

GUI 프로그래밍 소개

CUI와 GUI

그래픽의 사전적 의미는 상품화, 정보 제공, 엔터테인먼트 등을 벽, 캔버스, 컴퓨터 화면, 종이 같은 표면에 나타내는 시각적 표현이다.

예: 사진, 드로잉, 그래프, 다이어그램, 이미지 등이 있다.

그래픽은 문자나 숫자보다 더 빠르고 쉽게 정보를 전달할 수 있기 때문에 컴퓨터 분야에서 매우 중요하다.

비교적 최근에 출시된 Java는 초기 버전부터 그래픽을 고려해서 설계한다.

사용자와 컴퓨터의 상호작용 방식이다.

```
Current date is Tue 1-01-1980  
Enter new date:  
Current time is 7:40:27.13  
Enter new time:  
  
The IBM Personal Computer DOS  
Version 1.10 (C)Copyright IBM Corp 1981, 1982  
  
A:\>dir /w  
COMMAND.COM   FORMAT.COM    CHDIR.COM    SVS.COM    DISCOPY.COM  
DISKCOMP.COM   COMP.COM    EXE2BIN.COM   MDIR.COM    EDLIN.COM  
DEBUG.COM     LINK.EXE    BASIC.COM    BASICA.COM  ARK.DAT  
SAMPLES.BAS    MORTGAGE.BAS  COLUMNAR.BAS CALENDAR.BAS MUSIC.BAS  
SERKEY.BAS    CIRCLE.BAS   PIEDMNT.BAS  SPACE.BAS  WALL.BAS  
COMM.BAS  
26 File(s)  
A:\>dir command.com  
COMMAND.COM  4950  5-07-02  12:00p  
1 File(s)  
A:\>
```

(a) CUI 방식



(b) GUI 방식

리눅스 가비올에서 많이 사용함.

AWT (Abstract Windows Toolkit)

운영체제가 제공하는 네이티브 UI 컴포넌트를 이용하는 자바 라이브러리이다.
중량 컴포넌트라고 하며, 운영체제에 따라 외형이 다르다.(운영체제의 자원을 많이 사용한다.)

Swing(스윙) : AWT 포함

운영체제의 도움을 받지 않고 순수하게 자바로 작성되어 있기 때문에 스윙 컴포넌트를 경량 컴포넌트라고 한다. (자신만의 UI를 사용한다.)

모든 스윙 컴포넌트는 AWT컴포넌트와 완전히 호환된다. 즉, AWT의 클래스를 상속받아 사용한다.

- 컴포넌트 : 부품이란 뜻으로 프로그램을 구성하는 부품이며 클래스로 이루어져 있다.

Java AWT	Java Swing (이걸 사용할 것!)
AWT는 플랫폼에 의존적이다. (운영체제에 의존적임)	스윙은 플랫폼에 독립적이다. (어떤 운영체제든 동일한 모습)
AWT 컴포넌트는 용량이 크다. <u>중량적</u>	스윙 컴포넌트는 용량이 가볍다. <u>경량적</u>
AWT 교체할 수 있는 룩앤플(look and feel)을 지원하지 않는다.	스윙은 교체할 수 있는 룩앤플(look and feel)을 지원한다.
컴포넌트의 개수가 적다.	컴포넌트의 개수가 많다.

룩앤플 : 동일성 있는 디자인

GUI 작성 절차

1. 컨테이너를 생성한다. (자바에서는 5가지 정도의 컨테이너를 제공한다.)
2. 컴포넌트를 추가한다.

스윙을 이용한 GUI 기초

스윙의 특징

룩앤플

풍선 도움말 (어떤 개체에 마우스를 올리면 알려주는 것)

더블 버퍼링 (스윙에서는 자동으로 주어지는 기능)

MVC 모델 (Model, View(보이는 인터페이스), Controller(View를 통해 들여오는 기능을 제어)) -> 최신 기법이다.

이미지 아이콘 (AWT는 문자열만 가능했다.)

보더

컴포넌트와 컨테이너의 개념

- 컴포넌트

버튼, 레이블, 텍스트 필드 등 GUI를 작성하는 기본적인 빌딩 블록을 의미한다.
사용자 인터페이스를 생성하는 객체로, 윈도우 시스템에서는 컨트롤에 해당한다.

- 컨테이너

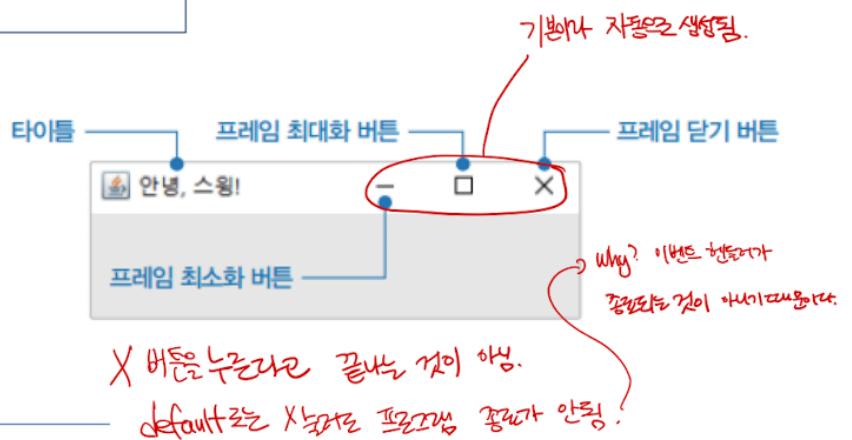
컴포넌트를 부착하는 특수한 컴포넌트를 의미한다.
컴포넌트를 부착할 수 있는 프레임이나 패널 등이 대표적인 컨테이너 클래스다.



스윙 기반의 GUI 프로그램 예제

```
import javax.swing.JFrame;

public class HelloSwingDemo {
    public static void main(String[] args) {
        JFrame f = new JFrame(); ← 프레임 생성.
        f.setTitle("안녕, 스윙"); ← 제목
        f.setSize(300, 100); ← 프레임 크기
        f.setVisible(true); ← true를 해야 화면에 뜬다
    }
}
```



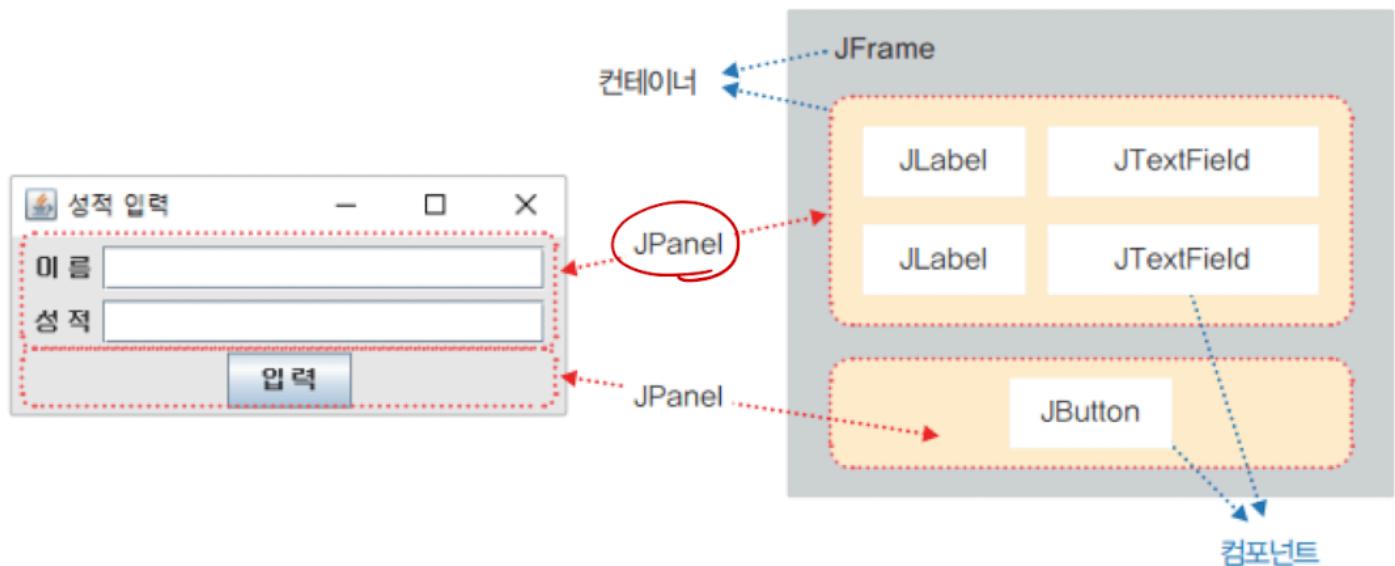
컨테이너 생성과 컴포넌트 추가

컨테이너와 컴포넌트

컨테이너는 내부의 배치 관리자를 사용해 컴포넌트 위치를 결정하고 자신에게 부착한다.

스윙 애플리케이션을 작성하려면 스윙 애플리케이션의 최상위 컨테이너인 프레임을 생성해야 된다.

프레임 생성 후 컴포넌트를 부착하거나 다른 컨테이너인 패널을 부착하여 컴포넌트를 추가한다.



JFrame

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

복잡한 구조로 구성되어 있지만, 개발자가 자주 접하는 부분은 메뉴바와 컨텐트페인이다.

- 생성자

```
JFrame()           // 타이틀이 없는 JFrame 객체를 생성한다. (setTitle()을 이용한다.)  
JFrame(String title) // 명시된 타이틀을 가진 JFrame 객체를 생성한다. (처음부터 만들어 주
```

는 것이다.)

- JFrame 클래스에서 사용할 수 있는 주요 메소드

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. ← 이전 사용하면서 X버튼으로 종료할 수 있음.
void setIconImage(Image image)	윈도우 아이콘을 설정한다.
void setLayout(LayoutManager manager)	윈도우의 배치 관리자를 설정한다.
void setJMenuBar(JMenuBar menubar)	프레임의 메뉴바를 주어진 메뉴바로 지정한다.

이 외에도 Component 클래스가 제공하는 add(), Window 클래스가 제공하는 setVisible(), Frame 클래스가 제공하는 setResizable() 등도 자주 사용된다.

추가..

`add(component)` : 프레임에 컴포넌트를 추가한다. (많이 쓰인다.)

`setLocation(x, y), 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. (true : 창 크기 조절 가능)

- JFrame 클래스에서 사용할 수 있는 상수

상수	설명
DISPOSE_ON_CLOSE	종료할 때 모든 자원을 반납한다.
DO_NOTHING_ON_CLOSE	종료할 때 아무 일도 하지 않는다. ← 거의 사용하지 않음.
EXIT_ON_CLOSE	종료할 때 애플리케이션도 강제로 종료한다.
HIDE_ON_CLOSE	종료할 때 창을 숨긴다. ← minimize와 비슷(동일?)

`setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` ← X(HEC) 누르면 종료!

위에서 정한 상수 사용.

- ### ● 예제

```

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();
    }
}

```

→ 2개의 Class가 필요함.

*이 방식은 주로 사용방법.

```

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();
    }
}

```

→ 1개의 Class로 만들기 가능



프레임에 컨포넌트 추가

스윙 컨포넌트를 프레임에 부착하려면 Container 클래스가 제공하는 add() 메소드를 호출한다.

```

import javax.swing.JButton;
import javax.swing.JFrame;

public class JFrame3Demo extends JFrame {
    JFrame3Demo() {
        setTitle("안녕, 스윙!"); ← 제목지정

        JButton b = new JButton("버튼"); ← 컨포넌트 생성중. (버튼 객체 만들기)
        add(b); ← 버튼을 추가시켜 중.
        ↑ 버튼이 이미 보고 사용.
        () 빈 생성자를 만들어도됨.

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); ← X 버튼을 클릭
        setSize(300, 100);
        setVisible(true);
    }

    public static void main(String[] args) {
        new JFrame3Demo();
    }
}

```



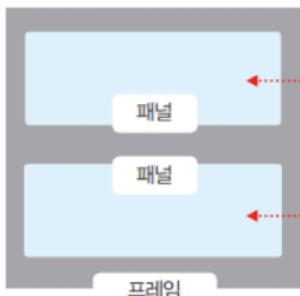
↳ 전체가 버튼임.

Layout을 이용해 자정을 개혁해야됨.

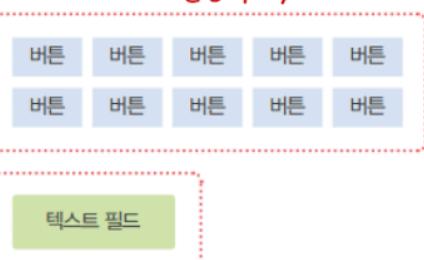
패널로 프레임에 컨포넌트 추가

패널로 프레임에 컴포넌트 추가

- 예



최상위 패널 : JFrame 그 아래 다른 컴포넌트를 추가!



← 패널이라는 컨테이너를 사용해
묶어둔다. (구분하기 쉬워)
⇒ 유지보수

// 플로 레이아웃과 더블 버퍼를 가진 JPanel 객체를 생성한다.

JPanel()
↳ 기본FlowLayout 레이아웃을 사용.

// 플로 레이아웃과 명시된 더블 버퍼 전략을 가진 JPanel 객체를 생성한다.

JPanel(boolean isDoubleBuffered)

// 명시된 레이아웃과 더블 버퍼를 가진 JPanel 객체를 생성한다.

JPanel(LayoutManager manager)

// 명시된 레이아웃과 더블 버퍼 전략을 가진 JPanel 객체를 생성한다.

JPanel(LayoutManager manager, boolean isDoubleBuffered)

■ 패널로 프레임에 컴포넌트 추가

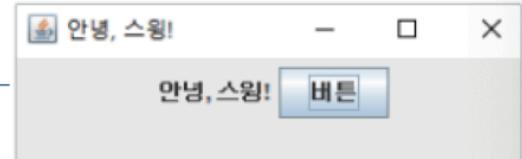
- 예제 : [sec03/JFrame4Demo](#)

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class JFrame4Demo extends JFrame {
    JFrame4Demo() {
        setTitle("안녕, 스윙!");
        JPanel p = new JPanel(); ← 패널 만들어줌.
        JLabel l = new JLabel("안녕, 스윙!"); ← Label로 글자임.
        JButton b = new JButton("버튼"); ← 버튼생성.
        p.add(l); ← 패널에 Label 추가
        p.add(b); ← 패널에 button 추가
        add(p); ← 패널 추가

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 100);
        // pack(); ← 패널의 최대한의 크기로 만들려음.
        setVisible(true);
    }

    public static void main(String[] args) {
        new JFrame4Demo();
    }
}
```



setSize() 메서드 호출



pack() 메서드 호출

기본의 border
(꽉 치우는 것)

기본의 Flow
(왼쪽부터 차례로)

주의! JPanel p = new JPanel(); ← 패널 만들어줌.
JLabel l = new JLabel("안녕, 스윙!"); ← Label로 글자임.
JButton b = new JButton("버튼"); ← 버튼생성.
p.add(l); ← 패널에 Label 추가
p.add(b); ← 패널에 button 추가
add(p); ← 패널 추가

컴포넌트 배치

배치 관리자의 역할(Layout Manager)

부착할 컴포넌트 위치를 결정해서 적절히 배치하며, 컨테이너의 크기가 변하면 컴포넌트를 재배치한다.

배치 관리자의 종류

FlowLayout

BorderLayout

GridLayout

CardLayout

... 등등

- 생성자를 이용하는 방법

```
JPanel panel = new JPanel(new BorderLayout());
```

- setLayout() 메소드를 이용

```
panel.setLayout(new FlowLayout());
```

- 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	
JFrame	BorderLayout
JWindow	
JApplet	
JPanel	FlowLayout

FlowLayout 배치 관리자

```

FlowLayout()           // 중앙 정렬, 5픽셀 간격의 FlowLayout 객체를
생성한다.
FlowLayout(int align) // 명시된 정렬, 5픽셀 간격의 FlowLayout 객체를
생성한다.
FlowLayout(int align, int hgap, int vgap) // 명시된 정렬 및 간격의 FlowLayout 객체를 생
성한다.

```

단점 : 창 크기에 따라 배치가 달라진다.

- 정렬

```

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

```

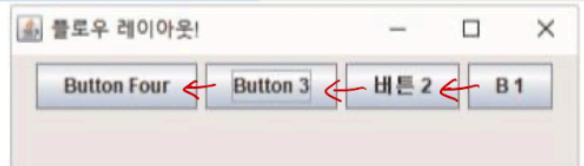
- 배치 방향 변경

```
p.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
```

- 예제

■ FlowLayout 배치 관리자

- 예제 : [sec04/FlowLayoutDemo](#)



```
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);  
}
```

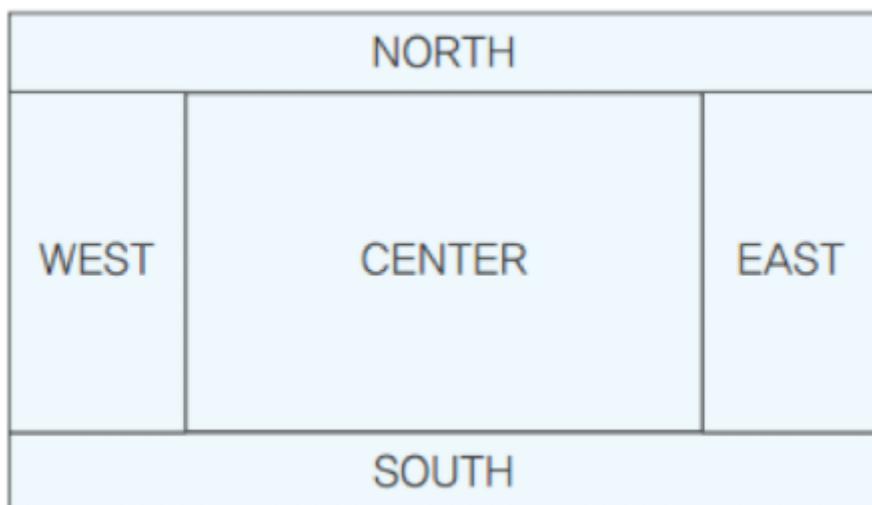


창이 작아지면
아래로 내려감.

원쪽부터 넣음.

이기 때문에 원쪽부터 → 순서

BorderLayout 배치 관리자



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

- 영역에 컴포넌트를 추가하기 위한 메소드

앞에 위치한 경우 사용

`Component add(Component comp, int index)`

컨테이너에 부착시킬 컴포넌트

뒤에 위치한 경우 대문자 문자열 사용

`Component add(String name, Component comp)`

BorderLayout의 위치

동 : BorderLayout.EAST

서 : BorderLayout.WEST

남 : BorderLayout.SOUTH

북 : BorderLayout.NORTH

중앙 : BorderLayout.CENTER

BorderLayout의 위치

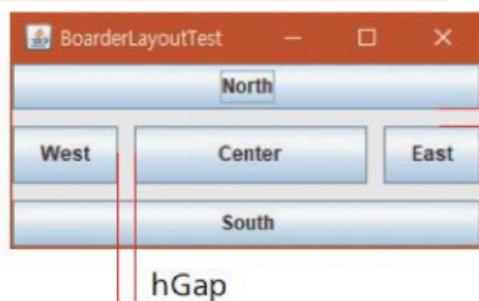
동 : "East"

서 : "West"

남 : "South"

북 : "North"

중앙 : "Center"



- `BorderLayout()`
- `BorderLayout(int hGap, int vGap)`

- 예제

```
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);  
}
```

GirdLayout 배치 관리자

GridLayout 객체를 생성할 때 행과 열의 개수를 0이상의 정수로 명시한다.
행이나 열의 값이 0이면 필요한 만큼의 행이나 열을 생성한다. 그러나 행과 열의 개수로
동시에 0은 사용할 수 없다.

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



- 예제

```

GridLayoutDemo() {
    setTitle("그리드 레이아웃!");
    setLayout(new GridLayout(0, 3)); (잘 사용하지 않음)

    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);
}

```



CardLayout 배치 관리자

<code>CardLayout()</code>	// 간격이 없는 CardLayout 객체를 생성한다.
<code>CardLayout(int hgap, int vgap)</code>	// 간격이 주어진 CardLayout 객체를 생성한다.

- CardLayout 클래스가 제공하는 메소드

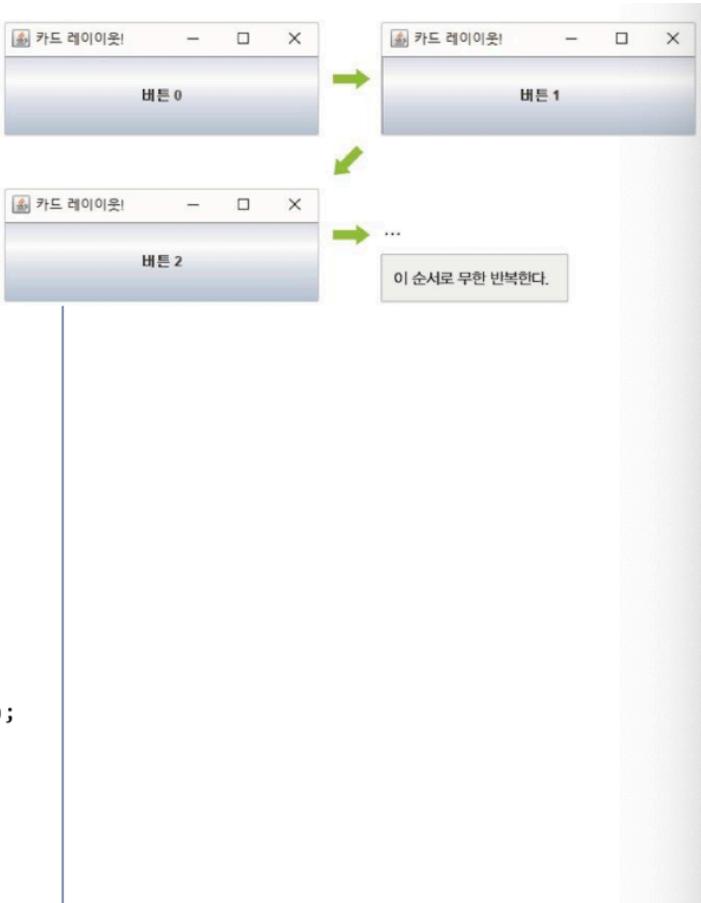
메서드	설명
<code>void first(Container parent)</code>	첫 번째 컴포넌트를 선택한다.
<code>void next(Container parent)</code>	다음 컴포넌트를 선택한다.
<code>void previous(Container parent)</code>	이전 컴포넌트를 선택한다.
<code>void last(Container parent)</code>	마지막 컴포넌트를 선택한다.

- 예제

■ CardLayout 배치 관리자

- 예제 : [sec04/CardLayoutDemo](#)

```
public class CardLayoutDemo extends JFrame {  
    CardLayout layout;  
  
    public void rotate() {  
        while (true) {  
            try {  
                Thread.sleep(500); ← 0.5초  
            } 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

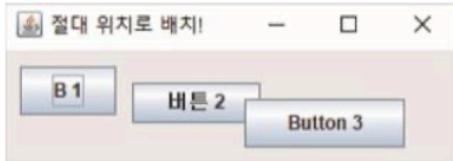
배치 관리자가 없을 때는 절대 좌표로 컴포넌트를 배치한다.

플랫폼 환경이 다르거나 프레임의 크기가 변경되는 등 외부 원인으로 컴포넌트 크기와 위치가 개발자의 원래 의도와는 다르게 나타날 수 있다.

컴포넌트의 크기와 위치를 `setSize()`, `setLocation()`, `setBounds()` 메소드를 이용해 개발자가 지정하는 번거로움을 감수한다.

- 예제

- 예제 : [sec04/NoLayoutDemo](#)



```
NoLayoutDemo() {  
    setTitle( "절대 위치로 배치! " );  
  
    JPanel p = new JPanel();  
    p.setLayout(null);  
  
    JButton b1 = new JButton( "B 1" );  
    b1.setBounds(10, 10, 60, 30); ← 간격을 적어줌.  
    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);  
}
```

스윙 비주얼 디자이너 : WindowBuilder

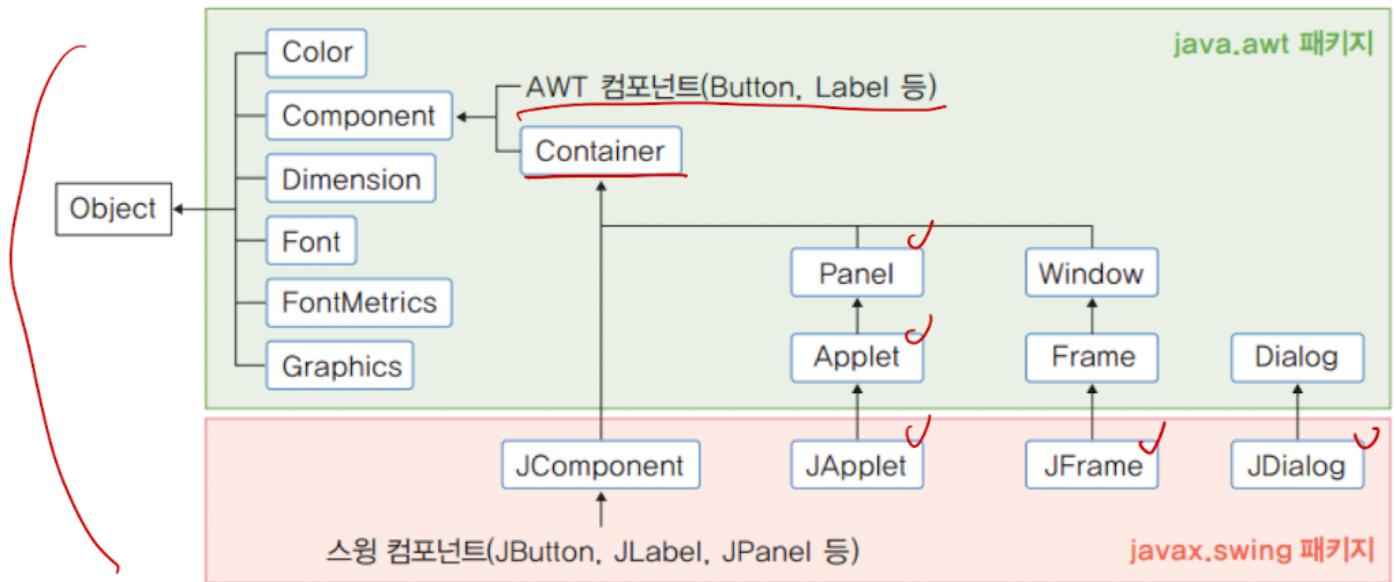
주요 컴포넌트

AWT 패키지와 스윙 패키지

GUI 프로그래밍을 위한 헬퍼 클래스

-> 그래픽, 색상, 폰트, 레이아웃 배치 관리자 등은 java.awt 패키지의 가족이다. 따라서 이와 같은 클래스를 이용하려면 java.awt 패키지를 불러와야 한다.

GUI 컴포넌트의 상속 관계



자주 사용하는 AWT 패키지

`java.awt` : AWT의 GUI 컴포넌트, 색상, 폰트, 그래픽, 레이아웃 배치 관리자 등 관련된 클래스를 포함한다.

`java.awt.event` : AWT와 스윙의 이벤트 클래스, 각종 리스너 인터페이스, 어댑터 클래스를 포함한다.

java.awt.image 등

스윙 패키지와 하부 패키지

`javax.swing` : 기본적인 GUI 관련 클래스를 포함한다.

`javax.swing.border` : Border 인터페이스와 각종 구현 클래스를 포함한다.

`javax.swing.event` : 스윙에 추가된 각종 이벤트 클래스와 리스너 인터페이스를 포함한다.

`javax.swing.tree` : 스윙의 트리를 지원하는 인터페이스와 각종 구현 클래스를 포함한다.

Component 클래스

컴포넌트의 공통 속성과 크기, 모양, 색상, 폰트, 이동, 삭제, 이벤트 처리 등을 수행할 수 있는 메소드를 제공한다.

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.

get
반환

set
설정

Container 클래스

다른 컨테이너 내부에 포함될 수 없는 최상위 컨테이너인 프레임, 다이얼로그, 애플릿이 있다.

다른 컨테이너에 포함될 수 있는 패널 또는 스크롤 페인 등이 있다.

메서드	설명
Component add(Component comp)	컨테이너에 컴포넌트를 부착한다. 부착 최상위 컨테이너
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.

JComponent 클래스

모든 스윙 컴포넌트의 부모 클래스이다.



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 스윙 컴포넌트

JLabel

텍스트를 표시할 수 있는 공간이다.

JLabel

편집이 불가능한 → 완전 불가능하지 않은. (개별자 입장)

● 이벤트와 관계없이 단순히 텍스트나 이미지를 표시
불가능 하는 것은 사용자의 입장에서!

// 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)

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.

JButton

클릭되면 어떤 동작을 실행하는 버튼이다.

사용자가 직접 작동해서 제어할 수 있는 컴포넌트 중 하나이다.

사용자가 클릭하면 ActionEvent를 발생한다.

JButton(Icon icon) // 이미지만 있는 버튼을 생성한다.

JButton(String text) // 텍스트만 있는 버튼을 생성한다.

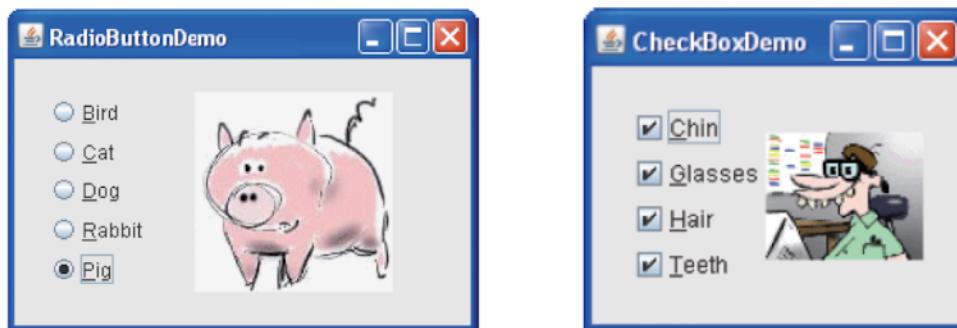
JButton(String text, Icon icon) // 텍스트와 이미지가 있는 버튼을 생성한다.

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.

- 버튼은 사용자가 클릭했을 경우, 이벤트를 발생하여 원하는 동작을 하게 하는데 이용된다



- **JButton** – 가장 일반적인 버튼이다.
- **JCheckBox** – 체크박스 버튼
- **JRadioButton** – 라디오 버튼으로 그룹 중의 하나의 버튼만 체크할 수 있다.
- **JToggleButton** – 2가지 상태를 가지고 토글이 가능한 버튼이다.



JTextField

사용자가 한 줄의 텍스트를 입력할 수 있는 공간이다.

JTextArea 및 JEditPane과 함께 JTextComponent의 자식 클래스이다.

```

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

JTextArea

여러 행에 걸쳐 문자열을 입력하거나 편집할 수 있는 스윙 컴포넌트이다.
사용자가 문자열을 입력한 후 엔터 키를 누르면 ActionEvent가 발생한다.

```

JTextArea()                      // 텍스트 영역을 생성한다.
JTextArea(String text)           // 초기 문자열이 있는 텍스트 필드를 생성한
다.
JTextArea(int rows, int columns) // 주어진 행과 열이 있는 텍스트 필드를 생
성한다.
  
```

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

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.

JComboBox

다수의 항목 중에 하나를 선택하며, 컴포넌트에 텍스트와 이미지를 추가할 수 있다. 항목을 선택하면 ActionEvent가 발생하고 항목을 변경하면 ItemEvent가 발생한다.

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

메서드	설명
void add(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.