08

자바 GUI 스윙 기초

학습 목표

1. 이벤트 기반 GUI 프로그래밍 이해
2. 자바 GUI 패키지 이해
3. 스윙으로 GUI 프로그램 작성
4. 컨테이너와 컴포넌트, 배치
5. FlowLayout 배치관리자 활용
6. BorderLayout 배치관리자 활용
7. GridLayout 배치관리자 활용
8. 배치 관리자 없는 컨테이너 만들기

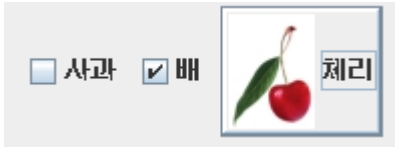
자바의 GUI(Graphical User Interface)

- GUI 응용프로그램
  - GUI
    - 사용자가 편리하게 입출력 할 수 있도록 그래픽으로 화면을 구성하고, 마우스나 키보드로 입력 받을 수 있도록 지원하는 사용자 인터페이스
  - 자바 언어에서 GUI 응용프로그램 작성
    - AWT와 Swing 패키지에 강력한 GUI 컴포넌트 제공
    - 쉬운 GUI 프로그래밍

- AWT와 Swing 패키지
  - AWT(Abstract Windowing Toolkit) 패키지
    - 자바가 처음 나왔을 때부터 배포된 GUI 패키지, 최근에는 거의 사용하지 않음
    - AWT 컴포넌트는 중량 컴포넌트(heavy weight component)
      - AWT 컴포넌트의 그리기는 운영체제에 의해 이루어지며, 운영체제에 의 자원을 많이 소모하고 부담을 줌
      - 운영체제가 직접 그리기 때문에 속도는 빠름
  - Swing 패키지
    - AWT 기술을 기반으로 작성된 자바 라이브러리
    - 모든 AWT 기능 + 추가된 풍부하고 화려한 고급 컴포넌트
    - AWT 컴포넌트를 모두 스윙으로 재작성. AWT 컴포넌트 이름 앞에 J자를 덧붙임
    - 순수 자바 언어로 구현
    - 스윙 컴포넌트는 경량 컴포넌트(light weight component)
      - 스윙 컴포넌트는 운영체제의 도움을 받지 않고 직접 그리기 때문에 운영체제에 부담주지 않음
    - 현재 자바의 GUI로 사용됨

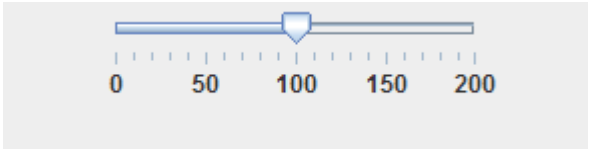
스윙 컴포넌트 예시

JButton

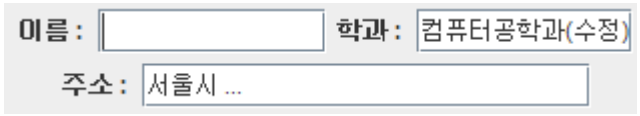


JCheckBox

JRadioButton



JSlider

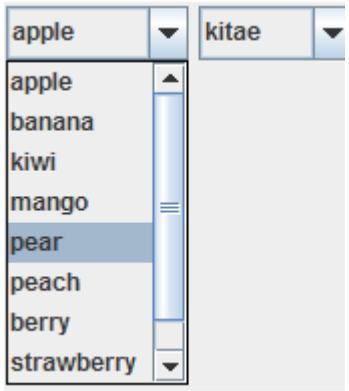


JTextField

JPasswordField

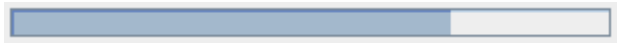
JSpinner

JTextArea

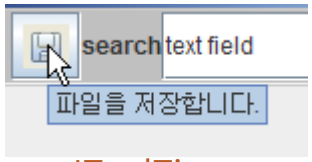


JComboBox

JList

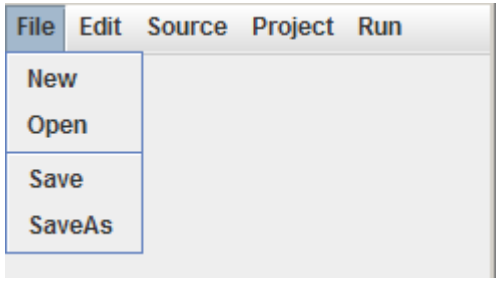


JProgressBar

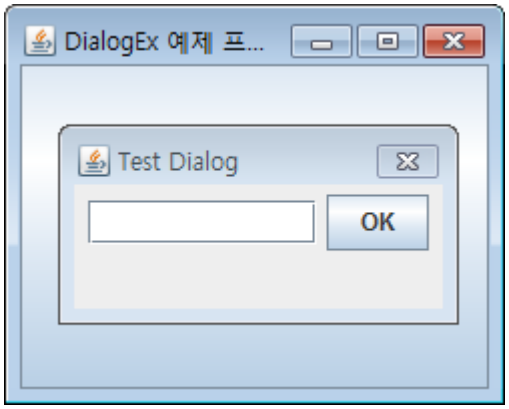


JToolTip

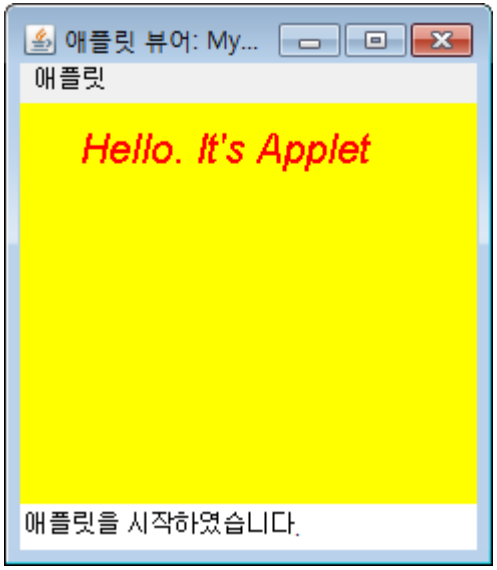
JScrollPane



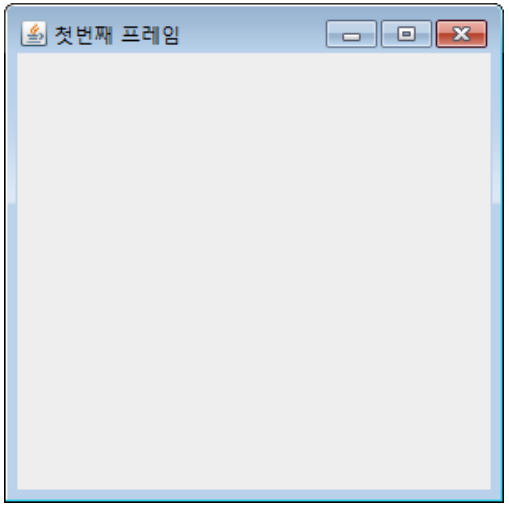
JMenu



JDialog



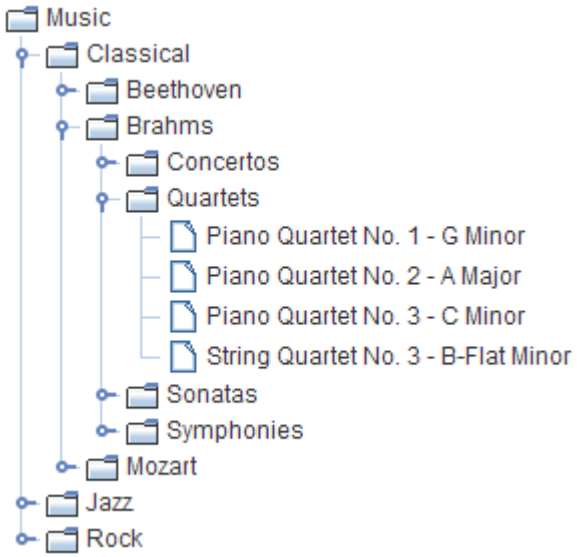
JApplet



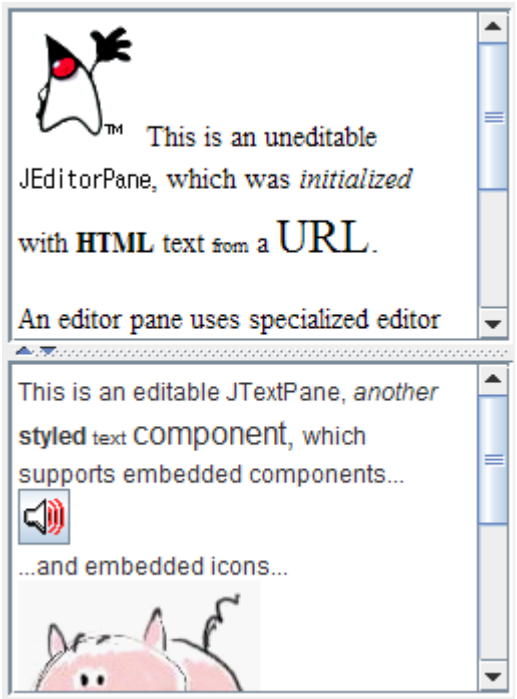
JFrame

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

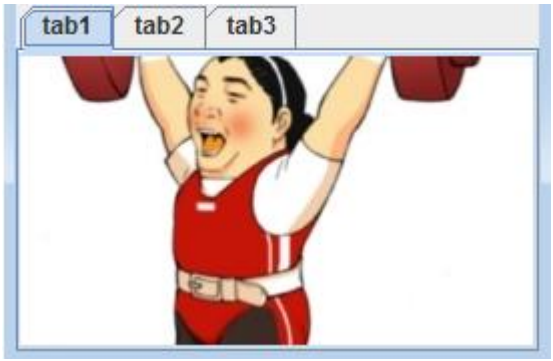


JTree



JEditorPane and JTextPane

JToolBar

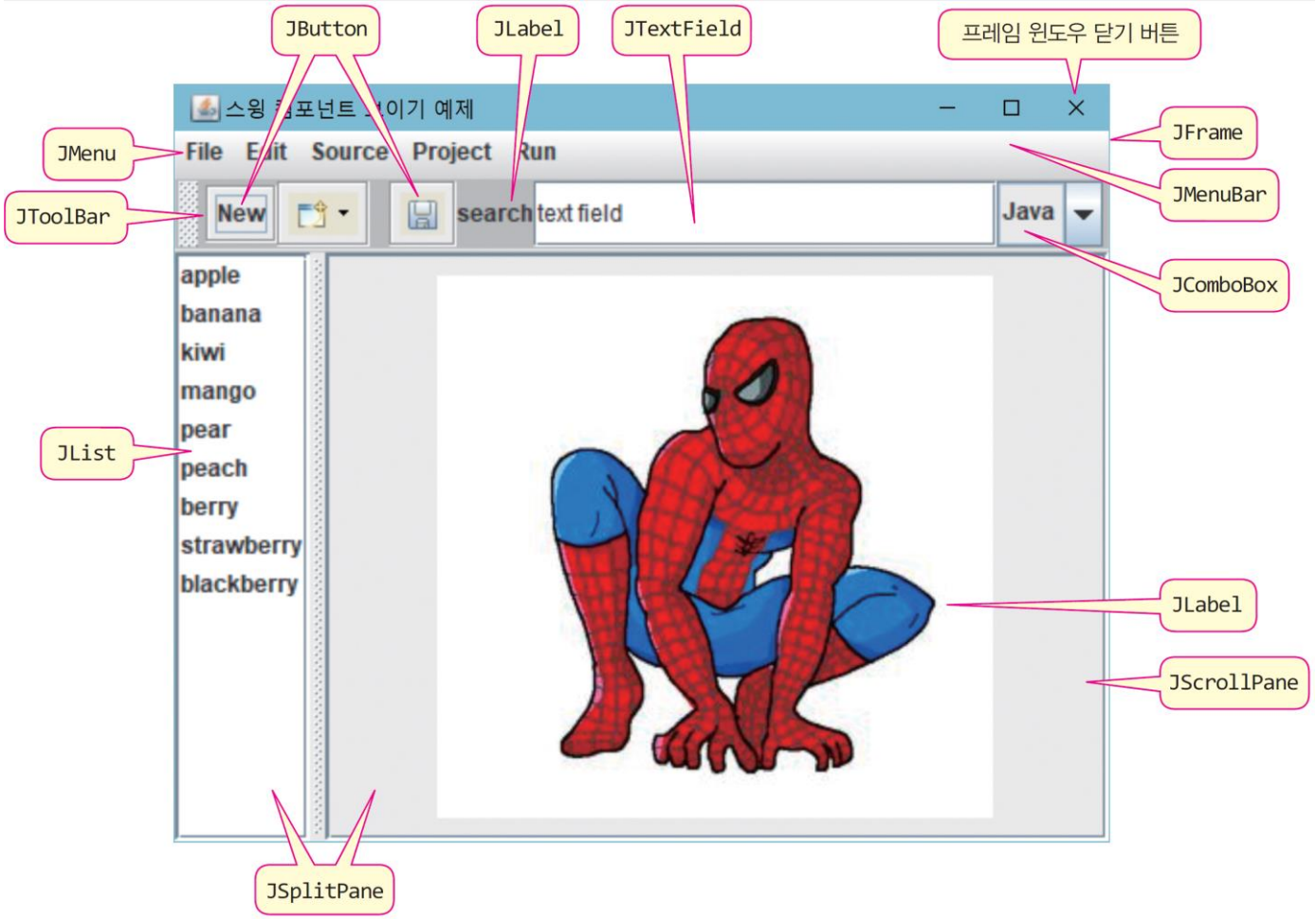


JTabbedPane

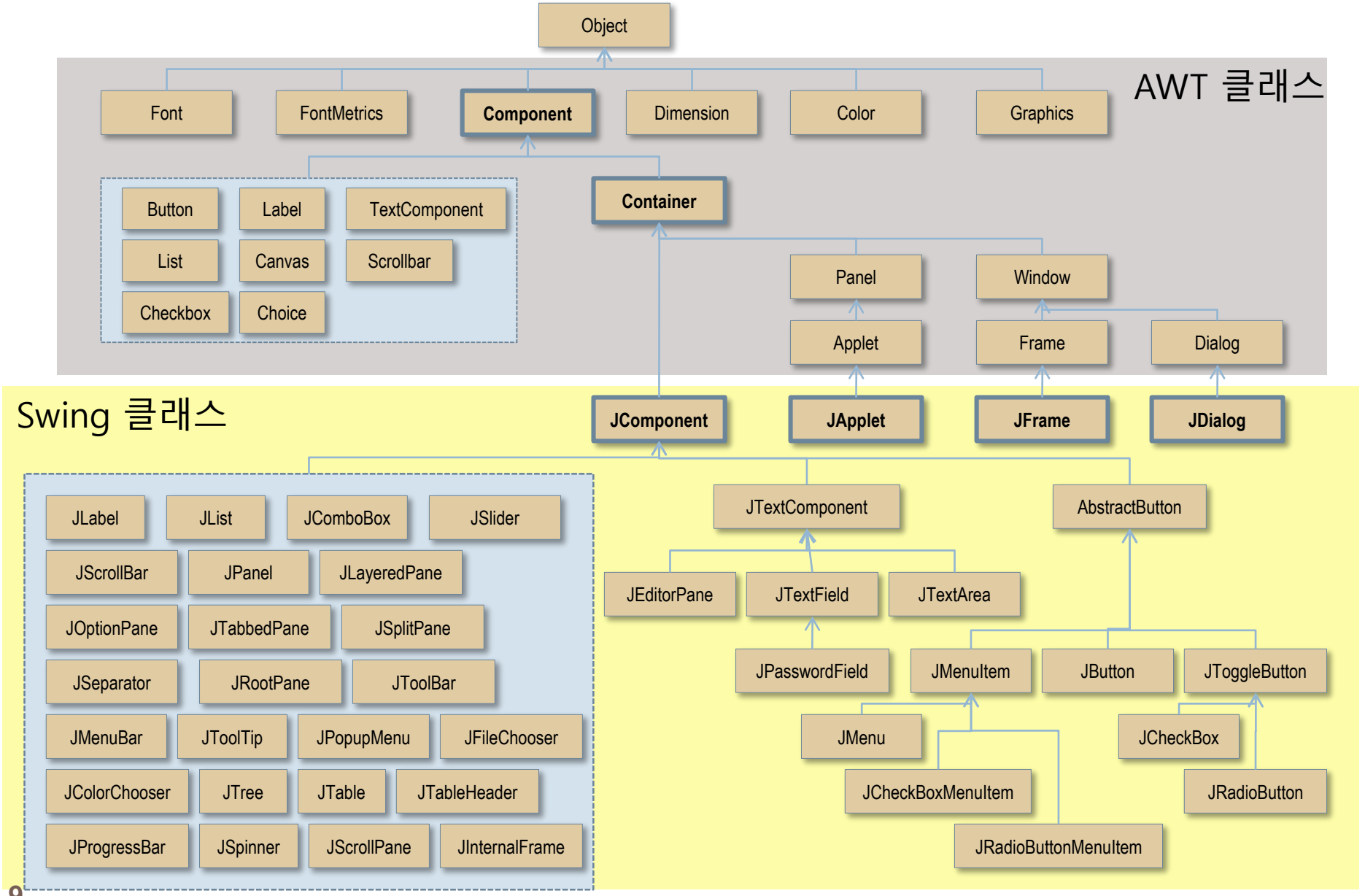
JSplitPane

스윙 GUI 프로그램 샘플

- 스윙 응용프로그램은 스윙 컴포넌트를 이용하여 레고 블록을 조립하듯이 작성



GUI 패키지 계층 구조


Object
AWT 클래스
Font
FontMetrics
Component
Dimension
Color
Graphics
Button
Label
TextComponent
List
Canvas
Scrollbar
Checkbox
Choice
Container
Panel
Window
Applet
Frame
Dialog
Swing 클래스
JComponent
JApplet
JFrame
JDialog
JLabel
JList
JComboBox
JSlider
JScrollBar
JPanel
JLayeredPane
JOptionPane
JTabbedPane
JSplitPane
JSeparator
JRootPane
JToolBar
JMenuBar
JToolTip
JPopupMenu
JFileChooser
JColorChooser
JTree
JTable
JTableHeader
JProgressBar
JSpinner
JScrollPane
JInternalFrame
JTextComponent
AbstractButton
JEditorPane
JTextField
JTextArea
JPasswordField
JMenuItem
JButton
JToggleButton
JMenu
JCheckBox
JCheckBoxMenuItem
JRadioButton
JRadioButtonMenuItem

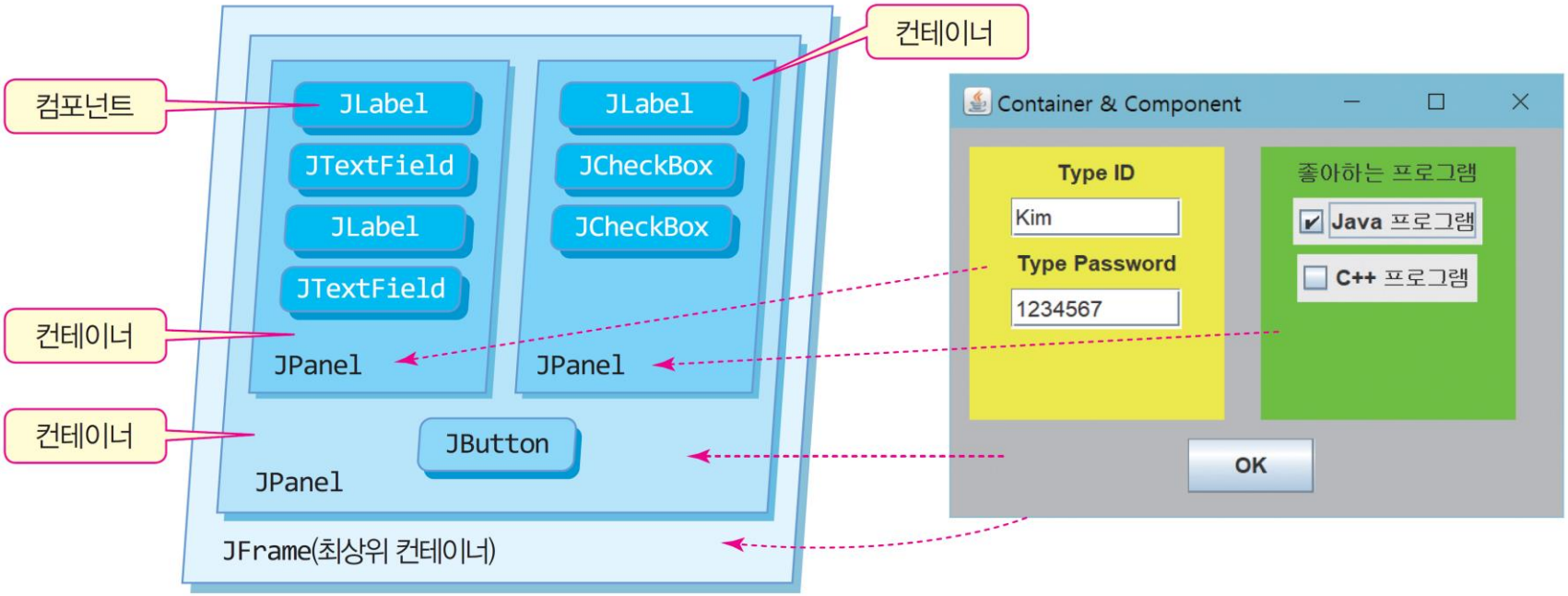
컨테이너와 컴포넌트

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

- 컴포넌트
  - 컨테이너에 포함되어야 화면에 출력될 수 있는 GUI 객체
  - 다른 컴포넌트를 포함할 수 없는 순수 컴포넌트
  - 모든 GUI 컴포넌트가 상속받는 클래스 : java.awt.Component
  - 스윙 컴포넌트가 상속받는 클래스 : javax.swing.JComponent

- 최상위 컨테이너
  - 다른 컨테이너에 포함되지 않고도 화면에 출력되며 독립적으로 존재 가능한 컨테이너
    - 스스로 화면에 자신을 출력하는 컨테이너 : JFrame, JDialog, JApplet

컨테이너와 컴포넌트의 포함관계



스윙 컨테이너와 컴포넌트의 포함 관계

스윙 GUI 프로그램

최상위 컨테이너를 바닥에 깔고, 그 위에 컨테이너를 놓고,
다시 컴포넌트를 쌓아가는 방식, 즉 레고 블록을 쌓는 듯이 GUI 프로그램을 작성한다.

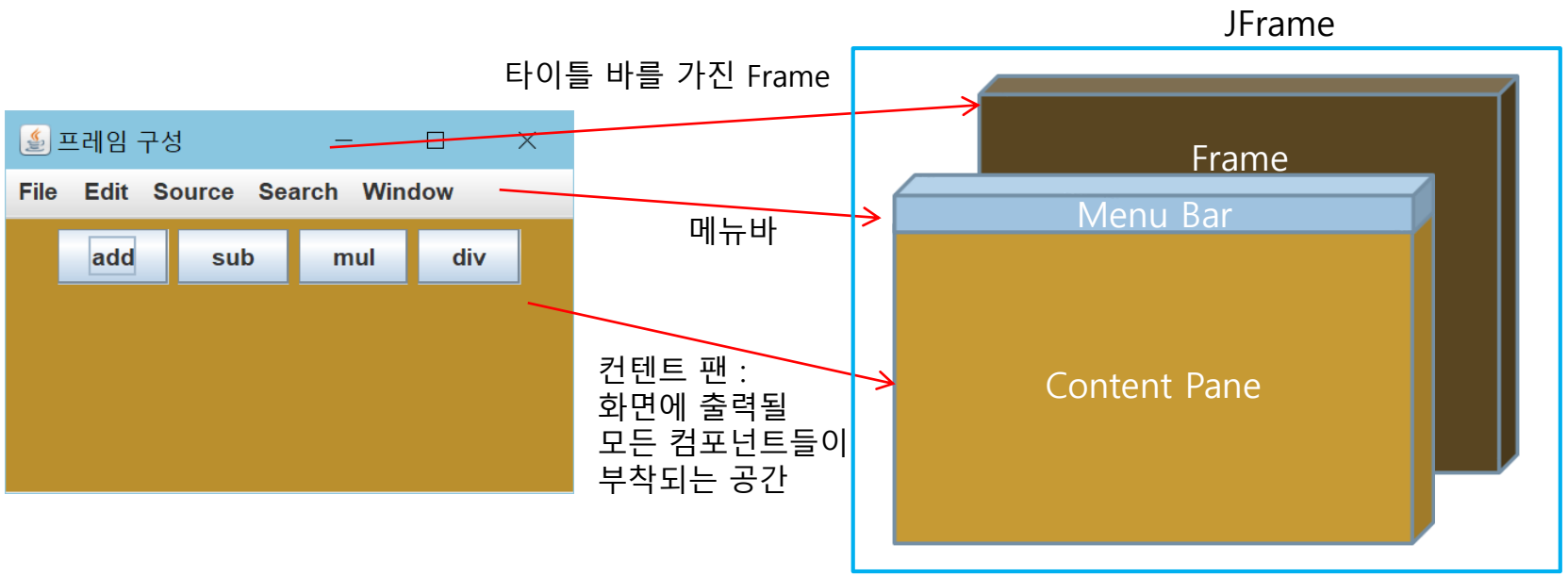
스윙 GUI 프로그램 만들기

- 스윙 GUI 프로그램을 만드는 과정
  1. 스윙 프레임 만들기
  2. main() 메소드 작성
  3. 스윙 프레임에 스윙 컴포넌트 붙이기

- 스윙 프로그램 작성에 필요한 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

- 스윙 프레임 : 모든 스윙 컴포넌트를 담는 최상위 컨테이너
  - JFrame을 상속받아 구현
  - 컴포넌트들은 화면에 보이려면 스윙 프레임에 부착되어야 함
    - 프레임을 닫으면 프레임에 부착된 모든 컴포넌트가 보이지 않게 됨
- 스윙 프레임(JFrame) 기본 구성
  - 프레임 – 스윙 프로그램의 기본 틀
  - 메뉴바 – 메뉴들이 부착되는 공간
  - **컨텐트팬 – GUI 컴포넌트들이 부착되는 공간**



프레임 만들기, JFrame 클래스 상속

- 스윙 프레임
  - JFrame 클래스를 상속받은 클래스 작성
  - 프레임의 크기 반드시 지정 : setSize() 호출
  - 프레임을 화면에 출력하는 코드 반드시 필요 : setVisible(true) 호출

JFrame을 상속받은 MyFrame 작성

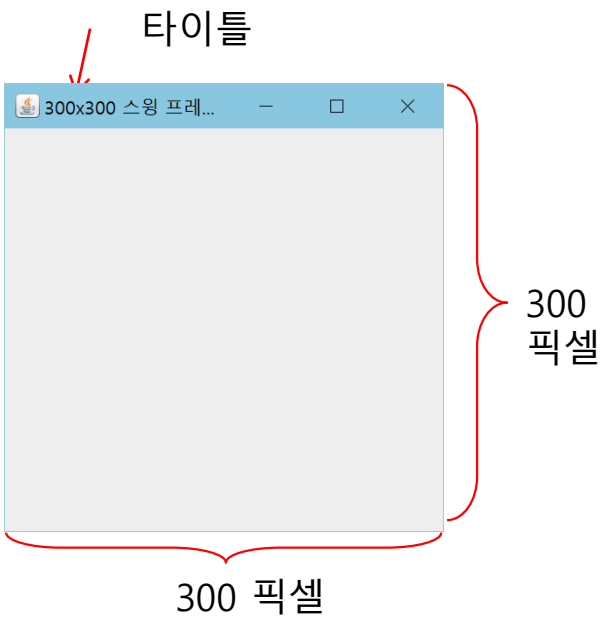
타이틀 설정

프레임 크기 지정

프레임을 화면에 출력

MyFrame 객체 즉 스윙 프레임 생성

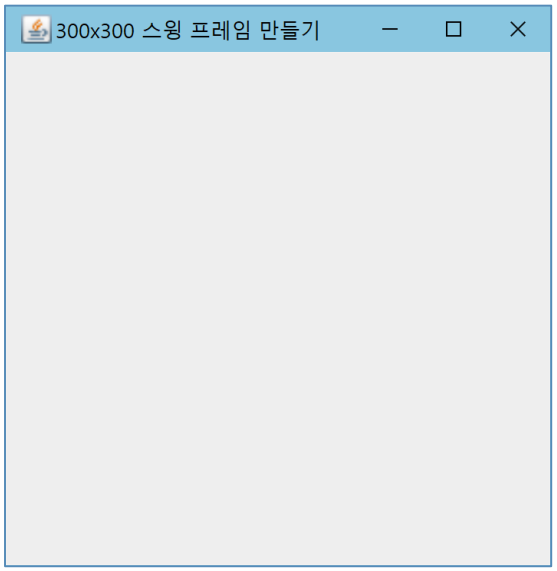
```
import javax.swing.*;

public class MyFrame extends JFrame {
    public MyFrame() {
        setTitle("300x300 스윙 프레임 만들기");

        setSize(300, 300);

        setVisible(true);
    }

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



예제 8-1 : 300x300 크기의 스윙 프레임 만들기

300×300 크기의 스윙 프레임을 만들어라.

```
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();
   }
}
```



스윙 응용프로그램에서 main()의 기능과 위치

- 스윙 응용프로그램에서 main()의 기능 최소화 바람직
  - 스윙 응용프로그램이 실행되는 시작점으로서의 기능만
  - 스윙 프레임을 생성하는 정도의 코드로 최소화

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

프레임에 컴포넌트 붙이기

- 타이틀 달기
  - super()나 setTitle() 이용

```
MyFrame() { // 생성자
    super("타이틀문자열");
}
```

```
MyFrame() { // 생성자
    setTitle("타이틀문자열");
}
```

- 컨텐트팬에 컴포넌트 달기
  - 컨텐트팬이란?
    - 스윙 컴포넌트들이 부착되는 공간
  - 컨텐트팬 알아내기
    - 스윙 프레임에 붙은 디폴트 컨텐트팬 알아내기
  - 컨텐트팬에 컴포넌트 붙이기
  - 컨텐트팬 변경

```
public class MyFrame extends JFrame {
    MyFrame() {
        ...
        // 프레임의 컨텐트팬을 알아낸다.
        Container contentPane = getContentPane();
    }
    ...
}
```

```
// 버튼 컴포넌트 생성
JButton button = new JButton("Click");
contentPane.add(button); // 컨텐트팬에 버튼 부착
```

```
class MyPanel extends JPanel {
    ... // JPanel을 상속받은 패널을 구현한다.
}
// frame의 컨텐트팬을 MyPanel 객체로 변경
frame.setContentPane(new MyPanel());
```

Tip. 컨텐트팬에 대한 JDK 1.5 이후의 추가 사항

- JDK 1.5 이전
  - 프레임의 컨텐트팬을 알아내어 반드시 컨텐트팬에 컴포넌트 부착

```
Container c = frame.getContentPane();
c.add(new JButton("Click")); // 컨텐트팬에 직접 컴포넌트 부착
```

- JDK 1.5 이후 추가된 사항
  - 프레임에 컴포넌트를 부착하면 프레임이 대신 컨텐트팬에 부착

```
frame.add(new JButton("Click"));
// 프레임이 버튼 컴포넌트를 컨텐트팬에 대신 부착
```

- 저자의 결론
  - JDK1.5이전처럼 직접 컨텐트팬에 컴포넌트를 부착하는 것이 바람직함
  - 컨텐트팬 다루기 능력 필요하기 때문
  - 컴포넌트의 부모가 프레임이 아닌, 컨텐트팬임을 알고 명확히 사용할 필요

예제 8-2 : 3개의 버튼 컴포넌트를 가진 스윙 프레임 만들기

다음 그림과 같이 콘텐트팬의 배경색을 오렌지색으로 하고, OK, Cancel, Ignore 버튼을 부착한 스윙 프로그램을 작성하라.

FlowLayout의 배치관리자는 뒤에서 배울 내용으로서, 컴포넌트를 순서대로 부착하는 일을 맡은 객체

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

public class ContentPaneEx extends JFrame {
    public ContentPaneEx() {
        setTitle("ContentPane과 JFrame 예제"); // 프레임의 타이틀 달기
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Container contentPane = getContentPane(); // 컨텐트팬 알아내기
        contentPane.setBackground(Color.ORANGE); // 오렌지색 배경 설정
        contentPane.setLayout(new FlowLayout()); // 컨텐트팬에 FlowLayout
                                                 //    배치관리자 달기

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

        setSize(300, 150); // 프레임 크기 300x150 설정
        setVisible(true); // 화면에 프레임 출력
    }

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

스윙 응용프로그램의 종료

- 응용프로그램 내에서 스스로 종료하는 방법

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

  - 언제 어디서나 무조건 종료

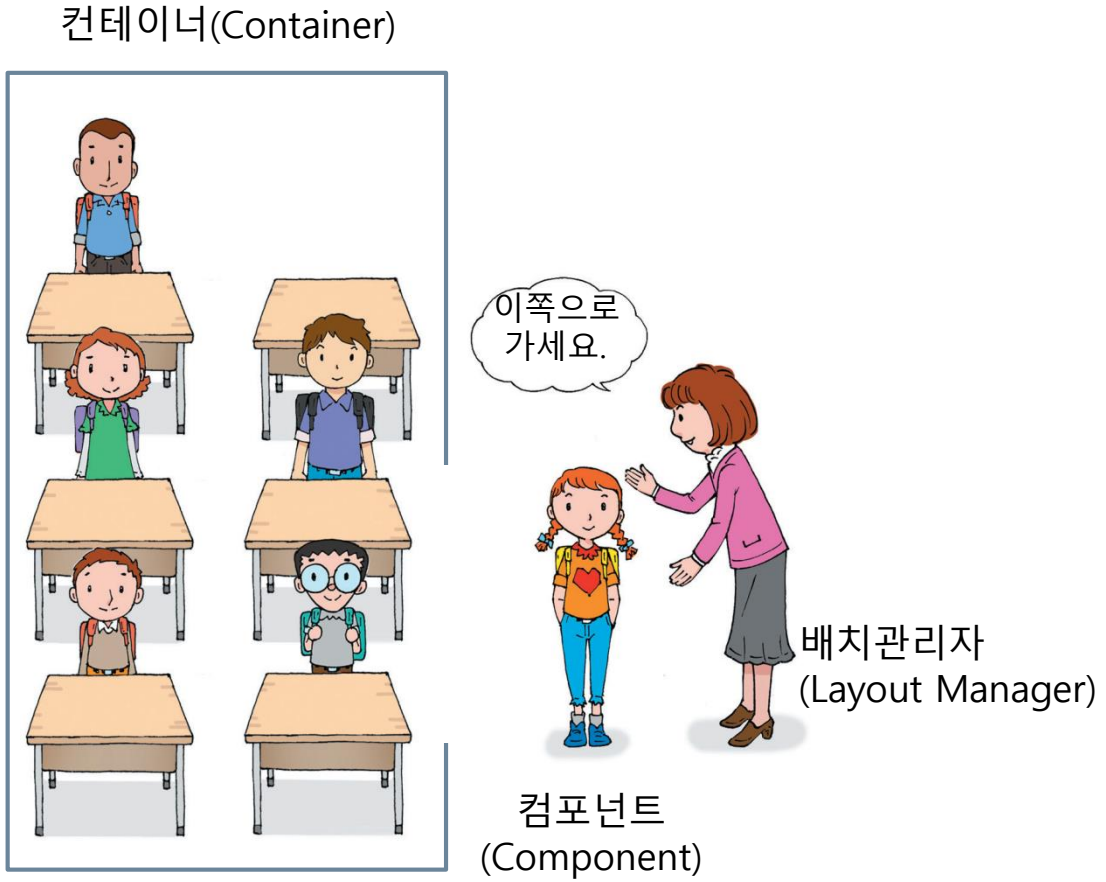
- **프레임의 오른쪽 상단의 종료버튼(X)이 클릭되면 어떤 일이 일어나는가?**
  - 프레임 종료, 프레임 윈도우를 닫음
    - 프레임이 화면에서 보이지 않게 됨
  - 프레임이 보이지 않게 되지만 응용프로그램이 종료한 것 아님
    - 키보드나 마우스 입력을 받지 못함
    - 다시 setVisible(true)를 호출하면, 보이게 되고 이전 처럼 작동함

- 프레임 종료버튼이 클릭될 때, 프레임과 함께 프로그램을 종료시키는 방법

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

컨테이너와 배치, 배치관리자 개념

- 컨테이너의 배치관리자
  - 컨테이너마다 하나의 배치관리자 존재
  - 컨테이너에 부착되는 컴포넌트의 위치와 크기 결정
  - 컨테이너의 크기가 변경되면, 컴포넌트의 위치와 크기 재결정



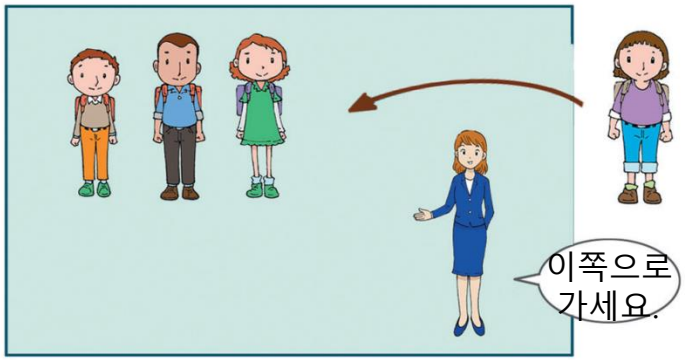
배치 관리자 대표 유형 4 가지

- FlowLayout 배치관리자
  - 컴포넌트가 삽입되는 순서대로 왼쪽에서 오른쪽으로 배치
  - 배치할 공간이 없으면 아래로 내려와서 반복한다.
- BorderLayout 배치관리자
  - 컨테이너의 공간을 동(EAST), 서(WEST), 남(SOUTH), 북(NORTH), 중앙(CENTER)의 5개 영역으로 나눔
  - 5개 영역 중 응용프로그램에서 지정한 영역에 컴포넌트 배치
- GridLayout 배치관리자
  - 컨테이너를 프로그램에서 설정한 동일한 크기의 2차원 격자로 나눔
  - 컴포넌트는 삽입 순서대로 좌에서 우로, 다시 위에서 아래로 배치
- CardLayout
  - 컨테이너의 공간에 카드를 쌓아 놓은 듯이 컴포넌트를 포개어 배치

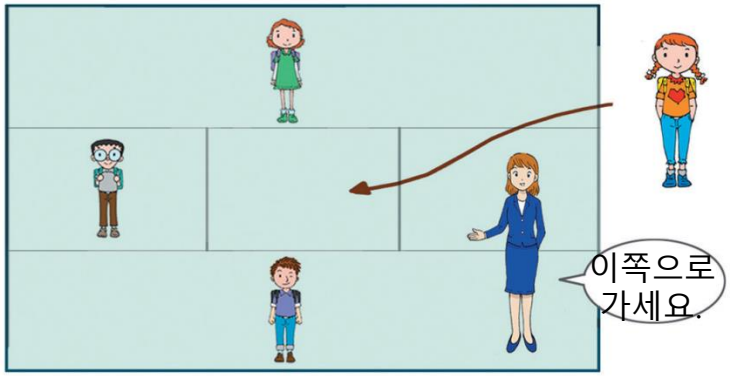
배치 관리자 대표 유형 4 가지

- java.awt 패키지에 구현되어 있음

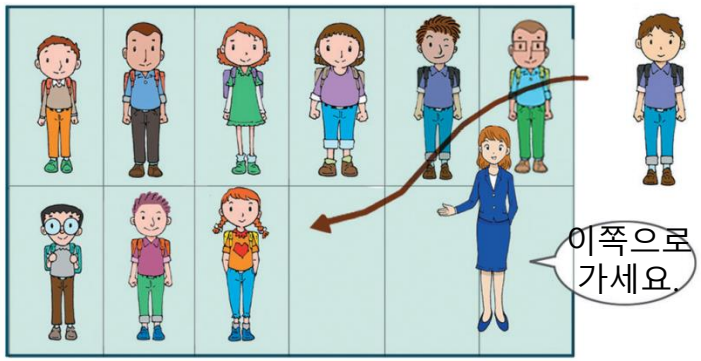
FlowLayout



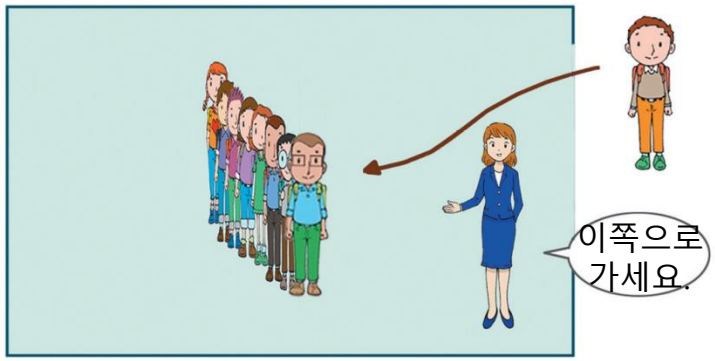
BorderLayout



GridLayout



CardLayout



컨테이너와 디폴트 배치관리자

- 컨테이너의 디폴트 배치관리자
  - 컨테이너 생성시 자동으로 생성되는 배치관리자

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

컨테이너에 새로운 배치관리자 설정

- 컨테이너에 새로운 배치관리자 설정
  - setLayout(LayoutManager lm) 메소드 호출
    - lm을 새로운 배치관리자로 설정
  - 사례
    - JPanel 컨테이너에 BorderLayout 배치관리자를 설정하는 예

```
JPanel p = new JPanel();
p.setLayout(new BorderLayout()); // JPanel에 BorderLayout 설정
```

    - 컨텐트팬의 배치관리자를 FlowLayout 배치관리자로 설정

```
Container c = frame.getConentPane(); // 프레임의 컨텐트팬 알아내기

c.setLayout(new FlowLayout()); // 컨텐트팬에 FlowLayout 설정
```

    - 오류

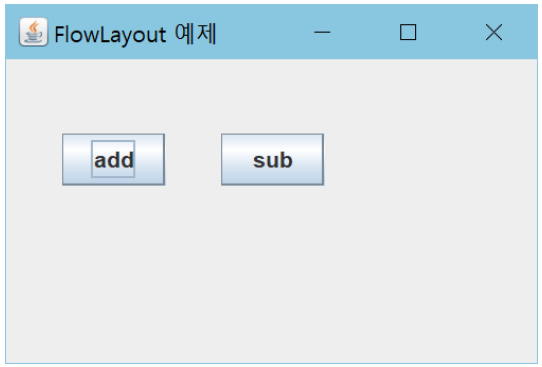
오류

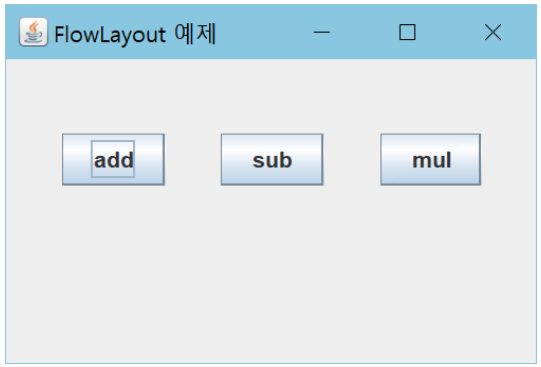
```
c.setLayout(~~FlowLayout~~); // 오류
```

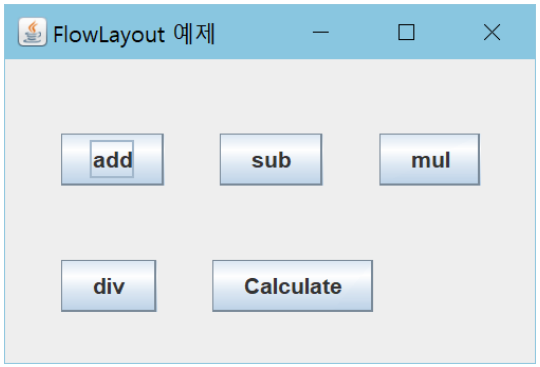
FlowLayout 배치관리자

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

```
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"));
```

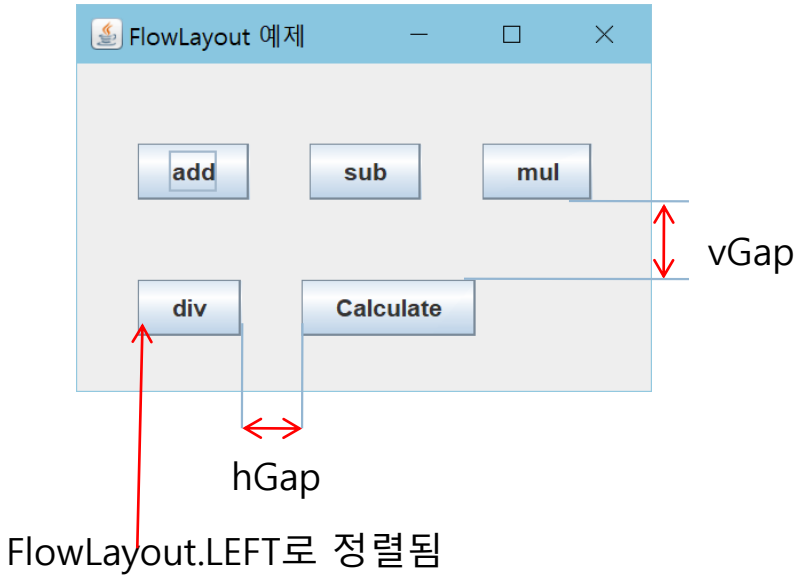






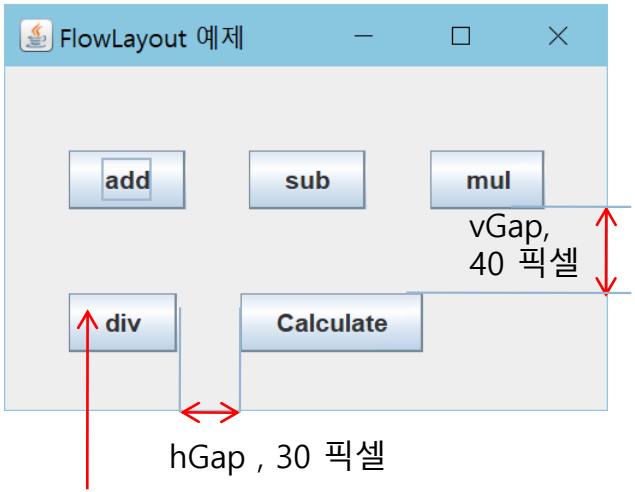
FlowLayout의 생성자

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



예제 8-3 : FlowLayout 배치관리자 활용

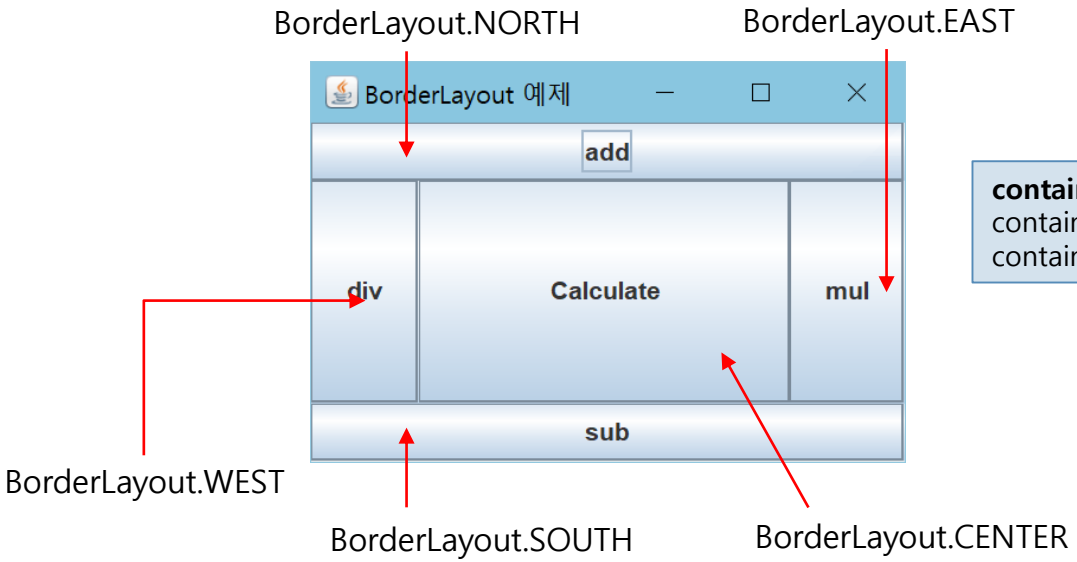
FlowLayout 배치관리자를 사용하여 다음 그림과 같이 5개의 버튼을 배치하라.




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

public class FlowLayoutEx extends JFrame {
    public FlowLayoutEx() {
        setTitle("FlowLayout 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container contentPane = getContentPane(); // 컨텐트팬 알아내기

        // 왼쪽 정렬로, 수평 간격을 30, 수직 간격을 40 픽셀로 배치하는
        // FlowLayout 생성
        contentPane.setLayout(new FlowLayout(FlowLayout.LEFT, 30, 40));

        contentPane.add(new JButton("add"));
        contentPane.add(new JButton("sub"));
        contentPane.add(new JButton("mul"));
        contentPane.add(new JButton("div"));
        contentPane.add(new JButton("Calculate"));

        setSize(300, 200); // 프레임 크기 300x200 설정
        setVisible(true); // 화면에 프레임 출력
    }

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

BorderLayout 배치관리자

- 배치방법
  - 컨테이너 공간을 5 구역으로 분할, 배치
    - 동, 서, 남, 북, 중앙
  - 배치 방법
    - add(Component comp, int index)
      - *comp를 index의 공간에 배치*




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

BorderLayout 생성자와 add() 메소드

- 생성자
  - BorderLayout()
  - BorderLayout(int hGap, int vGap)
    - hGap : 좌우 두 컴포넌트 사이의 수평 간격, 픽셀 단위(디폴트 : 0)
    - vGap : 상하 두 컴포넌트 사이의 수직 간격, 픽셀 단위(디폴트 : 0)

- add() 메소드
  - void add(Component comp, int index)
    - comp 컴포넌트를 index 위치에 삽입한다.
    - index : 컴포넌트의 위치

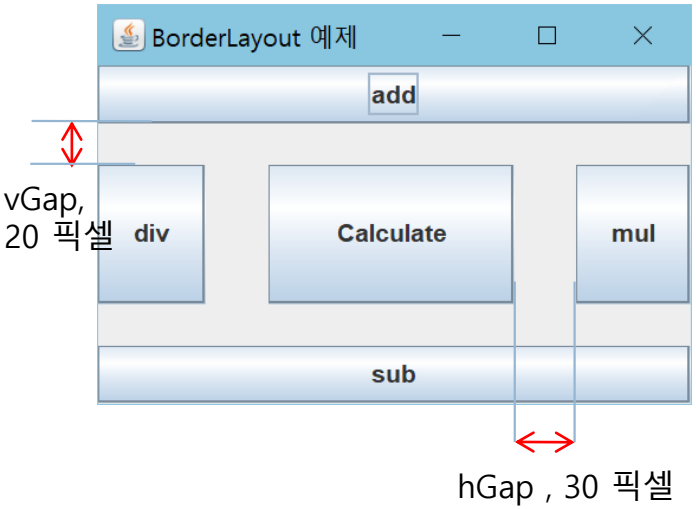
      동 : BorderLayout.EAST　　서 : BorderLayout.WEST

      남 : BorderLayout.SOUTH　　북 : BorderLayout.NORTH

      중앙 : BorderLayout.CENTER

예제 8-4 : BorderLayout 배치관리자 활용

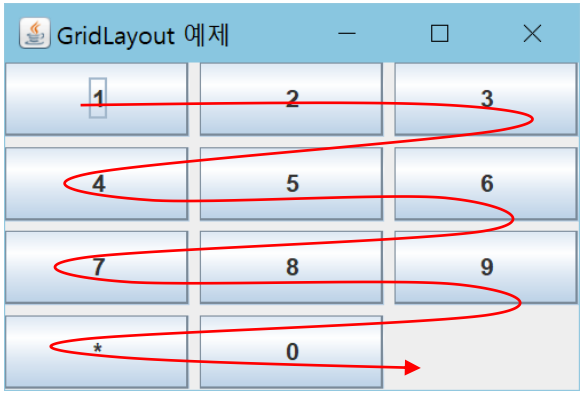
BorderLayout 배치관리자를 사용하여 다음 그림과 같이 5개의 버튼을 배치하라.




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

public class BorderLayoutEx extends JFrame {
    piublic BorderLayoutEx() {
        setTitle("BorderLayout 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container contentPane = getContentPane(); // 컨텐트팬 알아내기

        // 컨텐트팬에 BorderLayout 배치관리자 설정
        contentPane.setLayout(new BorderLayout(30, 20));

        contentPane.add(new JButton("Calculate"), BorderLayout.CENTER);
        contentPane.add(new JButton("add"), BorderLayout.NORTH);
        contentPane.add(new JButton("sub"), BorderLayout.SOUTH);
        contentPane.add(new JButton("mul"), BorderLayout.EAST);
        contentPane.add(new JButton("div"), BorderLayout.WEST);

        setSize(300, 200); // 프레임 크기 300x200 설정
        setVisible(true); // 프레임을 화면에 출력
    }

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

GridLayout 배치관리자

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




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

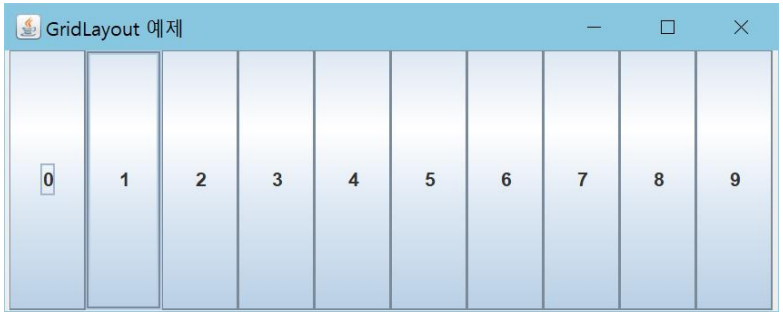
- 4x3 그리드 레이아웃 설정
- 총 11 개의 버튼이 순서대로 add 됨
- 수직 간격 vGap : 5 픽셀
- 수평 간격 hGap : 5 픽셀

GridLayout 생성자

- 생성자
  - GridLayout()
  - GridLayout(int rows, int cols)
  - GridLayout(int rows, int cols, int hGap, int vGap)
    - rows : 격자의 행수 (디폴트 : 1)
    - cols : 격자의 열수 (디폴트 : 1)
    - hGap : 좌우 두 컴포넌트 사이의 수평 간격, 픽셀 단위(디폴트 : 0)
    - vGap : 상하 두 컴포넌트 사이의 수직 간격, 픽셀 단위(디폴트 : 0)
    - rows x cols 만큼의 셀을 가진 격자로 컨테이너 공간을 분할, 배치

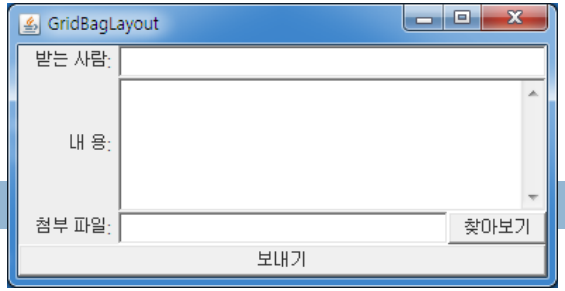
예제 8-5 : GridLayout 배치관리자를 사용하는 예

GridLayout을 활용하여 다음 그림과 같이 한 줄에 10개의 버튼을 동일한 크기로 배치하는 스윙 프로그램을 작성하라.




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

public class GridLayoutEx extends JFrame {
    public GridLayoutEx() {
        super("GridLayout 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container contentPane = getContentPane();

        // 1x10의 GridLayout 배치관리자
        contentPane.setLayout(new GridLayout(1, 10));

        for(int i=0; i<10; i++) { // 10개의 버튼 부착
            String text = Integer.toString(i); // i를 문자열로 변환
            JButton button = new JButton(text);
            contentPane.add(button); // 컨텐트팬에 버튼 부착
        }
        setSize(500, 200);
        setVisible(true);
    }
    public static void main(String[] args) {
        new GridLayoutEx();
    }
}
```

GridBagLayout 배치관리자



- 배치방법
  - GridLayout과 유사하나, 각 영역은 서로 다른 크기로 지정될 수 있으며, 인접한 열 또는 행으로 확장이가능
  - 설정순서

1) GridBagLayout의 오브젝트 생성

```
GridBagLayout gb = new GridBagLayout();
```

2) GridBagConstraints의 오브젝트 생성

```
GridBagConstraints gbc = new GridBagConstraints();
```

3) LayoutManager를 GridBagLayout으로 설정

```
setLayout(gb);
```

4) GridBagConstraints의 멤버 변수 설정

```
gbc.weightx = 1.0; 등
```

5) GridBagLayout에 컴포넌트와 Component에 사용할 Constraints 설정

```
gb.setConstraints(component, gbc);
```

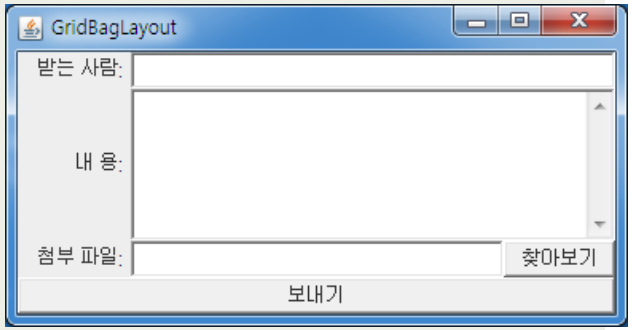
6) Container에 Component붙이기

```
add(component);
```

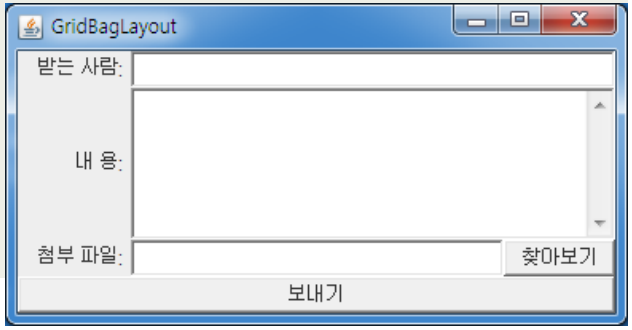
GridBagConstraints 클래스의 멤버 변수

멤버 변수명	기능
gridx gridy	Component가 표시될 격자의 좌표 지정 - 좌측상단은 gridx=0, gridy=0 - 지정하지 않으면 왼쪽에서 오른 쪽으로 차례대로 붙음
gridwidth gridheight	Component가 차지할 격자의 폭과 높이 지정 - 기본값은 1 - REMAINDER : 행의 마지막, 열의 마지막에 위치 - RELATIVE : REMAINDER 전에 위치 - 행과 열의 마지막 컴포넌트 옆에 위치
weightx weighty	Component가 크기를 비율로 지정 - 0 : Container 크기가 변해도 원래 크기 유지 - 0 이외의 값 : 같은 행에 있는 Component간의 비율 계산
fill	Component가 격자보다 작을 때의 처리 지정 - NONE : Component크기 유지 - BOTH :격자 크기에 맞춤 - HORIZONTAL : 수평만 맞춤 - VERTICAL : 수직만 맞춤
ipadx ipady	Component와 격자 사이의 거리
insets	격자와 격자 사이의 거리
anchor	격자 안에서의 Component 위치 - CENTER, NORTH, SOUTH, EAST, WEST 등

```
import java.awt.*;
import javax.swing.*;
class GridBagFrame extends JFrame
{
    //삽입할 컴포넌트 생성
    private Label lblReceive = new Label("받는 사람: ", Label.RIGHT);
    private Label lblContent = new Label("내 용: ", Label.RIGHT);
    private Label lblFile = new Label("첨부 파일: ", Label.RIGHT);
    private Button btnSearch = new Button("찾아보기");
    private Button btnSend = new Button("보내기");
    private TextField toPerson = new TextField(40);
    private TextField file = new TextField(30);
    private TextArea content = new TextArea(5, 40);
    //GridBagLayout 생성
    private GridBagLayout gBag = new GridBagLayout();

    public GridBagFrame()
    {
        this.setTitle("GridBagLayout");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLayout(gBag);
        //GridBagLayout에 삽입  (c, x, y, w, h)
        this.insert(lblReceive,   0, 0, 1, 1);
        this.insert(toPerson,     1, 0, 3, 1);
        this.insert(lblContent,   0, 1, 1, 1);
        this.insert(content,      1, 1, 3, 1);
        this.insert(lblFile,      0, 2, 1, 1);
        this.insert(file,         1, 2, 2, 1);
        this.insert(btnSearch,    3, 2, 1, 1);
        this.insert(btnSend,      0, 3, 4, 1);
```



```
        //this.setSize(400, 200);
        this.pack();
        this.setLocationRelativeTo(this.getOwner());
        this.setVisible(true);
        toPerson.requestFocus();
    }

    private void insert(Component cmpt, int x, int y, int w, int h)
    {
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.fill = GridBagConstraints.BOTH
        gbc.gridx = x
        gbc.gridy = y
        gbc.gridwidth = w
        gbc.gridheight = h
        this.gBag.setConstraints(cmpt, gbc);
        this.add(cmpt);
    }
}

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



배치관리자 없는 컨테이너

- 배치관리자가 없는 컨테이너가 필요한 경우
  - 응용프로그램에서 직접 컴포넌트의 크기와 위치를 결정하고자 하는 경우
    1. 컴포넌트의 크기나 위치를 개발자 임의로 결정하고자 하는 경우
    2. 게임 프로그램과 같이 시간이나 마우스/키보드의 입력에 따라 컴포넌트들의 위치와 크기가 수시로 변하는 경우
    3. 여러 컴포넌트들이 서로 겹쳐 출력하고자 하는 경우

- 컨테이너의 배치 관리자 제거 방법
  - container.setLayout(null);

```
JPanel p = new JPanel();
p.setLayout(null); // JPanel의 배치관리자 삭제
```

  - 컨테이너의 배치관리자가 없어지면, 컴포넌트에 대한 어떤 배치도 없음
    - 추가된 컴포넌트의 크기가 0으로 설정, 위치는 예측할 수 없게 됨

```
// 패널 p에는 배치관리자가 없으면 아래 두 버튼은 배치되지 않는다.

p.add(new JButton("click")); // 폭과 높이가 0인 상태로 화면에 보이지 않는다.
p.add(new JButton("me!")); // 폭과 높이가 0인 상태로 화면에 보이지 않는다.
```

컴포넌트의 절대 위치와 크기 설정

- 배치관리자가 없는 컨테이너에 컴포넌트를 삽입할 때
  - 프로그램에서 컴포넌트의 절대 크기와 위치 설정
  - 컴포넌트들이 서로 겹치게 할 수 있음

- 컴포넌트의 크기와 위치 설정 메소드
  - void setSize(int width, int height) // 컴포넌트 크기 설정
  - void setLocation(int x, int y) // 컴포넌트 위치 설정
  - void setBounds(int x, int y, int width, int height) // 위치와 크기 동시 설정

  - 예) 버튼을 100×40 크기로 하고, JPanel의 (50, 50) 위치에 배치

```
JPanel p = new JPanel();
p.setLayout(null); // 패널 p의 배치관리자 제거

JButton clickButton = new JButton("Click");
clickButton.setSize(100, 40);                    // 버튼 크기를 100×40으로 지정
clickButton.setLocation(50, 50);                 // 버튼 위치를 (50, 50)으로 지정
p.add(clickButton);                  // 패널 내 (50, 50)에 100×40 크기의 버튼 출력
```

예제 8-6 : 배치관리자 없는 컨테이너에 컴포넌트를 절대 위치와 절대 크기로 지정

다음 그림과 같이 컨텐트팬에 배치관리자를 삭제하고 9개의 버튼과 하나의 문자열을 출력하는 프로그램을 작성하라.


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

public class NullContainerEx extends JFrame {
    public NullContainerEx() {
        setTitle("배치관리자 없이 절대 위치에 배치하는 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container contentPane = getContentPane();
        contentPane.setLayout(null); // 컨텐트팬의 배치관리자 제거

        JLabel la = new JLabel("Hello, Press Buttons!");
        la.setLocation(130, 50); // la를 (130,50) 위치로 지정
        la.setSize(200, 20); // la를 200x20 크기로 지정
        contentPane.add(la); // la를 컨텐트팬에 부착

        // 9개의 버튼 컴포넌트를 생성하고 동일한 크기로 설정한다.
        // 위치는 서로 겹치게 설정한다.
        for(int i=1; i<=9; i++) {
            JButton b = new JButton(Integer.toString(i)); // 버튼 생성
            b.setLocation(i*15, i*15); // 버튼의 위치 설정
            b.setSize(50, 20); // 버튼의 크기는 동일하게 50x20
            contentPane.add(b); // 버튼을 컨텐트팬에 부착
        }
        setSize(300, 200);
        setVisible(true);
    }
    public static void main(String[] args) {
        new NullContainerEx();
    }
}
```