21. 배치관리자

학습목표

• 프로그래머가 원하는 GUI 프로그램을 작성하려면 각 컴포넌트의 크기와 위치를 지정할 수 있어야 한다. 자바에서는 컴포넌트의 위치를 사용자가 지정할 수도 있고 아니면 배치 관리자에 맡길 수 있다. 배치관리자란 자동적으로 컴포넌트의 위치와 크기를 결정하는 객체이다. 이번 장에서는 배치관리자를 사용하여서 원하는 형태대로 컴포넌트를 배치하는 방법에 대하여 학습한다.

1. 배치관리자의 개요

자바에서는 컴포넌트의 배치를 배치관리자(layout manager)를 사용하여 자동화할 수 있다. 배치관리자는 컨테이너 안에 존재하는 컴포넌트들의 크기와 위치를 자동적으로 관리하는 객체이다. java.awt 패키지에는 여러 가지의 배치를 제공하는 배치 관리자가 제공되며 같은 개수의 버튼을 가지고 있더라도 배치관리자에 따라 상당히 달라 보일 수 있다.

• FlowLayout

컴포넌트들을 왼쪽에서 오른쪽으로 버튼을 배치한다. 패널과 애플릿의 디폴트 배치관리자이다.

• BorderLayout

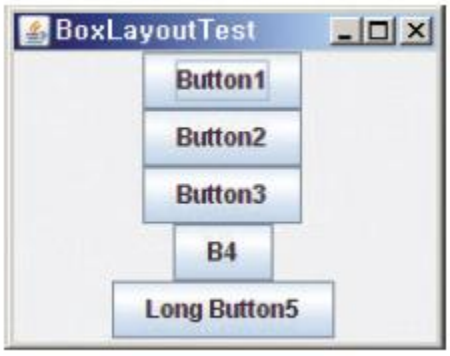
컴포넌트들이 5개의 영역인 North(상), South(하), East(우측), West(좌측), Center(중앙) 중 하나로 추가된다. 프레임, 대화상자와 같은 최상위 컨테이너의 디폴트 배치관리자이다.

• GridLayout

GridLayout은 컴포넌트를 격자 모습으로 배치한다.

• BoxLayout

BoxLayout은 컴포넌트를 하나의 행이나 열에 배치한다. 컴포넌트를 정렬할 수도 있다.



• CardLayout

CardLayout은 여러 장의 카드 위에 컴포넌트를 배치한다. 카드는 변경할 수 있다.

2. 배치관리자를 사용하는 방법

컨테이너 클래스에 배치관리자를 설정하려면 먼저 new 연산자를 이용하여 배치관리자 객체를 만들고 컨테이너 클래스의 setLayout()메소드를 사용하여 배치관리자로 지정한다.

- **생성자를 이용하는 방법 : JPanel panel = new JPanel(new BorderLayout());**
- **setLayout() 메소드 이용 : panel.setLayout(new FlowLayout());**

1) 크기와 정렬 힌트 제공하기

프로그래머가 컴포넌트의 크기와 힌트를 배치 관리자에게 주고 싶은 경우에는 setMinimumSize(), setPreferredSize(), setMaximumSize() 메소드를 사용할 수 있다. 정렬에 대한 힌트를 주려면 setAlignmentX()와 setAlignmentY() 메소드를 이용한다.

```
component.setMaximumSize(new Dimension(Integer.MAX_VALUE,Integer.MAX_VALUE));
button.setMaximumSize(new Dimension(300, 200));                    // 최대 크기 힌트
button.setAlignmentX(JComponent.CENTER_ALIGNMENT);          // 중앙 정렬 힌트
```

2) 배치 방향 설정

컨테이너는 보통 왼쪽에서 오른쪽으로 또 위에서 아래로 컴포넌트를 정렬한다. 하지만 국가에 따라서 방향이 달라질 수 있다. 컨테이너의 방향을 설정하기 위해서는 컴포넌트의 메소드인 setComponentOrientation()을 사용하든지 applyComponentOrientation()을 사용한다.

```
panel.applyComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
```

각각의 컴포넌트가 가능한 모든 영역을 차지하도록 하려면 BorderLayout을 사용해야 한다. 그리고 모든 컴포넌트가 같은 크기를 가지기를 바란다면 GridLayout을 사용해야 한다. 각 컴포넌트를 한 줄로 나열시키고 싶을 경우에는 FlowLayout을 사용한다.

3. FlowLayout 클래스

FlowLayout은 가장 간단한 배치관리자로서 각 컴포넌트들은 하나의 줄에서 차례로 배치되고 더 이상 공간이 없으면 다음 줄에 배치된다. 같은 줄에서 컴포넌트들이 정렬되는 방식(중앙 정렬이 디폴트)도 지정이 가능하다. FlowLayout은 JPanel 클래스의 디폴트 배치관리자이다.

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 매개변수는 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.

(예제)

FlowTest.java

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

class MyFlow extends JFrame {
        public MyFlow() {
                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);
        }
}
public class FlowTest {
        public static void main(String argv[]) {
                MyFlow mf = new MyFlow();
        }
}
```

4. BorderLayout

BorderLayout은 5개의 영역으로 구분하고 각각의 영역에 컴포넌트를 배치할 수 있다.

"North", "South", "East", "West", "Center" 등의 문자열도 사용 할 수 있고 아니면 BorderLayout 안의 정적 필드인 다음과 같은 상수를 사용할 수도 있다.

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

BorderLayout에서 컴포넌트를 추가할 때는 어떤 영역에 추가할 것인지를 지정하여야 한다.

```
panel.add(aComponent, BorderLayout.PAGE_START);   //북쪽에 배치
```

영역에 지정하지 않으면 컴포넌트는 중앙(Center)에 놓여지고 여러 개의 컴포넌트를 같은 영역에 추가하면 마지막으로 추가된 컴포넌트만 표시된다. BorderLayout은 프레임의 디폴트 배치관리자이다.

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

(예제)

BorderTest.java

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

class MyBorder extends JFrame {
	public MyBorder() {
		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);
	}
}

public class BorderTest {
	public static void main(String argv[]) {
		MyBorder	mb = new MyBorder();
	}
}
```

컴포넌트를 추가할 때는 BorderLayout.LINE_END와 같이 컴포넌트가 위치할 영역을 지정하여야 한다. 만약 같은 위치에 컴포넌트들이 추가되면 뒤 컴포넌트에 가려서 앞의 컴포넌트가 보이지 않는다.

5. GridLayout

GridLayout은 컴포넌트들을 격자 모양으로 배치한다. 모든 컴포넌트들의 크기는 같게 되며 컨테이너의 모든 공간은 컴포넌트로 채워진다. 윈도우의 크기를 바꾸면 GridLayout은 컴포넌트의 크기를 변경하여 윈도우의 크기를 맞춘다.

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.

setLayout(new GridLayout(3,0));

(예제)

GridTest.java

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

class MyGrid extends JFrame {
	public MyGrid() {
		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);
	}
}

public class GridTest {
	public static void main(String argv[]) {
		MyGrid mg = new MyGrid();
	}
}
```

6. BoxLayout

BoxLayout은 컴포넌트를 다른 컴포넌트 위에 쌓거나 컴포넌트를 하나의 행에 배치할 수 있다.
BoxLayout은 FlowLayout의 하나의 버전으로 간주할 수 있으나 좀 더 기능이 많다.

(예제)

BoxTest.java

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

class MyBox extends JFrame {
	public MyBox() {
		setTitle("BoxLayoutTest");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JPanel panel = new JPanel();

		// Y축 방향으로 컴포넌트를 쌓는다.
		panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

		makeButton(panel, "Button1");
		makeButton(panel, "Button2");
		makeButton(panel, "Button3");
		makeButton(panel, "B4");
		makeButton(panel, "Long Button5");
		add(panel);
		pack();
		setVisible(true);
	}
	private void makeButton(JPanel panel, String text) {
		JButton button = new JButton(text);
		button.setAlignmentX(Component.CENTER_ALIGNMENT);
		panel.add(button);
	}
}

public class BoxTest {
	public static void main(String args[]) {
		MyBox mb = new MyBox();
	}
}
```

7. CardLayout

CardLayout 클래스는 여러 개의 카드가 쌓여 있는 형태로 컴포넌트를 배치하는 경우에 사용한다. CardLayout 클래스를 사용하였을 경우 항상 맨 위의 컴포넌트만이 보인다.

생성자

생성자	설명
CardLayout()	디폴트 생성자
CardLayout(int hgap, int vgap)	hgap과 vgap은 컴포넌트 사이의 수평 간격과 수직 간격을 픽셀로 표시한다. 기본값은 0이다.

메소드

메소드	설명
first (Container parent)	첫 번째 카드를 선택한다.
next (Container parent)	다음 카드를 선택한다. 만약 현재 선택된 카드가 마지막이면 처음으로 간다.
previous (Container parent)	이전 카드를 선택한다.
last (Container parent)	마지막 카드를 선택한다.
show (Container parent, String s)	문자열 s로 지정한 카드를 선택 선택한다.

(예제)

카드를 순회할 수 있는 사용자 인터페이스를 가지는 예제 프로그램을 만들어보자.

CardTest.java

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

class MyCard extends JFrame implements ActionListener {
	JPanel panel;
	Cards cards;

	public MyCard() {
		setTitle("CardLayoutTest");
		setSize(400, 200);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		panel = new JPanel(new GridLayout(0, 5, 10, 0)); //5개의 열이 있으면 수평간격은 10
		addButton("<<", panel);
		addButton("<", panel);
		addButton(">", panel);
		addButton(">>", panel);
		addButton("종료", panel);
		add(panel, "North");
```

```
                cards = new Cards();
                add(cards, "Center");
                setVisible(true);
        }

        void addButton(String str, Container target) {
                JButton button = new JButton(str);
                button.addActionListener(this);
                target.add(button);
        }

        private class Cards extends JPanel {
                CardLayout layout;

                public Cards() {
                        layout = new CardLayout();
                        setLayout(layout);
                        for (int i = 1; i <= 10; i++) {
                                add(new JButton("현재 카드의 번호는 " + i + "입니다"), "Center");
                        }
                }
        }

        public void actionPerformed(ActionEvent e) {           // 이벤트 처리
                if (e.getActionCommand().equals("종료")) {
                        System.exit(0);
                } else if (e.getActionCommand().equals("<<")) {
                        cards.layout.first(cards);
                } else if (e.getActionCommand().equals("<")) {
                        cards.layout.previous(cards);
                } else if (e.getActionCommand().equals(">")) {
                        cards.layout.next(cards);
                } else if (e.getActionCommand().equals(">>")) {
                        cards.layout.last(cards);
                }
        }
}

public class CardTest {
        public static void main(String args[]) {
                MyCard mc = new MyCard();
```

```
        }
}
```

• 컴포넌트를 가능한 크게 나타내고 싶은 경우
GridLayout이나 BorderLayout을 사용
• 몇개의 컴포넌트를 자연스러운 크기로 한줄로 나타내고 싶은 경우
FlowLayout을 사용하던지 BoxLayout을 사용
• 몇개의 컴포넌트를 행과 열로 동일한 크기로 나타내고 싶은 경우
GridLayout을 사용
• 몇개의 컴포넌트를 행과 열로 나타내는데 각 컴포넌트의 크기가 다르거나 간격, 정렬 방식을 다르게 내고 싶은 경우
BoxLayout을 사용

8. 절대 위치로 배치하기

어떤 경우에는 컴포넌트들을 배치관리자 없이도 배치해야만 하는 경우도 있다. 즉 컨테이너 안의 컴포넌트들의 크기와 위치가 외부의 영향을 받지 않는 경우가 그렇다. 배치관리자는 사용하지 않으려면 다음과 같이 설정하면 된다.

①배치 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();
```

(예제 1)

AbsoluteTest.java

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

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

        public MySetBound() {
                setTitle("Absolute Position Test");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
                setSize(300, 200);
                JPanel p = new JPanel();
                p.setLayout(null);                    // 배치관리자를 설정하지 않는다.

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

                b1.setBounds(20, 5, 95, 30);          // 위치(20,5)에 폭95, 높이 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[]) {
                MySetBound msb = new MySetBound ();
        }
}
```

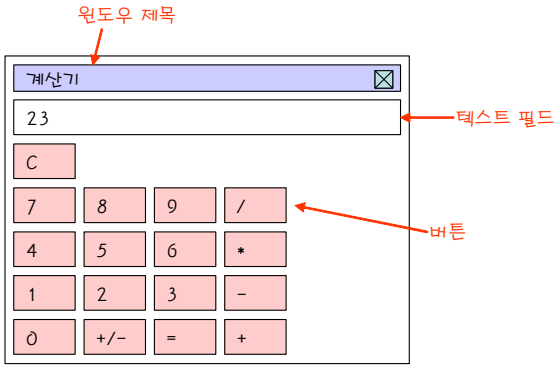
(예제 2)

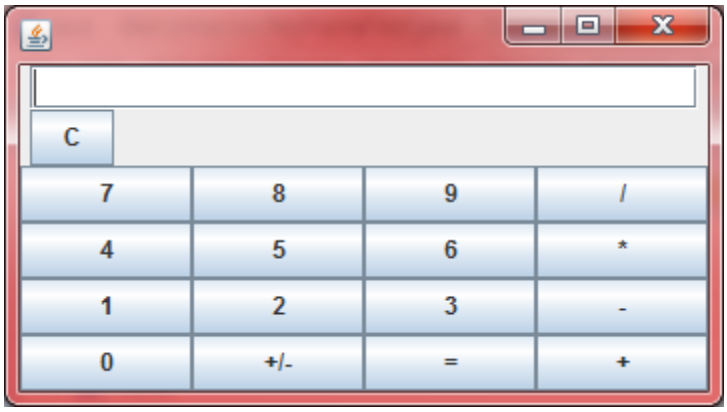
MySetBoundRateTest.java

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

class MySetBoundRate extends JFrame {
        JButton button;
        JTextField t1;
        JTextField t2;
        JTextField t3;
        private JPanel panel,panel1,panel2,panel3, panel4;

        public MySetBoundRate()
        {
                setSize(230,150);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
                setTitle("이자 계산기");

                JPanel panel=new JPanel(new GridLayout(0, 1));
                JPanel panel1=new JPanel();
                JPanel panel2=new JPanel();
                JPanel panel3=new JPanel();
                JPanel panel4=new JPanel();

                JLabel label1=new JLabel("원금을 입력하시오");
                t1=new JTextField(5);
                panel1.add(label1);
                panel1.add(t1);

                JLabel label2=new JLabel("이율을 입력하시오");
                t2=new JTextField(5);
                panel2.add(label2);
                panel2.add(t2);

                button=new JButton("변환");
                panel3.add(button);

                t3=new JTextField(20);
                panel4.add(t3);

                panel.add(panel1);
                panel.add(panel2);
                panel.add(panel3);
                panel.add(panel4);
                this.add(panel);

                setVisible(true);
        }
}
public class MySetBoundRateTest {
        public static void main(String[] arge)
        {
                MySetBoundRateTest msbrt=new MySetBoundRateTest();
        }
}
```

[계산기 setLayout 문제]





[Swing 문제 1]