들이지만 구현부를 정의해 놓았습니다. 이것이 리스너 인터페이스를 구현했을 때의 단점이라면 단점입니다. 실제 사용되지 않는 메서드도 구현해야 합니다.

핸들러 클래스는 리스너 인터페이스를 구현해서만 만들 수 있는 것은 아닙니다. 뒤에서 설명되는 어댑터(Adapter)클래스를 상속받아 핸들러 클래스를 만들 수 있습니다. 어댑터 클래스를 이용하면 이벤트에 필요한 콜백메서드만 재정의 해서 사용하면 됩니다. 그러나 어댑터 클래스를 상속받아 핸들러를 작성하면 다른 클래스를 상속받을 수 없는 단점이 있습니다. 인터페이스를 구현하는 방법으로 핸들러를 작성하면 사용하지 않는 메서드를 재정의 해 줘야 하는 불편함이 있지만 요즘의 IDE 툴에서는 자동으로 코드를 만들어 주기 때문에 그러한 불편함을 해결할 수 있을 것입니다. 그리고 리스너 인터페이스를 구현해서 핸들러를 만든다면 mouseClicked() 메서드를 mouseClick() 또는 mouseclicked() 등으로 메서드 이름을 잘못 정의할 염려도 없습니다. 그리고 자바의 단일 상속의 문제점도 해결할 수 있을 것입니다.

5.1.5. 리스너 인터페이스와 메서드

다음은 자바에서 제공하는 이벤트와 이벤트 리스너, 그리고 리스너 인터페이스에 정의되어 있는 메서드들을 표로 정리해서 보여주고 있습니다. 대부분의 이벤트 클래스에 리스너 인터페이스를 하나씩 가지고 있지만 몇몇 이벤트 클래스는 리스너 인터페이스가 2개 이상인 경우도 있으므로 주의 깊게 보시기 바랍니다. 그리고 Add 메서드는 핸들러를 이벤트 소스에 연결시키기 위한 메서드입니다. 앞에서도 언급했지만 메서드 이름이 규칙성을 가지고 있기 때문에 외우는 것은 어렵지 않을 것입니다.

Event	Listener	Interface Method	Add Method
ActionEvent	ActionListener	actionPerformed(ActionEvent e)	addActionListener()
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)	addAdjustmentListener()
ComponentEvent	ComponentListener	componentHidden(ComponentEvent e) componentMoved(ComponentEvent e) componentResized(ComponentEvent e) componentShown(ComponentEvent e)	addComponentListener()
ContainerEvent	ContainerListener	componentAdded(ComponentEvent e) componentRemoved(ComponentEvent e)	addListener()
FocusEvent	FocusListener	focusGained(FocusEvent e) focusLost(FocusEvent e)	addFocusListener()
ItemEvent	ItemListener	itemStateChanged(ItemEvent e)	addItemListener()
KeyEvent	KeyListener	keyPressed(KeyEvent e) keyReleased(KeyEvent e) keyTyped(KeyEvent e)	addKeyListener()
MouseEvent	MouseListener	mouseClicked(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mousePressed(MouseEvent e) mouseReleased(MouseEvent e)	addMouseListener()
	MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)	addMouseMotionListener()
MouseWheelEvent	MouseWheelListener	mouseWheelMoved(MouseWheelEvent e)	addMouseWheelListener()
TextEvent	TextListener	textValueChanged(TextEvent e)	addTextListener()
WindowEvent	WindowListener	windowActivated(WindowEvent e) windowClosed(WindowEvent e) windowClosing(WindowEvent e) windowDeactivated(WindowEvent e) windowDeiconified(WindowEvent e) windowIconified(WindowEvent e) windowOpened(WindowEvent e)	addWindowListener()
	WindowFocusListener	windowGainedFocus(WindowEvent e) windowLostFocus(WindowEvent e)	addWindowFocusListener()
	WindowStateListener	windowStateChanged(WindowEvent e)	addWindowStateListener()

위의 표에 나타난 이벤트와 이벤트 리스너가 자바 이벤트의 모든 것은 아니므로 더 자세한 것은 API 문서를 참고하세요.

다음 표는 컴포넌트 타입에 따라 발생할 수 있는 이벤트를 나타낸 것입니다. 컴포넌트에 따라 사용할 수 있는 이벤트가 정해져 있음을 보여주고 있습니다. 여러분은 이벤트가 발생할 컴포넌트를 결정한 다음 어떤 이벤트를 적용시켜야 할지 결정해야 합니다.

Component Type	Act	Adj	Cmp	Cnt	Foc	Itm	Key	Mou	MM	MW	Txt	Win
Button	✔		✔		✔		✔	✔	✔	✔		
Canvas			✔		✔		✔	✔	✔	✔		
Checkbox			✔		✔	✔	✔	✔	✔	✔		
CheckboxMenuItem						✔						
Choice			✔		✔	✔	✔	✔	✔	✔		
Component			✔		✔		✔	✔	✔	✔		
Container			✔	✔	✔		✔	✔	✔	✔		
Dialog			✔	✔	✔		✔	✔	✔	✔		✔
Frame			✔	✔	✔		✔	✔	✔	✔		✔
Label			✔		✔		✔	✔	✔	✔		
List	✔		✔		✔	✔	✔	✔	✔	✔		
MenuItem	✔											
Panel			✔	✔	✔		✔	✔	✔	✔		
Scrollbar		✔	✔		✔		✔	✔	✔	✔		
ScrollPane			✔	✔	✔		✔	✔	✔	✔		
TextArea			✔		✔		✔	✔	✔	✔	✔	
TextField	✔		✔		✔		✔	✔	✔	✔	✔	
Window			✔	✔	✔		✔	✔	✔	✔		✔

Act - ActionListener
Adj - AdjustmentListener
Cmp - ComponentListener
Cnt - ContainerListener
Foc - FocusListener
Itm - ItemListener
Key - KeyListener
Mou - MouseListener
MM - MouseMotionListener
MW - MouseWheelListener
Txt - TextListener
Win - WindowListener

5.2. 이벤트 프로그래밍

지금까지 이벤트 클래스들 그리고 이벤트 핸들러와 리스너에 대해 알아보았습니다. 이제 본격적으로 이벤트 프로그래밍은 어떻게 하는지 알아보겠습니다.

이벤트 프로그래밍 순서는 다음과 같습니다.

1) 이벤트가 발생될 컴포넌트에 대해 이벤트 소스와 이벤트를 선택합니다.
2) 선택한 이벤트의 처리를 위해 이벤트 핸들러를 작성합니다.
3) addXxxListener() 메서드를 이용하여 이벤트 핸들러를 추가합니다.

이벤트를 처리하기 위해 만드는 리스너 인터페이스를 구현하는 핸들러 클래스는 콜백메서드를 어떤 클래스에 구현하느냐에 따라 5가지 방법이 있습니다.

- 이벤트 발생 클래스와 동일 클래스 : 이벤트가 발생하는 클래스에서 Listener 인터페이스를 구현합니다.
- 이벤트가 발생하는 클래스와 별개의 클래스 : 별도로 클래스를 만들어 Listener 인터페이스를 구현합니다.
- Inner 클래스 : 클래스 안에 내부 클래스를 선언하여 핸들러를 정의합니다.
- Local 클래스 : 메서드 안에 Local 클래스를 선언하여 핸들러로 정의합니다.
- Anonymous 클래스 : addXxxListener() 메서드 인자로 직접 핸들러를 구현합니다.

5.2.1. 이벤트 발생 클래스와 동일 클래스에 핸들러 구현

다음 프로그램은 이벤트가 발생하는 클래스와 동일 클래스에서 핸들러를 만든 예를 보인 것입니다.

exam/java/chapter12/event/ListenerExample1.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class ListenerExample1 implements WindowListener {
7:     private Frame f;
8:
9:     public ListenerExample1() {
10:       f = new Frame("Listener인터페이스를 이용한 윈도우 종료");
11:    }
12:
13:    public void launchFrame() {
14:       f.addWindowListener( this );
```

```
15:        f.setSize(300, 200);
16:        f.setVisible(true);
17:     }
18:
19:     public static void main(String[] args) {
20:        ListenerExample1 lt1 = new ListenerExample1();
21:        lt1.launchFrame();
22:     }
23:
24:     public void windowClosing(WindowEvent e) {
25:        System.exit(0);
26:     }
27:     public void windowOpened(WindowEvent e) {}
28:     public void windowClosed(WindowEvent e) {}
29:     public void windowIconified(WindowEvent e) {}
30:     public void windowDeiconified(WindowEvent e) {}
31:     public void windowActivated(WindowEvent e) {}
32:     public void windowDeactivated(WindowEvent e) {}
33: }
```

5.2.2. 별도의 클래스로 핸들러 구현

다음 프로그램은 이벤트가 발생하는 클래스와 별도의 클래스로 핸들러를 만든 예를 보인 것입니다.
이벤트 처리 핸들러를 별도의 클래스로 만들었습니다. 그리고 핸들러 클래스를 ListenerExample2 클래스에서 리스너에 등록합니다.

exam/java/chapter12/event/MyWindowHandler.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.event.*;
4:
5:  public class MyWindowHandler implements WindowListener {
6:
7:     public void windowClosing(WindowEvent e) {
8:        System.exit(0);
9:     }
10:
11:    public void windowOpened(WindowEvent e) {}
12:    public void windowClosed(WindowEvent e) {}
13:    public void windowIconified(WindowEvent e) {}
14:    public void windowDeiconified(WindowEvent e) {}
15:    public void windowActivated(WindowEvent e) {}
16:    public void windowDeactivated(WindowEvent e) {}
17: }
```

exam/java/chapter12/event/ListenerExample2.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:
5:  public class ListenerExample2 {
6:     private Frame f;
7:
8:     public ListenerExample2() {
9:        f = new Frame("Listener인터페이스를 이용한 윈도우 종료");
10:    }
11:
12:    public void launchFrame() {
13:       f.addWindowListener( new MyWindowHandler() );
14:       f.setSize(300, 200);
15:       f.setVisible(true);
16:    }
17:
18:    public static void main(String[] args) {
19:       ListenerExample2 lt2 = new ListenerExample2();
20:       lt2.launchFrame();
21:    }
22: }
```

5.2.3. Inner 클래스로 핸들러 구현

다음 프로그램은 이벤트가 발생하는 클래스 안의 클래스인 Inner 클래스로 핸들러를 만든 예를 보인 것입니다.

exam/java/chapter12/event/ListenerExample3.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class ListenerExample3 {
7:     private Frame f;
8:
9:     public ListenerExample3() {
10:       f = new Frame("Listener인터페이스를 이용한 윈도우 종료");
11:    }
12:
13:    public void launchFrame() {
14:       f.addWindowListener( new MyInnerHandler() );
15:       f.setSize(300, 200);
16:       f.setVisible(true);
```

```
17:     }
18:
19:     public static void main(String[] args) {
20:        ListenerExample3 lt3 = new ListenerExample3();
21:        lt3.launchFrame();
22:     }
23:
24:     private class MyInnerHandler implements WindowListener {
25:
26:        public void windowClosing(WindowEvent e) {
27:           System.exit(0);
28:        }
29:
30:        public void windowOpened(WindowEvent e) {}
31:        public void windowClosed(WindowEvent e) {}
32:        public void windowIconified(WindowEvent e) {}
33:        public void windowDeiconified(WindowEvent e) {}
34:        public void windowActivated(WindowEvent e) {}
35:        public void windowDeactivated(WindowEvent e) {}
36:     }
37: }
```

클래스 내부에 핸들러 클래스를 선언하였습니다. 선언된 핸들러는 다른 클래스에서 사용하지 못하도록 private로 선언하였습니다.

5.2.4. Local 클래스로 핸들러 구현

다음 프로그램은 핸들러 객체를 사용하는 메서드 안에 Local 클래스로 선언한 예를 보인 것입니다.

exam/java/chapter12/event/ListenerExample4.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class ListenerExample4 {
7:     private Frame f;
8:
9:     public ListenerExample4() {
10:       f = new Frame("Listener인터페이스를 이용한 윈도우 종료");
11:    }
12:
13:    public void launchFrame() {
14:       class MyInnerHandler implements WindowListener {
15:          public void windowClosing(WindowEvent e) {
```

```
16:              System.exit(0);
17:           }
18:           public void windowOpened(WindowEvent e) {}
19:           public void windowClosed(WindowEvent e) {}
20:           public void windowIconified(WindowEvent e) {}
21:           public void windowDeiconified(WindowEvent e) {}
22:           public void windowActivated(WindowEvent e) {}
23:           public void windowDeactivated(WindowEvent e) {}
24:        }
25:
26:        f.addWindowListener( new MyInnerHandler() );
27:        f.setSize(300, 200);
28:        f.setVisible(true);
29:     }
30:
31:     public static void main(String[] args) {
32:        ListenerExample4 lt4 = new ListenerExample4();
33:        lt4.launchFrame();
34:     }
35: }
```

핸들러클래스를 실제 사용하는 메서드 안에 선언할 수 있습니다. 이 경우에는 이벤트 핸들러를 사용하기 전에 클래스가 구현되어 있어야 합니다. 즉, 소스코드에서 객체를 만드는 코드보다 클래스를 선언하는 코드가 앞에 나와야 한다는 것입니다.

5.2.5. 익명 클래스로 핸들러 구현

다음 프로그램은 Anonymous 클래스로 핸들러를 만든 예를 보인 것입니다.

exam/java/chapter12/event/ListenerExample5.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class ListenerExample5 {
7:     private Frame f;
8:
9:     public ListenerExample5() {
10:       f = new Frame("Listener인터페이스를 이용한 윈도우 종료");
11:    }
12:
13:    public void launchFrame() {
14:       f.addWindowListener( new WindowListener() {
15:             public void windowClosing(WindowEvent e) {
16:                System.exit(0);
```

```
17:             }
18:
19:             public void windowOpened(WindowEvent e) {}
20:             public void windowClosed(WindowEvent e) {}
21:             public void windowIconified(WindowEvent e) {}
22:             public void windowDeiconified(WindowEvent e) {}
23:             public void windowActivated(WindowEvent e) {}
24:             public void windowDeactivated(WindowEvent e) {}
25:          }
26:       );
27:
28:       f.setSize(300, 200);
29:       f.setVisible(true);
30:    }
31:
32:    public static void main(String[] args) {
33:       ListenerExample5 lt5 = new ListenerExample5();
34:       lt5.launchFrame();
35:    }
36: }
```

add메서드 인자로 핸들러클래스를 선언하였습니다. 이러한 클래스는 클래스 이름이 없이 선언되므로 amonymous클래스라고 합니다.

5.3. Adapter 클래스

고수준 이벤트 리스너는 하나의 메서드만 갖는 반면 저수준(Low-Level)리스너는 2개 이상의 메서드를 갖습니다. 특히 앞의 WindowListener 경우 최대 7개의 메서드를 가지고 있는데, 이 리스너가 인터페이스로 선언되면 윈도우가 닫히는 단 하나의 처리를 위해 6개의 불필요한 메서드까지 구현해 주어야 합니다.
이와 같은 문제점을 해소하기 위해 어댑터 클래스를 이용할 수 있습니다. 어댑터 클래스는 리스너 인터페이스의 모든 메서드를 사용하지 않는 디폴트 메서드로 정의해 놓은 클래스입니다. 이 어댑터 클래스를 상속받아 이벤트처리 객체로 등록하고, 원하는 메서드를 재정의하여 사용하면 불필요한 메서드까지 선언하는 수고를 덜 수 있습니다.
어댑터 클래스의 이름은 "*이벤트이름*+Adapter"형태로 되어 있습니다. 예를 들어 WindowListener의 경우에는 WindowAdapter가 있습니다. 모든 이벤트에 어댑터 클래스가 존재하는 것은 아닙니다. 메서드가 하나만 존재하는 인터페이스는 어댑터 클래스가 존재하지 않습니다. 대부분의 어댑터 클래스는 이벤트이름에 Adapter가 붙어있지만, 특이하게도 MouseMotionListener의 경우에는 이벤트 이름이 MouseEvent이지만 어댑터 이름은 MouseMotionAdapter입니다.

어댑터 클래스를 상속받아 핸들러를 만들 경우에도 여러 가지 종류의 클래스로 만들 수가 있습니다. 다음의 예들을 살펴보면 12.2절의 코드들에 비해 훨씬 간결해 진 것을 볼 수 있습니다.

5.3.1. 이벤트 발생 클래스와 동일 클래스에 핸들러 구현

다음 프로그램은 이벤트가 발생하는 클래스와 동일 클래스에서 어댑터클래스를 상속받아 핸들러를 구현하는 예입니다.

exam/java/chapter12/event/AdapterExample1.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class AdapterExample1 extends WindowAdapter {
7:     private Frame f;
8:
9:     public AdapterExample1() {
10:       f = new Frame("Adapter클래스를 이용한 윈도우 종료");
11:    }
12:
13:    public void launchFrame() {
```

```
14:         f.addWindowListener( this );
15:         f.setSize(300, 200);
16:         f.setVisible(true);
17:     }
18:
19:     public static void main (String[] args) {
20:         AdapterExample1 at1 = new AdapterExample1();
21:         at1.launchFrame();
22:     }
23:
24:     public void windowClosing(WindowEvent e) {
25:         System.exit(0);
26:     }
27: }
```

5.3.2. 별도의 클래스로 핸들러 구현

다음 프로그램은 이벤트가 발생하는 클래스와 별도의 클래스로 핸들러를 만든 예입니다.

exam/java/chapter12/event/MyWindowAdapterHandler.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.event.*;
4:
5:  public class MyWindowAdapterHandler extends WindowAdapter {
6:
7:      public void windowClosing(WindowEvent e) {
8:          System.exit(0);
9:      }
10:
11: }
```

exam/java/chapter12/event/AdapterExample2.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:
5:  public class AdapterExample2 {
6:      private Frame f;
7:
8:      public AdapterExample2() {
9:          f = new Frame("Adapter클래스를 이용한 윈도우 종료");
10:     }
11:
12:     public void launchFrame() {
13:         f.addWindowListener( new MyWindowAdapterHandler() );
14:         f.setSize(300, 200);
```

```
15:        f.setVisible(true);
16:     }
17:
18:     public static void main (String[] args) {
19:        AdapterExample2 at2 = new AdapterExample2();
20:        at2.launchFrame();
21:     }
22: }
```

5.3.3. Inner 클래스로 핸들러 구현

다음 프로그램은 이벤트가 발생하는 클래스 안에 Inner 클래스로 핸들러를 만든 예를 보인 것입니다.

exam/java/chapter12/event/AdapterExample3.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class AdapterExample3 {
7:     private Frame f;
8:
9:     public AdapterExample3() {
10:       f = new Frame("Adapter클래스를 이용한 윈도우 종료");
11:    }
12:
13:    public void launchFrame() {
14:       f.addWindowListener( new MyInnerHandler() );
15:       f.setSize(300, 200);
16:       f.setVisible(true);
17:    }
18:
19:    public static void main (String[] args) {
20:       AdapterExample3 at3 = new AdapterExample3();
21:       at3.launchFrame();
22:    }
23:
24:    private class MyInnerHandler extends WindowAdapter {
25:       public void windowClosing(WindowEvent e) {
26:          System.exit(0);
27:       }
28:    }
29:
30: }
```

5.3.4. Local 클래스로 핸들러 구현

다음 프로그램은 메서드 안에 Local클래스로 핸들러를 구현한 예를 보인 것입니다.

exam/java/chapter12/event/AdapterExample4.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class AdapterExample4 {
7:     private Frame f;
8:
9:     public AdapterExample4() {
10:       f = new Frame("Adapter클래스를 이용한 윈도우 종료");
11:    }
12:
13:    public void launchFrame() {
14:       class MyInnerHandler extends WindowAdapter {
15:          public void windowClosing(WindowEvent e) {
16:             System.exit(0);
17:          }
18:       }
19:
20:       f.addWindowListener( new MyInnerHandler() );
21:       f.setSize(300, 200);
22:       f.setVisible(true);
23:    }
24:
25:    public static void main (String[] args) {
26:       AdapterExample4 at4 = new AdapterExample4();
27:       at4.launchFrame();
28:    }
29:
30: }
```

5.3.5. 익명 클래스로 핸들러 구현

다음 프로그램은 Anonymous 클래스로 핸들러를 만든 예를 보인 것입니다.

exam/java/chapter12/event/AdapterExample5.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class AdapterExample5 {
```

```
7:      private Frame f;
8:
9:      public AdapterExample5() {
10:        f = new Frame("Adapter클래스를 이용한 윈도우 종료");
11:     }
12:
13:     public void launchFrame() {
14:        f.addWindowListener( new WindowAdapter() {
15:              public void windowClosing(WindowEvent e) {
16:                 System.exit(0);
17:              }
18:           }
19:        );
20:
21:        f.setSize(300, 200);
22:        f.setVisible(true);
23:     }
24:
25:     public static void main (String[] args) {
26:        AdapterExample5 at5 = new AdapterExample5();
27:        at5.launchFrame();
28:     }
29: }
```

5.4. 주요 이벤트 클래스

이제 각 이벤트별 리스너 인터페이스와 어댑터 클래스에 대해 설명하고 발생할 수 있는 메서드에 대해 구체적으로 언급하겠습니다.

5.4.1. ActionEvent

Action 이벤트는 Button, List, MenuItem, TextField가 다음과 같은 상황일 때 발생됩니다.

- Button : 마우스를 누를 때
- List : 아이템을 선택하거나 선택을 해제했을 때
- MenuItem : 메뉴아이템을 선택했을 때
- TextField : 텍스트필드에서 엔터키를 눌렀을 때

12.4.1.1 ActionListener

동작(Action) 이벤트를 처리할 리스너의 기능을 정의하는 인터페이스입니다. 따라서 동작 이벤트를 처리하려면 이 인터페이스를 구현해야 하고, 객체를 생성한 후 컴포넌트의 addActionListener() 메서드를 이용하여 컴포넌트에 등록합니다. 동작이 일어나면 해당 ActionListener의 actionPerformed() 메서드가 호출됩니다.

ActionListener 메서드

◦ void actionPerformed(ActionEvent e) : 동작이 발생할 때 호출됩니다.

다음 프로그램은 Action 이벤트의 사용 예를 보인 것입니다.

exam/java/chapter12/event/ActionExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class ActionExample implements ActionListener {
7:     private Frame f;
8:     private MenuBar mb;
9:     private Menu fileMenu;
10:    private MenuItem exitItem;
```

```
11:    private TextField inputField;
12:    private Button sendButton;
13:    private Panel p = new Panel();
14:
15:    public ActionExample() {
16:       f = new Frame("Action Event");
17:       mb = new MenuBar();
18:       fileMenu = new Menu("파일");
19:       exitItem = new MenuItem("종료");
20:       p = new Panel();
21:       inputField = new TextField();
22:       sendButton = new Button("Send");
23:    }
24:
25:    public void launchFrame() {
26:       f.setMenuBar(mb);
27:       mb.add(fileMenu);
28:       fileMenu.add(exitItem);
29:       p.setLayout(new BorderLayout());
30:       p.add(inputField, BorderLayout.CENTER);
31:       p.add(sendButton, BorderLayout.EAST);
32:       f.add(p, BorderLayout.SOUTH);
33:       exitItem.addActionListener(this);
34:       inputField.addActionListener(this);
35:       sendButton.addActionListener(this);
36:       f.setSize(200, 200);
37:       f.setVisible(true);
38:    }
39:    public static void main(String[] args) {
40:       ActionExample win = new ActionExample();
41:       win.launchFrame();
42:    }
43:
44:    public void actionPerformed(ActionEvent e) {
45:       if(e.getSource() == exitItem) {
46:          System.out.println("종료메뉴가 선택되었습니다");
47:          System.exit(0);
48:       }else if(e.getSource() == inputField) {
49:          System.out.println("텍스트 필드에서 엔터 입력");
50:          System.out.println(inputField.getText() + "입력");
51:          inputField.setText("");
52:       }else if(e.getSource() == sendButton) {
53:          System.out.println("버튼이 클릭되었습니다");
54:          System.out.println(inputField.getText() + "입력");
55:          inputField.setText("");
56:          inputField.requestFocus();
57:       }
58:    }
59: }
```

5.4.2. AdjustmentEvent

스크롤바의 값이 변경되었을 때 발생합니다.

12.4.2.1 AdjustmentListener

리스너가 구현해야할 기능을 정의하고 있는 인터페이스로 Adjustment 이벤트를 처리하려면 이 인터페이스를 구현해야 합니다.
AdjustmentListener를 사용하면 스크롤바 컴포넌트에 등록하기 위해 다음과 같은 메서드를 사용합니다.

- void addAdjustmentListener(AdjustmentListener l) : Adjustment Listener를 추가합니다.
- void removeAdjustmentListener(AdjustmentListener l) : 주어진 Listener를 제거합니다.

AdjustmentListener 메서드

- void adjustmentValueChanged(AdjustmentEvent e) : 스크롤바 컴포넌트의 값이 변할 때 호출됩니다.

다음 프로그램은 Adjustment 이벤트를 사용하는 예를 보인 것입니다.

exam/java/chapter12/event/AdjustmentExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class AdjustmentExample implements AdjustmentListener {
7:
8:     private Frame f;
9:     private Scrollbar mySlider;
10:
11:    public AdjustmentExample() {
12:       f = new Frame("Adjustment Event");
13:       mySlider = new Scrollbar(Scrollbar.HORIZONTAL, 100, 20, 0, 255);
14:    }
15:
16:    public void launchFrame() {
17:       f.addWindowListener( new WindowAdapter() {
18:          public void windowClosing(WindowEvent e) {
19:             System.exit(0);
20:          }
21:       });
22:       f.add(mySlider, BorderLayout.SOUTH);
```

```
23:         mySlider.addAdjustmentListener(this);
24:         f.setSize(300, 100);
25:         f.setVisible(true);
26:     }
27:
28:     public static void main(String[] args) {
29:         AdjustmentExample win = new AdjustmentExample();
30:         win.launchFrame();
31:     }
32:
33:     public void adjustmentValueChanged(AdjustmentEvent e) {
34:         System.out.println(e.getValue() + "으로 값이 변경됨");
35:     }
36: }
```

5.4.3. ComponentEvent

MenuItem과 CheckboxMenuItem을 제외한 모든 컴포넌트에서 발생하는 이벤트입니다.

12.4.3.1 ComponentListener

Component 이벤트의 리스너를 정의한 인터페이스이며, 이벤트를 처리하려면 이 ComponentListener 인터페이스를 implements하여 메서드를 구현하거나 ComponentAdapter 클래스를 상속받아야 합니다.
컴포넌트의 addComponentListener() 메서드에 의해 등록되며, 컴포넌트 크기나 위치 또는 가시화(Visibility) 등의 변화가 일어날 때 Component 이벤트가 발생합니다.

12.4.3.2 ComponentAdapter

Component 이벤트 처리를 위한 abstract adapter 클래스를 제공합니다. 이 클래스는 ComponentListener 인터페이스가 정의하고 있는 메서드를 구현하고 있지만, 메서드의 몸체는 비어 있기 때문에 처리하고자 하는 이벤트의 메서드만 재정의 해주면 됩니다.

ComponentListener 메서드

- void componentMoved(ComponentEvent e) : 컴포넌트 위치 변경시 호출됩니다.
- void componentResized(ComponentEvent e) : 컴포넌트 크기 변경시 호출됩니다.
- void componentShown(ComponentEvent e) : 컴포넌트를 보이게 할 때 호출됩니다.
- void componentHidden(ComponentEvent e) : 컴포넌트를 보이지 않게 할 때 호출됩니다.

다음 프로그램은 Component 이벤트의 사용 예를 보인 것입니다.

exam/java/chapter12/event/ComponentExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class ComponentExample implements ComponentListener {
7:
8:     private Frame f;
9:
10:    public ComponentExample() {
11:       f = new Frame("Component Event");
12:    }
13:
14:    public void launchFrame() {
15:       f.addWindowListener( new WindowAdapter() {
16:          public void windowClosing(WindowEvent e) {
17:             System.exit(0);
18:          }
19:       });
20:       f.addComponentListener(this);
21:       f.setSize(300, 100);
22:       f.setVisible(true);
23:    }
24:
25:    public static void main(String[] args) {
26:       ComponentExample win = new ComponentExample();
27:       win.launchFrame();
28:    }
29:
30:    public void componentMoved(ComponentEvent e) {
31:       System.out.println(e.getSource() + " 위치 변경");
32:    }
33:    public void componentResized(ComponentEvent e) {
34:       System.out.println(e.getSource() + " 크기 변경");
35:    }
36:    public void componentShown(ComponentEvent e) {
37:       System.out.println(e.getSource() + "가 보여짐");
38:    }
39:    public void componentHidden(ComponentEvent e) {
40:       System.out.println(e.getSource() + "가 없어짐");
41:    }
42: }
```

5.4.4. ContainerEvent

컨테이너에서 컴포넌트의 이동이나 크기 변경 또는 보이거나 보이지 않게 할 때 발생하는 저수준 이벤트이며, 단순하게 알려주는 목적으로 사용된다. AWT에서는 컴포넌트의 크기 변경 및 이동이 자동으로 처리되며, 이 이벤트 처리 여부에 상관없이 레이아웃을 수행하게 하는데 사용됩니다.
컴포넌트를 컨테이너에 추가하거나 삭제하면 컨테이너 이벤트가 발생되고 ContainerListener 또는 ContainerAdatper에 전달됩니다.

- static int COMPONENT_ADDED : 컴포넌트 추가.
- static int COMPONENT_REMOVED : 컴포넌트 제거.
- static int CONTAINER_FIRST : 컨테이너 이벤트 id의 시작번호.
- static int CONTAINER_LAST : 컨테이너 이벤트 id의 마지막번호.
- Component getChild() : 이벤트에 영향을 받는 컴포넌트를 반환합니다.
- Container getContainer() : 이벤트가 발생한 컴포넌트를 반환합니다.

12.4.4.1 ContainerListener

Container 이벤트를 처리할 수 있는 기능을 정의하는 인터페이스로 이벤트를 처리할 클래스는 ContainerListener 인터페이스가 정의한 모든 메서드를 구현하거나, ContainerAdapter 클래스의 관련 메서드를 재정의 하여 상속받아야 합니다.

12.4.4.2 ContainerAdapter

컨테이너 이벤트를 처리할 추상 클래스를 제공합니다. ContainerListener 인터페이스가 정의한 메서드를 구현하고 있지만 메서드의 몸체는 비어 있습니다. 따라서 ContainerAdapter의 메서드 중 처리하고자 하는 메서드만 재 정의해 줍니다.

ContainerListener 메서드

- void componentAdded(ContainerEvent e) : 컴포넌트가 컨테이너에 추가될 때 호출됩니다.
- void componentRemoved(ContainerEvent e) : 컴포넌트가 컨테이너에서 제거될 때 호출됩니다.

다음 프로그램은 Container 이벤트의 사용 예를 보인 것입니다.

exam/java/chapter12/event/ContainerExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
```

```
6:  public class ContainerExample implements ContainerListener {
7:
8:      private Frame f;
9:      private Button btn1;
10:
11:     public ContainerExample() {
12:         f = new Frame("Container Event");
13:         btn1 = new Button("Button");
14:     }
15:
16:     public void launchFrame() {
17:         f.addWindowListener( new WindowAdapter() {
18:             public void windowClosing(WindowEvent e) {
19:                 System.exit(0);
20:             }
21:         });
22:         f.addContainerListener(this);
23:         f.add(btn1, BorderLayout.SOUTH);
24:         f.setSize(300, 100);
25:         f.setVisible(true);
26:     }
27:
28:     public static void main(String[] args) {
29:         ContainerExample win = new ContainerExample();
30:         win.launchFrame();
31:     }
32:
33:     public void componentAdded(ContainerEvent e) {
34:         System.out.println("컴포넌트 추가됨");
35:     }
36:     public void componentRemoved(ContainerEvent e) {
37:         System.out.println("컴포넌트 삭제됨");
38:     }
39: }
```

5.4.5. FocusEvent

컴포넌트가 키보드 포커스를 얻거나 잃을 때 발생하는 저수준 이벤트입니다. 이러한 이벤트는 MenuItem컴포넌트를 제외한 거의 모든 컴포넌트에서 발생하며, FocusListener 또는 FocusAdapter에 전달되고, 컴포넌트 클래스의 addFocusListener() 메서드를 이용하여 등록할 수 있습니다.

이 때, 포커스의 변화는 다음과 같은 두 가지가 있습니다.
일시적인 포커스 변화 : requestFocus() 메서드를 호출하거나 탭 키로 컴포넌트들 사이에

포커스를 이동시킬 때 발생하는 이벤트입니다.

영구적인 포커스 변화 : 윈도우의 비활성화 또는 스크롤바의 끌기와 같이 다른 동작의 간접적인 결과로 발생하여, 일시적으로 포커스를 얻거나 잃을 때 발생하는 이벤트입니다. 이런 경우 윈도우가 다시 활성화되거나 해당 동작이 끝나면 포커스는 자동으로 전 상태로 복구됩니다.

위의 두 가지 포커스 이벤트 모두 FOCUS_GAINED와 FOCUS_LOST 이벤트 id를 발생시키며, 이 때 isTemporary() 메서드를 이용하여 이벤트 종류를 얻을 수 있습니다.

- static int FOCUS_FIRST : 포커스 이벤트 id의 시작번호.
- static int FOCUS_LAST : 포커스 이벤트 id의 마지막번호.
- static int FOCUS_GAINED : 포커스를 얻었음을 표시.
- static int FOCUS_LOST : 포커스를 잃었음을 표시.
- boolean isTemporary() : 포커스 변화 이벤트가 일시적인지 영구적인지를 조사합니다. 일시적일 경우 true를 반환합니다.

12.4.5.1 FocusListener

포커스 이벤트를 처리할 기능을 정의하는 인터페이스이며, 처리할 클래스는 FocusListener가 정의한 모든 메서드를 구현하거나, 추상 클래스인 FocusAdapter를 상속받아 메서드를 재정의해야 합니다.

12.4.5.2 FocusAdapter

이 클래스는 FocusListener가 정의하고 있는 메서드를 구현하고 있지만, 몸체는 비어 있어 FocusAdapter의 메서드 중 자신이 처리하고자 하는 메서드만 재정의 해주면 됩니다.

FocusListener 메서드

- void focusGained(FocusEvent e) : 컴포넌트가 키보드 포커스를 얻었을 때 호출됩니다.
- void focusLost(FocusEvent e) : 컴포넌트가 키보드 포커스를 잃었을 때 호출됩니다.

다음 프로그램은 Focus 이벤트의 사용 예를 보인 것입니다.

exam/java/chapter12/event/FocusExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
```

```
6:  public class FocusExample implements FocusListener {
7:
8:      private Frame f;
9:      private Button btn1;
10:     private TextField textField;
11:
12:     public FocusExample() {
13:         f = new Frame("Focus Event");
14:         btn1 = new Button("Button1");
15:         textField = new TextField();
16:     }
17:
18:     public void launchFrame() {
19:         btn1.addFocusListener(this);
20:         textField.addFocusListener(this);
21:         f.addWindowListener( new WindowAdapter() {
22:             public void windowClosing(WindowEvent e) {
23:                 System.exit(0);
24:             }
25:         });
26:         f.add(btn1, BorderLayout.SOUTH);
27:         f.add(textField, BorderLayout.NORTH);
28:         f.setSize(300, 100);
29:         f.setVisible(true);
30:     }
31:
32:     public static void main(String[] args) {
33:         FocusExample win = new FocusExample();
34:         win.launchFrame();
35:     }
36:
37:     public void focusGained(FocusEvent e) {
38:         System.out.println(e.getSource() + "의 포커스 얻음");
39:     }
40:     public void focusLost(FocusEvent e) {
41:         System.out.println(e.getSource() + "의 포커스 잃음");
42:     }
43: }
```

5.4.6. KeyEvent

Input의 하위클래스로 컴포넌트 내에서 발생한 키 입력에 대한 기능을 제공해 줍니다. 텍스트필드 종류들의 컴포넌트에서 발생할 수 있는 저수준 이벤트로, 키를 누르거나 놓거나 칠(타이프 pressed and released)때 이벤트가 발생됩니다. 컴포넌트 클래스의 addKeyListener() 메서드에 의해 KeyListener 또는 KeyAdapter 객체에 전달됩니다.

"Key typed"는 고수준 이벤트로 플랫폼이나 키보드 레이아웃에 독립적이며, 문자가 입력되면 발생하고 문자를 정확하게 알 수 있는 방법을 제공해줍니다.
가장 간단한 경우 "Key typed"는 키의 눌림(pressed)과 놓임(released) 이벤트의 조합으로 만들어지며, 키를 계속 누르고 있을 때도 연속적인 "Key typed" 이벤트가 발생합니다. 그러나 F1과 같은 동작 키 또는 Shift 키에 대해서는 이벤트가 발생하지 않습니다.

getKeyChar()는 키에 해당하는 유니코드나 CHAR_UNDEFINED를 반환해 줍니다. 키를 누르거나 놓을 경우 getKeyCode()가 키의 코드 값을 반환하고, 키를 칠 때는 VK_UNDEFINED를 반환합니다. "Key pressed"와 "Key released"는 저수준 이벤트로 플랫폼과 키보드 레이아웃에 의존합니다. 이 이벤트는 키를 누르거나(눌려있는 상태) 놓을 때마다 생성되고, 문자 입력이 없는 F1 키와 Shift 키 등은 modifier 키가 눌렸는지를 알 수 있습니다.

getKeyCode()를 이용하면 해당키에 대한 가상 키 값을 알 수 있습니다. 예를 들어, Shift 키를 누르면 VK_SHIFT 코드를 갖는 KEY_PRESSED 이벤트가 발생하고, 'a'를 누르면 VK_A 코드를 갖는 KEY_PRESSED 이벤트가 발생합니다. 또, 'a' 키를 놓으면 VK_A 코드를 갖는 KEY_RELEASED 이벤트가 발생합니다. 물론, 이 때 KEY_TYPED 이벤트가 발생하고 그 때의 키문자는 'A'가 됩니다.
F1 키와 같이 문자로 나타나지 않는 키는 KEY_TYPED 이벤트를 발생시키지 못합니다. 또 모든 키가 가상 키코드를 생성할 수 있는 것은 아닙니다.

KeyEvent 클래스가 제공하는 기능을 살펴보면 다음과 같습니다.

- static char CHAR_UNDEFINED : 유효하지 않은 유니코드 문자.
- static int KEY_FIRST : 키 이벤트 id의 시작번호.
- static int KEY_LAST : 키 이벤트 id의 마지막번호.
- static int KEY_PRESSED : "Key pressed" 이벤트가 발생했음을 나타내는 상수
- static int KEY_RELEASED : "Key released" 이벤트가 발생했음을 나타내는 상수
- static int KEY_TYPED : "Key typed" 이벤트가 발생했음을 나타내는 상수
- static int VK_ : 가상 키코드 상수.
- char getKeyChar() : 이벤트 내의 키와 관련된 문자를 반환합니다.
- int getKeyCode() : 이벤트 내의 키와 관련된 정수 키코드를 반환합니다.
- static String getKeyModifiersText(int modifiers) : "Shift" 또는 "Ctrl+Shift" 와 같은 문자열을 반환합니다.
- static String getKeyText(intkeyCode) : "HOME", "F1" 또는 "A"와 같은 문자열을 반환합니다.
- boolean isActionKey() : 이벤트 내의 키가 동작키인지를 반환합니다.
- String paramString() : 이벤트를 구분할 수 있는 매개변수 문자열을 반환합니다.
- void setKeyChar(char keyChar) : logical character를 나타내는 keyChar 값을 반환합니다.
- void setKeyCode(int keyCode) : physical key를 가리키는 keyCode 값을 설정합니다.

◦ void setModifiers(int modifiers) : shift, ctrl, alt, meta 등과 같은 추가적인 키를 나타내는 modifier를 설정합니다.

12.4.6.1 KeyListener

키 이벤트를 처리하는 기능을 가지고 있는 인터페이스입니다. 처리하려는 클래스는 KeyListener가 정의한 모든 메서드를 구현하거나, KeyAdapter 클래스의 관련 메서드를 재정의 해야 합니다.

12.4.6.2 KeyAdapter

키 이벤트를 처리하는 추상 클래스로 KeyListener의 메서드를 구현하지만 메서드의 몸체는 비어 있습니다. 처리하려고 하는 메서드만 재정의 해주면 됩니다.

KeyListener 메서드

◦ void keyPressed(KeyEvent e) : 키를 누를 때 호출됩니다.
◦ void keyReleased(KeyEvent e) : 키를 놓았을 때 호출됩니다.
◦ void keyTyped(KeyEvent e) : 키를 칠(눌렀다 놓음) 때 호출됩니다.

다음 프로그램은 키 이벤트의 예를 보인 것입니다.

exam/java/chapter12/event/KeyExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class KeyExample implements KeyListener {
7:
8:     private Frame f;
9:     private TextField textField;
10:
11:    public KeyExample() {
12:       f = new Frame("Key Event");
13:       textField = new TextField();
14:    }
15:
16:    public void launchFrame() {
17:       textField.addKeyListener(this);
18:       f.addWindowListener( new WindowAdapter() {
```

```
19:           public void windowClosing(WindowEvent e) {
20:              System.exit(0);
21:           }
22:        });
23:        f.add(textField, BorderLayout.SOUTH);
24:        f.setSize(300, 100);
25:        f.setVisible(true);
26:     }
27:
28:     public static void main(String[] args) {
29:        KeyExample win = new KeyExample();
30:        win.launchFrame();
31:     }
32:
33:     public void keyPressed(KeyEvent e) {
34:        System.out.println(e.getKeyChar() + " Pressed");
35:     }
36:     public void keyReleased(KeyEvent e) {
37:        System.out.println(e.getKeyChar() + " Released");
38:     }
39:     public void keyTyped(KeyEvent e) {
40:        System.out.println(e.getKeyChar() + " Typed");
41:     }
42: }
```

5.4.7. MouseEvent

마우스 이벤트는 두 가지가 있습니다. click, enter, exit 등과 같은 이벤트와 moves, drag 등의 동작 이벤트가 있습니다. 마우스 이벤트 객체는 컴포넌트 클래스의 MouseListener나 MouseAdapter 객체에 전달됩니다. MouseEvent 객체는 컴포넌트 클래스의 addMouseMotionListener() 메서드를 이용하여 MouseMotionListener나 MouseMotionAdapter 객체에 전달됩니다. 마우스를 누르면 이벤트가 발생하고 modifier 필드에 저장된 후 MouseListeners 객체에 보내집니다.

- static int MOUSE_FIRST : 마우스 이벤트 id의 시작번호.
- static int MOUSE_LAST : 마우스 이벤트 id의 마지막번호.
- static int MOUSE_PRESSED : 마우스 버튼의 누른 상태
- static int MOUSE_RELEASED : 마우스의 버튼이 놓였음
- static int MOUSE_CLICKED : 마우스의 버튼이 클릭되었음
- static int MOUSE_ENTERED : 마우스가 컴포넌트의 영역 내로 들어왔음
- static int MOUSE_EXITED : 마우스가 컴포넌트의 영역 밖으로 나갔음
- static int MOUSE_MOVED : 마우스가 움직였음
- static int MOUSE_DRAGGED : 마우스가 드래그 되었음

- int getClickCount() : 마우스가 클릭 된 수를 반환합니다.
- Point getPoint() : 이벤트 소스 컴포넌트에 상대적인 마우스 커서의 좌표를 반환합니다.
- int getX() : 이벤트 소스 컴포넌트에 상대적인 마우스 커서의 x 좌표를 반환합니다.
- int getY() : 이벤트 소스 컴포넌트에 상대적인 마우스 커서의 y 좌표를 반환합니다.
- boolean isPopupTrigger() : 플랫폼을 위한 팝업 메뉴를 나타나게 할 이벤트인지를 반환합니다.
- void translatePoint(int x, int y) : 이벤트가 발생한 좌표를 현 좌표에서 수평으로 x값만큼, 수직으로 y값만큼 더한 새로운 위치로 옮깁니다.

12.4.7.1 MouseListener

마우스 이벤트를 처리할 기능을 정의한 인터페이스입니다. 마우스 이벤트를 처리하려면 MouseListener 인터페이스의 모든 메서드를 구현하거나, MouseAdapter 클래스의 관련 메서드를 재 정의하여야 합니다.

12.4.7.2 MouseAdapter

마우스 이벤트를 처리하는 abstract adapter 클래스로 MouseListener 인터페이스의 메서드를 구현해야하지만 메서드의 몸체는 비어있습니다. 구현은 처리하고자 하는 이벤트의 메서드만 재정의 하면 됩니다.

MouseListener 메서드

- void mouseClicked(MouseEvent e) : 컴포넌트에서 마우스를 클릭하면 호출됩니다.
- void mouseEntered(MouseEvent e) : 마우스가 컴포넌트 영역 안으로 들어왔을 때 호출됩니다.
- void mouseExited(MouseEvent e) : 마우스가 컴포넌트 영역 밖으로 나갈 때 호출됩니다.
- void mousePressed(MouseEvent e) : 컴포넌트에서 마우스 버튼을 누르면 호출됩니다.
- void mouseReleased(MouseEvent e) : 컴포넌트에서 마우스 버튼을 놓으면 호출됩니다.

12.4.7.3 MouseMotionListener

마우스 동작 이벤트를 처리할 수 있는 기능을 가지고 있는 인터페이스로 MouseMotionListener 인터페이스의 모든 메서드를 구현하거나, MouseMotionAdapter의 관련 메서드를 재정의 해야 합니다.

12.4.7.4 MouseMotionAdapter

마우스 동작 이벤트를 처리하는 클래스로 MouseMotionListener 인터페이스의 메서드를 구현하고 있지만, 메서드의 몸체는 비어있습니다. 구현은 처리하고자 하는 이벤트의 메서드를 재정의 해주면 됩니다.

MouseMotionListener 메서드

- void mouseDragged(MouseEvent e) : 컴포넌트에서 마우스 버튼이 눌려진 상태로 드래그 될 때 호출됩니다.
- void mouseMoved(MouseEvent e) : 컴포넌트에서 마우스가 움직일 때 호출됩니다.

다음 프로그램은 마우스 이벤트의 예를 보인 것입니다. 캔바스를 생성하고 마우스를 클릭하면 클릭한 위치에 좌표가 나타나게 하기 위해 Canvas 클래스를 상속받아 MyCanvas 클래스를 작성했습니다.

exam/java/chapter12/event/MyCanvas.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class MyCanvas extends Canvas implements MouseListener {
7:
8:     int x=0, y=0;
9:     public MyCanvas() {
10:       addMouseListener(this);
11:    }
12:
13:    public void paint(Graphics g) {
14:       g.drawString(x+","+y, x, y);
15:    }
16:
17:    public void mouseClicked(MouseEvent e) {
18:       if(e.getModifiers() == MouseEvent.BUTTON1_MASK) {
19:          System.out.println("Left button clicked");
20:       }else if(e.getModifiers() == MouseEvent.BUTTON2_MASK) {
21:          System.out.println("Center button clicked");
22:       }else if(e.getModifiers() == MouseEvent.BUTTON3_MASK) {
23:          System.out.println("Right button clicked");
24:       }
25:       Point p = e.getPoint();
26:       x = (int)(p.getX());
27:       y = (int)(p.getY());
28:       repaint();
29:    }
30:
31:    public void mousePressed(MouseEvent e) {
32:       System.out.println("mousePressed");
33:    }
```

```
34:     public void mouseReleased(MouseEvent e) {
35:         System.out.println("mouseReleased");
36:     }
37:     public void mouseEntered(MouseEvent e) {
38:         System.out.println("mouseEntered");
39:     }
40:     public void mouseExited(MouseEvent e) {
41:         System.out.println("mouseExited");
42:     }
43: }
```

exam/java/chapter12/event/MouseExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class MouseExample {
7:
8:      private Frame f;
9:      private Canvas canvas;
10:
11:     public MouseExample() {
12:         f = new Frame("Mouse Event");
13:         canvas = new MyCanvas();
14:     }
15:
16:     public void launchFrame() {
17:         f.addWindowListener( new WindowAdapter() {
18:             public void windowClosing(WindowEvent e) {
19:                 System.exit(0);
20:             }
21:         });
22:         f.add(canvas,BorderLayout.CENTER);
23:         f.setSize(300, 200);
24:         f.setVisible(true);
25:     }
26:
27:     public static void main(String[] args) {
28:         MouseExample win = new MouseExample();
29:         win.launchFrame();
30:     }
31: }
```

MouseEvent클래스와 관련된 리스너는 MouseListener 인터페이스와 MouseMotionListener 인터페이스가 있습니다. 그중에서 마우스의 움직임을 감지하기 위해서는 MouseMotionListener 인터페이스를 구현해야 합니다.
다음 코드는 마우스 드래그이벤트를 처리하는 예입니다.

exam/java/chapter12/event/MouseMotionExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class MouseMotionExample implements MouseMotionListener {
7:
8:     private Frame f;
9:
10:    public MouseMotionExample() {
11:       f = new Frame("Mouse Event");
12:    }
13:
14:    public void launchFrame() {
15:       f.addWindowListener( new WindowAdapter() {
16:          public void windowClosing(WindowEvent e) {
17:             System.exit(0);
18:          }
19:       });
20:       f.addMouseMotionListener(this);
21:       f.setSize(300, 200);
22:       f.setVisible(true);
23:    }
24:
25:    public static void main(String[] args) {
26:       MouseMotionExample win = new MouseMotionExample();
27:       win.launchFrame();
28:    }
29:
30:    public void mouseDragged(MouseEvent e) {
31:       System.out.println("mouseDragged");
32:    }
33:
34:    public void mouseMoved(MouseEvent e) {
35:       System.out.println("mouseMoved");
36:    }
37: }
```

5.4.8. ItemEvent

아이템의 선택과 선택해제 여부를 나타내는 고주순 이벤트입니다. 리스트 컴포넌트처럼 ItemSelectable 객체가 생성하는 고수준 이벤트로, ItemSelectable 객체가 포함한 아이템이 선택되거나 선택 해제된 경우 발생합니다. 이 이벤트는 컴포넌트 클래스의 addItemListener() 메서드를 이용하여 모든 ItemListener 객체에 전달됩니다. 이 때, 리스너(Listener)는 마우스의 움직임이나 클릭 같은 개개의 이벤트는 알 필요가 없고 아이템이

선택되거나 선택 해제되는 등의 의미 있는 이벤트만 인지하면 됩니다.

- static int ITEM_FIRST : 아이템 이벤트 id의 시작번호
- static int ITEM_LAST : 아이템 이벤트 id의 마지막번호
- static int SELECTED : 선택 해제된 아이템이 다시 선택된 상태변화
- static int DESELECTED : 선택된 아이템이 선택 해제된 상태변화
- static int ITEM_STATE_CHANGED : 아이템의 상태가 변했음
- Object getItem() : 이벤트가 발생한 아이템 객체를 반환합니다.
- ItemSelectable getItemSelectable() : 이벤트가 발생한 소스 컴포넌트를 반환합니다.
- int getStateChange() : 상태 변화의 종류를 반환합니다.

ItemSelectable 인터페이스는 선택 가능한 아이템을 갖는 컴포넌트의 기능을 정의하며 구체적인 기능을 살펴보면 다음과 같습니다.

- void addItemListener(ItemListener l) : 아이템 Listener를 추가합니다.
- void removeItemListener(ItemListener l) : 주어진 아이템 Listener를 삭제합니다.

12.4.8.1 ItemListener

아이템 이벤트를 처리할 수 있는 기능을 가진 인터페이스로, 처리하려면 메서드를 구현해야 합니다.

ItemListener 메서드

- void itemStateChanged(ItemEvent e) : 아이템 상태가 변했을 때 호출됩니다.

다음 프로그램은 아이템 이벤트의 기능을 이해할 수 있는 예입니다.

exam/java/chapter12/event/ItemExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class ItemExample implements ItemListener {
7:
8:     private Frame f;
9:     private List list;
10:
11:    public ItemExample() {
12:       f = new Frame("Item Event");
13:       list = new List();
```

```
14:     }
15:
16:     public void launchFrame() {
17:        list.add("Solaris");
18:        list.add("Windows");
19:        list.add("Mac");
20:        list.add("MS-DOS");
21:        list.addItemListener(this);
22:        f.addWindowListener( new WindowAdapter() {
23:           public void windowClosing(WindowEvent e) {
24:              System.exit(0);
25:           }
26:        });
27:        f.add(list, BorderLayout.CENTER);
28:        f.pack();
29:        f.setVisible(true);
30:     }
31:
32:     public static void main(String[] args) {
33:        ItemExample win = new ItemExample();
34:        win.launchFrame();
35:     }
36:
37:     public void itemStateChanged(ItemEvent e) {
38:        System.out.println(list.getSelectedItem() + "Selected");
39:     }
40: }
```

5.4.9. TextEvent

텍스트의 변화 유무를 나타내는 이벤트로 TextComponent 같은 객체에 의해 생성되는 고수준 이벤트입니다. 객체에 포함된 텍스트가 변경되었을 때 발생하며, TextComponent 클래스의 addTextListener() 메서드를 이용하여 TextListener 객체에 전달됩니다.

- static int TEXT_FIRST : 텍스트 이벤트 id의 시작번호.
- static int TEXT_LAST : 텍스트 이벤트 id의 마지막번호.
- static int TEXT_VALUE_CHANGED : 텍스트가 변했음

12.4.9.1 TextListener

아이템 이벤트를 처리할 수 있는 인터페이스로 메서드를 구현해주면 됩니다.

TextListener 메서드

◦ void textValueChanged(TextEvent e) : 텍스트의 내용이 변했을 때 호출됩니다.

다음 프로그램은 텍스트 이벤트를 이해할 수 있는 예제입니다.

exam/java/chapter12/event/TextExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class TextExample implements TextListener {
7:
8:     private Frame f;
9:     private TextField textField;
10:
11:    public TextExample() {
12:       f = new Frame("Text Event");
13:       textField = new TextField();
14:    }
15:
16:    public void launchFrame() {
17:       textField.addTextListener(this);
18:       f.addWindowListener( new WindowAdapter() {
19:          public void windowClosing(WindowEvent e) {
20:             System.exit(0);
21:          }
22:       });
23:       f.add(textField, BorderLayout.SOUTH);
24:       f.setSize(300, 100);
25:       f.setVisible(true);
26:    }
27:
28:    public static void main(String[] args) {
29:       TextExample win = new TextExample();
30:       win.launchFrame();
31:    }
32:
33:    public void textValueChanged(TextEvent e) {
34:       System.out.println("Text Changed");
35:    }
36: }
```

5.4.10. WindowEvent

윈도우 상태의 변화를 나타내는 저수준 이벤트로, 윈도우가 열리거나, 닫히거나, 닫히는 중

이거나, 활성화/비활성화 되었거나, 아이콘화/정상화되었을 경우 윈도우 객체에 의해 발생됩니다. 이 이벤트는 윈도우 클래스의 addWindowListener() 메서드를 이용하여 WindowListener 또는 WindowAdapter 객체에 전달됩니다.

- static int WINDOW_ACTIVATED : 윈도우가 활성화되었음.
- static int WINDOW_CLOSED : 윈도우가 닫혔음.
- static int WINDOW_CLOSING : 윈도우가 닫히고 있음.
- static int WINDOW_DEACTIVATED : 윈도우가 비활성화 되었음.
- static int WINDOW_DEICONIFIED : 윈도우가 정상화 되었음.
- static int WINDOW_FIRST : 윈도우 이벤트 id의 시작번호.
- static int WINDOW_ICONIFIED : 윈도우가 아이콘화 되었음.
- static int WINDOW_LAST : 윈도우 이벤트 id의 마지막번호.
- static int WINDOW_OPENED : 윈도우가 열렸음.
- Window getWindow() : 이벤트가 발생한 이벤트 소스 컴포넌트를 반환합니다.

12.4.10.1 WindowListener

윈도우 이벤트를 처리하는 콜백메서드를 정의한 인터페이스입니다. WindowListener 인터페이스를 구현하여 이벤트 핸들러를 정의할 수 있습니다.

WindowListener 메서드

- void windowActivated(WindowEvent e) : 윈도우가 활성화되었을 때 호출됩니다.
- void windowClosed(WindowEvent e) : 윈도우가 닫혔을 때 호출됩니다.
- void windowClosing(WindowEvent e) : 윈도우가 닫히고 있을 때 호출됩니다.
- void windowDeactivated(WindowEvent e) : 윈도우가 비활성화 되었을 때 호출됩니다.
- void windowDeiconified(WindowEvent e) : 윈도우가 정상 상태로 되었을 때 호출됩니다.
- void windowIconified(WindowEvent e) : 윈도우가 아이콘화 되었을 때 호출됩니다.
- void windowOpened(WindowEvent e) : 윈도우가 열렸을 때 호출됩니다.

다음 프로그램은 윈도우 이벤트의 사용 예를 보인 것입니다.

exam/java/chapter12/event/WindowExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class WindowExample implements WindowListener {
7:
8:     private Frame f;
9:
```

```
10:     public WindowExample() {
11:         f = new Frame("Window Event");
12:     }
13:
14:     public void launchFrame() {
15:         f.addWindowListener(this);
16:         f.setSize(300, 100);
17:         f.setVisible(true);
18:     }
19:
20:     public static void main(String[] args) {
21:         WindowExample win = new WindowExample();
22:         win.launchFrame();
23:     }
24:
25:     public void windowOpened(WindowEvent e) {
26:         System.out.println("windowOpened");
27:     }
28:
29:     public void windowClosing(WindowEvent e) {
30:         System.out.println("windowClosing");
31:         System.exit(0);
32:     }
33:
34:     public void windowClosed(WindowEvent e) {
35:           System.out.println("windowClosed");
36:     }
37:     public void windowIconified(WindowEvent e) {
38:           System.out.println("windowIconified");
39:     }
40:     public void windowDeiconified(WindowEvent e) {
41:           System.out.println("windowDeiconified");
42:     }
43:     public void windowActivated(WindowEvent e) {
44:           System.out.println("windowActivated");
45:     }
46:     public void windowDeactivated(WindowEvent e) {
47:           System.out.println("windowDeactivated");
48:     }
49: }
```

12.4.10.2 WindowFocusListener

JDK 1.4에 추가된 윈도우 포커스 변화 이벤트를 받는 인터페이스입니다. 윈도우가 활성창이 되어 포커스를 얻었을 때 호출되는 windowGainedFocus(WindowEvent) 메서드와 윈도우가 비활성화 되어 포커스를 잃었을 때 호출되는

windowLostFocus(WindowEvent) 메서드를 가지고 있습니다.

다음은 윈도우 포커스 예를 나타냈습니다.

exam/java/chapter12/event/WindowFocusExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class WindowFocusExample implements WindowFocusListener {
7:
8:     private Frame f;
9:
10:    public WindowFocusExample() {
11:       f = new Frame("WindowFocus Event");
12:    }
13:
14:    public void launchFrame() {
15:       f.addWindowListener( new WindowAdapter() {
16:          public void windowClosing(WindowEvent e) {
17:             System.exit(0);
18:          }
19:       });
20:       f.addWindowFocusListener(this);
21:       f.setSize(300, 100);
22:       f.setVisible(true);
23:    }
24:
25:    public static void main(String[] args) {
26:       WindowFocusExample win = new WindowFocusExample();
27:       win.launchFrame();
28:    }
29:
30:    public void windowGainedFocus(WindowEvent e) {
31:       System.out.println("윈도우 포커스 얻음");
32:    }
33:
34:    public void windowLostFocus(WindowEvent e) {
35:       System.out.println("윈도우 포커스 잃음");
36:    }
37: }
```

12.4.10.3 WindowStateListener

JDK 1.4에 추가된 윈도우 상태 이벤트를 받기 위한 인터페이스입니다. 윈도우가 아이콘화

되거나, 최대크기, 원래크기로의 상태가 변했을 때 호출되는 windowStateChanged(WindowEvent) 메서드를 가지고 있습니다.

다음은 윈도우 상태변화를 감지할 수 있는 예제입니다.

exam/java/chapter12/event/WindowStateExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class WindowStateExample implements WindowStateListener {
7:
8:     private Frame f;
9:
10:    public WindowStateExample() {
11:       f = new Frame("WindowState Event");
12:    }
13:
14:    public void launchFrame() {
15:       f.addWindowListener( new WindowAdapter() {
16:          public void windowClosing(WindowEvent e) {
17:             System.exit(0);
18:          }
19:       });
20:       f.addWindowStateListener(this);
21:       f.setSize(300, 100);
22:       f.setVisible(true);
23:    }
24:
25:    public static void main(String[] args) {
26:       WindowStateExample win = new WindowStateExample();
27:       win.launchFrame();
28:    }
29:
30:    public void windowStateChanged(WindowEvent e) {
31:       System.out.println("윈도우 상태 변화");
32:    }
33: }
```

12.4.10.4 WindowAdapter

윈도우 이벤트를 처리하는 클래스로 WindowFocusListener, WindowListener, WindowState Listener 인터페이스를 정의하고 메서드를 구현한 클래스입니다. WindowAdapter 클래스를 상속받아 이벤트 핸들러를 정의할 수 있습니다. 어댑터 클래

스를 상속받아 핸들러를 정의하면 이벤트 처리를 하지 않는 메서드들은 재정의 하지 않아도 됩니다.

다음은 윈도우 어댑터를 이용하여 이벤트 핸들러를 구현한 예입니다. 윈도우 종료 버튼이 클릭되었을 때의 이벤트만 처리하기 위해서 windowClosing() 메서드만 재정의 하고 있습니다. 이벤트처리를 하고 싶지 않은 메서드는 재정의 하지 않아도 되기 때문에 코드가 조금 간결해지는 장점이 있습니다. 그러나 어댑터를 이용한 이벤트 핸들러 작성은 다중상속이 안 되는 문제점을 피할 수 없습니다.

exam/java/chapter12/event/WindowAdapterExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class WindowAdapterExample extends WindowAdapter {
7:
8:      private Frame f;
9:
10:     public WindowAdapterExample() {
11:         f = new Frame("WindowAdapter");
12:     }
13:
14:     public void launchFrame() {
15:         f.addWindowListener(this);
16:         f.setSize(300, 100);
17:         f.setVisible(true);
18:     }
19:
20:     public static void main(String[] args) {
21:         WindowAdapterExample win = new WindowAdapterExample();
22:         win.launchFrame();
23:     }
24:
25:     public void windowClosing(WindowEvent e) {
26:         System.out.println("windowClosing");
27:         System.exit(0);
28:     }
29: }
```

5.4.11. MouseWheelEvent

JDK 1.4에 추가되었으며 마우스의 휠 버튼를 사용하기 위한 이벤트입니다.

12.4.11.1 MouseWheelListener

마우스의 휠 이벤트를 처리하려면 MouseWheelListener 인터페이스를 구현하여 이벤트 핸들러를 작성합니다. 마우스 휠이 독작하면 mouseWheelMoved() 메서드가 호출됩니다.

MouseWheelListener 메서드

◦ void mouseWheelMoved(MouseWheelEvent e) : 마우스의 휠을 움직였을 때 호출됩니다.

다음 프로그램은 MouseWheel 이벤트의 사용 예를 보인 것입니다. 이 예제는 두 개의 클래스를 포함하고 있습니다.

exam/java/chapter12/event/MouseWheelExample.java

```
1:  package exam.java.chapter12.event;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:
6:  public class MouseWheelExample{
7:
8:     private Frame f;
9:     private NumCanvas canvas;
10:
11:    public MouseWheelExample() {
12:       f = new Frame("MouseWheel Event");
13:       canvas = new NumCanvas();
14:    }
15:
16:    public void launchFrame() {
17:       f.addWindowListener( new WindowAdapter() {
18:          public void windowClosing(WindowEvent e) {
19:             System.exit(0);
20:          }
21:       });
22:       f.add(canvas);
23:       f.setSize(300, 200);
24:       f.setVisible(true);
25:    }
26:
27:    public static void main(String[] args) {
28:       MouseWheelExample win = new MouseWheelExample();
29:       win.launchFrame();
30:    }
31:
32: }//end main class
33:
34: class NumCanvas extends Canvas implements MouseWheelListener  {
35:
```

```
36:    int i=0;
37:    NumCanvas() {
38:       addMouseWheelListener(this);
39:    }
40:
41:    public void paint(Graphics g) {
42:       g.setFont(new Font("궁서체", Font.BOLD, 100));
43:       g.drawString(i+"", 100, 100);
44:    }
45:
46:    public void mouseWheelMoved(MouseWheelEvent e) {
47:       if(e.getScrollType() == MouseWheelEvent.WHEEL_UNIT_SCROLL) {
48:          i = i + e.getUnitsToScroll()/3;
49:       }
50:       repaint();
51:    }
52: }//end canvas class
```

앞에서 설명한 이벤트들 외에도 여러 가지 많은 이벤트들이 있습니다. 그러나 본 교재에서 모든 이벤트들을 언급할 수는 없습니다. 교재에 나와 있지 않은 이벤트들은 API Documents를 참고하기 바랍니다.

5.5. 요점 정리

1. 이벤트 프로그래밍

위임형 이벤트 모델 방식

이벤트 처리 방법

1. 이벤트 소스 선택 : Button에서 이벤트 발생된다고 가정함
2. 이벤트 핸들러 구현
   Listener 인터페이스를 구현
   class MyEventHandler implements XxxListener { }
   XxxListener 안에 있는 콜백메서드를 구현
3. 이벤트소스와 이벤트 핸들러 연결
   이벤트소스객체.addXxxListener(이벤트핸들러객체)

이벤트 핸들러 구현은 두 가지 방법이 있음
   Listener인터페이스를 implements
   Adapter 클래스를 extends

6. Swing

이 장에서 AWT를 사용할 때 보다 더 다양하고 화려한 애플리케이션을 만들 수 있는 스윙에 대해 상세하게 설명하고자 한다. 스윙이 AWT와 다른 점은 AWT가 플랫폼의형의 컴포넌트를 이용하는 반면, 스윙은 비교적 부하가 적은 경량 컴포넌트를 이용하므로 전체적인 프로그램의 실행이 빠르고 작성이 용이한 이점을 가지고 있습니다.

13장의 주요 내용은 다음과 같습니다.

- JFrame, JPanel
- JButton
- Icon
- JLabel, JCheckBox, JRadioButton, JToggleButton
- JTextField, JTextArea, JTextPane, JPasswordField, JEditorPane
- JScrollBar, JSlider
- JComboBox, JList
- Borders
- 툴팁
- JTabbedPane
- JSplitPane
- 메뉴와 도구상자
- 스윙의 레이아웃 관리자

6.1. 스윙의 기본적인 이해

스윙에서 사용할 수 있는 각종 컴포넌트를 설명하기에 앞서 스윙에 관계된 용어와 정의를 간단하게 익히고 구체적인 내용을 설명하기로 하겠습니다.

6.1.1. JFC

자바는 JDK1.1.x에서 JDK1.2로 넘어갈 때 많은 변화가 있었습니다. 특히 AWT의 단순한 사용자 인터페이스를 개선할 목적으로 5가지 그래픽 API 세트를 추가했는데 바로 JFC(Java Foundation Classes)입니다. 이 라이브러리는 자바 1.2.x 이상에 추가되어 기존 AWT의 단순함을 벗어나 상업용으로 사용할 수 있는 화려한 사용자 인터페이스를 제공하게 되었습니다.

JFC에 포함된 5가지 API를 설명하면 다음과 같습니다.

Ⅰ AWT : AWT는 JDK1.1.x에도 포함되어 있습니다.
Ⅱ Java 2D : 자바 2D는 그래픽 기능을 보다 다양하게 표현하기 위한 라이브러리로 2D 그래픽 프로그램을 개발할 때 이용하는 고수준의 API를 제공합니다. 참고로 Java 2D를 경험하려면 아래와 같이 JDK 1.2 이상 버전에서 JAVA_HOME 폴더의 demo 폴더에서 Java2Demo.html이나 Java2Demo.jar를 실행시켜보세요.

```
%JAVA_HOME%/demo/jfc/Java2D>java -jar
Java2Demo.jar
```

Ⅲ 접근성 : 스크린 확대 등의 기능을 제공하는데, 이를 이용하면 특별한 입출력 장비를 제어할 수 있는 프로그램을 개발할 수 있습니다. 예를 들어 음성으로 인식되는 키보드나, 터치 스크린을 제어하는 프로그램을 개발할 수 있습니다.
Ⅳ 드래그 앤 드롭 : 이 기능은 자바 애플리케이션과 자바가 아닌 기존 프로그램과의 데이터 교환에 사용됩니다.
Ⅴ 스윙(Swing) : 스윙을 이용하면 AWT를 사용할 때 다양하고 화려한 애플리케이션을 만들 수 있습니다. 또, AWT가 플랫폼 의존적인 컴포넌트를 이용하는 반면 스윙은 플랫폼 독립적인 경량 컴포넌트(light weight component)를 이용합니다. 경량 컴포넌트는 AWT 1.1에서 소개되었으며, 운영체제에 독립적인 컴포넌트를 작성할 수 있습니다.

6.1.2. 룩앤필(Look and Feel)

스윙을 설명할 때 가장 먼저 언급하는 용어가 "Look and Feel"입니다. "Look and Feel"

을 "Pluggable Look and Feel"이라고도 하는데, 이는 운영체제와 독립적인 애플리케이션을 개발 할 수 있게 해주는 스윙만의 독특한 특징입니다.
앞서 설명한 AWT로 애플리케이션을 만들어 실행시키면 사용하는 플랫폼에 따라 모양이 달라지는 것을 알 수 있을 것입니다. 즉, AWT 애플리케이션을 윈도우에서 수행했을 때와, 리눅스의 X-Window에서 실행했을 때의 화면이 조금 다릅니다. 이것은 매우 중요한 의미를 갖습니다. AWT가 화면에 GUI 컴포넌트를 그릴 때, 운영체제에게 요청을 하므로 운영체제마다 그리는 방식이나 모양이 달라 결과가 같지 않는 것입니다. 물론 중요한 문제는 아닐 수도 있습니다. 그러나 룩엔필을 이용하면 AWT와 다르게 개발한 프로그램 모습을 운영체제에 상관없이 똑같은 모양으로 보여줄 수 있습니다. 이것은 AWT처럼 그림을 그리는 일을 운영체제에 맡기는 것이 아니라 그 일 조차도 자바가 한다는 것입니다. 그러므로 개발자는 자신만의 고유한 형태로 GUI 컴포넌트를 만들어 사용할 수 있습니다. 예를 들어 5각형 모양의 버튼이 있다면 운영체제의 종류에 관계없이 모양 그대로 화면에 출력할 수 있습니다. 뿐만 아니라, 윈도우 화면에서 X-Window의 기분을 낼 수 있도록 GUI의 모양을 바꿀 수도 있습니다.

다음과 같이 하여 예제 파일을 실행해보세요.

```
%JAVA_HOME%/demo/jfc/SwingSet2>java -jar
SwingSet2.jar
```

AWT의 경우 대부분의 GUI 컴포넌트는 Component 클래스를 상속받아 구현되지만 스윙 GUI 컴포넌트는 JComponent라는 클래스를 상속받아 구현되어 잇습니다. 따라서, 대부분의 컴포넌트 이름이 J로 시작되는 것을 알 수 있습니다.

다음은 자바공식 사이트의 스윙 UI 컴포넌트들에 대한 설명이 있는 사이트입니다. 직접 방문하여 다양한 모양의 룩엔필 컴포넌트들을 채험해 보시기 바랍니다.

```
http://docs.oracle.com/javase/tutorial/uiswing/components
/
```

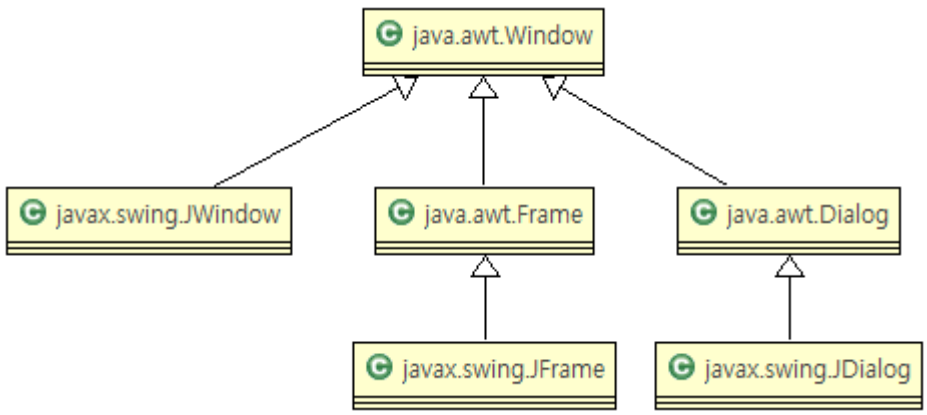
6.2. 스윙 컴포넌트

스윙은 17개의 패키지로 구성되어 있습니다. 이중에서 javax.swing은 스윙의 기본 패키지로서 컴포넌트, 어덥터, 그리고, 기본컴포넌트 모델에 대한 클래스를 가지고 있습니다. 스윙컴포넌트는 AWT컴포넌트보다 수가 많고 기능이 세분화되어 있으며, JComponent를 부모클래스로 갖습니다.

6.2.1. JFrame

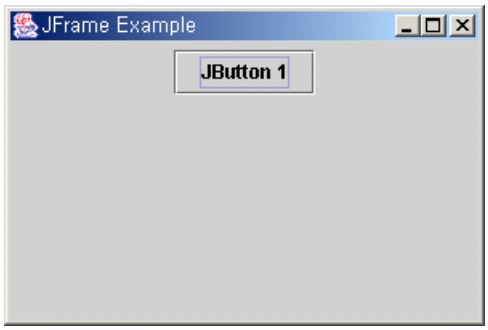
AWT의 Frame, Window, Dialog 클래스처럼 스윙에도 JFrame, JWindow, JDialog 클래스가 있습니다. 하지만 이들 클래스는 스윙의 다른 컴포넌트와는 달리 윈도우클래스를 부모클래스로 가지고 있습니다. 이는 JWindow, JFrame, JDialog가 경량 컴포넌트가 아니라는 것을 의미합니다.

다음은 JWindow와 JFrame의 상속 구조를 나타낸 그림입니다.



JFrame는 AWT의 Frame 클래스와 비슷한 기능을 가지고 있습니다. JFrame이 awt의 Frame과 다른점은 JWindow나 JDialog 클래스처럼 공유된다는 점입니다. Frame에서와 같이 컴포넌트를 Frame에 add() 시키거나 setLayout() 메소드에 의해 레이아웃 관리자를 변경시키는 일은 없지만, 컨텐트영역(content pane)이라는 객체를 얻어 이곳에 컴포넌트를 연결하거나 레이아웃 속성을 변경 할 수 있습니다.

다음은 JFrame을 이용하여 윈도우를 생성한 결과화면입니다.



다음은 앞의 결과화면을 출력시키기 위한 코드입니다.

exam/java/chapter13/swing/JFrameExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import javax.swing.*;
4:  import java.awt.*;
5:
6:  public class JFrameExample {
7:     private JFrame f;
8:     private Container con;
9:
10:    public JFrameExample() {
11:       f = new JFrame("JFrame Example");
12:    }
13:
14:    public void launchFrame() {
15:       con = f.getContentPane();
16:       con.setLayout(new FlowLayout());
17:       con.add(new JButton("JButton 1"));
18:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
19:       f.setSize(300, 200);
20:       f.setVisible(true);
21:    }
22:
23:    public static void main (String args[]) {
24:       JFrameExample win = new JFrameExample();
25:       win.launchFrame();
26:    }
27: }
```

AWT에서는 Frame객체에 위젯(컴포넌트)들을 추가할 수 있었습니다. 그러나 Swing 에서는 JFrame에 위젯들을 직접 추가할 수 없고, 반드시 컨텐트 영역(Content Pane)을 통해서 추가할 수 있습니다. 위의 코드에서 15라인처럼 컨텐트 영역을 사용하는 이유는 윈도우 안쪽이 JRootPane로 구성되어 있어 AWT처럼 윈도우의 내부 작업을 보호할 수 없기 때문입니다.

다음은 AWT의 Frame에서 컴포넌트를 연결하는 방법과, 스윙에서 컨탠트영역을 이용하여 컴포넌트를 연결하는 방법을 비교한 것입니다.

AWT의 Frame에서 컴포넌트를 연결하는 방법은 다음과 같습니다.

```
frame.setLayout(new FlowLayout());
frame.add(component);
```

JFrame에서 컨텐트영역을 사용하는 경우입니다.

```
Container con = jFrame.getContentPane();
con.setLayout(new FlowLayout());
con.add(component);
```

JFrame은 윈도우 종료이벤트 처리를 하지 않아도 프레임 종료버튼을 누르면 창이 사라지게 됩니다. 윈도우에 setDefaultCloseOperation() 메소드를 이용하면 JFrame이 종료될 때 세 가지 동작을 지정할 수 있습니다.

```
f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

⊙ 다음은 setDefaultCloseOperation() 메소의 인자로 올 수 있는 JFrame의 상수 값들입니다.

- DO_NOTHING_ON_CLOSE : AWT의 Frame과 같은 동작을 합니다.
- HIDE_ON_CLOSE : 기본값으로 사용자가 종료하려고 하면 윈도우가 화면에서 사라집니다. setVisible(true)를 사용하면 다시 보이게 할 수 있습니다.
- DISPOSE_ON_CLOSE : 사용자가 종료하려고 하면 완전히 메모리를 반환하고 프로그램을 종료시킵니다.

JRootPane

JRootPane은 글래스영역(Glass Pane)과 레이어드영역(Layered Pane)을 가지는 컨테이너입니다. 글래스영역은 기본 값이 보이지 않는 형태인 반면, 레이어드영역은 보이는 형태입니다.

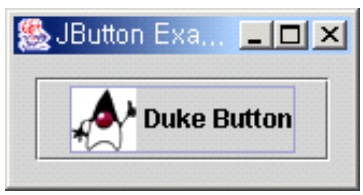
- 레이어드영역은 다시 선택적 메뉴바와 컨텐트 영역이라는 두 개의 객체로 구성되는데, 컨텐트영역은 AWT의 Window, Dialog 또는 Frame의 안쪽에서 일어나는 형태로 작업할 수 있습니다.
- 글래스 영역에 컴포넌트를 배치하면 항상 컨텐트 영역 위에 나타나게 되는데, 이러한 기능은 팝업메뉴나 툴팁 등 문자열이 항상 위에 나타나게 할 때 사용합니다.

6.2.2. JPanel

JPanel은 AWT의 Panel과 기능이 유사합니다. 그래픽 출력시 화면이 떨리거나 깜빡이는 현상을 예방하는 더블버퍼링 기능을 제공합니다.
JPanel 클래스는 다음 JButton 예제에서 설명하겠습니다.

6.2.3. JButton

JButton은 java.awt.Button과 유사하고 ActionListener를 사용하여 이벤트를 처리할 수 있습니다. 또 버튼에 아이콘을 나타내는 등 다양하게 버튼을 표현할 수 있으며, 생성자나 setIcon() 메소드를 이용해 아이콘을 사용합니다.
ImageIcon는 이미지를 아이콘으로 만들어 주는 클래스입니다. 스윙에서 ImageIcon 클래스를 사용하면 Image 클래스를 사용할 때 이미지가 로딩되는 과정을 모니터링 하거나 Image의 객체가 디스크에 저장(serialization)될 수 없는 문제를 해결할 수 있습니다. 이클립스에서 실행할 경우에 그림파일은 프로젝트 폴더에 있어야 합니다.



다음 프로그램은 JButton을 이용하여 아이콘을 갖는 버튼을 나타내는 예입니다.

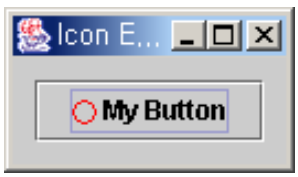
exam/java/chapter13/swing/JButtonExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JButtonExample {
7:
8:     private JFrame f;
9:     private JPanel panel;
10:    private JButton myButton;
11:    private Container con;
12:    private Icon dukeIcon = new ImageIcon("dukeicon.gif");
13:
14:    public JButtonExample() {
15:       f = new JFrame("JButton Example");
16:       panel = new JPanel();
17:       myButton = new JButton("Duke Button");
```

```
18:        myButton.setIcon(dukeIcon);
19:     }
20:
21:     public void launchFrame() {
22:        panel.add(myButton);
23:        con = f.getContentPane();
24:        con.setLayout (new FlowLayout());
25:        con.add(panel);
26:        f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
27:        f.pack();
28:        f.setVisible(true);
29:     }
30:
31:     public static void main(String args[]) {
32:        JButtonExample win = new JButtonExample();
33:        win.launchFrame();
34:     }
35: }
```

6.2.4. 아이콘

아이콘은 고정된 크기의 그림이나 이미지를 표시할 때 사용됩니다. 보통 JButton이나 JComponent의 아이콘으로 사용됩니다. 아이콘처럼 동작하는 객체를 만들려면 Icon 인터페이스를 implements하여 paintIcon(), getIconWidth(), getIconHeignt() 메소드를 구현합니다. paintIcon() 메소드는 그림을 렌더링 하는데 사용됩니다.



다음 프로그램은 Icon인터페이스를 구현하여 임의의 아이콘 클래스를 만든 예를 보인 것입니다. 결과는 빨간색 원모양을 갖는 아이콘입니다.

exam/java/chapter13/swing/RedOvalIcon.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class RedOvalIcon implements Icon {
7:     public void paintIcon (Component c, Graphics g, int x, int y) {
8:        g.setColor(Color.red);
9:        g.drawOval (x, y, getIconWidth(), getIconHeight());
```

```
10:     }
11:
12:     public int getIconWidth() {
13:        return 10;
14:     }
15:
16:     public int getIconHeight() {
17:        return 10;
18:     }
19: }
```

다음 프로그램은 앞에서 만든 아이콘을 버튼에 나타내는 예제입니다.

exam/java/chapter13/swing/IconExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class IconExample {
7:
8:     private JFrame f;
9:     private JPanel panel;
10:    private JButton myButton;
11:    private Container con;
12:    private Icon myIcon = new RedOvalIcon();
13:
14:    public IconExample() {
15:       f = new JFrame("Icon Example");
16:       panel = new JPanel();
17:       myButton = new JButton("My Button");
18:    }
19:
20:    public void launchFrame() {
21:       myButton.setIcon(myIcon);
22:       panel.add(myButton);
23:       con = f.getContentPane();
24:       con.setLayout (new FlowLayout());
25:       con.add(panel);
26:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
27:       f.pack();
28:       f.setVisible(true);
29:    }
30:
31:    public static void main(String args[]) {
32:       IconExample win = new IconExample();
33:       win.launchFrame();
34:    }
35: }
```

6.2.5. JLabel

JLabel은 java.awt.Label과 유사하지만 확장된 기능은 아이콘을 사용하거나 아이콘의 상대적인 위치에 나타내는 기능을 가지고 있습니다.

다음 프로그램은 JLabel의 사용 예입니다.

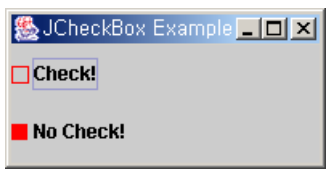
exam/java/chapter13/swing/JLabelExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JLabelExample {
7:
8:     private JFrame f;
9:     private JPanel panel;
10:    private JLabel plainLabel, dukeLabel;
11:    private Container con;
12:
13:    public JLabelExample() {
14:       f = new JFrame("JLable Example");
15:       panel = new JPanel();
16:       plainLabel = new JLabel("Plain Small Label");
17:       dukeLabel = new JLabel("Duke Big Label");
18:    }
19:
20:    public void launchFrame() {
21:       Font dukeFont = new Font("Serif", Font.BOLD + Font.ITALIC, 32);
22:       dukeLabel.setFont(dukeFont);
23:
24:       Icon dukeIcon = new ImageIcon("dukeicon.gif");
25:       dukeLabel.setIcon(dukeIcon);
26:
27:       panel.setLayout(new BorderLayout());
28:       plainLabel.setHorizontalAlignment(JLabel.CENTER);
29:       panel.add(plainLabel, BorderLayout.NORTH);
30:       panel.add(dukeLabel, BorderLayout.CENTER);
31:
32:       con = f.getContentPane();
33:       con.setLayout (new FlowLayout());
```

```
34:         con.add(panel);
35:         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
36:         f.pack();
37:         f.setVisible(true);
38:     }
39:
40:     public static void main(String[] args) {
41:         JLabelExample win = new JLabelExample();
42:         win.launchFrame();
43:     }
44: }
```

6.2.6. JCheckBox

JCheckBox는 AWT의 Checkbox와 유사하지만 라디오버튼을 나타내기 위해 CheckboxGroup을 사용할 필요가 없으며, 체크박스가 선택되었는지의 여부를 아이콘으로 표시할 수 있습니다.



다음 프로그램은 JCheckBox가 선택여부를 아이콘으로 표시하기 위해 임의로 아이콘 클래스를 만든 예입니다.

exam/java/chapter13/swing/ToggleIcon.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class ToggleIcon implements Icon {
7:
8:      private boolean state;
9:
10:     public ToggleIcon(boolean state) {
11:         this.state = state;
12:     }
13:
14:     public void paintIcon (Component c, Graphics g, int x, int y) {
15:         g.setColor(Color.red);
16:         if(state) {
```

```
17:            g.fillRect(x, y, getIconWidth(), getIconHeight());
18:         }else {
19:            g.drawRect(x, y, getIconWidth(), getIconHeight());
20:         }
21:     }
22:
23:     public int getIconWidth() {
24:        return 10;
25:     }
26:
27:     public int getIconHeight() {
28:        return 10;
29:     }
30: }
```

다음 프로그램은 아이콘을 갖는 JCheckBox 예입니다.

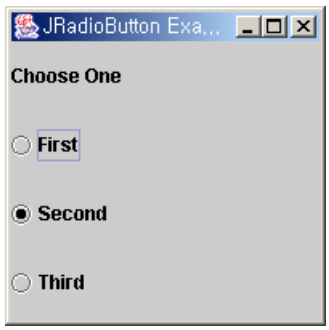
exam/java/chapter13/swing/JCheckBoxExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JCheckBoxExample {
7:
8:     private JFrame f;
9:     private JPanel panel;
10:    private JCheckBox jcb1, jcb2;
11:    private Container con;
12:    private Icon unchecked = new ToggleIcon(false);
13:    private Icon checked = new ToggleIcon(true);
14:
15:    public JCheckBoxExample() {
16:       f = new JFrame("JCheckBox Example");
17:       panel = new JPanel();
18:       jcb1 = new JCheckBox("Check!", false);
19:       jcb2 = new JCheckBox("No Check!", true);
20:    }
21:
22:    public void launchFrame() {
23:       panel.setLayout(new GridLayout(2,1));
24:       jcb1.setIcon(unchecked);
25:       jcb1.setSelectedIcon(checked);
26:       jcb2.setIcon(unchecked);
27:       jcb2.setSelectedIcon(checked);
28:       panel.add(jcb1);
29:       panel.add(jcb2);
30:
31:       con = f.getContentPane();
```

```
32:         con.setLayout (new BorderLayout());
33:         con.add(panel);
34:         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
35:         f.setSize(200, 100);
36:         f.setVisible(true);
37:     }
38:
39:     public static void main(String[] args) {
40:         JCheckBoxExample win = new JCheckBoxExample();
41:         win.launchFrame();
42:     }
43: }
```

6.2.7. JRadioButton

AWT에서 라디오 버튼을 나타낼 때에는 Checkbox와 CheckboxGroup 클래스를 연결하여 작성했지만 스윙은 별도의 JRadioButon 클래스를 제공하고 있습니다. JRadioButton을 ButtonGroup에 연결하여 라디오버튼 동작을 합니다. CheckboxGroup처럼 ButtonGroup도 라디오 버튼의 기능을 할뿐 눈에 보이지는 않습니다.



다음 프로그램은 JRadioButton을 이용하여 라디오 버튼을 나타내는 예입니다.

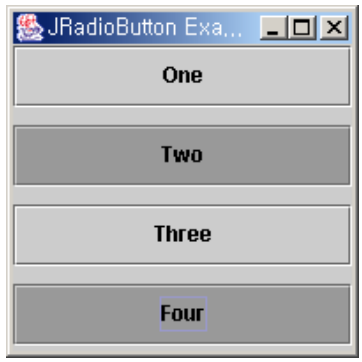
exam/java/chapter13/swing/JRadioButtonExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:  import javax.swing.*;
6:
7:  public class JRadioButtonExample {
8:
9:      private JFrame f;
10:     private JPanel panel;
```

```
11:    private JRadioButton jrb1, jrb2, jrb3;
12:    private ButtonGroup group = new ButtonGroup();
13:    private Container con;
14:
15:    public JRadioButtonExample() {
16:       f = new JFrame("JRadioButton Example");
17:       panel = new JPanel();
18:       jrb1 = new JRadioButton("First");
19:       jrb2 = new JRadioButton("Second");
20:       jrb3 = new JRadioButton("Third");
21:    }
22:
23:    public void launchFrame() {
24:       panel.setLayout(new GridLayout(4,1));
25:
26:       group.add(jrb1);
27:       jrb1.setMnemonic(KeyEvent.VK_1); //Alt+1
28:       group.add(jrb2);
29:       jrb2.setSelected(true);
30:       jrb2.setMnemonic(KeyEvent.VK_2); //Alt+2
31:       group.add(jrb3);
32:       jrb3.setMnemonic(KeyEvent.VK_3); //Alt+3
33:
34:       panel.add(new JLabel("Choose One"));
35:       panel.add(jrb1);    panel.add(jrb2);          panel.add(jrb3);
36:
37:       con = f.getContentPane();
38:       con.setLayout (new BorderLayout());
39:       con.add(panel);
40:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
41:       f.setSize(200, 200);
42:       f.setVisible(true);
43:    }
44:    public static void main(String[] args) {
45:       JRadioButtonExample win = new JRadioButtonExample();
46:       win.launchFrame();
47:    }
48: }
```

6.2.8. JToggleButton

JToggleButton은 JCheckBox와 JRadioButton의 상위 클래스입니다. Button과 유사한 모양을 하고 있지만 버튼을 누르면 눌려진 상태를 유지한다는 점이 다릅니다.



다음 프로그램은 JToggleButton 클래스를 이용하여 토글 버튼을 나타내는 예입니다.

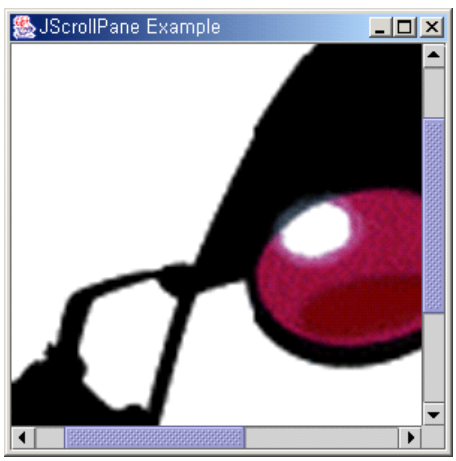
exam/java/chapter13/swing/JToggleButtonExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JToggleButtonExample {
7:
8:     private JFrame f;
9:     private JPanel panel;
10:    private JToggleButton jtb1, jtb2, jtb3, jtb4;
11:    private Container con;
12:
13:    public JToggleButtonExample() {
14:       f = new JFrame("JRadioButton Example");
15:       panel = new JPanel();
16:       jtb1 = new JToggleButton("One");
17:       jtb2 = new JToggleButton("Two");
18:       jtb3 = new JToggleButton("Three");
19:       jtb4 = new JToggleButton("Four");
20:    }
21:
22:    public void launchFrame() {
23:       panel.setLayout(new GridLayout(4, 1, 10, 10));
24:
25:       panel.add(jtb1);
26:       panel.add(jtb2);
27:       panel.add(jtb3);
28:       panel.add(jtb4);
29:
30:       con = f.getContentPane();
31:       con.setLayout(new BorderLayout());
32:       con.add(panel);
33:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
```

```
34:         f.setSize(200, 200);
35:         f.setVisible(true);
36:     }
37:
38:     public static void main(String[] args) {
39:         JToggleButtonExample win = new JToggleButtonExample();
40:         win.launchFrame();
41:     }
42: }
```

6.2.9. JScrollPane

JScrollPane은 AWT 1.1의 ScrollPane처럼 자동스크롤 기능을 가지고 있습니다. 이 클래스는 ScrollPaneLayout 방식을 사용하여 출력합니다. 중요한 사항은 JScrollPane이 스크롤될 때 객체를 추가하는데 JViewport를 사용한다는 점입니다. 사용법은 getViewport() 메소드를 사용한 후 객체를 뷰포트에 추가시킵니다. JViewport는 보이는 화면보다 더 큰 영역을 사용할 수 있는 기능을 제공하여 JScrollPane와 사용될 수 있습니다.



다음 프로그램은 JScrollPane을 나타내는 예입니다.

exam/java/chapter13/swing/JScrollPaneExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JScrollPaneExample {
7:
```

```
8:      private JFrame f;
9:      private JPanel panel;
10:     private Icon dukeIcon;
11:     private JLabel dukeLabel;
12:     private JScrollPane scrollPane;
13:
14:     private Container con;
15:
16:     public JScrollPaneExample() {
17:        f = new JFrame("JScrollPane Example");
18:        panel = new JPanel();
19:        dukeIcon = new ImageIcon("bigDuke.gif");
20:        dukeLabel = new JLabel(dukeIcon);
21:        scrollPane = new JScrollPane(dukeLabel);
22:     }
23:
24:     public void launchFrame() {
25:        panel.setLayout(new BorderLayout());
26:        panel.add(scrollPane, BorderLayout.CENTER);
27:        con = f.getContentPane();
28:        con.setLayout(new BorderLayout());
29:        con.add(panel);
30:        f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
31:        f.setSize(300, 300);
32:        f.setVisible(true);
33:     }
34:
35:     public static void main(String[] args) {
36:        JScrollPaneExample win = new JScrollPaneExample();
37:        win.launchFrame();
38:     }
39: }
```

6.2.10. JTextComponents

JTextComponent는 javax.swing.text 패키지에 있으며, 단순한 편집기 기능을 가지고 있는 컴포넌트입니다.

JTextComponent의 주요 메소드를 살펴보면 다음과 같습니다.

- copy() : 현재 선택된 블록의 문자열을 클립보드에 복사합니다.
- cut() : 현재 선택된 블록의 문자열을 클립보드에 옮겨놓습니다.
- paste() : 클리보드에 있는 내용을 텍스트 컴포넌트에 붙입니다.
- getSelectedText() : 텍스트 컴포넌트에서 선택되어 있는 문자열을 반환합니다.

- setSelectionStart() : 블록의 시작점은 설정합니다.
- setSelectionEnd() : 블록의 끝점을 설정합니다.
- selectAll() : 텍스트 컴포넌트의 모든 내용을 선택합니다.
- replaceSelection() : 현재 선택되어 있는 문자열을 주어진 문자열로 바꿉니다.
- getText() : 텍스트 컴포넌트로부터 문자열을 반환합니다.
- setText() : 텍스트 컴포넌트에 문자열을 지정한다.
- setEditable() : 편집 가능 여부를 설정합니다.
- setCaretPosition() : 캐럿 위치를 설정합니다.

JTextComponent는 직접 객체를 작성할 수는 없고 상속되는 다른 문자열처리 클래스에서 사용합니다. JTextComponent객체는 AWT의 텍스트 컴포넌트와 마찬가지로 패널에 연결될 수 있습니다. JTextComponent의 하위 클래스로는 JTextField, JTextArea, JEditorPane, JPasswordField, JTextPane 등이 있습니다.

6.2.10.1. JTextField와 JTextArea

JTextArea에 JScrollPane을 연결하여 사용하는 점을 제외하면 JTextField와 JTextArea는 AWT의 java.awt.TextField, java.awt.TextArea와 유사합니다.
JTextField는 setHorizontalAlignment() 메소드를 이용 문자열의 위치를 조정할 수 있습니다. 지정할 수 있는 위치는 LEFT, CENTER, RIGHT가 있으며 기본값은 LEFT입니다.

6.2.10.2. JTextPane

JTextPane은 정형화된 텍스트 이미지의 표시 기능을 제공하는 문자 편집기입니다.

다음 프로그램은 JTextField, JTextArea, JTextPane의 사용 예를 보인 것입니다.

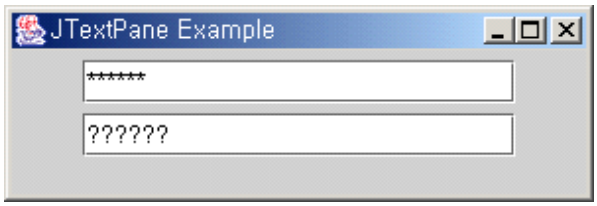
exam/java/chapter13/swing/JTextPaneExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:  import javax.swing.text.*;
6:
7:  public class JTextPaneExample {
8:     private JFrame f;
9:     private JPanel panel;
10:    private JTextField jtField;
11:    private JTextArea jtArea;
```

```
12:     private JTextPane jtPane;
13:     private Container con;
14:
15:     public JTextPaneExample() {
16:        f = new JFrame("JTextPane Example");
17:        panel = new JPanel();
18:        jtField = new JTextField();
19:        jtArea = new JTextArea();
20:        jtPane = new JTextPane();
21:     }
22:
23:     public void launchFrame() {
24:        jtField.setText("JTextField : Single Line");
25:        jtArea.setText("JTextArea :\nMulti Line");
26:
27:        MutableAttributeSet attr = new SimpleAttributeSet();
28:        StyleConstants.setFontFamily(attr, "Serif");
29:        StyleConstants.setFontSize(attr, 18);
30:        StyleConstants.setBold(attr, true);
31:        jtPane.setCharacterAttributes(attr, false);
32:        jtPane.setText("JTextPane :\nMulitLine");
33:
34:        panel.setLayout(new BorderLayout());
35:        panel.add(jtField, BorderLayout.NORTH);
36:        panel.add(jtArea, BorderLayout.CENTER);
37:        panel.add(jtPane, BorderLayout.SOUTH);
38:        con = f.getContentPane();
39:        con.add(panel);
40:        f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
41:        f.setSize(200, 200);
42:        f.setVisible(true);
43:     }
44:     public static void main(String[] args) {
45:        JTextPaneExample win = new JTextPaneExample();
46:        win.launchFrame();
47:     }
48: }
```

6.2.10.3. JPasswordField

JPasswordField는 JTextField와 같지만 사용자가 입력한 문자를 보여주지 않고 대신에 "*" 문자를 표시하는 텍스트 필드입니다. 대체 텍스트를 다른 문자를 표시하고 싶으면 setEchoChar() 메소드를 사용하면 변경이 가능합니다.



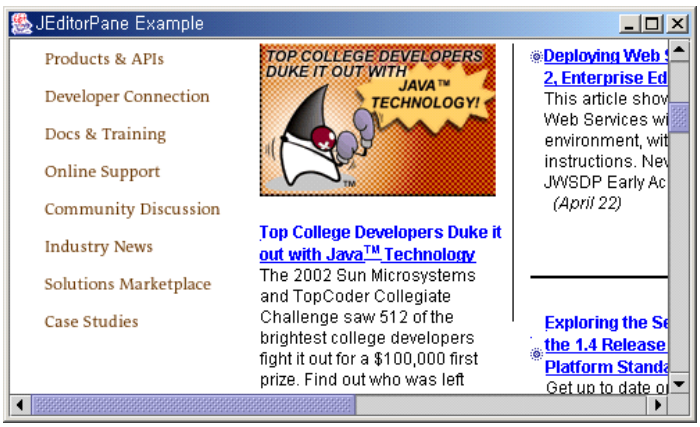
다음 프로그램은 JPasswordField를 이용하여 패스워드 필드를 나타내는 예입니다.

exam/java/chapter13/swing/JPasswordFieldExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JPasswordFieldExample {
7:     private JFrame f;
8:     private JPanel panel;
9:     private JPasswordField jpass1, jpass2;
10:    private Container con;
11:
12:    public JPasswordFieldExample() {
13:       f = new JFrame("JTextPane Example");
14:       panel = new JPanel();
15:       jpass1 = new JPasswordField(20);
16:       jpass2 = new JPasswordField(20);
17:    }
18:
19:    public void launchFrame() {
20:       panel.add(jpass1);
21:       jpass2.setEchoChar('?');
22:       panel.add(jpass2);
23:       con = f.getContentPane();
24:       con.add(panel);
25:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
26:       f.setSize(300, 100);
27:       f.setVisible(true);
28:    }
29:
30:    public static void main(String[] args) {
31:       JPasswordFieldExample win = new JPasswordFieldExample();
32:       win.launchFrame();
33:    }
34: }
```

6.2.10.4. JEditorPane

JEditorPane은 HTML문서나 RTF와 같은 특별한 형식의 문서를 표시하고 편집하는 클래스입니다. 이 클래스를 이용하면 도움말을 HTML로 제공하거나 URL을 이용 다른 홈페이지의 문서를 보여줄 수 있습니다. JEditorPane에서 이벤트 처리를 하면 하이퍼링크를 사용할 수 있으며, setEditable(false)로 설정하면 읽기만 가능합니다.



다음 프로그램은 JEditorPane에 자바 홈페이지를 표시한 예입니다.

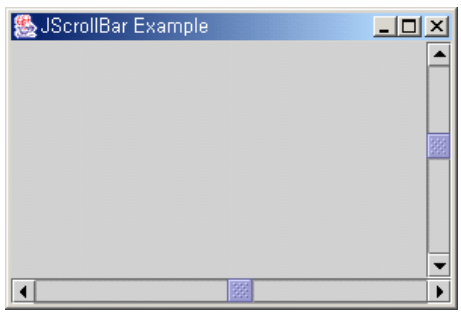
exam/java/chapter13/swing/JEditorPaneExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:  import java.io.*;
6:  import java.net.*;
7:
8:  public class JEditorPaneExample {
9:     private JFrame f;
10:    private JPanel panel;
11:    private JEditorPane jeditPane;
12:    private JScrollPane jsPane;
13:    private Container con;
14:
15:    public JEditorPaneExample() {
16:       f = new JFrame("JEditorPane Example");
17:       panel = new JPanel();
18:       jeditPane = new JEditorPane();
19:       jsPane = new JScrollPane();
20:    }
21:
22:    public void launchFrame() {
23:       try{
```

```
24:            URL url = new URL("http://java.sun.com");
25:            jeditPane.setPage(url);
26:         }catch(IOException e) {
27:            e.printStackTrace();
28:         }
29:
30:         jeditPane.setEditable(false);
31:         jsPane.getViewport().add(jeditPane);
32:         panel.setLayout(new BorderLayout());
33:         panel.add(jsPane, BorderLayout.CENTER);
34:
35:         con = f.getContentPane();
36:         con.setLayout(new BorderLayout());
37:         con.add(panel);
38:         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
39:         f.setSize(400, 200);
40:         f.setVisible(true);
41:      }
42:
43:      public static void main(String[] args) {
44:         JEditorPaneExample win = new JEditorPaneExample();
45:         win.launchFrame();
46:      }
47: }
```

6.2.11. JScrollBar

JScrollBar는 java.awt.Scrollbar의 스윙버전입니다.



다음 프로그램은 JScrollBar의 사용 예입니다.

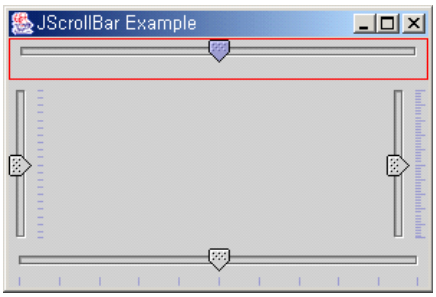
exam/java/chapter13/swing/JScrollBarExample.java

```
1:  package exam.java.chapter13.swing;
2:
```

```
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JScrollBarExample {
7:  private JFrame f;
8:     private JPanel panel;
9:     private JScrollBar vBar, hBar;
10:    private Container con;
11:
12:    public JScrollBarExample() {
13:       f = new JFrame("JScrollBar Example");
14:       panel = new JPanel();
15:
16:       //JScrollBar 파라미터 : orientation, value, extent, min, max
17:       vBar = new JScrollBar(JScrollBar.VERTICAL, 40, 5, 0, 100);
18:       hBar = new JScrollBar(JScrollBar.HORIZONTAL, 50, 5, 0, 100);
19:    }
20:
21:    public void launchFrame() {
22:       panel.setLayout(new BorderLayout());
23:       panel.add(vBar, BorderLayout.EAST);
24:       panel.add(hBar, BorderLayout.SOUTH);
25:
26:       con = f.getContentPane();
27:       con.add(panel);
28:
29:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
30:       f.setSize(300, 200);
31:       f.setVisible(true);
32:    }
33:
34:    public static void main(String[] args) {
35:       JScrollBarExample win = new JScrollBarExample();
36:       win.launchFrame();
37:    }
38: }
```

6.2.12. JSlider

JSlider는 JScrollBar와 비슷하지만 스크롤바가 있는곳에 눈금을 표시할 수 있습니다.



다음 프로그램은 JSlider를 나타내는 예입니다.

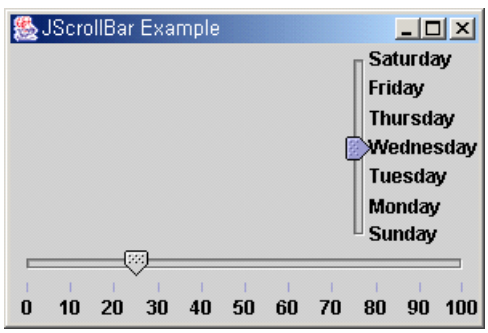
exam/java/chapter13/swing/JSliderExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JSliderExample {
7:     private JFrame f;
8:     private JPanel panel;
9:     private JSlider eastSlider, westSlider;
10:    private JSlider southSlider, northSlider;
11:    private Container con;
12:
13:    public JSliderExample() {
14:       f = new JFrame("JScrollBar Example");
15:       panel = new JPanel();
16:       eastSlider = new JSlider(JSlider.VERTICAL,0,100,50);
17:       westSlider = new JSlider(JSlider.VERTICAL,0,100,50);
18:       southSlider = new JSlider(JSlider.HORIZONTAL,0,100,50);
19:       northSlider = new JSlider(JSlider.HORIZONTAL,0,100,50);
20:    }
21:
22:    public void launchFrame() {
23:       panel.setLayout(new BorderLayout());
24:       eastSlider.setPaintTicks(true);
25:       eastSlider.setMajorTickSpacing(10);
26:       eastSlider.setMinorTickSpacing(2);
27:       panel.add(eastSlider, BorderLayout.EAST);
28:       westSlider.setPaintTicks(true);
29:       westSlider.setMinorTickSpacing(5);
30:       panel.add(westSlider, BorderLayout.WEST);
31:       southSlider.setPaintTicks(true);
32:       southSlider.setMajorTickSpacing(10);
33:       panel.add(southSlider, BorderLayout.SOUTH);
34:       northSlider.setPaintTicks(true);
35:       northSlider.setBorder( BorderFactory.createLineBorder(Color.red) );
36:       panel.add(northSlider, BorderLayout.NORTH);
37:
```

```
38:         con = f.getContentPane();
39:         con.add(panel);
40:         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
41:         f.setSize(300, 200);
42:         f.setVisible(true);
43:     }
44:
45:     public static void main(String[] args) {
46:         JSliderExample win = new JSliderExample();
47:         win.launchFrame();
48:     }
49: }
```

앞의 예에서처럼 JSlider는 단순한 눈금표시 외에도 라벨이나 숫자 그리고 컴포넌트를 표시할 수 있습니다. setPaintLabels(true)를 사용하면 숫자라벨을 나타낼 수 있습니다. 눈금 간격을 10으로 설정하고 슬라이더의 영역이 0에서 100까지이면 슬라이더는 0, 10, 20, ... 100으로 표시됩니다.

해시테이블(Hashtable)을 이용하면 사용자가 원하는 라벨을 편리하게 표시할 수 있습니다. 해시테이블에 key는 정수 값을, value는 JLabel을 쌍으로 저장한 다음 JSlider의 setLabelTable() 메소드를 이용하면 JSlider 에 라벨 표시를 좀 더 쉽게 할 수 있습니다.



다음 프로그램은 JSlider에 라벨을 부여한 예입니다.

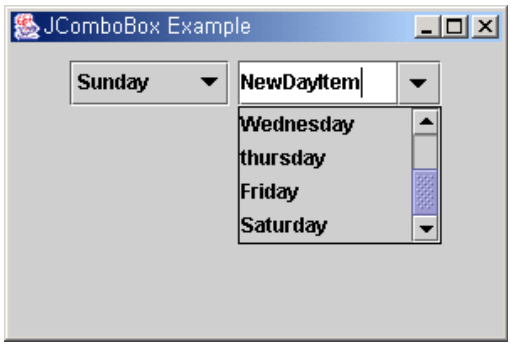
exam/java/chapter13/swing/JSliderExample2.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:  import java.util.*;
6:
7:  public class JSliderExample2 {
8:     private JFrame f;
9:     private JPanel panel;
```

```
10:    private JSlider eastSlider;
11:    private JSlider southSlider;
12:    private Container con;
13:
14:    public JSliderExample2() {
15:       f = new JFrame("JScrollBar Example");
16:       panel = new JPanel();
17:       eastSlider = new JSlider(JSlider.VERTICAL,0,6,3);
18:       southSlider = new JSlider(JSlider.HORIZONTAL,0,100,25);
19:    }
20:
21:    public void launchFrame() {
22:       panel.setLayout(new BorderLayout());
23:       Hashtable<Integer, JLabel> ht = new Hashtable<>();
24:       ht.put(new Integer(0), new JLabel("Sunday"));
25:       ht.put(new Integer(1), new JLabel("Monday"));
26:       ht.put(new Integer(2), new JLabel("Tuesday"));
27:       ht.put(new Integer(3), new JLabel("Wednesday"));
28:       ht.put(new Integer(4), new JLabel("Thursday"));
29:       ht.put(new Integer(5), new JLabel("Friday"));
30:       ht.put(new Integer(6), new JLabel("Saturday"));
31:       eastSlider.setLabelTable(ht);
32:       eastSlider.setPaintLabels(true);
33:       panel.add(eastSlider, BorderLayout.EAST);
34:
35:       southSlider.setPaintTicks(true);
36:       southSlider.setMajorTickSpacing(10);
37:       southSlider.setPaintLabels(true);
38:       panel.add(southSlider, BorderLayout.SOUTH);
39:
40:       con = f.getContentPane();
41:       con.add(panel);
42:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
43:       f.setSize(300, 200);
44:       f.setVisible(true);
45:    }
46:    public static void main(String[] args) {
47:       JSliderExample2 win = new JSliderExample2();
48:       win.launchFrame();
49:    }
50: }
```

6.2.13. JComboBox

JComboBox는 AWT의 Choice와 유사하지만 리스트 이외의 항목을 새로 추가할 수 있는 편집기능을 제공합니다.



다음 프로그램은 JComboBox의 사용 예입니다.

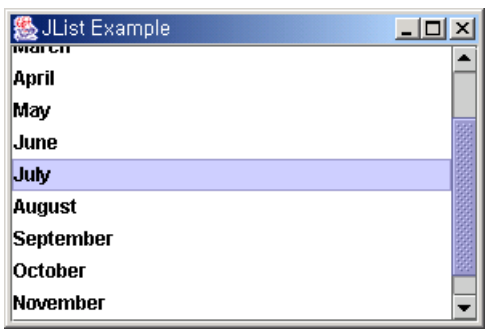
exam/java/chapter13/swing/JComboBoxExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JComboBoxExample {
7:     private JFrame f;
8:     private JPanel panel;
9:     private String day[] = {
10:           "Sunday", "Monday", "Tuesday", "Wednesday",
11:           "Thursday", "Friday", "Saturday"          };
12:    private JComboBox<String> dayList1, dayList2;
13:    private Container con;
14:
15:    public JComboBoxExample() {
16:       f = new JFrame("JFrame Example");
17:       panel = new JPanel();
18:       dayList1 = new JComboBox<>();
19:       dayList2 = new JComboBox<>();
20:    }
21:
22:    public void launchFrame() {
23:       con = f.getContentPane();
24:       con.setLayout(new FlowLayout());
25:
26:       for(int i=0; i < day.length; i++) {
27:          dayList1.addItem(day[i]);
28:          dayList2.addItem(day[i]);
29:       }
30:       dayList2.setEditable(true);
31:       dayList2.setMaximumRowCount(4);
32:       panel.add(dayList1);
33:       panel.add(dayList2);
34:
```

```
35:         con.add(panel);
36:         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
37:         f.setSize(300, 200);
38:         f.setVisible(true);
39:     }
40:
41:     public static void main (String args[]) {
42:         JComboBoxExample win = new JComboBoxExample();
43:         win.launchFrame();
44:     }
45: }
```

6.2.14. JList

JList는 AWT의 List와 유사하지만 사용이 더 편리합니다. 리스트에 항목을 지정할 때 String[] 이나 Vector의 객체를 setListData() 메소드에 지정할 수 있습니다. AWT의 List는 기본으로 스크롤바를 제공하지만, JList는 JScrollPane에 JList를 연결해야 스크롤바가 나타납니다.



다음 프로그램은 스크롤바를 갖는 JList 예입니다.

exam/java/chapter13/swing/JListExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JListExample {
7:      private JFrame f;
8:      private JPanel panel;
9:      private JList<String> dayList;
10:     private Container con;
11:     private JScrollPane jScroll;
```

```
12:     private String year[] = {
13:             "January", "February", "March", "April", "May",
14:             "June", "July", "August", "September", "October",
15:             "November", "DEcember"              };
16:
17:     public JListExample() {
18:         f = new JFrame("JList Example");
19:         panel = new JPanel();
20:         dayList = new JList<>(year);
21:         jScroll = new JScrollPane(dayList);
22:     }
23:
24:     public void launchFrame() {
25:         panel.setLayout(new BorderLayout());
26:         panel.add(jScroll, BorderLayout.CENTER);
27:
28:         con = f.getContentPane();
29:         con.setLayout(new BorderLayout());
30:         con.add(panel);
31:         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
32:         f.setSize(300, 200);
33:         f.setVisible(true);
34:     }
35:
36:     public static void main (String args[]) {
37:         JListExample win = new JListExample();
38:         win.launchFrame();
39:     }
40: }
```

6.2.15. Borders

javax.swing.border 패키지에는 컴포넌트의 테두리를 표시하는 여러 클래스가 있습니다. border 클래스를 만들려면 인터페이스를 구현한 다음 작성하는데, 인터페이스에는 다음과 같은 세 가지 메소드를 통해 테두리를 정의합니다.

- Insets getBorderInsets(Component c) : 테두리를 그리는데 필요한 영역을 지정합니다.
- boolean isBorderOpaque() : 테두리 영역이 투명한지를 판단합니다.
- void paintBorder (Component c, Graphics g, int x, int y, int width, int height) : 지정된 영역에 테두리를 그리는 방법을 지정합니다.

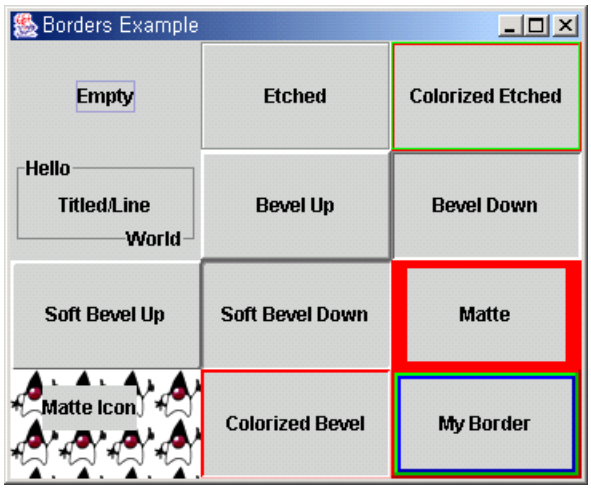
다음 프로그램은 Border인터페이스를 구현한 다음 임의의 border 클래스를 정의하는 예입니다.

exam/java/chapter13/swing/MyBorder.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.border.*;
5:
6:  public class MyBorder implements Border {
7:
8:     Color color;
9:
10:    public MyBorder (Color c) {
11:       color = c;
12:    }
13:
14:    public void paintBorder (Component c, Graphics g, int x, int y, int width, int
    height) {
15:       Insets insets = getBorderInsets(c);
16:       g.setColor (color);
17:       g.fillRect (x, y, 2, height);
18:       g.fillRect (x, y, width, 2);
19:       g.setColor (color.darker());
20:       g.fillRect (x+width-insets.right, y, 2, height);
21:       g.fillRect (x, y+height-insets.bottom, width, 2);
22:    }
23:    public boolean isBorderOpaque() {
24:       return false;
25:    }
26:    public Insets getBorderInsets(Component c) {
27:       return new Insets (2, 2, 2, 2);
28:    }
29: }
```

테두리에 대한 동작은 JComponent에 정의되어 있으며 상속받아 이용합니다. 스윙은 9가지 종류의 테두리를 가지고 있으며, 사용자가 직접 만들어 사용할 수도 있습니다.

- AbstractBorder : Border 인터페이스를 구현한 추상클래스 입니이다.
- BevelBorder : 3D 모양의 테두리를 갖습니다.
- CompoundBorder : 중첩된 테두리를 갖습니다.
- EmptyBorder : 선이 그려지지 않는 테두리를 갖습니다.
- EtchedBorder : 홈이 파인 것처럼 보이는 테두리를 갖습니다.
- LineBorder : 단색의 테두리를 갖습니다.
- MatteBorder : 아이콘이나 색을 가지는 테두리를 갖습니다.
- SoftBevelBorder : 부드러운 색의 3D 테두리를 갖습니다.
- TitledBorder : 임의의 위치에 제목을 표시한 테두리를 갖습니다.



다음 프로그램은 여러 가지 경계선을 갖는 버튼의 예입니다.

exam/java/chapter13/swing/BorderExample.java

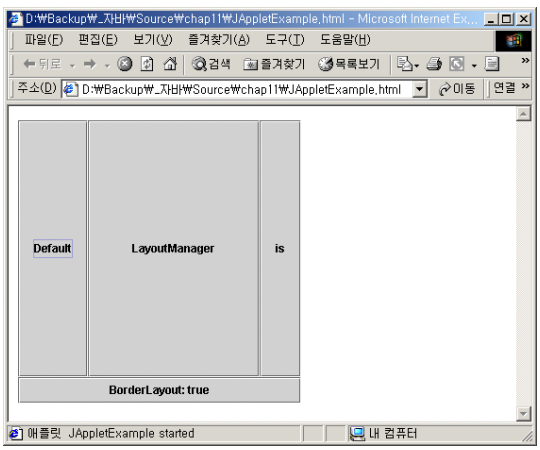
```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:  import javax.swing.border.*;
6:
7:  public class BorderExample {
8:     private JFrame f;
9:     private JPanel panel;
10:    private Container con;
11:
12:    public BorderExample() {
13:       f = new JFrame("Borders Example");
14:       panel = new JPanel();
15:    }
16:
17:    public void launchFrame() {
18:       panel.setLayout(new GridLayout(4, 3));
19:
20:       JButton emptyButton = new JButton("Empty");
21:       emptyButton.setBorder (new EmptyBorder (1,1,1,1));
22:       panel.add(emptyButton);
23:
24:       JButton etchedButton = new JButton("Etched");
25:       etchedButton.setBorder (new EtchedBorder ());
26:       panel.add(etchedButton);
27:
28:       JButton cEtchedButton = new JButton("Colorized Etched");
29:       cEtchedButton.setBorder (new EtchedBorder (Color.red, Color.green));
30:       panel.add(cEtchedButton);
31:
```

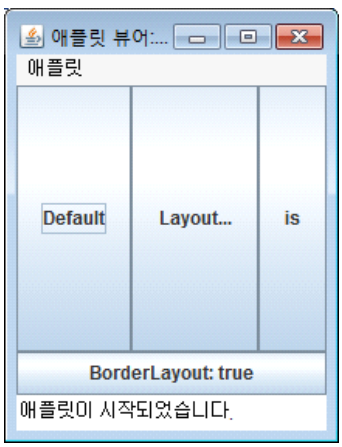
```
32:        JButton titledButton = new JButton("Titled/Line");
33:        titledButton.setBorder(
               new TitledBorder (
                   new TitledBorder(LineBorder.createGrayLineBorder(), "Hello"),
                   "World",
                   TitledBorder.RIGHT,
                   TitledBorder.BOTTOM)
           );
34:        panel.add(titledButton);
35:
36:        JButton bevelUpButton = new JButton("Bevel Up");
37:        bevelUpButton.setBorder(new BevelBorder(BevelBorder.RAISED));
38:        panel.add(bevelUpButton);
39:
40:        JButton bevelDownButton = new JButton("Bevel Down");
41:        bevelDownButton.setBorder(new BevelBorder(BevelBorder.LOWERED));
42:        panel.add(bevelDownButton);
43:
44:        JButton b = new JButton("Soft Bevel Up");
45:        b.setBorder(new SoftBevelBorder(SoftBevelBorder.RAISED));
46:        panel.add(b);
47:
48:        b = new JButton("Soft Bevel Down");
49:        b.setBorder(new SoftBevelBorder(SoftBevelBorder.LOWERED));
50:        panel.add(b);
51:
52:        b = new JButton("Matte");
53:        b.setBorder(new MatteBorder(5,10,5,10, Color.red));
54:        panel.add(b);
55:
56:        b = new JButton("Matte Icon");
57:        Icon icon = new ImageIcon("dukeicon.gif");
58:        b.setBorder(new MatteBorder(10, 20, 30, 40, icon));
59:        panel.add(b);
60:
61:        b = new JButton ("Colorized Bevel");
62:        b.setBorder(new BevelBorder(BevelBorder.RAISED, Color.red,
    Color.pink));
63:        panel.add(b);
64:
65:        b = new JButton("My Border");
66:        b.setBorder(new CompoundBorder(
67:           new MyBorder(Color.red),
68:           new CompoundBorder(new MyBorder(Color.green),
69:           new MyBorder(Color.blue))));
70:        panel.add(b);
71:
72:        con = f.getContentPane();
73:        con.setLayout(new BorderLayout());
74:        con.add(panel, BorderLayout.CENTER);
```

```
75:         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
76:         f.pack();
77:         f.setVisible(true);
78:     }
79:
80:     public static void main (String args[]) {
81:         BorderExample win = new BorderExample();
82:         win.launchFrame();
83:     }
84: }
```

6.2.16. JApplet

스윙 컴포넌트를 처리하는 애플릿을 작성하려면 AWT의 애플릿 클래스 대신 JApplet클래스를 상속받아 사용할 수있습니다. JApplet을 이용하면 JMenuBar를 사용할 수 있을 뿐만 아니라 스윙 컴포넌트의 그림 그리기 기능을 수행할 수 있습니다. JFrame처럼 JApplet은 JContentPane을 사용하여 컴포넌트를 연결시키게 되며, 기본 레이아웃 관리자는 BorderLayout입니다.





다음은 JApplet 예입니다.

exam/java/chapter13/swing/JAppletExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JAppletExample extends JApplet {
7:
```

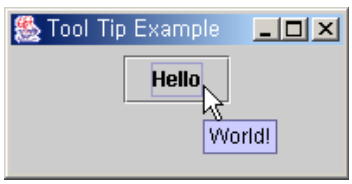
```
8:     private static final long serialVersionUID = -1L;
9:
10:    public void init () {
11:       Container con = getContentPane();
12:       JButton jb = new JButton ("Default");
13:       con.add (jb, BorderLayout.WEST);
14:       jb = new JButton ("LayoutManager");
15:       con.add (jb, BorderLayout.CENTER);
16:       jb = new JButton ("is");
17:       con.add (jb, BorderLayout.EAST);
18:       jb = new JButton ("BorderLayout: " + (con.getLayout() instanceof
    BorderLayout));
19:       con.add (jb, BorderLayout.SOUTH);
20:    }
21: }
```

JAppletExample.html

```
1:  <applet code=JAppletExample.class width=300 height=300>
2:  </applet>
```

6.2.17. 툴팁(tool tip)

툴팁이란 마우스가 화면상의 특정한 객체 위를 지날 때 표시되는 문자열을 의미합니다. 스윙은 JToolTip 클래스에서 이러한 기능을 제공합니다. 툴팁을 사용하려면 JComponent 클래스의 setToolTipText() 메소드를 이용합니다.



다음 프로그램은 버튼에 툴팁이 나오게 하는 예입니다.

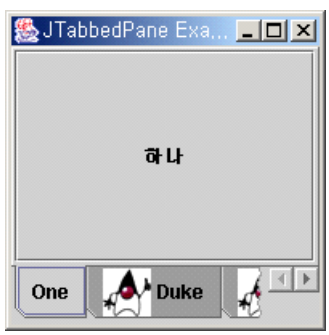
exam/java/chapter13/swing/JToolTipExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import javax.swing.*;
4:  import java.awt.*;
5:
6:  public class JToolTipExample {
7:     private JFrame f;
8:     private JButton myButton;
9:     private Container con;
10:
```

```
11:     public JToolTipExample() {
12:         f = new JFrame("Tool Tip Example");
13:         myButton = new JButton("Hello");
14:     }
15:
16:     public void launchFrame() {
17:         con = f.getContentPane();
18:         con.setLayout(new FlowLayout());
19:         con.add(myButton);
20:         myButton.setToolTipText("World!");
21:         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
22:         f.setSize(200, 100);
23:         f.setVisible(true);
24:     }
25:
26:     public static void main (String args[]) {
27:         JToolTipExample win = new JToolTipExample();
28:         win.launchFrame();
29:     }
30: }
```

6.2.18. JTabbedPane

JTabbedPane은 탭 제어기능을 제공하며, 여러 개의 패널을 사용할 수 있는 인터페이스를 제공합니다. AWT의 CardLayout 기본 동작과 유사하지만 사용이 편리한 이점을 가지고 있습니다. 인자로 JTabbedPane.TOP, JTabbedPane.BOTTOM, JTabbedPane.LEFT, JTabbedPane.RIGHT를 사용하여 탭의 위치를 지정할 수 있습니다.
JTabbedPane의 기본 레이아웃은 JTabbedPane.WRAP_TAB_LAYOUT이지만, 생성자의 두 번째 인자로 JTabbedPane.SCROLL_TAB_LAYOUT를 사용하면 프레임 크기가 작을 때 스크롤바를 나타낼 수 있습니다.



새로운 패널을 추가하려면 다음의 세 가지 addTab()메소드를 이용합니다.

◦ addTab(String title, Component component) : 문자열이 표시된 탭을 작성하고 탭이 선택되면 컴포넌트가 표시됩니다.
◦ addTab(String title, Icon icon, Component component) : 탭의 제목과 함께 아이콘을 표시합니다.
◦ addTab(String title, Icon icon, Component component, String tip) : 탭의 제목과 함께 아이콘과 툴팁을 표시합니다.

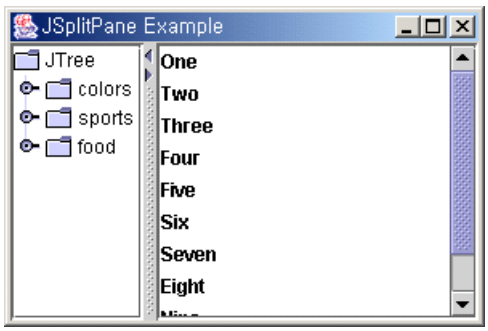
다음 프로그램은 JTabbedPane을 나타내는 예입니다.

exam/java/chapter13/swing/JTabbedPaneExample.java

```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JTabbedPaneExample {
7:     private JFrame f;
8:     private JTabbedPane tabbedPane;
9:     private Container con;
10:
11:    public JTabbedPaneExample() {
12:       f = new JFrame("JTabbedPane Example");
13:       tabbedPane = new JTabbedPane(JTabbedPane.BOTTOM,
    JTabbedPane.SCROLL_TAB_LAYOUT);
14:    }
15:
16:    public void launchFrame() {
17:       tabbedPane.addTab("One", new JButton("하나"));
18:       tabbedPane.addTab("Duke", new ImageIcon("dukeicon.gif" ), new
    JButton("Duke"));
19:       tabbedPane.addTab("Three", new ImageIcon("dukeicon.gif"), new
    JButton("Three"), "Hello");
20:
21:       con = f.getContentPane();
22:       con.add(tabbedPane);
23:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
24:       f.setSize(200, 200);
25:       f.setVisible(true);
26:    }
27:
28:    public static void main(String[] args) {
29:       JTabbedPaneExample win = new JTabbedPaneExample();
30:       win.launchFrame();
31:    }
32: }
```

6.2.19. JSplitPane

JSplitPane은 하나의 컨테이너에 두 개의 컴포넌트를 함께 표시하는 인터페이스를 작성할 수 있게 합니다. JSplitPane 안에는 또 다른 JSplitPane이 올 수 있어서 사실상 여러 개의 영역을 작성할 수 있습니다. 또 영역을 분할할 때는 수평방향 분할과 수직방향 분할을 지정할 수 있습니다. setContinuousLayout 속성은 영역 분리자가 움직일 때 영역을 새로 표시할 것인가를 결정하는데 사용됩니다. 또 dividerLocation을 설정하여 영역 구분선이 표시되는 위치를 지정할 수 있습니다.



다음 프로그램은 JSplitPane을 나타내는 예입니다. 프로그램을 실행시켰을 때에 나타나는 스크롤바는 JSplitPane에 의해 나타난 것이 아니고, JScrollPane에 의해 나타난 것입니다.

exam/java/chapter13/swing/JSplitPaneExample.java

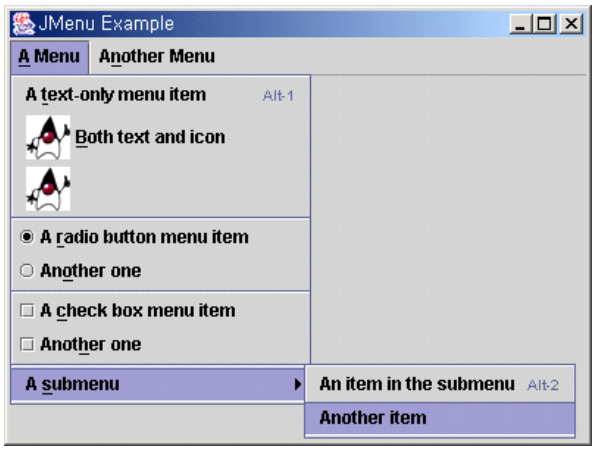
```
1:  package exam.java.chapter13.swing;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class JSplitPaneExample {
7:     private JFrame f;
8:     private JTree tree;
9:     private JList<String> list;
10:    private JScrollPane left, right;
11:    private JSplitPane jsPane;
12:    private String[] listItem = {"One", "Two", "Three", "Four", "Five", "Six",
    "Seven", "Eight", "Nine", "Ten"};
13:    private Container con;
14:
15:    public JSplitPaneExample() {
16:       f = new JFrame("JSplitPane Example");
17:       tree = new JTree();
18:       list = new JList<>(listItem);
19:       left = new JScrollPane(tree);
20:       right = new JScrollPane(list);
21:       jsPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, left, right);
```

```
22:     }
23:
24:     public void launchFrame() {
25:        jsPane.setDividerLocation(0.5);
26:        jsPane.setOneTouchExpandable(true);
27:        con = f.getContentPane();
28:        con.add(jsPane);
29:        f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
30:        f.setSize(300, 200);
31:        f.setVisible(true);
32:     }
33:
34:     public static void main(String[] args) {
35:        JSplitPaneExample win = new JSplitPaneExample();
36:        win.launchFrame();
37:     }
38: }
```

6.3. 메뉴와 도구상자 컴포넌트

6.3.1. 주 메뉴

스윙에서 사용하는 메뉴는 AWT보다 다양한 기능을 가지고 있습니다. 메뉴 클래스(JMenuItem, JCheckBoxMenuItem, JMenu, JMenuBar)는 모두 JComponent에서 상속되며, JMenuBar는 모든 컨테이너에 사용할 수 있습니다. 예를 들어 JApplet 클래스에 setJMenuBar() 메소드를 사용하면 JMenuBar를 연결할 수 있습니다. 뿐만 아니라 JMenuItem에 아이콘을 사용할 수 있는 JRadioButtonMenuItem이 추가되었습니다. 메뉴를 구분하는 선을 넣으려면 AWT에서처럼 addSeparator() 메소드를 이용합니다.



다음 프로그램은 앞의 그림처럼 스윙에서 메뉴를 나타내는 예입니다.

exam/java/chapter13/menu/JMenuExample.java

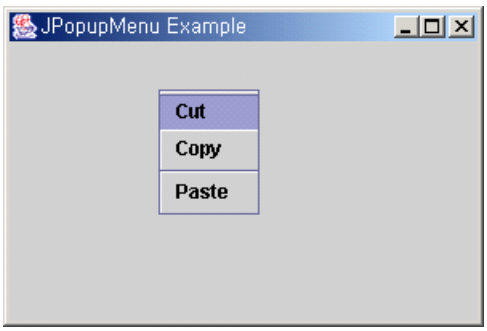
```
1:  package exam.java.chapter13.menu;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:  import javax.swing.*;
6:  public class JMenuExample implements ActionListener, ItemListener {
7:     private JFrame f;
8:     private Container con;
9:     private JMenuBar menuBar;
10:    private JMenu menu, submenu;
11:    private JMenuItem menuItem;
12:    private JCheckBoxMenuItem cbMenuItem;
13:    private JRadioButtonMenuItem rbMenuItem;
14:
15:    public JMenuExample() {
```

```
16:        f = new JFrame("JMenu Example");
17:    }
18:
19:    public void launchFrame() {
20:        //메뉴바 생성
21:        menuBar = new JMenuBar();
22:        f.setJMenuBar(menuBar);
23:
24:        //메뉴 생성
25:        menu = new JMenu("A Menu");
26:        menu.setMnemonic(KeyEvent.VK_A);
27:        menu.addActionListener(this);
28:        menuBar.add(menu);
29:
30:        //JMenuItem 그룹
31:        menuItem = new JMenuItem("A text-only menu item", KeyEvent.VK_T);
32:        menuItem.setAccelerator(
    KeyStroke.getKeyStroke( KeyEvent.VK_1, ActionEvent.ALT_MASK));
33:        menuItem.addActionListener(this);
34:        menu.add(menuItem);
35:
36:        menuItem = new JMenuItem("Both text and icon",
    new ImageIcon("dukeicon.gif"));
37:        menuItem.setMnemonic(KeyEvent.VK_B);
38:        menuItem.addActionListener(this);
39:        menu.add(menuItem);
40:
41:        menuItem = new JMenuItem(new ImageIcon("dukeicon.gif"));
42:        menuItem.setMnemonic(KeyEvent.VK_D);
43:        menuItem.addActionListener(this);
44:        menu.add(menuItem);
45:        //라디오 버튼 모양 메뉴 아이템
46:        menu.addSeparator();
47:        ButtonGroup group = new ButtonGroup();
48:        rbMenuItem = new JRadioButtonMenuItem("A radio button menu item");
49:        rbMenuItem.setSelected(true);
50:        rbMenuItem.setMnemonic(KeyEvent.VK_R);
51:        rbMenuItem.addItemListener(this);
52:        group.add(rbMenuItem);
53:        menu.add(rbMenuItem);
54:
55:        rbMenuItem = new JRadioButtonMenuItem("Another one");
56:        rbMenuItem.setMnemonic(KeyEvent.VK_O);
57:        rbMenuItem.addItemListener(this);
58:        group.add(rbMenuItem);
59:        menu.add(rbMenuItem);
60:
61:        //체크박스 모양 메뉴 아이템
62:        menu.addSeparator();
63:        cbMenuItem = new JCheckBoxMenuItem("A check box menu item");
```

```
64:        cbMenuItem.setMnemonic(KeyEvent.VK_C);
65:        cbMenuItem.addItemListener(this);
66:        menu.add(cbMenuItem);
67:
68:        cbMenuItem = new JCheckBoxMenuItem("Another one");
69:        cbMenuItem.setMnemonic(KeyEvent.VK_H);
70:        cbMenuItem.addItemListener(this);
71:        menu.add(cbMenuItem);
72:
73:        //부메뉴를 갖는 메뉴아이템
74:        menu.addSeparator();
75:        submenu = new JMenu("A submenu");
76:        submenu.setMnemonic(KeyEvent.VK_S);
77:
78:        menuItem = new JMenuItem("An item in the submenu");
79:        menuItem.setAccelerator(
    KeyStroke.getKeyStroke( KeyEvent.VK_2, ActionEvent.ALT_MASK));
80:        submenu.add(menuItem);
81:
82:        menuItem = new JMenuItem("Another item");
83:        submenu.add(menuItem);
84:        menu.add(submenu);
85:
86:        //두번째 메뉴 작성
87:        menu = new JMenu("Another Menu");
88:        menu.setMnemonic(KeyEvent.VK_N);
89:        menuBar.add(menu);
90:
91:        con = f.getContentPane();
92:        con.setLayout(new FlowLayout());
93:        f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
94:        f.setSize(400, 300);
95:        f.setVisible(true);
96:     }
97:
98:     public static void main (String args[]) {
99:        JMenuExample win = new JMenuExample();
100:       win.launchFrame();
101:    }
102:    public void actionPerformed(ActionEvent e) {
103:       System.out.println(e.getSource());
104:    }
105:    public void itemStateChanged(ItemEvent e) {
106:       System.out.println(e.getSource());
107:    }
108:}
```

6.3.2. 팝업 메뉴

JPopupMenu는 모든 컴포넌트에서 사용할 수 있으며 AWT의 PopupMenu와 유사합니다. JPopupMenu에서도 구분선을 넣으려면 addSeparator() 메소드를 이용합니다.



다음 프로그램은 마우스 오른쪽 버튼을 누르면 팝업메뉴가 나타나는 예입니다.

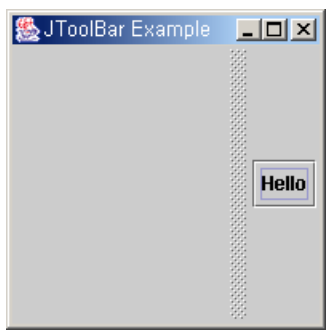
exam/java/chapter13/menu/JPopupMenuExample.java

```
1:  package exam.java.chapter13.menu;
2:
3:  import java.awt.*;
4:  import java.awt.event.*;
5:  import javax.swing.*;
6:
7:  public class JPopupMenuExample {
8:     private JFrame f;
9:     private JPanel panel;
10:    private JPopupMenu popup;
11:    private Container con;
12:
13:    public JPopupMenuExample() {
14:       f = new JFrame("JPopupMenu Example");
15:       panel = new JPanel();
16:       popup = new JPopupMenu();
17:    }
18:
19:    public void launchFrame() {
20:       popup.add(item = new JMenuItem("Cut"));
21:       popup.add(item = new JMenuItem("Copy"));
22:       popup.addSeparator();
23:       popup.add(item = new JMenuItem("Paste"));
24:       popup.setInvoker(panel);
25:
26:       panel.addMouseListener (new MouseAdapter() {
27:          public void mousePressed (MouseEvent e) {
28:             if (e.isPopupTrigger()) {
```

```
29:                 popup.show (e.getComponent(), e.getX(), e.getY());
30:             }
31:          }
32:          public void mouseReleased (MouseEvent e) {
33:             if (e.isPopupTrigger()) {
34:                popup.show (e.getComponent(), e.getX(), e.getY());
35:             }
36:          }
37:       });
38:       con = f.getContentPane();
39:       con.setLayout(new BorderLayout());
40:       con.add(panel);
41:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
42:       f.setSize(300, 200);
43:       f.setVisible(true);
44:    }
45:
46:    public static void main (String args[]) {
47:       JPopupMenuExample win = new JPopupMenuExample();
48:       win.launchFrame();
49:    }
50: }
```

6.3.3. JToolBar

JToolBar는 툴바의 모양을 제공하는 클래스로 좌.우.상.하에 위치시킬 수 있으며 기본값(default)으로 별도의 창에 분리시킬 수도 있습니다. 별도의 창에 분리되는 기능을 없애려면 setFloatable(false)를 이용합니다.



다음 프로그램은 앞의 그림처럼 툴바를 나타내는 예입니다.

exam/java/chapter13/menu/JToolBarExample.java

```
1:  package exam.java.chapter13.menu;
2:
```

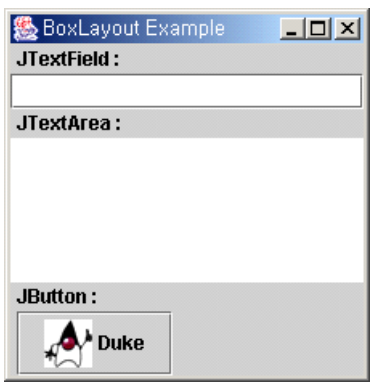
```
3:  import javax.swing.*;
4:  import java.awt.*;
5:
6:  public class JToolBarExample {
7:     private JFrame f;
8:     private JPanel panel;
9:     private JToolBar toolbar;
10:    private JButton myButton;
11:    private Container con;
12:
13:    public JToolBarExample() {
14:       f = new JFrame("JFrame Example");
15:       panel = new JPanel();
16:       toolbar = new JToolBar();
17:       myButton = new JButton("Hello");
18:    }
19:
20:    public void launchFrame() {
21:       panel.setLayout(new BorderLayout());
22:       toolbar.add(myButton);
23:       panel.add(toolbar, BorderLayout.EAST);
24:
25:       con = f.getContentPane();
26:       con.setLayout(new BorderLayout());
27:       con.add(panel);
28:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
29:       f.setSize(200, 200);
30:       f.setVisible(true);
31:    }
32:
33:    public static void main (String args[]) {
34:       JToolBarExample win = new JToolBarExample();
35:       win.launchFrame();
36:    }
37: }
```

6.4. 스윙의 레이아웃 관리자

스윙에는 ScrollPaneLayout, ViewportLayout, BoxLayout, OverlayLayout의 4가지 주요한 레이아웃 관리자가 있는데, 이들의 특징은 다음과 같습니다.

6.4.1. BoxLayout

X축이나 Y축에 따라 컴포넌트를 배치할 수 있습니다. 예를 들면, Y축 BoxLayout을 사용하면 컴포넌트는 위에서부터 아래로 정렬되고, X축 BoxLayout을 사용하면 왼쪽에서 오른쪽으로 컴포넌트가 정렬됩니다.
GridLayout과는 달리 컴포넌트가 주어진 축을 기준으로 각기 다른 영역을 차지하게 되는데, Y축을 기준으로 한 BoxLayout에 사용된 JTextField는 작은 영역을 차지하게 됩니다. 좌우로 배치되는 경우는 너비가 가장 큰 컴포넌트에 맞춰지고, 상하로 배치되는 경우는 폭이 가장 큰 것에 맞추어져 표시됩니다.
박스레이아웃 생성자는 두 개의 파라미터를 사용하는데, 첫 번째 파라미터는 컨테이너를 지시하고 두 번째 파라미터는 기준이 되는 축을 표시합니다.



다음 프로그램은 BoxLayout 예입니다.

exam/java/chapter13/layout/BoxLayoutExample.java

```
1:  package exam.java.chapter13.layout;
2:
3:  import java.awt.*;
4:  import javax.swing.*;
5:
6:  public class BoxLayoutExample {
7:     private JFrame f;
8:     private JPanel panel;
```

```
9:     private JTextField textField;
10:    private JTextArea textArea;
11:    private JButton button;
12:    private Container con;
13:
14:    public BoxLayoutExample() {
15:       f = new JFrame("BoxLayout Example");
16:       panel = new JPanel();
17:       textField = new JTextField();
18:       textArea = new JTextArea(5, 20);
19:       button = new JButton("Duke", new ImageIcon("dukeicon.gif"));
20:    }
21:
22:    public void launchFrame() {
23:       panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
24:       panel.add(new JLabel("JTextField : "));
25:       panel.add(textField);
26:       panel.add(new JLabel("JTextArea : "));
27:       panel.add(textArea);
28:       panel.add(new JLabel("JButton : "));
29:       panel.add(button);
30:
31:       con = f.getContentPane();
32:       con.add(panel);
33:       f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE );
34:       f.pack();
35:       f.setVisible(true);
36:    }
37:    public static void main(String[] args) {
38:       BoxLayoutExample win = new BoxLayoutExample();
39:       win.launchFrame();
40:    }
41: }
```

Box클래스

Box는 기본 레이아웃 매니저가 BoxLayout인 클래스로 BoxLayout 컨테이너와 더불어 컴포넌트 배열에 유용한 static 메소드를 가지고 있습니다.

다음은 Box클래스의 주요 메소드를 나타낸 것입니다.

- createVerticalStrut(int) : 간격을 위해 고정 높이 컴포넌트를 반환합니다.
- createHorizontalStrut(int) : 간격을 위해 고정 폭 컴포넌트를 반환합니다.
- createVerticalGlue() : 컴포넌트 사이의 초과된 공간을 흡수하기 위해 확장된 높이 컴포넌트를 반환합니다.
- createHorizontalGlue() : 컴포넌트 사이의 초과된 공간을 흡수하기 위해 확장된 폭 컴포넌트를 반환합니다.

◦ createGlue() : 높이는 Y축으로 확장되고 폭은 X축으로 확장된 컴포넌트를 반환합니다.
◦ createRigidArea(Dimension) : 고정 높이 고정 폭을 반환합니다.

6.4.2. ScrollPaneLayout

ScrollPaneLayout은 JScrollPane에서 사용하는 레이아웃 관리자입니다. 직접 만들지 않고 자동으로 생성됩니다. 이 레이아웃은 JScrollPane에 9개의 다른 영역을 지정합니다.

◦ 하나의 JViewport : 내용의 중심에 지정합니다.
◦ 두 개의 JScrollBar 객체 : 수평과 수직 스크롤바에 지정합니다.
◦ 두 개의 JViewport 객체 : 행과 열에 지정합니다.
◦ 네 개의 Component 객체 : 각 모서리에 지정합니다.

JScrollPane은 모서리를 LOWER_LEFT_CORNER, LOWER_RIGHT_CORNER, UPPER_LEFT_ CORNER, UPPER_RIGHT_CORNER로 지정합니다.
JViewport는 자체 컴포넌트를 가질 수 있는 컨테이너로 매우 유연한 정렬기능을 갖고, 자신만의 레이아웃 관리자인 ViewportLayout을 사용합니다.

6.4.3. ViewportLayout

ViewportLayout은 JViewport가 사용하는 레이아웃 매니저로 사용자가 직접 레이아웃을 사용하지 않고 JViewPort 객체와 자동으로 연결되어 JViewPort 속성에 따라 내부 컴포넌트의 위치가 결정됩니다.