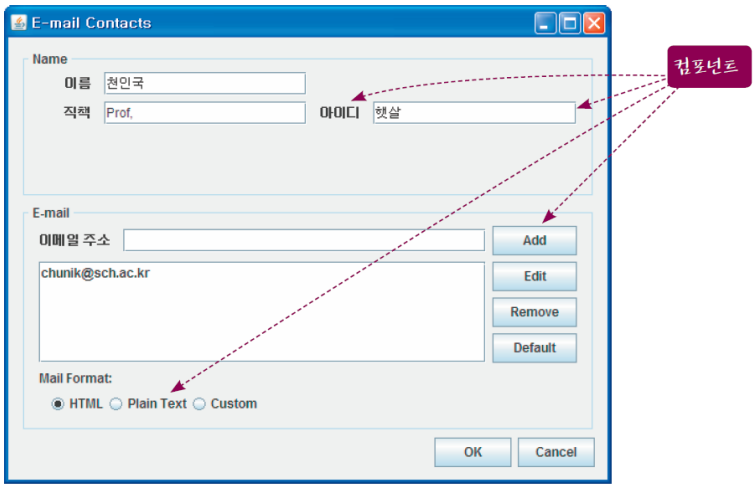
20. 그래픽 사용자 인터페이스 개요

학습목표

- 현재 사용되는 거의 모든 프로그램은 그래픽 사용자 인터페이스를 가지고 있다. 자바로 그래픽 사용자 인터페이스를 만드는데 필요한 내용을 알아보고 쉽게 작성할 수 있다. 그래픽 사용자 인터페이스의 기초를 학습한다.

1. 그래픽 사용자 인터페이스

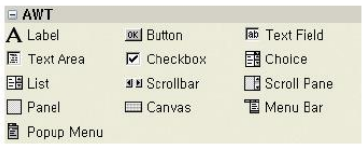
콘솔-기반이란 MS-DOS창과 같이 텍스트만을 사용하여 사용자와 대화하는 것으로 현재는 많이 사용되지 않는 방식이다. 현재 대부분의 프로그램은 버튼이나 스크롤바와 같은 그래픽 사용자 인터페이스(Graphical User Interface : GUI)를 사용한다. GUI를 이용하면 사용자와 상호 작용하는 프로그램을 쉽게 작성할 수 있다. GUI는 컴포넌트들로 만들어진다. 여기서 **컴포넌트(component)**란 레이블, 버튼이나 텍스트 필드와 같은 GUI를 작성하는 기본적인 빌딩 블록을 의미하는 것으로 윈도우 시스템에서는 컨트롤(control)이라고도 부른다.



그래픽 사용자 인터페이스는 컴포넌트들로 제작된다

1) AWT와 스윙

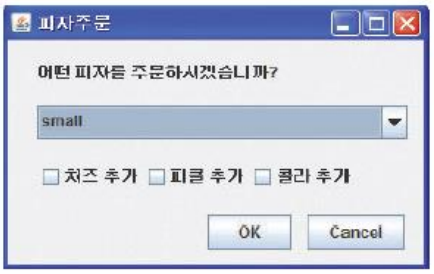
현재 자바에서 사용할 수 있는 GUI 객체 중 하나는 **AWT(Abstract Windows Toolkit)**이고 또 하나는 **스윙(Swing)**이다. AWT는 운영체제가 제공하는 자원을 이용하여서 컴포넌트를 생성한다. AWT의 장점은 여러 플랫폼에서 쉽게 컴포넌트를 제공할 수 있다는 점이다. 하지만 컴포넌트가 플랫폼에 종속적이기 때문에 실행되는 플랫폼에 따라서 컴포넌트의 모습이 달라지게 된다. 따라서 일관된 화면을 제공하는 것이 어렵게 된다.



(a) AWT 컴포넌트

(b) 스윙 컴포넌트

반면에 스윙(Swing)은 AWT와는 달리, 컴포넌트가 자바로 작성되어 있기 때문에 어떤 플랫폼에서도 일관된 화면을 보여줄 수 있다. AWT는 자바의 초기 버전에서 제공되던 것으로 현재는 스윙의 사용이 권장되고 있다.



스윙은 AWT를 대체하기 위하여 AWT의 컴포넌트 이름 앞에 J를 붙인 동일한 기능을 하는 컴포넌트들을 대체용으로 제공한다.

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

2. 스윙의 소개

스윙은 결코 단순한 컴포넌트 툴킷이 아니다. 스윙에는 되돌리기 기능, 변경가능한 텍스트 패키지, 통합 세계화 기능 등이 포함되어 있다. 또한 스윙은 다양한 룩앤필(look and feel)을 제공한다. 드래그앤 드롭 기능, 이벤트 처리, 윈도우 관리 등의 기능도 제공한다.

스윙으로 만든 사용자 인터페이스(출저: java.sun.com)

1) 스윙의 특징

스윙은 JFC(Java Foundation Class)의 일부이다. JFC는 애플리케이션에 그래픽 사용자 인퍼페이스, 풍부한 그래픽 기능, 다양한 언어와 다양한 입력 장치들을 지원한다. JFC가 제공하는 기능들을 정리하면 다음과 같다.

- **스윙 GUI 컴포넌트**

형식화된 텍스트 입력이나 패스워드 필드 동작과 같은 복잡한 기능들이 제공된다.

- **자바 2D API**

그림이나 이미지, 애니메이션 기능을 제공한다. 교체가능한 룩앤필(Look-and-Feel) 지원한다.

- **교체가능한 룩앤필 지원**

룩앤필이란 특정한 운영체제가 제공하는 컴포넌트의 모습과 비슷한 외관을 만드는 것이다.

- **데이터 전송**

자르기, 복사, 붙이기, 드래그앤 드롭 등의 데이터 전송 기능은 스윙 안에 내장되어 내장되어 있다. 되돌리기(undo)와 되풀이(redo) 기능을 손쉽게 제공한다.

2) 스윙의 패키지

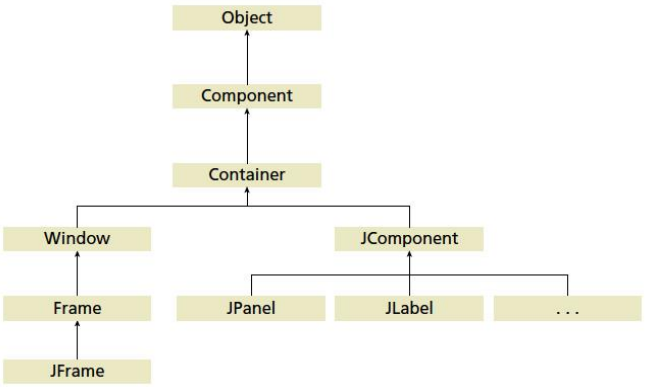
스윙 API는 대단히 강력하고 유연하지만 대단히 방대하며 모두 18개의 공용 패키지로 구성되어 있다.

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

하지만 대부분의 프로그램은 스윙 API 중에서 아주 작은 부분 집합만을 사용한다.

- javax.swing
- javax.swing.event

3) 스윙 클래스 계층 구조

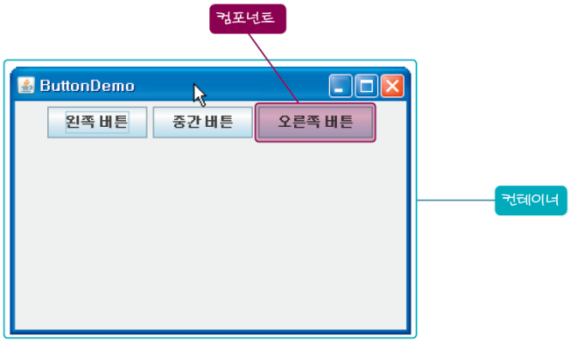


컴포넌트	설 명
Object	모든 클래스는 Object로부터 시작된다. 스윙 컴포넌트들도 Object부터 시작된다.
Component	컴포넌트 클래스는 화면에 표시되어서 사용자와 상호 작용하는 시각적인 객체를 나타낸다. 이 클래스는 모든 스윙 클래스들이 가지고 있는 공통적인 기본 메소드를 정의한다.
Container	컨테이너는 내부에 다른 컴포넌트를 추가할 수 있는 기능을 제공한다.
Window	이것은 경계선, 타이틀바, 버튼을 가지고 있는 컨테이너 객체 중의 하나인 윈도우를 정의한다.
Frame	윈도우의 하나로 자바 GUI 애플리케이션의 기초가 된다.
JFrame	Frame 클래스를 스윙의 출시에 맞추어 변경하는 것이다. 우리가 만드는 스윙 애플리케이션은 적어도 하나의 JFrame 객체를 가지고 있다.
JComponent	JComponent 클래스는 프레임을 제외한 다른 스윙 컴포넌트의 기초가 된다.
JPanel	이 클래스는 컨테이너로서 레이블이나 버튼, 텍스트 필드와 같은 다른 컴포넌트를 배치하고 정렬하는데 사용되는 컨테이너이다. 하나이상의 패널이 프레임에 추가된다
JLabel	레이블은 간단한 문자열을 나타내는데 사용되는 컴포넌트이다.

3. 컨테이너

1) 컨테이너란?

자바가 제공하는 컴포넌트는 크게 기본 컴포넌트와 컨테이너 컴포넌트로 나누어진다. 컨테이너란 다른 컴포넌트들을 내부에 넣을 수 있는 컴포넌트를 의미한다.



• 기본 컴포넌트

JButton, JLabel, JCheckbox, JChoice, JList, JMenu, JTextField, JScrollbar, JTextArea, JCanvas 등이 여기에 속한다.

• 컨테이너 컴포넌트

다른 컴포넌트를 안에 포함할 수 있는 컴포넌트로서 JFrame, JDialog, JApplet, JPanel, JScrollPane 등이 여기에 속한다.

2) 컨테이너의 종류

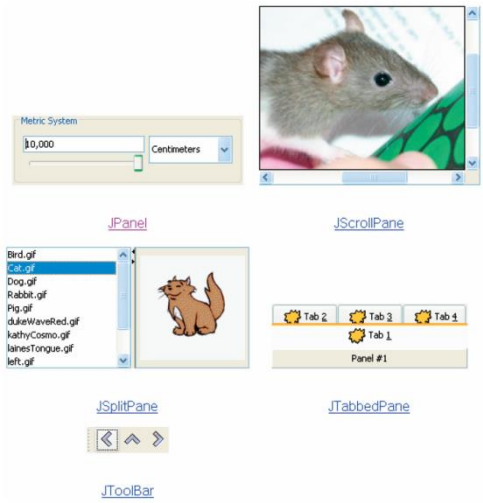
컨테이너는 다시 최상위 컨테이너와 일반적인 컨테이너로 나누어진다.

• 최상위 컨테이너

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

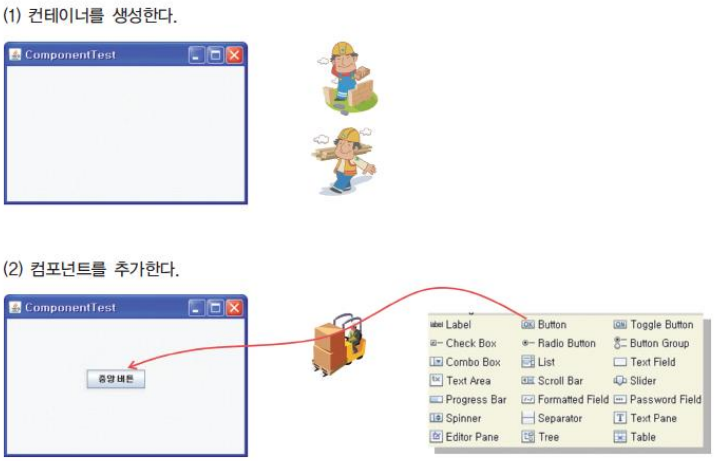
• 일반 컨테이너란

일반적인 컨테이너란 다른 컨테이너 안에 포함될 수 있는 컨테이너로 패널(JPanel), 스크롤페인(JScrollPane) 등을 의미한다.



4. GUI 작성 절차

애플리케이션의 GUI를 작성하려면 먼저 하나의 최상위 컨테이너를 생성하여야 한다. 스윙에는 JFrame, JDialog, JApplet 3가지의 최상위 컨테이너가 존재한다. JFrame은 윈도우와 메뉴를 가지는 일반적인 데스크탑 애플리케이션에 적합하며, JDialog는 메뉴가 없는 대화상자 형식의 간단한 애플리케이션에 사용되고, JApplet은 애플릿을 작성하는데 사용된다. 최상위 컨테이너가 생성되었다면 다음 단계는 애플리케이션에 필요한 컴포넌트를 생성하여서 컨테이너에 추가하는 것이다.



(프레임 생성하기_1)

FrameTest.java

```
import javax.swing.*;

public class FrameTest {
        public static void main(String[] args) {
                JFrame f = new JFrame("Frame Test");
                f.setSize(300, 200); // 프레임의 크기 설정
                f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                f.setVisible(true);
        }
}
```

JFrame의 객체 생성

프로그램설명

```
import javax.swing.*;
```

먼저 스윙 클래스들은 javax.swing 패키지 안에 들어 있다. 여기서 javax는 자바 확장 패키지(java extension package)를 의미한다. 따라서 스윙을 사용하려면 무조건 javax.swing 패키지를 포함하여야 한다.

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

이어서 클래스 FrameTest를 정의하고 main()을 작성한다. main() 안에서 new 연산자를 이용하여 JFrame 객체를 생성한다. JFrame 클래스 생성자의 매개변수는 프레임의 제목이다. 참조 변수 f가 생성된 객체를 가리킨다.

```
f.setSize(300, 200);
```

참조 변수 f를 통하여 setSize()를 호출하여서 프레임의 크기를 설정한다. 현재 프레임의 크기가 가로 300픽셀이고 세로 200 픽셀이 된다. 프레임의 크기를 변경하지 않으면 0×0의 크기를 가진다.

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

사용자가 프레임의 오른쪽 상단에 있는 close 버튼을 눌렀을 경우에 어떤 동작은 취하느냐를 설정한다. 기본 설정은 close 버튼을 누르면 현재 프레임만 닫치지만 프로그램은 종료하지 않는다. 우리는 close 버튼을 눌렀을 경우에 전체 프로그램을 종료하도록 설정한다.

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

마지막 문장은 setvisible(true)을 호출한다. 이것은 프레임을 화면에 나타나게 만든다. 만약 이 문장이 없다면 프레임은 생성되지만 사용자는 볼 수가 없다. 이렇게 하는 이유는 화면에 프레임이 나타나지 않은 상태에서 다른 컴포넌트들이 추가하기 위해서이다.

(프레임 생성하기_2)

MyFrameTest.java

```
import javax.swing.*;

class MyFrame extends JFrame {
        public MyFrame() {
                setSize(300, 200);
```

Jframe을 상속하여 MyFrame을 정의

```
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setTitle("MyFrame");
                setVisible(true);
        }
}

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

MyFrame의 객체 생성

(프레임 생성하기_3)

MyFrame.java

```
import javax.swing.*;

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

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

Jframe을 상속하여서 MyFrame을 정의

main()이 MyFrame 안으로 이동

(컴포넌트 생성 및 추가_1)

컨테이너가 생성되었다면 원하는 컴포넌트 객체들을 컨테이너에 추가한다. 자바 윈도우 애플리케이션의 경우에는 대개 JFrame 클래스 객체가 컨테이너가 되고 애플릿의 경우에는 JApplet 클래스가 컨테이너가 된다. add()메소드를 이용하면 컴포넌트를 컨테이너에 추가할 수 있다.

MyFrameTest1.java

```
import javax.swing.*;

class MyFrame extends JFrame {
        public MyFrame() {
                JButton button = new JButton("버튼");   // 버튼 생성
                this.add(button);                       // 버튼을 프레임에 추가
                setSize(300, 200);                      // 프레임 크기 설정
```

컴포넌트 생성 및 추가

```
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setTitle("MyFrame");                    // 프레임 타이틀 설정
                setVisible(true);                       // 프레임을 화면에 표시
        }
}

public class MyFrameTest1 {
        public static void main(String[] args) {
                MyFrame f = new MyFrame();          // 프레임 객체 생성
        }
}
```

버튼이 전체 화면을 차지하는 이유는 컴포넌트를 배치하는 배치 관리자가 BorderLayout이기 때문이다. BorderLayout은 동서남북, 중앙으로 컴포넌트를 배치할 수 있다. 현재 버튼은 중앙에 배치되었다. 다른 배치관리자를 사용하면 버튼의 크기와 위치가 달라진다.

(컴포넌트 생성 및 추가_2)

배치관리자인 FlowLayout이란 컴포넌트를 물이 흐르듯이 순차적으로 배치하는 방식인데 이 배치관리자는 setLayout()메소드를 호출하여 설정하면 된다.

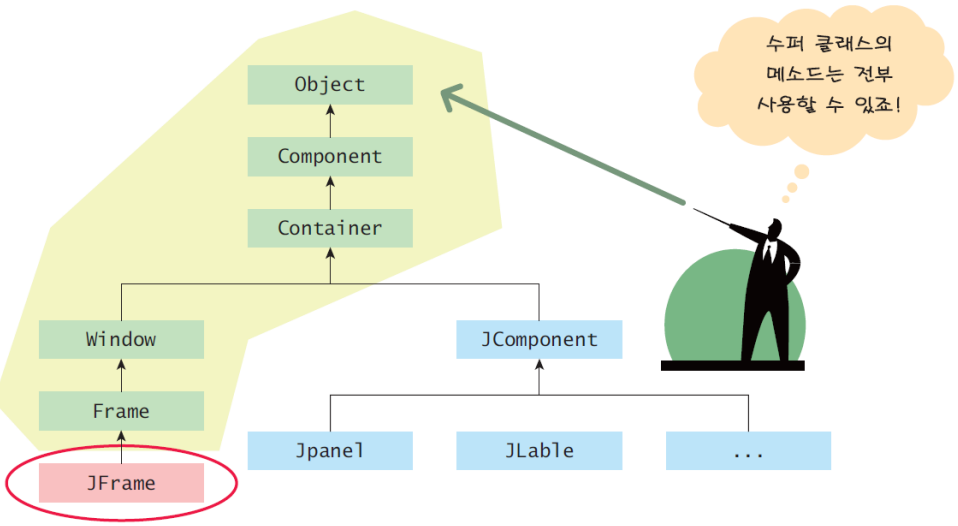
MyFrameTest2.java

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

class MyFrame extends JFrame {
        public MyFrame() {
                JButton button = new JButton("버튼");
                this.add(button);
                setSize(300, 200);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setTitle("MyFrame");
                setLayout(new FlowLayout());
                setVisible(true);
        }
}

public class MyFrameTest2 {
        public static void main(String[] args) {
                MyFrame f = new MyFrame();
        }
}
```

프레임의 배치 관리자 설정

5. 프레임 속성 변경

프레임도 객체이므로 객체의 속성을 변경하려면 set으로 시작되는 설정자 메소드를 사용한다.



스윙 관련 클래스의 계층 구조

1) 조상 클래스

Object

모든 클래스는 Object로부터 시작된다. 스윙 컴포넌트들도 Object부터 시작된다.

Component

컴포넌트 클래스는 화면에 표시되어서 사용자와 상호 작용하는 시각적인 객체를 나타낸다.

Container

내부에 다른 컴포넌트를 추가할 수 있는 기능을 제공한다. 예를 들어서 이 클래스의 add()를 사용하면 컨테이너 안에 컴포넌트를 추가할 수 있다.

Window

경계선, 타이틀 바, 버튼을 가지고 있는 윈도우를 정의한다.

Frame

자바 GUI 애플리케이션의 기초가 된다.

JFrame

Frame 클래스를 스윙의 출시에 맞추어 변경한 것이다.

2) 중요 메소드

setLocation(x, y) , setBounds(x, y, width, height), setSize(width, height)

프레임의 위치와 크기를 설정한다.

setIconImage(IconImage)

윈도우 시스템에 타이틀 바, 태스크 스위처에 표시할 아이콘을 알려준다.

setTitle()

타이틀 바의 제목을 변경한다.

setResizable(boolean)

사용자가 크기를 조절할 수 있는지를 설정한다.

MyFrameTest2.java

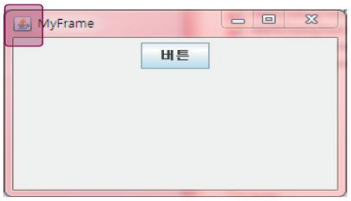
```
01  ...
02
03  class MyFrame extends JFrame {
04      public MyFrame() {
05          Toolkit kit = Toolkit.getDefaultToolkit();
06          Dimension screenSize = kit.getScreenSize();
07          setSize(300, 200);
08          setLocation(screenSize.width / 2, screenSize.height / 2)
09          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10          setTitle("MyFrame");
11          Image img = kit.getImage("icon.gif");
12          setIconImage(img);
13          setLayout(new FlowLayout());
14          JButton button = new JButton("버튼");
15          this.add(button);
16          setVisible(true);
17      }
18  }
19  ...
```

현재 화면의 크기를 얻는다.

프레임의 위치를 현재 화면의 중앙으로 한다.

아이콘을 icon.gif로 변경 icon.gif는 이클립스의 프로젝트 폴더로 드래그한다.

실행결과



6. 기초 컴포넌트

1) 프레임

프레임은 메뉴를 붙일 수 있는 윈도우이다.

생성자

메소드	설명
JFrame()	타이틀이 없는 새로운 프레임을 만든다.
JFrame(String title)	지정된 타이틀의 새로운 프레임을 만든다.

메소드

생성자 또는 메소드	설명
void add(Component c)	지정된 컴포넌트를 프레임에 추가한다.
JMenuBar getJMenuBar()	이 프레임에 대한 메뉴를 얻는다.
void pack()	프레임을 크기를 추가된 컴포넌트들의 크기에 맞도록 조절한다.
void remove(Component c)	지정된 컴포넌트를 프레임에서 제거한다.
void setDefaultCloseOperation()	사용자가 프레임을 닫을 때 수행되는 동작을 설정한다. 일반적으로 JFrame.EXIT_ON_CLOSE으로 지정한다.
void setIconImage(Icon image)	프레임이 최소화되었을때의 아이콘 지정
void setLayout(LayoutManager layout)	프레임위에 놓이는 컴포넌트들을 배치하는 배치관리자 지정, 디폴트는 BorderLayout 배치관리자
void setLocation(int x, int y)	프레임의 x좌표와 y좌표를 지정한다.
void setResizeable(boolean value)	프레임의 크기 변경 허용 여부
void setSize(int width, int height)	프레임의 크기 설정
void setMenuBar(JMenuBar menu)	현재 프레임에 메뉴바를 붙인다.

2) 패널

패널(panel)은 컴포넌트들을 포함하고 있도록 설계된 컨테이너 중의 하나이다. 레이블이나 버튼과 같은 컴포넌트들을 화면에 표시하는 일반적인 방법은 패널에 이들 컴포넌트를 추가하고 그 패널을 프레임에 추가하는 것이다. 물론 패널을 쓰지않고 프레임에 컴포넌트들을 직접 추가할 수도 있지만 별도의 패널을 쓰는 것이 유지 보수 및 배치 관리에 좋은 경우가 많다. 패널은 컴포넌트를 붙일 수 있는 판이다

생성자

메소드	설명
JPanel()	새로운 패널을 생성한다.
JPanel(boolean isDoubleBuffered)	만약 매개변수가 참이면 더블 버퍼링을 사용한다.
JPanel(LayoutManager layout)	지정된 배치 관리자를 사용하는 패널을 생성한다.

메소드

메소드	설명
void add(Component c)	지정된 컴포넌트를 패널에 추가한다.
void remove(Component c)	지정된 컴포넌트를 패널에서 제거한다.
void setLayout(LayoutManager layout)	배치 관리자를 지정한다. 디폴트는 FlowLayout이다.
void setLocation(int x, int y)	패널의 위치를 지정한다.
void setSize(int width, int height)	패널의 크기를 지정한다.
void setToolTipText(String text)	사용자가 마우스를 패널의 빈 곳에 올려놓으면 툴팁을 표시한다.

패널에 컴포넌트를 추가하려면 역시 add()메소드를 사용한다.

```
Panel panel = new Panel();
panel.add(new Button("시작"));
panel.add(new Button("종료""));
```

3) 레이블

레이블(label)은 컴포넌트 중에서 아마 가장 간단한 것이다. 레이블은 읽기 전용 텍스트를 표시하기 위한 컴포넌트이다. 레이블은 다양한 용도록 사용하는데 컴포넌트들의 캡션을 표시하거나 도움이 되는 정보 또는 계산의 결과를 표시하는데 사용될 수 있다. 레이블을 생성하려면 JLabel 클래스를 이용한다.

생성자

메소드	설명
JLabel()	새로운 레이블을 생성한다.
JLabel(String text)	지정된 텍스트를 표시하는 레이블을 생성한다.

메소드

메소드	설명
String getText()	레이블의 텍스트를 반환한다.
void setText(String text)	레이블의 텍스트를 설정한다.
void setToolTipText(String text)	사용자가 마우스를 레이블 위에 올려놓으면 툴팁을 표시한다.
void setVisible(boolean value)	레이블을 보이게 하거나 감춘다.

레이블을 생성할 때 표시할 텍스트를 생성자에 넘긴다

```
JLabel label = new JLabel("안녕하세요?");
```

레이블 객체를 생성하고 나중에 레이블의 텍스트를 설정하는 방법도 있다.

```
JLabel label = new JLabel();
label.setText("안녕하세요?");
```

MyFrameTest3.java

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

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

		JPanel panel = new JPanel();                    // 패널 생성
		JLabel label = new JLabel("안녕하세요?");    // 레이블 생성
		JButton button = new JButton("버튼");       // 버튼 생성
		panel.add(label);                               // 패널에 레이블 추가
		panel.add(button);                              // 패널에 버튼 추가
		add(panel);                                     // 패널을 프레임에 추가
		setVisible(true);
	}
}
```

```
public class MyFrameTest3 {
	public static void main(String[] args) {
		MyFrame f = new MyFrame();
	}
}
```

패널은 컨테이너이므로 눈에 보이는 부분은 없으며 패널의 배치 관리자는 디폴트가 FlowLayout이므로 변경할 필요가 없다.

4) 버튼

버튼(button)은 사용자가 클릭했을 경우, 이벤트를 발생하여 원하는 동작을 하게 하는데 이용된다. 버튼에서 변경 가능한 것은 버튼 안의 텍스트, 버튼 텍스트의 폰트, 버튼의 전경색, 배경색, 그리고 버튼의 상태(활성, 비활성)이다.

생성자

메소드	설명
Button ()	레이블이 없는 버튼을 생성한다.
Button (String label)	지정된 레이블의 버튼을 생성한다.

메소드

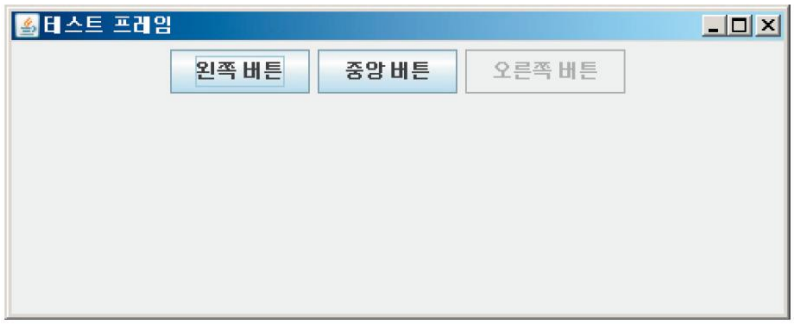
메소드	설명
String getText()	버튼의 현재 텍스트를 반환한다.
void setText(String text)	버튼의 텍스트를 설정한다.
void doClick()	사용자가 버튼을 누른 것처럼 이벤트를 발생한다.
void setBorderPainted(booelan value)	버튼의 경계를 나타내거나 감춘다.
void setContentAreaFilled(boolean value)	버튼의 배경을 채울 것인지를 지정한다.
void setEnabled(boolean value)	버튼을 활성화하거나 비활성화한다.
void setRolloverEnabled(boolean value)	마우스가 버튼 위에 있으면 경계를 진하게 하는 롤오버 효과를 설정
void setToolTipText(String text)	사용자가 마우스를 버튼 위에 올려놓으면 툴팁을 표시한다.
void setVisible(boolean value)	버튼을 보이게 하거나 감춘다.

MyFrameTest4.java

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

class MyFrame extends JFrame {
	public MyFrame() {
	 setSize(500, 200);
	 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        setTitle("테스트 프레임");

        JPanel panel = new JPanel();                    // 패널 생성
        JButton b1 = new JButton();                     // 버튼 생성
        b1.setText("왼쪽 버튼");                        // 버튼의 텍스트 설정
        JButton b2 = new JButton("중앙 버튼");          // 버튼 생성
        JButton b3 = new JButton("오른쪽 버튼");        // 버튼 생성
        b3.setEnabled(false);                           // 세 번째 버튼을 불활성으로 설정

        // 컴포넌트를 패널에 추가
        panel.add(b1);
        panel.add(b2);
        panel.add(b3);

        add(panel);                                     // 패널을 프레임에 추가
        setVisible(true);                               // 프레임을 화면에 표시
    }
}
public class MyFrameTest4 {
    public static void main(String[] args) {
        MyFrame f = new MyFrame();
    }
}
```

실행결과



5) 텍스트 필드

텍스트 필드(text field)는 입력이 가능한 한 줄의 텍스트 필드를 만드는 데 사용된다. 편집을 가능하게 할 수도 있고, 불가능하게 할 수도 있다.

생성자

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

메소드

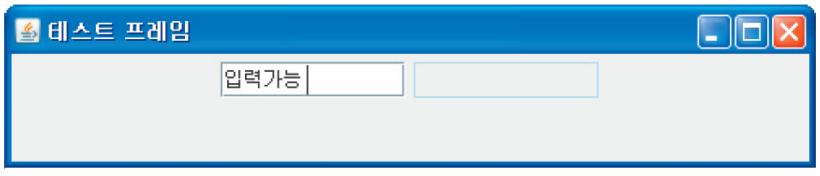
메소드	설명
`void setText(String text)`	지정된 문자열을 텍스트 필드에 쓴다.
`String getText()`	텍스트 필드에 입력된 문자열을 반환한다.
`void setEditable(boolean)` `boolean isEditable()`	사용자가 텍스트를 입력할 수 있는지 없는지를 설정하고 반환한다.

MyFrameTest5.java

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

class MyFrame extends JFrame {
	public MyFrame() {
		setSize(500, 100);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setTitle("테스트 프레임");

		JPanel panel = new JPanel();
		JTextField t1 = new JTextField(10);
		JTextField t2 = new JTextField(10);
		t2.setEditable(false);

		// 컴포넌트를 패널에 추가
		panel.add(t1);
		panel.add(t2);

		add(panel); // 패널을 프레임에 추가
		setVisible(true);
	}
}
public class MyFrameTest5 {
	public static void main(String[] args) {
		MyFrame f = new MyFrame();
	}
}
```

텍스트 필드 생성

실행결과



[Swing 예제 1]

ExamPanelTest.java

```
import java.awt.*;
import javax.swing.*;
class ExamPanel extends JFrame{
	public ExamPanel(){
		setSize(600,150);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setTitle("패널 테스트");

		JPanel plFirst = new JPanel();
		JPanel plSecond = new JPanel();
		JPanel plThird = new JPanel();

		JLabel lblPizzaMenu = new JLabel("자바 피자 입니다. 피자의 종류를 선택하세요");
		plSecond.add(lblPizzaMenu);

		JButton btnBul = new JButton("불고기 피자");
		JButton btnPotato = new JButton("포테이토 피자");
		JButton btnSweetPotato = new JButton("고구마 피자");
		plThird.add(btnBul);
		plThird.add(btnPotato);
		plThird.add(btnSweetPotato);

		plFirst.add(plSecond);
		plFirst.add(plThird);;
		add(plFirst);
		setVisible(true);
	}
}
public class ExamPanelTest{
	public static void main(String[] args){
		ExamPanel f = new ExamPanel();
	}
}
```