JAVA 입문 : 이론과 실습

제 12장
그래픽 사용자 인터페이스

차례

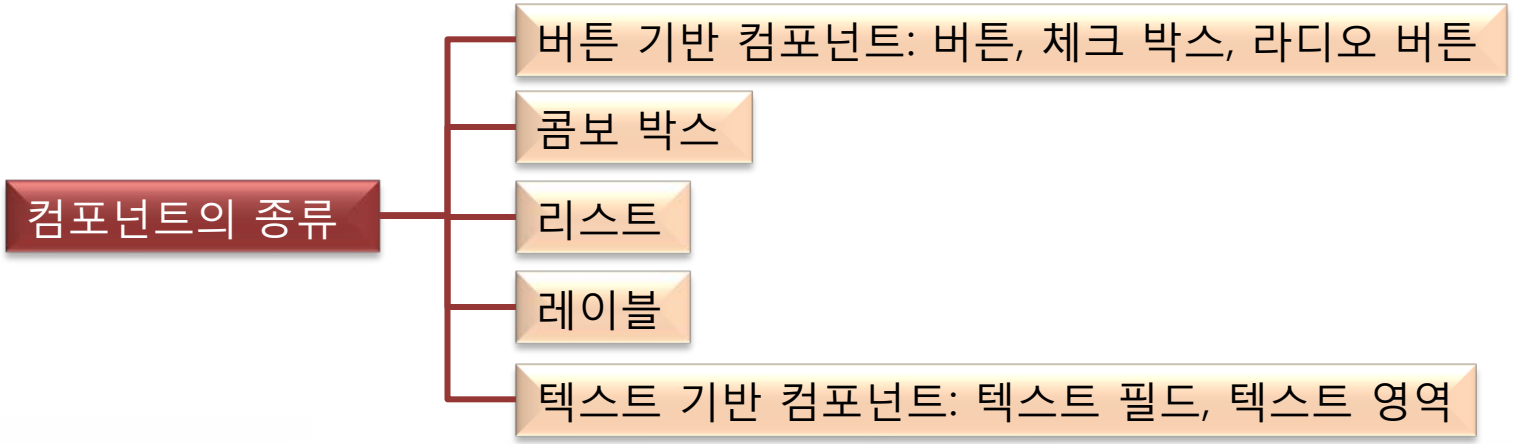
- 컴포넌트(Component)
- 컨테이너(Container)
- 레이아웃(Layout)
- 이벤트 처리
- GUI 프로그래밍

GUI

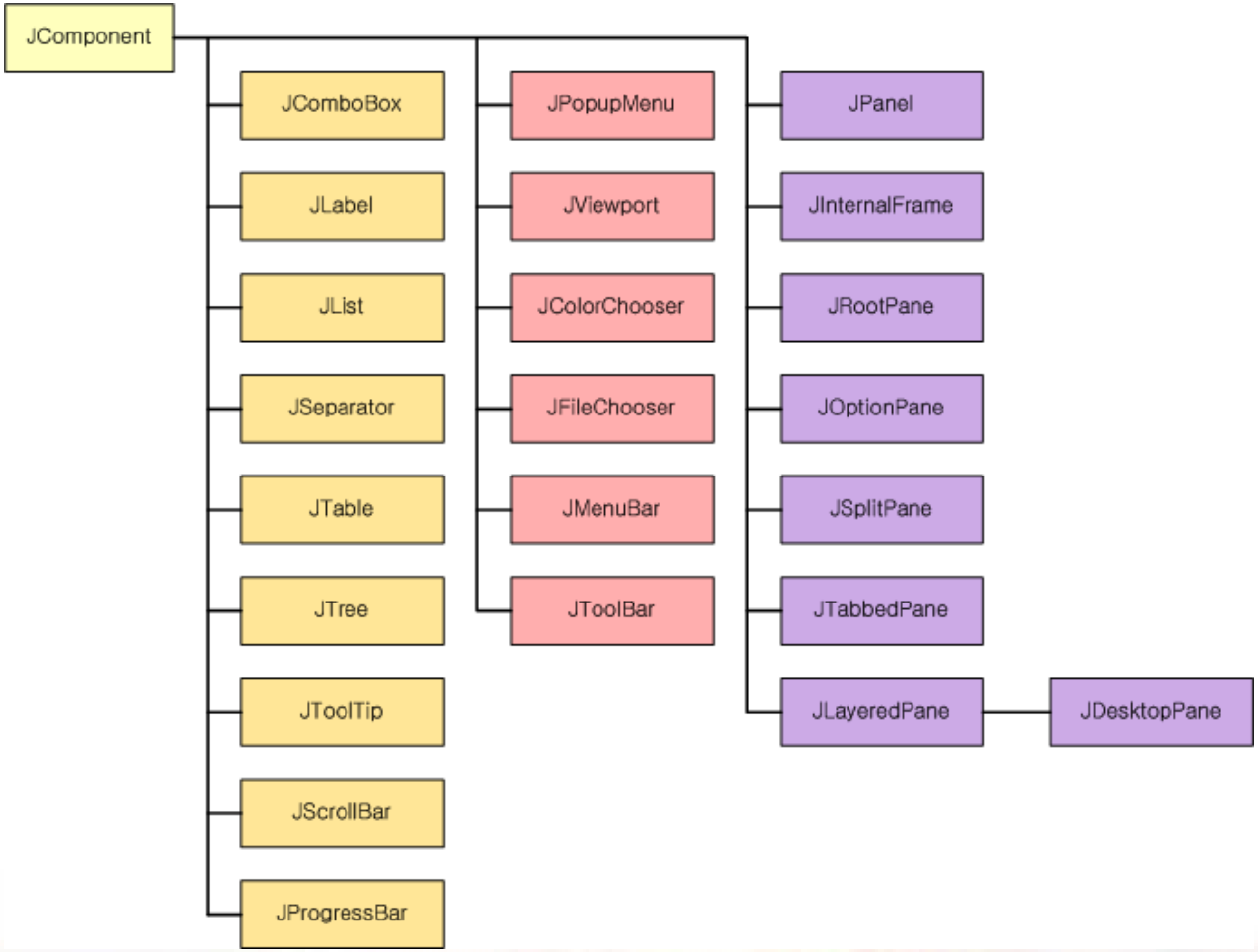
- 자바 애플리케이션
  - 콘솔 애플리케이션 : 문자 입출력
  - 그래픽 애플리케이션 : GUI를 이용
- GUI란?
  - Graphical User Interface
  - 그래픽 사용자 인터페이스
- GUI 애플리케이션을 작성하기 위한 패키지
  - AWT(**A**bstract **W**indow **T**oolkit)
    - java.awt
  - 스윙(Swing)
    - javax.swing
    - AWT를 개선한 GUI 클래스 라이브러리
    - 최근에 개발되는 GUI 애플리케이션에서 주로 사용

컴포넌트란?

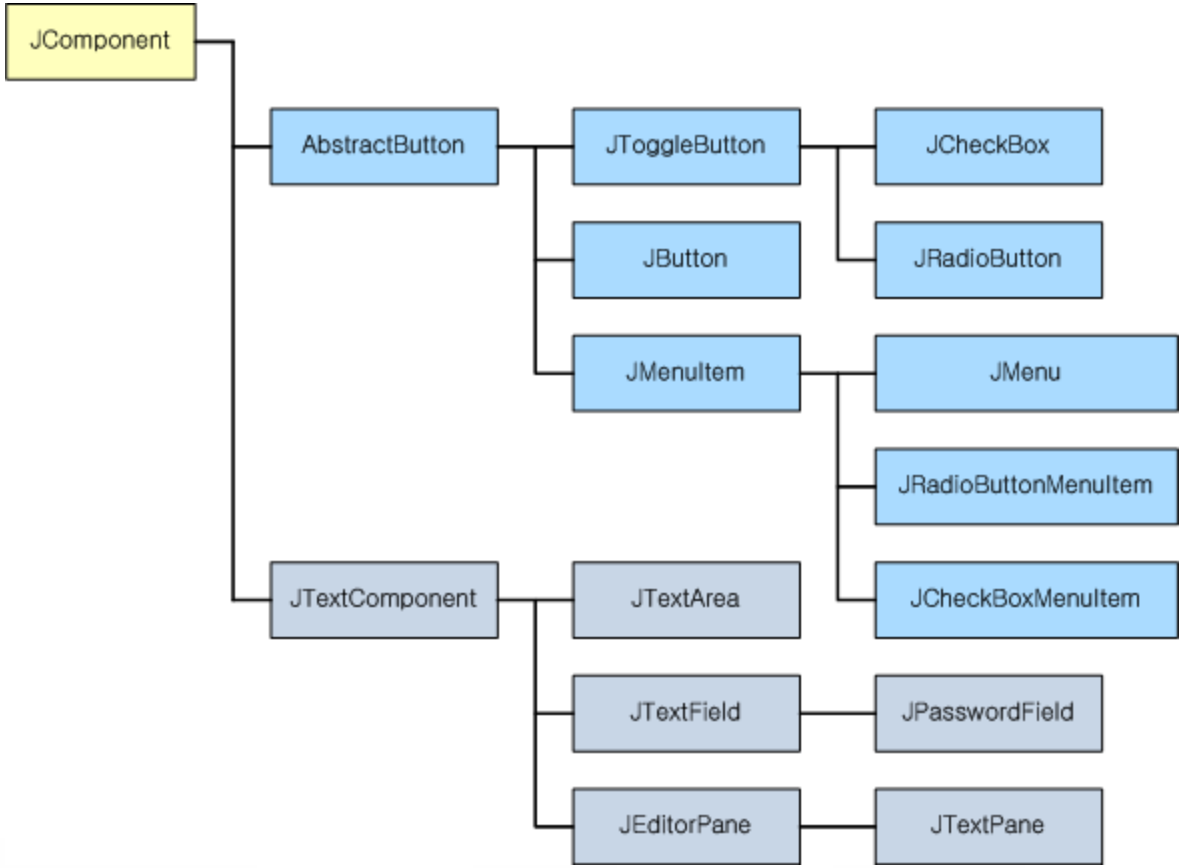
- 컴포넌트(component)
  - 사용자와 상호작용을 할 수 있도록 화면에 표시되어지는 그래픽 아이템
    - 각 컴포넌트에 따른 이벤트 처리기를 작성하여 등록
  - 종류
    - 버튼, 콤보 박스, 리스트, 레이블, 텍스트, 리스트 등



스윙 컴포넌트 클래스의 계층도 [1/2]



스윙 컴포넌트 클래스의 계층도 [2/2]



버튼 기반 컴포넌트

- 버튼 기반 컴포넌트
  - AbstractButton 추상 클래스를 상속받은 컴포넌트를 의미
- 버튼 기반 컴포넌트의 종류
  - 버튼
  - 체크 박스
  - 라디오 버튼

버튼 [1/2]

- 버튼(button)
  - 사용자 입력을 받는 가장 간단한 방법으로 버튼을 눌렀을 경우 이벤트가 발생
  - JButton 클래스의 객체

- 버튼 생성

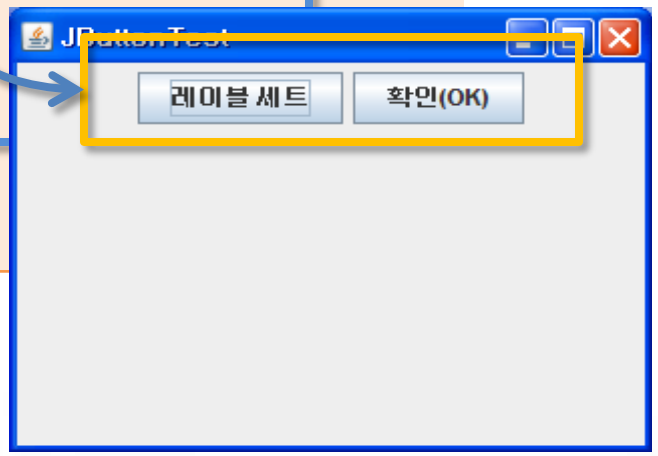
```
JButton()          // 이름이 없는 버튼 생성
JButton(String)    // 주어진 이름의 버튼 생성
```

- 버튼 주요 메소드
  - setText(String) : 버튼의 이름을 줄 때 사용
  - getText() : 버튼의 이름값을 얻음

버튼 [2/2]

[예제 12.1 - ButtonTest.java]

```
import java.awt.*;
import javax.swing.*;
public class ButtonTest {
    public static void main(String[] args) {
        JFrame  f = new JFrame("JButtonTest");

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(320, 240);
        f.setLayout(new FlowLayout());

        JButton button1 = new JButton();
        f.add(button1);
        f.add(new JButton("확인(OK)"));
        button1.setText("레이블 세트");

        f.setVisible(true);
    }
}
```



체크 박스 [1/2]

- 체크 박스(check box)
  - 주어진 항목이 선택되었는지를 시각적으로 나타내는 컴포넌트
  - JCheckBox 클래스의 객체

- 체크 박스 생성

```
▶ JCheckBox()        // 이름이 없는 체크 박스를 생성
▶ JCheckBox(String)  // 주어진 이름의 체크 박스를 생성
```

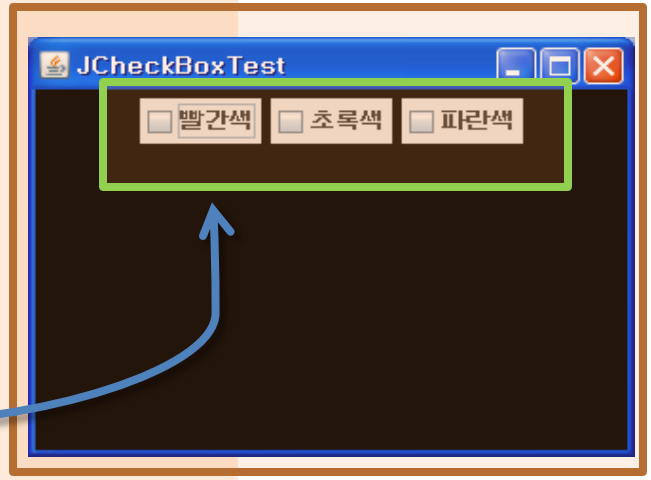
- 체크 박스 주요 메소드
  - setText(String) : 체크 박스의 이름을 줄 때 사용
  - getText() : 체크 박스의 이름값을 얻음
  - isSelected() : 체크 박스 선택 유무
  - setSelected(boolean) : 선택 상태를 지정

체크 박스 [2/2]

- 체크 박스 예제

[예제 12.2 - CheckBoxTest.java]

```
import java.awt.*;
import javax.swing.*;
public class CheckBoxTest {
    public static void main(String[] args) {
        JFrame f = new JFrame("JCheckBoxTest");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setLayout(new FlowLayout());
        f.setBackground(Color.BLACK);

        f.add(new JCheckBox("빨간색"));
        f.add(new JCheckBox("초록색"));
        f.add(new JCheckBox("파란색"));

        f.setVisible(true);
    }
}
```



라디오 버튼 [1/2]

- 라디오 버튼(radio button)
  - 주어진 항목 중에서 오직 하나만을 선택할 수 있는 컴포넌트
  - JRadioButton 클래스 객체

- 라디오 버튼 생성

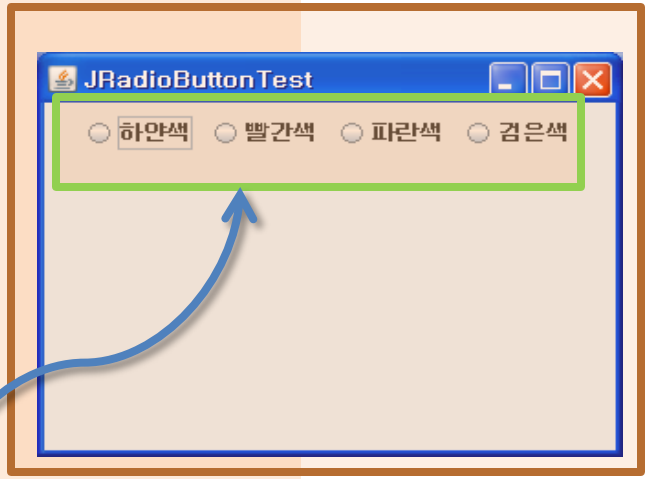
```
ButtonGroup grp = new ButtonGroup();  // 버튼 그룹 생성
// ...
rb1 = new JRadioButton("하얀색");       // 라디오 버튼 생성
grp.add(rb1);                          // 그룹에 등록
rb2 = new JRadioButton("빨간색");       // 라디오 버튼 생성
grp.add(rb2);                          // 그룹에 등록
```

라디오 버튼 [2/2]

- 라디오 버튼 예제

[예제 12.3 - RadioButtonTest.java]

```
import java.awt.*;
import javax.swing.*;
public class RadioButtonTest {
    public static void main(String[] args) {
        JFrame f = new JFrame("JRadioButtonTest");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setLayout(new FlowLayout());

        ButtonGroup grp = new ButtonGroup();
        JRadioButton[] rb = new JRadioButton[4];
        rb[0] = new JRadioButton("하얀색");
        rb[1] = new JRadioButton("빨간색");
        rb[2] = new JRadioButton("파란색");
        rb[3] = new JRadioButton("검은색");
        for (JRadioButton c : rb) {
            grp.add(c);
            f.add(c);
        }
        f.setVisible(true);
    }
}
```



콤보 박스 [1/2]

- 콤보 박스(combo box)
  - 사용자가 박스를 클릭하면 선택할 항목이 나타나는 드롭다운(drop-down)형식의 컴포넌트
  - JComboBox 클래스의 객체
  - 콤보 박스는 클릭한 경우에만 리스트가 나타나므로 공간을 효율적으로 사용

- 콤보 박스 생성

```
String[] Items = {"하얀색", "빨간색", "파란색", "검은색"};  // 항목 저장
JComboBox cb = new JComboBox(Items);                        // 콤보 박스 생성
```

콤보 박스 [2/2]

- 콤보 박스 예제

[예제 12.4 - ComboBoxTest.java]

```
import java.awt.*;
import javax.swing.*;
public class ComboBoxTest {
    public static void main(String[] args) {
        JFrame f = new JFrame("JComboBoxTest");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setLayout(new FlowLayout());

        String[] Items = {"하얀색", "빨간색", "파란색", "검은색"};
        JComboBox cb = new JComboBox(Items);
        // cb.addActionListener(new CBAction());  // 이벤트 처리기 등록
        f.add(cb);

        f.setVisible(true);
    }
}
```

리스트 [1/4]

- 리스트(list)
  - 콤보 박스와 매우 유사하지만, 항상 선택할 수 있는 항목들이 보임
  - JList 클래스의 객체

- 리스트 생성

```
▶ public JList()              // 선택 항목이 없는 빈 리스트를 생성
▶ public JList(Object[])  // Object 객체의 배열이 항목이 됨
```

리스트 [2/4]

- 리스트 예제 – 리스트를 위한 이벤트 처리

[예제 12.5 - ListTest.java]

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
class ListSelection implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent e) {
        JList src = (JList)e.getSource();
        String value = (String)src.getSelectedValue();

        if (value.equals("하얀색"))
            src.getParent().setBackground(Color.WHITE);
        else if (value.equals("빨간색"))
            src.getParent().setBackground(Color.RED);
        else if (value.equals("초록색"))
            src.getParent().setBackground(Color.GREEN);
        else if (value.equals("파란색"))
            src.getParent().setBackground(Color.BLUE);
        else if (value.equals("검은색"))
            src.getParent().setBackground(Color.WHITE);
    }
}
```

리스트 [3/4]

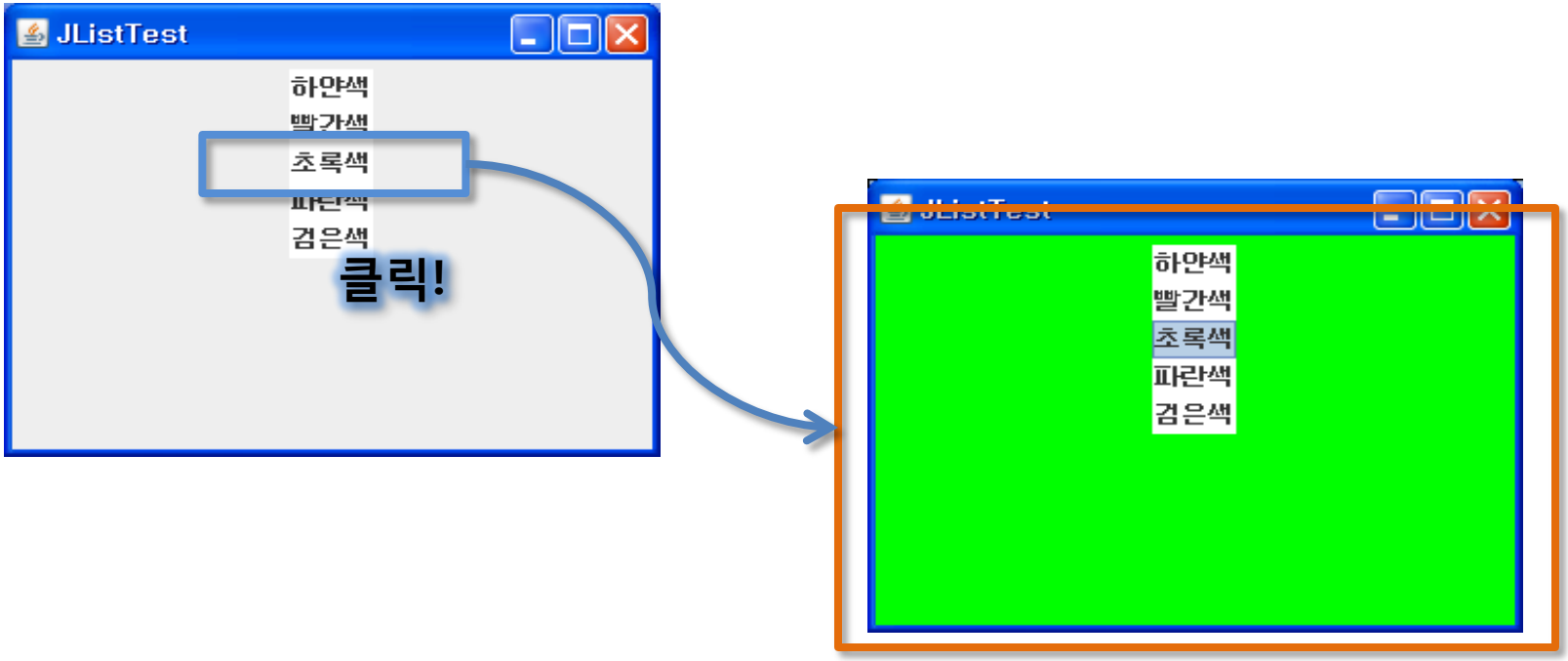
- 리스트 예제 - 컴포넌트 배치

[예제 12.5 - ListTest.java](cont.)

```
public class ListTest {
    public static void main(String[] args) {
        JFrame f = new JFrame("JListTest");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        f.setLayout(new FlowLayout());
        String[] Items = {"하얀색", "빨간색", "초록색", "파란색", "검은색"};
        JList list = new JList(Items);
        list.addListSelectionListener(new ListSelection());  // 이벤트 등록
        f.add(list);

        f.setVisible(true);
    }
}
```

리스트 [4/4]

- 리스트 예제 – 실행 결과



레이블

- 레이블(label)
  - 주로 텍스트 필드와 함께 사용하여 텍스트 필드의 의미를 설명하는 역할
  - 단순한 텍스트 문자열이며 사용자의 입력에 반응하지 않음
  - JLabel 클래스의 객체

- 레이블 생성

  ▶ public JLabel(String text) { this(text, LEFT); }
  ▶ public JLabel(String text, int alignment)

- 정렬 방법
  - 왼쪽 정렬(Component.LEFT_ALIGNMENT)
  - 오른쪽 정렬(Component.RIGHT_ALIGNMENT)
  - 가운데 정렬(Component.CENTER_ALIGNMENT)

텍스트 필드

- 텍스트 필드(text field)
  - 사용자로부터 한 줄의 입력을 받는 경우에 사용
  - JTextField 클래스의 객체

- 텍스트 필드 생성

```
public JTextField()                          // 이름 없는 텍스트 필드
public JTextField(String text)               // 이름 있는 텍스트 필드
public JTextField(int columns)               // 지정된 글자 수를 갖는 텍스트 필드
public JTextField(String text, int columns)  // 이름과 지정된 글자 수를
                                                갖는 텍스트 필드
```

- 입력 내용을 화면에 표시하지 않는 경우
  - JPasswordField 클래스 사용

텍스트 영역

- 텍스트 영역(text area)
  - 사용자로부터 여러 줄의 입력을 받는 경우에 사용
  - JTextArea 클래스의 객체

- 텍스트 영역 생성

```
public JTextArea()                                  // 이름 없는 텍스트 영역
public JTextArea(String text)                       // 이름 있는 텍스트 영역
public JTextArea(int rows, int columns)             // 지정된 줄과 칸을 갖는 텍스트 영역
public JTextArea(String text, int rows, int columns) // 이름과 지정된 줄/칸을 갖는
                                                      텍스트 영역을 생성
```

텍스트 필드, 텍스트 영역, 버튼 조합 예제 [1/2]

- 컴포넌트 배치

[예제 12.6 - LabelAndTextTest.java]

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class LabelAndTextTest extends JFrame {
    JTextField tf = new JTextField("name", 10);
    JPasswordField pf = new JPasswordField(8);
    JTextArea ta = new JTextArea(3, 25);
    JButton b = new JButton("OK");

    public LabelAndTextTest(String title) {  // 생성자
        super(title);

        this.add(new JLabel("Enter your name"));
        this.add(tf);
        this.add(new JLabel("Enter your password"));
        this.add(pf);
        this.add(ta);
        this.add(b);

        b.addActionListener(new OKAction());
    }
```

텍스트 필드, 텍스트 영역, 버튼 조합 예제 [2/2]

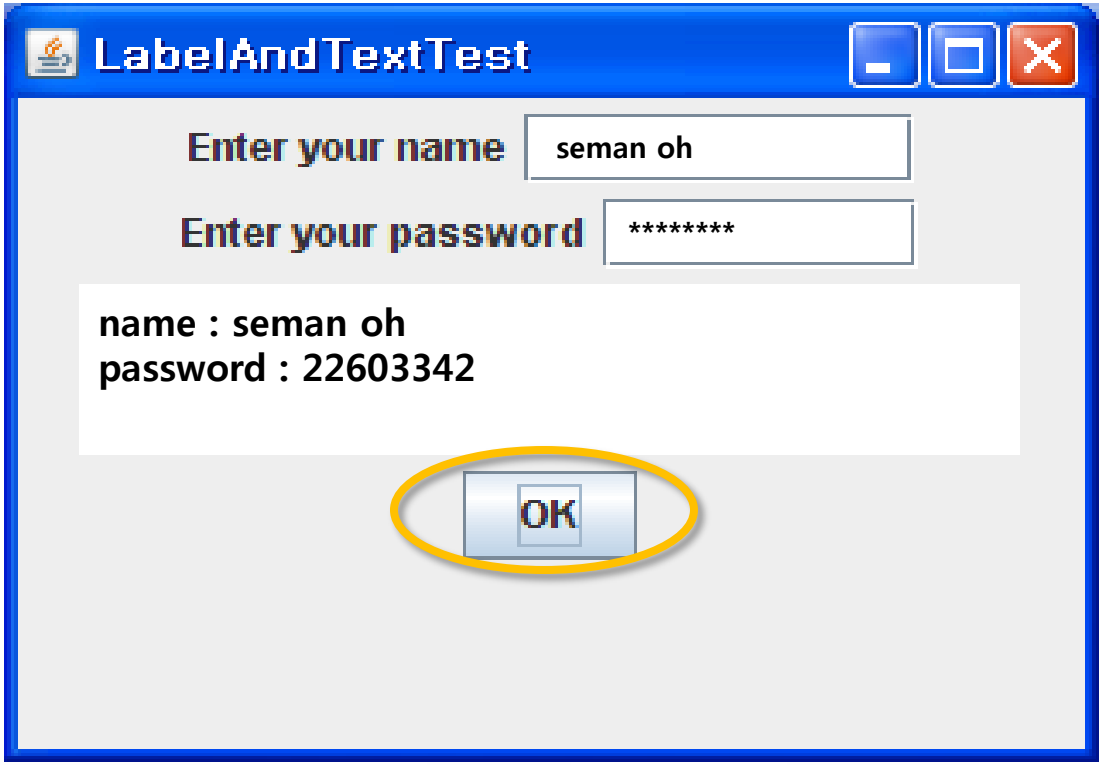
- 화면에 그리는 메소드 및 이벤트 처리기

[예제 12.6 - LabelAndTextTest.java](cont.)

```
public static void main(String[] args) {
        LabelAndTextTest f = new LabelAndTextTest("LabelAndTextTest");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setLayout(new FlowLayout());
        f.setVisible(true);
    }
    class OKAction implements ActionListener {
        public void actionPerformed(ActionEvent e) {  // 이벤트 처리기
            ta.setText("name : ");
            ta.append(tf.getText() + "₩n");
            ta.append("password : " + new String(pf.getPassword()));
        }
    }
}
```

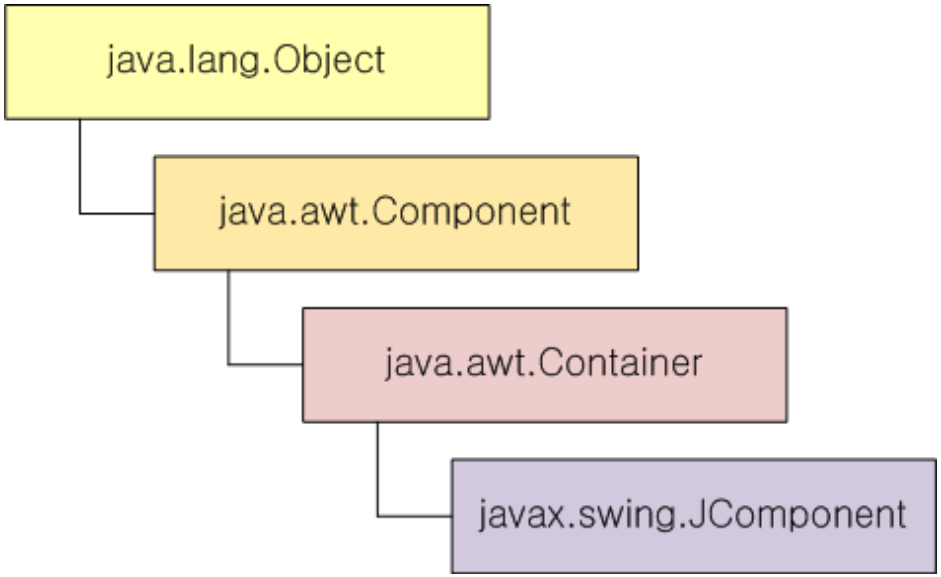
텍스트 필드, 텍스트 영역, 버튼 조합 예제 [2/2]

- 실행 결과



컴포넌트 클래스 [1/4]

- JComponent 클래스의 계층구조



- JComponent 클래스의 주요 메소드 [표 12.1] 참조

컴포넌트 클래스 [2/4]

- JComponent 클래스의 주요 메소드(Cont.)

메소드	설명
Color getBackground()	설정된 컴포넌트의 배경색을 반환
Border getBorder()	설정된 컴포넌트의 경계를 반환
Font getFont()	설정된 컴포넌트의 폰트를 반환
Color getForeground()	설정된 컴포넌트의 전경색을 반환
Point getLocation()	컴포넌트의 위치를 반환
Dimension getSize()	컴포넌트의 크기를 반환
boolean isEnabled()	컴포넌트의 사용 가능 여부를 반환
boolean isVisible()	컴포넌트의 표시 가능 여부를 반환
void paintComponent(Graphics g)	컴포넌트가 그려질 필요가 있을 때, 스윙이 호출하는 메소드.

컴포넌트 클래스 [3/4]

- JComponent 클래스의 주요 메소드

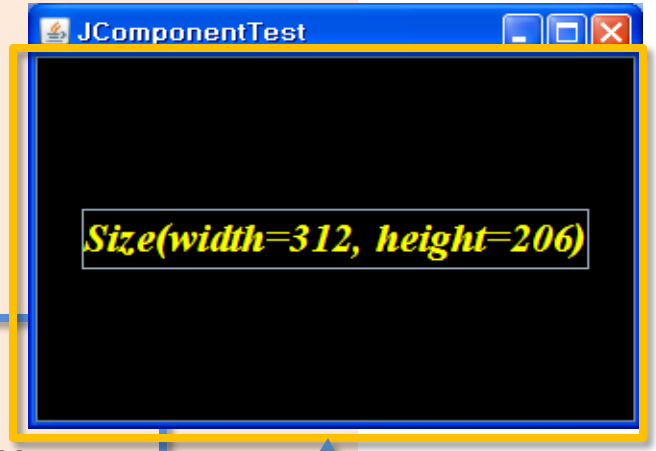
메소드	설명
void repaint(Rectangle r)	특정 영역만을 다시 그려야 할 때, 스윙이 호출하는 메소드
void setBackground(Color bg)	컴포넌트의 배경색을 설정
void setBorder(Border border)	컴포넌트의 경계를 설정
void setEnabled(boolean enabled)	컴포넌트의 사용 가능 여부를 설정
void setFont(Font font)	컴포넌트의 폰트를 설정
void setForeground(Color fg)	컴포넌트의 전경색을 설정
void setLocation(int x, int y)	컴포넌트의 위치를 설정
void setSize(int witdh, int height)	컴포넌트의 폭과 높이를 설정
void setVisible(boolean aFlag)	컴포넌트의 표시 가능 여부를 설정

컴포넌트 클래스 [4/4]

- 폰트의 모양과 배경/전경색을 설정하는 예제

[예제 12.7 - ComponentTest.java]

```
import java.awt.*;
import javax.swing.*;
public class ComponentTest {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        f.setSize(320, 240);
        f.setTitle("JComponentTest");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton b = new JButton();
        f.add(b);

        b.setFont(new Font(Font.SERIF, Font.ITALIC | Font.BOLD, 22));
        b.setBackground(Color.BLACK);
        b.setForeground(Color.YELLOW);
        Dimension d = b.getSize();
        b.setText("Size(width=" + d.width + ", height=" + d.height + ")");

        f.setVisible(true);
    }
}
```



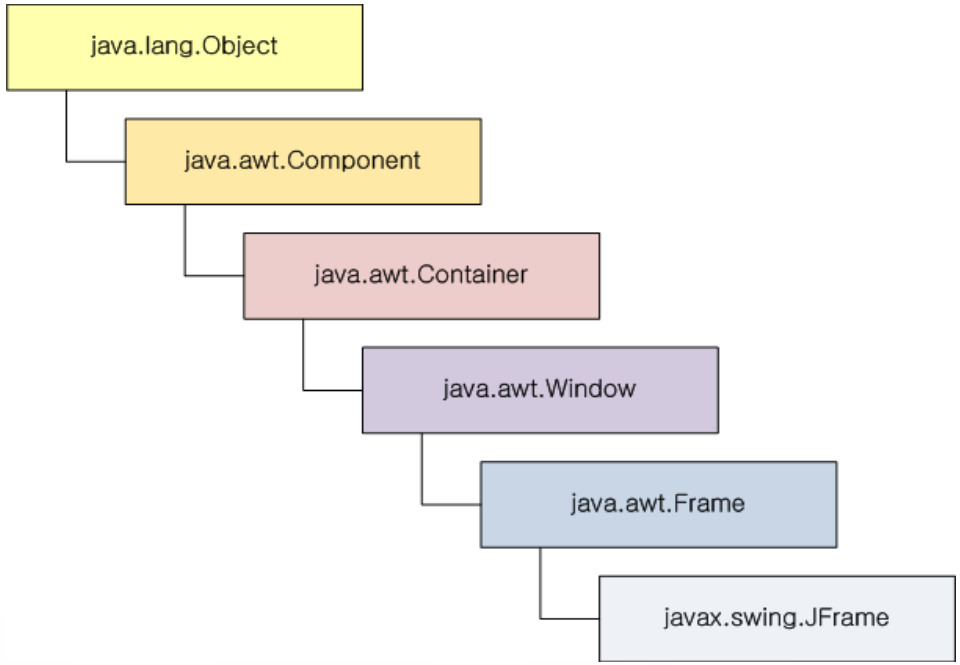
컨테이너

- 컨테이너(container)
  - 자신의 영역 안에 컴포넌트들을 포함하거나 다른 컨테이너들을 포함하는 것

- 컨테이너의 종류
  - 최상위 컨테이너(top-level container): 다른 컨테이너에 포함 될 수 없음
    - Jframe : 윈도우 프레임을 나타냄
    - Jdialog : 대화 상자
    - Japplet : 애플릿
  - 일반 컨테이너 : 다른 컨테이너에 포함 될 수 있음
    - Jpanel : 다른 컨테이너나 컴포넌트를 포함할 때 주로 사용
    - JScrollPane : 내용의 크기에 따라 스크롤 바를 표시, 표시 영역 조정
    - JTabbedPane : 같은 공간을 고용하는 여러 패널을 탭으로 구분

JFrame [1/2]

- JFrame
  - 제목 표시줄이 있는 윈도우 프레임을 나타내는 최상위 컨테이너
- JFrame 클래스의 계층구조

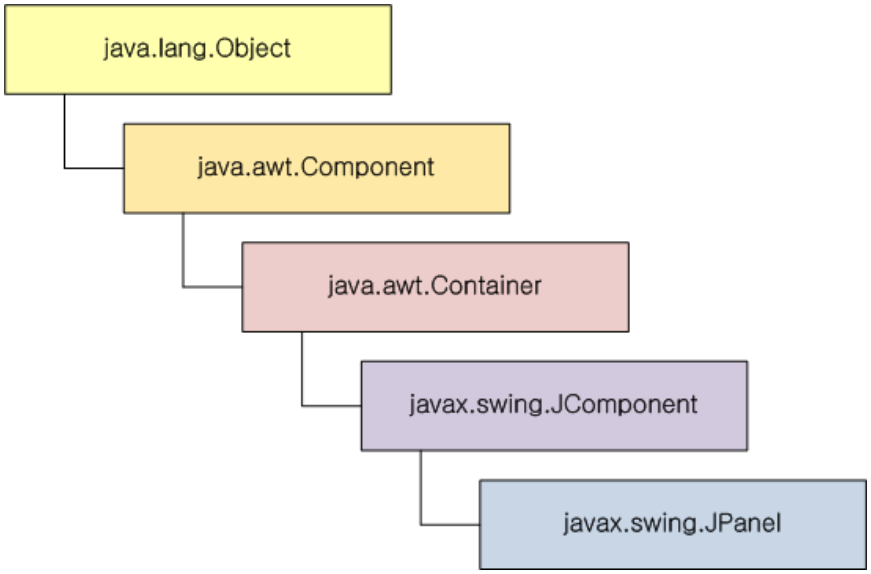


JFrame [2/2]

- Jframe 생성
  - JFrame() // 제목표시줄이 공란인 프레임을 생성
  - JFrame(String title) // 제목표시줄이 title인 프레임을 생성
- Jframe 주요 메소드
  - Container getContentPane() // 컨텐츠 페인 객체를 반환
  - setTitle(String title) // 프레임의 제목 설정
  - setDefaultCloseOperation(int operation) // 디폴트 닫힘 연산을 설정
- 디폴트 닫힘 연산에 올 수 있는 상수
  - DISPOSE_ON_CLOSE : 닫을 때, 모든 자원을 반환
  - DO_NOTHING_ON_CLOSE : 닫을 때, 아무 일도 하지 않음
  - EXIT_ON_CLOSE : 닫을 때, 윈도우를 화면에서 없애고 프로그램을 종료
  - HIDE_ON_CLOSE : 닫을 때, 윈도우를 화면에서 숨김(기본값)

JPanel

- JPanel
  - 다른 요소들을 포함할 때 사용하는 컨테이너
- JPanel 클래스의 계층구조



컨테이너 클래스

- java.awt.Container 클래스
  - Container 클래스를 확장하여 정의
- Container 클래스의 주요 메소드

메소드	설명
Component add(Component comp)	컴포넌트를 컨테이너에 추가
Component[] getComponents()	모든 컴포넌트를 반환
LayoutManager getLayout()	컨테이너에서 사용하는 배치 관리자를 반환
void invalidate()	컨테이너를 무효화함, 일반적으로 무효화된 컨테이너는 화면에 다시 그려짐
void remove(Component comp)	컴포넌트를 컨테이너에서 삭제
void removeAll()	모든 컴포넌트를 삭제
void setLayout(LayoutManager mgr)	배치 관리자를 설정
void update(Graphic g)	컨테이너 내용을 갱신

레이아웃

- 레이아웃(layout)
  - 여러 개의 컴포넌트들을 포함할 수 있고, 각 컴포넌트의 위치와 크기를 결정
  - 레이아웃의 종류
    - 보더 레이아웃(Border Layout)
    - 플로우 레이아웃(Flow Layout)
    - 그리드 레이아웃(Grid Layout)
    - 카드 레이아웃(Card Layout)
    - 그리드백 레이아웃(Grid Bag Layout)

보더 레이아웃 [1/3]

- 보더 레이아웃(Border Layout)
  - 컨테이너에 컴포넌트의 위치를 동, 서, 남, 북, 중앙으로 배치할 때 사용
  - 동, 서, 남, 북의 위치는 일정한 크기, 나머지 여백은 중앙이 차지함
- 윈도우 프레임을 나타내는 JFrame은 레이아웃을 지정하지 않으면 기본적으로 보더 레이아웃

보더 레이아웃 [2/3]

- setLayout() 메소드를 사용하여 보더 레이아웃을 지정

```
setLayout(new BorderLayout());
setLayout(new BorderLayout(5, 5));   // 가로 5픽셀, 세로 5픽셀의 간격
```

- add() 메소드
  - 레이아웃의 동(“East")/서("West")/남("South")/북("North")/중앙("Center")의 위치를 지정

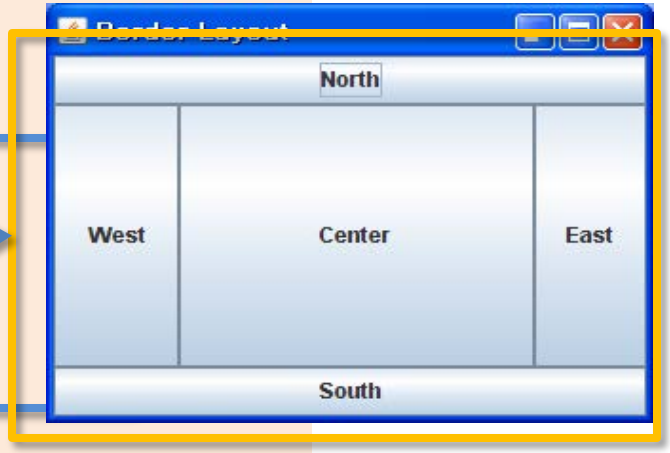
```
add("Center", new JButton("Center"));
```

보더 레이아웃 [3/3]

- 보더 레이아웃 예제

[예제 12.8 - BorderLayoutTest.java]

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutTest {
    public static void main(String[] args) {
        JFrame f = new JFrame("Border Layout");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        f.setLayout(new BorderLayout());
        f.add("East", new JButton("East"));
        f.add("West", new JButton("West"));
        f.add("South", new JButton("South"));
        f.add("North", new JButton("North"));
        f.add("Center", new JButton("Center"));

        f.setVisible(true);
    }
}
```



플로우 레이아웃 [1/2]

- 플로우 레이아웃(Flow Layout)
  - 컨테이너 안에 컴포넌트들을 가로 방향으로 배치할 때 사용
  - 왼쪽에서 오른쪽 방향으로 배치, 위에서 아래로 물 흐르듯이 배치
- JPanel은 레이아웃을 지정하지 않으면 기본적으로 플로우 레이아웃이 설정

```
setLayout(new FlowLayout());
setLayout(new FlowLayout(FlowLayout.CENTER));      // 가운데 정렬
setLayout(new FlowLayout(FlowLayout.RIGHT, 5, 5)); // 가로 5픽셀,
                                                   세로 5픽셀의 간격
```

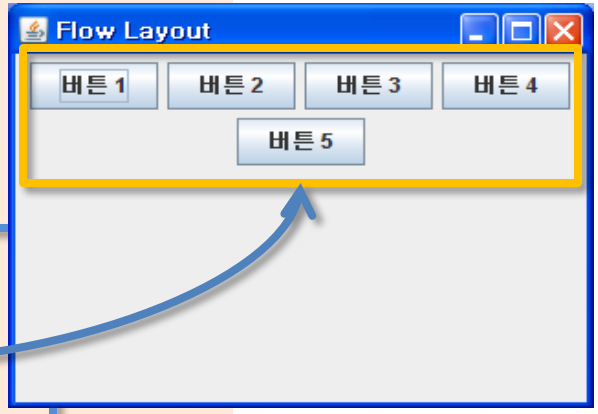
- 컴포넌트에 대한 정렬 방식 지정
  - LEFT, CENTER, RIGHT 상수 등을 제공

플로우 레이아웃 [2/2]

- 플로우 레이아웃 예제

[예제 12.9 - FlowLayoutTest.java]

```
import java.awt.*;
import javax.swing.*;
public class FlowLayoutTest {
    public static void main(String[] args) {
        JFrame f = new JFrame("Flow Layout");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        f.setLayout(new FlowLayout());
        f.add(new JButton("버튼 1"));
        f.add(new JButton("버튼 2"));
        f.add(new JButton("버튼 3"));
        f.add(new JButton("버튼 4"));
        f.add(new JButton("버튼 5"));

        f.setVisible(true);
    }
}
```



그리드 레이아웃 [1/2]

- 그리드 레이아웃(Grid Layout)
  - 컨테이너 안에 있는 모든 컴포넌트들을 격자 내에 배치
    - 컴포넌트 크기 동일

- 그리드 레이아웃을 생성하기 위해 행과 열의 수를 지정

```
setLayout(new GridLayout(3, 4))         // 3행 4열의 그리드
setLayout(new GridLayout(2, 0))         // 2행의 그리드
setLayout(new GridLayout(3, 2, 5, 5))   // 3행 2열의 그리드
                                        // 가로 5픽셀, 세로 5픽셀의 간격
```

그리드 레이아웃 [2/2]

- 그리드 레이아웃 예제

[예제 12.10 - GridLayoutTest.java]

```
import java.awt.*;
   import javax.swing.*;
   public class GridLayoutTest {
       public static void main(String[] args) {
           JFrame f = new JFrame("Grid Layout");
           f.setSize(320, 240);
           f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

           f.setLayout(new GridLayout(3, 2, 5, 5));
           f.add(new JButton("버튼 1"));
           f.add(new JButton("버튼 2"));
           f.add(new JButton("버튼 3"));
           f.add(new JButton("버튼 4"));
           f.add(new JButton("버튼 5"));
           f.add(new JButton("버튼 6"));

           f.setVisible(true);
       }
   }
```

카드 레이아웃 [1/4]

- 카드 레이아웃(Card Layout)
  - 카드를 쌓아놓은 형태로 컨테이너를 볼 수 있도록 배치
  - 한 번에 한 개의 컨테이너 객체(카드)를 볼 수 있도록 배치

- 카드 레이아웃 생성

```
JPanel cards = new JPanel();
cards.setLayout(new CardLayout());
JPanel card1 = new JPanel();
JPanel card2 = new JPanel();
cards.add("첫번째카드", card1);
cards.add("두번째카드", card2);
cards.show(cards, "첫번째카드");
```

카드 레이아웃 [2/4]

- 카드 레이아웃 예제 – 컴포넌트 배치

[예제 12.11 - CardLayoutTest.java]

```
public class CardLayoutTest implements ActionListener {
    JPanel cards;
    public void createUI() {
        cards = new JPanel();
        JPanel card1 = new JPanel();
        card1.setBackground(Color.blue);
        JButton b1 = new JButton("Next");
        b1.setActionCommand("Next");
        b1.addActionListener(this);
        card1.add(b1);

        JPanel card2 = new JPanel();
        card2.setBackground(Color.yellow);
        JButton b2 = new JButton("Prev");
        b2.setActionCommand("Prev");
        b2.addActionListener(this);
        card2.add(b2);

        cards.setLayout(new CardLayout());
        cards.add(card1, "First");
        cards.add(card2, "Second");
    }
```

카드 레이아웃 [3/4]

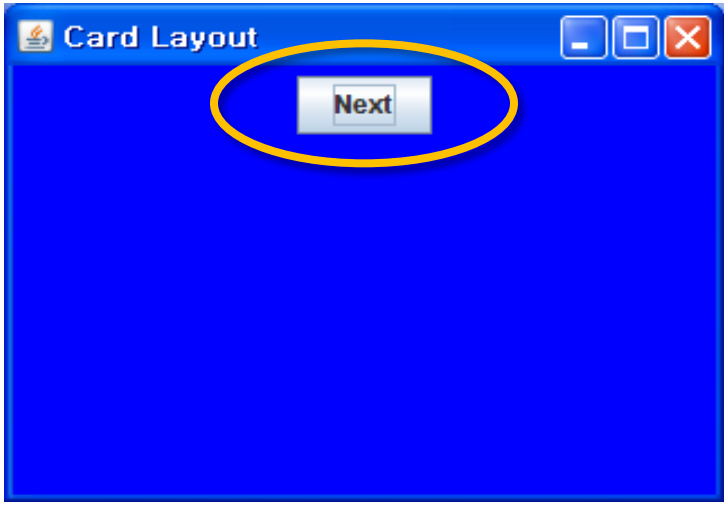
- 카드 레이아웃 예제 – 이벤트 처리 및 보여주기

[예제 12.11 - CardLayoutTest.java](cont.)

```
    public void actionPerformed(ActionEvent e) {
        CardLayout cl = (CardLayout) (cards.getLayout());
        String str = e.getActionCommand();

        if (str.equals("Next"))
            cl.show(cards, "Second");
        else if (str.equals("Prev"))
            cl.show(cards, "First");
    }
    public static void main(String[] args) {
        JFrame f = new JFrame("Card Layout");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        CardLayoutTest clt = new CardLayoutTest();
        clt.createUI();
        f.add(clt.cards);

        f.setVisible(true);
    }
}
```

카드 레이아웃 [4/4]

- 실행 결과



그리드백 레이아웃

- 그리드백 레이아웃(Grid Bag Layout)
  - 컴포넌트의 위치와 크기를 자유롭게 생성할 수 있는 레이아웃
- 그리드백 레이아웃 생성

```
JFrame f = new JFrame();

GridbagLayout gbl = new GridbagLayout();
GridbagConstraints gbc = new GridbagConstraints();

f.setLayout(gbl);
gbc.weightx = 1.0;
gbl.setConstraints(button, gbc);
f.add(button);
```

- 컨스트레인트(constraints) 변수를 사용하여 컴포넌트의 위치와 크기를 지정
  - gridx, gridy, gridwidth, gridheight, ipadx, ipady, anchor, weightx, weighty

이벤트 처리

- 이벤트(event)
  - 사용자 행동(action)에 의해 발생하는 사건
- 이벤트 처리기(event handler)
  - 특정 이벤트가 발생하면 그와 관련된 메소드를 통해 처리하는 것
- 이벤트-드리븐 프로그래밍(event-driven programming)
  - 이벤트와 이벤트 처리기를 통해 프로그램을 작성하는 방식
- **위임(delegation) 모델**
  - 자바에서 이벤트 처리하는 모델
  - 이벤트가 발생하는 컴포넌트에 이벤트 처리를 위한 리스너 객체(listener object)를 등록하여 이벤트 처리하는 방법
  - 이벤트 처리를 위한 코드가 애플리케이션 코드와 분리됨으로써 재사용성이 높음

이벤트 처리 방법 [1/7]

- 이벤트 프로그래밍을 위한 순서

① 이벤트 클래스를 import

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

② **이벤트 타입의 클래스를 정의**

```
class ButtonListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {  // 이벤트 처리기
           // ...
        }  }
```

③ **이벤트 리스너 객체를 생성**

```
ButtonListener bt = new ButtonListener();
```

④ **해당하는 컴포넌트에 이벤트 리스너 객체를 등록**

```
b1 = new Button("OK");
b1.addActionListener(bt);
```

이벤트 처리 방법 [2/7]

- 이벤트의 종류
  - 모든 컴포넌트에서 발생할 수 있는 공통 이벤트
    - MouseEvent : 마우스 이동이나 마우스 버튼이 클릭된 경우에 발생한다.
    - KeyEvent : 키보드의 키를 누른 경우에 발생한다.
    - ComponentEvent : 컴포넌트의 크기나 위치, 보이기 가부가 변경될 때 발생한다.
    - FocusEvent : 포커스를 얻거나 잃을 때 발생한다.

이벤트 처리 방법 [3/7]

- 이벤트의 종류 (cont.)
  - 컴포넌트에서 발생할 수 있는 이벤트의 종류

이벤트 / 컴포넌트	Action	Caret	Change	Document / UndoableEdit	Item	ListSelection	Window
JButton	○		○		○		
JCheckBox	○		○		○		
JCheckBoxMenuItem	○		○		○		
JComboBox	○				○		
JDialog							○
JFrame							○
JList						○	
JMenuItem	○		○		○		
JRadioButton	○		○		○		
JSlider			○				
JTextArea		○		○			
JTextField	○	○		○			
JTextPane		○		○			

이벤트 처리 방법 [4/7]

- 리스너 인터페이스
  - 구현 시 모든 메소드를 정의해야 함
  - 각 이벤트에 따른 리스너 인터페이스 및 이벤트 등록/제거 메소드

이벤트	리스너 인터페이스	이벤트 등록 및 제거 메소드
ActionEvent	ActionListener	addActionListener() removeActionListener()
ItemEvent	ItemListener	addItemListener() removeItemListener()
MouseEvent	MouseListener	addMouseListener() removeMouseListener()
	MouseMotionListener	addMouseMotionListener() removeMouseMotionListener()
KeyEvent	KeyListener	addKeyListener() removeKeyListener()
FocusEvent	FocusListener	addFocusListener() removeFocusListener()
WindowEvent	WindowListener	addWindowListener() removeWindowListener()

이벤트 처리 방법 [5/7]

- 어답터 클래스
  - 필요한 메소드만 작성 가능
  - 리스너 인터페이스/어답터 클래스의 메소드

리스너 인터페이스	어답터 클래스	메소드
ActionListener	없음	actionPerformed(ActionEvent)
ItemListener	없음	itemStateChanged(ItemEvent)
MouseListener	MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)

이벤트 처리 방법 [6/7]

- 어답터 클래스
  - 리스너 인터페이스/어답터 클래스의 메소드(cont.)

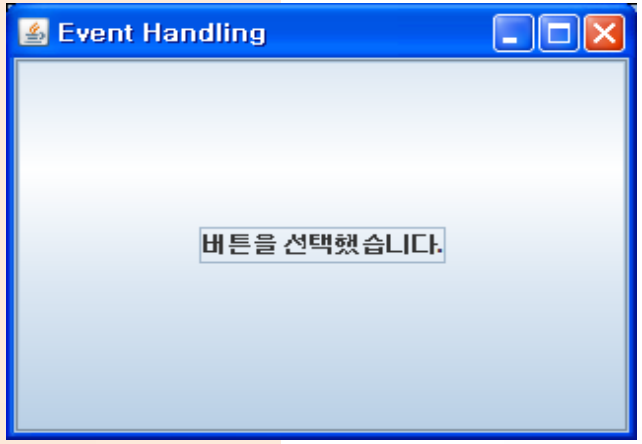
리스너 인터페이스	어답터 클래스	메소드
MouseMotionListener	MouseMotionAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
KeyListener	KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
FocusListener	FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)
WindowListener	WindowAdapter	windowOpened(WindowEvent) windowClosing(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent) windowConified(WindowEvent) windowDeiconified(WindowEvent)
TextListener	없음	textValueChanged(TextEvent)

이벤트 처리 방법 [7/7]

- 이벤트 처리 예제

[예제 12.12 - SimpleEventHandling.java]

```
import java.awt.event.*;
  import javax.swing.*;
  class ActionEventHandler implements ActionListener {
      public void actionPerformed(ActionEvent e) {
          JButton b = (JButton)e.getSource();
          b.setText("버튼을 선택했습니다.");
      }
  }
  public class SimpleEventHandling {
      public static void main(String[] args) {
          JFrame f = new JFrame("Event Handling");
          f.setSize(320, 240);
          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

          JButton b = new JButton("버튼을 누르세요.");
          b.addActionListener(new ActionEventHandler());
          f.add(b);

          f.setVisible(true);
      }
  }
```



GUI 프로그래밍

- 동적인 프로그램 작성에 용이
- 예제 프로그램
  - 마우스 이벤트 처리 응용
    - 마우스 버튼을 누르거나 떼면, MouseEvent라는 이벤트가 발생
    - 마우스를 움직이면 MouseEvent라는 이벤트가 발생
  - 색이 변하는 문자열 만들기
    - 스레드 사용
  - 이미지 애니메이션
    - Timer 클래스 사용

마우스 이벤트 처리 응용 [1/3]

- 마우스 이벤트 예제 – 그리기 및 이벤트 처리기 등록

[예제 12.13 - MouseEventTest.java]

```
class WhiteboardComponent extends JComponent {
    int x, y, w, h;

    WhiteboardComponent() {   // 생성자
        x = 0;    y = 0;    w = 0;    h = 0;

        addMouseListener(new MouseEventHdl());            // 처리기 등록
        addMouseMotionListener(new MouseMotionHdl()); // 처리기 등록
    }
    public void paintComponent(Graphics g) {
        g.setColor(Color.BLACK);
        g.drawString("마우스로 사각형을 그리세요.", 20, 50);
        g.setColor(Color.RED);
        g.drawRect(x, y, w, h);
    }
    public void setStartPoint(int x, int y) {
        this.x = x;             this.y = y;
    }
    public void setEndPoint(int x, int y) {
        w = Math.abs(this.x - x);             h = Math.abs(this.y - y);
    }
```

마우스 이벤트 처리 응용 [2/3]

- 마우스 이벤트 예제 – 이벤트 처리

[예제 12.13 - MouseEventTest.java](cont.)

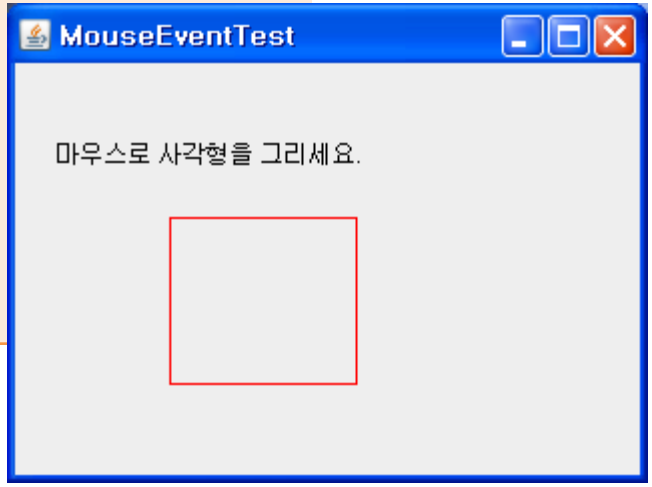
```
    class MouseEventHdl extends MouseAdapter {
        public void mousePressed(MouseEvent e) {      // 이벤트 처리기
            setStartPoint(e.getX(), e.getY());
        }
        public void mouseReleased(MouseEvent e) {     // 이벤트 처리기
            setEndPoint(e.getX(), e.getY());
            repaint();
        }
    }
    class MouseMotionHdl extends MouseMotionAdapter {
        public void mouseDragged(MouseEvent e) {      // 이벤트 처리기
            setEndPoint(e.getX(), e.getY());
            repaint();
        }
    }
}
```

마우스 이벤트 처리 응용 [3/3]

- 마우스 이벤트 예제

[예제 12.13 - MouseEventTest.java](cont.)

```
public class MouseEventTest {
    public static void main(String[] args) {
        JFrame f = new JFrame("MouseEventTest");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        f.add(new WhiteboardComponent());

        f.setVisible(true);
    }
}
```



색이 변하는 문자열 만들기 [1/4]

- 텍스트 애니메이션 예제 - 그리기

[예제 12.14 - TextAnimation.java]

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class TextAnimationComponent extends JComponent {
    public String str;

    public void paintComponent(Graphics g) {
        g.setFont(new Font("Dialog", Font.BOLD, 30));
        for (int i = 0; i < str.length(); i++) {
            int rVal, gVal, bVal;

            rVal = (int)(Math.random() * 256);
            gVal = (int)(Math.random() * 256);
            bVal = (int)(Math.random() * 256);

            g.setColor(new Color(rVal, gVal, bVal));
            g.drawString(str.substring(i, i + 1), 25 + i * 25, 50);
        }
    }
}
```

색이 변하는 문자열 만들기 [2/4]

- 텍스트 애니메이션 예제 – 이벤트 처리 및 스레드

[예제 12.14 - TextAnimation.java](cont.)

```
public void initComponents() {
        TextAnimation ta = this;

        ta.f = new JFrame();
        JFrame f = ta.f;
        f.setTitle("Text Animation");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.getContentPane().setBackground(Color.white);
        f.setSize(600, 250);

        ta.c = new TextAnimationComponent();
        TextAnimationComponent c = ta.c;
        f.add(c);
        ta.b = new JButton();
        JButton b = ta.b;
        b.addActionListener(ta);
        b.setText("Press to start the animation.");
        f.add("South", b);

        f.setVisible(true);
    }
```

색이 변하는 문자열 만들기 [3/4]

- 텍스트 애니메이션 예제 –스레드

[예제 12.14 - TextAnimation.java](cont.)

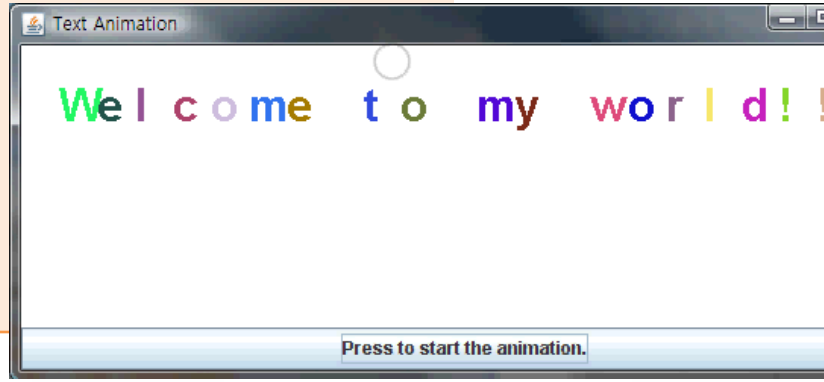
```
public void setMessage(String message) {
    c.str = message;
}
public String getMessage() {
    return c.str;
}
public void run() {
    while (thr != null) {
        try {
            c.repaint();
            Thread.sleep(1 * 1000);
        } catch (InterruptedException e) {
        }
    }
}
```

색이 변하는 문자열 만들기 [1/4]

- 텍스트 애니메이션 예제 – 이벤트 처리

[예제 12.14 - TextAnimation.java](cont.)

```
        public void actionPerformed(ActionEvent e) {
            if (thr == null) {
                thr = new Thread(this);
                thr.start();
                b.setText("Press to stop the animation.");
            } else {
                thr = null;
                b.setText("Press to start the animation.");
            }
        }
        public static void main(String[] arg) {
            TextAnimation ta = new TextAnimation();

            ta.initComponents();
            ta.setMessage("Welcome to my world!!!");
        }
    }
```



이미지 애니메이션 [1/4]

- 이미지 애니메이션 예제 – 타이머 이벤트 처리

[예제 12.15 - ImageAnimationTest.java]

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.image.BufferedImage;
import java.io.*;
import javax.imageio.ImageIO;

class ImageAnimationComponent extends JComponent {
    Image[] images;
    int curImageIndex;
    Timer t;

    class TimerHandler implements ActionListener {
        public void actionPerformed(ActionEvent e) { // 타이머 이벤트 처리기
            curImageIndex++;   // 다음 이미지 지정
            curImageIndex %= images.length;
            repaint();
        }
    }
```

이미지 애니메이션 [2/4]

- 이미지 애니메이션 예제 – 타이머 사용

[예제 12.15 - ImageAnimationTest.java]

```
    ImageAnimationComponent(String imageName, int imageCount) {  // 생성자
        try {                                   // 이미지 읽기
            images = new Image[imageCount];
            for (int i = 0; i < images.length; i++)
                images[i] = ImageIO.read(new File(imageName + i + ".jpg"));
        } catch (IOException e) { }
        t = new Timer(250, new TimerHandler());   // 타이머 객체 생성
    }
    public void paintComponent(Graphics g) {      // 이미지 그리기
        g.drawImage(images[curImageIndex], 0, 0, null);
    }
    public void start() {
        t.start();              // 타이머 작동 시작
    }
    public void stop() {
        t.stop();               // 타이머 작동 중지
    }
    boolean isRunning() {
        return t.isRunning();  // 타이머 작동 여부 반환
    }
}
```

이미지 애니메이션 [3/4]

- 이미지 애니메이션 예제 – 버튼 이벤트 처리

[예제 12.15 - ImageAnimationTest.java]

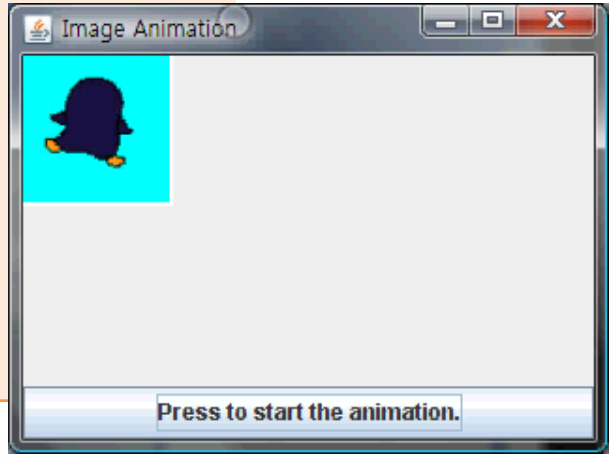
```
public class ImageAnimationTest implements ActionListener {
    ImageAnimationComponent c;

    public void actionPerformed(ActionEvent e) { // 버튼 이벤트 처리기
        JButton b = (JButton)e.getSource();
        if (c.isRunning()) {
            c.stop();
            b.setText("Press to start the animation.");
        } else {
            c.start();
            b.setText("Press to stop the animation.");
        }
    }
```

이미지 애니메이션 [4/4]

- 이미지 애니메이션 예제 - 보여주기

[예제 12.15 - ImageAnimationTest.java]

```
public static void main(String[] arg) {
        JFrame  f = new JFrame("Image Animation");
        f.setSize(320, 240);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        ImageAnimationTest ia = new ImageAnimationTest();
        ia.c = new ImageAnimationComponent("penguin", 8);
        f.add(ia.c);

        JButton b = new JButton("Press to start the animation.");
        f.add("South", b);
        b.addActionListener(ia);    // 이벤트 처리기 등록

        f.setVisible(true);
    }
}
```



단원 요약 [1/3]

- 컴포넌트(component)
  - 사용자와 상호작용을 할 수 있도록 화면에 표시되어지는 그래픽 표현들
  - 종류
    - 버튼, 콤보 박스, 리스트, 레이블, 텍스트, 리스트 등등
- 컨테이너(Container)
  - 자신의 영역 안에 컴포넌트들을 포함하거나 다른 컨테이너들을 포함하는 것
  - 최상위 컨테이너(top-level container): 다른 컨테이너에 포함 될 수 없음
    - Jframe , Jdialog, Japplet
  - 일반 컨테이너 : 다른 컨테이너에 포함 될 수 있음
    - Jpanel , JScrollPane, JTabbedPane

단원 요약 [2/3]

- 레이아웃(layout)
  - 여러 개의 컴포넌트들을 포함할 수 있고, 각 컴포넌트의 위치와 크기를 결정
  - 레이아웃의 종류
    - 보더 레이아웃(Border Layout)
    - 플로우 레이아웃(Flow Layout)
    - 그리드 레이아웃(Grid Layout)
    - 카드 레이아웃(Card Layout)
    - 그리드백 레이아웃(Grid Bag Layout)

단원 요약 [3/3]

- 이벤트 처리
  - 이벤트(event) : 사용자 행동(action)에 의해 발생하는 사건
  - 이벤트 처리기(event handler) : 특정 이벤트가 발생하면 그와 관련된 메소드를 통해 처리하는 것
  - 이벤트 프로그래밍을 위한 순서
    ① 이벤트 클래스를 import
    ② 이벤트 타입의 클래스를 정의
    ③ 이벤트 리스너 객체를 생성
    ④ 해당하는 컴포넌트에 이벤트 리스너 객체를 등록

- GUI 프로그래밍
  - 마우스 이벤트 처리 응용
  - 색이 변하는 문자열 만들기
  - 이미지 애니메이션