23. 스윙컴포넌트 I

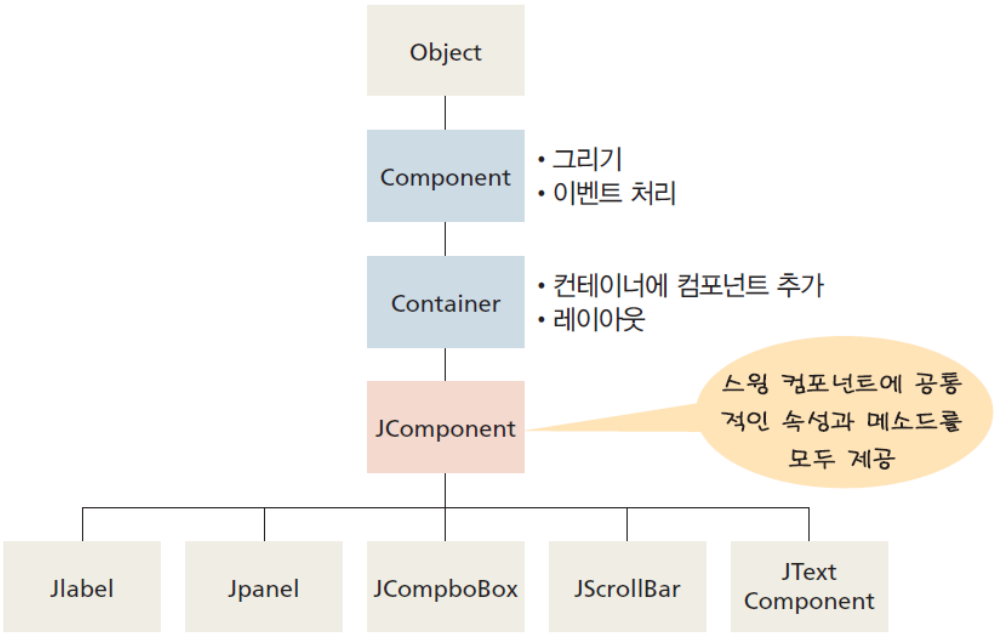
학습목표

- 이번장에서는 기초적인 스윙 컴포넌트들을 학습하여 본다. 텍스트에 관련된 컴포넌트, 스크롤페인, 체크박스, 라디오 버튼 등이 다루어진다

1. JComponent 클래스

거의 모든 컴포넌트가 JComponent를 상속받는다. JComponent 클래스에는 스윙 컴포넌트들이 공통적으로 가져야 하는 속성과 메소드가 포함되어 있다.

JComponent 클래스는 Component와 Container 클래스를 상속받는다. Component 클래스는 화면 그리기와 이벤트 처리 기능을 제공한다. Container 클래스는 주로 컨테이너 클래스에 컴포넌트를 추가하거나 제거하는 기능과 레이아웃 기능을 제공한다.



컴포넌트 외관 변경

메소드	설명
`void setBorder(Border)` `Border getBorder()`	컴포넌트의 경계를 설정한다.
`void setForeground(Color)` `void setBackground(Color)`	컴포넌트의 전경색과 배경색을 설정한다.
`Color getForeground()` `Color getBackground()`	컴포넌트의 전경색과 배경색을 얻는다.
`void setFont(Font)` `Font getFont()`	컴포넌트의 폰트를 설정한다.
`void setCursor(Cursor)` `Cursor getCursor()`	컴포넌트의 커서를 설정한다.

컴포넌트의 상태 변경

메소드	설명
void setToolTipText(String)	툴팁에 표시되는 텍스트를 설정한다.
void setName(String) String getName()	컴포넌트의 이름을 설정한다.
boolean isShowing()	컴포넌트가 화면에 표시되고 있으면 true 반환
void setEnabled(boolean) boolean isEnabled()	컴포넌트를 활성화하거나 비활성화한다.
void setVisible(boolean) boolean isVisible()	컴포넌트를 화면에 표시한다.

공통 이벤트 처리

메소드	설명
void addMouseListener(MouseListener) void removeMouseListener(MouseListener)	마우스 리스너를 추가하거나 제거한다.
...	...

컴포넌트 그리기

메소드	설명
void repaint() void repaint(int, int, int, int)	컴포넌트의 일부나 전체를 다시 그리라고 요청한다.
void revalidate()	컨테이너 안의 컴포넌트를 다시 배치하라고 요청한다.
void paintComponent(Graphics)	컴포넌트를 그린다. 만약 사용자 맞춤형 컴포넌트에서는 이 메소드를 재정의하여서 그림을 그린다.

컨테이너에 추가/삭제

메소드	설명
Component add(Component)	컴포넌트를 컨테이너에 추가
void remove(Component) void removeAll()	컨테이너에서 컴포넌트를 삭제
Container getParent()	컴포넌트의 컨테이너를 반환한다.

레이아웃

메소드	설명
int getWidth(), int getHeight()	필셀 단위의 컴포넌트 크기 반환
Dimension getSize()	필셀 단위의 컴포넌트 크기 반환
int getX(), int getY()	부모 컨테이너 안에서의 컴포넌트의 상대적인 위치 반환
Point getLocation()	부모 컨테이너 안에서의 상대적인 위치 반환
Point getLocationOnScreen()	화면에서의 절대 위치 반환
void setLocation(int, int) void setLocation(Point)	부모 컨테이너 안에서의 컴포넌트의 좌표 지정, 레이아웃 관리자가 없는 상태에서만 유효하다.
void setBounds(int, int, int, int)	부모 컨테이너 안에서의 크기와 위치 설정, 레이아웃 관리자가 없는 상태에서만 유효하다.

크기와 위치 정보 얻기

메소드	설명
void setPreferredSize(Dimension) void setMaximumSize(Dimension) void setMinimumSize(Dimension)	컴포넌트의 크기를 설정한다.
void setAlignmentX(float) void setAlignmentY(float)	컨테이너 안에서 컴포넌트들의 정렬을 지정한다.
void setLayout(LayoutManager) LayoutManager getLayout()	배치 관리자를 설정한다.

2. 컴포넌트에 이미지 표시

레이블과 버튼에는 텍스트뿐만 아니라 이미지도 표시할 수 있다. 레이블과 버튼에 이미지를 추가로 표시한 것이다.

레이블과 버튼에 이미지를 표시하려면 먼저 ImageIcon 인스턴스를 생성하여야 한다. ImageIcon은 JPEG, GIF, PNG 이미지 파일을 읽을 수 있다.

```
ImageIcon image = new ImageIcon("image.gif");
```

레이블에 이미지를 지정하는 가장 간단한 방법은 setIcon() 메소드를 사용하는 것이다.

```
JLabel label = new JLabel("이미지 레이블");
label.setIcon(image);
```

레이블에 이미지가 설정되면 레이블의 텍스트는 오른쪽에 표시된다. 버튼에 이미지를 추가하는 과정도 레이블과 아주 유사하다.

(예제)

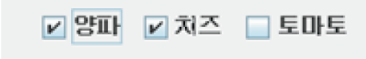
ImageLabelTest.java

```
import javax.swing.*;

import java.awt.*;
import java.awt.event.*;

public class ImageLabelTest extends JFrame implements ActionListener {
        private JPanel panel;
        private JLabel label;
        private JButton button;

        public ImageLabelTest() {
                setTitle("이미지 레이블");
                setSize(300, 250);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                panel = new JPanel();
                label = new JLabel("이미지를 보려면 아래 버튼을 누르세요");

                button = new JButton("이미지 레이블");
                ImageIcon icon = new ImageIcon("icon.gif");
                button.setIcon(icon);
                button.addActionListener(this);
                panel.add(label);
                panel.add(button);

                add(panel);
                setVisible(true);
        }
        public static void main(String[] args) {
                ImageLabelTest t = new ImageLabelTest();
        }
        public void actionPerformed(ActionEvent e) {
                ImageIcon dog = new ImageIcon("dog.gif"); // 이미지 아이콘 객체 생성
                label.setIcon(dog);                       // 레이블에 이미지 추가
                label.setText(null);
        }
}
```

버튼에 이미지를 설정한다

3. 체크 박스

체크 박스(check box)란 체크된 상태와 체크되지 않은 상태 중에서 하나를 선택할 수 있는 컨트롤이다. 체크 박스는 흔히 사용자로 하여금 YES와 NO 중에서 하나를 선택하게 하는데 사용된다. 생성하기 위해서 JCheckBox 클래스를 사용한다.



1) 생성자와 메소드

	이름	설명
생성자	JCheckBox()	레이블이 없는 Checkbox를 생성한다.
	JCheckbox(String label)	지정된 레이블의 Checkbox를 생성한다.
	JCheckbox(String label, boolean selected)	지정된 상태와 레이블을 가지는 Checkbox를 생성한다.
메소드	String getText()	체크 박스에 표시되는 텍스트를 가져온다.
	Boolean isSelected()	만약 체크 박스가 선택되었으면 true를 반환한다.
	void setSelected(boolean value)	매개변수가 true이면 체크 박스를 체크 상태로 만든다.
	void setText(String text)	체크 박스 텍스트를 설정한다.

체크 박스는 다음과 같이 생성하면 된다

```
onion = new JCheckBox("양파");
```

만약 생성자에서 초기 상태를 주지 않으면 체크되지 않은 체크 박스가 생성된다. 체크된 상태를 원하면 다음과 같이 두 번째 매개변수를 주면 된다

```
onion = new JCheckBox("양파", true);
```

체크 박스를 강제적으로 true나 false로 설정하려면 setSelected()를 사용한다.

```
onion.setSelected(true);
```

2) 이벤트 처리

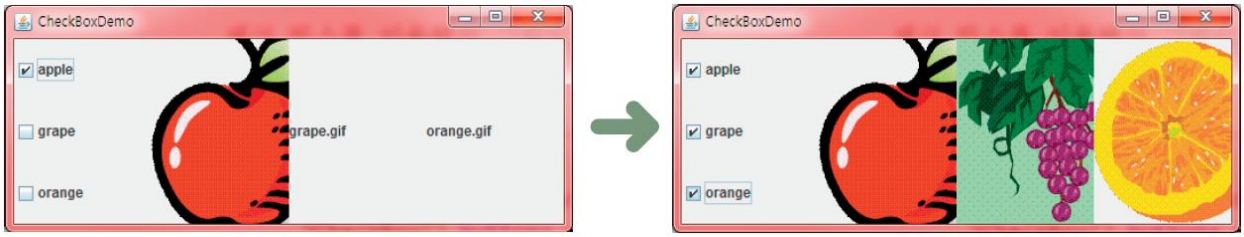
체크 박스가 선택되었을 경우의 이벤트를 처리하고 싶으면 Item 이벤트를 처리하여야 한다.

```
class MyItemEvent implements ItemListener {
        …
        onion.addItemListener(this);
        …
        public void itemStateChange(ItemEvent e) {
                if(onion.isSelected() == true) msg = "OK";
                else msg = "OFF";
                System.out.println(msg);
        }
```

```
}
```

(예제)

체크 박스를 이용해서 이미지 출력



CheckBoxTest.java

```
import java.awt.GridLayout;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.ImageIcon;
import javax.swing.JCheckBox;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class CheckBoxTest extends JPanel implements ItemListener {
	JCheckBox[] buttons = new JCheckBox[3];
	String[] fruits = { "apple", "grape", "orange" };
	JLabel[] pictureLabel = new JLabel[3];
	ImageIcon[] icon = new ImageIcon[3];

	public CheckBoxTest() {
		super(new GridLayout(0, 4));
		// 체크 박스 생성, 체크 박스에 아이템 리스너를 등록
		for (int i = 0; i < 3; i++) {
			buttons[i] = new JCheckBox(fruits[i]);
			buttons[i].addItemListener(this);
			pictureLabel[i] = new JLabel(fruits[i] + ".gif");
			icon[i] = new ImageIcon(fruits[i] + ".gif");
		}
		// 체크 박스들을 하나의 컬럼으로 묶는다.
		JPanel checkPanel = new JPanel(new GridLayout(0, 1));
		for (int i = 0; i < 3; i++){
			checkPanel.add(buttons[i]);
```

```
                }
                add(checkPanel);
                add(pictureLabel[0]);
                add(pictureLabel[1]);
                add(pictureLabel[2]);
        }
        /** 체크 박스의 아이템 이벤트를 처리한다. */
        public void itemStateChanged(ItemEvent e) {
                ImageIcon image = null;
                // 선택된 체크 박스를 얻는다.
                Object source = e.getItemSelectable();
                for (int i = 0; i < 3; i++) {
                        if (source == buttons[i]) {
                                // 체크가 해제되었으면
                                if (e.getStateChange() == ItemEvent.DESELECTED)
                                        pictureLabel[i].setIcon(null);
                                else
                                        pictureLabel[i].setIcon(icon[i]);
                        }
                }
        }
        public static void main(String[] args) {
                JFrame frame = new JFrame("CheckBoxDemo");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                JComponent newContentPane = new CheckBoxTest();
                newContentPane.setOpaque(true);
                frame.setContentPane(newContentPane);
                frame.setSize(500, 200);
                frame.setVisible(true);
        }
}
```

이미지 파일명(apple.gif, grape.gif, orange.gif)

4. 라디오 버튼

라디오 버튼(radio button)은 하나의 그룹 안에서는 한 개의 버튼만 선택할 수 있다. 만약 하나의 라디오 버튼을 클릭하면 다른 버튼은 자동적으로 선택이 해제된다. 라디오 버튼을 생성하기 위해서는 두 개의 클래스를 이용하는데 JRadioButton으로 라디오 버튼을 생성하고 ButtonGroup으로 버튼들을 그룹핑하는데 사용한다.

- 생성자와 메소드

	이름	설명
생성자	JRadioButton(String text)	라디오 버튼을 생성한다.
	JRadioButton(String text, boolean selected)	초기 상태를 가지는 라디오 버튼을 생성한다.

첫 번째 생성자는 선택되지 않은 라디오 버튼을 생성한다.

```
JRadioButton radio1 = new JRadioButton("Small Size");
```

두 번째 생성자를 사용하면 선택 상태를 지정할 수 있다.

```
JRadioButton radio1 = new JRadioButton("Small Size", true);
```

ButtonGroup을 이용하여 그룹핑한다.

```
ButtonGroup group = new ButtonGroup();
```
<- ButtonGroup 객체를 생성한다.

```
group.add(radio1);
group.add(radio2);
group.add(radio3);
```
<- 라디오 버튼들을 ButtonGroup 객체에 추가한다.

- 이벤트 처리

만약 라디오 버튼이 눌려지는 순간, 어떤 작업을 실행하고 싶다면 액션 이벤트를 처리하여야 한다.

```
private class RadioButtonListener implements ActionListener{
        public void actionPerformed(ActionEvent e) {
                if(e.getsource() == radio1){
                        ...
                }
        }
}
```

라디오 버튼이 눌려졌는지를 검사하고 싶다면 다음과 같이 isSelected()를 사용한다.

```
if(radio1.isSelected()) {
        // 라디오버튼이 선택되었으면 실행되는 코드를 여기에 놓는다.
}
```

만약 코드로 선택하려면 doClick()을 사용한다.

(예제)

RadioButtonTest.java

```
import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.JOptionPane;
import java.awt.event.*;
import java.awt.*;

class CoffeeOrder extends JFrame implements ActionListener {
        private JRadioButton small, medium, large;
        private JLabel text;
        private JPanel topPanel, sizePanel, resultPanel;

        public CoffeeOrder() {
                setTitle("라디오 버튼 테스트");
                setSize(300, 150);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                topPanel = new JPanel();
                JLabel label = new JLabel("어떤 크기의 커피를 주문하시겠습니까?");
                topPanel.add(label);
                add(topPanel, BorderLayout.NORTH);
                // 라디오 버튼 생성
                sizePanel = new JPanel();
                small = new JRadioButton("Small Size");
                medium = new JRadioButton("Medium Size");
                large = new JRadioButton("Large Size");
                // 버튼 그룹 생성하고 라디오 버튼 추가
                ButtonGroup size = new ButtonGroup();
                size.add(small);
                size.add(medium);
                size.add(large);
                // 버튼에 이벤트 리스너 등록
                small.addActionListener(this);
                medium.addActionListener(this);
                large.addActionListener(this);
                // 버튼을 패널에 추가하고 패널을 프레임에 추가
                sizePanel.add(small);
                sizePanel.add(medium);
                sizePanel.add(large);
                add(sizePanel, BorderLayout.CENTER);
```

```
                resultPanel = new JPanel();
                text = new JLabel("크기가 선택되지 않았습니다.");
                text.setForeground(Color.red);
                resultPanel.add(text);
                add(resultPanel, BorderLayout.SOUTH);
                setVisible(true);
        }
        public void actionPerformed(ActionEvent e) {
                if (e.getSource() == small) {
                        text.setText("Small 크기가 선택되었습니다.");
                }
                if (e.getSource() == medium) {
                        text.setText("Medium 크기가 선택되었습니다.");
                }
                if (e.getSource() == large) {
                        text.setText("Large 크기가 선택되었습니다.");
                }
        }
}
public class RadioButtonTest extends JFrame {
        public static void main(String[] args) {
                new CoffeeOrder();
        }
}
```

• 경계 만들기

경계(border)란 시각적으로 컴포넌트들을 그룹핑할 때 사용하는 장식적인 요소이다. 일반적으로 체크박스와 라디오 버튼을 그룹핑할 때 사용하고 경계 객체인 Border를 생성하기 위해서는 다음 중에서 하나를 선택하여 사용하면 된다. 아래의 메소드들은 모두 BorderFactory 클래스의 정적 메소드이다. 경계가 생성되면 이 경계 객체를 패널의 setBorder() 메소드를 이용하여 패널에 부착하면 된다.

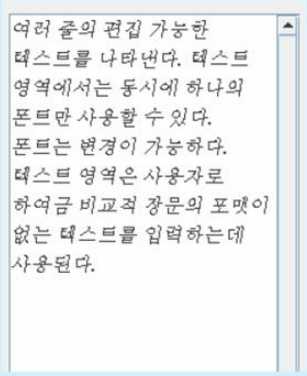
메소드	설명
`Border createLineBorder(Color)` `Border createLineBorder(Color, int)`	직선으로 된 경계를 생성한다. 첫 번째 매개변수는 선의 색상이고 두 번째 매개변수는 선의 폭이다(단위는 픽셀).
`TitledBorder createTitledBorder(String)`	제목이 붙여진 경계를 생성한다. 문자열 매개변수가 제목을 나타낸다.

앞의 라디오버튼예제에 경계를 생성하려면 Border 객체를 생성하고 sizePanel에 경계를 설정하여야 한다.

```
Border border = BorderFactory.createTitledBorder("크기");
sizePanel.setBorder(border);
```

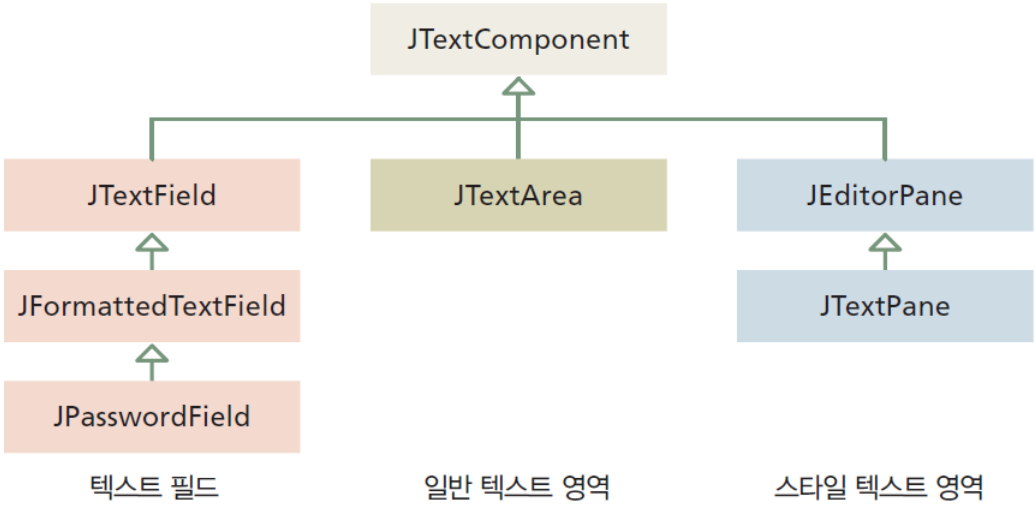
5. 텍스트 컴포넌트

텍스트 컴포넌트들은 텍스트를 표시하며 사용자가 텍스트를 편집하거나 입력할 수 있게 한다. 아이디 입력과 같은 한줄로 된 텍스트를 입력하는 데는 텍스트 필드(text field)를 사용한다. 메모장과 같이 여러 줄을 입력받으려면 단순 텍스트 영역(plain text area)을 사용하면 된다. 단순 텍스트 영역은 하나의 폰트만 사용할 수 있다. 만약 여러 개의 폰트를 사용하고 중간에 이미지나 사운드가 들어 있는 복잡한 문서라면 스타일 텍스트 영역(styled text area)을 이용하면 된다.

텍스트 컴포넌트	카테고리	설명
JTextField JPasswordField JFormattedTextField	텍스트 필드	 한 줄로 된 텍스트를 표시하고 편집할 수있다. 일반적으로 사용자로부터 적은 양의 텍스트를 입력받을 때 사용한다.
JTextArea	단순 텍스트 영역	 여러 줄의 편집 가능한 텍스트를 나타낸 다. 텍스트 영역에서는 동시에 하나의 폰트만 사용할 수 있다. 폰트는 변경이 가능하다. 텍스트 영역은 사용자로 하여금 비교적 장문의 포맷이 없는 텍스트를 입력하는 데 사용된다.
JEditorPaneJTextPane	스타일 텍스트 영역	 스타일 텍스트 영역은 두 개 이상의 폰트를 동시에 사용할 수 있다. 스타일 텍스트 영역은 이미지 컴포넌트도 내장할 수 있다. 스타일 텍스트 영역은 고급 수요에 적합한 강력한 다용도 컴포넌트이다. 또 다른 컴포넌트보다 맞춤화 기능이 강력하다. 하지만 스타일 텍스트 영역은 초기에 설정해주어야 하는 것들이 많다.

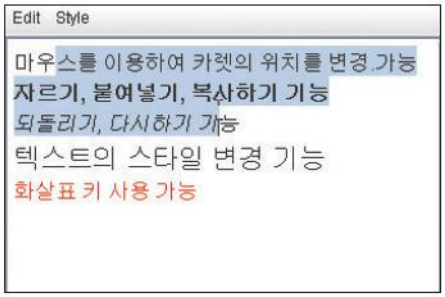
(1) JtextComponent 클래스

모든 텍스트 컴포넌트들은 JtextComponent라고 하는 동일한 수퍼 클래스로부터 상속된다. JtextComponent는 텍스트 조작에 필요한 공통적인 기능들을 포함하고 있다. 이 클래스는 후손 클래스들을 위하여 데이터와 뷰를 분리하는 MVC(Model-View-Controller)구조를 제공한다.



텍스트 컴포넌트들의 상속 계층도

- 컴포넌트가 가진 데이터는 document라고 하는 이름의 인스턴스에 저장된다. document는 MVC에서 모델(Model)에 해당한다.
- "오려두기", "복사하기", "붙이기", "삽입하기"와 같은 기본적인 텍스트 편집 기능을 제공한다. 이들 기능들은 메뉴와 버튼들과 연결될 수 있다.
- 단축키 기능을 제공한다.
- 무제한의 되돌림 기능(undo)과 다시하기 기능(redo)을 제공한다.
- 교체가능한 카렛과 카렛 변경 리스터와 내비게이션 필터를 제공한다.
- 문서 필터를 제공한다.



메뉴나 버튼을 만들어서 JtextComponent가 제공하는 메소드를 호출해 주어야 한다. 비록 오른쪽 화면이 JtextPane의 인스턴스를 사용하고 있지만 모든 기능은 JtextComponent 클래스로부터 상속받은 것이다.

- 텍스트 입출력

메소드	설명
`void read(Reader r, Object obj)` `void write(Writer w)`	텍스트를 읽거나 쓴다.
`String getText()`	컴포넌트에 입력된 Text를 반환한다.
`setText(String str)`	Text를 str으로 설정한다.

- 속성 설정 기능

메소드	설명
`void setEditable(boolean)` `boolean isEditable()`	편집 가능 여부를 설정한다.
`void setDragEnabled(boolean)` `boolean getDragEnabled()`	드래그 가능 여부를 설정한다.

- 텍스트 편집 기능

메소드	설명
`String getSelectedText()`	현재 선택된 텍스트를 반환한다.
`void selectAll()` `void select(int start, int end)`	모든 텍스트를 선택하거나 start와 end사이의 텍스트를 선택한다.

`void setSelectionStart(int start) void setSelectionEnd(int end)`
`int getSelectionStart()`
`int getSelectionEnd()`
	현재 선택된 위치를 설정하거나 반환한다.

`void cut()`
`void copy()`
`void paste()`
	시스템의 클립보드에 텍스트를 자르거나 복사하거나 붙여넣는다.

`void replaceSelection(String s)`	선택된 텍스트를 대치한다.

- 카렛 조작 기능 (카렛(caret)이란 입력의 위치를 표시하는 커서이다.)

메소드	설명

`void setCaretPosition(int p)`
`void moveCaretPosition(int p)`
`int getCaretPosition()`
	문서 내의 카렛의 위치를 설정하거나 반환한다.

이들 메소드들은 텍스트 필드(JTextField)나 텍스트 영역(JTextArea)을 비롯한 모든 후손 클래스들이 사용할 수 있음을 명심하여야 한다.

6. 텍스트 필드(Text Field)

텍스트 필드(Text Field)는 입력이 가능한 한 줄의 필드를 만드는데 사용한다. 편집을 가능하게 할 수도 있고 불가능하게 할 수도 있다. 편집을 불가능하게 설정하면 읽기 전용이 된다. 텍스트 필드 안에서 문자열을 선택할 수 있고 선택한 문자열을 복사하여 붙이는 것도 가능하다

JTextField는 두 개의 자식 클래스를 가지는데 JpasswordField는 패스워드 입력처럼 사용자가 입력하는 문자를 보여주지 않는 컴포넌트이다. JformattedTextField는 사용자가 입력할 수 있는 문자 집합을 제한한다.

- 생성자

생성자	설명
JTextField()	TextField를 생성한다.
JTextField(int columns)	지정된 칸수를 가지고 있는 TextField를 생성한다.
JTextField(String text)	지정된 문자열로 초기화된 TextField를 생성한다.
JTextField(String text, int columns)	지정된 칸 수를 가지고 있고, 지정된 문자열로 초기화된 TextField을 생성한다.

텍스트 필드 생성

```
JTextField textfield = new JTextField(30);                  // 30자 크기의 텍스트 필드를 만든다.
JTextField textfield = new JTextField("Initial String");    // 초기화 문자열
```

텍스트 필드의 텍스트 읽어오기(getText())

```
System.out.println(textField.getText());
```

텍스트 필드에 텍스트 입력하기(setText())

```
textField.setText("Seoul");
```

텍스트 필드의 텍스트 제거하기

```
textField.setText("");
```

텍스트 필드의 텍스트를 모두 선택하기(selectAll())

```
textField.selectAll();
```

키보드 포커스를 텍스트 필드에 넣기

```
textField.requestFocus();
```

- 메소드

메소드	설명
void setText(String text)	지정된 문자열을 텍스트 필드에 쓴다.
String getText()	텍스트 필드에 입력된 문자열을 반환한다.
void setEditable(boolean) boolean isEditable()	사용자가 텍스트를 입력할 수 있는지 없는지를 설정하고 반환한다.
void setColumns(int); int getColumns()	텍스트 필드의 컬럼의 크기를 설정하고 반환한다.
void setHorizontalAlignment(int); int getHorizontalAlignment()	텍스트의 정렬 방법을 설정한다. JTextField.LEADING, JTextField.CENTER, JTextField.TRAILING 등으로 설정할 수 있다.
void addActionListener(ActionListener) void removeActionListener(ActionListener)	액션 리스너를 추가하고 제거한다.
void selectAll()	텍스트 필드의 모든 문자를 선택한다.

• 이벤트 처리

텍스트 필드에 입력한 후에 [Enter]키를 누르면 액션 이벤트가 발생한다.

(예제)

사용자로부터 정수를 입력받은 후에 정수의 제곱을 구하여 결과를 출력 전용의 텍스트 필드를 이용하여 나타낸다.

MyTextFieldTest.java

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

class MyTextField extends JFrame {
	private JButton button;
	private JTextField text, result;

	public MyTextField() {
		setSize(300, 130);
		setTitle("제곱 계산하기");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		ButtonListener listener = new ButtonListener(); // 리스너 객체 생성

		JPanel panel = new JPanel();
		panel.add(new JLabel("숫자 입력: ")); // 레이블 생성
		text = new JTextField(15); // 컬럼수가 15인 텍스트 필드 생성
		text.addActionListener(listener); // 텍스트 필드에 리스너 연결
		panel.add(text);

		panel.add(new JLabel("제곱한 값: "));
		result = new JTextField(15); // 결과를 나타낼 텍스트 필드
		result.setEditable(false); // 편집 불가 설정
		panel.add(result);

		button = new JButton("OK");
		button.addActionListener(listener); // 버튼에 리스너 등록
		panel.add(button);
		add(panel);
		setVisible(true);
	}
	// 텍스트 필드와 버튼의 액션 이벤트 처리
	private class ButtonListener implements ActionListener {
```

```
                public void actionPerformed(ActionEvent e) {
                        if (e.getSource() == button || e.getSource() == text) {
                                String name = text.getText();
                                int value = Integer.parseInt(name);
                                result.setText(" " + value * value);
                                text.requestFocus();
                        }
                }
        }
}
public class MyTextFieldTest extends JFrame {
        public static void main(String[] args) {
                new MyTextField();
        }
}
```

(1) 패스워드 필드

패스워드 필드는 암호를 입력받을 때 사용한다. 패스워드 필드에 사용자가 암호를 입력하면 글자들이 모두 *문자로 표시된다. 패스워드 필드의 생성자 및 메소드는 대부분 텍스트 필드와 같다.

메소드	설명
`void setEchoChar(char c)`	화면에 대신 보여주는 문자를 지정한다.
`char getEchoChar()`	화면에 대신 보여주는 문자를 반환한다.
`char[] getPassword()`	사용자가 입력한 암호를 문자 배열 형태로 반환한다.

(2) 형식화 텍스트 필드(날짜, 숫자, 금액, 전화 번호 형식에 맞는 데이터를 입력받을때 사용한다.)

JformattedTextField 클래스에 의하여 지원되며 JTextField에다가 포매터(formatter)와 값(value)이라는 속성을 추가한다. 포매터는 텍스트 필드의 값을 텍스트로 변환하여서 표시하고 반대로 텍스트를 값으로 변환하는 역할을 한다. 많이 사용되는 포매터로는 날짜를 입력할 수 있는 DateFormatter, 다양한 형식의 숫자를 입력하는 NumberFormatter, ###-###와 같은 전화번호를 입력할 수 있는 MaskFormatter등이 있다.

```
❶ dateField = new JFormattedField(new DateFormatter());
❷ dateField.setValue(new Date());
❸ dateField.addpropertyChangeListener("value", this);
```

①은 JFormattedField의 객체를 생성한다. 생성자에는 날짜 변환을 담당하는 포매터 객체가 매개변수 로 주어져 있다.

②는 dateField의 "값(value)" 속성을 현재 날짜로 설정한다.

③은 "값(value)" 속성이 변경되면 현재의 객체가 리스너로 호출되도록 등록한다. 형식화 텍스트 필드 의 값이 변경되면 현재 객체의 propertyChange() 메소드가 호출된다.

값속성이 변경되면 현재의 객체가 리스너로 호출되도록 등록한다. 형식화 텍스트 필드의 값이 변경되면 현재 객체의 PropertyChange()메소드가 호출된다.

(예제)
간단한 예제로 숫자와 날짜를 입력할 수 있도록 형식화 텍스트 필드를 가진 애플리케이션을 작성하여 보자.

FormattedFieldTest.java

```
import java.awt.*;
import java.awt.event.*;
import java.text.*;
import java.util.Date;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.*;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeEvent;

class FormattedField extends JFrame implements PropertyChangeListener {
	// 필드 정의
	private double amount = 100000;
	private Date date;
	// 필드의 이름을 표시하는 레이블
	private JLabel amountLabel;
	private JLabel dateLabel;
	// 데이터 입력을 위한 포매터
	private JFormattedTextField amountField;
	private JFormattedTextField dateField;

	public FormattedField() {
		super();
		setSize(300, 100);
		// 레이블을 생성한다.
		amountLabel = new JLabel("금액: ");
		dateLabel = new JLabel("날짜: ");
		// 숫자를 입력받을 수 있는 텍스트 필드를 생성하고 속성을 설정한다.
		amountField = new JFormattedTextField(new NumberFormatter());
		amountField.setValue(new Integer(100));
		amountField.setColumns(10);
		amountField.addPropertyChangeListener("value", this);
		// 날짜를 입력받을 수 있는 텍스트 필드를 생성하고 속성을 설정한다.
		dateField = new JFormattedTextField(new DateFormatter());
```

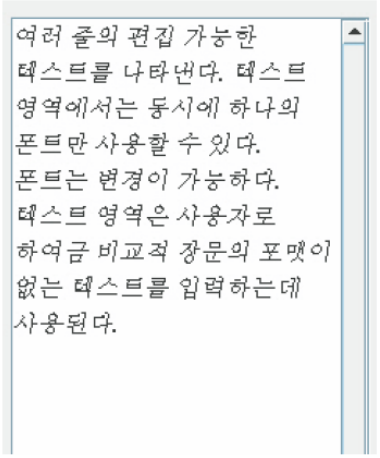
```
                dateField.setValue(new Date());
                dateField.setColumns(10);
                dateField.addPropertyChangeListener("value", this);
                // 패널에 텍스트 필드를 배치한다.
                JPanel panel1 = new JPanel();
                JPanel panel2 = new JPanel();
                panel1.add(amountLabel);
                panel1.add(amountField);
                panel2.add(dateLabel);
                panel2.add(dateField);
                add(panel1, BorderLayout.NORTH);
                add(panel2, BorderLayout.CENTER);
                setVisible(true);
        }
        // 필드의 값(value) 속성이 변경되면 호출된다.
        public void propertyChange(PropertyChangeEvent e) {
                Object source = e.getSource();
                if (source == amountField) {
                        amount = ((Number) amountField.getValue()).doubleValue();
                        System.out.println(amount);
                } else if (source == dateField) {
                        date = (Date) (dateField.getValue());
                        System.out.println(date);
                }
        }
}
public class FormattedFieldTest extends JFrame {
        public static void main(String[] args) {
                new FormattedField();
        }
}
```

중간점검

1. 텍스트 필드에서 엔터키나 리턴키를 치면 발생하는 이벤트는 무엇인가?
2. 패스워드를 입력받을 때 사용하는 클래스는?
3. 텍스트 필드에서 getText()로 받은 텍스트를 정수로 변환하려면 어떤 방법을 써야 하는가?

7. 텍스트 영역

텍스트 영역(TextArea)은 텍스트 필드와 비슷하지만 한 줄이 아니라 여러 줄의 텍스트가 들어갈 수 있다.



생성자

생성자	설명
JTextArea()	비어있는 새로운 TextArea를 생성한다.
JTextArea(int rows, int columns)	지정된 줄수와 칸수의 비어있는 TextArea를 생성한다.
JTextArea(String text)	지정된 Text가 입력된 TextArea를 생성한다.

텍스트 영역을 생성하고 초기화

```
textArea = new JTextArea(10, 30);     // 10행 30열의 텍스트 영역을 만든다.
```

JTextArea 생성자의 두개의 매개변수는 각각 텍스트 영역이 나타내야 하는 행과 열의 개수이다. 여기서는 텍스트 영역이 스크롤이 되지 않는데 스크롤이 되게 하려면 여기다가 스크롤 페인을 붙여야 한다. 텍스트 영역에 텍스트를 가져오려면 getText()를 사용하고 텍스트를 넣으려면 setText()를 사용한다.

setEditable(false) 호출은 텍스트 영역을 편집 불가능으로 만든다. 하지만 선택은 가능하고 복사도 가능하지만 텍스트의 내용을 변경할 수 없을 뿐이다.

```
textArea.setEditable(false);
```

텍스트를 추가하려면 append()를 사용한다.

```
textArea.append("겁이 많은 개일수록 큰 소리로 짖는다");
```

메소드

메소드	설명
void setEditable(boolean) boolean isEditable()	사용자가 텍스트 영역에 있는 텍스트의 편집 여부를 설정하거나 반환한다(JTextComponent에 정의).
void setText(String) String getText()	텍스트 영역에 나타난 텍스트를 설정하거나 반환한다. (JTextComponent에 정의)
void setColumns(int); int getColumns()	텍스트 영역의 열의 개수 설정하거나 반환한다.
void setRows(int); int getRows()	텍스트 영역의 행의 개수 설정하거나 반환한다. 이것은 영역의 높이에 대한 힌트이다.
int setTabSize(int)	하나의 탭이 몇 개의 문장에 해당되는지를 설정한다.
int setLineWrap(boolean)	한 줄이 너무 길면 줄을 자동으로 분리할 것인지를 설정한다. 디폴트는 false이다.
void selectAll()	텍스트 영역의 모든 문자를 선택한다. (JTextComponent에 정의)
void append(String)	텍스트 영역의 끝에 문자열을 추가한다.
void insert(String, int)	지정된 위치에 지정된 문자열을 삽입한다.
void replaceRange(String, int, int)	지정된 문자열로 시전에서 종점까지를 바꾼다.
int getLineCount()	줄의 개수를 반환한다.

(예제) 텍스트 필드와 텍스트 영역을 생성항 후에 텍스트 필드에 입력된 텍스트를 텍스트 영역에 추가한다.

TextAreaTest.java

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

class MyTextArea extends JFrame implements ActionListener {
        protected JTextField textField;
        protected JTextArea textArea;

        public MyTextArea() {
                setTitle("Text Area Test");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                textField = new JTextField(30);
                textField.addActionListener(this);
```

```
                textArea = new JTextArea(10, 30);
                textArea.setEditable(false);

                add(textField, BorderLayout.NORTH);
                add(textArea, BorderLayout.CENTER);

                pack();
                setVisible(true);
        }
        public void actionPerformed(ActionEvent evt) {
                String text = textField.getText();
                textArea.append(text + "₩n");
                textField.selectAll();
                // 카렛을 텍스트 영역의 맨 끝으로 가게 하려면 다음과 같이 문장을 사용하면 된다
                textArea.setCaretPosition(textArea.getDocument().getLength());
        }
}
public class TextAreaTest extends JFrame {
        public static void main(String[] args) {
                new MyTextArea();
        }
}
```

(종합예제) - 사용자로부터 아이디, 패스워드, 날짜 등을 받아서 텍스트 영역에 표시하는 프로그램 작성

TextTest.java

```
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
import javax.swing.*;
import javax.swing.text.MaskFormatter;

class TotalExam extends JFrame implements ActionListener {
        protected JTextField id;
        protected JPasswordField password;
        protected JFormattedTextField date;
        protected JTextArea textArea;

        public TotalExam() throws ParseException {
                super("종합 예제");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 2));      // 3행 2열의 그리드 배치

        id = new JTextField(30);        // 최대 30 글자의 텍스트 필드
        id.addActionListener(this);
        panel.add(new JLabel("ID:"));
        panel.add(id);

        password = new JPasswordField(30);
        password.addActionListener(this);
        panel.add(new JLabel("PASSWORD:"));
        panel.add(password);

        MaskFormatter mf = new MaskFormatter("####.##.##"); // 숫자만 입력(#-숫자의미)
        mf.setPlaceholderCharacter('_'); // 입력 위치에 '_'을 표시
        date = new JFormattedTextField(mf);
        date.addActionListener(this);
        panel.add(new JLabel("DATE:"));
        panel.add(date);

        textArea = new JTextArea(10, 30); // 텍스트 영역 생성
        textArea.setEditable(false); // 편집 불가로 설정

        add(panel, BorderLayout.NORTH); // 프레임의 북쪽에 배치
        add(textArea, BorderLayout.CENTER); // 프레임의 중앙에 배치
        pack();
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        // 텍스트 컴포넌트에서 사용자가 엔터키를 치면
        if (e.getSource() == id || e.getSource() == password || e.getSource() == date) {
            String text = "id = " + id.getText() + " password = "
                    + password.getText() + " date = " + date.getText();
            textArea.append(text + "₩n");      // 텍스트 영역에 추가한다.
            id.selectAll();      // 선택
            password.selectAll();
            date.selectAll();
            // 카렛 이동
            textArea.setCaretPosition(textArea.getDocument().getLength());
        }
```

```
        }
}
public class TextTest {
        public static void main(String[] args) throws ParseException {
                TotalExam frame = new TotalExam();
        }
}
```

중간점검

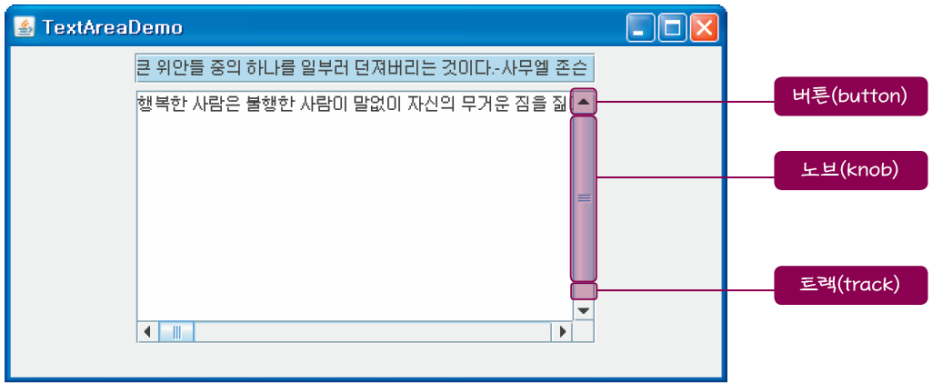
1. 버튼을 누르면 "I've clicked"라는 문자열을 텍스트 영역에 추가하는 프로그램을 설계하고 작성하여 보라.
2. 텍스트 영역에다 텍스트를 추가하는 메소드는?

8. 스크롤 페인

스크롤페인(scroll pane)은 컴포넌트에 스크롤 기능을 제공한다. 스크롤 기능을 추가하려면 스크롤페인에 컴포넌트를 집어 넣어야 한다. 텍스트 영역에 스크롤 기능을 추가하려면 텍스트 영역을 스크롤 페인의 생성자 매개 변수로 넘겨서 스크롤 페인에 추가하면 된다.

```
textArea = new JTextArea(10, 30);                    // 텍스트 영역을 생성한다.
JScrollPane scrollPane = new JScrollPane(textArea);  // ❶
```

① JScrollPane의 어떤 메소드도 호출할 필요가 없는데 그 이유는 스크롤 페인은 자동적으로 모든 일을 처리하기 때문이다. 즉 스크롤바를 생성하고 사용자가 스크롤 노브를 움직이면 클라이언트를 다시 그린다.



• 생성자와 메소드

생성자 또는 메소드	설명
JScrollPane() JScrollPane(Component) JScrollPane(int, int) JScrollPane(Component, int, int)	스크롤 페인을 생성한다. Component 매개변수가 스크롤 페인의 클라이언트가 된다. 두 개의 int형 매개변수는 수직과 수평 방향의 스크롤 바 정책이 된다.
void setViewportView(Component)	스크롤 페인의 클라이언트를 설정한다.
void setVerticalScrollBarPolicy(int) int getVerticalScrollBarPolicy()	수직 방향의 정책을 설정하거나 읽어온다. 가능한 값은 다음과 같다. VERTICAL_SCROLLBAR_AS_NEEDED (디폴트), VERTICAL_SCROLLBAR_ALWAYS, VERTICAL_SCROLLBAR_NEVER.
void setHorizontalScrollBarPolicy(int) int getHorizontalScrollBarPolicy()	수직 방향의 정책을 설정하거나 읽어온다. 가능한 값은 다음과 같다. HORIZONTAL_SCROLLBAR_AS_NEEDED(디폴트) HORIZONTAL_SCROLLBAR_ALWAYS, HORIZONTAL_SCROLLBAR_NEVER.

생성자에 전달되는 두 개의 정수형 매개 변수는 스크롤바의 생성 여부에 대한 정책이다.

• HORIZONTAL_SCROLLBAR_ALWAYS/VERTICAL_SCROLLBAR_ALWAYS : 수평/수직 방향으로 항상 스크롤바가 보이게 된다.

• HORIZONTAL_SCROLLBAR_AS_NEEDED/VERTICAL_SCROLLBAR_AS_NEEDED : 수평/수직 방향으로 필요할 때만 스크롤바가 보이게 된다.

• HORIZONTAL_SCROLLBAR_NEVER/VERTICAL_SCROLLBAR_NEVER : 수평/수직 방향으로 스크롤바를 보이지 않게 한다.

수평 방향은 항상 보이고 수직 방향은 필요할 때만 보이도록 한다.

```
JScrollPane scroll = new JScrollPane(textArea,
            JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYES);
```