

9

자바 GUI 기초, AWT와 스윙(Swing)

자바의 GUI(Graphical User Interface)



- GUI 목적
  - 그래픽 이용, 사용자에게 이해하기 쉬운 모양으로 정보 제공
  - 사용자는 마우스나 키보드를 이용하여 쉽게 입력
- 자바 GUI 특징
  - 강력한 GUI 컴포넌트 제공, 쉬운 GUI 프로그래밍
- 자바의 GUI 프로그래밍 방법
  - GUI 컴포넌트와 그래픽 이용
    - AWT 패키지와 Swing 패키지에 제공되는 메카니즘 이용
    - AWT - java.awt 패키지
    - Swing - javax.swing 패키지

AWT와 Swing 패키지



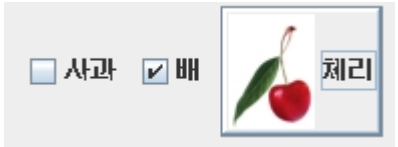
- AWT(Abstract Windowing Toolkit)
  - 자바가 처음 나왔을 때 함께 배포된 GUI 라이브러리
  - java.awt 패키지
  - AWT 컴포넌트는 중량 컴포넌트(Heavy weight components)
    - AWT 컴포넌트는 native(peer) 운영체제의 GUI 컴포넌트의 도움을 받아 작동
    - 운영체제에 많은 부담. 오히려 처리 속도는 빠름
- Swing(스윙)
  - AWT 기술을 기반으로 순수 자바 언어로 만들어진 라이브러리
    - 모든 AWT 기능 + 추가된 풍부하고 화려한 고급 컴포넌트
    - AWT 컴포넌트에 J자가 덧붙여진 이름의 클래스
    - 그 외 J 자로 시작하는 클래스
  - javax.swing 패키지
  - Swing 컴포넌트는 경량 컴포넌트(Light weight components)
    - native(peer) 운영체제에 의존하지 않음

스윙 컴포넌트 예시





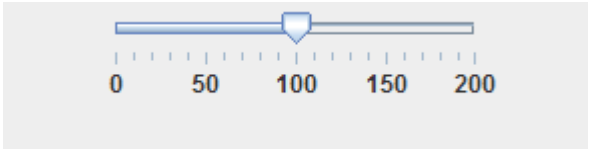
JButton



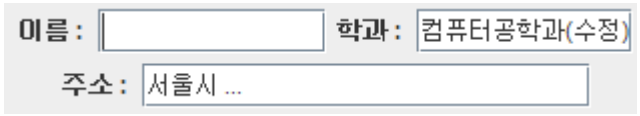
JCheckBox



JRadioButton



JSlider



JTextField



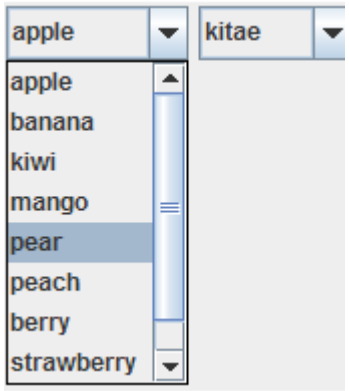
JPasswordField



JSpinner



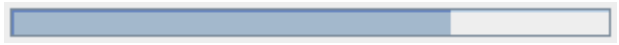
JTextArea



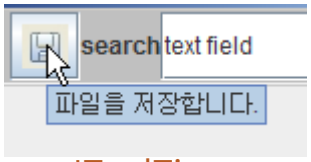
JComboBox



JList



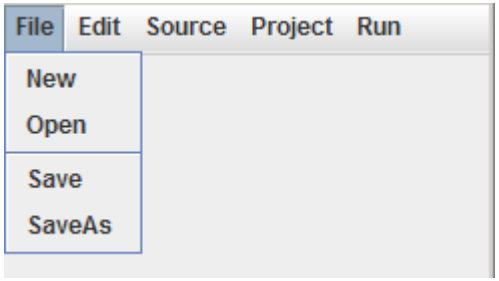
JProgressBar



JToolTip



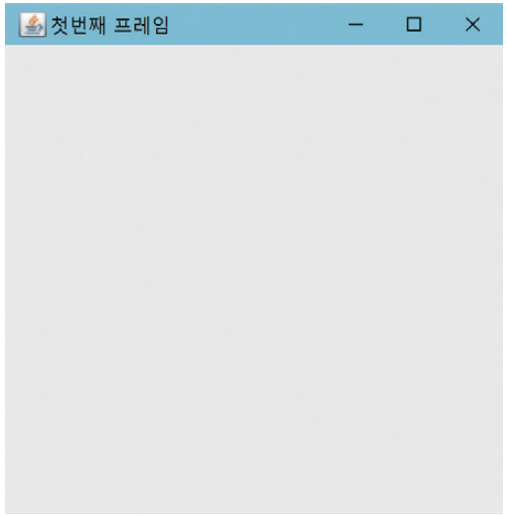
JScrollPane



JMenu



JDialog



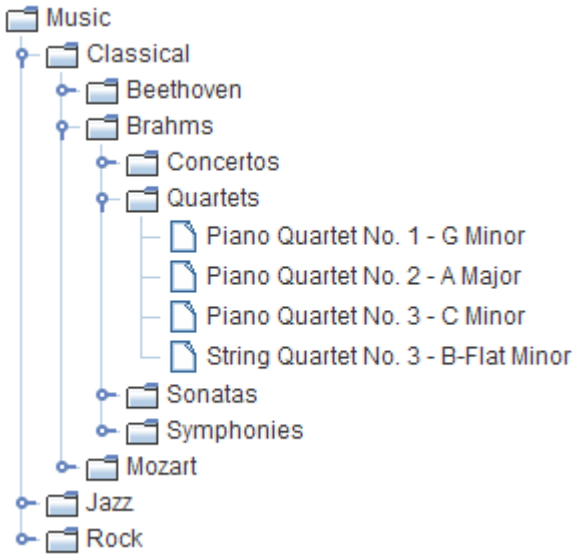
JFrame



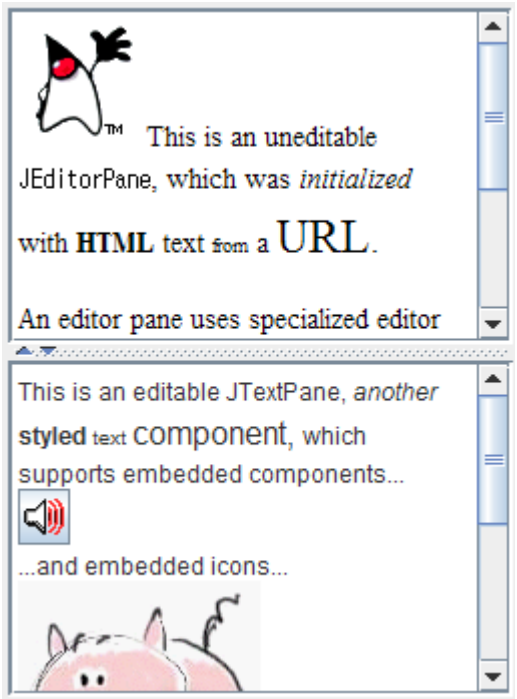
JColorChooser

First Name	Last Name	Favorite Color	Favorite Movie	Favorite Number	Favorite Food
Mike	Albers	Green	Brazil	44	
Mark	Andrews	Blue	Curse of the Dem...	3	
Brian	Beck	Black	The Blues Brothers	2.718	
Lara	Bunni	Red	Airplane (the whol...	15	
Roger	Brinkley	Blue	The Man Who Kn...	13	
Brent	Christian	Black	Blade Runner (Dir...	23	

JTable



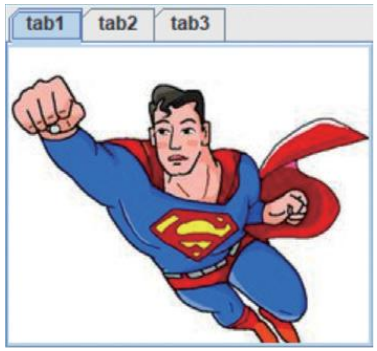
JTree



JEditorPane and JTextPane



JToolBar



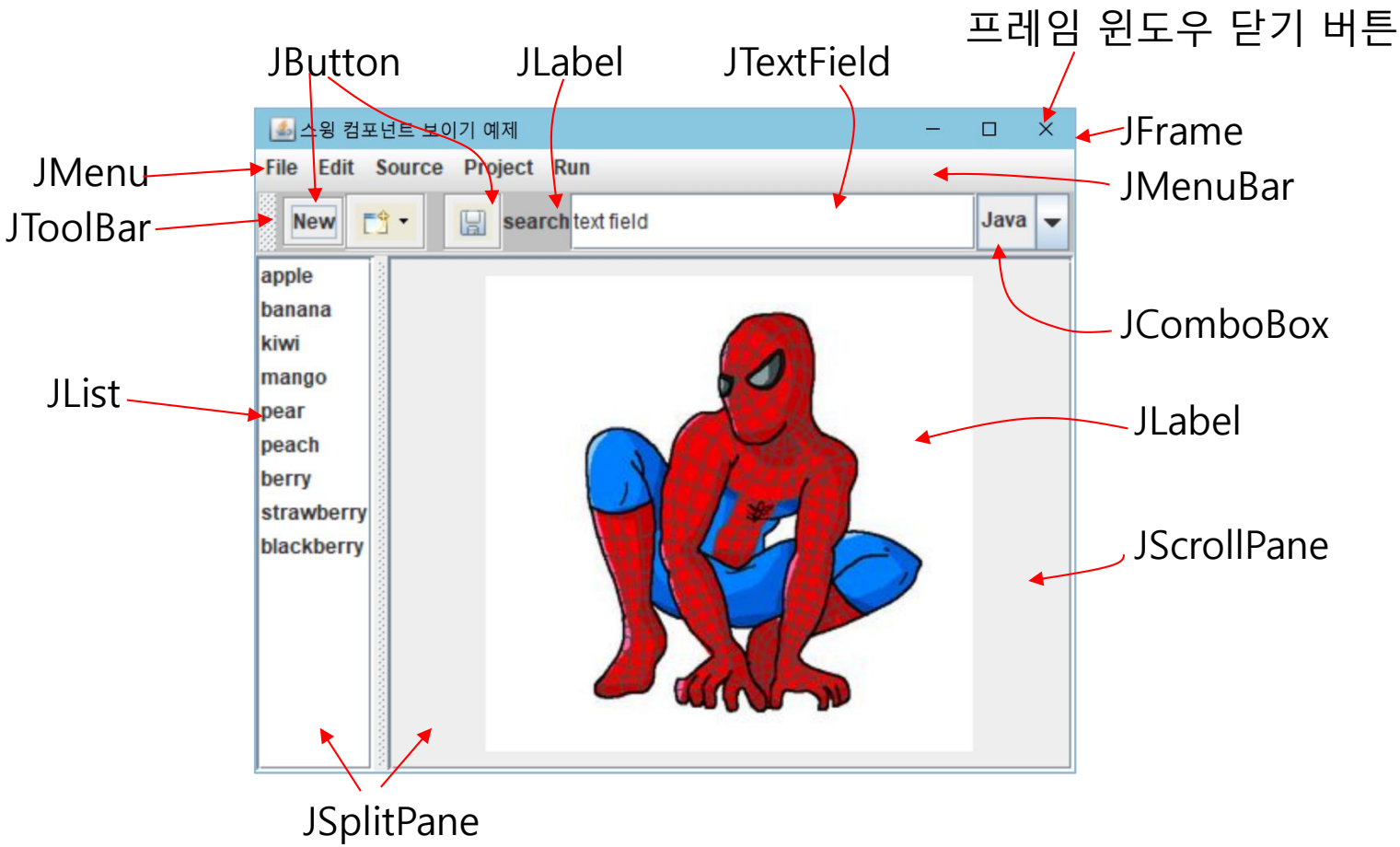
JTabbedPane



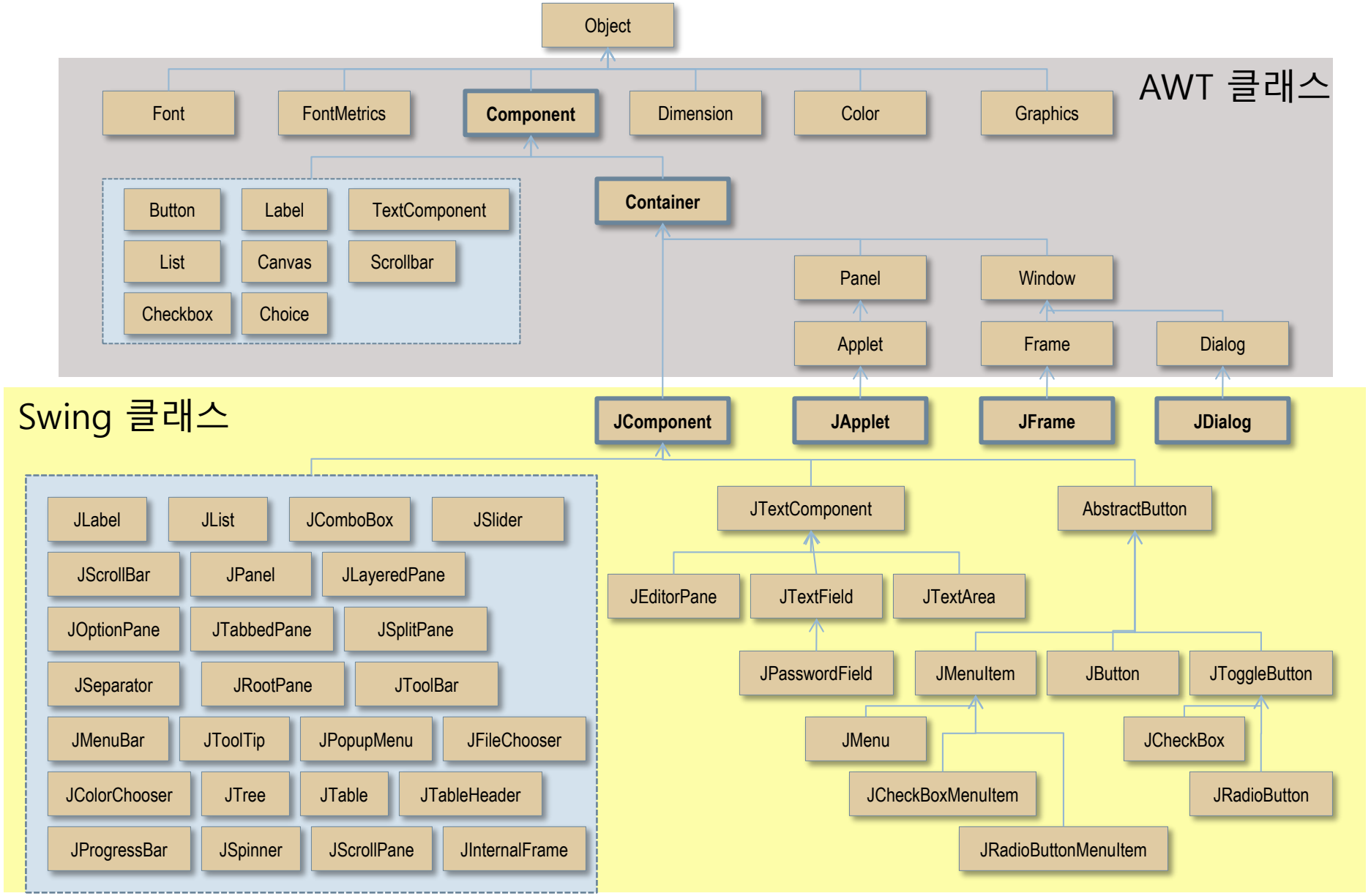
JSplitPane

Swing 으로 만든 GUI 프로그램 샘플





GUI 라이브러리 계층 구조



Swing 클래스의 특징



- 클래스 이름이 J 자로 시작
- 화려하고 다양한 컴포넌트로 쉽게 GUI 프로그래밍
- 스윙 컴포넌트는 2 가지 유형
  - JComponent를 상속받는 클래스
    - 대부분의 스윙 컴포넌트들이 이에 해당
  - AWT의 Container를 상속받는 몇 개의 클래스
    - JApplet, JDialog, JFrame 등
- JComponent
  - 스윙 컴포넌트의 공통 속성을 구현한 추상 클래스
    - ~~new JComponent()~~ 로 인스턴스를 생성할 수 없음
  - AWT의 Component를 상속받음

컨테이너와 컴포넌트



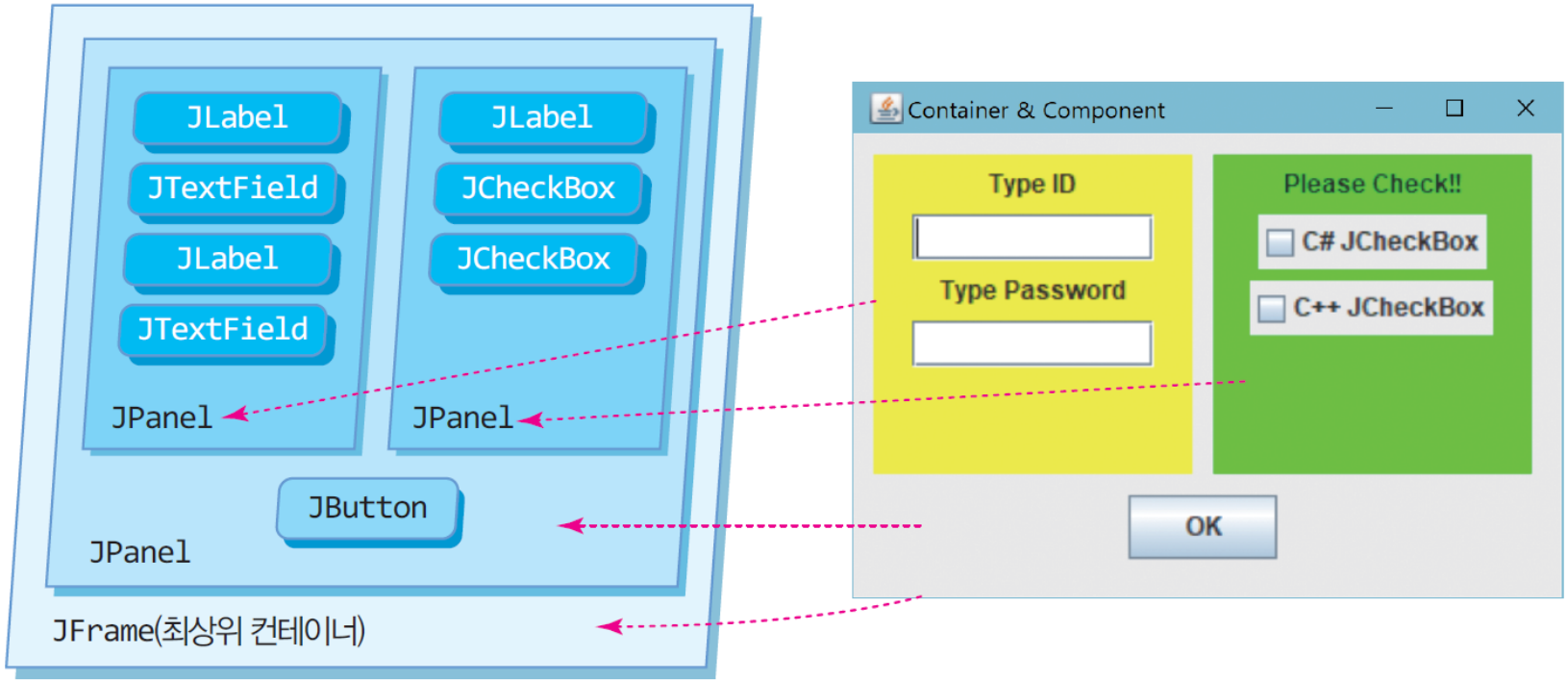
- 컨테이너
  - 다른 GUI 컴포넌트를 포함할 수 있는 컴포넌트
  - java.awt.Container 상속
  - 다른 컨테이너에 포함될 수 있음
  - 종류들
    - AWT 컨테이너 : Panel, Frame, Applet, Dialog, Window
    - Swing 컨테이너 : JPanel JFrame, JApplet, JDialog, JWindow

- 최상위 컨테이너
  - 다른 컨테이너에 속하지 않고 독립적으로 화면에 출력 가능한 컨테이너
    - JFrame, JDialog, JApplet
  - 모든 컴포넌트는 컨테이너에 포함되어야 화면에 출력 가능

- 컴포넌트
  - 컨테이너에 포함되어야 화면에 출력될 수 있는 순수 컴포넌트
  - 모든 컴포넌트는 java.awt.Component를 상속받음
  - 모든 스윙 컴포넌트는 javax.swing.JComponent를 상속받음

컨테이너와 컴포넌트의 포함관계





스윙의 컨테이너와 컴포넌트의 포함 관계

스윙 GUI 프로그램

스윙 GUI 프로그램 만들기



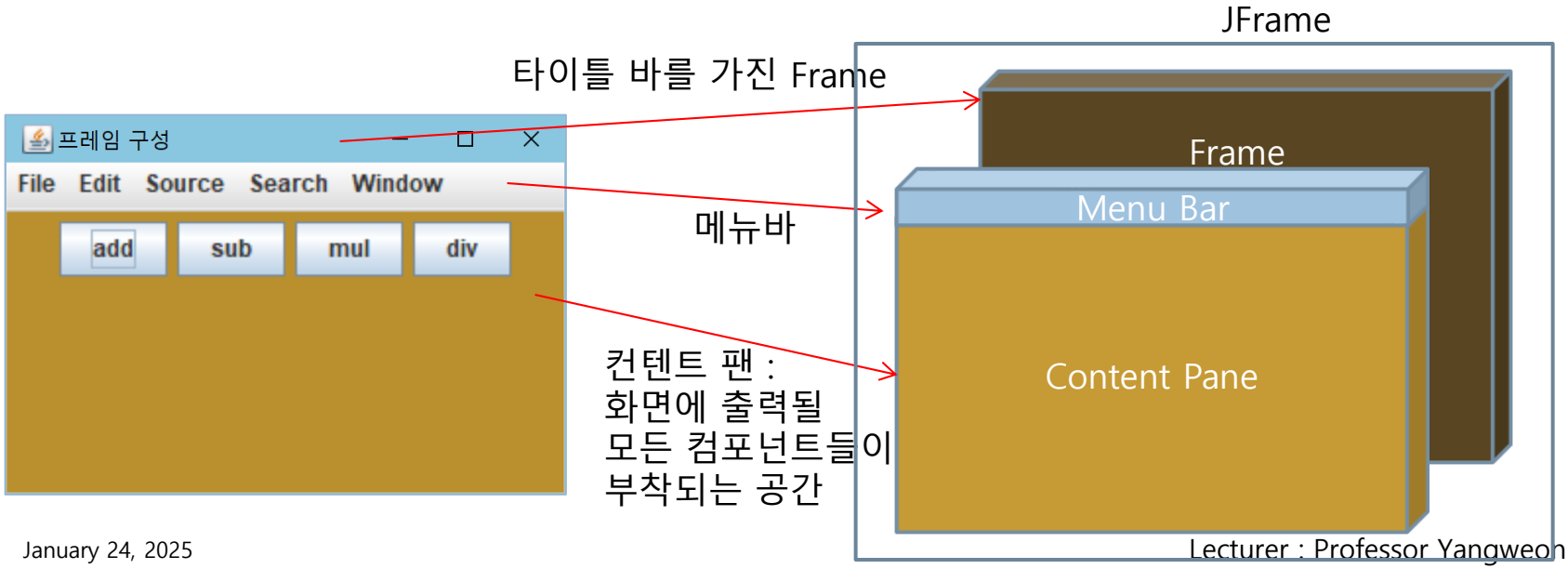
1. 스윙 프레임 작성
2. main() 메소드 작성
3. 프레임에 스윙 컴포넌트 붙이기

- 스윙 패키지 사용을 위한 import문
  - import java.awt.*; // 그래픽 처리를 위한 클래스들의 경로명
  - import java.awt.event.*; // AWT 이벤트 사용을 위한 경로명
  - import javax.swing.*; // 스윙 컴포넌트 클래스들의 경로명
  - import javax.swing.event.*; // 스윙 이벤트를 위한 경로명

스윙 프레임



- 모든 스윙 컴포넌트를 담는 최상위 GUI 컨테이너
  - JFrame을 상속받아 구현
  - 컴포넌트가 화면에 보이려면 스윙 프레임 내에 부착되어야 함
  - 프레임을 닫으면 프레임 내의 모든 컴포넌트가 보이지 않게 됨
- 스윙 프레임(JFrame) 기본 구성
  - 프레임 – 스윙 프로그램의 기본 틀
  - 메뉴바 – 메뉴들이 부착되는 공간
  - 컨텐트 팬 – GUI 컴포넌트들이 부착되는 공간

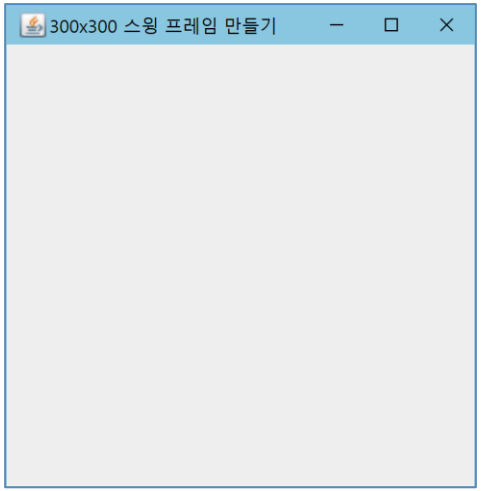


예제 9-1 : 300x300 크기의 스윙 프레임 만들기

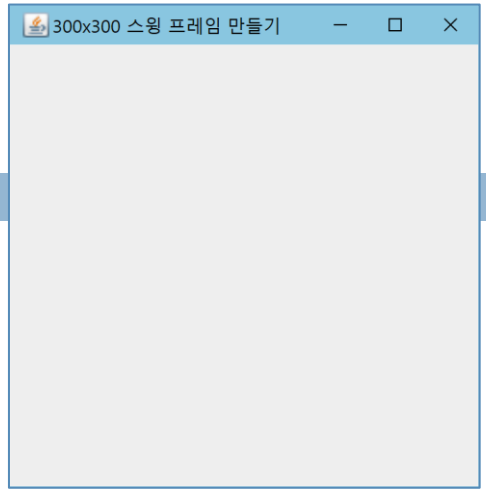


300×300 크기의 스윙 프레임을 작성하라.

```
import javax.swing.*;

public class MyFrame extends JFrame {
   public MyFrame() {
      setTitle("300x300 스윙 프레임 만들기");
      setSize(300,300); // 프레임 크기 300x300
      setVisible(true); // 프레임 출력
   }

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



main()의 위치





권장

```
import javax.swing.*;

public class MyFrame extends JFrame {
    MyFrame() {
        setTitle("첫번째 프레임");
        setSize(300,300);
        setVisible(true);
    }

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

main()을 프레임 클래스 내의 멤버로 작성

```
import javax.swing.*;

class MyFrame extends JFrame {
    MyFrame() {
        setTitle("첫번째 프레임");
        setSize(300,300);
        setVisible(true);
    }
}
public class MyApp {
    public static void main(String [] args) {
        MyFrame mf = new MyFrame();
    }
}
```

main()을 가진 다른 클래스 MyApp 작성

프레임에 컴포넌트 붙이기



타이틀 달기

```
public MyFrame() { // 생성자
   super("타이틀문자열"); // Jframe의 생성자 호출하여 타이틀 달기
   setTitle("타이틀문자열"); // 메소드를 호출하여 타이틀 달기
}
```

컨텐트팬 알아내기

```
Container contentPane = frame.getContentPane();
```

컨텐트팬에 컴포넌트 달기

```
Container c = frame.getContentPane();
JButton b = new JButton("Click");
c.add(b);
```

컨텐트팬 변경

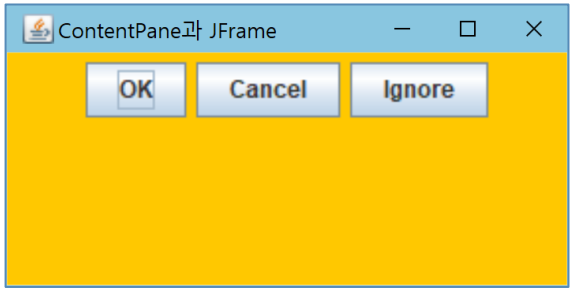
```
JPanel p = new JPanel();
frame.setContentPane(p);
```

예제 9-2 : 3개의 버튼 컴포넌트를 가진 스윙프레임 만들기



다음 그림과 같이 컨텐트팬의 배경색을 오렌지색으로 하고, 이곳에 OK, Cancel, Ignore 버튼들을 부착한 스윙 프로그램을 작성하라.

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

public class ContentPaneEx extends JFrame {
    public ContentPaneEx() {
        setTitle("ContentPane과 JFrame");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Container contentPane = getContentPane();
        contentPane.setBackground(Color.ORANGE);
        contentPane.setLayout(new FlowLayout());

        contentPane.add(new JButton("OK"));
        contentPane.add(new JButton("Cancel"));
        contentPane.add(new JButton("Ignore"));

        setSize(300, 150);
        setVisible(true);
    }

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



스윙 응용프로그램의 종료



- 응용프로그램 내에서 스스로 종료

```
System.exit(0);
```

  - 언제 어디서나 무조건 종료
- **프레임 종료버튼(X)이 클릭되면 어떤 일이 일어나는가?**
  - 프레임을 종료하여 프레임 윈도우가 닫힘
    - 프레임이 화면에서 보이지 않게 되고 응용프로그램이 사라짐
  - 프레임이 보이지 않게 되지만 응용프로그램이 종료한 것 아님
    - 키보드나 마우스 입력을 받지 못함
    - 다시 setVisible(true)를 호출하면 보이게 되고 이전 처럼 작동함
- 프레임 종료버튼이 클릭될 때 프레임을 닫고 응용 프로그램이 종료하도록 하는 방법

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

main() 종료 뒤에도 프레임이 살아 있는 이유?

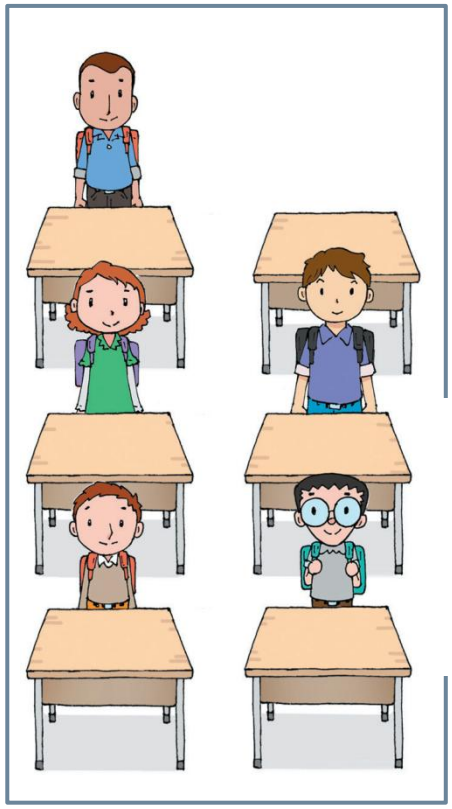


- 스윙 프로그램이 실행되는 동안 생성되는 스레드
  - 메인 스레드
    - main()을 실행하는 스레드
    - 자바 응용프로그램의 실행을 시작한 스레드
  - 이벤트 분배 스레드
    - 스윙 응용프로그램이 실행될 때 자동으로 실행되는 스레드
    - 이벤트 분배 스레드의 역할
      - 프레임과 버튼 등 GUI 화면 그리기
      - 키나 마우스 입력을 받아 이벤트를 처리할 코드 호출
- 자바 응용프로그램의 종료 조건
  - 실행 중인 사용자 스레드가 하나도 없을 때 종료
- 스윙 프로그램 main() 종료 뒤 프레임이 살아있는 이유
  - 메인 스레드가 종료되어도 이벤트 분배 스레드가 살아 있어 프레임 화면을 그리고 마우스나 키 입력을 받기 때문

컨테이너와 배치 개념



컨테이너(Container)





배치관리자
(Layout Manager)

컴포넌트
(Component)

1. 컨테이너마다 하나의 배치관리자가 존재하며, 삽입되는 모든 컴포넌트의 위치와 크기를 결정하고 적절히 배치한다.

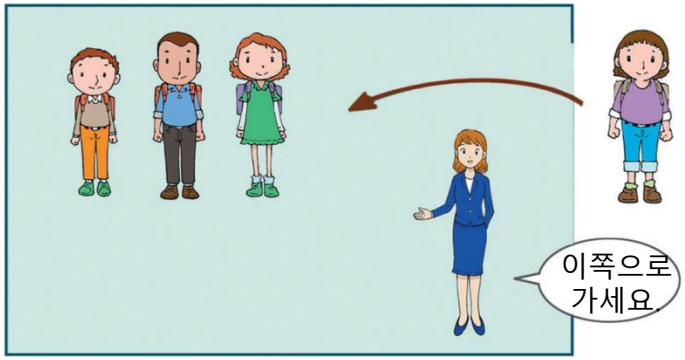
2. 컨테이너의 크기가 변하면 내부 컴포넌트들의 위치와 크기를 모두 재조절하고 재배치한다.

배치 관리자 대표 유형 4 가지

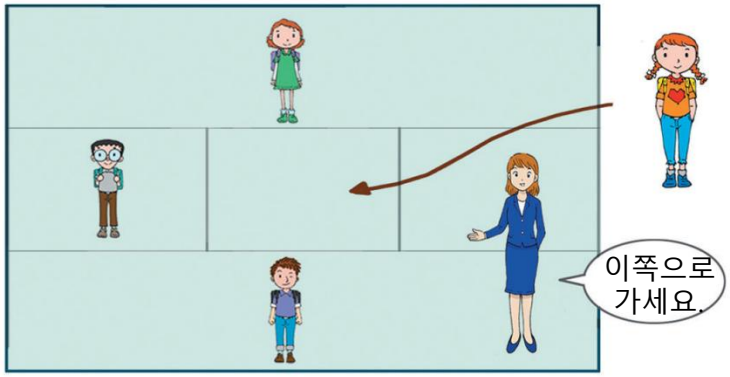


- java.awt 패키지에 구현되어 있음

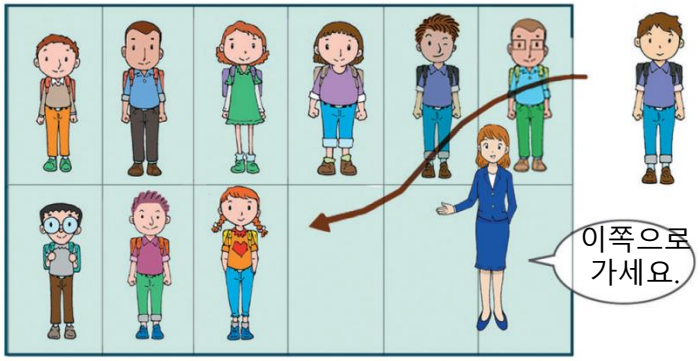
FlowLayout



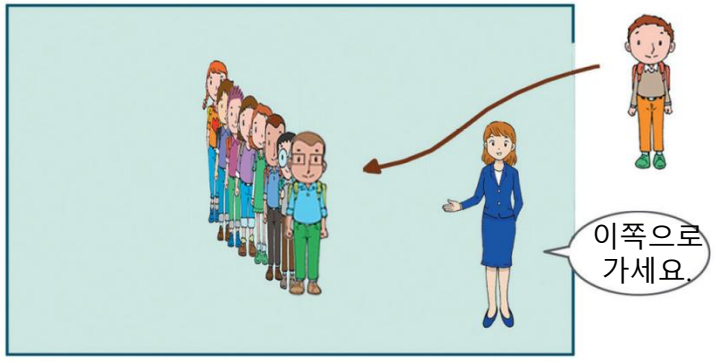
BorderLayout



GridLayout



CardLayout



컨테이너와 배치관리자



- 컨테이너의 디폴트 배치관리자
  - 컨테이너는 생성시 디폴트 배치관리자 설정

AWT와 스윙 컨테이너	디폴트 배치관리자
Window, JWindow	BorderLayout
Frame, JFrame	BorderLayout
Dialog, JDialog	BorderLayout
Panel, JPanel	FlowLayout
Applet, JApplet	FlowLayout

- 컨테이너에 새로운 배치관리자 설정
  - Container.setLayout(LayoutManager lm)
    - lm을 새로운 배치관리자로 설정

```
// JPanel 패널에 BorderLayout 배치관리자 설정

JPanel p = new JPanel();
p.setLayout(new BorderLayout());
```

```
// 컨텐트팬의 배치 관리자를 FlowLayout 으로 변경

Container c = frame.getConentPane(); // 컨텐트팬
c.setLayout(new FlowLayout());
```

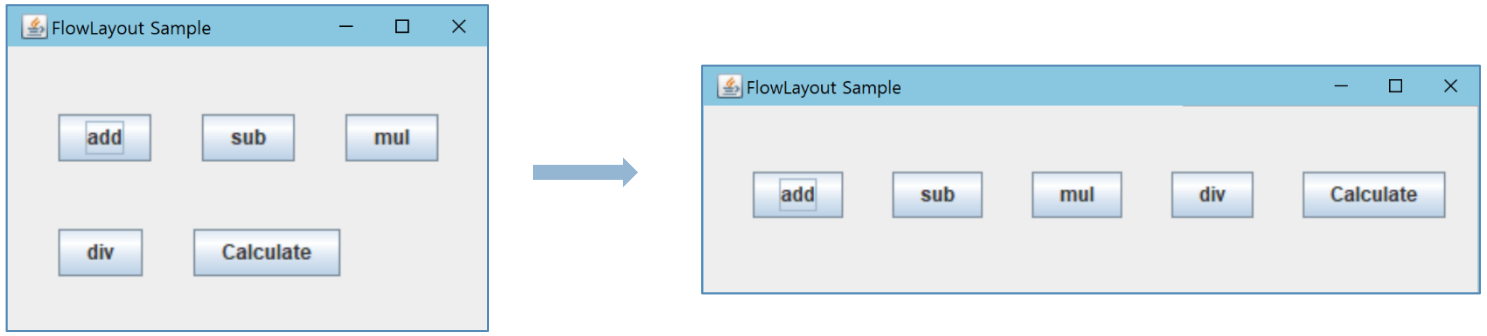
FlowLayout



- 배치방법
  - 컨테이너 공간 내에 왼쪽에서 오른쪽으로 배치
    - 다시 위에서 아래로 순서대로 컴포넌트를 배치한다.

```
container.setLayout(new FlowLayout());
container.add(new JButton("add"));
container.add(new JButton("sub"));
container.add(new JButton("mul"));
container.add(new JButton("div"));
container.add(new JButton("Calculate"));
```

  - 컨테이너의 크기가 변하면 컴포넌트 재배치



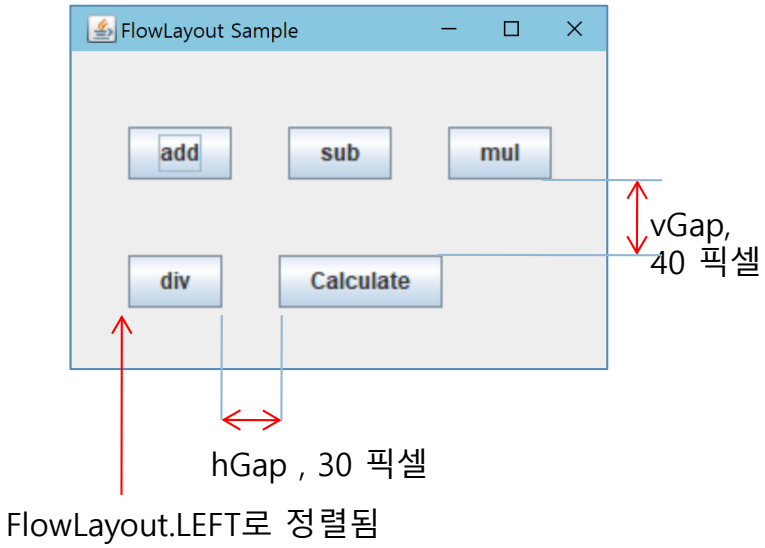
FlowLayout - 생성자와 속성



생성자

```
FlowLayout()
FlowLayout(int align)
FlowLayout(int align, int hGap, int vGap)
```

- align: 컴포넌트의 정렬 방법. 왼쪽 정렬(FlowLayout.LEFT), 오른쪽 정렬(FlowLayout.RIGHT), 중앙 정렬(FlowLayout.CENTER(디폴트))
- hGap: 좌우 컴포넌트 사이의 수평 간격, 픽셀 단위. 디폴트는 5
- vGap: 상하 컴포넌트 사이의 수직 간격, 픽셀 단위. 디폴트는 5

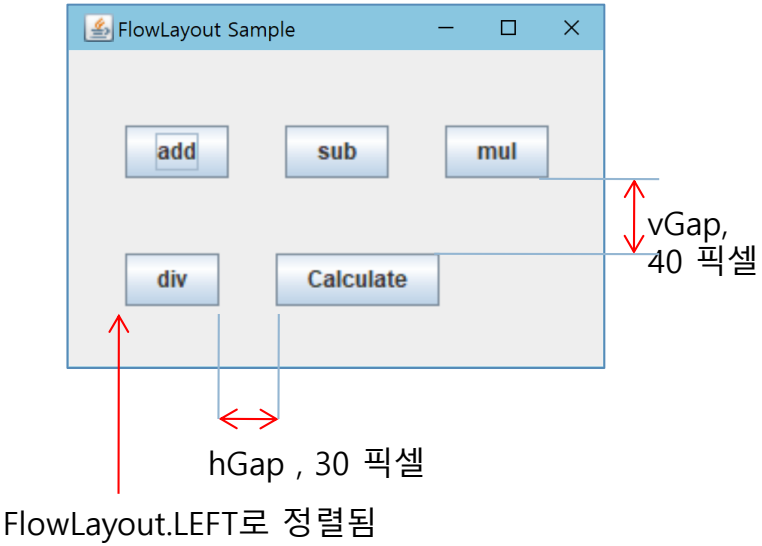


예제 9-3 : FlowLayout 배치관리자 활용



수평 간격이 30, 수직 간격을 40픽셀로 하고 LEFT로 정렬 배치하는 FlowLayout 배치관리자를 가진 컨텐트팬에 5개의 버튼 컴포넌트를 부착한 스윙 응용프로그램을 작성하라.

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

public class FlowLayoutEx extends JFrame {
    public FlowLayoutEx() {
        setTitle("FlowLayout Sample");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();

        c.setLayout(new FlowLayout(FlowLayout.LEFT, 30, 40));
        c.add(new JButton("add"));
        c.add(new JButton("sub"));
        c.add(new JButton("mul"));
        c.add(new JButton("div"));
        c.add(new JButton("Calculate"));

        setSize(300, 200);
        setVisible(true);
    }

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

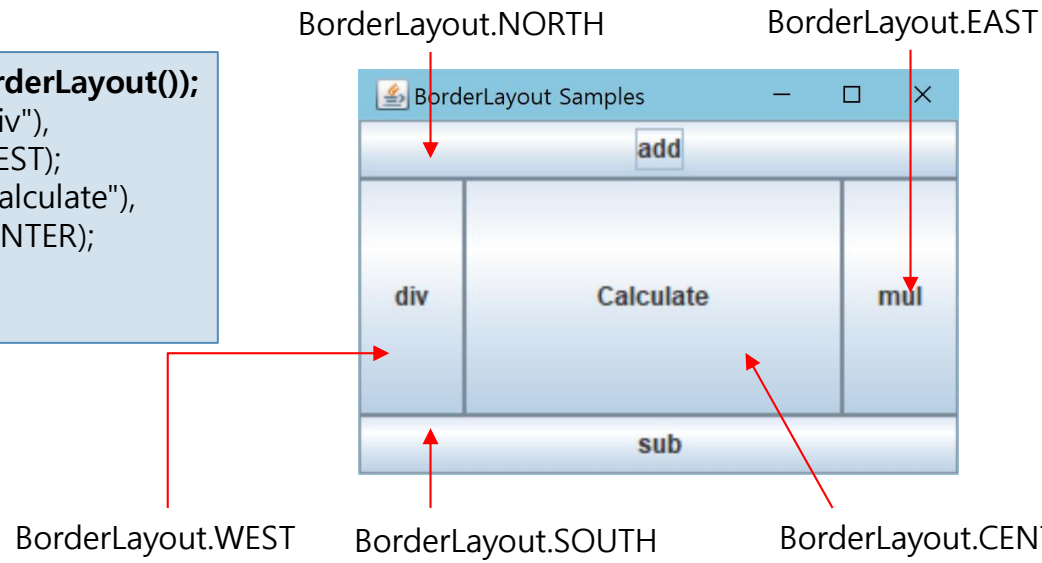


BorderLayout



- 배치방법
  - 컨테이너 공간을 5 구역으로 분할, 배치
    - East, West, South, North, Center
  - 배치 방법
    - add(Component comp, int index)
      - *comp를 index의 공간에 배치*
  - 컨테이너의 크기가 변하면 재배치

```
container.setLayout(new BorderLayout());
container.add(new JButton("div"),
                    BorderLayout.WEST);
container.add(new JButton("Calculate"),
                    BorderLayout.CENTER);
...
...
```



BorderLayout 생성자와 add() 멤버 함수



생성자

BorderLayout()

BorderLayout(int hGap, int vGap)

- hGap: 좌우 두 컴포넌트 사이의 수평 간격, 픽셀 단위. 디폴트는 0
- vGap: 상하 두 컴포넌트 사이의 수직 간격, 픽셀 단위. 디폴트는 0

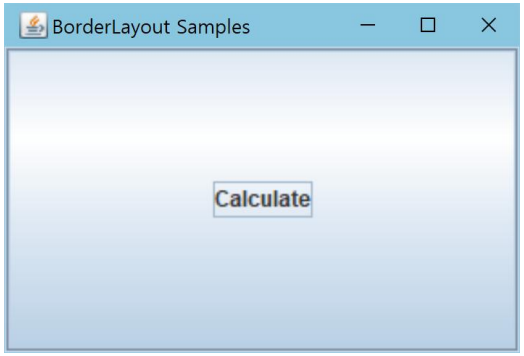
add() 멤버 함수

void add(Component comp, int index) comp 컴포넌트를 index 위치에 삽입한다.

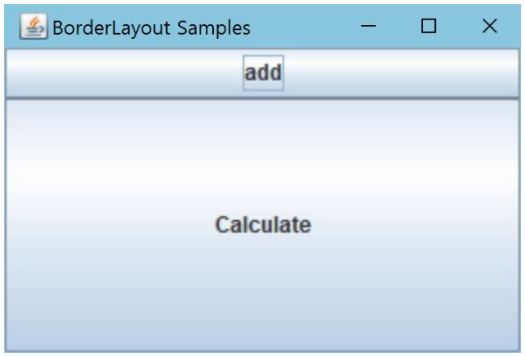
- comp: 컨테이너에 삽입되는 컴포넌트
- index: 컴포넌트의 위치
  - 동: BorderLayout.EAST
  - 서: BorderLayout.WEST
  - 남: BorderLayout.SOUTH
  - 북: BorderLayout.NORTH
  - 중앙: BorderLayout.CENTER

BorderLayout의 사용예

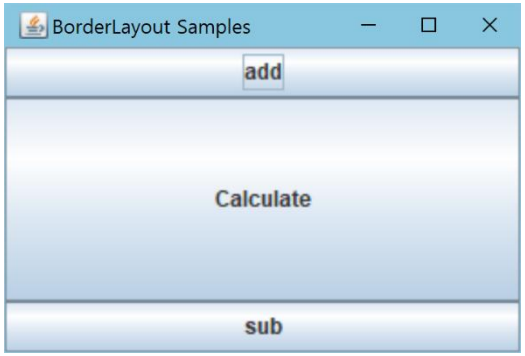




CENTER에 컴포넌트가 삽입될 때

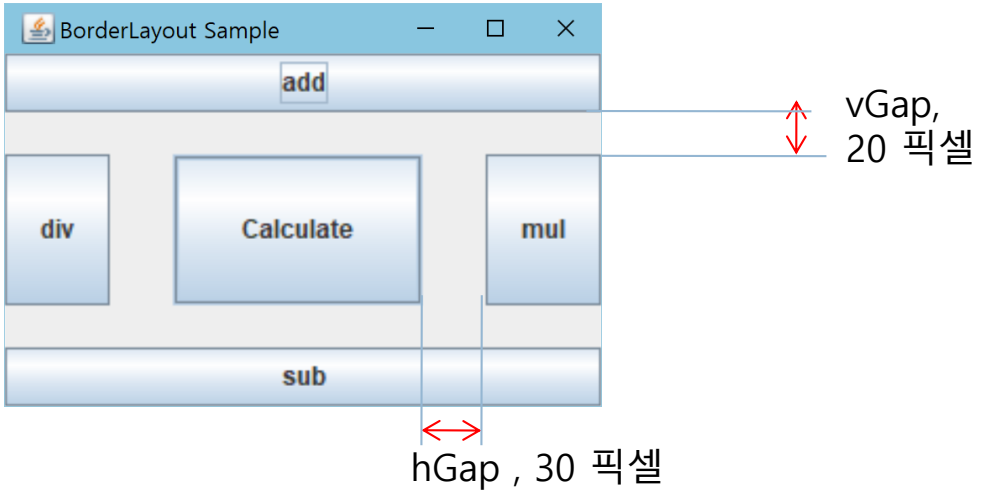


CENTER와 NORTH에 컴포넌트가 삽입될 때



CENTER, NORTH, SOUTH에 컴포넌트가 삽입될 때

new BorderLayout(30,20);
으로 배치관리자를
생성하였을 때

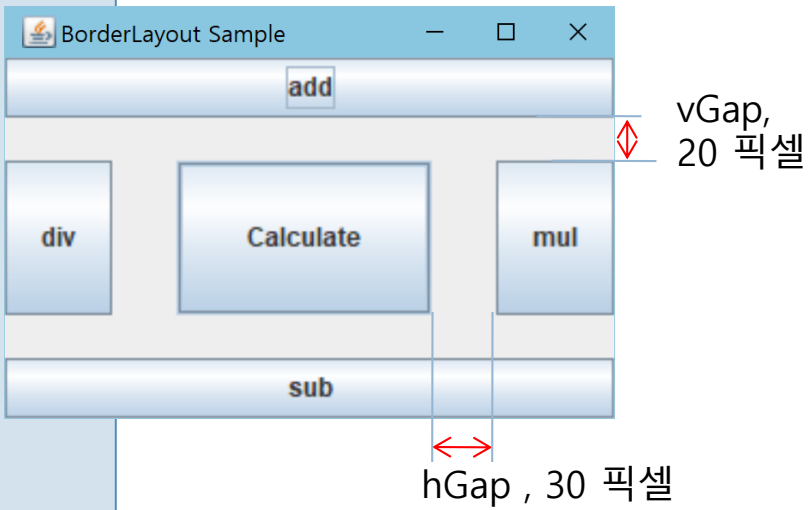


예제 9-4 : BorderLayout 배치관리자를 사용하는 예



BorderLayout 배치관리자를 사용하여 컨텐트팬에 다음과 같이 5개의 버튼 컴포넌트를 삽입하라.

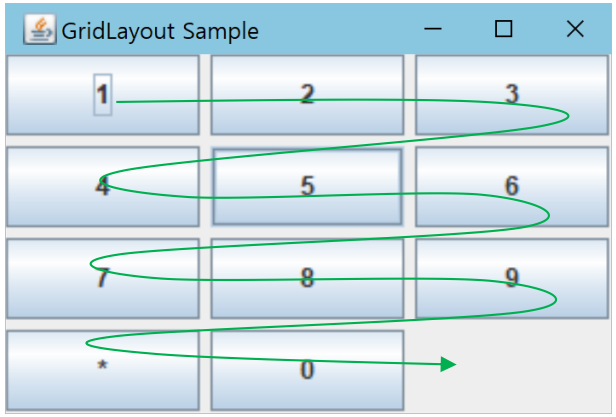
```
import javax.swing.*;
import java.awt.*;

public class BorderLayoutEx extends JFrame {
    public BorderLayoutEx() {
        setTitle("BorderLayout Sample");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Container c = getContentPane();

        c.setLayout(new BorderLayout(30, 20));
        c.add(new JButton("Calculate"), BorderLayout.CENTER);
        c.add(new JButton("add"), BorderLayout.NORTH);
        c.add(new JButton("sub"), BorderLayout.SOUTH);
        c.add(new JButton("mul"), BorderLayout.EAST);
        c.add(new JButton("div"), BorderLayout.WEST);

        setSize(300, 200); // 프레임 크기 300×200 설정
        setVisible(true); // 프레임을 화면에 출력
    }

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



GridLayout



- 배치방법
  - 컨테이너 공간을 동일한 사각형 격자(그리드)로 분할하고 각 셀에 하나의 컴포넌트 배치
    - 격자 구성은 생성자에 행수와 열수 지정
    - 셀에 왼쪽에서 오른쪽으로, 다시 위에서 아래로 순서대로 배치




```
container.setLayout(new GridLayout(4,3,5,5)); // 4×3 분할로 배치
container.add(new JButton("1")); // 상단 왼쪽 첫 번째 셀에 버튼 배치
container.add(new JButton("2")); // 그 옆 셀에 버튼 배치
```

- 4x3 그리드 레이아웃 설정
- 총 11 개의 버튼이 순서대로 add 됨
- 수직 간격 vGap : 5 픽셀
- 수평 간격 hGap : 5 픽셀

  - 컨테이너의 크기가 변하면 재배치
    - 크기 재조정

GridLayout 생성자와 속성



생성자

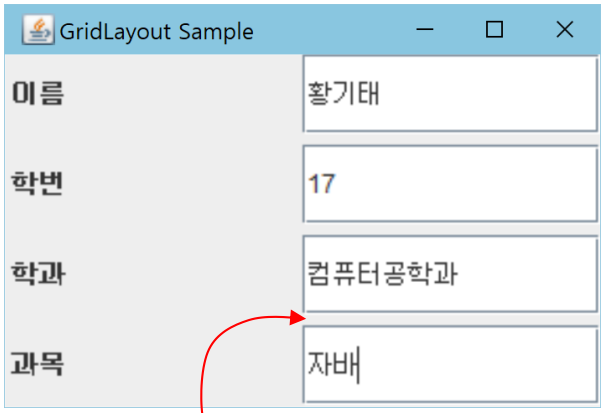
```
GridLayout()
GridLayout(int rows, int cols)
GridLayout(int rows, int cols, int hGap, int vGap)
```

- rows: 그리드의 행 수, 디폴트는 1
- cols: 그리드의 열 수, 디폴트는 1
- hGap: 좌우 컴포넌트 사이의 수평 간격, 픽셀 단위. 디폴트는 0
- vGap: 상하 컴포넌트 사이의 수직 간격, 픽셀 단위. 디폴트는 0

예제 9-5 : GridLayout으로 입력 폼 만들기



아래 화면과 같이 사용자로부터 입력받는 폼을 스윙 응용프로그램을 작성하라



두 행 사이의 수직 간격
vGap이 5 픽셀로 설정됨

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

public class GridLayoutEx extends JFrame {
    public GridLayoutEx() {
        setTitle("GridLayout Sample");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        GridLayout grid = new GridLayout(4, 2);
        grid.setVgap(5);

        Container c = getContentPane();
        c.setLayout(grid);
        c.add(new JLabel(" 이름"));
        c.add(new JTextField(""));
        c.add(new JLabel(" 학번"));
        c.add(new JTextField(""));
        c.add(new JLabel(" 학과"));
        c.add(new JTextField(""));
        c.add(new JLabel(" 과목"));
        c.add(new JTextField(""));
        setSize(300, 200);
        setVisible(true);
    }

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

배치관리자 없는 컨테이너



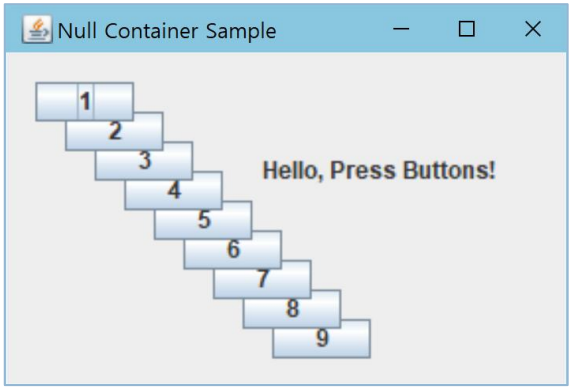
- 배치관리자가 없는 컨테이너란?
  - 응용프로그램에서 컴포넌트의 절대 크기와 절대 위치 결정
- 용도
  - 컴포넌트의 크기나 위치를 개발자 임의로 결정하고자 하는 경우
  - 게임 프로그램과 같이 시간이나 마우스/키보드의 입력에 따라 컴포넌트들의 위치와 크기가 수시로 변하는 경우
  - 여러 컴포넌트들이 서로 겹쳐 출력하고자 하는 경우
- 컨테이너의 배치 관리자 제거 방법
  - container.setLayout(null);

```
// JPanel의 배치관리자를 삭제하는 예

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

- 컴포넌트의 절대 크기와 절대 위치 설정
  - 프로그램 내에서 이루어져야 함
  - 다음 메소드 이용
    - 컴포넌트 크기 설정 : component.setSize(int width, int height);
    - 컴포넌트 위치 설정 : component.setLocation(int x, int y);
    - 컴포넌트 위치와 크기 동시 설정 : component.setBounds(int x, int y, int width, int height);

예제 9-6 : 배치관리자 없는 컨테이너에 컴포넌트를 절대 위치와 크기로 지정





원하는 위치에 원하는 크기로
컴포넌트를 마음대로
배치할 수 있다.

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

public class NullContainerEx extends JFrame {
    public NullContainerEx() {
        setTitle("Null Container Sample");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(null);

        JLabel la = new JLabel("Hello, Press Buttons!");
        la.setLocation(130, 50);
        la.setSize(200, 20);
        c.add(la);

        for(int i=1; i<=9; i++) {
            JButton b = new JButton(Integer.toString(i)); // 버튼 생성
            b.setLocation(i*15, i*15);
            b.setSize(50, 20);
            c.add(b); // 버튼을 컨텐트팬에 부착
        }
        setSize(300, 200);
        setVisible(true);
    }

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