Chapter 14

GUI 기초

Contents

소개

- 그래픽의 사전적 의미는 상품화, 정보 제공, 엔터테인먼트 등을 벽, 캔버스, 컴퓨터 화면, 종이 같은 표면에 나타내는 시각적 표현
- 그래픽의 예 : 사진, 드로잉, 그래프, 다이어그램, 이미지 등
- 그래픽은 문자나 숫자보다 더 빠르고 쉽게 정보를 전달할 수 있기 때문에 컴퓨터 분야에서 매우 중요
- 비교적 최근에 출시된 자바는 초기 버전부터 그래픽을 고려해서 설계
- 사용자와 컴퓨터의 상호작용 방식

(a) CUI 방식

(b) GUI 방식

소개(2)

- AWT
  - 운영체제가 제공하는 네이티브 UI컴포넌트를 이용하는 자바 라이브러리
  - 중량 컴포넌트라고 하며, 운영체제에 따라 외형이 다름
- 스윙
  - 운영체제의 도움을 받지 않고 순수하게 자바로 작성되어 있기 때문에 스윙 컴포넌트를 경량 컴포넌트라고 함
  - 모든 스윙 컴포넌트는 AWT컴포넌트와 완전히 호환
- JavaFX
  - RIA 시장을 장악하고 있던 어도비의 플래시와 MS의 실버라이트에 대항
  - 처음에는 JavaFX스크립트라는 별도의 언어로 개발되어서 불편
  - 2011년 JavaFX 2.0은 순수한 자바로 개발되었고 JavaFX 2.2는 JDK와 JRE에 포함
  - JavaFX의 장점을 포함하려면 밑바닥부터 다시 설계해야 한다는 부담감 때문에 현재는 스윙과 JavaFX가 양존

스윙의 특징

- 룩앤필 – 플랫폼 독립적인 외형
- 풍선 도움말 – 스윙 컴포넌트는 풍선 도움말을 지원
- 더블 버퍼링 – 그래픽 성능을 향상시키기 위한 기능 제공
- MVC 모델 기반
- 이미지 아이콘 사용 가능
- 보더 – javax.swing.border 패키지로 제공

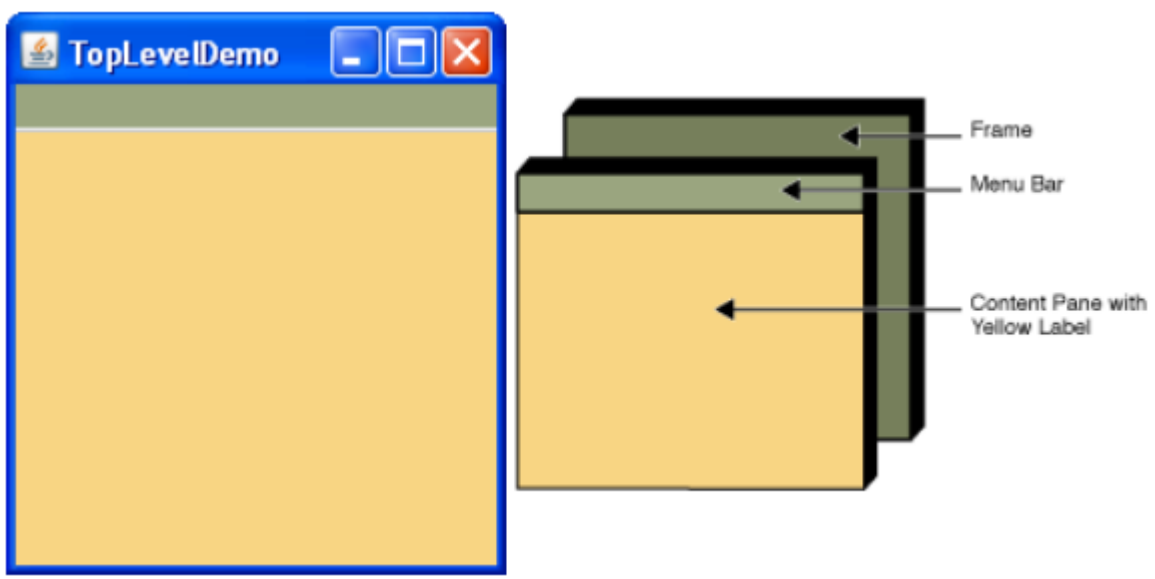
컴포넌트와 컨테이너의 개념

- 컴포넌트
  - 버튼, 레이블, 텍스트 필드 등 GUI를 작성하는 기본적인 빌딩 블록을 의미
  - 사용자 인터페이스를 생성하는 객체로, 윈도우 시스템에서는 컨트롤에 해당

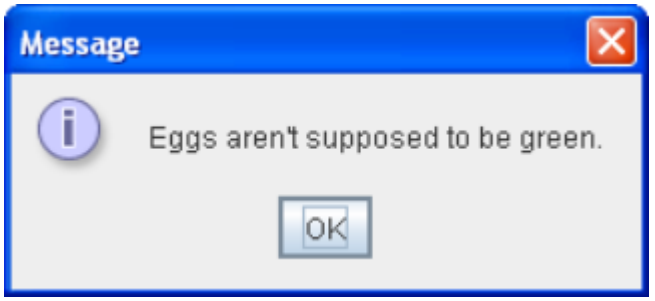
- 컨테이너
  - 컴포넌트를 부착하는 특수한 컴포넌트를 의미
  - 컴포넌트를 부착할 수 있는 프레임이나 패널 등이 대표적인 컨테이너 클래스

컴포넌트와 컨테이너의 개념(2)

- top-level 컨테이너
  - JFrame
  - JDialog
  - JApplet



JFrame



JDialog

스윙 기반의 GUI프로그램 맛보기

- 예제 : [예제 14-1]

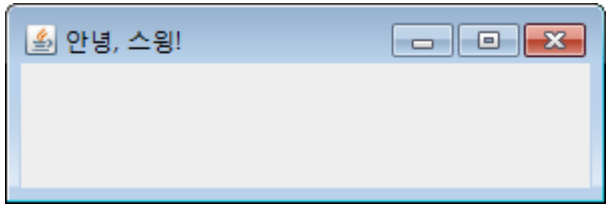
```
01 import javax.swing.JFrame;
02
03 public class HelloSwingDemo {
04   public static void main(String[] args) {
05     JFrame f = new JFrame();
06
07     f.setTitle("안녕, 스윙!");
08     f .setSize(300, 100);
09     f.setVisible(true);
10   }
11 }
```

GUI 프로그래밍에 필요한 패키지를 임포트한다.

프레임을 생성한다.

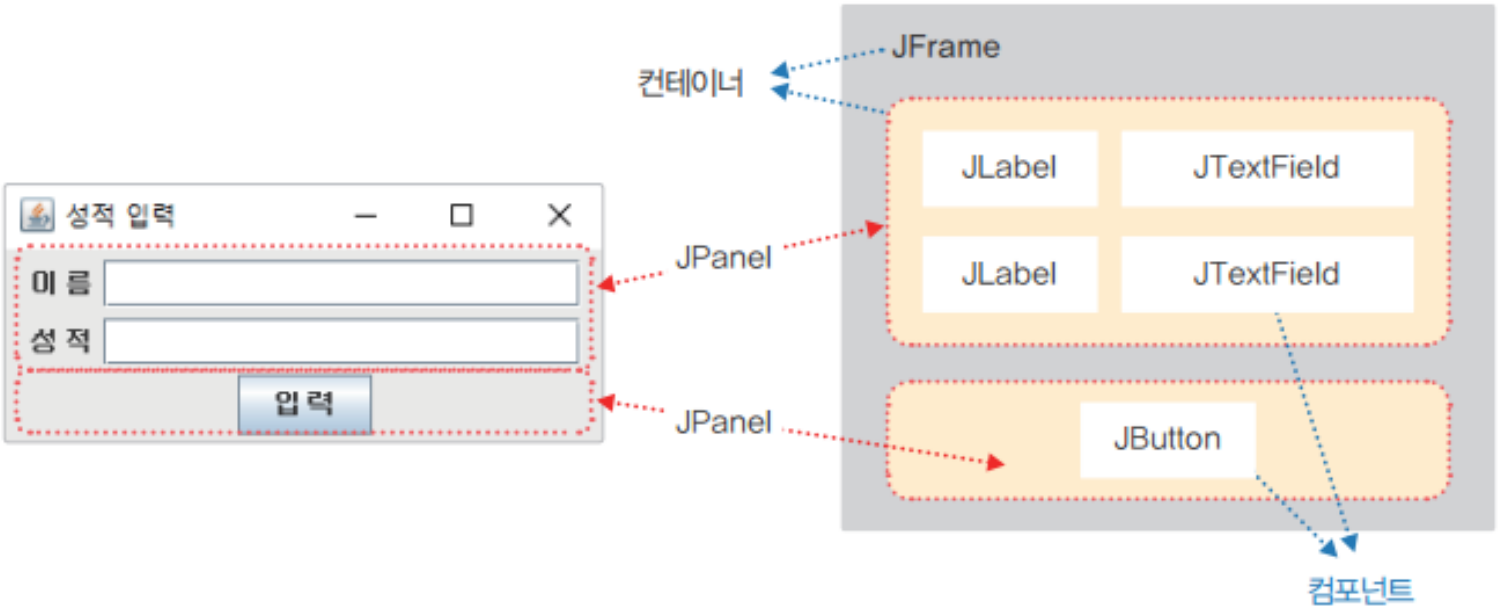
프레임의 타이틀과 크기를 설정한다.

프레임을 화면에 보이도록 한다.



컨테이너와 컴포넌트

- 컨테이너는 내부의 배치 관리자를 사용해 컴포넌트 위치를 결정하고 자신에게 부착
- 스윙 애플리케이션을 작성하려면 스윙 애플리케이션의 최상위 컨테이너인 프레임을 생성 필요



프레임 생성(1)

- JFrame

```
public class JFrame extends Frame
  implements WindowConstants, Accessible, RootPaneContainer
```

  - 복잡한 구조로 구성되어 있지만, 개발자가 자주 접하는 부분은 menu bar와 content pane
  - 주요 생성자

```
JFrame()                // 타이틀이 없는 JFrame 객체를 생성한다.
JFrame(String title)    // 명시된 타이틀을 가진 JFrame 객체를 생성한다.
```

  - 주요 메서드

메서드	설명
Container getContentPane( )	프레임의 컨텐트페인 객체를 반환한다.
void pack( )	컴포넌트를 부착하기에 적절한 윈도우 크기로 조절한다.
void setDefaultCloseOperation(int operation)	닫기 버튼을 클릭할 때 기본 작동을 결정한다.
void setIconImage(Image image)	윈도우 아이콘을 설정한다.
void setLayout(LayoutManager manager)	윈도우의 배치 관리자를 설정한다.
void setJMenuBar(JMenuBar menubar)	프레임의 메뉴바를 주어진 메뉴바로 지정한다.

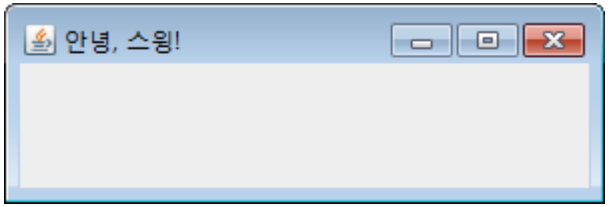
프레임 생성(2)

- 이외에도 Component 클래스가 제공하는 add( ), Window 클래스가 제공하는 setVisible( ), Frame 클래스가 제공하는 setResizable( ) 등도 자주 사용
- 주요 상수

상수	설명
DISPOSE_ON_CLOSE	종료할 때 모든 자원을 반납한다.
DO_NOTHING_ON_CLOSE	종료할 때 아무 일도 하지 않는다.
EXIT_ON_CLOSE	종료할 때 애플리케이션도 강제로 종료한다.
HIDE_ON_CLOSE	종료할 때 창을 숨긴다.

- 예제 : [예제 14-2]

```java
package sec03;

import javax.swing.JFrame;

class MyFrame extends JFrame {
    MyFrame() {
        setTitle("안녕, 스윙!");
        setSize(300, 100);
        setVisible(true);
    }
}

public class JFrame1Demo {
    public static void main(String[] args) {
        new MyFrame();
    }
}
```



- 예제 : [예제 14-3]

```java
package sec03;

import javax.swing.JFrame;

public class JFrame2Demo extends JFrame {
    JFrame2Demo() {
        setTitle("안녕, 스윙!");
        setSize(300, 100);
        setVisible(true);
    }

    public static void main(String[] args) {
        new JFrame2Demo();
    }
}
```

프레임에 컴포넌트 추가(1)

- 스윙 컴포넌트를 프레임에 부착하려면 Container 클래스가 제공하는 add( ) 메서드를 호출

- 예제 : [예제 14-3]


```
04  public class JFrame3Demo extends JFrame {
05    JFrame3Demo() {
06       setTitle("안녕, 스윙!");
07
08       JButton b = new JButton("버튼");
09       add(b);
10
11       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12       setSize(300, 100);
13       setVisible(true);
14    }
15
16    public static void main(String[] args) {
17       new JFrame3Demo();
18    }
19  }
```

버튼을 프레임의 컨텐트페인에 부착한다.

패널로 프레임에 컴포넌트 추가(1)

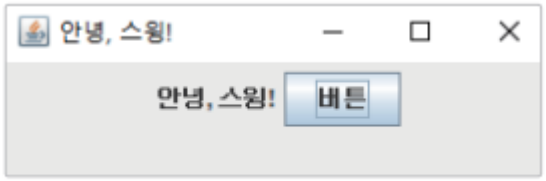
- JPanel
  - 다른 컴포넌트를 배치할 수 있는 가상의 투명한 공간 제공

  - 주요 생성자

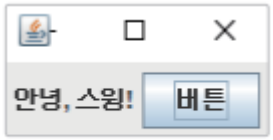
```
// 플로 레이아웃과 더블 버퍼를 가진 JPanel 객체를 생성한다.
JPanel()
```

패널로 프레임에 컴포넌트 추가(2)

- 예제 : [예제 14-5]



setSize( ) 메서드 호출



pack( ) 메서드 호출

```
06  public class JFrame4Demo extends JFrame {
07    JFrame4Demo() {
08       setTitle("안녕, 스윙!");
09
10       JPanel p = new JPanel();
11       JLabel l = new JLabel("안녕, 스윙!");
12       JButton b = new JButton("버튼");
13       p.add(l);
14       p.add(b);
15       add(p);
16
17       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18       setSize(300, 100);
19       // pack();
20       setVisible(true);
21    }
22
23    public static void main(String[] args) {
24       new JFrame4Demo();
25    }
```

패널을 생성한다.

레이블과 버튼을 생성

레이블과 버튼을 패널에 부착한다.

패널을 프레임에 부착한다.

setSize( )는 창 크기를 정하고, pack( )은 내용에 알맞게 창 크기를 조절한다.

배치 관리자의 역할과 종류

- 배치 관리자의 역할
  - 부착할 컴포넌트 위치를 결정해서 적절히 배치하며, 컨테이너의 크기가 변하면 컴포넌트를 재배치

- 배치 관리자 설정 및 제거

```
setLayout(new GridLayout());    // GridLayout으로 배치 관리자를 변경한다.
setLayout(null);                // 배치 관리자를 제거한다.
```

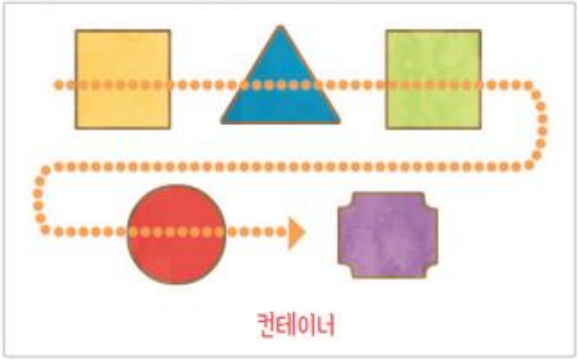
- 컨테이너와 기본 배치 관리자

컨테이너	기본 배치 관리자
JDialog	BorderLayout
JFrame	
JWindow	
JApplet	FlowLayout
JPanel	

배치 관리자로 컴포넌트 배치(1)

- FlowLayout
  - 주요 생성자

```
// 중앙 정렬, 5픽셀 간격의 FlowLayout 객체를 생성한다.
FlowLayout()

// 명시된 정렬, 5픽셀 간격의 FlowLayout 객체를 생성한다.
FlowLayout(int align)

// 명시된 정렬 및 간격의 FlowLayout 객체를 생성한다.
FlowLayout(int align, int hgap, int vgap)
```



  - 주요 상수(정렬 방법)

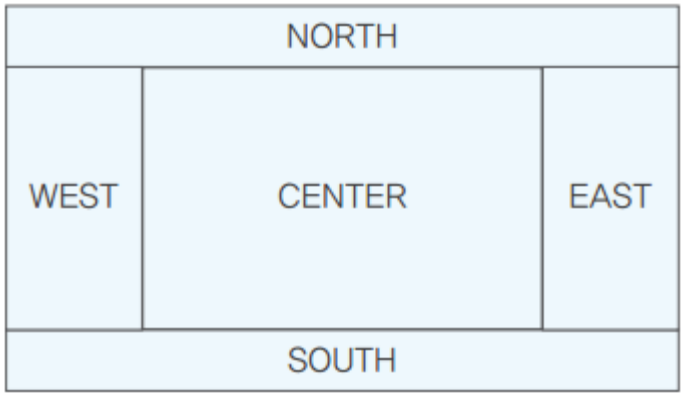
```
왼쪽 정렬 : FlowLayout.LEFT
오른쪽 정렬 : FlowLayout.RIGHT
중앙 정렬(기본) : FlowLayout.CENTER
```

• 예제 : [예제 14-6]

```java
package sec04;

import java.awt.ComponentOrientation;
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class FlowLayoutDemo extends JFrame {
    FlowLayoutDemo() {
        setTitle("플로우 레이아웃!");

        JPanel p = new JPanel(new FlowLayout());
        p.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);

        JButton b1 = new JButton("B 1");
        JButton b2 = new JButton("버튼 2");
        JButton b3 = new JButton("Button 3");
        JButton b4 = new JButton("Button Four");
        p.add(b1);
        p.add(b2);
        p.add(b3);
        p.add(b4);
        add(p);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 110);
        setVisible(true);
    }

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

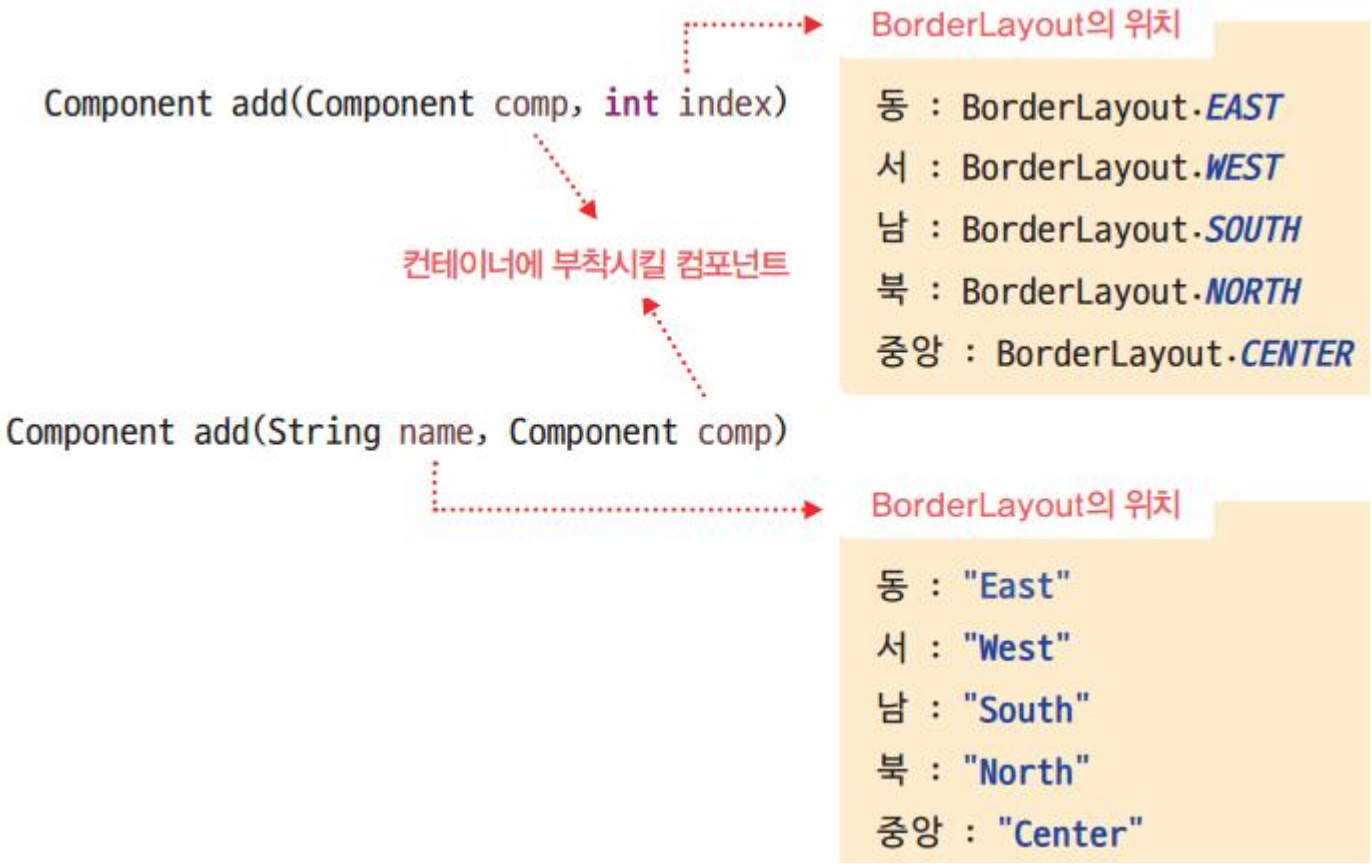
배치 관리자로 컴포넌트 배치(2)

- BorderLayout



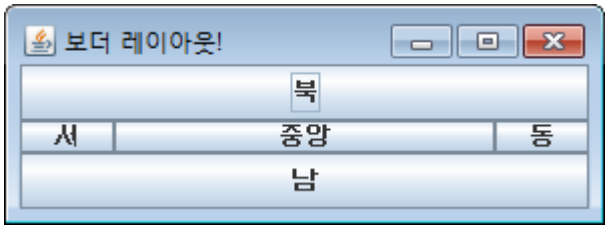
  - 주요 생성자

```
// 간격이 없는 BorderLayout 객체를 생성한다.
BorderLayout()

// 명시한 간격을 가진 BorderLayout 객체를 생성한다.
BorderLayout(int hgap, int vgap)
```


BorderLayout의 위치
Component add(Component comp, int index)
동 : BorderLayout.EAST
서 : BorderLayout.WEST
남 : BorderLayout.SOUTH
북 : BorderLayout.NORTH
중앙 : BorderLayout.CENTER
컨테이너에 부착시킬 컴포넌트
Component add(String name, Component comp)
BorderLayout의 위치
동 : "East"
서 : "West"
남 : "South"
북 : "North"
중앙 : "Center"

- 예제 : [예제 14-7]

```java
package sec04;

import java.awt.BorderLayout;

public class BorderLayoutDemo extends JFrame {
    BorderLayoutDemo() {
        setTitle("보더 레이아웃!");
        setLayout(new BorderLayout());

        add("East", new JButton("동"));
        add("West", new JButton("서"));
        add("South", new JButton("남"));
        add(new JButton("북"), BorderLayout.NORTH);
        add(new JButton("중앙"), BorderLayout.CENTER);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 110);
        setVisible(true);
    }

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



배치 관리자로 컴포넌트 배치(3)

- GridLayout
  - 주요 생성자

```
// 하나의 행과 열로 구성된 GridLayout 객체를 생성한다.
GridLayout()

// 명시된 행과 열로 구성된 GridLayout 객체를 생성한다.
GridLayout(int rows, int cols)

// 명시된 행과 열, 명시된 간격의 GridLayout 객체를 생성한다.
GridLayout(int rows, int cols, int hgap, int vgap)
```

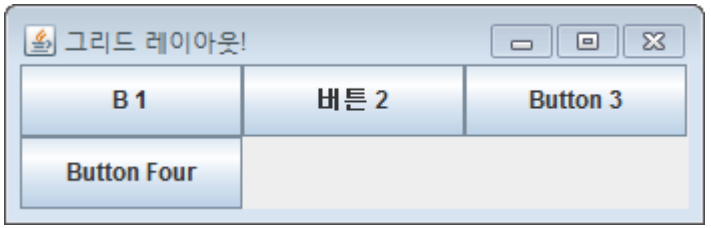
  - GridLayout객체를 생성할 때 행과 열의 개수를 0이상의 정수로 명시
  - 행이나 열의 값이 0이면 필요한 만큼의 행이나 열을 생성. 그러나 행과 열의 개수로 동시에 0은 사용 금지

Figure 1: Horizontal, Left-to-Right

Figure 2: Horizontal, Right-to-Left

- 예제 : [예제 14-8]

```java
package sec04;

import java.awt.GridLayout;

public class GridLayoutDemo extends JFrame {
    GridLayoutDemo() {
        setTitle("그리드 레이아웃!");
        setLayout(new GridLayout(0, 3));

        add(new JButton("B 1"));
        add(new JButton("버튼 2"));
        add(new JButton("Button 3"));
        add(new JButton("Button Four"));

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(350, 110);
        setVisible(true);
    }

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



배치 관리자로 컴포넌트 배치(4)

- CardLayout
  - 주요 생성자

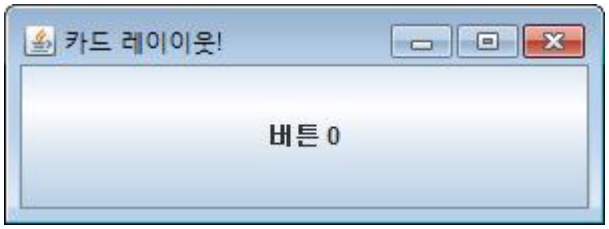
```
CardLayout()                          // 간격이 없는 CardLayout 객체를 생성한다.
CardLayout(int hgap, int vgap)        // 간격이 주어진 CardLayout 객체를 생성한다.
```

  - 주요 메서드(카드 컴포넌트 선택)

메서드	설명
void first(Container parent)	첫 번째 컴포넌트를 선택한다.
void next(Container parent)	다음 컴포넌트를 선택한다.
void previous(Container parent)	이전 컴포넌트를 선택한다.
void last(Container parent)	마지막 컴포넌트를 선택한다.

- 예제 : [예제 14-9]

```java
package sec04;

import java.awt.CardLayout;

public class CardLayoutDemo extends JFrame {
    CardLayout layout;

    public void rotate() {
        while (true) {
            try {
                Thread.sleep(500);
            } catch (Exception e) {
            }
            layout.next(this.getContentPane());
        }
    }

    CardLayoutDemo() {
        setTitle("카드 레이아웃!");
        layout = new CardLayout();
        setLayout(layout);

        add("0", new JButton("버튼 0"));
        add("1", new JButton("버튼 1"));
        add("2", new JButton("버튼 2"));

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 110);
        setVisible(true);
    }

    public static void main(String[] args) {
        new CardLayoutDemo().rotate();
    }
}
```



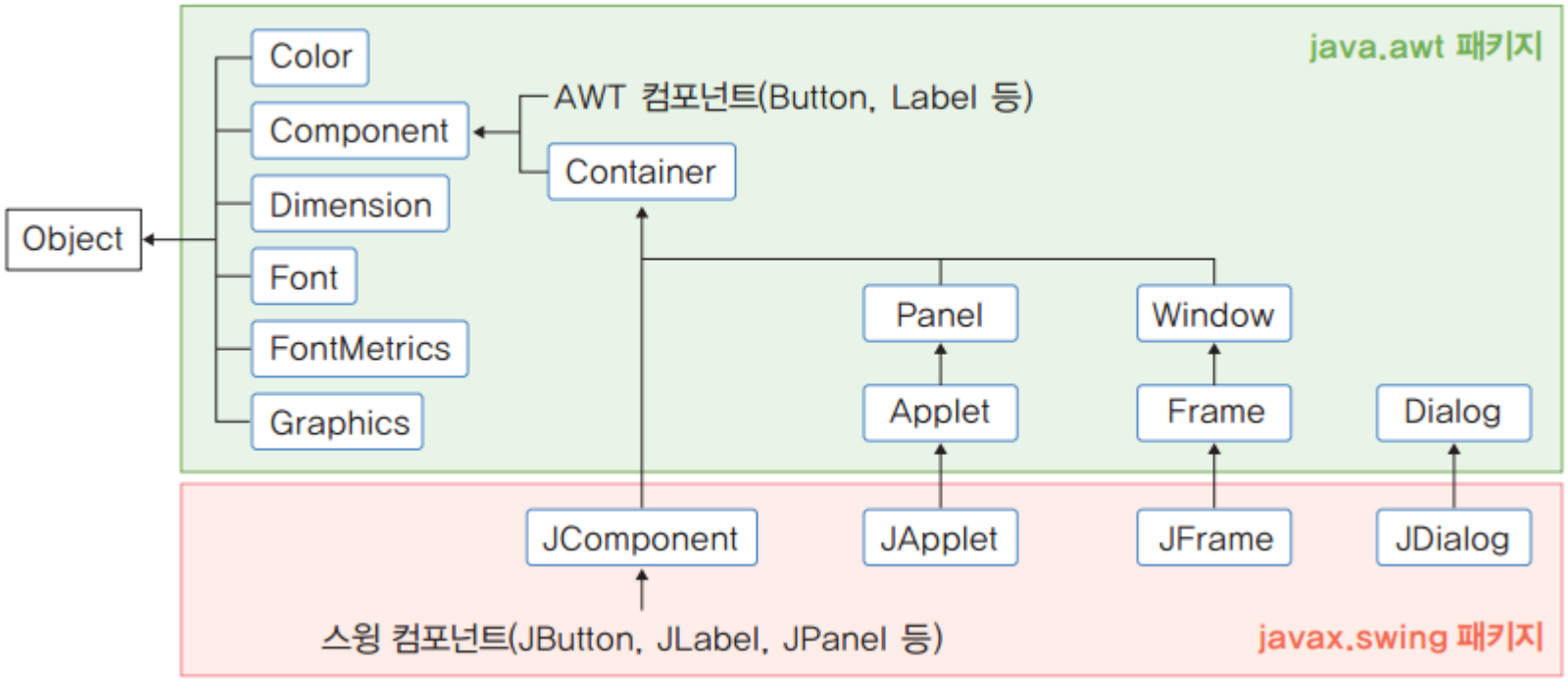
배치 관리자 없이 컴포넌트 배치

- 배치 관리자가 없을 때는 절대 좌표로 컴포넌트를 배치
- 플랫폼 환경이 다르거나 프레임의 크기가 변경되는 등 외부 원인으로 컴포넌트 크기와 위치가 개발자의 원래 의도와는 다르게 나타날 수 있다.
- 컴포넌트의 크기와 위치를 setSize( )와 setLocation( ) 또는 setBounds( ) 메서드를 이용해 개발자가 지정해야 함

- 예제 : [예제 14-10]

```java
package sec04;

import javax.swing.JButton;

public class NoLayoutDemo extends JFrame {
    NoLayoutDemo() {
        setTitle("절대 위치로 배치!");

        JPanel p = new JPanel();
        p.setLayout(null);

        JButton b1 = new JButton("B 1");
        b1.setBounds(10, 10, 60, 30);
        JButton b2 = new JButton("버튼 2");
        b2.setBounds(80, 20, 80, 25);
        JButton b3 = new JButton("Button 3");
        b3.setBounds(150, 30, 100, 30);
        p.add(b1);
        p.add(b2);
        p.add(b3);
        add(p);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 110);
        setVisible(true);
    }

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

GUI 컴포넌트의 상속 관계



AWT 패키지와 스윙 패키지

- 자주 사용하는 AWT 패키지
  - java.awt : AWT의 GUI 컴포넌트, 색상, 폰트, 그래픽, 레이아웃 배치 관리자 등 관련된 클래스를 포함
  - java.awt.event : AWT와 스윙의 이벤트 클래스, 각종 리스너 인터페이스, 어댑터 클래스를 포함
  - java.awt.color, java.awt.font, java.awt.image 등
- 스윙 패키지와 하부 패키지
  - javax.swing : 기본적인 GUI 관련 클래스를 포함
  - javax.swing.border : Border 인터페이스와 각종 구현 클래스를 포함
  - javax.swing.event : 스윙에 추가된 각종 이벤트 클래스와 리스너 인터페이스를 포함
  - javax.swing.tree : 스윙의 트리를 지원하는 인터페이스와 각종 구현 클래스를 포함
- GUI프로그래밍을 위한 헬퍼 클래스
  - 그래픽, 색상, 폰트, 레이아웃 배치 관리자 등은 java.awt 패키지의 가족
  - 따라서 이와 같은 클래스를 이용하려면 java.awt 패키지 임포트 필요

주요 컴포넌트(1)

- Component 클래스
  - 컴포넌트의 공통 속성과 크기, 모양, 색상, 폰트, 이동, 삭제, 이벤트 처리 등을 수행할 수 있는 메서드를 제공

메서드	설명
Color getBackground( )	컴포넌트의 배경색을 반환한다.
Color getForeground( )	컴포넌트의 전경색을 반환한다.
Graphics getGraphics( )	컴포넌트의 그래픽 컨텍스트를 반환한다.
String getName( )	컴포넌트의 이름을 반환한다.
Container getParent( )	컴포넌트를 포함하는 컨테이너를 반환한다.
Dimension getSize( )	컴포넌트의 크기를 반환한다.
void setBackground(Color c)	컴포넌트의 배경색을 설정한다.
void setEnabled(boolean b)	컴포넌트를 활성화 · 비활성화한다.
void setFont(Font f)	컴포넌트의 폰트를 설정한다.
void setForeground(Color c)	컴포넌트의 전경색을 설정한다.
void setLocation(Point p)	컴포넌트의 위치를 설정한다.
void setSize(Dimension d)	컴포넌트의 크기를 설정한다.
void setVisible(boolean b)	컴포넌트를 화면에 표시하거나 숨긴다.

주요 컴포넌트(2)

- Container 클래스
  - 다른 컨테이너 내부에 포함될 수 없는 최상위 컨테이너인 프레임, 다이얼로그, 애플릿
  - 다른 컨테이너에 포함될 수 있는 패널 또는 스크롤페인 등
  - 주요 메서드

메서드	설명
Component add(Component comp)	컨테이너에 컴포넌트를 부착한다.
Component add(Component comp, int index)	컨테이너에 컴포넌트를 명시한 위치에 부착한다.
void add(Component component, Object constraints)	두 번째 매개변수로 명시된 크기와 위치를 사용해 컨테이너에 컴포넌트를 부착한다.
Insets getInsets( )	컨테이너의 여백을 의미하는 인셋을 반환한다.
void remove(Component comp)	컴포넌트를 컨테이너에서 제거한다.
void remove(int index)	명시된 위치의 컴포넌트를 제거한다.
void setLayout(LayoutManager mgr)	컨테이너의 배치 관리자를 설정한다.

주요 컴포넌트(3)

- JComponent 클래스
  - 모든 스윙 컴포넌트의 부모 클래스
  - 주요 메서드

메서드	설명
Border getBorder( )	보더를 반환한다.
Dimension getPreferredSize( )	크기를 반환한다.
String getToolTipText( )	툴팁에 설정된 문자열을 반환한다.
void setBorder(Border border)	보더를 설정한다.
void setOpaque(boolean isOpaque)	투명 여부를 설정한다.
void setPreferredSize(Dimension preferredSize)	크기를 설정한다.
void setToolTipText(String text)	툴팁을 문자열로 설정한다.

주요 컴포넌트(4)

- JLabel
  - 이벤트와 관계없이 단순히 텍스트나 이미지를 표시
  - 주요 생성자

```
// 빈 레이블 컴포넌트를 생성한다.
JLabel()

// 이미지만 있는 레이블 컴포넌트를 생성한다.
JLabel(Icon image)

// 문자열만 있는 레이블 컴포넌트를 생성한다.
JLabel(String text)

// 문자열과 이미지가 둘 다 있는 레이블 컴포넌트를 생성한다.
// horizontalAlignment에 LEFT, RIGHT, CENTER 등 상수를 사용해 정렬할 수 있다.
JLabel(String text, Icon icon, int horizontalAlignment)
```

주요 컴포넌트(5)

- JLabel
  - 주요 메서드

메서드	설명
Icon getIcon( )	레이블이 가진 아이콘을 반환한다.
String getText( )	레이블이 가진 문자열을 반환한다.
void setIcon(Icon icon)	레이블에 명시된 아이콘을 설정한다.
void setText(String text)	레이블에 명시된 문자열을 설정한다.

주요 컴포넌트(6)

- JButton
  - 사용자가 직접 작동해서 제어할 수 있는 컴포넌트 중 하나
  - 사용자가 클릭하면 ActionEvent를 발생
  - 주요 생성자

```
JButton(Icon icon)                  // 이미지만 있는 버튼을 생성한다.
JButton(String text)                // 텍스트만 있는 버튼을 생성한다.
JButton(String text, Icon icon)     // 텍스트와 이미지가 있는 버튼을 생성한다.
```

  - 주요 메서드

메서드	설명
Icon getIcon( )	설정된 이미지를 반환한다.
String getText( )	설정된 문자열을 반환한다.
void setIcon(Icon icon)	버튼의 이미지를 설정한다.
void setMnemonic(char mnemonic)	단축키 문자를 설정한다.
void setText(String text)	버튼의 문자열을 설정한다.

주요 컴포넌트(7)

- JTextField
  - 한 행짜리 문자열 입력 창을 만드는 컴포넌트로, JTextArea 및 JEditPane과 함께 JTextComponent의 자식 클래스
  - 주요 생성자

```
// 주어진 열의 개수만큼 텍스트 필드를 생성한다.
JTextField(int columns)

// 초기 문자열이 있는 텍스트 필드를 생성한다.
JTextField(String text)

// 초기 문자열로 주어진 열의 개수만큼 텍스트 필드를 생성한다.
JTextField(String text, int columns)
```

주요 컴포넌트(8)

- JTextField
  - 주요 메서드

메서드	설명
int getCaretPosition( )	문자열이 삽입될 캐럿의 위치를 반환한다.
int getColumns( )	텍스트 필드에 설정된 열의 개수를 반환한다.
String getText( )	텍스트 필드에 포함된 문자열을 반환한다.
void setEditable(boolean b)	텍스트 필드의 편집 여부를 설정한다.
void setColumns(int columns)	텍스트 필드의 열 개수를 설정한다.
void setFont(Font f)	텍스트 필드의 폰트를 설정한다.
void setText(String text)	주어진 문자열로 설정한다.

주요 컴포넌트(9)

- JTextArea
  - 여러 행에 걸쳐 문자열을 입력하거나 편집할 수 있는 스윙 컴포넌트
  - 사용자가 문자열을 입력한 후 엔터 키를 누르면 ActionEvent가 발생
  - 주요 생성자

```
// 텍스트 영역을 생성한다.
JTextArea()

// 초기 문자열이 있는 텍스트 영역을 생성한다.
JTextArea(String text)

// 주어진 행과 열이 있는 텍스트 영역을 생성한다.
JTextArea(int rows, int columns)

// 초기 문자열로 주어진 행과 열이 있는 텍스트 영역을 생성한다.
JTextArea(String text, int rows, int columns)
```

주요 컴포넌트(10)

- JTextArea
  - 주요 메서드

메서드	설명
void append(String str)	주어진 문자열을 문서 끝에 추가한다.
int getLineCount( )	행 개수를 반환한다.
String getText( )	텍스트 영역에 포함된 문자열을 반환한다.
void insert(String str, int pos)	주어진 문자열을 pos 위치에 삽입한다.
void replaceRange(String str, int start, int end)	주어진 문자열로 start와 end 사이의 문자열을 교체한다.
void setEditable(boolean b)	텍스트 영역의 편집 여부를 설정한다.
void setFont(Font f)	주어진 폰트로 설정한다.
void setRows(int rows)	주어진 행 개수로 설정한다.

주요 컴포넌트(11)

- JComboBox
  - 다수의 항목 중에 하나를 선택하며, 컴포넌트에 텍스트와 이미지를 추가 가능
  - 항목을 선택하면 ActionEvent 발생, 항목을 변경하면 ItemEvent 발생
  - 주요 생성자

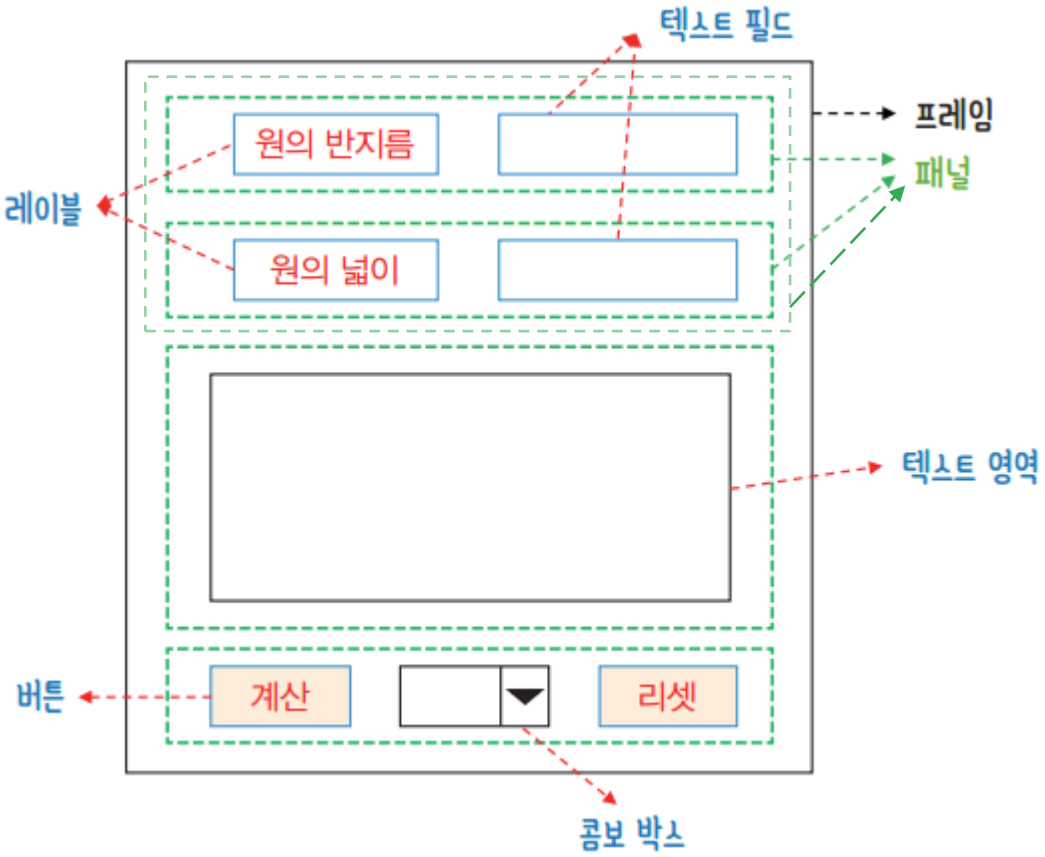
```
JComboBox()                          // 비어 있는 JComboBox 객체를 생성한다.
JComboBox(E[] items)                 // 배열을 사용해 JComboBox 객체를 생성한다.
JComboBox(Vector<E> items)           // 벡터를 사용해 JComboBox 객체를 생성한다.
```

  - 주요 메서드

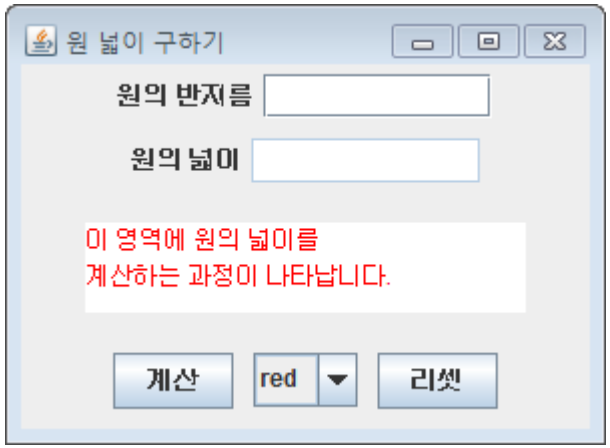
메서드	설명
void addItem(E item)	지정한 항목을 목록에 추가한다.
E getItemAt(int index)	지정한 인덱스의 항목을 목록에서 반환한다.
void insertItemAt(E item, int index)	지정한 항목을 지정한 인덱스에 추가한다.
void removeItem(Object anObject)	지정한 항목을 목록에서 제거한다.
void removeItemAt(int anIndex)	지정한 인덱스의 항목을 목록에서 제거한다.

스윙 컴포넌트 응용

- 예 : 다음과 같은 외형을 구성
  - BorderLayout 배치 관리자로 북쪽, 중앙, 남쪽으로 분리해서 표현하면 편리



- 예제 : [예제 14-14]




```
1  package sec05;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.FlowLayout;
6  import java.awt.GridLayout;
7  import javax.swing.JButton;
8  import javax.swing.JComboBox;
9  import javax.swing.JFrame;
10 import javax.swing.JLabel;
11 import javax.swing.JPanel;
12 import javax.swing.JTextArea;
13 import javax.swing.JTextField;
14
15 public class ComponentDemo extends JFrame {
16     ComponentDemo() {
17         setTitle("원 넓이 구하기");
18
19         setLayout(new BorderLayout(10, 10));
20         showNorth();
21         showCenter();
22         showSouth();
23
24         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         setSize(300, 220);
26         setVisible(true);
27     }
```

```
29      void showNorth() {
30          JPanel p1 = new JPanel();
31          JPanel p2 = new JPanel();
32          JPanel panel = new JPanel(new GridLayout(2, 0));
33
34          JLabel l1 = new JLabel("원의 반지름");
35          JLabel l2 = new JLabel("원의 넓이");
36
37          JTextField t1 = new JTextField(10);
38          JTextField t2 = new JTextField(10);
39          t2.setEnabled(false);
40
41          p1.add(l1);
42          p1.add(t1);
43          p2.add(l2);
44          p2.add(t2);
45          panel.add(p1);
46          panel.add(p2);
47
48          add(panel, BorderLayout.NORTH);
49      }
50
51      void showCenter() {
52          JPanel panel = new JPanel();
53
54          JTextArea area = new JTextArea(30, 20);
55          area.setText("이 영역에 원의 넓이를\n계산하는 과정이 나타납니다.");
56          area.setEditable(false);
57          area.setForeground(Color.RED);
58
59          panel.add(area);
60
61          add(panel, BorderLayout.CENTER);
62      }
```

```
63
64      void showSouth() {
65          String[] color = { "red", "blue" };
66
67          JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));
68
69          JButton cal = new JButton("계산");
70          JComboBox<String> cb = new JComboBox<>(color);
71          JButton reset = new JButton("리셋");
72
73          panel.add(cal);
74          panel.add(cb);
75          panel.add(reset);
76
77          add(panel, BorderLayout.SOUTH);
78      }
79
80      public static void main(String[] args) {
81          new ComponentDemo();
82      }
83  }
```