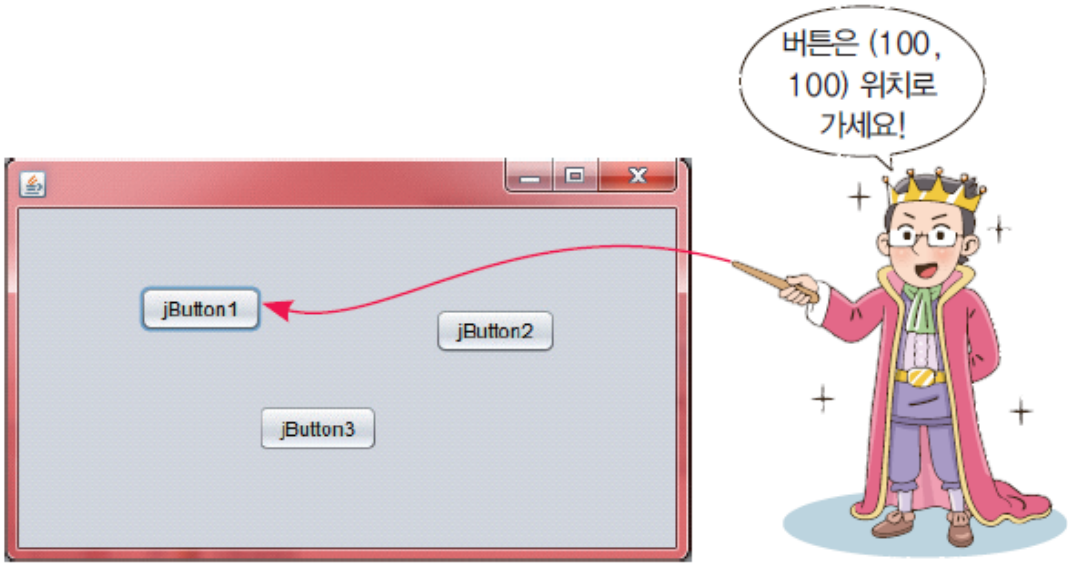
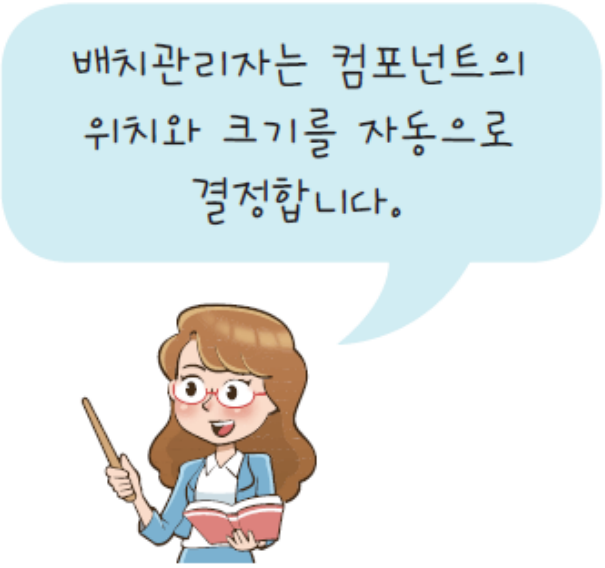
제8장 그래픽 사용자 인터페이스

- 배치 관리자

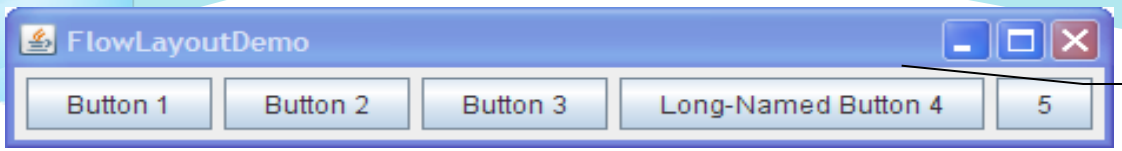
배치 관리자(layout manager)

- 컨테이너 안의 각 컴포넌트의 위치와 크기를 결정하는 작업

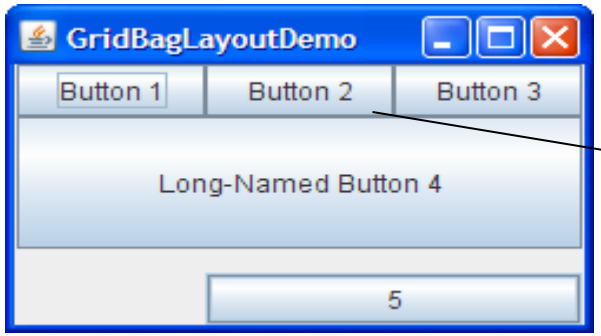




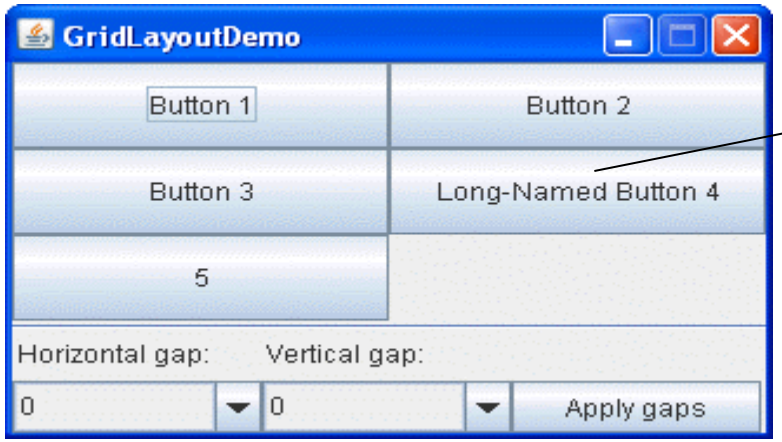
배치 관리자의 종류



FlowLayout

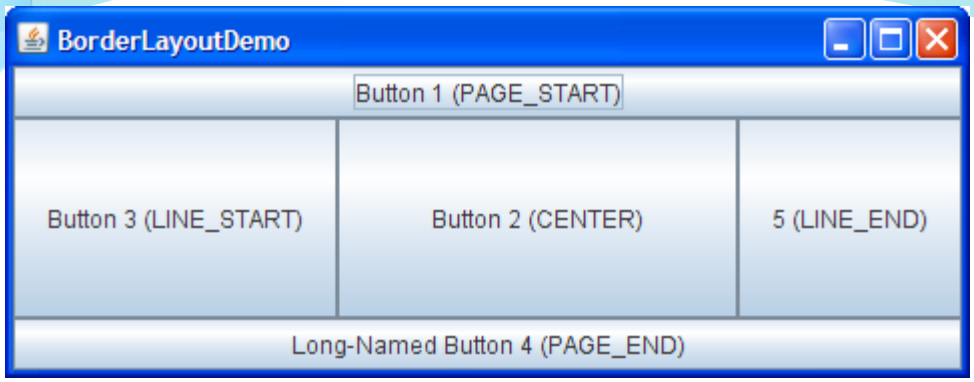


GridBagLayout

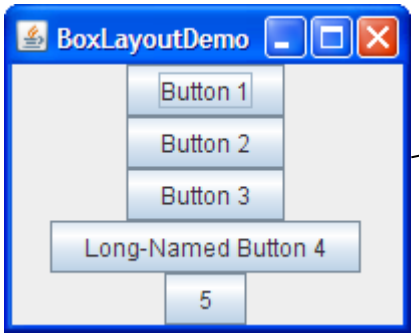


GridLayout

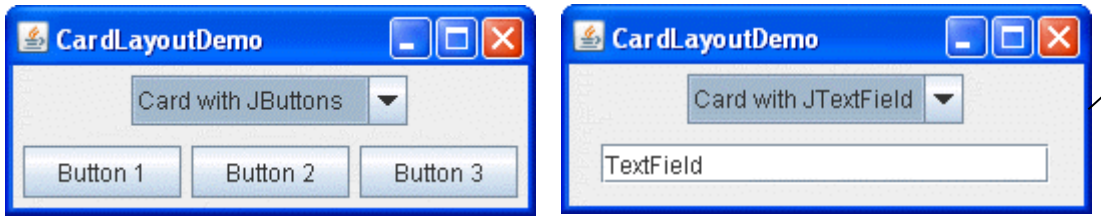
배치 관리자의 종류



BorderLayout



BoxLayout



CardLayout

배치 관리자의 설정

1. 생성자를 이용하는 방법

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

2. setLayout() 메소드 이용

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

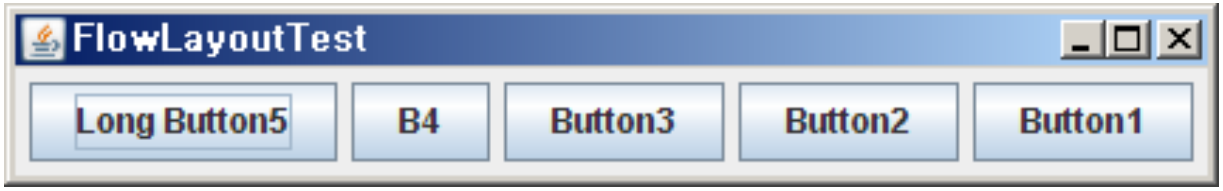
크기와 정렬 힌트

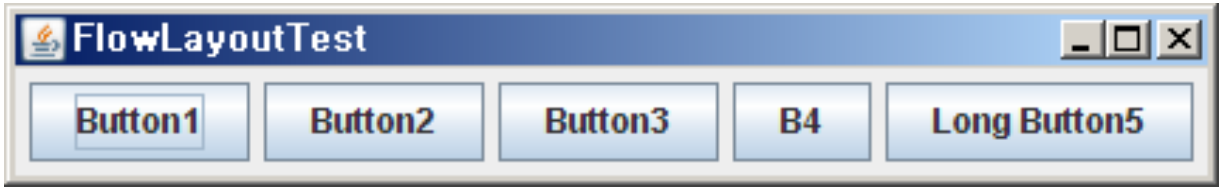
- 프로그래머가 컴포넌트의 크기와 힌트를 배치 관리자에게 주고 싶은 경우에 다음 메소드 사용
  - setMinimumSize()
  - setPreferredSize()
  - setMaximumSize()
- 최대 크기 힌드
  - button.setMaximumSize(new Dimension(300, 200));
- 중앙 정렬 힌트
  - button.setAlignmentX(JComponent.*CENTER_ALIGNMENT*);

배치 방향 설정

- 컴포넌트의 메소드인 setComponentOrientation()을 사용하던지 applyComponentOrientation()을 사용

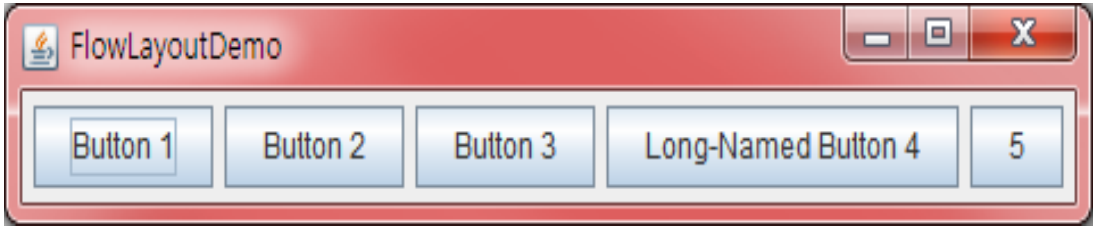
- (예)
  panel.applyComponentOrientation(ComponentOrientation.*RIGHT_TO_LEFT*);





FlowLayout

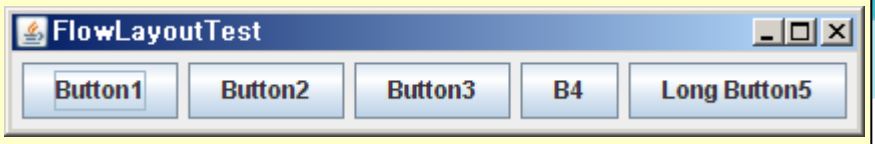
- 컴포넌트들을 왼쪽에서 오른쪽으로 버튼을 배치한다.
- 패널과 애플릿의 디폴트 배치 관리자이다.



FlowLayout 클래스

생성자	설명
FlowLayout()	새로운 FlowLayout 객체를 생성한다. 기본 설정은 중앙(center) 배치이며 간격은 세로, 가로 각각 5 픽셀이다.
FlowLayout(int align)	지정된 정렬 방식을 가진 새로운 FlowLayout 객체를 생성한다. 기본 설정은 중앙(center) 배치이며 간격은 세로, 가로 각각 5 픽셀이다. 정렬 매개 변수는 다음 중 하나이다. FlowLayout.LEADING, FlowLayout.CENTER, FlowLayout.TRAILING.
FlowLayout (int align, int hgap, int vgap)	지정된 정렬 방식과 수평 간격 hgap과 수직 간격 vgap을 가진 새로운 FlowLayout 객체를 생성한다.

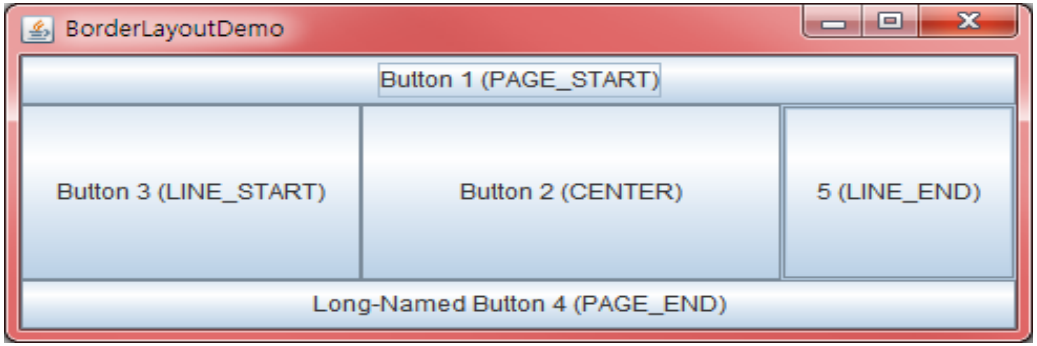
FlowLayout 예제




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

class MyFrame extends JFrame {
	public MyFrame() {

		setTitle("FlowLayoutTest");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		JPanel panel;
		// 패널을 생성하고 배치 관리자를 FlowLayout으로 설정
		panel = new JPanel();
		panel.setLayout(new FlowLayout(FlowLayout.CENTER));
		// 패널에 버튼을 생성하여 추가
		panel.add(new JButton("Button1"));
		panel.add(new JButton("Button2"));
		panel.add(new JButton("Button3"));
		panel.add(new JButton("B4"));
		panel.add(new JButton("Long Button5"));
		add(panel);

		pack();
		setVisible(true);
	}
}
```

BorderLayout

- BorderLayout은 5개의 영역으로 구분하고 각각의 영역에 컴포넌트를 배치

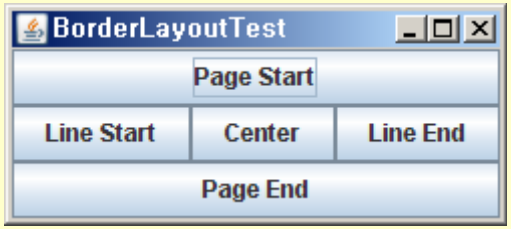


- PAGE_START (또는 NORTH)
- PAGE_END (또는 SOUTH)
- LINE_START (또는 WEST)
- LINE_END (또는 EAST)
- CENTER

BorderLayout 클래스

생성자 또는 메소드	설명
BorderLayout(int hgap, int vgap)	컴포넌트 사이의 수평 간격 hgap과 수직 간격 vgap을 을 가지는 BorderLayout 객체 생성
setHgap(int)	컴포넌트 사이의 수평 간격 설정(단위는 픽셀)
setVgap(int)	컴포넌트 사이의 수직 간격 설정

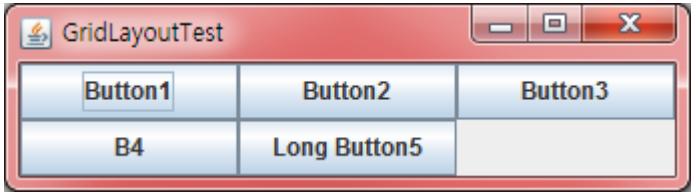
BorderLayout 예제




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

class MyFrame extends JFrame {
        public MyFrame() {

                setTitle("BorderLayoutTest");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                // 프레임은 디폴트로 BorderLayout 이므로 사실은 불필요
                setLayout(new BorderLayout());

                // 버튼을 추가한다.
                add(new JButton("Center"), BorderLayout.CENTER);
                add(new JButton("Line Start"), BorderLayout.LINE_START);
                add(new JButton("Line End"), BorderLayout.LINE_END);
                add(new JButton("Page Start"), BorderLayout.PAGE_START);
                add(new JButton("Page End"), BorderLayout.PAGE_END);

                pack();
                setVisible(true);
        }
}
```

GridLayout

- GridLayout은 컴포넌트들을 격자 모양으로 배치한다.

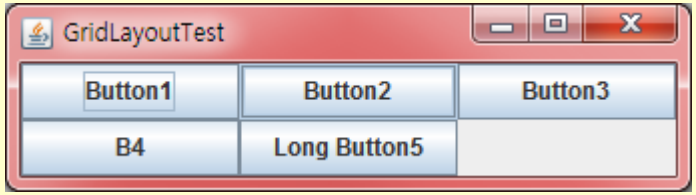


GridLayout 클래스

생성자	설명
GridLayout(**int** rows, **int** cols)	rows 행과 cols 열을 가지는 GridLayout 객체를 생성한다. 만약 rows나 cols가 0이면 필요한 만큼의 행이나 열이 만들어진다.
GridLayout(**int** rows, **int** cols, **int** hgap, **int** vgap)	rows 행과 cols 열을 가지는 GridLayout 객체를 생성한다. hgap과 vgap은 컴포넌트 사이의 수평 간격과 수직 간격으로 단위는 픽셀이다.

GridLayout 예제

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

class MyFrame extends JFrame {
	public MyFrame() {

		setTitle("GridLayoutTest");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		setLayout(new GridLayout(0, 3));    // 3개의 열과 필요한 만큼의 행

		add(new JButton("Button1"));
		add(new JButton("Button2"));
		add(new JButton("Button3"));
		add(new JButton("B4"));
		add(new JButton("Long Button5"));

		pack();
		setVisible(true);
	}
}
```



어떤 배치 관리자를 선택할 것인가?

- 컴포넌트를 가능한 크게 나타내고 싶은 경우
  - GridLayout이나 BorderLayout을 사용
- 몇개의 컴포넌트를 자연스러운 크기로 한줄로 나타내고 싶은 경우
  - FlowLayout을 사용하던지 BoxLayout을 사용한다.
- 몇개의 컴포넌트를 행과 열로 동일한 크기로 나타내고 싶은 경우
  - GridLayout을 사용하여야 한다.
- 몇개의 컴포넌트를 행과 열로 나타내는데 각 컴포넌트의 크기가 다르거나 간격, 정렬 방식을 다르게 내고 싶은 경우
  - BoxLayout을 사용하면 된다.

절대 위치로 배치하기

1. 배치 관리자를 null로 설정한다.
   - setlayout(**null**);

2. add() 메소드를 사용하여 컴포넌트를 컨테이너에 추가한다.
   - Button b = Button("Absolute Position Button");
   - add(b);

3. setBounds() 메소드를 사용하여 절대 위치와 크기를 지정한다.
   - b.setBounds(x, y, w, h);

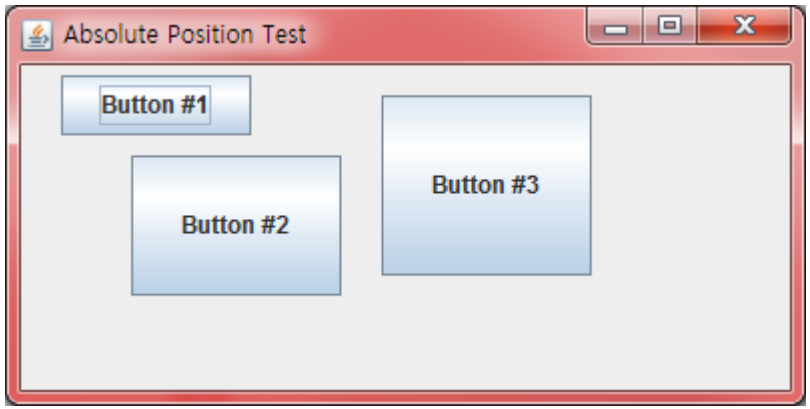
4. 컴포넌트의 repaint() 메소드를 호출한다.
   - b.repaint();

절대 위치 예제

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

class MyFrame extends JFrame {
        JButton b1;
        private JButton b2, b3;

        public MyFrame() {
                setTitle("Absolute Position Test");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setSize(300, 200);
                JPanel p = new JPanel();
                p.setLayout(null);

                b1 = new JButton("Button #1");
                p.add(b1);
                b2 = new JButton("Button #2");
                p.add(b2);
                b3 = new JButton("Button #3");
                p.add(b3);
```

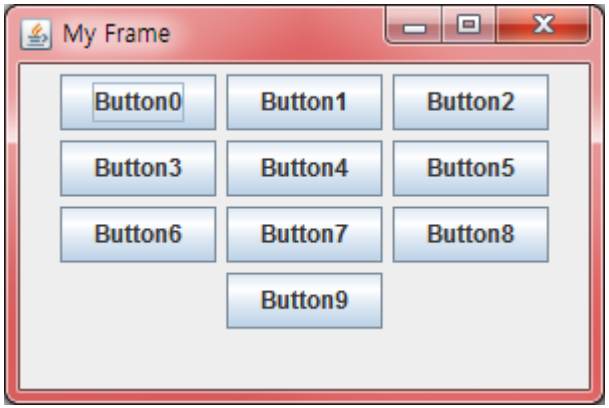
절대 위치 예제

```
            b1.setBounds(20, 5, 95, 30);
            b2.setBounds(55, 45, 105, 70);
            b3.setBounds(180, 15, 105, 90);
            add(p);
            setVisible(true);
        }
}

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



LAB: 계산기 예제

- 실행 결과를 참조하여서 다음 코드의 빈칸을 채우고 실행하여 보라.



SOLUTION

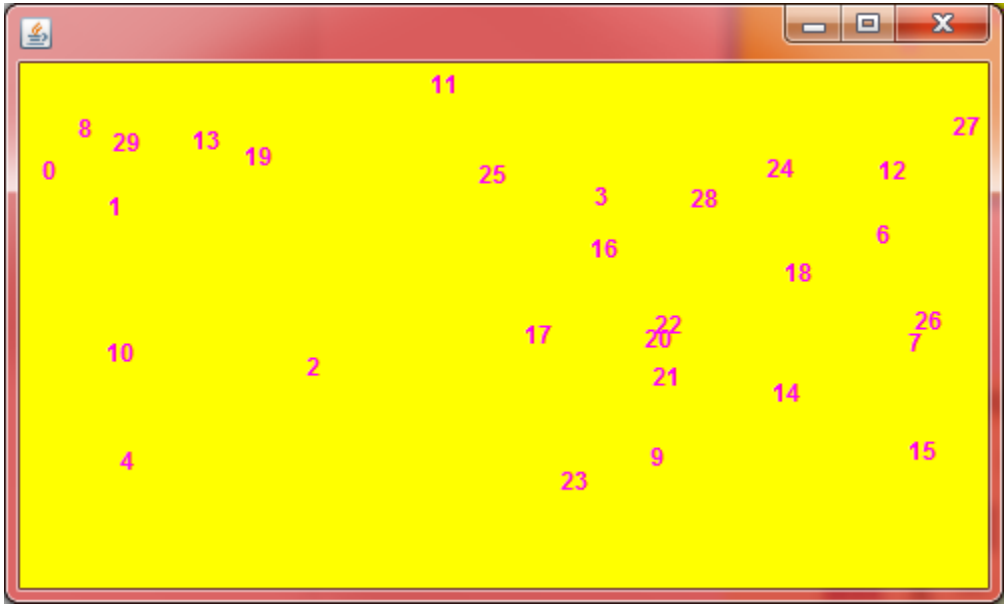
```
import java.awt.FlowLayout;
import javax.swing.*;

public class MyFrame extends JFrame {
  JPanel p1;

  public MyFrame() {
            setSize(300, 200);
            setTitle("My Frame");
            p1 = new JPanel();
            p1.setLayout(new FlowLayout());
            for (int i = 0; i < 10; i++)
                        p1.add(new JButton("Button" + i));
            add(p1);
            setVisible(true); // 프레임을 화면에 표시한다.
  }
}
```

SOLUTION

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

LAB

- 다음과 같이 난수를 발생하여서 레이블을 불규칙하게 배치하여 보자.

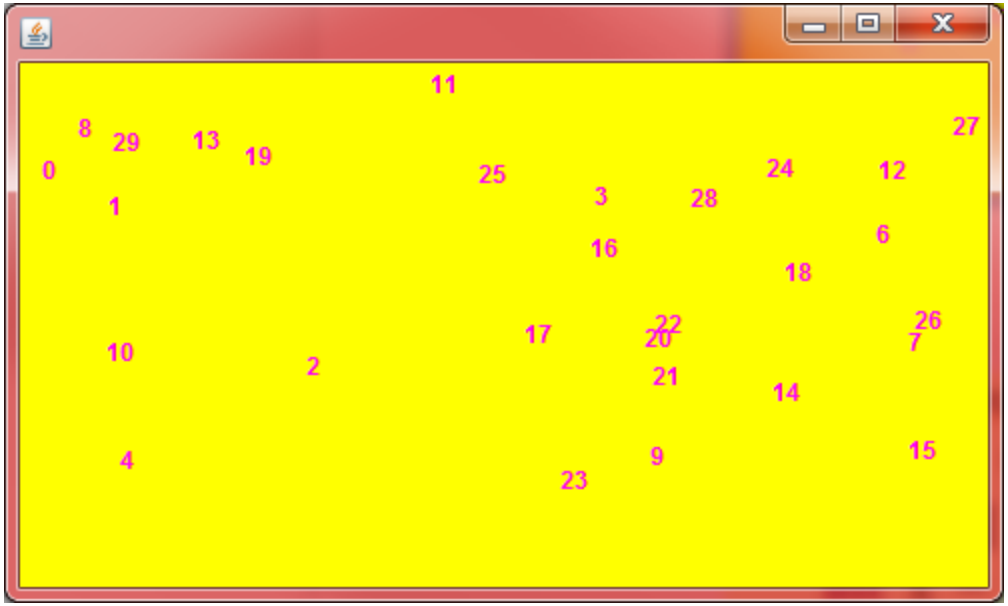


SOLUTION

```
public class MyFrame extends JFrame {
  JPanel p = new JPanel();
  JLabel[] labels = new JLabel[30];

  public MyFrame() {
          p.setLayout(null);
          p.setBackground(Color.YELLOW);
          for (int i = 0; i < 30; i++) {
                    labels[i] = new JLabel("" + i);
                    int x = (int) (500 * Math.random());
                    int y = (int) (200 * Math.random());
                    labels[i].setForeground(Color.MAGENTA);
                    labels[i].setLocation(x, y);
                    labels[i].setSize(20, 20);
                    p.add(labels[i]);
          }
          setSize(500, 300);
          add(p);
          setVisible(true); // 프레임을 화면에 표시한다.
  }
}
```

SOLUTION

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



Q & A