Power Java

제13장 그래픽 사용자 인터페이스 개요

이번 장에서 학습할 내용

- 그래픽 사용자 인터페이스
- 스윙의 소개
- 컨테이너
- GUI 작성 절차
- 기초 컴포넌트

그래픽 사용자 인터페이스

- 그래픽 사용자 인터페이스(Graphical User Interface, 간단히 GUI)
- 컴포넌트들로 구성

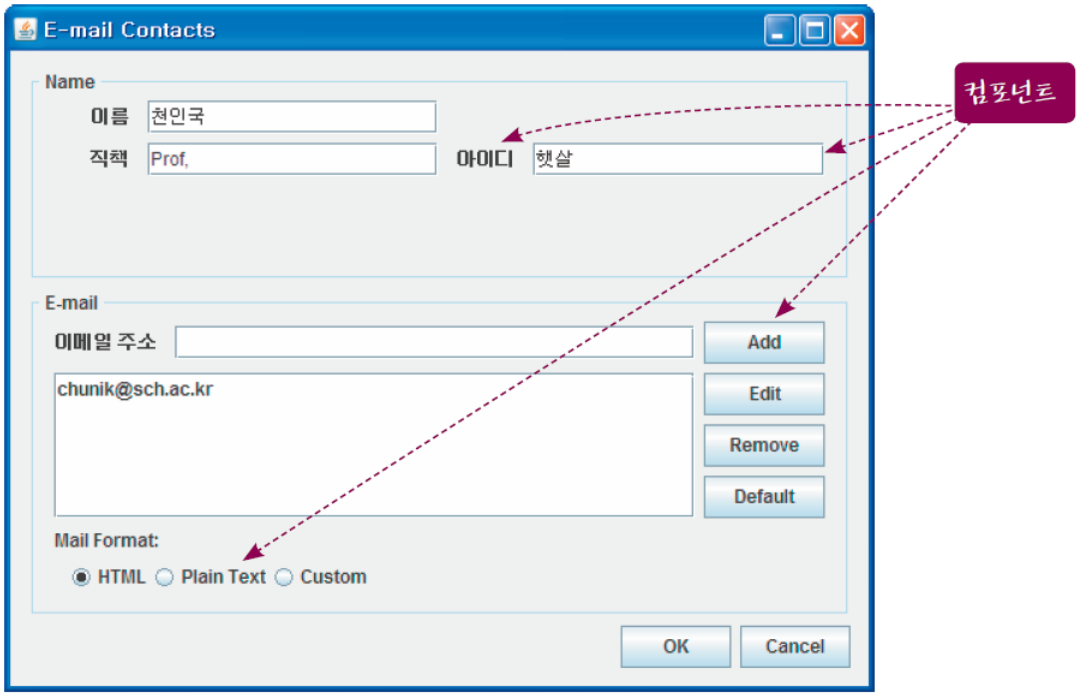
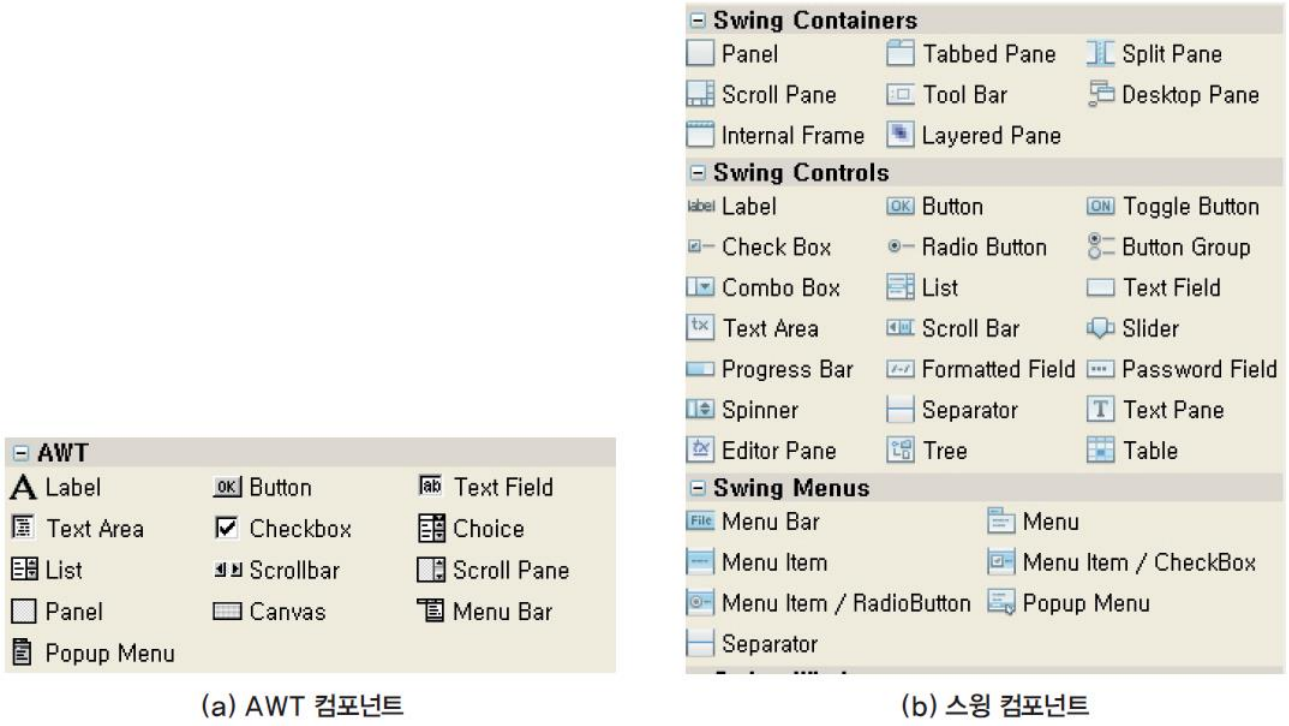


그림13-1. 그래픽 사용자 인터페이스는 컴포넌트들로 제작된다.

AWT와 스윙



(a) AWT 컴포넌트 (b) 스윙 컴포넌트

그림13-2. AWT와 스윙 컴포넌트

AWT와 SWING의 비교

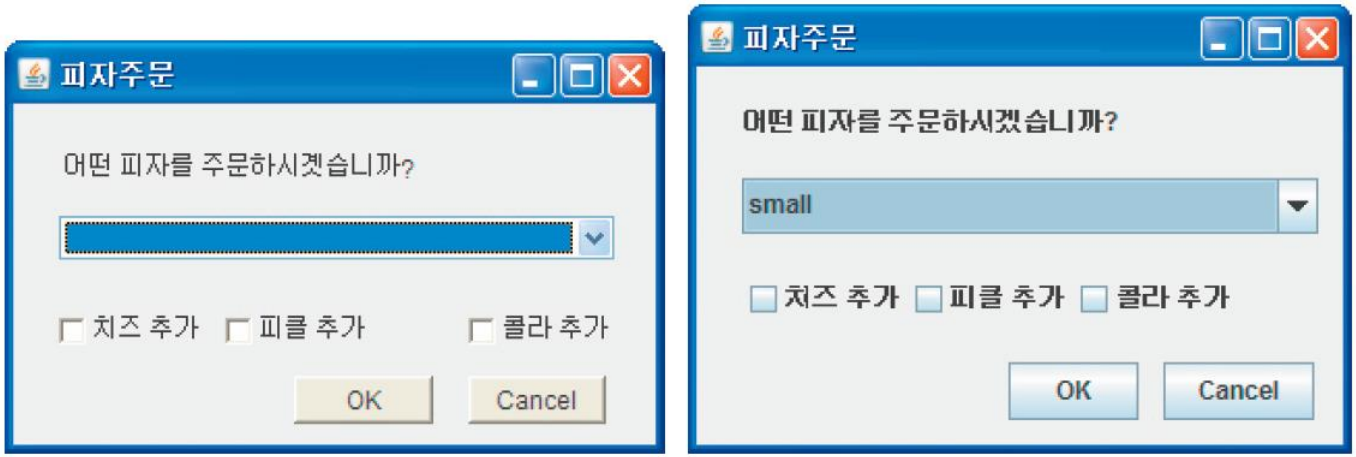


그림13-3. 동일한 GUI를 AWT와 스윙으로 만들어본 예

AWT와 스윙의 비교

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

스윙

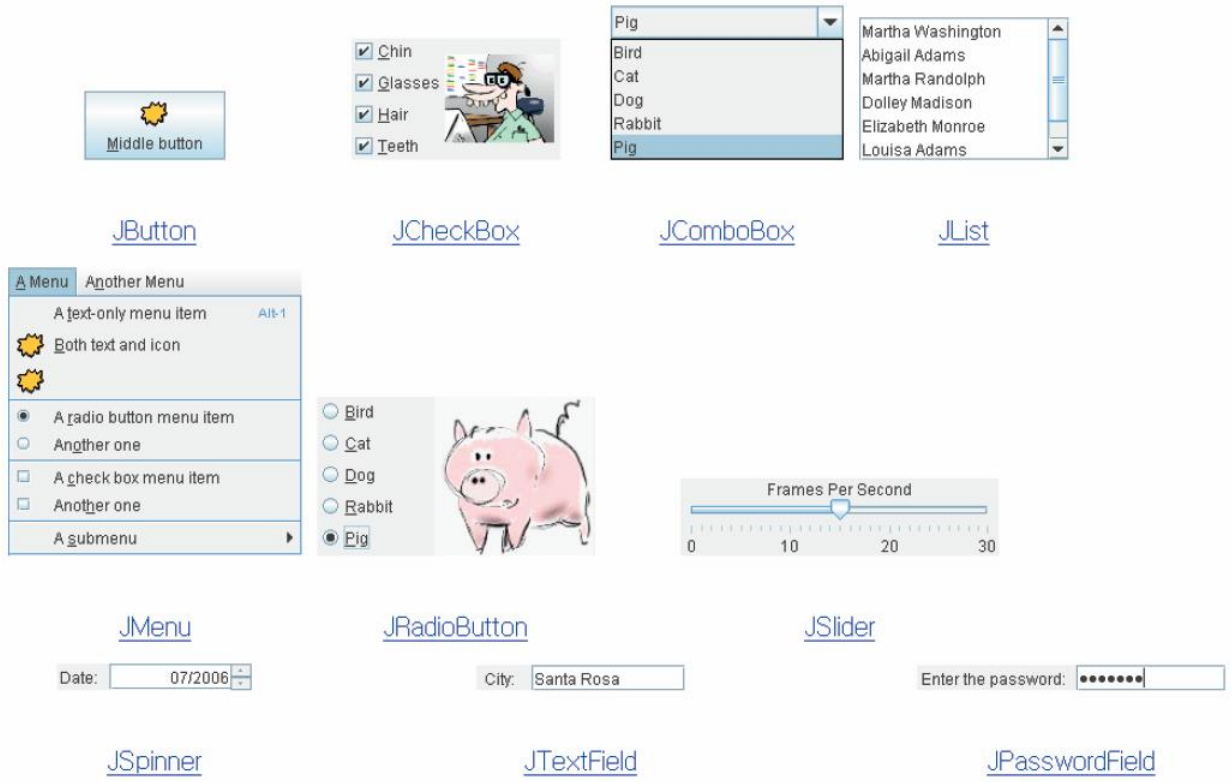


그림13-4. 스윙으로 만든 사용자 인터페이스(출저: java.sun.com)

스윙의 특징

- 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) 지원
- 교체 가능한 룩앤필 지원

- 데이터 전송
  - 자르기, 복사, 붙이기, 드래그앤 드롭 등의 데이터 전송 기능 제공
  - 되돌리기(undo)와 되풀이(redo) 기능을 손쉽게 제공

중간점검

중간점검

1. 그래픽 사용자 인터페이스를 구성하는 요소들을 무엇이라고 하는가?
2. AWT와 스윙의 차이점은 무엇인가?
3. 되돌리기, 복사 및 붙여넣기 등도 스윙에서 제공되는가?
4. 스윙의 클래스 계층도에서 `setVisible()` 메소드는 어떤 클래스들이 사용할 수 있는가?

컨테이너란?

- 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.

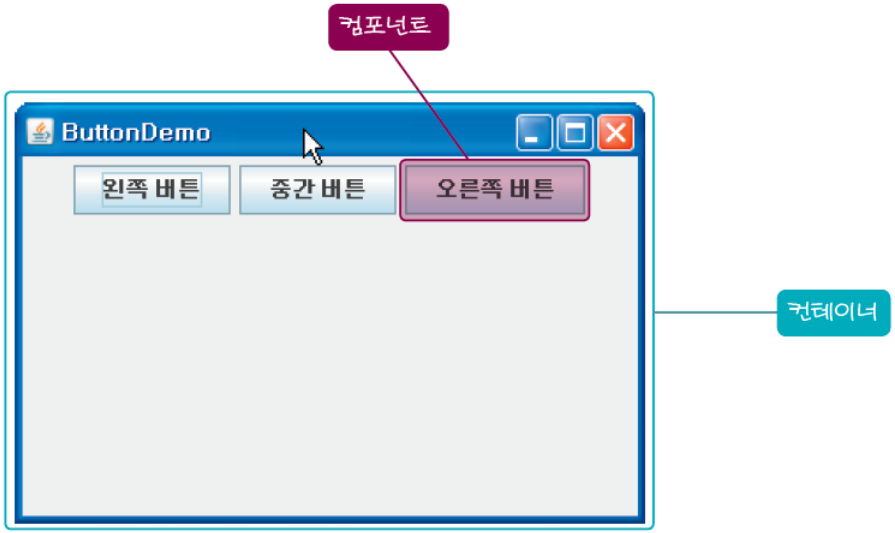


그림13-5. 컨테이너와 컴포넌트

컨테이너의 종류

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

그림13-6. 최상위 컨테이너(그림 출처: java.sun.com)

컨테이너의 종류

- 일반 컨테이너: 다른 컨테이너 안에 포함될 수 있는 컨테이너로 패널(JPanel), 스크롤 페인(JScrollPane) 등을 의미한다.

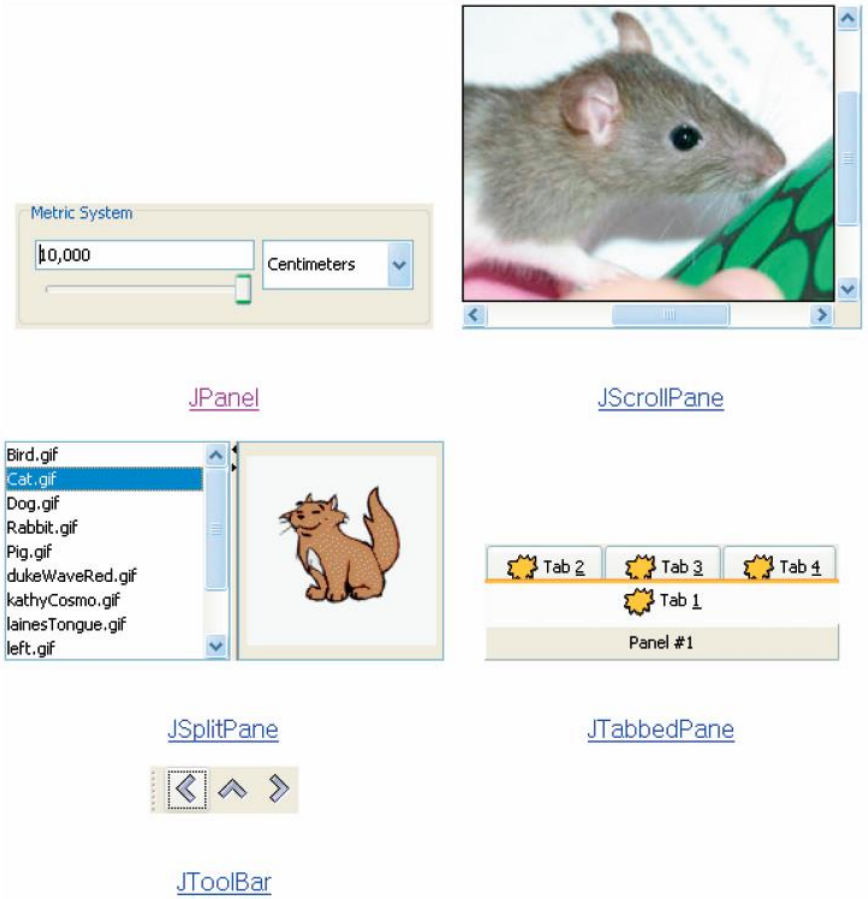


그림13-7. 일반 컨테이너(그림 출처: java.sun.com)

중간 점검 문제

중간점검

1. 컨테이너는 어떤 기능을 하는가?
2. 최상위 컨테이너의 특징은? 최상위 컨테이너에 속하는 클래스들을 나열하라.

GUI 작성 절차

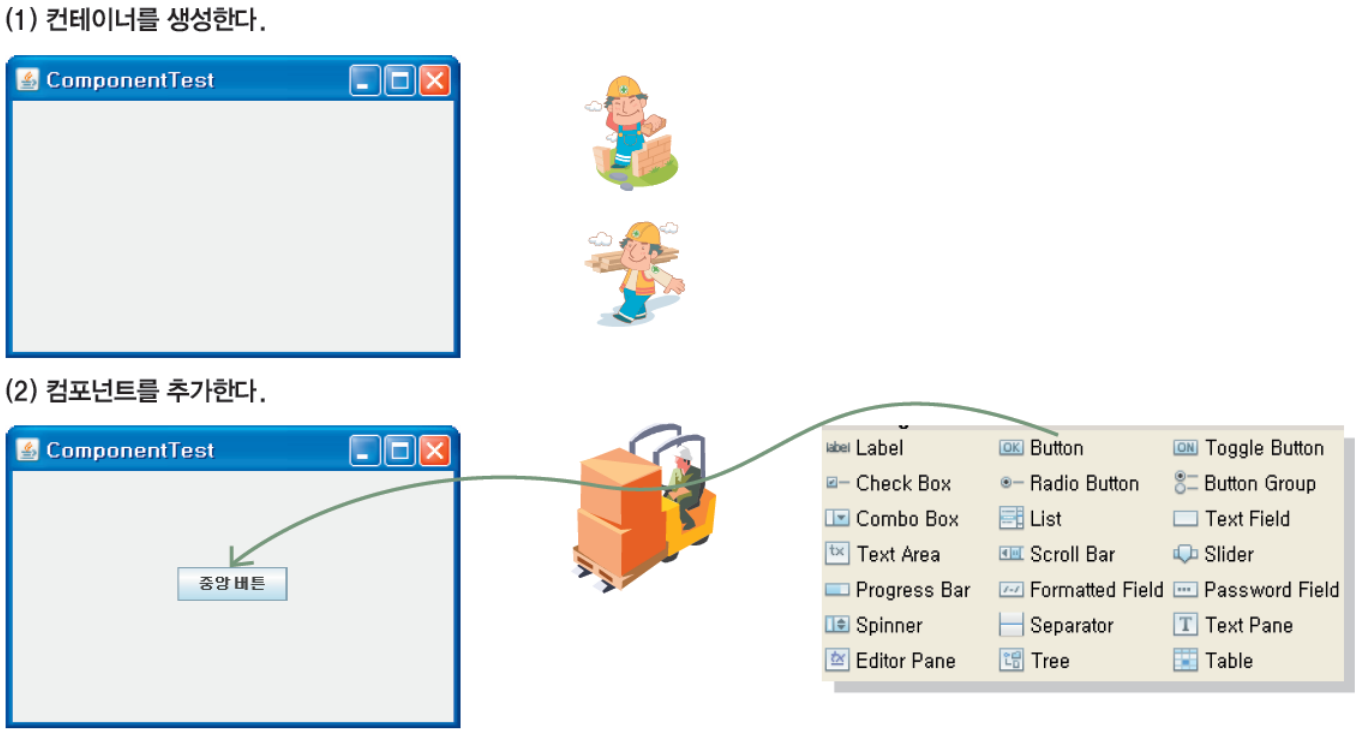


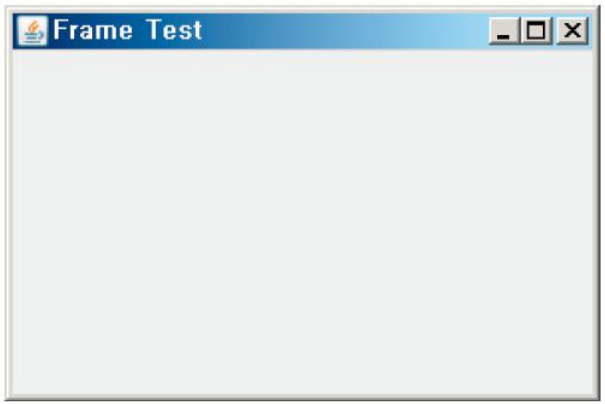
그림13-8. GUI작성순서

프레임 생성하기 #1

FrameTest.java

```
01  import javax.swing.*;
02
03  public class FrameTest {
04     public static void main(String[] args) {
05        JFrame f = new JFrame("Frame Test");
06        f.setSize(300, 200);
07        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
08        f.setVisible(true);
09     }
10  }
```

실행결과



프레임 생성하기 #1

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 이용하여서 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 생성 #2

MyFrameTest.java

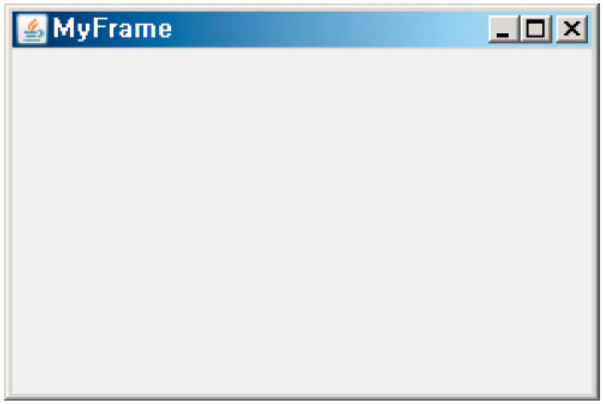
```
01  import javax.swing.*;
02
03  class MyFrame extends JFrame {
04      public MyFrame() {
05          setSize(300, 200);
06          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
07          setTitle("MyFrame");
08          setVisible(true)
09      }
10  }
11  public class MyFrameTest {
12      public static void main(String[] args) {
13          MyFrame f = new MyFrame();
14      }
15   }
```

JFrame을 상속받은 MyFrame을 정의한다.

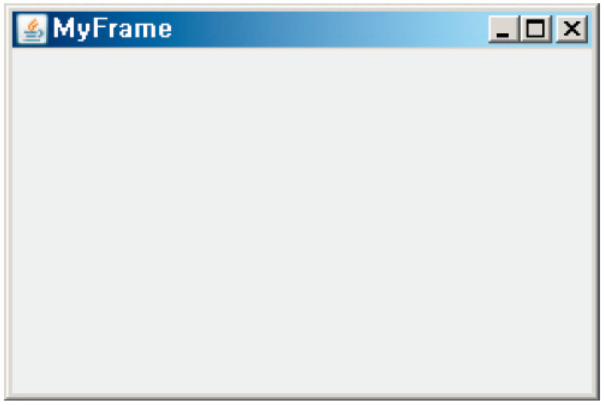
생성자에서 각종 초기화를 한다.

MyFrame 객체를 생성한다. 여기서부터 모든 일이 시작된다.

실행결과



실행결과



프로그램설명

```
class MyFrame extends JFrame {
```

MyFrame 클래스는 JFrame이라고 불리는 클래스를 상속받는다. MyFrame 클래스는 GUI 컴포넌트들을 표시하는 기초적인 프레임을 정의한다.

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

생성자가 하는 일은 프레임에 대하여 여러 가지 선택 사항들을 설정하는 것이다.

```
MyFrame f = new MyFrame();
```

main()에서 MyFrame 클래스의 새로운 인스턴스를 생성한다. 현재는 프레임을 생성만 할 뿐 특별한 작업은 하지 않는다. 그리고 프레임은 사용자가 close 버튼을 누르지 않는 한 종료되지 않는다. 생성자가 호출된다.

컴포넌트 생성 및 추가

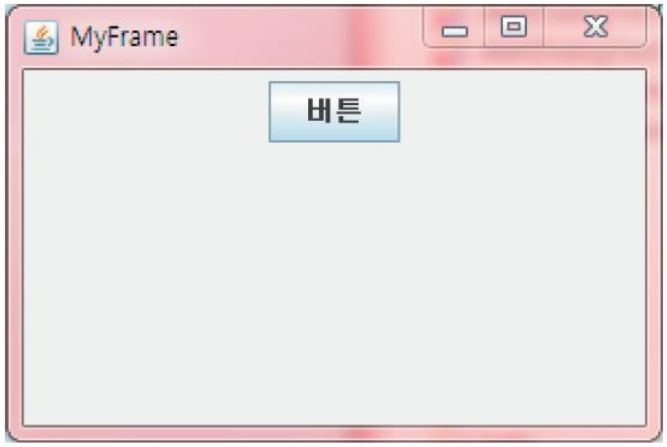
MyFrameTest1.java

```
01  import javax.swing.*;
02
03  class MyFrame extends JFrame {
04     public MyFrame() {
05        setSize(300, 200);
06        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
07        setTitle("MyFrame");
08
09        setLayout(new FlowLayout());
10        JButton button = new JButton("버튼");
11        this.add(button);
12        setVisible(true);
13     }
14  }
15  public class MyFrameTest1 {
16     public static void main(String[] args) {
17        MyFrame f = new MyFrame();
18     }
19  }
```

아직 배치 관리자를 학습하지 않았지만 버튼의 모습을 확실하게 볼 수 있도록 배치 관리자를 FlowLayout으로 변경하여 보자. FlowLayout이란 컴포넌트를 물이 흐르듯이 순차적으로 배치하는 방식이다. 배치 관리자는 setLayout() 메소드를 호출하면 된다. 배치 관리자 객체는 FlowLayout 클래스로 생성한다.

JButton의 객체를 생성한 후에 프레임에 추가한다.

실행결과



중간 점검

중간점검

1. 버튼이 있는 프레임을 생성하는 절차를 설명하라.
2. 프레임에 버튼을 두 개 추가하려면 어떻게 해야 하는가?

프레임의 속성 변경하기

- 프레임의 속성을 변경시키는 방법을 생각하여 보자.
- 조상 클래스들의 메소드도 사용할 수 있다!

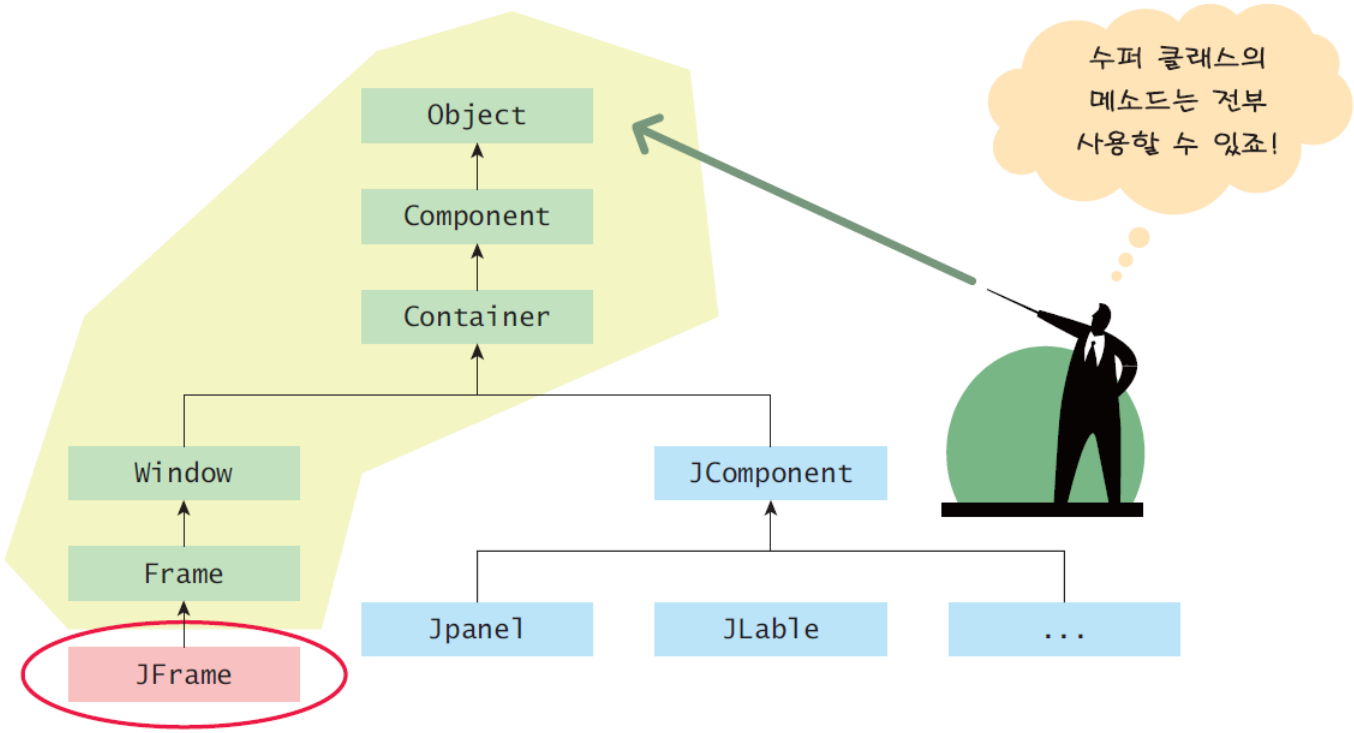


그림13-9. 스윙 관련 클래스의 계층 구조

조상 클래스

- Object
  - 모든 클래스는 Object로부터 시작된다. 스윙 컴포넌트들도 Object 부터 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.

중요한 메소드

- JFrame
  - Frame 클래스를 스윙의 출시에 맞추어 변경한 것이다.
- 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

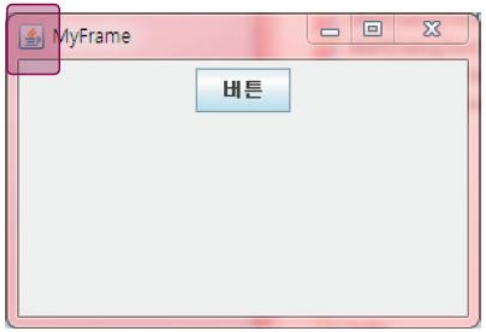
```
01  ...
02
03  class MyFrame extends JFrame {
04      public MyFrame() {
05          Toolkit kit = Toolkit.getDefaultToolkit();
06          Dimension screenSize = kit.getScreenSize();
07          setSize(300, 200);
08          setLocation(screenSize.width / 2, screenSize.height / 2)
09          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
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



프레임(frame)

- 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.

패널

- 패널(panel)은 컴포넌트들을 가질 수 있는 컨테이너

그림13-9. 패널은 컴포넌트를 붙일 수 있는 판이다.

패널의 메소드

- 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.

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

레이블

- 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.

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

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

예제

```
01  import java.awt.*;
02  import javax.swing.*;
03
04  class MyFrame extends JFrame {
05     public MyFrame() {
06        setSize(300, 200);
07        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
08        setTitle("MyFrame");
09
10        JPanel panel = new JPanel();
11        JLabel label = new JLabel("안녕하세요?");
12        JButton button = new JButton("버튼");
13        panel.add(label);
14        panel.add(button);
15        add(panel);
16        setVisible(true);
17     }
18  }
19  public class MyFrameTest3 {
20     public static void main(String[] args) {
21        MyFrame f = new MyFrame();
22     }
23  }
```

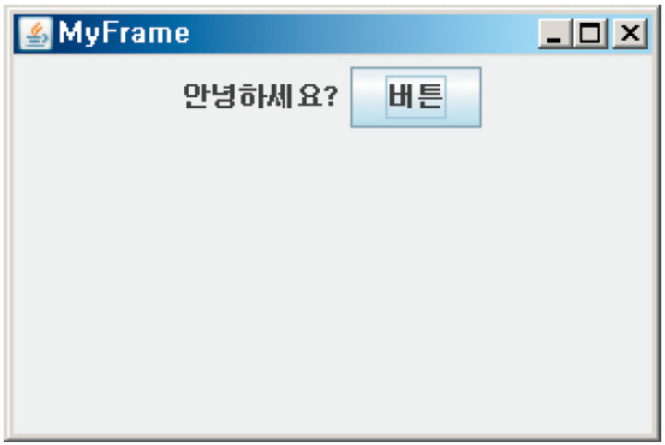
여기서는 컨테이너로 패널을 사용한다. JPanel()을 이용하여서 패널을 생성한 후에 레이블과 버튼을 패널에 붙이고 다시 패널을 프레임에 추가한다. 패널은 컨테이너이므로 눈에 보이는 부분은 없다. 패널의 배치 관리자는 디폴트가 FlowLayout이므로 변경할 필요가 없다.

레이블과 버튼 생성

패널에 레이블과 버튼을 추가한다.

패널을 프레임에 추가

실행결과



버튼

- 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.

예제

MyFrameTest4.java

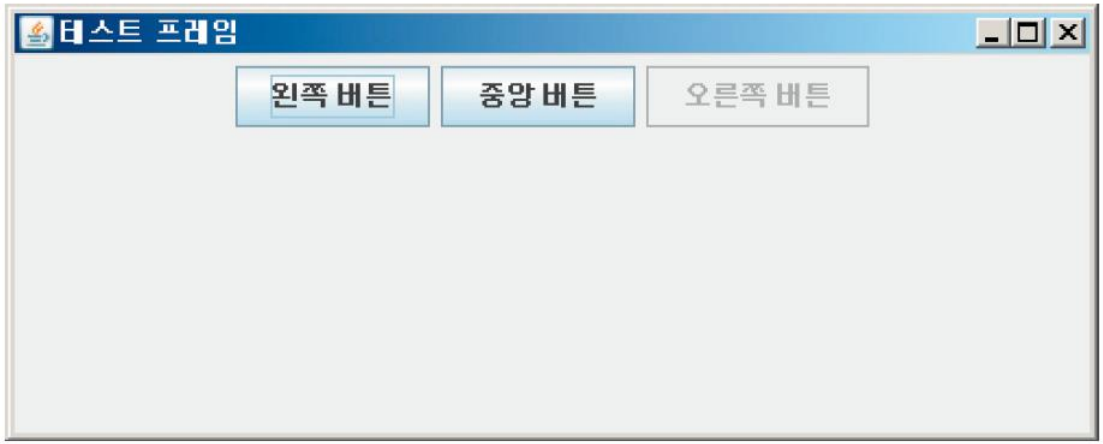
```
01  import java.awt.*;
02  import javax.swing.*;
03
04  class MyFrame extends JFrame {
05     public MyFrame() {
06        setSize(500, 200);
07        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
08        setTitle("테스트 프레임");
09
10        JPanel panel = new JPanel();
11        JButton b1 = new JButton();
12        b1.setText("왼쪽 버튼");
13        JButton b2 = new JButton("중앙 버튼");
14        JButton b3 = new JButton("오른쪽 버튼");
15        b3.setEnabled(false); // 세 번째 버튼을 불활성으로 설정
16
17        // 컴포넌트를 패널에 추가
18        panel.add(b1);
19        panel.add(b2);
20        panel.add(b3);
21
```

먼저 패널을 생성하고 세개의 버튼을 만들어서 추가하였다. 버튼의 레이블은 생성자를 통하여 전달할 수도 있고 setText() 메소드를 사용할 수도 있다. setEnabled() 메소드를 사용하여서 버튼을 불활성화하면 오른쪽 버튼처럼 나타난다.
현재까지는 버튼을 눌러도 아무런 반응이 나타나지 않는다. 아직까지는 이벤트 처리기를 붙이지 않은 탓이다. 다음 장에서 이벤트 처리를 학습하도록 하자.

예제

```
22          add(panel); // 패널을 프레임에 추가
23          setVisible(true);
24       }
25    }
26
27    public class MyFrameTest4 {
28       public static void main(String[] args) {
29          MyFrame f = new MyFrame();
30       }
31    }
```

실행결과



텍스트 필드

- 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.

예제

PyFrameTest5.java

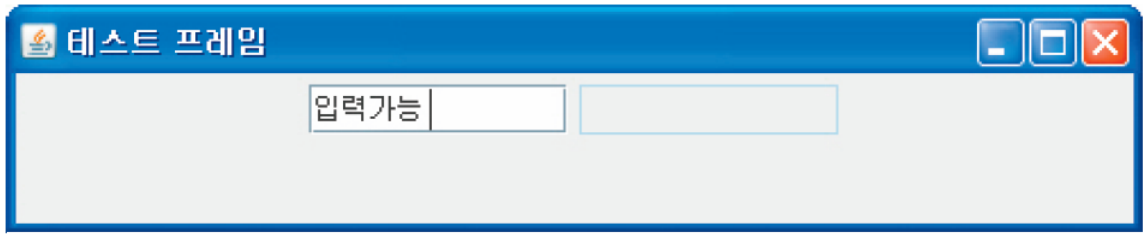
```
01  import java.awt.*;
02  import javax.swing.*;
03
04  class MyFrame extends JFrame {
05     public MyFrame() {
06
07        setSize(500, 100);
08        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
09        setTitle("테스트 프레임");
10
11        JPanel panel = new JPanel();
12        JTextField t1 = new JTextField(10);
13        JTextField t2 = new JTextField(10);
14        t2.setEditable(false);
15
16        panel.add(t1);  // 패널에 텍스트 필드를 추가한다.
17        panel.add(t2);
18
19        add(panel);     // 패널을 프레임에 추가
20        setVisible(true);
21     }
22  }
```

텍스트 필드를 2개 생성하여 패널에 추가한다.
두 번째 텍스트 필드는 입력 금지로 설정한다.

예제

```
23  public class MyFrameTest5 {
24      public static void main(String[] args) {
25          MyFrame f=new MyFrame();
26      }
27  }
```

실행결과



중간점검

1. 프레임에 하나의 버튼과 하나의 레이블을 가지고 있는 GUI를 작성하는 절차를 설명하라.
2. 패널에 버튼 3개를 추가하는 경우와 프레임에 버튼 3개를 추가하는 경우를 비교하여 보자. 외관이 어떻게 달라지는가?

Q & A