GUI 프로그래밍 2

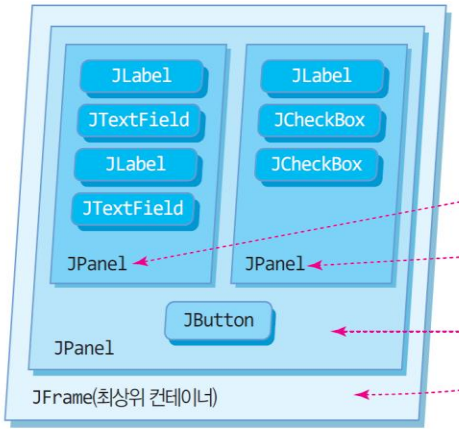
02. Swing으로 살펴보는 GUI 프로그래밍

레이아웃

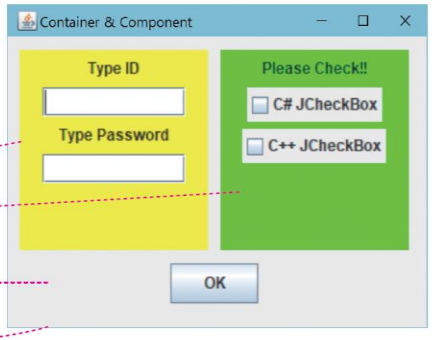
- 컴포넌트를 응용 프로그램에 배치하는 것을 말한다.

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



스윙의 컨테이너와 컴포넌트의 포함 관계



스윙 GUI 프로그램

최상위 컨테이너

- 다른 컨테이너에 속하지 않고 독립적으로 출력가능한 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

02. Swing으로 살펴보는 GUI 프로그래밍

■ 레이아웃 적용 방법

① 먼저 대상 컨테이너를 선택한 후 setLayout( ) 메서드로 원하는 레이아웃 객체를 생성해야 한다.

```
Frame frame = new Frame();
frame.setLayout(new BorderLayout());
```

② add( ) 메서드를 사용하여 컴포넌트들을 컨테이너에 추가

```
Button button1 = new Button("버튼1");
frame.add(button1, BorderLayout.WEST);
```

■ 레이아웃 : FlowLayout

- FlowLayout은 가장 기본적인 레이아웃으로, 일정한 높이와 간격이 있는 컴포넌트를 가로로 배열할 때 많이 사용.
- 버튼을 배열하거나 텍스트 박스를 연결해서 배열할 때 유용.

그림 8-7 FlowLayout 구조

02. Swing으로 살펴보는 GUI 프로그래밍

■ 레이아웃 : FlowLayout

표 8-2 FlowLayout 클래스의 주요 필드와 생성자, 메서드

필드	static int CENTER	각 행의 컴포넌트가 가운데 위치하게 한다.
	static int LEANDING	각 행의 컴포넌트가 컨테이너의 리딩 엣지(컨테이너의 왼쪽 단)에 정렬되게 한다.
	static int LEFT	각 행의 컴포넌트를 왼쪽으로 정렬한다.
	static int RIGHT	각 행의 컴포넌트를 오른쪽으로 정렬한다.
	static int TRAILING	각 행의 컴포넌트를 컨테이너 트레이닝 엣지(컨테이너의 오른쪽 단)에 정렬되게 한다.
생성자	FlowLayout( )	왼쪽/오른쪽으로 5단위 간격만큼 가운데에 위치시켜 FlowLayout을 생성한다.
	FlowLayout(int align)	align(정렬 속성)값에 따라 왼쪽/오른쪽으로 5단위 간격이 있는 FlowLayout을 생성한다.

02. Swing으로 살펴보는 GUI 프로그래밍

	FlowLayout(int align, int hgap, int vgap)	align(정렬 속성), hgap(수평 간격), vgap(수직 간격)이 있는 FlowLayout을 생성한다.
메서드	int getAlignment( )	정렬 상태값을 반환한다.
	int getHgap( )	수평 간격 설정값을 반환한다.
	int getVgap( )	수직 간격 설정값을 반환한다.
	void setAlignment(int align)	정렬 상태를 설정한다.
	void setHgap(int hgap)	수평 간격을 설정한다.
	void setVgap(int vgap)	수직 간격을 설정한다.

02. Swing으로 살펴보는 GUI 프로그래밍

레이아웃 : GridLayout

- GridLayout은 격자 모양의 배열을 지원하는 레이아웃이다. (예 : 계산기)

그림 8-8 GridLayout 구조

02. Swing으로 살펴보는 GUI 프로그래밍

■ 레이아웃 : GridLayout

표 8-3 GridLayout 클래스의 주요 생성자와 메서드

생성자	GridLayout( )	1행 1열의 GridLayout을 생성한다.
	GridLayout(int rows, int cols)	rows만큼 행과 cols만큼 열이 있는 GridLayout을 생성한다.
	GridLayout(int rows, int cols, int hgap, int vgap)	rows만큼 행, cols만큼 열, hgap만큼 수평 간격, vgap만큼 수직 간격이 있는 GridLayout을 생성한다.
메서드	void addLayoutComponent(String name, Component comp)	name 문자열이 있는 comp 컴포넌트를 추가한다.
	int getColumns( )	전체 열 개수를 반환한다.
	int getRows( )	전체 행 개수를 반환한다.
	int getHgap( )	수평 간격 설정값을 반환한다.
	void layoutContainer(Container parent)	parent 컨테이너를 배치한다.

02. Swing으로 살펴보는 GUI 프로그래밍

void setColumns(int cols)	전체 열 개수를 설정한다.
void setRows(int rows)	전체 행 개수를 설정한다.
void setHgap(int hgap)	수평 간격을 설정한다.
void setVgap(int vgap)	수직 간격을 설정한다.

02. Swing으로 살펴보는 GUI 프로그래밍

■ 레이아웃 : BorderLayout

- BorderLayout은 가운데에 있는 하나의 컴포넌트를 중심으로 네 방향에 컴포넌트를 배치하는 레이아웃이다.

그림 8-9 BorderLayout 구조

02. Swing으로 살펴보는 GUI 프로그래밍

■ 레이아웃 : BorderLayout

표 8-4 BorderLayout 클래스의 주요 필드와 생성자, 메서드

필드	static String CENTER	컨테이너의 가운데로 제약한다.
	static String EAST	컨테이너의 오른쪽으로 제약한다.
	static String LINE_END	행 방향 마지막에 위치한다.
	static String LINE_START	행 방향 맨 앞쪽에 위치한다.
	static String NORTH	컨테이너의 위쪽으로 제약한다.
	static String PAGE_END	맨 마지막 줄의 끝에 위치한다.
	static String PAGE_START	왼쪽 위 맨 앞쪽 행에 위치한다.
	static String SOUTH	컨테이너의 아래쪽으로 제약한다.
	static String WEST	컨테이너의 왼쪽으로 제약한다.

02. Swing으로 살펴보는 GUI 프로그래밍

생성자	BorderLayout( )	컴포넌트 간에 간격 없이 배치되는 BorderLayout을 생성한다.
	BorderLayout(int hgap, int vgap)	컴포넌트 간에 hgap만큼 수평 간격, vgap만큼 수직 간격이 있도록 BorderLayout을 생성한다.
메서드	int getHgap( )	수평 간격 설정값을 반환한다.
	int getVgap( )	수직 간격 설정값을 반환한다.
	void setHgap(int hgap)	수평 간격을 설정한다.
	void setVgap(int vgap)	수직 간격을 설정한다.

02. Swing으로 살펴보는 GUI 프로그래밍

■ 레이아웃 : CardLayout

- CardLayout은 등록된 컴포넌트 여러 개 중에서 하나의 컴포넌트만 보여 주는 레이아웃을 말한다

그림 8-10 CardLayout 구조

02. Swing으로 살펴보는 GUI 프로그래밍

■ 레이아웃 : CardLayout

표 8-5 CardLayout 클래스의 주요 생성자와 메서드

생성자	CardLayout( )	간격이 없는 CardLayout을 생성한다.
	CardLayout(int hgap, int vgap)	hgap만큼 수평 간격, vgap만큼 수직 간격이 있는 CardLayout을 생성한다.
메서드	int first(Container parent)	컨테이너의 처음 카드로 전환한다.
	int getHgap( )	수평 간격 설정값을 반환한다.
	int getVgap( )	수직 간격 설정값을 반환한다.
	void last(Container parent)	컨테이너의 마지막 카드로 전환한다.
	void next(Container parent)	컨테이너의 현재 카드에서 다음 카드로 전환한다.
	void previous(Container parent)	컨테이너의 현재 카드에서 이전 카드로 전환환다.

02. Swing으로 살펴보는 GUI 프로그래밍

void removeLayoutComponent (Component comp)	comp 컴포넌트를 레이아웃에서 삭제한다.
void setHgap(int hgap)	컴포넌트 간의 수평 간격을 설정한다.
void setVgap(int vgap)	컴포넌트 간의 수직 간격을 설정한다.
void show(Container parent, String name)	name 문자열이 있는 컴포넌트로 전환한다.

02. Swing으로 살펴보는 GUI 프로그래밍

■ 레이아웃 예제

예제 8-4 레이아웃 종합 프로그램 만들기 Ch8Ex4.java

```
01  package javabook.ch8;
02
03  import java.awt.*;
04  import java.awt.event.*;
05  import javax.swing.*;
06
07  public class Ch8Ex4 extends JFrame {
08      JButton button1 = new JButton("버튼1");
09      JButton button2 = new JButton("버튼2");
10      JButton button3 = new JButton("버튼3");
11      JButton button4 = new JButton("버튼4");
12      JButton button5 = new JButton("버튼5");
13
14      Panel p1 = new Panel();
15      Panel p2 = new Panel();
16
17      public void flowLayout() {
18          p1.setLayout(new FlowLayout());
19          p1.add(button1);
20          p1.add(button2);
21          p1.add(button3);
22          p1.add(button4);
23      }
24
25      public void gridLayout() {
26          p1.setLayout(new GridLayout(2, 2));
27          p1.add(button1);
28          p1.add(button2);
29          p1.add(button3);
30          p1.add(button4);
31      }
32
```

02. Swing으로 살펴보는 GUI 프로그래밍

```
33      public void borderLayout() {
34          p1.setLayout(new BorderLayout());
35          p1.add(button1, BorderLayout.NORTH);
36          p1.add(button2, BorderLayout.WEST);
37          p1.add(button3, BorderLayout.EAST);
38          p1.add(button4, BorderLayout.SOUTH);
39          p1.add(button5, BorderLayout.CENTER);
40      }
41
42      public void cardLayout() {
43          final CardLayout card = new CardLayout();
44          setLayout(card);
45          p1.add(button1);
46          p1.add(button2);
47          p1.add(button3);
48          p2.add(button4);
49          p2.add(button5);
50          add("p1", p1);
51          add("p2", p2);
52
53          button3.addActionListener(new ActionListener() {
54              @Override
55              public void actionPerformed(ActionEvent arg0) {
56                  card.show(getContentPane(), "p2");
57              }
58          });
59      }
```

02. Swing으로 살펴보는 GUI 프로그래밍

```
60
61      public Ch8Ex4() {
62          super("Layout Showcase");
63          getContentPane().add(p1);
64          flowLayout();
65          // gridLayout();
66          // borderLayout();
67          // cardLayout();
68          setDefaultCloseOperation(EXIT_ON_CLOSE);
69          setSize(300, 200);
70          setVisible(true);
71      }
72
73      public static void main(String[] args) {
74          Ch8Ex4 app = new Ch8Ex4();
75      }
76  }
```

테스트할 레이아웃만 남기고 주석 처리

02. Swing으로 살펴보는 GUI 프로그래밍

■ 이벤트 핸들링

- 이벤트는 GUI 프로그램에서 발생하는 특정 시점의 의미 있는 신호를 말하는 것으로, 마우스나 키보드 입력과 같이 사용자가 입력한 결과를 응용 프로그램에 전달하면서 발생한다.

그림 8-11 이벤트 처리 과정

02. Swing으로 살펴보는 GUI 프로그래밍

■ 이벤트 핸들링

표 8-6 이벤트 클래스의 종류

종류	설명
ActionEvent	컴포넌트에 정의된 행위가 발생했을 때 나타나는 이벤트이다. addActionListener 메서드로 등록하며, ActionListener에 이벤트를 넘긴다.
MouseEvent	컴포넌트에 마우스 액션이 발생할 때 나타나는 이벤트이다. addMouseListener 메서드로 등록하며, MouseListener, MouseAdapterListener에 이벤트를 전달한다.
MouseWheelEvent	컴포넌트 안에서 마우스 휠을 회전했을 때 나타나는 이벤트이다. addMouseWheelListener 메서드로 등록하며, MouseWheelListener에 이벤트를 전달한다.
KeyEvent	컴포넌트 안에서 키 입력으로 나타나는 이벤트이다. addKeyListener 메서드로 등록하며, KeyListener에 이벤트를 전달한다.

02. Swing으로 살펴보는 GUI 프로그래밍

ComponentEvent	컴포넌트의 이동, 크기 변경 등 변화가 있을 때 나타나는 이벤트이다. addComponentListener 메서드로 등록하며, ComponentListener에 이벤트를 전달한다.
ContainerEvent	컴포넌트를 추가하거나 삭제할 때 컨테이너에 변화가 발생하면 나타나는 이벤트이다. addContainerListener 메서드로 등록하며, ContainerListener에 이벤트를 전달한다.
ItemEvent	항목을 선택하거나 해제할 때 나타나는 이벤트이다. addItemListener 메서드로 등록하며, ItemListener에 이벤트를 전달한다.

02. Swing으로 살펴보는 GUI 프로그래밍

■ 이벤트 처리 구현 절차

- 이벤트는 각 컴포넌트에서 발생하며, 이벤트 핸들러에서 처리한다.

[일반적인 이벤트 처리 구현 절차]

- 이벤트 처리 구현 방식에 따라 이벤트를 처리할 핸들러를 사전에 정의한다.
- 이벤트 처리가 필요한 컴포넌트 대상을 정하고, 해당 컴포넌트에 이벤트 핸들러를 등록한다.
- 이벤트 핸들러 클래스에서 요구하는 메서드를 오버라이딩하고, 이벤트가 발생할 때 필요한 기능을 구현한다.

그림 8-12 버튼 이벤트 처리 과정

02. Swing으로 살펴보는 GUI 프로그래밍

■ 이벤트 처리 구현 방식

1. 프로그램 클래스가 핸들러 인터페이스를 구현
2. 별도의 이벤트 핸들러 클래스로 구현
3. 현재 클래스 파일에 내부 클래스로 구현
4. 컴포넌트에 핸들러를 등록할 때 익명의 내부 클래스로 구현

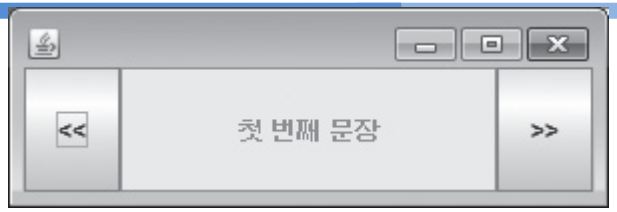
02. Swing으로 살펴보는 GUI 프로그래밍

■ 이벤트 처리 예제 (1)

예제 8-5 이벤트 처리 예제 : 프로그램 클래스로 핸들러 구현하기 Ch8Ex5.java

```
01  package javabook.ch8;
02
03  import java.awt.*;
04  import java.awt.event.*;
05  import javax.swing.*;
06
07  public class Ch8Ex5 extends JFrame implements ActionListener {
08      int index = 0;
09      String[] msgs = {"첫 번째 문장", "두 번째 문장", "세 번째 문장"};
10      JButton button1 = new JButton("<<");
11      JButton button2 = new JButton(">>");
12      JButton button3 = new JButton(msgs[0]);
13
14      public Ch8Ex5() {
15          BorderLayout layout = new BorderLayout();
```

02. Swing으로 살펴보는 GUI 프로그래밍




```
16          setLayout(layout);
17          button1.addActionListener(this);
18          button2.addActionListener(this);
19          button3.setEnabled(false);
20          add(button1, BorderLayout.WEST);
21          add(button2, BorderLayout.EAST);
22          add(button3, BorderLayout.CENTER);
23          setDefaultCloseOperation(EXIT_ON_CLOSE);
24          setSize(300, 100);
25          setVisible(true);
26      }
27
28      @Override
29      public void actionPerformed(ActionEvent e) {
30          Object obj = e.getSource();
31          if(obj == button1) {
32              index--;
33          }
34          else if(obj == button2) {
35              index++;
36          }
37          if(index > 2)
                index = 0;
            else if(index < 0) {
                index = 2;
            }
            button3.setText(msgs[index]);
        }

        public static void main(String[] args) {
            Ch8Ex5 app = new Ch8Ex5();
        }
    }
```

02. Swing으로 살펴보는 GUI 프로그래밍

■ 이벤트 처리 예제 (2)

예제 8-6 이벤트 처리 예제 : 익명의 내부 클래스로 구현하기 Ch8Ex6.java

```
01  package javabook.ch8;
02
03  import java.awt.*;
04  import java.awt.event.*;
05  import javax.swing.*;
06
07  public class Ch8Ex6 extends JFrame {
08      int index = 0;
09      String[] msgs = {"첫 번째 문장", "두 번째 문장", "세 번째 문장"};
10      JButton button1 = new JButton("<<");
11      JButton button2 = new JButton(">>");
12      JButton button3 = new JButton(msgs[0]);
13
14      public Ch8Ex6() {
15          BorderLayout layout = new BorderLayout();
16          setLayout(layout);
17          button1.addActionListener(new ActionListener() {
```

02. Swing으로 살펴보는 GUI 프로그래밍

```
18              @Override
19              public void actionPerformed(ActionEvent e) {
20                  index--;
21                  changeText();
22              }
23          });
24          button2.addActionListener(new ActionListener() {
25              @Override
26              public void actionPerformed(ActionEvent e) {
27                  index++;
28                  changeText();
29              }
30          });
31          button3.setEnabled(false);
32          add(button1, BorderLayout.WEST);
33          add(button2, BorderLayout.EAST);
34          add(button3, BorderLayout.CENTER);
35          setDefaultCloseOperation(EXIT_ON_CLOSE);
36          setSize(300, 100);
```

02. Swing으로 살펴보는 GUI 프로그래밍

```
37          setVisible(true);
38      }
39
40      public void changeText() {
41          if(index > 2)
42              index = 0;
43          else if(index < 0) {
44              index = 2;
45          }
46          button3.setText(msgs[index]);
47      }
48
49      public static void main(String[] args) {
50          Ch8Ex6 app = new Ch8Ex6();
51      }
52  }
```