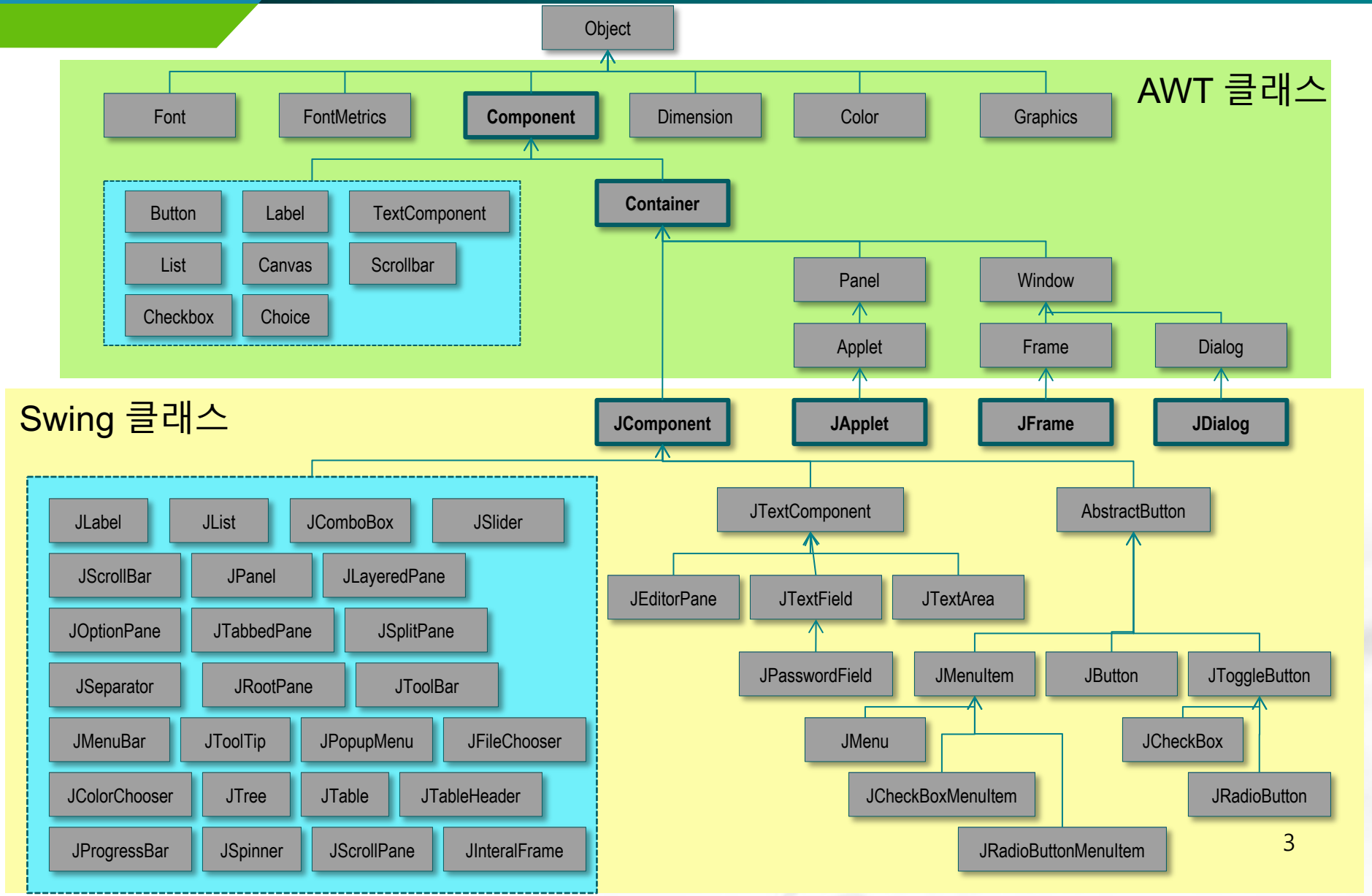
AWT

자바 GUI

- Java GUI(Graphic User Interface)
  - ✓ java.awt 패키지의 클래스 – jdk 8 까지만 지원
    - ❑ 만들기가 쉬움
    - ❑ OS자원을 이용하여 GUI를 출력
    - ❑ 플랫폼에 종속적
    - ❑ 느리고 무거움
  - ✓ javax.swing 패키지의 클래스 이용
    - ❑ 만들기가 AWT 보다는 어려움
    - ❑ 자체적인 GUI클래스로 출력
    - ❑ 플랫폼에 독립적
    - ❑ Double Buffering을 이용한 출력
    - ❑ 풍선 도움말 지원
    - ❑ 빠르고 가벼움
  - ✓ Java FX
    - ❑ 가볍고 풍부한 UI 제공
    - ❑ 레이아웃, 스타일, 애플리케이션 로직 분리 개발
    - ❑ 자바7 업데이트 6버전부터 JavaFX2.2를 JDK와 JRE에 포함

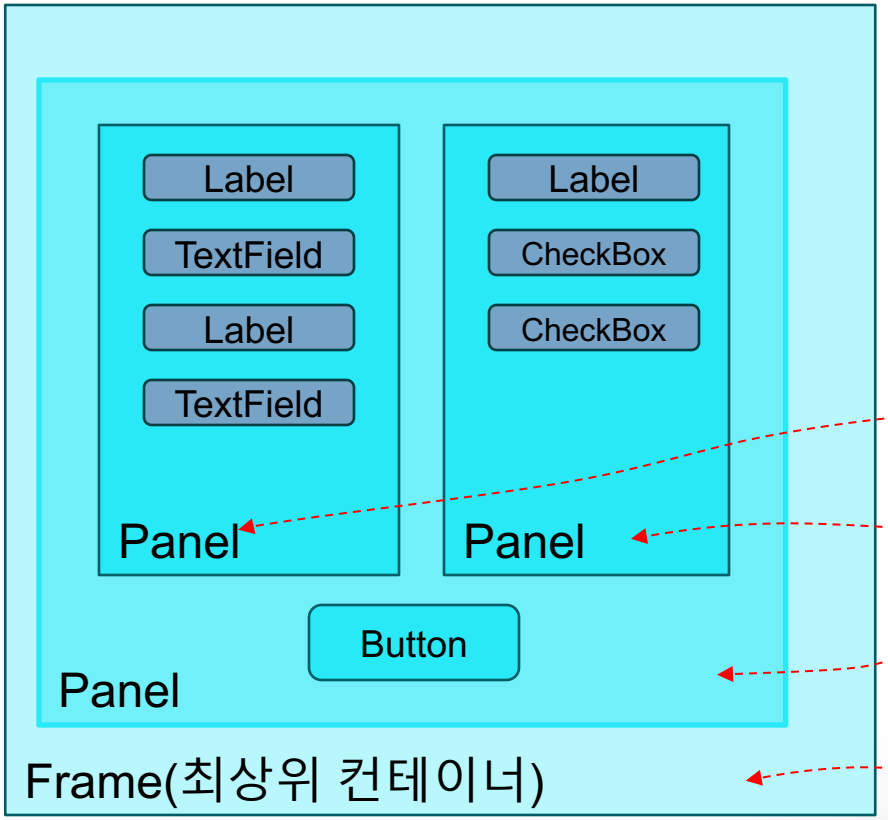
자바 GUI


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
JInteralFrame
JTextComponent
AbstractButton
JEditorPane
JTextField
JTextArea
JPasswordField
JMenuItem
JButton
JToggleButton
JMenu
JCheckBox
JCheckBoxMenuItem
JRadioButton
JRadioButtonMenuItem

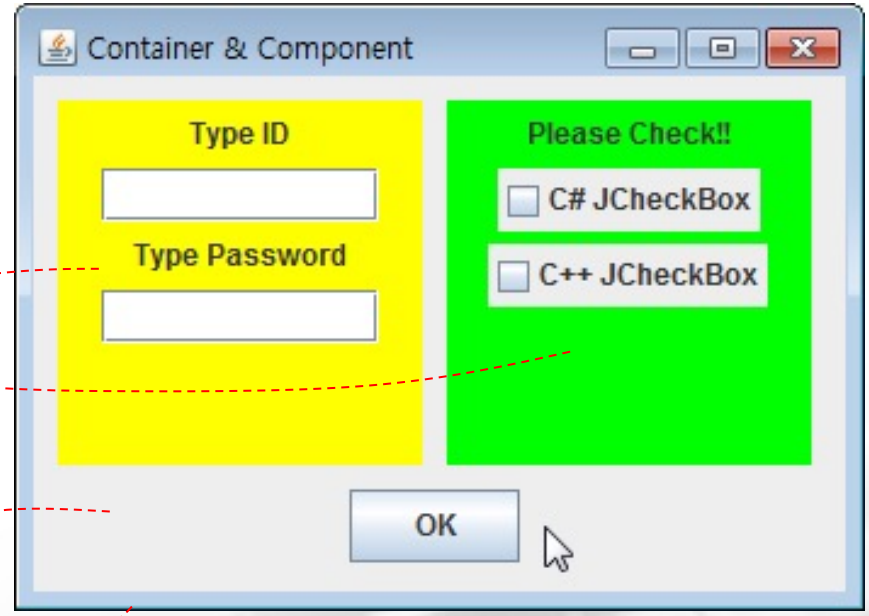
자바 GUI

- Component: 화면을 구성하는 부품
  - ✓ 화면에 출력될 수 있는 GUI 인스턴스
  - ✓ 모든 GUI Component의 최상위 클래스: java.awt.Component
  - ✓ 스윙 Component의 최상위 클래스: javax.swing.Jcomponent
- 컨테이너는 Component로부터 상속받은 하나의 윈도우 영역을 의미하는 것으로 자신의 영역에 다른 Component 또는 컨테이너를 포함시키고 관리하는 역할
  - ✓ 다른 Component를 포함할 수 있는 GUI Component
  - ✓ java.awt.Container를 상속받음
  - ✓ 다른 컨테이너에 포함될 수 있음
  - ✓ AWT 컨테이너: Panel, Frame, Applet, Dialog, Window
  - ✓ Swing 컨테이너: JPanel JFrame, JApplet, JDialog, JWindow
  - ✓ 최상위 컨테이너: 다른 컨테이너에 속하지 않고 독립적으로 존재 가능한 컨테이너
  - ✓ 스스로 화면 출력이 가능한 컨테이너: JFrame, JDialog, JApplet
  - ✓ Component는 화면에 출력이 가능한 컨테이너에 포함되어야 화면에 출력 가능

자바 GUI



AWT의 컨테이너와 Component의 포함 관계

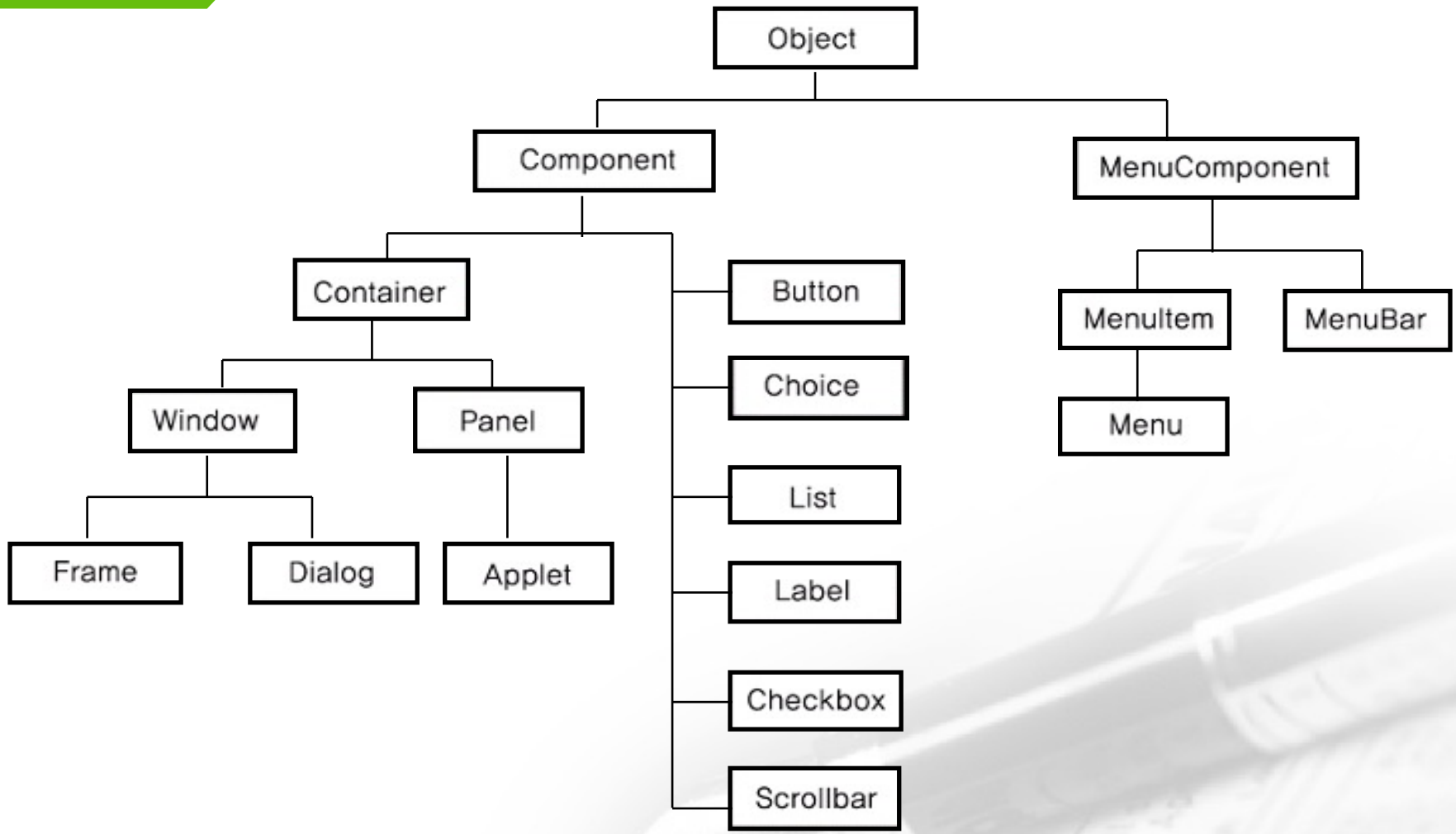


스윙 GUI 프로그램

자바 GUI

- 자주 사용하는 AWT 패키지
  - ✓ java.awt
  - ✓ java.awt.event
  - ✓ Java.awt.font
  - ✓ Java.awt.image

AWT Container



AWT Container

❖ Component의 주요 메서드

메서드	설 명
Color getBackground()	컴포넌트의 배경색을 얻는다.
void setBackground(Color c)	컴포넌트의 배경색을 지정된 색으로 변경한다.
Cursor getCursor()	컴포넌트에 지정된 커서를 얻는다.
void setCursor(Cursor c)	컴포넌트의 커서(마우스포인터)를 지정한다.
Font getFont()	컴포넌트에 지정되어 있는 Font를 얻는다.
void setFont(Font f)	컴포넌트의 Font를 지정한다.
Color getForeground()	컴포넌트의 전경색을 얻는다.
void setForeground(Color c)	컴포넌트의 전경색을 지정한다.
int getHeigtht()	컴포넌트의 높이(Height)를 얻는다.
int getWidth()	컴포넌트의 폭(Width)를 얻는다.
void setBounds(int x, int y, int width, int height)	컴포넌트의 위치(x, y)와 크기(width, height)를 지정한다.
Rectangle getBounds()	컴포넌트의 위치와 크기(Rectangle객체)를 얻는다.
Point getLocation()	컴포넌트의 위치를 얻는다.
void setLocation(int x, int y)	컴포넌트의 위치를 지정한다.
Dimension getSize()	컴포넌트의 크기를 얻는다.
void setSize(int width, int height)	컴포넌트의 크기를 지정한다.
boolean hasFocus()	컴포넌트가 현재 focus를 갖고 있는지 알려준다.
void requestFocus()	컴포넌트가 focus를 갖도록 한다.
void paint(Graphics g)	컴포넌트를 화면에 그린다.
void repaint()	컴포넌트를 화면에 다시 그린다.
void setEnabled(boolean b)	컴포넌트를 사용가능(true)/불가능(false) 하게 한다.
Container getParent()	컴포넌트가 포함되어져 있는 컨테이너(parent)를 얻는다.
void setVisible(boolean b)	컴포넌트가 화면에 보이게(true)/보이지 않게(false) 한다.

AWT Container

- Container
  - ✓ 독립적인 컨테이너 – 독립적으로 사용될 수 있으며 다른 Component나 종속적인 컨테이너를 포함할 수 있는 컨테이너

컨테이너	설 명
Frame	가장 일반적인 컨테이너로 윈도우와 모양이 같다. titlebar와 크기조절버튼, 닫기버튼을 가지고 있다. 그리고 메뉴를 추가할 수 있다.
Window	Frame의 조상이며, 경계선, titlebar, 크기조절버튼, 닫기 버튼이 없으며, 메뉴도 추가할 수 없다. 단지 컴포넌트를 담을 수 있는 평면공간만을 갖는다.
Dialog	Frame처럼, titlebar와 닫기버튼을 갖고 있지만, 메뉴는 가질 수 없으며 기본적으로 크기를 변경할 수 없다. 주로 프로그램사용자에게 메시지를 보여 주거나, 응답을 받는데 사용한다.

  - ✓ 종속적인 컨테이너 – 독립적으로 사용될 수 없으며 다른 컨테이너에 포함되어야 하며 다른 Component나 종속적인 컨테이너를 포함할 수 있는 컨테이너

컨테이너	설 명
Panel	평면공간으로 Frame과 같이 여러 컴포넌트를 담을 수 있으나 단독적으로 사용될 수는 없다.
ScrollPane	Panel과 같은 평면공간이지만, Panel과는 달리 단 하나의 컴포넌트만 포함할 수 있으며 자신보다 큰 컴포넌트가 포함되면 스크롤바가 자동적으로 나타난다.

AWT Container

- Container
  - ✓ Container의 주요 메서드

메서드	설 명
Component[] getComponents()	컨테이너에 포함되어 있는 모든 컴포넌트를 얻는다.
Component getComponent(int n)	컨테이너에 n번째로 추가된 컴포넌트를 얻는다.
Component getComponentAt(int x, int y)	컨테이너의 지정된 위치(x, y)에 있는 컴포넌트를 얻는다.
Component add(Component comp)	컨테이너에 컴포넌트를 추가한다.
void remove(Component comp)	컨테이너에서 지정된 컴포넌트를 제거한다.
Insets getInsets()	컨테이너의 경계의 크기를 알 수 있는 Inset객체를 얻는다.
LayoutManager getLayout()	컨테이너에 설정되어 있는 LayoutManager를 얻는다.
void setLayout(LayoutManager mgr)	컨테이너에 LayoutManager를 설정한다.

AWT Container

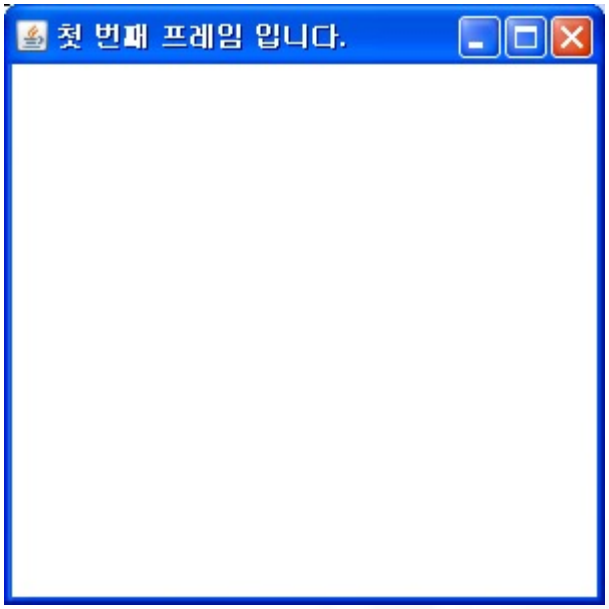
- Frame
  - ✓ 일반적인 응용프로그램에서 윈도우를 생성하기 위해 사용
  - ✓ 기본적으로 타이틀, 최소 버튼, 최대 버튼, 종료 버튼을 제공
  - ✓ 상위 클래스인 Window는 타이틀과 메뉴가 지원되지 않음
  - ✓ 화면에 안보이게 설정되어 있기 때문에 setVisible(true)을 호출해야 화면 상에 출력됨
  - ✓ Frame 클래스의 인스턴스를 생성해서 사용하기도 하고 Frame으로부터 상속받는 클래스를 이용해서 사용하기도 함
  - ✓ 생성자
    - ❑ Frame(): 타이틀이 빈 상태로 생성
    - ❑ Frame(String str): str을 타이틀을 지정하여 프레임을 생성

AWT Container

- Frame
  - ✓ Frame의 주요 메서드
    - ❑ static Frame[] getFrames(): 애플리케이션에서 생성한 모든 프레임을 리턴
    - ❑ MenuBar getMenuBar(): 프레임의 메뉴 바를 리턴
    - ❑ int getState(): 프레임의 상태를 리턴
    - ❑ String getTitle(): 타이틀 바의 문자열을 리턴
    - ❑ void remove(MenuComponent m): 프레임에서 지정한 메뉴를 제거
    - ❑ void setMenuBar(MenuBar mb): 프레임의 메뉴바를 지정
    - ❑ void setResizable(Boolean resizeable): 프레임의 크기를 사용자가 변경할 수 있게 할 것인지를 지정
    - ❑ void setSize (int width, int height): Component의 사이즈를 width 및 height 로 변경
    - ❑ void setVisible (boolean b): 파라미터 b 값에 따라 Component를 표시할 지 여부를 설정
    - ❑ void setLocation (int x, int y): 이 Component를 새로운 위치로 이동
    - ❑ void setBounds(int x, int y, int width, int height): 위치와 크기를 설정

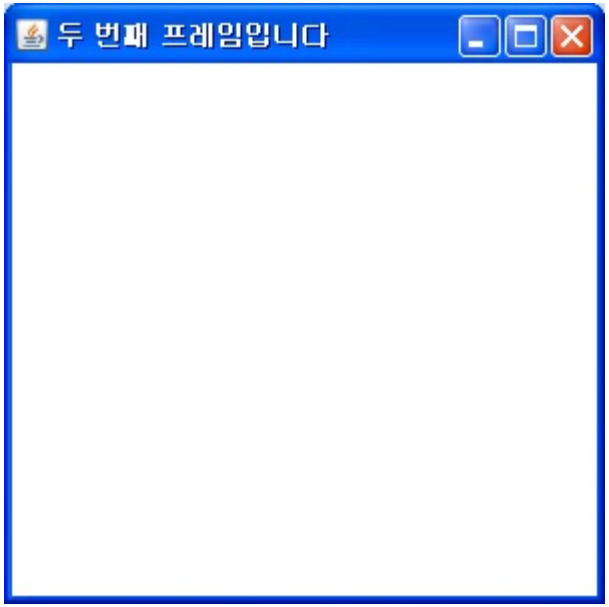
실습(FrameTest1.java)

```
import java.awt.*;
public class FrameTest1
{
        public static void main(String args[])
        {
                Frame f = new Frame();
                f.setTitle("첫 번째 프레임 입니다.");
                f.setBounds(100,100, 300,300);
                f.setVisible(true);
        }
}
```



실습(FrameTest.java)

```
import java.awt.*;
public class FrameTest extends Frame{
		public FrameTest()
		{
			super("두 번째 프레임입니다");
			setBounds(100,100, 300,300);
			setVisible(true);
		}
}
```



실습(FrameMain.java)

```
public class FrameMain {
        public static void main(String args[]) {
                FrameTest Obj = new FrameTest();
        }
}
```

AWT Container

- Panel
  - ✓ Component들을 그룹 별로 묶어서 처리할 때 사용
  - ✓ Frame에 Component들을 직접 붙이지 않고 Panel에 그룹 별로 붙이고 다시 Panel을 Frame에 붙이는 경우가 많음
  - ✓ 생성자
    - ❑ Panel(): 디폴트 레이아웃 매니저를 사용해 새로운 Panel을 생성
    - ❑ Panel(LayoutManager layout): 지정된 레이아웃 매니저를 이용하여 새로운 Panel을 생성

실습(PanelTest.java)

```
import java.awt.*;

public class PanelTest extends Frame
{
        public PanelTest(String str)
        {
                super(str);
                Panel panel1 = new Panel();
                panel1.setBackground(Color.lightGray);
                add(panel1);
                setSize(300,300);
                setVisible(true);
        }
}
```

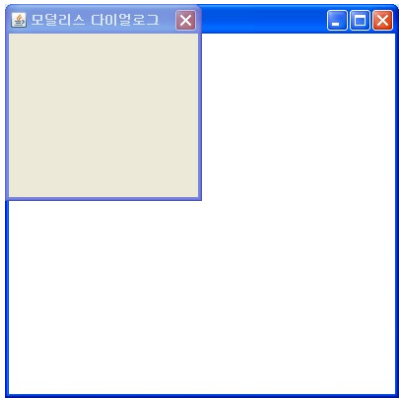
실습(PanelMain.java)

```
public class PanelMain {
		public static void main(String[] args)
		{
			new PanelTest("패널 테스트");
		}
}
```

AWT Container

- Dialog
  - ✓ 메인 윈도우 외에 메시지를 출력하거나 사용자로부터 자료를 입력 받을 때 사용하는 컨테이너
  - ✓ Modal Dialog: 화면에 출력되어 있는 동안 다른 윈도우로 제어권 이동이 되지 않는 대화상자 – 반대는 Modeless Dialog
  - ✓ 생성자
    - ❑ Dialog(Dialog owner): 제어권을 소유하는 Dialog가 owner인 Modeless Dialog를 생성
    - ❑ Dialog(Dialog owner, String title): title을 가진 Modeless Dialog를 생성
    - ❑ Dialog(Dialog owner, String title, Boolean modal): modal이 true이면 modal 다이얼로그를 아니면 modeless 다이얼로그를 생성
    - ❑ Dialog(Frame owner): 제어권을 owner 프레임이 소유하는 Modeless 다이얼로그 생성
    - ❑ Dialog(Frame owner, Boolean modal)
    - ❑ Dialog(Frame owner, String title)
    - ❑ Dialog(Frame owner, String title, Boolean modal)

실습(ModelessDialog.java)




```
import java.awt.*;
class ModelessDialog extends Frame
{
        static final long serialVersionUID = 1;
        public ModelessDialog()
        {
                super("다이얼로그 테스트");
                Dialog d = new Dialog(this, "모덜리스 다이얼로그");
                setBounds(0,0, 400, 400);
                setVisible(true);
                d.setSize(200,200);
                d.setVisible(true);
        }
}
```

실습(DialogMain.java)

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

AWT Component

- Button : 사용자가 클릭했을 때 작업을 수행하도록 하고자 하는 경우 사용하는 Component
  - ✓ 생성자
    - ❑ Button()
    - ❑ Button(String label)
  - ✓ 메서드
    - ❑ String getLabel()
    - ❑ void setLabel()

실습(버튼 만들기 – ButtonFrame.java)

```
import java.awt.Button;
import java.awt.Frame;
import java.awt.Panel;

public class ButtonFrame extends Frame {
        Button btn1, btn2, btn3;

        public ButtonFrame(String str) {
                super(str);
                Panel p = new Panel();
                btn1 = new Button(" 가위 ");
                btn2 = new Button(" 바위 ");
                btn3 = new Button("  보  ");
                p.add(btn1);
                p.add(btn2);
                p.add(btn3);
                add(p);
                btn3.setEnabled(false);
                setSize(200, 200);
                setVisible(true);
        }
}
```

실습(버튼 만들기-ButtonMain.java)

```
public class ButtonMain {
		public static void main(String[] args) {
			new ButtonFrame("버튼 만들기");
		}
}
```

AWT Component

- CheckBox
  - ✓ CheckBox(): 레이블이 없는 체크 박스 생성
  - ✓ CheckBox(String label): label을 설정한 체크 박스 생성
  - ✓ CheckBox(String label, boolean state): 지정된 label과 state를 넣어서 생성
  - ✓ Checkbox(String label, CheckboxGroup group, boolean state): 지정된 label이 붙은 체크 박스를 지정된 체크 박스 그룹에 구축해 지정된 상태로 생성
  - ✓ CheckboxGroup 인스턴스를 생성해서 지정해주면 라디오 버튼

실습(체크박스와 라디오 버튼)

```
import java.awt.*;
class Checkbox1 extends Frame {
        public Checkbox1(String str){
                super(str);
                Panel p = new Panel();
                Checkbox cbx1=new Checkbox("이대",true);
                Checkbox cbx2=new Checkbox("강남");
                Checkbox cbx3=new Checkbox("종로");
                p.add(cbx1);
                p.add(cbx2);
                p.add(cbx3);

                CheckboxGroup group = new CheckboxGroup();
                Checkbox cbx4=new Checkbox("이대",group,true);
                Checkbox cbx5=new Checkbox("강남",group,false);
                Checkbox cbx6=new Checkbox("종로",group,false);
                p.add(cbx4);
                p.add(cbx5);
                p.add(cbx6);

                add(p);
                setSize(180, 300);
                setVisible(true);
        }
}
```

실습(체크박스와 라디오 버튼)

```
public class CheckBoxMain
{
		public static void main(String[] args)
		{
			new Checkbox1("체크 박스 테스트");
		}
}
```

AWT Component

- Choice
  - ✓ List와 유사한 기능을 가지는 여러 아이템 중 하나만 선택할 수 있도록 해주는 Component
  - ✓ 생성자
    - ❑ Choice(): 생성
  - ✓ 주요 메서드
    - ❑ void add(String item): 항목을 추가
    - ❑ String getItem(int index): 선택 항목 리턴
    - ❑ int getItemCount(): 항목 수를 리턴
    - ❑ int getSelectedIndex(): 현재 선택된 항목의 인덱스를 리턴
    - ❑ String getSelectedItem(): 현재 선택된 항목의 문자열을 리턴
    - ❑ void insert(String item, int index): 항목을 지정된 위치에 삽입
    - ❑ void remove(int position): 지정된 위치의 항목을 삭제
    - ❑ void remove(String item): item 처음 표시되는 항목을 삭제
    - ❑ void removeAll(): 모든 항목 삭제
    - ❑ void select(int pos): 위치를 설정
    - ❑ void select (String str): 지정된 항목으로 설정

실습(콤보박스)


```
import java.awt.* ;
class ChoiceTest extends Frame
{
        Choice ch;
        public ChoiceTest(String str)
        {
                super(str);
                ch= new Choice();
                ch.addItem("이대");
                ch.addItem("강남");
                ch.addItem("종로");
                add(ch);
                setSize(300,100);
                setVisible(true);
        }
}
```

실습(콤보박스)

```
public class ChoiceMain
{
		public static void main(String[] args)
		{
			new ChoiceTest("콤보 박스 연습");
		}
}
```

AWT Component

- Label
  - ✓ 사각형의 영역에 문자열을 표시할 때 사용하는 Component
  - ✓ 생성자
    - ❑ Label()
    - ❑ Label(String text)
    - ❑ Label(String text, int alignment)
    - ❑ Label.CENTER, LEFT, RIGHT 등이 alignment에 사용
  - ✓ 메서드
    - ❑ getText(): text를 리턴
    - ❑ setText(String text): text로 레이블 설정
    - ❑ setAlignment(int align): 정렬 상태 설정

실습(레이블)

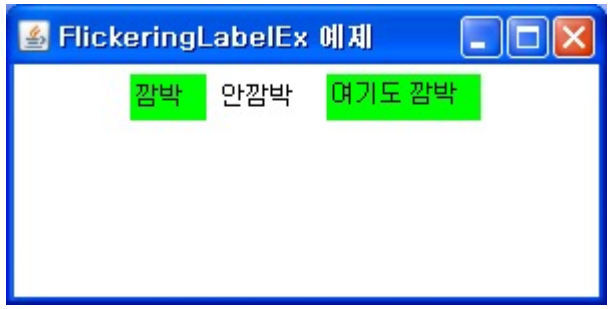

```
import java.awt.*;

class LabelTest extends Frame {
        Panel p;
        Label label1, label2, label3;

        public LabelTest(String str) {
                super(str);
                p = new Panel();
                label1 = new Label("이대");
                label2 = new Label("강남", Label.CENTER);
                label3 = new Label("종로", Label.LEFT);
                label1.setBackground(Color.red);
                label2.setBackground(Color.blue);
                label3.setBackground(Color.green);
                p.add(label1);
                p.add(label2);
                p.add(label3);
                add(p);
                setSize(300, 300);
                setVisible(true);
        }
}
```

실습(레이블)

```
public class LabelMain
{
		public static void main(String[] args)
		{
			new LabelTest("레이블 테스트");
		}
}
```

실습(깜박이는 레이블)

```
import java.awt.*;
class FlickeringLabel extends Label implements Runnable{
        public FlickeringLabel(String text) {
                super(text);
                Thread th = new Thread(this);
                th.start();
        }
        public void run() {
                int n=0;
                while(true) {
                        if(n == 0)
                                setBackground(Color.YELLOW);
                        else
                                setBackground(Color.GREEN);
                        if(n == 0) n = 1;
                        else n = 0;
                        try {
                                Thread.sleep(500);
                        }
                        catch(InterruptedException e) {
                                return;
                        }
                }
        }
}
```



실습(깜박이는 레이블)

```
class FlickeringLabelEx extends Frame {
        public FlickeringLabelEx() {
                setTitle("FlickeringLabelEx 예제");

                // 깜박이는 레이블 생성
                FlickeringLabel fLabel = new FlickeringLabel("깜박");
                // 깜박이지 않는 레이블 생성
                Label label = new Label("안깜박");
                // 깜박이는 레이블 생성
                FlickeringLabel fLabel2 = new FlickeringLabel("여기도 깜박");
                Panel p = new Panel();
                p.add(fLabel);
                p.add(label);
                p.add(fLabel2);
                add(p);
                setSize(300,150);
                setVisible(true);
        }
}
```

실습(깜박이는 레이블)

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

AWT Component

- List
  - ✓ 여러 개의 항목을 보여주고 사용자가 하나 또는 여러 개의 항목을 선택할 수 있도록 지원하는 Component
  - ✓ 기본적으로 하나의 항목을 선택할 수 있지만 MultipleMode를 설정하면 한 번에 여러 개의 항목을 선택 가능
  - ✓ 생성자
    - ❑ List(): 비어 있는 리스트 생성
    - ❑ List(int rows): 숫자만큼의 항목을 보여주는 리스트 생성
    - ❑ List(int rows, boolean mutipleMode): 숫자만큼의 항목을 보여주고 멀티 선택이 가능한 리스트 생성

AWT Component

- List
  - 메서드
    - void add (String item): item 추가
    - void add (String item, int index): 선택한 위치에 item 추가
    - void deselect (int index): 지정된 인덱스에 있는 항목을 선택 해제
    - String getItem (int index): 인덱스에 해당하는 항목을 리턴
    - int getItemCount(): 항목 수를 리턴
    - String [] getItems(): 리스트내의 항목을 리턴
    - int getSelectedIndex(): 선택된 항목의 인덱스 리턴
    - int[] getSelectedIndexes(): 선택된 항목들의 인덱스 리턴
    - String getSelectedItem(): 선택된 항목의 문자열을 리턴
    - String [] getSelectedItems(): 선택된 항목들의 문자열을 리턴
    - void remove(int position): 항목 삭제
    - void remove(String item): item 항목 삭제
    - void removeAll(): 항목 전체 삭제

실습(리스트)

```
import java.awt.*;
class ListTest extends Frame
{
		Panel p;
		List list;
		public ListTest(String str)
		{
				super(str);
				p=new Panel();
				list=new List(2,true);
				list.add("이대");
				list.add("강남");
				list.add("종로");
				list.add("구로");
				p.add(list);
				add(p);
				setSize(300,100);
				setVisible(true);
		}
}
public class ListMain{
		public static void main(String[] args) {
				new ListTest("멀티 선택 리스트");
		}
}
```

AWT Component

- TextField
  - ✓ 한 줄 내의 텍스트를 입력 받거나 편집할 수 있는 Component
  - ✓ 생성자
    - ❑ TextField(): 비어있는 텍스트 필드 인스턴스 생성
    - ❑ TextField(int columns): 지정한 컬럼 수만큼 문자를 보여줄 수 있는 크기로 텍스트 필드 인스턴스 생성
    - ❑ TextField(String text): 초기 텍스트를 설정해서 생성
    - ❑ TextField(String text, int columns): 초기 텍스트를 설정하고 보여줄 수 있는 크기를 설정해서 생성
  - ✓ 메서드
    - ❑ setEchoChar(char c): 보여지는 문자를 설정
    - ❑ int getCaretPosition (): 텍스트 내에서 현재 마우스 포인터의 위치를 리턴
    - ❑ String getText (): 텍스트 내의 표시되는 텍스트를 리턴
    - ❑ void select (int selectionStart, int selectionEnd): 시작위치부터 마지막 위치까지 선택
    - ❑ void selectAll (): 텍스트 Component 내의 모든 텍스트를 선택
    - ❑ void setBackground (Color c): 백그라운드 칼라를 설정
    - ❑ void setCaretPosition (int position): caret의 위치를 설정
    - ❑ void setText (String t): 표시되는 텍스트를 지정된 텍스트로 설정

실습(텍스트필드)

```
import java.awt.* ;
class TextFieldTest extends Frame
{
        public TextFieldTest(String str)
        {
                super(str);
                Panel p = new Panel();
                Label lbl1 = new Label(" 아 이 디:");
                Label lbl2 = new Label(" 비밀번호:");
                TextField txt1 = new TextField("ID", 20 );
                TextField txt2 = new TextField(20);
                txt2.setEchoChar('*');
                p.add(lbl1);
                p.add(txt1);
                p.add(lbl2);
                p.add(txt2);
                add(p);
                setSize(200,200);
                setVisible(true);
        }
}
```

실습(텍스트필드)

```
public class TextFieldMain {
		public static void main(String[] args) {
				new TextFieldTest("TextField 테스트");
		}

}
```

AWT Component

- TextArea
  - ✓ 여러 줄의 텍스트를 입력 받거나 편집할 수 있는 Component
  - ✓ 상수
    - ❑ static int SCROLLBARS_BOTH: 수평과 수직 스크롤 바
    - ❑ static int SCROLLBARS_HORIZONTAL_ONLY: 수평 스크롤 바
    - ❑ static int SCROLLBARS_NONE
    - ❑ static int SCROLLBARS_VERTICAL_ONLY: 수직 스크롤 바
  - ✓ 생성자
    - ❑ TextArea(): 새로운 텍스트 영역을 생성
    - ❑ TextArea(int rows, int columns): 행수와 열수를 지정해서 생성
    - ❑ TextArea(String text): 초기값을 설정
    - ❑ TextArea (String text, int rows, int columns)
    - ❑ TextArea (String text, int rows, int columns, int scrollbars): 행수와 열수를 지정하고 스크롤 바 표시여부를 설정해서 생성

실습(TextArea)

```
import java.awt.* ;
class TextAreaTest extends Frame {
        public TextAreaTest(String str){
                super(str);
                Panel p = new Panel();
                TextArea txt1 = new TextArea(10,30);
                TextField txt2 = new TextField("Hello Java",20);
                txt1.setText("  Java World ");
                txt1.setBackground(Color.yellow);
                txt1.setFont(new Font("궁서체",Font.BOLD, 10));
                txt1.setForeground(Color.blue);
                txt2.setEchoChar('@');
                txt2.setForeground(Color.red);
                p.add(txt1);
                p.add(txt2);
                add(p);
                setSize(200,200);
                setVisible(true);
        }
}
```

실습(TextArea)

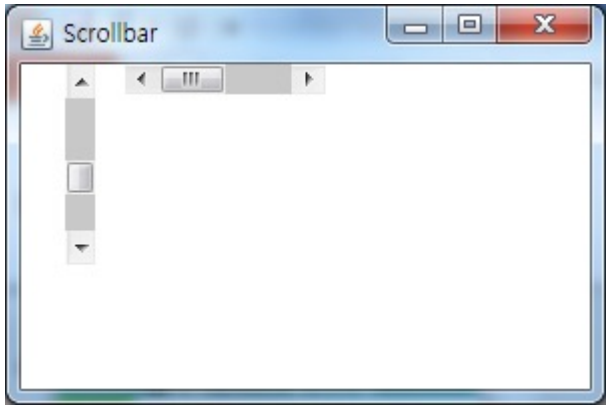
```
public class TextAreaMain {
		public static void main(String[] args) {
				new TextAreaTest("Text 테스트");
		}

}
```

AWT Component

❖ ScrollBar: 사용자가 범위 내의 값을 선택하도록 하고자 하는 경우 사용

메서드 또는 생성자	설 명
Scrollbar() Scrollbar(int orientation) Scrollbar(int orientation, int value, int visible, int min, int max)	orientation - Scrollbar의 종류를 지정해준다. Scrollbar.VERTICAL, Scrollbar.HORIZONTAL 중의 하나 value - Scrollbar의 초기값 visible - Scroll버튼(bubble)의 크기 min - Scrollbar가 가질 수 있는 최소값 max - Scrollbar가 가질 수 있는 최대값
int getValue()	현재 설정된 Scrollbar의 값을 얻어온다.
void setValue(int newValue)	Scrollbar의 값을 지정된 값(newValue)으로 설정한다.

실습(스크롤 바)



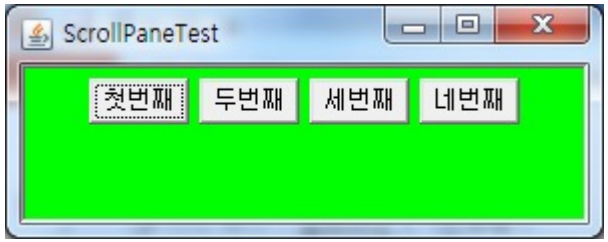

```
import java.awt.*;

class ScrollbarTest {
        public static void main(String args[]) {
                Frame f = new Frame("Scrollbar");
                f.setSize(300, 200);
                f.setLayout(null);

                Scrollbar hor = new Scrollbar(Scrollbar.HORIZONTAL, 0, 50, 0, 100);
                hor.setSize(100, 15);
                hor.setLocation(60, 30);
                Scrollbar ver = new Scrollbar(Scrollbar.VERTICAL, 50, 20, 0, 100);
                ver.setSize(15, 100);
                ver.setLocation(30, 30);

                f.add(hor);
                f.add(ver);
                f.setVisible(true);
        }
}
```

AWT Component

- ScrollPane: 한정된 영역에서 영역보다 큰 Component를 출력할 때 사용하는 것으로 하나의 Component만 배치 가능

메서드 또는 생성자	설 명
ScrollPane(int scrollbarDisplayPolicy)	srollbarDisplayPolicy - 아래 값 중의 하나를 지정한다. SCROLLBARS_ALWAYS - 스크롤바가 항상 보이게 한다. SCROLLBARS_AS_NEEDED - 필요할 때만 보이게 한다. SCROLLBARS_NEVER - 항상 보이지 않도록 한다.
ScrollPane()	ScrollPane의 객체를 생성한다.

실습(스크롤 패인)




```
import java.awt.*;

class ScrollPaneTest {
        public static void main(String args[]) {
                Frame f = new Frame("ScrollPaneTest");
                f.setSize(300, 200);

                ScrollPane sp = new ScrollPane();
                Panel p = new Panel();
                p.setBackground(Color.green);
                p.add(new Button("첫번째"));                    // Button을 Panel에 포함
                p.add(new Button("두번째"));
                p.add(new Button("세번째"));
                p.add(new Button("네번째"));

                sp.add(p);              // Panel을 ScrollPane에 포함시킨다.
                f.add(sp);              // ScrollPane을 Frame에 포함시킨다.
                f.setVisible(true);
        }
}
```

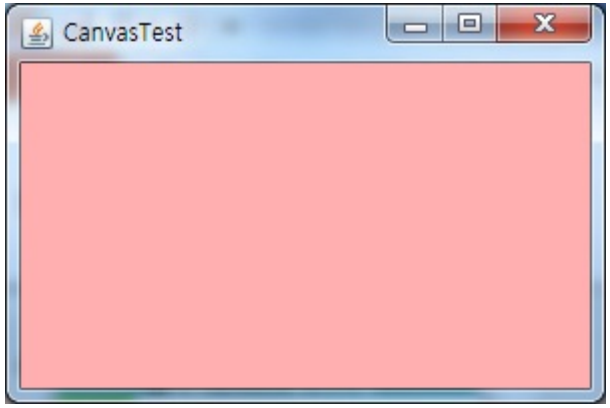
AWT Component

- Canvas: 특정한 모양을 갖지 않고 사각형의 영역만을 가지는 Component로 주로 그림을 표시할 때 사용
- Menu
  - ✓ MenuBar: 메뉴 표시줄을 만들 때 사용
  - ✓ Menu: 메뉴 표시줄에 보이게 될 메뉴를 만들 때 사용
  - ✓ MenuItem: 메뉴에 포함될 세부 메뉴를 만들 때 사용
  - ✓ CheckboxMenuItem: 체크 박스가 있는 메뉴
  - ✓ PopupMenu: 동적 메뉴를 만들 때 사용
  - ✓ 생성

```
MenuBar 메뉴바인스턴스 = new MenuBar();
Menu 메뉴인스턴스 = new Menu(메뉴명);
메뉴바인스턴스.add(메뉴 인스턴스);
MenuItem 세부메뉴인스턴스 = new MenuItem("세부 메뉴 명);
메뉴명.add(세부메뉴명);
```

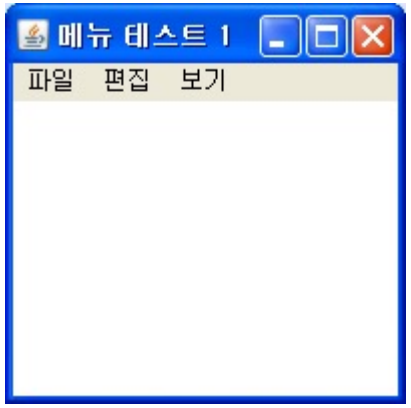
실습(캔버스)

```
import java.awt.*;

class CanvasTest {
        public static void main(String args[]) {
                Frame f = new Frame("CanvasTest");
                f.setSize(300, 200);
                Canvas c = new Canvas();
                c.setBackground(Color.pink);

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



실습(메뉴)

```
import java.awt.*;

class MenuTest extends Frame {
        public MenuTest(String str) {
                super(str);
                MenuBar mb = new MenuBar();

                Menu file = new Menu("파일");
                MenuItem file_new = new MenuItem("새로 만들기");
                MenuItem file_open = new MenuItem("열기");
                MenuItem file_save = new MenuItem("저장");
                MenuItem file_newname = new MenuItem("저장");

                file.add(file_new);
                file.add(file_open);
                file.add(file_save);
                file.add(file_newname);
```



실습(메뉴)

```
        Menu edit = new Menu("편집");
        MenuItem edit_undo = new MenuItem("실행취소");
        MenuItem edit_cut = new MenuItem("잘라내기");
        MenuItem edit_copy = new MenuItem("복사");
        MenuItem edit_paste = new MenuItem("붙여넣기");
        edit.add(edit_undo);
        edit.add(edit_cut);
        edit.add(edit_copy);
        edit.add(edit_paste);

        Menu view = new Menu("보기");
        CheckboxMenuItem view_status = new CheckboxMenuItem("상태표시줄");
        view.add(view_status);

        mb.add(file);
        mb.add(edit);
        mb.add(view);

        setMenuBar(mb);
        setSize(200, 200);
        setVisible(true);
    }
}
```

실습(메뉴)

```
public class MenuMain{
		public static void main(String[] args) {
				new MenuTest("메뉴 테스트");
		}
}
```

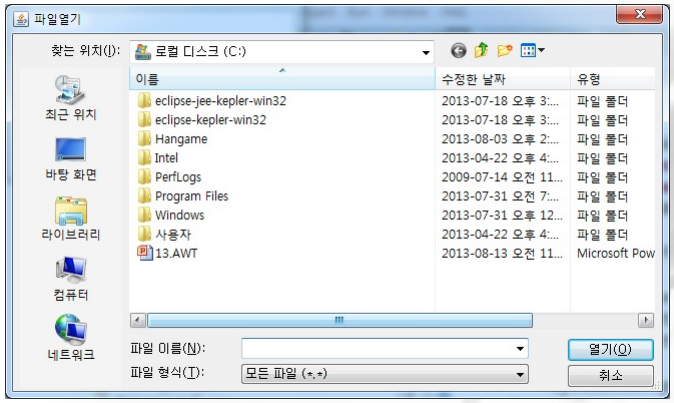
AWT Component

❖ FileDialog: 파일을 열거나 저장할 때 사용하는 대화상자

메서드 또는 생성자	설 명
FileDialog(Frame parent, String title, int mode)	parent - 어떤 Frame에 속한 것인지 지정. title - FileDialog의 titlebar에 나타날 text지정 mode - FileDialog.LOAD, FileDialog.SAVE 중 하나 선택
FileDialog(Frame parent, String title)	mode를 생략하면, 디폴드로 FileDialog.LOAD가 사용된다.
String getFile()	FileDialog에 의해 선택된 파일의 이름을 얻는다.
String getDirectory()	FileDialog에 의해 선택된 파일의 경로(path)를 얻는다.
void setFile(String file)	FileDialog에 지정된 파일을 설정한다.
void setDirectory(String dir)	FileDialog에 지정된 디렉토리를 설정한다.

실습(파일 대화 상자)

```
import java.awt.*;

public class FileDialogTest {
        public static void main(String args[]) {
                Frame f = new Frame("Parent");
                f.setSize(300, 200);

                FileDialog fileOpen = new FileDialog(f, "파일열기", FileDialog.LOAD);

                f.setVisible(true);
                fileOpen.setDirectory("./");
                fileOpen.setVisible(true);

                System.out.println(fileOpen.getDirectory() + fileOpen.getFile());
        }
}
```

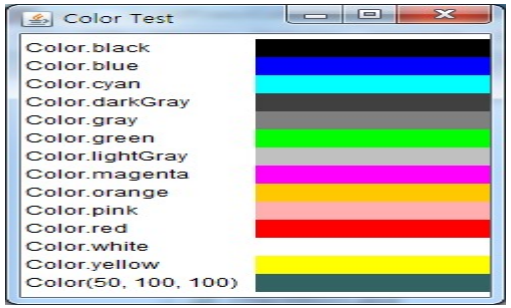


AWT Component

❖ Color:색상 관련 클래스

메서드 또는 생성자	설 명
Color(int r, int g, int b)	r - red, g - green, b - blue r, g, b 모두 0~255사이의 정수값을 갖는다.
Color(float r, float g, float b)	r - red, g - green, b - blue r, g, b 모두 0.0~1.0사이의 실수값을 갖는다.
Color(int r, int g, int b, int a)	a - alpha값(불투명도)으로 0~255사이의 정수값
Color(float r, float g, float b, float a)	a - alpha값(불투명도)으로 0.0~1.0사이의 실수값

실습(색상)




```
import java.awt.*;

class ColorTest {
        public static void main(String args[]) {
                Frame f = new Frame("Color Test");
                f.setLayout(new GridLayout(14, 2));
                Panel p1 = new Panel();
                p1.setBackground(Color.black);
                Panel p2 = new Panel();
                p2.setBackground(Color.blue);
                Panel p3 = new Panel();
                p3.setBackground(Color.cyan);
                Panel p4 = new Panel();
                p4.setBackground(Color.darkGray);
                Panel p5 = new Panel();
                p5.setBackground(Color.gray);
                Panel p6 = new Panel();
                p6.setBackground(Color.green);
                Panel p7 = new Panel();
                p7.setBackground(Color.lightGray);
                Panel p8 = new Panel();
                p8.setBackground(Color.magenta);
                Panel p9 = new Panel();
                p9.setBackground(Color.orange);
```

실습(색상)

```
Panel p10 = new Panel();
p10.setBackground(Color.pink);
Panel p11 = new Panel();
p11.setBackground(Color.red);
Panel p12 = new Panel();
p12.setBackground(Color.white);
Panel p13 = new Panel();
p13.setBackground(Color.yellow);
Panel p14 = new Panel();
p14.setBackground(new Color(50, 100, 100));
```

실습(색상)

```
f.add(new Label("Color.black"));
f.add(p1);
f.add(new Label("Color.blue"));
f.add(p2);
f.add(new Label("Color.cyan"));
f.add(p3);
f.add(new Label("Color.darkGray"));
f.add(p4);
f.add(new Label("Color.gray"));
f.add(p5);
f.add(new Label("Color.green"));
f.add(p6);
f.add(new Label("Color.lightGray"));
f.add(p7);
f.add(new Label("Color.magenta"));
f.add(p8);
f.add(new Label("Color.orange"));
f.add(p9);
f.add(new Label("Color.pink"));
f.add(p10);
f.add(new Label("Color.red"));
f.add(p11);
f.add(new Label("Color.white"));
f.add(p12);
```

실습(색상)

```
                f.add(new Label("Color.yellow"));
                f.add(p13);
                f.add(new Label("Color(50, 100, 100)"));
                f.add(p14);
                f.setSize(250, 300);
                f.setVisible(true);
        }
}
```

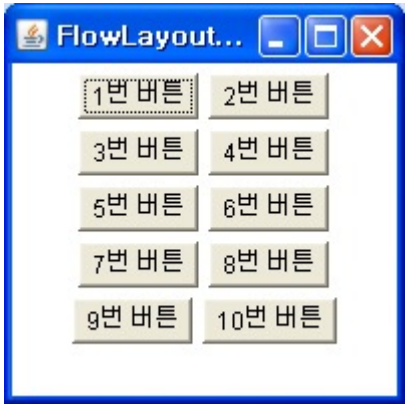
배치 관리자

- 배치 관리자
  - ✓ 레이아웃에 따라 Component들을 배치하기 위한 인스턴스
  - ✓ setLayout()을 이용해서 설정하고 종류로는 FlowLayout, BorderLayout, GridLayout, CardLayout, null 등 이 있음

- FlowLayout
  - ✓ Panel의 기본 레이아웃으로 Component들을 수평으로 순서대로 나열하는 레이아웃
  - ✓ 수평으로 배치하다가 더 이상 공간이 없으면 다음 줄로 이동해서 배치

실습(FlowLayout)

```
import java.awt.*;
class FlowLayoutTest extends Frame
{
        FlowLayout f = new FlowLayout();
        Button btn[] = new Button[10];
        public FlowLayoutTest(String str)
        {
                super(str);
                setLayout(f) ;
                for( int i = 0; i < 10; i++)
                {
                        btn[i] = new Button((i+1)+"번 버튼");
                        add(btn[i]);
                }
                setBounds(100,100,200,200);
                setVisible(true);
        }
}

public class FlowLayoutMain{
        public static void main(String[] args) {
                new FlowLayoutTest("FlowLayout 테스트");
        }
}
```



배치 관리자

- BorderLayout
  - ✓ 5개의 영역으로 구분하여 출력
  - ✓ 프레임은 기본적으로 BorderLayout

North
West Center East
South

  - ✓ Center 영역은 다른 영역에 아무것도 없으면 그 영역을 포함해서 보여줌
  - ✓ 하나의 영역에 한 개의 Component만 배치
  - ✓ add("위치", 배치할 Component)를 이용해서 배치
  - ✓ 위치는 add를 할 때 위의 방향을 문자열로 지정하고 뒤에 Component를 대입

실습(BorderLayout)

```
import java.awt.*;
class BorderLayoutTest extends Frame
{
		public BorderLayoutTest(String str)
		{
				super(str);
				setLayout(new BorderLayout());
				add("North", new Button("북쪽"));
				add("West", new Button("서쪽"));
				add("East", new Button("동쪽"));
				add("Center", new Button("중앙"));
				add("South", new Button("남쪽"));

				setSize(200,200);
				setVisible(true);
		}
}

public class BorderLayoutMain{
		public static void main(String[] args) {
				new BorderLayoutTest("BorderLayout 예제");
		}
}
```

배치 관리자

- GridLayout
  - ✓ 격자 모양으로 배치
  - ✓ 생성할 때 행과 열의 수를 지정해서 생성
  - ✓ Component는 add 메서드를 이용하면 순서대로 자동 추가
  - ✓ 행과 열 수보다 Component의 수가 많으면 자동적으로 늘어남

실습(GridLayout)

```
import java.awt.*;
class GridLayoutTest extends Frame
{
		Button btn[] = new Button[6];
		public GridLayoutTest(String str)
		{
			super(str);
			setLayout(new GridLayout(3,2) );
			for(int i = 0; i <  6; i++)
			{
				btn[i] = new Button(i+1 + "번 버튼");
				add(btn[i]);
			}
			setSize(200,200);
			setVisible(true);
		}
}


public class GridLayoutMain{
		public static void main(String[] args) {
			new GridLayoutTest("GridLayout Test");
		}
}
```

배치 관리자

- CardLayout
  - ✓ 여러 개의 카드를 쌓은 것처럼 보여주는 레이아웃으로 한번에 하나만 출력
  - ✓ next 메서드를 이용해서 다음 화면을 출력

- Layout에 null을 설정하면 좌표와 크기를 직접 설정해서 배치
  - ✓ Component를 원하는 크기로 원하는 곳에 직접 배치하고자 할 때 사용

실습(CardLayout)


```
import java.awt.*;
import java.awt.event.*;
class CardLayoutTest extends Frame {
        CardLayout card = new CardLayout();
        Panel panel[] = new Panel[5];
        Color color[] = {Color.red, Color.blue, Color.yellow, Color.green,Color.cyan};
        public CardLayoutTest(String str) {
                super(str);
                setLayout(card);

                MouseHandle mouse = new MouseHandle();

                for(int i = 0; i < 5; i++)
                {
                        panel[i] = new Panel();
                        panel[i].setBackground(color[i]);
                        panel[i].addMouseListener(mouse);
                        add(""+ (i+1), panel[i]);
                }
                setSize(200,200);
                setVisible(true);
        }
```

실습(CardLayout)

```
        public class MouseHandle extends MouseAdapter        {
                public void mouseClicked(MouseEvent e)       {
                        card.next(CardLayoutTest.this);
                }
        }
}


public class CardLayoutMain{
        public static void main(String[] args) {
                new CardLayoutTest("CardLayout Test");
        }
}
```

AWT 이벤트 처리

Event

- 이벤트
  - ✓ 사용자 또는 프로그램 코드에 의해서 발생하는 사건
  - ✓ 버튼을 클릭하거나 윈도우 종료단추를 클릭하는 행동
  - ✓ 이벤트가 발생하면 Java 프로그램은 JVM에게 이벤트를 전달하고 JVM은 발생된 이벤트를 처리하기 위해 이벤트 인스턴스를 생성해서 이벤트 인스턴스를 가지고 이벤트를 처리할 리스너가 등록되어 있는지 확인하고 이를 처리할 리스너가 존재한다면 리스너가 핸들러를 호출해서 처리
- ActionEvent: 버튼이 클릭되거나 리스트, 메뉴 등이 선택되었을 때 발생하는 이벤트
  - ✓ ActionListener 인터페이스의 actionPerformed(ActionEvent e) 메서드를 이용해서 처리
  - ✓ ActionEvent 클래스의 멤버

필드명	해당 키
ALT_MASK	ALT 키
CTRL_MASK	Ctrl 키
SHIFT_MASK	Shift 키

메소드	해당 키
getActionCommand()	이벤트를 발생시킨 객체의 문자열(화면에 보이는)을 가져온다.
getSource()	이벤트를 발생시킨 객체의 위치값을 가져온다.
getModifiers()	이벤트가 발생되었을 때 같이 사용된 modifier키들을 가져온다.

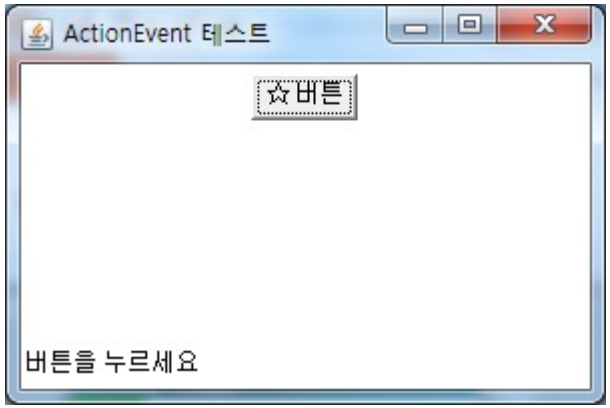
Event Handling

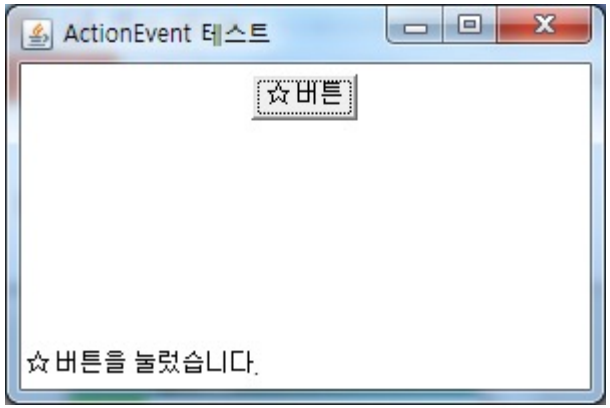
- 이벤트 처리
  - ✓ 처리하고자 하는 이벤트 Listener를 클래스에 implements
  - ✓ 이벤트에 따라 호출되는 메서드를 재정의
    - ❑ 넘어오는 매개변수의 getSource()가 이벤트를 발생시킨 Component를 리턴
  - ✓ Component에 이벤트 리스너를 추가
    - ❑ Component.add이벤트리스너(인스턴스명);

- 이벤트 핸들링 하는 방법
  - ✓ 하나의 클래스에서 출력과 처리를 모두 하는 방법
  - ✓ 출력과 처리를 서로 다른 클래스에서 하는 방법
  - ✓ Anonymous Class를 이용하는 방법

Event Handling

❖ 하나의 클래스에서 출력과 이벤트 처리





ActionEvent1.java

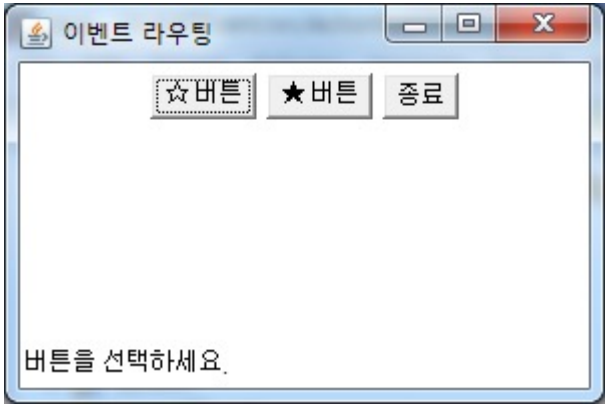
```
import java.awt.*;
import java.awt.event.*;
class ActionEvent1 extends Frame implements ActionListener
{
        Label lbl;
        Button btn;
        public ActionEvent1(String str)
        {
                super(str);
                lbl = new Label("버튼을 누르세요");
                btn = new Button("☆ 버튼");
                btn.addActionListener(this);
                Panel panel = new Panel();
                panel.add(btn);
                add("Center", panel);
                add("South", lbl);
                setSize(300,200);
                setVisible(true);
        }
        public void actionPerformed(ActionEvent e) {
                lbl.setText("☆ 버튼을 눌렀습니다.");
        }
}
```

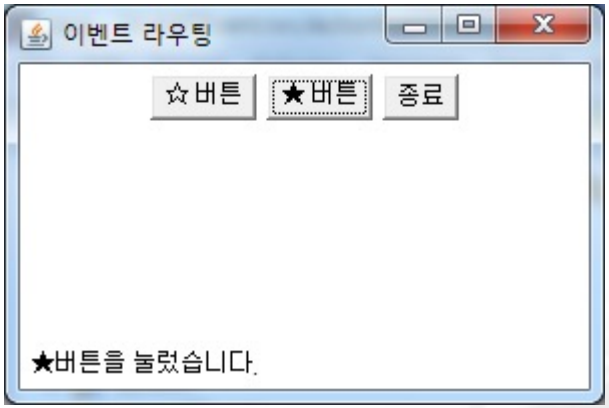
ActionEventMain.java

```
public class ActionEventMain {
		public static void main(String[] args) {
			new ActionEvent1("ActionEvent 테스트");
		}
}
```

Event Handling

- 이벤트 라우팅
  - ✓ 하나의 메서드에서 2개 이상의 인스턴스 이벤트를 처리하는 방법
  - ✓ 이벤트 처리 메서드의 매개변수가 가지고 있는 getActionCommand()나 getSource() 메서드를 이용





EventRoutingMain

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

class ActionEvent2 extends Frame implements ActionListener {
        Label lbl;
        Button btn1, btn2, btn3;

        public ActionEvent2(String str) {
                super(str);
                lbl = new Label("버튼을 선택하세요.");
                btn1 = new Button("☆ 버튼");
                btn2 = new Button("★ 버튼");
                btn3 = new Button("종료");
                btn1.addActionListener(this);
                btn2.addActionListener(this);
                btn3.addActionListener(this);
                Panel panel = new Panel();
                panel.add(btn1);
                panel.add(btn2);
                panel.add(btn3);
                add("Center", panel);
                add("South", lbl);
                setSize(300, 200);
                setVisible(true);
        }
```

EventRoutingMain

```
		public void actionPerformed(ActionEvent e) {
			Object obj = e.getSource();
			if ((Button) obj == btn1) {
				lbl.setText("☆ 버튼을 눌렀습니다.");
			} else if ((Button) obj == btn2) {
				lbl.setText(" ★버튼을 눌렀습니다.");
			} else if ((Button) obj == btn3) {
				lbl.setText("프로그램을 종료합니다.");
				try {
					Thread.sleep(2000);
				} catch (Exception e1) {
				}
				System.exit(0);
			}

		}
}

public class EventRoutingMain {
		public static void main(String[] args) {
			new ActionEvent2("이벤트 라우팅");
		}
}
```

ActionEventRouting

ActionEventRouting

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

class ActionEvent3 extends Frame implements ActionListener {
        TextField text;
        Button btn;
        Choice combo;
        // 중복된 데이터 저장을 막기 위해서 사용
        HashSet<String> hs = new HashSet<String>();

        public ActionEvent3() {
                super("Enter 테스트");
                setLayout(null);
                text = new TextField(20);
                btn = new Button("추가");
                combo = new Choice();
                text.setBounds(100, 100, 60, 20);
                btn.setBounds(170, 100, 40, 20);
                combo.setBounds(100, 130, 100, 40);
                add(btn);
                add(text);
```

ActionEventRouting

```
                add(combo);
                text.addActionListener(this);
                btn.addActionListener(this);
                setBounds(100, 100, 300, 300);
                setVisible(true);
        }

        public void actionPerformed(ActionEvent e) {
                if (hs.add(text.getText())) {
                        combo.add(text.getText());
                }
                text.setText("");
        }
}

public class ActionEventRouting {
        public static void main(String[] args) {
                new ActionEvent3();
        }

}
```

AWTPatternMain

❖ 동일한 예제를 2개의 클래스(출력과 이벤트 처리 부분)로 처리

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

class View extends Frame {
	TextField text;
	Button btn;
	Choice combo;
	// 중복된 데이터 저장을 막기 위해서 사용
	HashSet<String> hs = new HashSet<String>();

	public View() {
		super("Enter 테스트");
		setLayout(null);
		text = new TextField(20);
		btn = new Button("추가");
		combo = new Choice();
		text.setBounds(100, 100, 60, 20);
		btn.setBounds(170, 100, 40, 20);
		combo.setBounds(100, 130, 100, 40);
```

AWTPatternMain

```
                add(btn);
                add(text);
                add(combo);
                EventHandler eventHandler = new EventHandler(text, combo, hs);
                text.addActionListener(eventHandler);
                btn.addActionListener(eventHandler);
                setBounds(100, 100, 300, 300);
                setVisible(true);
        }
}
```

AWTPatternMain

```
class EventHandler implements ActionListener {
	private TextField text;
	private Choice combo;
	HashSet<String> hs;

	public EventHandler(TextField text, Choice combo, HashSet<String> hs) {
		super();
		this.text = text;
		this.combo = combo;
		this.hs = hs;
	}

	public void actionPerformed(ActionEvent e) {
		if (hs.add(text.getText())) {
			combo.add(text.getText());
		}
		text.setText("");
	}
}
```

AWTPatternMain

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

AnonymousEventMain

- 인터페이스나 추상 클래스로부터 상속받는 클래스는 상속하는 클래스를 만들 필요 없이 Anonymous 클래스를 이용하는 것이 가능 – 안드로이드에서 많이 이용

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

class ActionEvent5 extends Frame {
	TextField text;
	Button btn;
	Choice combo;
	// 중복된 데이터 저장을 막기 위해서 사용
	HashSet<String> hs = new HashSet<String>();

	public ActionEvent5() {
		super("Enter 테스트");
		setLayout(null);
		text = new TextField(20);
		btn = new Button("추가");
		combo = new Choice();
		text.setBounds(100, 100, 60, 20);
		btn.setBounds(170, 100, 40, 20);
		combo.setBounds(100, 130, 100, 40);
```

AnonymousEventMain

```
                add(btn);
                add(text);
                add(combo);
                text.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                if (hs.add(text.getText())) {
                                        combo.add(text.getText());
                                }
                                text.setText("");
                        }
                });
                btn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                if (hs.add(text.getText())) {
                                        combo.add(text.getText());
                                }
                                text.setText("");
                        }
                });
                setBounds(100, 100, 300, 300);
                setVisible(true);
        }
}
```

AnonymousEventMain

```
public class AnonymousEventMain {
	public static void main(String[] args) {
			new ActionEvent5();
	}
}
```

ItemEvent

- ItemEvent
  - ✓ 체크 박스나 라디오 버튼의 아이템을 선택하거나 해제하는 경우 또는 콤보 박스에서 아이템을 선택할 때 발생하는 이벤트
  - ✓ ItemListener 인터페이스의 itemStateChanged(ItemEvent e) 메서드를 이용해서 처리
  - ✓ ItemEvent 클래스의 멤버

필드명	해당 키
SELECTED	항목이 선택
DESELECTED	항목이 해제

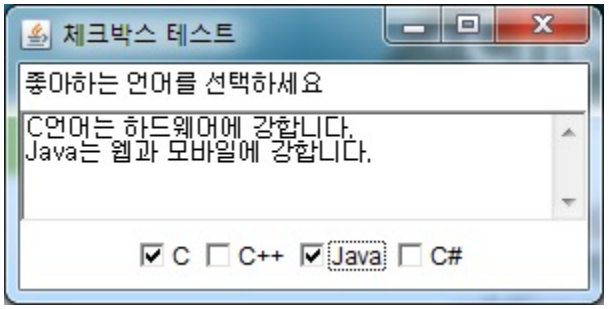
메소드	해당 키
getItem()	이벤트를 발생시킨 항목을 가져온다.
getStateChange()	이벤트의 발생으로 변화된 값을 가져온다.

ItemEvent

- ItemEvent 처리 - ItemListener
  - ✓ 체크박스나 Choice, 리스트 등에서 선택이 변경되었을 때 발생하는 이벤트
  - ✓ 체크박스의 상태는 getState()를 이용해서 확인 가능
  - ✓ Choice나 리스트는 getSelectedIndex()나 getSelectedItem()를 이용해서 현재 선택된 항목의 인덱스나 문자열을 리턴
  - ✓ getStateChange()를 이용해서 선택사항이 변경된 사실을 전달 받을 수 있음

CheckBoxEvent

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

class CheckBoxEvent extends Frame implements ItemListener {
        Panel panel;
        Checkbox ckb1, ckb2, ckb3, ckb4;
        TextArea ta;
        Label lbl;

        public CheckBoxEvent(String str) {
                        super(str);
                        lbl = new Label("좋아하는 언어를 선택하세요");
                        ta = new TextArea();
                        panel = new Panel();
                        ckb1 = new Checkbox("C");
                        ckb2 = new Checkbox("C++");
                        ckb3 = new Checkbox("Java");
                        ckb4 = new Checkbox("C#");

                        ckb1.addItemListener(this);
                        ckb2.addItemListener(this);
                        ckb3.addItemListener(this);
                        ckb4.addItemListener(this);
```



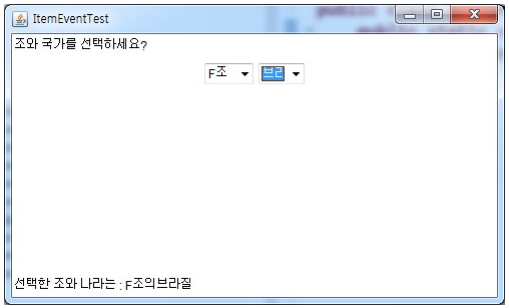
CheckBoxEvent

```
                panel.add(ckb1);
                panel.add(ckb2);
                panel.add(ckb3);
                panel.add(ckb4);
                add("North", lbl);
                add("Center", ta);
                add("South", panel);
                setSize(300, 150);
                setVisible(true);
        }

        public void itemStateChanged(ItemEvent e) {
                ta.setText("");
                if (ckb1.getState())
                        ta.append("C언어는 하드웨어에 강합니다. ₩n");
                if (ckb2.getState())
                        ta.append("C++는 STL이 훌륭합니다. ₩n");
                if (ckb3.getState())
                        ta.append("Java는 웹과 모바일에 강합니다. ₩n");
                if (ckb4.getState())
                        ta.append("C#은 윈도우 프로그램을 쉽게 만들 수 있습니다. ₩n");
        }
}
```

CheckBoxEvent

```
public class CheckBoxEventMain {
	public static void main(String[] args) {
			new CheckBoxEvent("체크박스 테스트");
	}
}
```

ItemEventMain




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

class ItemEventView extends Frame implements ItemListener {
		Choice lst_1, lst_2;
		Label lbl_info;

		String[] group = { "A조", "B조", "C조", "D조", "E조", "F조", "G조", "H조" };
		String[][] national = { { "독일", "에콰도르", "폴란드", "코스타리카" }, { "잉글랜드", "스웨덴
", "트리니다드 토바고", "파라과이" },
					{ "코트디부아르", "네델란드", "아르헨티나", "세르비아" }, { "포르투
칼", "멕시코", "이란", "앙골라" }, { "이탈리아", "체코", "미국", "가나" },
					{ "브라질", "일본", "호주", "크로아티아" }, { "프랑스", "스위스", "대
한민국", "토고" }, { "스페인", "우크라이나", "튀니지", "사우디아라비아" } };
```

ItemEventMain

```
public ItemEventView(String str) {
        super(str);
        Label lbl = new Label("조와 국가를 선택하세요?");
        Panel panel = new Panel();
        lbl_info = new Label();
        lst_1 = new Choice();
        lst_2 = new Choice();
        lst_1.addItemListener(this);
        lst_2.addItemListener(this);
        for (int i = 0; i < group.length; i++)
                lst_1.add(group[i]);

        lst_2.add("        ");
        panel.add(lst_1);
        panel.add(lst_2);

        add("North", lbl);
        add("Center", panel);
        add("South", lbl_info);

        setSize(500, 300);
        setVisible(true);
}
```

ItemEventMain

```
        public void itemStateChanged(ItemEvent e) {
                Choice obj = (Choice) e.getSource();
                String str = "선택한 조와 나라는 : ";
                if (obj == lst_1) {
                        lst_2.removeAll();
                        int j = lst_1.getSelectedIndex();
                        for (int i = 0; i < national[j].length; i++) {
                                lst_2.add(national[j][i]);
                        }
                } else {
                        str += lst_1.getSelectedItem();
                        str += "의" + lst_2.getSelectedItem();
                        lbl_info.setText(str);
                }
        }
}
```

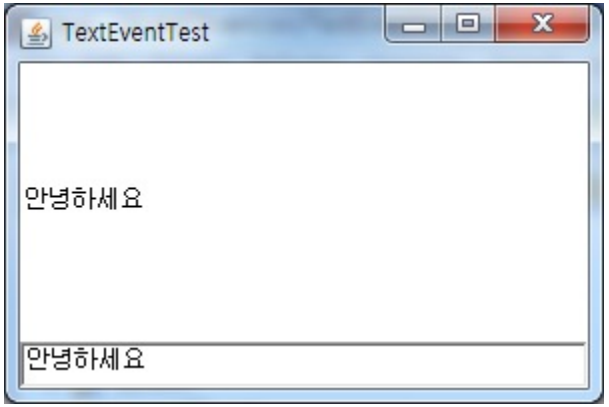
ItemEventMain

```
public class ItemEventMain {
	public static void main(String[] args) {
			new ItemEventView("아이템 이벤트");
	}

}
```

TextEvent

- Text Event
  - ✓ 텍스트 필드나 에어리어에 키보드 입력이 있을 때 처리하는 이벤트
  - ✓ TextListener 인터페이스의 textValueChanged(TextEvent e) 메서드로 처리

TextEvent




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

class TextEventView extends Frame implements TextListener {
        TextField txt;
        Label lbl;

        public TextEventView(String str) {
                super(str);
                txt = new TextField("텍스트를 입력하세요.", 20);
                txt.addTextListener(this);
                lbl = new Label("");
                add("Center", lbl);
                add("South", txt);
                setSize(300, 200);
                setVisible(true);
        }

        public void textValueChanged(TextEvent e) {
                lbl.setText(txt.getText());
        }
}
```

TextEvent

```
public class TextEventMain {
	public static void main(String[] args) {
		new TextEventView("텍스트 이벤트");
	}
}
```

KeyEvent

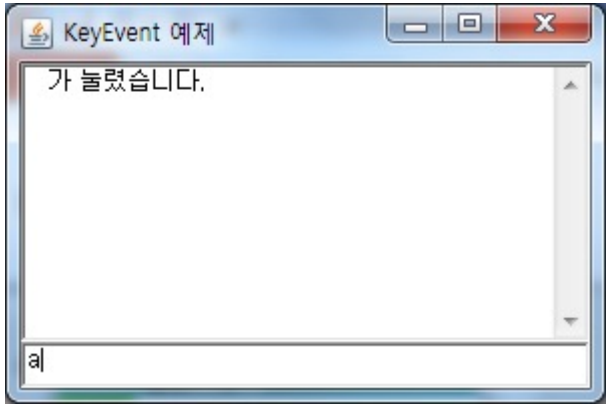
- 키보드 이벤트
  - ✓ 키보드를 통해서 키 입력이 오면 발생하는 이벤트
  - ✓ KeyListener 인터페이스를 이용해서 처리
  - ✓ KeyListener 이벤트 처리 메서드
    - ❑ keyPressed(KeyEvent e): Component에서 키가 눌러지면 호출
    - ❑ keyReleased(KeyEvent e): 키에서 뗄 때 호출
    - ❑ keyTyped(KeyEvent e): 키보드를 통해 문자가 입력되었을 때 호출
  - ✓ KeyEvent 의 멤버

필드명	해당 키
VK_0 ~ VK_9	숫자 0 ~ 9
VK_A ~ VK_Z	영문자 A ~ Z
VK_F1 ~ VK_F24	기능키 F1 ~ F24
VK_ENTER	Enter 키
VK_SPACE	Space 키
VK_BACKSPACE	BackSpace 키
KEY_PRESSED	키가 눌려진 이벤트 의미
KEY_RELEASED	키가 눌렸다 놓아진 이벤트 의미
KEY_TYPED	키 입력 이벤트 의미

메소드	내용
getKeyChar()	이벤트에 의해 입력된 문자값을 가져온다.
getKeyCode()	이벤트에 의해 입력된 문자에 해당하는 코드값을 가져온다.

KeyEvent

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

class KeyEventTest extends Frame implements KeyListener {
        TextArea txt_info;
        TextField txt;

        public KeyEventTest(String str) {
                super(str);
                txt = new TextField();
                txt.addKeyListener(this);
                add("Center", txt_info = new TextArea());
                add("South", txt);
                setSize(300, 200);
                setVisible(true);
        }
```



KeyEvent

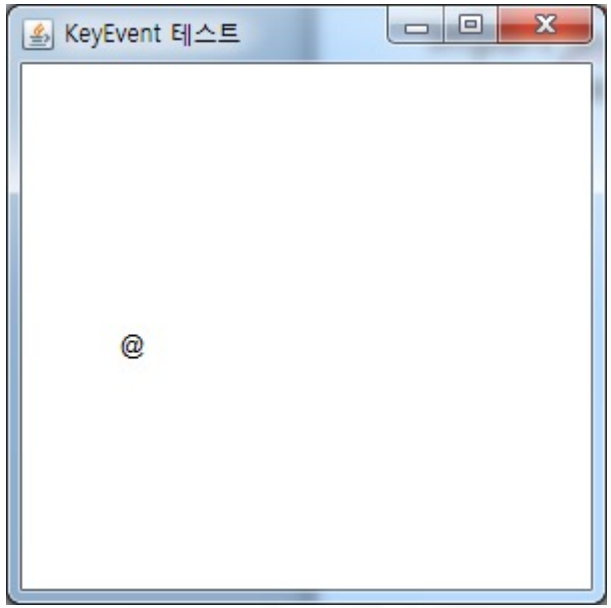
```
        public void keyTyped(KeyEvent e) {
                txt_info.setText(e.getKeyChar() + "가 입력되었습니다.₩n");
        }

        public void keyPressed(KeyEvent e) {
                txt_info.setText(e.getKeyChar() + "가 눌렸습니다.₩n");
        }

        public void keyReleased(KeyEvent e) {
                txt_info.setText(e.getKeyChar() + "를 놓았습니다.₩n");
        }
}
```

KeyEvent

```
public class KeyEventMain {
	public static void main(String[] args) {
		new KeyEventTest("KeyEvent 예제");
	}

}
```

KeyMove

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

class KeyMove extends Frame implements KeyListener {
        Label label = new Label("@");
        int x = 100;
        int y = 100;

        public KeyMove() {
                super("KeyEvent 테스트");
                setLayout(null);
                label.setBounds(x, y, 20, 20);
                add(label);
                setBounds(300, 300, 300, 300);
                setVisible(true);
                addKeyListener(this);
        }
```



KeyMove

```
public void keyReleased(KeyEvent e) {
}

public void keyTyped(KeyEvent e) {
}

public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_DOWN) {
                y = y + 5;
                label.setBounds(x, y, 20, 20);
        }
        if (e.getKeyCode() == KeyEvent.VK_UP) {
                y = y - 5;
                label.setBounds(x, y, 20, 20);
        }
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
                x = x - 5;
                label.setBounds(x, y, 20, 20);
        }
        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
                x = x + 5;
                label.setBounds(x, y, 20, 20);
        }
}
}
```

KeyMove

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

}
```

Mouse Event

- MouseEvent(JList를 제외한 거의 모든 Component)
  - ✓ 마우스 버튼을 누르거나 특정 Component 안에 진입하거나 벗어날 때 호출되는 이벤트
  - ✓ MouseListener 인터페이스를 이용해서 처리
  - ✓ MouseListener 인터페이스의 메서드
    - ❑ mouseClicked(MouseEvent e): 마우스를 클릭했을 때 호출
    - ❑ mouseEntered(MouseEvent e): 마우스 커서가 Component 영역에 들어오면 호출
    - ❑ mouseExited(MouseEvent e): 마우스 커서가 Component 영역에서 벗어나면 호출
    - ❑ mousePressed(MouseEvent e): 마우스 버튼이 눌러지면 호출
    - ❑ mouseReleased(MouseEvent e): 마우스 버튼이 눌러졌다 띄어지면 호출
  - ✓ MouseEvent 클래스의 멤버

필드명	설명
MOUSE_CLICKED	마우스 버튼이 클릭된 경우 발생되는 이벤트
MOUSE_ENTERED	마우스 커서가 컴포넌트 영역으로 들어왔을 때 발생되는 이벤트
MOUSE_EXITED	마우스 커서가 컴포넌트 영역 밖으로 나가면 발생되는 이벤트
MOUSE_PRESSED	마우스 버튼이 눌려졌을 때 발생하는 이벤트
MOUSE_RELEASED	마우스 버튼이 눌렸다 띄어졌을 때 발생하는 이벤트
MOUSE_DRAGGED	마우스 버튼이 클릭된 상태에서 움직일 때 발생되는 이벤트
MOUSE_MOVED	마우스 커서가 움직일 때 발생되는 이벤트

메소드	내용
getClickCount()	마우스가 눌려진 횟수를 얻어온다
getPoint()	마우스 이벤트가 발생한 좌표를 얻어온다
getX()	마우스 이벤트가 발생한 X좌표를 얻어온다.
getY()	마우스 이벤트가 발생한 Y좌표를 얻어온다.

Mouse Event

- MouseMotionEvent(JList를 제외한 거의 모든 Component)
  - ✓ 마우스를 움직이면 호출되는 이벤트
  - ✓ MouseMotionListener 인터페이스를 이용해서 처리
  - ✓ 메서드
    - ❑ mouseDragged(MouseEvent e): 마우스 버튼이 눌러진 상태로 이동하면 호출
    - ❑ mouseMoved(MouseEvent e): 마우스를 이동하면 호출

Mouse Event

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

class MoustEventView extends Frame implements MouseListener {
        public MoustEventView(String str) {
                super(str);
                addMouseListener(this);
                setSize(300, 300);
                setVisible(true);
        }

        public void mouseClicked(MouseEvent e) {
                System.out.println("클릭");
        }

        public void mousePressed(MouseEvent e) {
                System.out.println("누름");
        }

        public void mouseReleased(MouseEvent e) {
                System.out.println("마우스업");
        }
```

Mouse Event

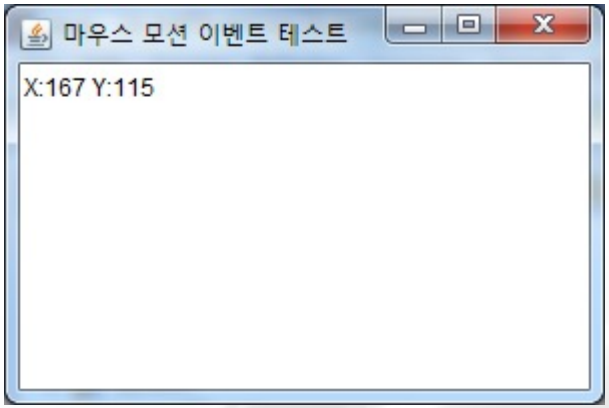
```
	public void mouseEntered(MouseEvent e) {
		System.out.println("마우스가 윈도우 영역 안으로 들어옴");
	}

	public void mouseExited(MouseEvent e) {
		System.out.println("마우스가 윈도우 영역을 벗어남");
	}
}
```

Mouse Event

```
public class MouseEventMain {
	public static void main(String[] args) {
			new MoustEventView("마우스 이벤트 테스트");
	}
}
```

Mouse Motion Event

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

class MouseMotionEventTest extends Frame implements MouseMotionListener {
        Label lbl;

        public MouseMotionEventTest(String str) {
                super(str);
                lbl = new Label();
                add("North", lbl);
                addMouseMotionListener(this);
                setSize(300, 200);
                setVisible(true);
        }

        public void mouseDragged(MouseEvent e) {
                lbl.setText("마우스 드래그");
        }

        public void mouseMoved(MouseEvent e) {
                lbl.setText("X:" + e.getX() + " Y:" + e.getY());
        }
}
```



Mouse Motion Event

```
public class MouseMotionMain {
		public static void main(String[] args) {
				new MouseMotionEventTest("마우스 모션 이벤트 테스트");
		}

}
```

Window Event

- WindowEvent(JFrame)
  - ✓ 윈도우를 활성화, 아이콘화, 비활성화 작업 시 발생하는 이벤트
  - ✓ WindowListener 인터페이스를 이용해서 처리
  - ✓ 메서드
    - windowOpened(WindowEvent e): 윈도우가 열릴 때
    - windowClosing(WindowEvent e): 윈도우가 닫힐 때
    - windowClosed(WindowEvent e): 시스템 닫힌 후
    - windowActivated(WindowEvent e): 윈도우가 활성화 되었을 때
    - windowDeactivated(WindowEvent e): 윈도우가 비 활성화 되었을 때
    - windowIconified(WindowEvent e): 윈도우가 아이콘화 되었을 때
    - windowDeiconfied(WindowEvent e): 윈도우가 최소화 상태에서 원래대로 되돌아 올 때

필드명	설명
WINDOW_ACTIVATED	윈도우가 활성화될 때 발생
WINDOW_DEACTIVATED	윈도우가 비활성화될 때 발생
WINDOW_CLOSED	윈도우가 닫힐 때 발생
WINDOW_CLOSING	윈도우가 사용자의 요청으로 닫힐 때 발생
WINDOW_ICONIFIED	윈도우가 아이콘화될 때 발생
WINDOW_OPENED	윈도우가 생성될 때 발생

메소드	내용
getWindow()	이벤트가 발생된 윈도우를 가져온다

Window Event

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

class WindowEventView extends Frame implements WindowListener {
	public WindowEventView(String str) {
			super(str);
			addWindowListener(this);
			setSize(300, 200);
			setVisible(true);
	}
	public void windowOpened(WindowEvent e) {
			System.out.println("윈도우 열기");
	}
	public void windowClosing(WindowEvent e) {
			System.out.println("윈도우 닫기");
			System.exit(0);
	}
	public void windowClosed(WindowEvent e) {
			System.out.println("윈도우 최소화");
	}

	public void windowActivated(WindowEvent e) {
			System.out.println("윈도우 활성화");
	}
```

Window Event

```
        public void windowDeactivated(WindowEvent e) {
                System.out.println("윈도우 비활성화");
        }
        public void windowIconified(WindowEvent e) {
                System.out.println("윈도우 아이콘화");
        }
        public void windowDeiconified(WindowEvent e) {
                System.out.println("윈도우 아이콘 해제");
        }
}
```

Window Event

```
public class WindowEventMain {
	public static void main(String[] args) {
			new WindowEventView("윈도우 이벤트");
	}

}
```

Adapter Class

- 이벤트 리스너를 이용한 방법은 인터페이스를 이용하기 때문에 리스너에 선언된 모든 메서드를 전부 구현을 해야하는 번거러움이 있음
- Adapter 클래스는 이벤트 리스너 인터페이스 중에서 추상 메서드가 두 개 이상 존재하는 인터페이스를 구현한 추상 클래스이며 인터페이스에 있는 모든 메서드를 비어있는 상태로 Overriding 한 상태이기 때문에 처리하고자 하는 이벤트 처리 메서드만 재정의 해서 이벤트를 처리

이벤트	이벤트 리스너	이벤트 어댑터
ComponentEvent	ComponentListener	ComponentAdapter
ContainerEvent	ContainerListener	ContainerAdapter
FocusEvent	FocusListener	FocusAdapter
KeyEvent	KeyListener	KeyAdapter
MouseEvent	MouseListener	MouseAdapter
MouseMotionEvent	MouseMotionListener	MouseMotionAdapter
WindowEvent	WindowListener	WindowAdapter

Adapter Class

```
import java.awt.*;
import java.awt.event.*;
class AdapterView extends Frame {
	Button btn1;
	Button btn2;
	public TextField text;
	public AdapterView() {
			super("Adapter를 이용한 이벤트 처리");
			setLayout(null);
			text = new TextField();
			btn1 = new Button("버튼1");
			btn2 = new Button("버튼2");
			text.setBounds(100, 100, 60, 20);
			btn1.setBounds(110, 130, 40, 20);
			btn2.setBounds(150, 130, 40, 20);
			add(btn1);
			add(btn2);
			add(text);
			btn1.addFocusListener(new FocusHandler(text));
			btn2.addFocusListener(new FocusHandler(text));
			addWindowListener(new WindoEventHandler());
			setBounds(100, 100, 300, 300);
			setVisible(true);
	}
}
```

Adapter Class

```
class FocusHandler extends FocusAdapter {
	TextField text;

	public FocusHandler(TextField tf) {
		text = tf;
	}

	public void focusGained(FocusEvent e) {
		Button obj = (Button) e.getSource();
		text.setText(obj.getLabel() + "포커스를 얻음");
	}
}

class WindoEventHandler extends WindowAdapter {
	public void windowClosing(WindowEvent e) {
		System.exit(0);
	}
}
```

Adapter Class

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

}
```

Swing

Component, 컨테이너 레이아웃 매니저 그리고 이벤트를 기반으로 하는 시스템 독립적인 GUI 프로그래밍을 위한 API

SWING

- JFrame
  - ✓ 클래스 계층

    java.lang.Object

    java.awt.Component

    java.awt.Container

    java.awt.Window

    java.awt.Frame

    javax.swing.Jframe
  - ✓ 구현된 인터페이스: ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants
  - ✓ 생성자
    - ❑ JFrame (): 타이틀이 없는 프레임을 생성
    - ❑ JFrame (String title): 타이틀이 있는 프레임을 생성

SWING

- JFrame
  - ✓ 구성
    - JRootPane: 실질적인 윈도우 기능을 수행하는 경량의 컨테이너로 JFrame의 getRootPane() 메서드로부터 얻어낼 수 있음
      - JRootPane은 glassPane과 layeredPane으로 구성되어 있고 layeredPane은 JMenuBar와 contentPane을 포함
    - layeredPane: 여러 층의 패널을 포함 할 수 있는 패널로 윗 부분은 menuBar와 아래 부분은 contentPane으로 구성
    - glassPane: 기본적으로 숨겨진 상태로 되어 있으며 다른 패널 위에 존재하는 패널
      - 마우스 이벤트를 처리하기 위해 가장 먼저 루트 페인에 추가
    - contentPane: 일반적인 Component들을 가질 수 있는 패널
      - 프레임 인스턴스의 JRootPane으로 부터 getContentPane()메서드를 이용해서 얻을 수 있음
    - menuBar: 윈도우의 메뉴를 제공하는 역할로 생략이 가능한 선택항목
      - JMenu, JmenuItem 등을 이용해서 메뉴를 구성하여 setJMenuBar()메서드를 이용해서 등록

SWING

- JFrame
  - ✓ 메서드
    - MenuBar getMenuBar(): 프레임의 메뉴바를 리턴
    - int getState(): 프레임의 상태를 리턴
    - String getTitle(): 타이틀 바의 문자열을 리턴
    - void remove(MenuComponent m): 프레임에서 지정한 메뉴를 제거
    - void setMenuBar(MenuBar mb): 프레임의 메뉴바를 지정
    - void setResizable(Boolean resizeable): 프레임의 크기를 사용자가 변경할 수 있게 할 것인지를 지정
    - void setSize (int width, int height): 사이즈를 width 및 height 로 변경
    - pack(): 컨테이너의 크기를 구성 요소들의 크기로 설정
    - void setVisible (boolean b): 파라미터 b 값에 의해 이 Component를 표시하는지 또는 비 표시로 설정
    - void setLocation (int x, int y) 이 Component를 새로운 위치로 이동
    - void setBounds (int x, int y, int width, int height)
    - getContentPane(): 프레임의 contentPane 오브젝트를 돌려줍니다.
    - setIconImage(Image image): 프레임의 최소화된 아이콘에 표시되는 이미지를 설정
    - add(Component): Component를 부착

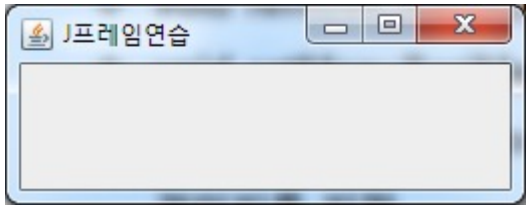
SWING

- JFrame
  - ✓ 기본적으로 타이틀, 최소버튼, 최대버튼, 종료버튼을 지원
  - ✓ 기본으로 화면에 안 보이도록 설정되어 있기 때문에 반드시 setVisible(true)값으로 설정해야 화면상에 출력
  - ✓ 만드는 방법은 JFrame 인스턴스를 직접 생성할 수 있고 JFrame으로부터 상속받은 클래스를 작성 한 후 인스턴스를 생성 할 수 있음
  - ✓ 종료 이벤트 처리는 setDefaultCloseOperation(EXIT_ON_CLOSE)을 호출해서 처리
  - ✓ 화면 크기 구하기

```
Tookit t = Tookit.getDefaultTookit();
Dimension screenSize = t.getScreenSize();
```

SWING

```
import javax.swing.*;

class JFrameView extends JFrame {
		public JFrameView() {
				super("J프레임");
				setSize(260, 100);
				setVisible(true);
		}
}
```



SWING

```
public class JFrameMain {
		public static void main(String[] args)
		{
			JFrameView  f = new JFrameView();
			f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		}
}
```

SWING

- JPanel
  - ✓ Component들을 그룹 별로 묶어서 처리할 때 사용하는 컨테이너
  - ✓ 일반적으로 JFrame에 Component들을 직접 붙이지 않고 JPanel에 그룹별로 붙이고 다시 Panel을 Frame에 붙이는 경우가 많음
  - ✓ 생성자
    - ❑ JPanel(): 디폴트 레이아웃 매니저를 사용해 새로운 Panel을 생성
    - ❑ JPanel(LayoutManager layout): 지정된 레이아웃 매니저를 이용하여 새로운 Panel을 생성
    - ❑ JPanel(boolean isDoubleBuffered)

SWING

- JScrollPane
  - ✓ 부모의 화면 크기보다 더 큰 내부 Component 들을 보여주는 Component
  - ✓ 포함된 Component의 원래 크기를 유지한 상태에서 화면을 스크롤 할 수 있도록 해주는 Component
  - ✓ 스크롤이 필요한 Component(JList, JTextArea, JTextPane 등)는 JScrollPane에 포함시켜서 출력해야 하는데 그렇게 하지 않으면 처음 설정한 크기 만큼만 출력되고 부모 화면 크기보다 큰 영역은 출력할 수 없음
  - ✓ 포함된 Component의 크기가 뷰 영역보다 큰 경우 자동으로 스크롤 바를 생성
  - ✓ JScrollPane에는 add라는 메서드를 이용해서 1개의 Component만 배치가 가능

SWING

❖ JScrollPane

자료형	필드명	설명
static int	HORIZONTAL_SCROLLBAR_ALWAYS	수평 스크롤바를 항상 보여주는 정책이다.
	VERTICAL_SCROLLBAR_ALWAYS	수직 스크롤바를 항상 보여주는 정책이다.
static int	HORIZONTAL_SCROLLBAR_AS_NEEDED	수평 스크롤바를 필요할 때에만 보여주는 정책이다.
	VERTICAL _SCROLLBAR_AS_NEEDED	수직 스크롤바를 필요할 때에만 보여주는 정책이다.
	HORIZONTAL_SCROLLBAR_NEVER	수평 스크롤바를 사용하지 않는 정책이다.
	VERTICAL _SCROLLBAR_NEVER	수직 스크롤바를 사용하지 않는 정책이다.

생성자	설명
JScrollPane( )	비어있는 새로운 JScrollPane 객체를 생성한다. 기본적으로 필요할 때 수평, 수직 스크롤바가 나타난다.
JScrollPane(Compoent view)	지정한 뷰 객체를 보여주는 JScrollPane 객체를 생성한다. 뷰보다 컴포넌트의 내용이 크면 수평, 수직 스크롤바가 나타난다.
JScrollPane(Compoent view, int vsbPolicy, int hsbPolicy)	지정한 뷰 객체를 보여주는 JScrollPane 객체를 생성한다. 지정한 수직, 수평 스크롤바 표시 정색(책)에 따라 스크롤바가 보여진다.
JScrollPane(int vsbPolicy, int hsbPolicy)	지정한 수직, 수평 스크롤바 표시 정책에 따라 JScrollPane 객체를 생성한다.

SWING

- component
  - ✓ 화면을 구성하는 작은 부품
  - ✓ 컨테이너의 add(컨테이너 또는 Component 인스턴스)를 통해 부착
  - ✓ 메서드
    - ❑ Color getBackground(): Component의 배경 색상을 리턴
    - ❑ Color getForeground(): Component의 전경 색상을 리턴
    - ❑ int getHeight(): Component의 높이를 리턴
    - ❑ Point getLocation(): Component의 위치를 리턴
    - ❑ String getName(): Component의 이름을 리턴
    - ❑ Container getParent(): Component의 부모를 리턴
    - ❑ Dimension getSize(): Component의 사이즈를 오브젝트로 리턴
    - ❑ int getWidth(): Component의 현재의 폭을 리턴
    - ❑ int getX (), getY ()
    - ❑ void setBackground (Color c): Component의 배경색을 설정
    - ❑ void setBounds (int x, int y, int width, int height): Component를 이동해 사이즈 변경
    - ❑ void setEnabled (boolean b): 사용 가능 여부 설정
    - ❑ void setFocusable (boolean focusable): Component의 포커스 상태 설정
    - ❑ void setVisible (boolean visible): Component의 출력 여부설정

SWING

- Component
  - ✓ 메서드
    - ❑ void setForeground (Color c): 전경색 설정
    - ❑ void setLocation (int x, int y), setLocation (Point p)
    - ❑ void setMaximumSize (Dimension maximumSize): 최대 사이즈 설정
    - ❑ void setMinimumSize (Dimension minimumSize): 최소 사이즈 설정
    - ❑ void setName (String name): 이름 설정
    - ❑ void setSize (Dimension d), setSize (int width, int height)

SWING

- Border
  - ✓ 경계선으로 AbstractBorder가 최상위 클래스
  - ✓ AbstractBorder의 하위 클래스: BevelBorder, CompoundBorder, EmptyBorder, EtchedBorder, LineBorder, MatteBorder, SoftBevelBorder, TitledBorder(문자열을 설정할 수 있는 보더)
  - ✓ Border 인스턴스를 생성한 후 Component의 setBorder()를 이용해서 설정
- setTooltipText(일반 텍스트 또는 html코드)를 이용해서 툴 팁 작성 가능

SWING

❖ JLabel

✓ JLabel: Component에 텍스트와 이미지 설정해서 출력하는 컴포넌트

✓ getText와 setText(String str)을 이용해서 텍스트 설정

생성자	설명
JLabel(Icon image)	지정한 아이콘을 보여주는 레이블 객체를 생성한다.
JLabel(Icon image, int horizontalAlignment)	지정한 아이콘을 지정한 수평정렬 방식에 따라 보여주는 레이블 객체를 생성한다.
JLabel(String text, int horizontalAlignment)	지정한 문자열을 지정한 수평정렬 방식에 따라 보여주는 레이블 객체를 생성한다.
JLabel(String text, Icon icon, int horizontalAlignment)	지정한 텍스트와 지정한 아이콘을 지정한 수평정렬 방식에 따라 보여주는 레이블 객체를 생성한다.

반환형	메서드	설명
void	setHorizontalAlignment(int alignment)	수평 정렬 상태를 정한다.
	setVerticalAlignment(int alignment)	수직 정렬 상태를 정한다.
	setHorizontalTextPostion(int textPosition)	아이콘에 대한 텍스트의 수평 위치를 정한다.
	setVerticalTextPostion(int textPosition)	아이콘에 대한 텍스트의 수직 위치를 정한다.
	setIconTextGap(int iconTextGap)	아이콘과 텍스트 사이의 간격을 지정한다.

SWING

- AbstractButton
  - ✓ JButton, JToggleButton, JRadioButton 등의 모든 버튼의 종류를 추상화한 클래스로 버튼 클래스들은 이 클래스로부터 상속을 받음
  - ✓ 버튼들의 모양이나 기능은 다르지만 공통적인 기능들은 이 클래스에 구현

반환형	메서드	설명
void	setModal(ButtonModel newModel)	버튼의 모델을 정한다.
	setMargin(Insets m)	버튼의 경계선과 텍스트와의 경계를 정한다.
	setRolloverEnabled(boolean b)	롤오버 효과의 활성화 상태를 정한다.
	setIcon(Icon defaultIcon)	버튼의 기본 아이콘을 정한다.
	setSelectedIcon(Icon selectedIcon)	버튼이 선택된 상태의 아이콘을 정한다.
	setDisableIcon(Icon disabledIcon)	버튼이 비활성화된 상태의 아이콘을 정한다.
	setDisableSelectedIcon(Icon disabledSelectedIcon)	버튼이 비활성화되고 선택된 상태의 아이콘을 정한다.
	setPressedIcon(Icon pressedIcon)	버튼이 눌려진 상태의 아이콘을 정한다.
	setRolloverIcon(Icon rolloverIcon)	마우스가 버튼 위에 놓여진 상태의 아이콘을 정한다.
void	setHorizontalTextPosition(int textPostion)	버튼의 수평 텍스트의 위치를 지정한다.
	setVerticalTextPosition(int textPostion)	버튼의 수직 텍스트의 위치를 지정한다.

SWING

- JButton: 텍스트와 이미지를 모두 넣을 수 있고 다양한 상태에서 보여줄 버튼의 이미지를 각각 지정하여 Roll-Over 버튼을 만들 수도 있음

생성자	설명
JButton(Icon icon)	지정한 아이콘을 보여주는 버튼 객체를 생성한다.
JButton(String text, Icon icon)	지정한 텍스트와 아이콘을 보여주는 버튼 객체를 생성한다.

SWING

- JToggleButton
  - ✓ 버튼의 상태로 기본과 선택된 상태 두 가지를 가지는 버튼
  - ✓ 기본 상태에서 한 번 클릭하면 선택상태로 유지하고 있다가 다시 선택하면 기본 상태로 변경되는 버튼
  - ✓ 일반적으로 기본 상태와 선택된 상태를 구분하기 위해서 서로 다른 아이콘을 지정하여 사용

생성자	설명
JToggleButton(Icon icon)	지정한 아이콘을 보여주는 토글 버튼 객체를 생성한다.
JToggleButton(Icon icon, boolean selected)	지정한 아이콘을 보여주며 버튼의 선택 여부를 지정하여 토글 버튼 객체를 생성한다.
JToggleButton(String text)	지정한 텍스트를 보여주는 토글 버튼 객체를 생성한다.
JToggleButton(String text, boolean selected)	지정한 텍스트를 보여주며 버튼의 선택 여부를 지정하여 토글 버튼 객체를 생성한다.

SWING

❖ JRadioButton

- ✓ AWT의 Checkbox 클래스를 이용한 라디오 형 체크박스와 유사한 형태의 Component
- ✓ 여러 항목 중에서 하나의 항목만 선택할 수 있도록 만든 Component로 여러 개의 항목을 그룹으로 묶어주기 위하여 ButtonGroup 클래스를 이용해서 그룹으로 사용

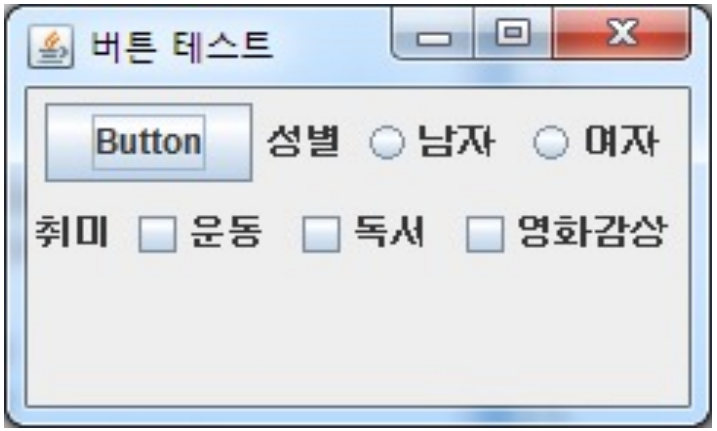
생성자	설명
JRadioButton(Icon icon)	지정한 아이콘을 보여주는 라디오 버튼 객체를 생성한다.
JRadioButton(Icon icon, boolean selected)	지정한 아이콘을 보여주며 버튼의 선택 여부를 지정하여 라디오 버튼 객체를 생성한다.
JRadioButton(String text)	지정한 텍스트를 보여주는 라디오 버튼 객체를 생성한다.
JRadioButton(String text, boolean selected)	지정한 텍스트를 보여주며 버튼의 선택 여부를 지정하여 라디오 버튼 객체를 생성한다.

SWING

❖ JCheckBox

생성자	설명
JCheckBox(Icon icon)	지정한 아이콘을 보여주는 체크박스 객체를 생성한다.
JCheckBox(Icon icon, boolean selected)	지정한 아이콘을 보여주며 버튼의 선택 여부를 지정하여 체크박스 객체를 생성한다.
JCheckBox(String text)	지정한 텍스트를 보여주는 체크박스 객체를 생성한다.
JCheckBox(String text, boolean selected)	지정한 텍스트를 보여주며 버튼의 선택 여부를 지정하여 체크박스 객체를 생성한다.

JButtonTest.java




```
import javax.swing.*;
public class JButtonTest extends JFrame
{
        public JButtonTest()
        {
                super("버튼 테스트");
                JPanel buttonPanel = new JPanel();
                JButton btn = new JButton("Button");
                buttonPanel.add(btn);

                                        JLabel lblMale = new JLabel("성별");
                JRadioButton rb1 = new JRadioButton("남자");
                JRadioButton rb2 = new JRadioButton("여자");
                ButtonGroup bg = new ButtonGroup();
                bg.add(rb1);
                bg.add(rb2);
                buttonPanel.add(lblMale);
                buttonPanel.add(rb1);
                buttonPanel.add(rb2);

                add(buttonPanel);
```

JButtonTest.java

```
                JLabel lblHobby = new JLabel("취미");
                JCheckBox check1 = new JCheckBox("운동");
                JCheckBox check2 = new JCheckBox("독서");
                JCheckBox check3 = new JCheckBox("영화감상");
                buttonPanel.add(lblHobby);
                buttonPanel.add(check1);
                buttonPanel.add(check2);
                buttonPanel.add(check3);

                setLocation(150,200);
                setSize(242,150);
                setVisible(true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
}
```

SwingMain.java

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

SWING

- JTextField
  - ✓ 생성자
    - JTextField()
    - JTextField (int columns): 지정된 열의 수를 갖는 텍스트 필드 생성
    - JTextField (String text): 지정된 텍스트로 초기화 한 텍스트 필드
    - JTextField (String text, int columns)
  - ✓ 메서드
    - int getCaretPosition(): 텍스트 내에서 현재 마우스 포인터의 위치를 리턴
    - String getText(): 텍스트 내의 표시되는 텍스트를 리턴
    - void select(int selectionStart, int selectionEnd): 시작위치부터 마지막 위치까지 선택
    - void selectAll (): 텍스트 Component 내의 모든 텍스트를 선택
    - void setBackground (Color c): 백그라운드 칼라를 설정
    - void setCaretPosition (int position): caret의 위치를 설정
    - void setText (String t): 표시되는 텍스트를 지정된 텍스트로 설정

SWING

❖ JPasswordField

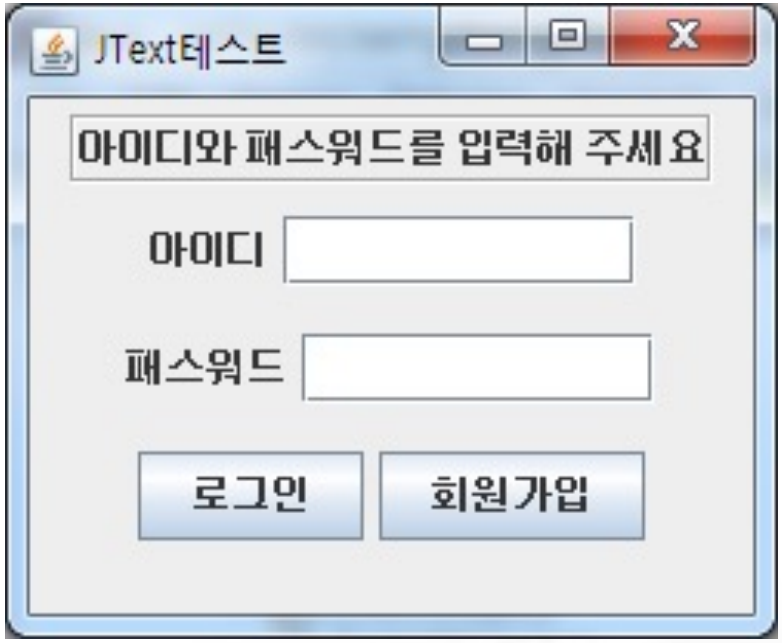
반환형	메서드	설명
char[ ]	getPassword( )	현재 입력된 암호를 얻어온다.
char	getEchoChar( )	사용자가 입력한 문자를 반향 문자로 화면에 나타낸다.
void	setEchoChar(char c)	지정한 문자를 반향 문자로 지정한다.

SWING

- JTextArea
  - ✓ 여러 줄을 입력할 수 있는 Component로 자체적으로 스크롤이 처리되지 않으므로 JscrollPane에 포함시켜 스크롤을 지원합니다.
  - ✓ 생성자
    - ❑ JTextArea ()
    - ❑ JTextArea (int rows, int columns)
    - ❑ JTextArea (String text)
    - ❑ JTextArea (String text, int rows, int columns)
  - ✓ 한 줄의 폭이 텍스트 영역의 폭보다 클 때 자동으로 줄 바꿈을 해주지 않으므로 옵션을 통하여 설정을 해야 합니다.
  - ✓ 줄 바꿈을 할 때는 보통 문자 단위로 하지만 setWrapStyleWord() 메서드를 해서 줄 바꿈 옵션을 수정 가능

반환형	메서드	설명
void	setLineWrap(boolean wrap)	줄바꿈 정책을 사용할지를 지정한다.
	setWrapStyleWord(Boolean word)	단어 단위로 줄바꿈 정책을 사용할지를 지정한다.
	setTabSize(int size)	탭의 크기를 지정한다.

JTextTest.java




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

public class JTextTest extends JFrame
{
   JLabel lbl,la1,la2;
   JTextField id;
   JPasswordField passwd;
   JPanel idPanel,paPanel,loginPanel;
   JButton b1,b2;

   public JTextTest()
   {
      super( "JText테스트" );
      setLayout( new FlowLayout() );
      EtchedBorder eborder =  new EtchedBorder();

      lbl = new JLabel( "아이디와 패스워드를 입력해 주세요" );
      lbl.setBorder(eborder);
      add( lbl );
```

JTextTest.java

```
        idPanel = new JPanel();
        la1 = new JLabel("아이디");
        id = new JTextField(10);
        idPanel.add( la1 );
        idPanel.add( id );
        paPanel = new JPanel();
        la2 = new JLabel("패스워드");
        passwd = new JPasswordField(10);
        paPanel.add( la2 );
        paPanel.add( passwd );
        loginPanel = new JPanel();
        b1 = new JButton("로그인");
        b2 = new JButton("회원가입");
        loginPanel.add( b1 );
        loginPanel.add( b2 );
        add(idPanel);
        add(paPanel);
        add(loginPanel); setSize(250, 200);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

SwingMain.java

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

❖ JComboBox

생성자	설명
JComboBox(ComboBoxModel aModel)	지정한 콤보박스 모델을 사용하는 콤보박스 객체를 생성한다.
JComboBox(Object[ ] items)	지정한 배열의 자료를 보여주는 콤보박스 객체를 생성한다.
JComboBox(Vector items)	지정한 벡터의 자료를 보여주는 콤보박스 객체를 생성한다.

반환형	메서드	설명
void	addActionListener(ActionListener l)	콤보박스에서 아이템을 선택하였을 때 발생하는 이벤트를 받기 위해 지정된 ActionListener를 추가한다.
	addItemListener(ItemListener aListener)	콤보박스에서 아이템의 선택이 바뀠을 때 발생하는 이벤트를 받기 위해 지정된 ItemListener를 추가한다.
	showPopup( )	팝업 윈도우를 보이게 한다.
void	hidePopup( )	팝업 윈도우를 숨긴다.
Object	getItemAt(int index)	지정한 인덱스의 아이템을 얻어온다.
void	addItem(Object anObject)	지정한 아이템을 목록에 추가한다.
	insertItemAt(Object anObject, int index)	지정한 아이템을 지정한 인덱스 위치에 추가한다.
	removeAllItems( )	모든 아이템을 제거한다.
	removeItem(Object anObject)	지정된 아이템을 목록에서 제거한다.
	removeItemAt(int anIndex)	지정된 인덱스의 아이템을 목록에서 제거한다.
	setEditable(Boolean aFlag)	콤보박스의 편집 가능 상태를 지정한다.

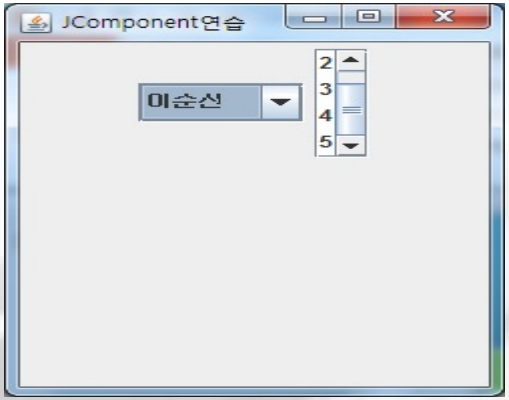
❖ JList

생성자	설명
JList(ListModel dataModel)	지정한 리스트 모델을 사용하는 리스트 객체를 생성한다.
JList(Object[ ] listData)	지정한 배열의 자료를 보여주는 리스트 객체를 생성한다.
JList(Vector listData)	지정한 벡터의 자료를 보여주는 리스트 객체를 생성한다.

반환형	메서드	설명
void	addListSelectionListener (ListSelectionListener listener)	리스트에서 선택하는 항목이 바뀌었을 때 발생하는 이벤트를 받기 위해 지정된 ListSelectionListener를 추가한다.
	ensureIndexIsVisible(int index)	지정한 인덱스의 아이템이 보여줄 수 있도록 리스트를 스크롤시킨다.
	setSelectionBackground(Color selectionBackground)	선택된 아이템의 배경색을 지정한다.
	setSelectionForeground(Color selectionForeground)	선택된 아이템의 전경색을 지정한다.
	setSelectionModel(ListSelection Model selectionModel)	지정된 ListSelectionModel로 지정한다.
	setSelectionMode(int selectionMode)	아이템의 선택모드를 selectionMode로 지정한다. • SINGLE_SELCTION: 한번에 하나의 아이템만 선택 • SINGLE_INTERVAL_SELECTION: 연속된 아이템 선택 • MULTIPLE_INTERVAL_SELECTION: 불연속된 아이템 선택
boolean	isSelectionEmpty( )	선택된 아이템이 있는지 검사하여 있으면 true, 없으면 false으로 얻어온다.
int	getSelectedIndex( )	선택된 첫 번째 아이템의 인덱스를 얻어온다.
Int[ ]	getSelectedIndices( )	선택된 모든 아이템의 인덱스를 배열로 얻어온다.
Object	getSelectedValue( )	선택된 아이템을 얻어온다.
Object[ ]	getSelectedValues( )	선택된 모든 아이템을 배열로 얻어온다.
void	setSelectedIndex(int index)	지정된 인덱스의 아이템을 선택된 상태로 만든다.
void	setSelectedValue(Object anObject, boolean shouldScroll)	지정된 객체를 가진 아이템을 선택된 상태로 만들고 shouldScroll이 true면 그 아이템이 보이도록 스크롤한다.
	setModel(ListModel model)	지정된 모델로 리스트의 모델을 지정한다.
	setListData(Object[ ] listData)	지정된 배열 자료를 사용하는 모델을 만들고 지정한다.
	setListData(Vector listData)	지정된 벡터 자료를 사용하는 모델을 만들고 지정한다.

ComboList.java

```
public class ComboList extends JFrame {
        JPanel p, p1, p2;
        JList<String> jl;
        JComboBox<String> com;
        String name[] = { "이순신", "김유신", "강감찬 ", "을지문덕", "세종대왕" };
        String count[] = { "1", "2", "3", "4", "5" };

        public ComboList() {
                super("JComponent연습");
                Container c = getContentPane();
                c.setLayout(new FlowLayout());
                p = new JPanel(new GridLayout(1, 2));

                com = new JComboBox<String>(name);
                JScrollPane s = new JScrollPane(com);
                c.add(s);
                jl = new JList<String>(count);
                JScrollPane s1 = new JScrollPane(jl);
                jl.setVisibleRowCount(4);
                c.add(s1);
                setSize(250, 300);
                setVisible(true);
        }
}
```



SwingMain.java

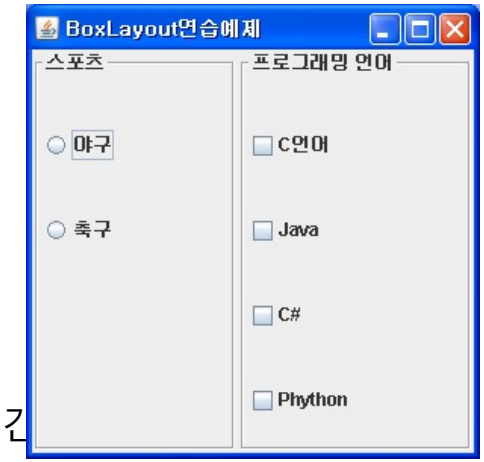
```
import javax.swing.*;

public class SwingMain {
		public static void main(String args[]) {
				ComboList obj = new ComboList();
				obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		}
}
```

SWING

- Swing에 추가된 Layout
  - ✓ Box: Component들을 왼쪽에서 오른쪽으로 수평 배치되거나 위에서 아래로 수직으로 배치하도록 해 주는 인스턴스
  - ✓ 생성
    - ❑ Box.createVerticalBox(): 수평으로 Component를 추가할 수 있는 레이아웃 생성
    - ❑ Box.createHorizontalBox(); 수직으로 Component 추가
    - ❑ Box.createVerticalStrut(int width): 빈 공간 확보
    - ❑ Box.createHorizontalStrut(int width): 빈 공간 확보

BoxLayoutTest.java

```
public class BoxLayoutTest extends JFrame {
        private static final long serialVersionUID = 1L;

        public BoxLayoutTest(){
                super( "BoxLayout연습예제" );
                JRadioButton r1,r2;

                //수직 배치 박스 레이아웃을 생성
                Box left = Box.createVerticalBox();
                //보기 좋게 배치하기 위래 투명 Component로 공간
                left.add(Box.createVerticalStrut(30));
                //라디오버튼을 위한 버튼 그룹 생성
                ButtonGroup radioGroup = new ButtonGroup();
                //버튼 인스턴스를 생성 후 버튼 그룹에 추가
                radioGroup.add(r1 = new JRadioButton("야구"));
                //버튼 인스턴스를 Box인스턴스 left에 추가
                left.add(r1);
                left.add(Box.createVerticalStrut(30));
                radioGroup.add(r2 = new JRadioButton("축구"));
                left.add(r2);
```



BoxLayoutTest.java

```
//Box인스턴스 left를 패널에 추가
JPanel leftPanel = new JPanel(new BorderLayout());
//패널의 테두리선을 에칭효과로 지정
leftPanel.setBorder(
                                new TitledBorder( new EtchedBorder(),"스포츠"));
leftPanel.add(left, BorderLayout.CENTER);

//수직으로 배치하는 박스레이아웃 생성
Box right = Box.createVerticalBox();
right.add(Box.createVerticalStrut(30));
//체크박스를 생성해 박스레이아웃에 추가
right.add(new JCheckBox("C언어"));
right.add(Box.createVerticalStrut(30));
right.add(new JCheckBox("Java"));
right.add(Box.createVerticalStrut(30));
right.add(new JCheckBox("C#"));
right.add(Box.createVerticalStrut(30));
right.add(new JCheckBox("Phython"));
```

BoxLayoutTest.java

```
                JPanel rightPanel = new JPanel(new BorderLayout());
                rightPanel.setBorder(
                                        new TitledBorder(new EtchedBorder(), "프로그래밍 언
어"));
                rightPanel.add(right, BorderLayout.CENTER);

                //수평으로 배치하는 박스레이아웃을 생성해
                //패널leftPanel과 rightPanel을 이 박스 레이아웃에 배치
                Box center = Box.createHorizontalBox();
                center.add(leftPanel);
                center.add(rightPanel);
                add(center,   BorderLayout.CENTER);

                setSize(300,300);
                setVisible(true);
        }
}
```

SwingMain.java

```
import javax.swing.*;

public class SwingMain {
		public static void main(String[] args) {
				BoxLayoutTest box = new BoxLayoutTest();
				box.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		}
}
```

SWING

- JTable
  - ✓ 데이터를 행과 열로 구성되어 있는 테이블 형식으로 보여주는 Component
  - ✓ JTable 역시 Scrollable 인터페이스가 구현되어 있어 JScrollPane에 붙여 스크롤을 할 수 있도록 생성할 수 있으며 데이터를 출력하기 위한 모델로는 TableModel 클래스를 사용
  - ✓ JTable을 사용하기 위해서는 먼저 데이터를 저장할 모델을 만들고 뷰 인 JTable에 연결해서 화면에 출력
  - ✓ 생성자

생성자	설명
JTable( )	기본 자료 모델, 기본 컬럼 모델, 기본 선택 모델로 초기화된 기본 테이블 객체를 생성한다.
JTable(int numRows, int numColumns)	DefaultTableModel을 이용해 빈 셀을 numRows의 행과 numColumns의 열의 수만큼 테이블 객체를 생성한다.
JTable(Object[ ] rowData, Object columnNames)	2차원 배열에 값들을 보여주는 테이블 객체를 생성한다. 주어진 테이블 테이터와 컬럼 이름을 가진다.
JTable(TableModel dm)	주어진 dm을 가지는 자료 모델, 기본 컬럼 모델, 기본 선택 모델로 초기화된 테이블 객체를 생성한다.
JTable(TableModel dm, TableColumnModel cm)	주어진 dm을 가지는 자료 모델, cm을 가지는 컬럼 모델, 기본 선택 모델로 초기화된 테이블 객체를 생성한다.
JTable(TableModel dm, TableColumnModel cm, ListSelectionModel sm)	주어진 dm을 가지는 자료 모델, cm을 가지는 컬럼 모델, sm을 가지는 선택 모델로 초기화된 테이블 객체를 생성한다.
JTable(Vector rowData, Vector columnNames)	주어진 Vectors의 Vector의 값들을 보여주는 테이블 객체를 생성한다. 주어진 테이블 자료와 컬럼 이름을 가진다.

❖ JTable

✓ 메서드

반환형	메서드	설명
void	setModel(TableModel dataModel)	주어진 자료 모델을 테이블의 자료 모델로 지정한다.
	setColumnModel (TableColumnModel columnModel)	주어진 컬럼 모델을 테이블의 컬럼 모델로 지정한다.
	setCellEditor(TableCellEditor anEditor)	주어진 셀 자료를 테이블의 셀 에디터로 지정한다.

반환형	메서드	설명
void	setValueAt(Object aValue, int row, int column)	데이블 모델의 셀에 주어진 row, column 위치에 자료를 설정한다.
Object	getValueAt(int row, int column)	주어진 row, column 위치에 있는 셀값을 얻어온다.

반환형	메서드	설명
void	setAutoResizeMode(int mode)	컬럼의 자동 크기 조절 모드를 설정한다.
	setIntercellSpacing (Dimension intercellSpacing)	셀 사이의 간격을 설정한다.
	setSelectionBackground (Color selectionBackground)	선택된 셀의 배경색을 설정한다.
	setSelectionForeground (Color selectionForeground)	선택된 셀의 전경색을 설정한다.
	setShowHorizontalLines (boolean showHorizontalLines)	셀 사이의 수평 구분선을 그릴지를 설정한다.
	setShowVerticalLines(boolean showVerticalLines)	셀 사이의 수직 구분선을 그릴지를 설정한다.

❖ JTable

✓ 메서드

반환형	메서드	설명
void	clearSelction( )	선택된 행과 열, 셀을 지운다.
	selectAll( )	전체 테이블의 셀, 행과 열을 선택한다.
	setCellSelectionEnabled (boolean cellSelectionEnabled)	행과 열을 동시에 선택할 수 있도록 하는 선택모드를 사용할지를 설정한다.
	setColumnSelectionAllowed (boolean columnSelectionAllowed)	테이블의 열을 동시에 선택할 수 있도록 하는 선택모드를 사용할지를 설정한다.
void	setRowSelectionAllowed (boolean rowSelectionAllowed)	테이블의 행을 동시에 선택할 수 있도록 하는 선택모드를 사용할지를 설정한다.
	setSelectionMode(int selectionMode)	테이블의 선택 모드를 단일 선택하거나, 연속적 단일선택, 복수 선택 모드로 설정한다.
int	getSelectedColumn( )	선택된 열의 첫번째 위치를 얻어온다. 선택된 열이 없으면 –1을 얻어온다.
Int[ ]	getSelectedColumns( )	선택된 열들을 배열 형태로 얻어온다.
int	getSelectedRow( )	선택된 행의 첫번째 위치를 얻어온다. 선택된 열이 없으면 –1을 얻어온다.
Int[ ]	getSelectedRows( )	선택된 행들을 배열 형태로 얻어온다.
boolean	isCellSelected(int row, int column)	주어진 위치의 셀이 선택 상태인지를 얻어온다.

반환형	메서드	설명
int	getColumnCount( )	열 수를 얻어온다.
	getRowCount( )	행 수를 얻어온다.
Object	getValueAt(int rowIndex, int columnIndex)	주어진 위치의 셀값을 얻어온다.
void	setValueAt(Object aValue, introwIndex, int columnIndex)	주어진 위치의 셀값을 설정한다.

- TableModel
  - ✓ JTable에 출력되는 데이터를 표현하는 것으로 Model에 변화가 생기면 JTable에 적용이 되서 출력
  - ✓ 생성자

DefaultTableModel () 열이 0, 행이 0 의 테이블인 디폴트의 DefaultTableModel 를 구축합니다.
DefaultTableModel (int rowCount, int columnCount) rowCount 와 columnCount 의 객체치가 null 인 DefaultTableModel 를 구축합니다.
DefaultTableModel (Object [][] data, Object [] columnNames) DefaultTableModel 를 구축해,setDataVector 메소드에 data 와 columnNames 를 건네주어 테이블을 초기화합니다.
DefaultTableModel (Object [] columnNames, int rowCount) columnNames 내의 요소수와 같은 수의 열을 가져,rowCount 의 객체치가 null 인 DefaultTableModel 를 구축합니다.
DefaultTableModel (Vector columnNames, int rowCount) columnNames 내의 요소수와 같은 수의 열을 가져,rowCount 의 객체치가 null 인 DefaultTableModel 를 구축합니다.
DefaultTableModel (Vector data, Vector columnNames) DefaultTableModel 를 구축해,setDataVector 메소드에 data 와 columnNames 를 건네주어 테이블을 초기화합니다.

SWING

- TableModel
  - ✓ 메서드
    - addColumn (Object columnName): 모델에 열을 추가
    - addRow (Object [] rowData) : 모델의 마지막에 행을 추가
    - getColumnCount (): 데이터 테이블내의 열의 수를 리턴
    - getColumnName (int column): 열의 이름을 리턴
    - int getRowCount (): 데이터 테이블내의 행의 수를 리턴
    - getValueAt (int row, int column): row 및 column 에 위치하는 셀의 값을 리턴
    - insertRow (int row, Object [] rowData): 모델의 row 에 행을 추가
    - moveRow (int start, int end, int to): start 로부터 end 까지의 1 행 또는 복수 행을, to 의 위치로 이동
    - void setValueAt (Object aValue, int row, int column): column 및 row 에 위치하는 셀의 값을 설정
    - updateUI(): JTable을 다시 출력

JTableTest.java

```
import java.awt.*;

import javax.swing.*;
import javax.swing.table.*;

public class JTableTest extends JFrame
{
    public JTableTest()
            {
        super("JTable 테스트");
        String title[] = {"번호", "이름", "나이"};
        String data[][] = {
            {"1","김    구 ", "40"},
            {"2","이순신", "41"},
            {"3","강감찬", "42"}
        };
        //JTable table = new JTable(data, title);
        DefaultTableModel model = new DefaultTableModel(data, title);
        JTable table = new JTable(model);
        JScrollPane sp = new JScrollPane(table);
        getContentPane().add(sp, BorderLayout.CENTER);
        setSize(300,150);
        setVisible(true);
    }
}
```

SwingMain.java

```
import javax.swing.*;

public class SwingMain {
    public static void main(String args[]){
        JTableTest jtable = new JTableTest();
        jtable.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

SWING

- Pane
  - ✓ 특정 목적에 맞도록 작성된 Component
  - ✓ JEditorPane
    - ❑ 여러 가지 형태의 문서를 보여줄 수 있는 Component
    - ❑ 일반 텍스트나 html, rtf 문서 출력 가능

생성자	설명
JEditorPane( )	새로운 JEditorPane 객체를 생성한다.
JEditorPane(String url)	지정한 url의 내용을 보여주는 JEditorPane 객체를 생성한다.
JEditorPane(String type, String text)	지정한 type과 text의 내용을 보여주는 JEditorPane 객체를 생성한다.
JEditorPane(URL initialPage)	지정한 url을 보여주는 JEditorPane 객체를 생성한다.

SWING

- JTabbedPane
  - ✓ 여러 패널을 담을 때 사용하는 Component
  - ✓ 일반적으로 기능별로 분류된 옵션들을 동시에 보여 줄 필요가 없고 필요 시 하나의 패널만 보여주기 위해서 사용을 하는 Component
  - ✓ 타이틀이나 아이콘을 가지는 탭을 클릭함으로써 여러 개의 패널 중에 선택된 탭으로 교체되면서 화면에 보여주는 Component
  - ✓ 탭의 위치는 상하좌우에 위치할 수 있는데 기본적으로는 패널의 왼쪽 위(Top)에 배치

생성자	설명
JTabbedPane( )	비어있는 새로운 탭 패널 객체를 생성한다. 기본적인 탭의 위치는 Top 이다.
JTabbedPane(int tabPlacement)	지정된 탭의 위치를 가지는 새로운 탭 패널 객체를 생성한다. 탭의 위치는 LEFT, RIGHT, TOP, BOTTOM값을 가질 수 있다.

SWING

❖ JTabbedPane

반환형	메서드	설명
Component	add(Component component)	컴포넌트의 이름으로 탭 타이틀을 정하여 component를 붙인다.
	add(Component component, int index)	컴포넌트의 이름으로 탭 타이틀을 정하고 index로 탭의 위치를 정한 후에 component를 붙인다.
void	add(Component component, constraints)	component를 탭에 붙인다. 만약, Object constraints가 문자열이거나 Icon인 경우에는 이것을 이용하여 탭 타이틀로 정한다.
	add(Component component, Object constraints, int index)	index로 탭의 위치를 정하고 component를 탭에 붙인다. 만약, constraints가 문자열이거나 Icon인 경우에는 이것을 이용하여 탭 타이틀로 정한다.
	addTab(String title, Component component)	맨 마지막 탭 위치에 지정한 타이틀을 가지고 컴포넌트를 추가한다.
	addChangeListener(ChangeListener l)	탭 패널에서 선택된 탭의 인덱스가 바뀌었을 때 발생하는 이벤트를 받기 위해 지정된 ChangeListener를 추가한다.
	addTab(String title, Icon icon, Component component)	맨 마지막 탭 위치에 지정한 타이틀과 아이콘을 가지고 컴포넌트를 추가한다.
	insertTab(String title, Icon icon, Component component, String tip, int index)	index 위치에 컴포넌트를 추가한다. 지정한 타이틀과 아이콘을 탭에 보여준다.
	remove(Component component)	지정한 컴포넌트를 탭 패널에서 제거한다.
	removeTabAt(int index)	지정한 인텍스의 탭을 탭 패널에서 제거한다.
	setComponentAt(int index, Component component)	지정한 위치의 탭이 보여줄 컴포넌트를 지정한다.

JTabbedPaneTest.java

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

class JTable1 extends JPanel{
    public JTable1(){
        String title[] = {"번호", "이름", "나이"};
        String data[][] = {
            {"1","박문석", "40"},
            {"2","이종범", "40"},
            {"3","강호동", "40"}
        };
        JTable table = new JTable(data, title);
        JScrollPane sp = new JScrollPane(table);
       add(sp);
    }
}
```

JTabbedPaneTest.java

```
class JTable2 extends JPanel {
  public JTable2(){
       String title[] = {"번호", "직업", "소속"};
        String data[][] = {
            {"1","프로그래머", "프리랜서"},
            {"2","야구선수", "KIA"},
            {"3","MC", "프리랜서"}
        };
        JTable table = new JTable(data, title);
        JScrollPane sp = new JScrollPane(table);
        add(sp);
    }
}
```

JTabbedPaneTest.java

```
public class JTabbedPaneTest extends JFrame{
      JTabbedPane tab;
      JTable1 j1;
                  JTable2 j2;
      public JTabbedPaneTest()
                 {
           super("JTabbedPane 연습예제");
           tab = new JTabbedPane(JTabbedPane.TOP);

           JPanel one = new JPanel();
                              j1 = new JTable1();
           JPanel two = new JPanel();
                              j2 = new JTable2();

           one.add(j1);
          two.add(j2);

          tab.addTab("기본 데이터", one);
          tab.addTab("기타 데이터", two);

          getContentPane().add(tab, BorderLayout.CENTER);
          setSize(500,200);
          setVisible(true);
     }
}
```

SwingMain.java

```
import javax.swing.*;

public class SwingMain {
    public static void main(String args[]){
        JTabbedPaneTest jt= new JTabbedPaneTest();
                            jt.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

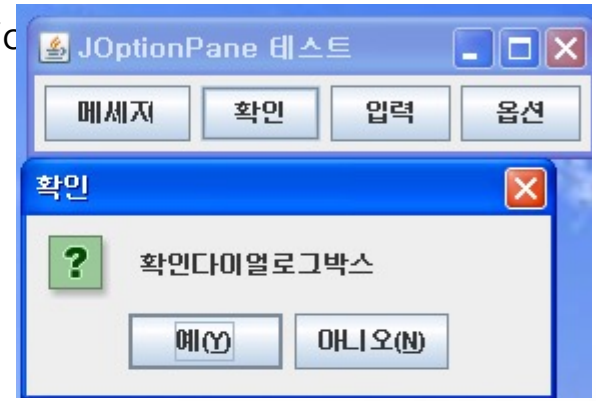
SWING

- JOptionPane
  - ✓ 프로그램을 실행하는 도중에 사용자로부터 데이터를 입력 받거나 특정한 메시지를 출력시켜 확인 시키는 작업들을 할 수 있도록 지원하는 Component
  - ✓ 자체적인 기능을 가지고 있는 것은 아니기 때문에 인스턴스를 생성한 후 반드시 showDialog()메서드를 이용해서 어떤 다이얼로그 박스를 출력시킬 것 인가를 설정해야 함

종류	기능	호출 함수
MessageDialog	사용자에게 메시지를 보여주는 다이얼로그 박스	showMessageDialog( )
ConfirmDialog	Yes, No, Cancel과 같은 버튼으로 확인하는 다이얼로그 박스	showConfirmDialog( )
InputDialog	사용자로부터 자료를 입력받기 위한 다이얼로그 박스	showInputDialog( )
OptionDialog	위 세 가지를 포함하여 맞춘 다이얼로그 박스	showOptionDialog( )

JOptionPaneTest.java

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

public class JOptionPaneTest extends JFrame implements Actio
        JButton btn[] = new JButton[4];
        String[] str = { "로그인", "회원가입" };

        public JOptionPaneTest() {
                super("JOptionPane 테스트");
                setLayout(new FlowLayout());

                btn[0] = new JButton("메세지");
                btn[1] = new JButton("확인");
                btn[2] = new JButton("입력");
                btn[3] = new JButton("옵션");

                for (JButton n : btn) {
                        add(n);
                        n.addActionListener(this);
                }
                pack();
                setLocation(300, 300);
                setVisible(true);
        }
```



JOptionPaneTest.java

```
		public void actionPerformed(ActionEvent e) {
			if (e.getActionCommand() == "메세지") {
				JOptionPane.showMessageDialog(this, "메세지다이얼로그박스", "
메세지",
						JOptionPane.INFORMATION_MESSAGE);
			} else if (e.getActionCommand() == "확인") {
				JOptionPane.showConfirmDialog(this, "확인다이얼로그박스", "확인
",
						JOptionPane.YES_NO_OPTION);
			} else if (e.getActionCommand() == "입력") {
				JOptionPane.showInputDialog(this, "입력다이얼로그박스", "입력",
						JOptionPane.YES_NO_OPTION);
			} else if (e.getActionCommand() == "옵션") {
				JOptionPane.showOptionDialog(this, "옵션다이얼로그박스", "옵션",
						JOptionPane.YES_NO_CANCEL_OPTION,
						JOptionPane.INFORMATION_MESSAGE, null,
str, str[1]);
			}
		}

}
```

SwingMain.java

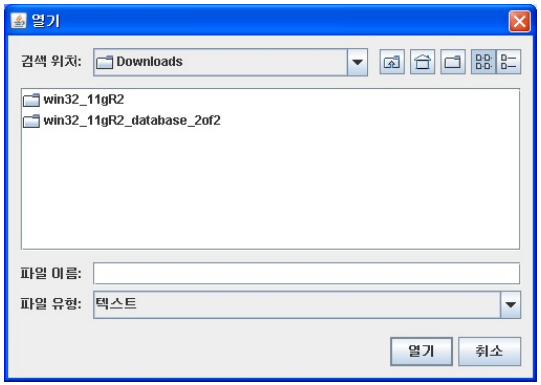
```
public class SwingMain {
		public static void main(String[] args) {
				new JOptionPaneTest();
		}
}
```

SWING

- JFileChooser
  - ✓ 프로그램을 실행하는 도중에 데이터를 파일로부터 불러오거나 파일에 저장할 수 있도록 파일 선택 다이얼로그 상자를 사용할 수 있도록 만든 Component
  - ✓ 파일 선택창은 FileSystemView, FileView, FileFilter 등과 같은 컨트롤러가 조합되어 생성
  - ✓ FileSystemView는 파일 시스템과 디렉토리 정보를 제공하고, FileView는 디렉토리 내부에 있는 파일들에 대한 정보를 제공
  - ✓ FileFilter는 파일을 원하는 종류만 보여줄 수 있도록 걸러주는 역할
  - ✓ 멤버 필드
    - ❑ static int CANCEL_OPTION: 취소를 선택했을 때 리턴되는 값
    - ❑ static int APPROVE_OPTION: 예나 확인을 선택했을 때 리턴되는 값
  - ✓ 생성자
    - ❑ JFileChooser(): 디폴트 경로
    - ❑ JFileChooser(File currentDirectory): 지정한 경로를 보여줌
    - ❑ JFileChooser(String currentDirectoryPath): 지정한 경로를 보여줌

JFileChooserTest.java

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

class JFileChooserTest extends JFrame implements ActionListener {
        JFileChooser fc;
        public JFileChooserTest()
        {
                fc = new JFileChooser();
                fc.setMultiSelectionEnabled(true);
                JButton btn = new JButton("파일 선택");
                btn.addActionListener(this);
                add("North",btn);
                setBounds(0,0,200,200);
                setVisible(true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
```



JFileChooserTest.java

```
		public void actionPerformed(ActionEvent e)
		{
			FileNameExtensionFilter filter = new FileNameExtensionFilter("텍스트", "txt");
			fc.setFileFilter(filter);
			int result = fc.showOpenDialog(this);
			if(result == JFileChooser.APPROVE_OPTION)
			{
				File [] f = fc.getSelectedFiles();
				int i=0;
				for(File n : f)
					System.out.println(++i + "번째 선택한 파일:" +
n.getName());
			}
		}
}
```

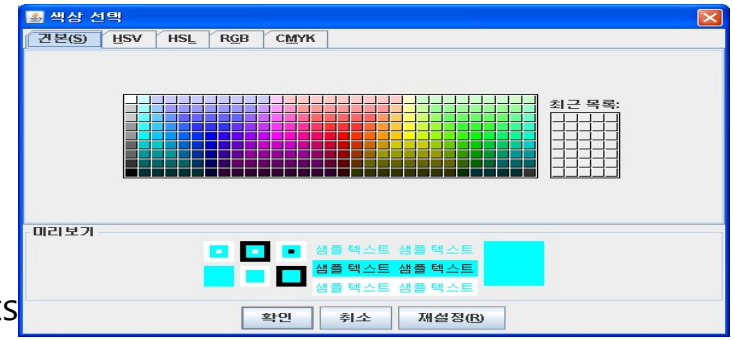
SwingMain.java

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

SWING

- JColorChooser
  - ✓ 색상을 선택할 수 있는 다이얼로그
  - ✓ 생성자
    - ❑ JColorChooser(): 초기 색이 흰색
    - ❑ JColorChooser(Color initialColor): 초기 색이 매개 변수
    - ❑ JColorChooser (ColorSelectionModel model): 지정된 선택 판 모델을 가지고 생성
  - ✓ 메서드
    - ❑ static JDialog createDialog (Component c, String title, boolean modal,JColorChooser chooserPane, ActionListener okListener, ActionListener cancelListener): [OK],[Cancel], 및 [Reset] 버튼과 함께 지정된 ColorChooser 구획을 가지는 새로운 다이얼로그를 생성
    - ❑ Color getColor(): 색상을 리턴
    - ❑ void setColor(Color color): 색 설정
    - ❑ void setColor(int c): 색 설정
    - ❑ void setColor(int r, int g, int b): 색 설정
    - ❑ static Color showDialog(Component component, String title, Color initialColor) 모달 다이얼로그로 화면에 표시

JColorChooserTest.java




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

public class JColorChooserTest extends JFrame implements
{
        JLabel lbl;
        JButton btn;
        public JColorChooserTest()
        {
                super("JColorChooser 테스트");
                btn = new JButton("색상 대화 상자 호출");
                lbl = new JLabel("색상 출력");
                add("North",lbl);
                add("Center",btn);
                btn.addActionListener(this);
                setBounds(300, 300, 300, 200);
                setVisible(true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
```

JColorChooserTest.java

```
		public void actionPerformed(ActionEvent e)
		{
			if(e.getActionCommand()=="색상 대화 상자 호출")
			{
				Color rgb=JColorChooser.showDialog(this,"색상 선택",
Color.CYAN);
				lbl.setText("색상:" + rgb);
				btn.setBackground(rgb);
			}
		}
		}
```

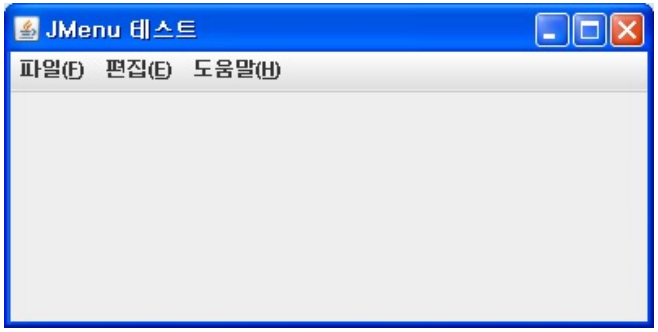
SwingMain.java

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

SWING

- Menu
  - ✓ JMenuBar, JMenu, JMenuItem으로 구성
  - ✓ 사용 방법
    - ❑ 1단계: JMenuBar 인스턴스를 생성
    - ❑ 2단계: setJMenuBar(JMenuBar인스턴스)를 이용해서 컨테이너에 메뉴바 부착
    - ❑ 3단계: JMenu인스턴스를 생성해서 JMenuBar 인스턴스의 add(JMenu인스턴스)로 부착
    - ❑ 4단계: JMenuItem 인스턴스를 생성해서 JMenu인스턴스의 add (JMenuItem 인스턴스)로 부착
  - ✓ 스윙에서는 JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem의 메뉴 아이템이 제공
  - ✓ 메뉴 인스턴스에서 setMnemonic(int mnemonic)을 이용해서 단축키도 지정이 가능 - ALT와 조합되서 단축키가 됩니다.

JMenuTest.java




```
import javax.swing.*;
public class JMenuTest extends JFrame
{
   private JRadioButtonMenuItem colorItems[], fonts[];
   private JCheckBoxMenuItem styleItems[];
   private ButtonGroup fontGroup, colorGroup;

   public JMenuTest()
   {
      super( "JMenu 테스트" );
      JMenuBar bar = new JMenuBar();
      setJMenuBar( bar );
      JMenu fileMenu = new JMenu( "파일(F)" );
               fileMenu.setToolTipText("파일 메뉴입니다");
      fileMenu.setMnemonic( 'F' );
      JMenuItem newItem = new JMenuItem( "새파일(N)" );
      newItem.setMnemonic( 'N' );
      fileMenu.add( newItem );
      JMenuItem openItem = new JMenuItem( "열기(O)" );
      openItem.setMnemonic( 'O' );
      fileMenu.add( openItem );
      JMenuItem saveItem = new JMenuItem( "저장(S)" );
      saveItem.setMnemonic( 'S' );
      fileMenu.add( saveItem );
```

JMenuTest.java

```
 JMenuItem exitItem = new JMenuItem( "닫기(X)" );
exitItem.setMnemonic( 'X' );
fileMenu.add( exitItem );
bar.add( fileMenu );

JMenu formatMenu = new JMenu( "편집(E)" );
          formatMenu.setToolTipText("편집 메뉴입니다");
formatMenu.setMnemonic( 'E' );
String colors[] = { "검정", "파랑", "빨강", "초록" };
JMenu colorMenu = new JMenu( "색상(C)" );
colorMenu.setMnemonic( 'C' );
colorItems = new JRadioButtonMenuItem[ colors.length ];
colorGroup = new ButtonGroup();
for ( int i = 0; i < colors.length; i++ )
          {
   colorItems[ i ] = new JRadioButtonMenuItem( colors[ i ] );
   colorMenu.add( colorItems[ i ] );
   colorGroup.add( colorItems[ i ] );
}

colorItems[ 0 ].setSelected( true );
formatMenu.add( colorMenu );
formatMenu.addSeparator();
```

JMenuTest.java

```
String fontNames[] = { "굴림", "바탕", "고딕" };
JMenu fontMenu = new JMenu( "글꼴(T)" );
        fontMenu.setMnemonic( 'T' );
fonts = new JRadioButtonMenuItem[ fontNames.length ];
fontGroup = new ButtonGroup();

for ( int i = 0; i < fonts.length; i++ )
        {
   fonts[ i ] = new JRadioButtonMenuItem( fontNames[ i ] );
   fontMenu.add( fonts[ i ] );
   fontGroup.add( fonts[ i ] );
}

fonts[ 0 ].setSelected( true );
fontMenu.addSeparator();

String styleNames[] = { "굵게", "기울임" };
styleItems = new JCheckBoxMenuItem[ styleNames.length ];
```

JMenuTest.java

```
for ( int i = 0; i < styleNames.length; i++ )
                {
        styleItems[ i ] = new JCheckBoxMenuItem( styleNames[ i ] );
        fontMenu.add( styleItems[ i ] );
    }
    formatMenu.add( fontMenu );
    bar.add( formatMenu );
            JMenu helpMenu = new JMenu( "도움말(H)" );
            helpMenu.setToolTipText("도움말 메뉴입니다");
            helpMenu.setMnemonic( 'H' );
JMenuItem helpItem = new JMenuItem( "도움말항목(L)" );
    helpItem.setMnemonic( 'L' );
    helpMenu.add(helpItem);
            bar.add( helpMenu );
setSize( 400, 200 );
    setVisible(true);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
  }
}
```

SwingMain.java

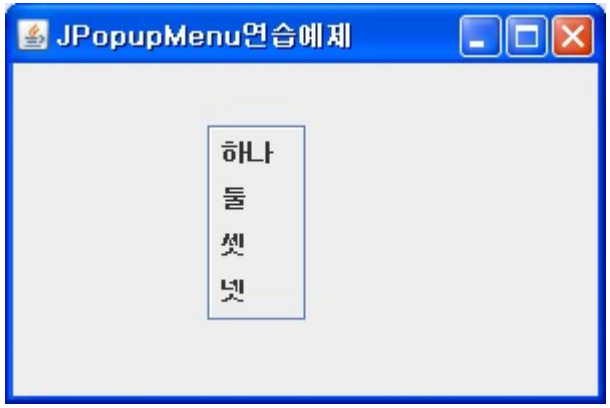
```
public class SwingMain {
		public static void main( String args[] )
   {
      new JMenuTest();
   }
}
```

Swing의 Component

- JPopupMenu
  - ✓ 생성자
    - ❑ JPopupMenu (): 디폴트 생성자
    - ❑ JPopupMenu (String label): label을 타이틀로 갖는 팝업 메뉴
  - ✓ 메서드
    - ❑ JMenuItem add(JMenuItem menuItem): menuItem 추가
    - ❑ void show (Component invoker, int x, int y): Component 내에서 x, y 좌표로 지정된 위치에 pop-up menu를 표시
  - ✓ 마우스 이벤트가 발생해야 하고 이때 MouseEvent 인스턴스의 isPopupTrigger() 메서드를 확인해서 값이 true이면 마우스 오른쪽 버튼을 누른 것이므로 팝업메뉴를 화면에 출력하면 됩니다.

JPopupTest.java

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

public class JPopupTest extends JFrame
{
   JMenuItem items[];
   public JPopupTest()
   {
      super( "JPopupMenu연습예제" );

      final JPopupMenu popupMenu = new JPopupMenu();
      String str[] = { "하나", "둘", "셋", "넷" };
      items = new JMenuItem[4];

      for ( int i = 0; i < items.length; i++ )
                {
         items[i] = new JMenuItem(str[ i ]);
         popupMenu.add(items[ i ]);
      }
```



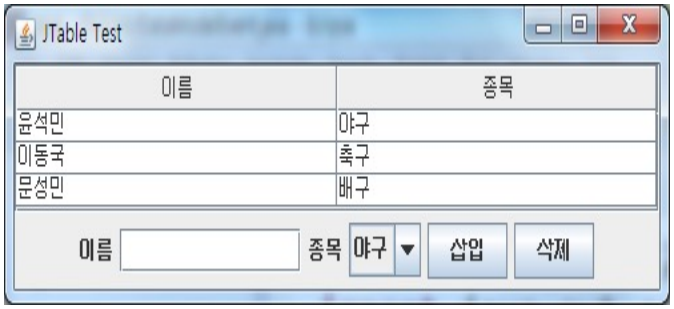
JPopupTest.java

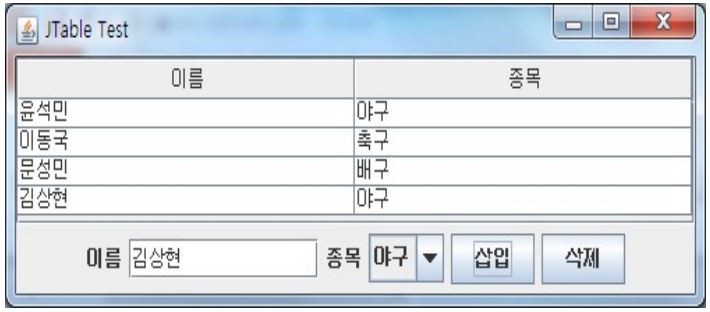
```
addMouseListener
                (
        new MouseAdapter() {
           public void mousePressed( MouseEvent e )
              { checkForTriggerEvent( e ); }

           public void mouseReleased( MouseEvent e )
              { checkForTriggerEvent( e ); }

           public void checkForTriggerEvent( MouseEvent e ){
              if ( e.isPopupTrigger() )
                 popupMenu.show( e.getComponent(), e.getX(), e.getY() );
           }
        }
     );
     setSize( 300, 200 );
     setVisible(true);
              setDefaultCloseOperation(EXIT_ON_CLOSE);
  }
}
```

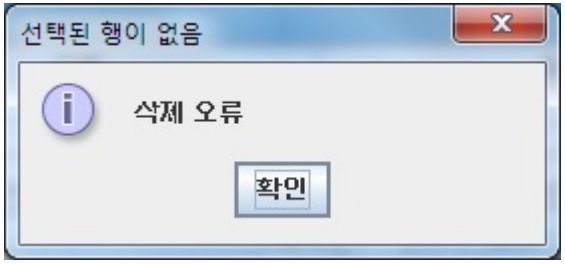
SwingMain.java

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

Table 사용


JTable Test
이름
종목
윤석민
야구
이동국
축구
문성민
배구
이름
종목
야구
삽입
삭제


JTable Test
이름
종목
윤석민
야구
이동국
축구
문성민
배구
김상현
야구
이름
김상현
종목
야구
삽입
삭제


선택된 행이 없음
삭제 오류
확인

DataModel

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

public class DataModelEvent extends JFrame{
		public DataModelEvent() {
				String rows[][] = { { "윤석민", "야구" }, { "이동국", "축구" }, { "문성민", "배구
" } };
				String headers[] = { "이름", "종목" };
				String sports[] = { "야구", "축구", "배구", "기타" };

				setTitle("JTable Test");
				setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
				JTable table = new JTable(new DefaultTableModel(rows, headers));

				JScrollPane scrollPane = new JScrollPane(table);
				add(scrollPane, BorderLayout.CENTER);
```

DataModel

```
        JPanel panel = new JPanel();
        JTextField text = new JTextField(12);
        JComboBox<String> comboBox = new JComboBox<String>(sports);
        comboBox.setMaximumRowCount(4);

        JButton btn1 = new JButton("삽입");
        JButton btn2 = new JButton("삭제");

        btn1.addActionListener(new AddActionListener(table, text, comboBox));
        btn2.addActionListener(new RemoveActionListener(table));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        panel.add(new JLabel("이름"));
        panel.add(text);
        panel.add(new JLabel("종목"));
        panel.add(comboBox);
        panel.add(btn1);
        panel.add(btn2);

        add("South", panel);

        setSize(500, 150);
        setVisible(true);
    }
}
```

DataModel

```
class AddActionListener implements ActionListener
{
		JTable table;
		JTextField text;
		JComboBox <String>combo;
		AddActionListener(JTable table, JTextField text, JComboBox<String> combo)
		{
			this.table = table;
			this.text = text;
			this.combo = combo;
		}

		public void actionPerformed(ActionEvent e)
		{
			String arr[] = new String[2];
			arr[0] = text.getText();
			arr[1] = (String)combo.getSelectedItem();
			DefaultTableModel model = (DefaultTableModel)table.getModel();
			model.addRow(arr);
		}
}
```

DataModel

```
class RemoveActionListener implements ActionListener
{
		JTable table;
		RemoveActionListener(JTable table)
		{
			this.table = table;
		}

		public void actionPerformed(ActionEvent e)
		{
			int row = table.getSelectedRow();
			if ( row == -1)
			{
				JOptionPane.showMessageDialog(null,"삭제 오류","선택된 행이 없
음",JOptionPane.INFORMATION_MESSAGE);
				return;
			}
			DefaultTableModel model = (DefaultTableModel) table.getModel();
			model.removeRow(row);
		}
}
```

SwingMain.java

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

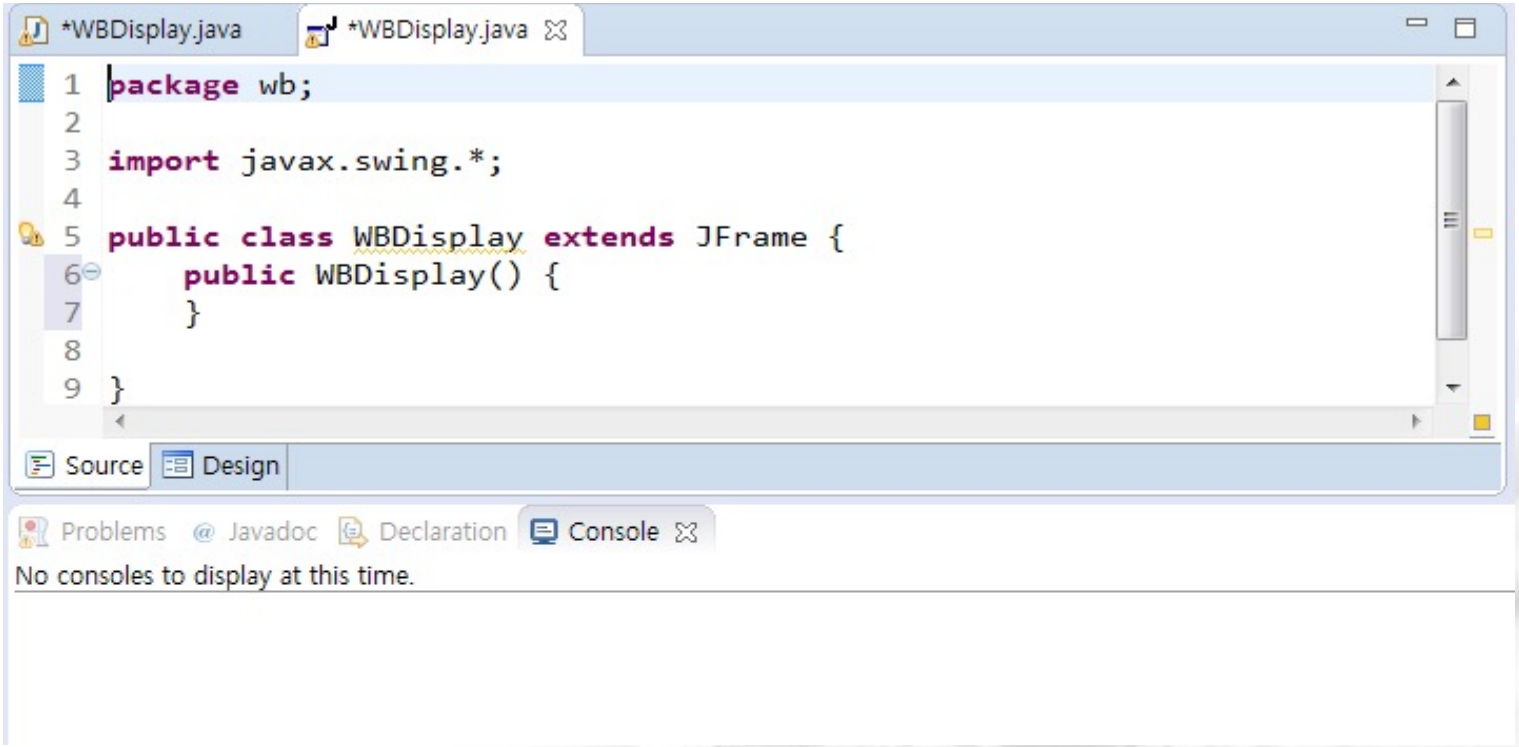
WindowBuilder

WindowBuilder

- Window Builder
  - ✓ 자바 GUI 프로그래밍을 위한 이클립스 플러그인
  - ✓ SWT 디자이너와 스윙 디자이너로 구성
  - ✓ 자바 소스 코드에 의존하지 않고 비주얼 디자이너와 레이아웃 도구 등 간단한 양식을 작성하는 창을 이용해서 자동으로 자바 코드를 생성
  - ✓ 속성 변경도 속성 편집기를 이용해서 할 수 있으며 이벤트 처리 기능 추가를 드래그 앤 드롭으로 쉽게 구현할 수 있도록 합니다.
  - ✓ 컴파일 및 실행을 위해서 별도의 라이브러리를 사용하지 않음
  - ✓ Eclipse Market Place 에서 검색해서 설치

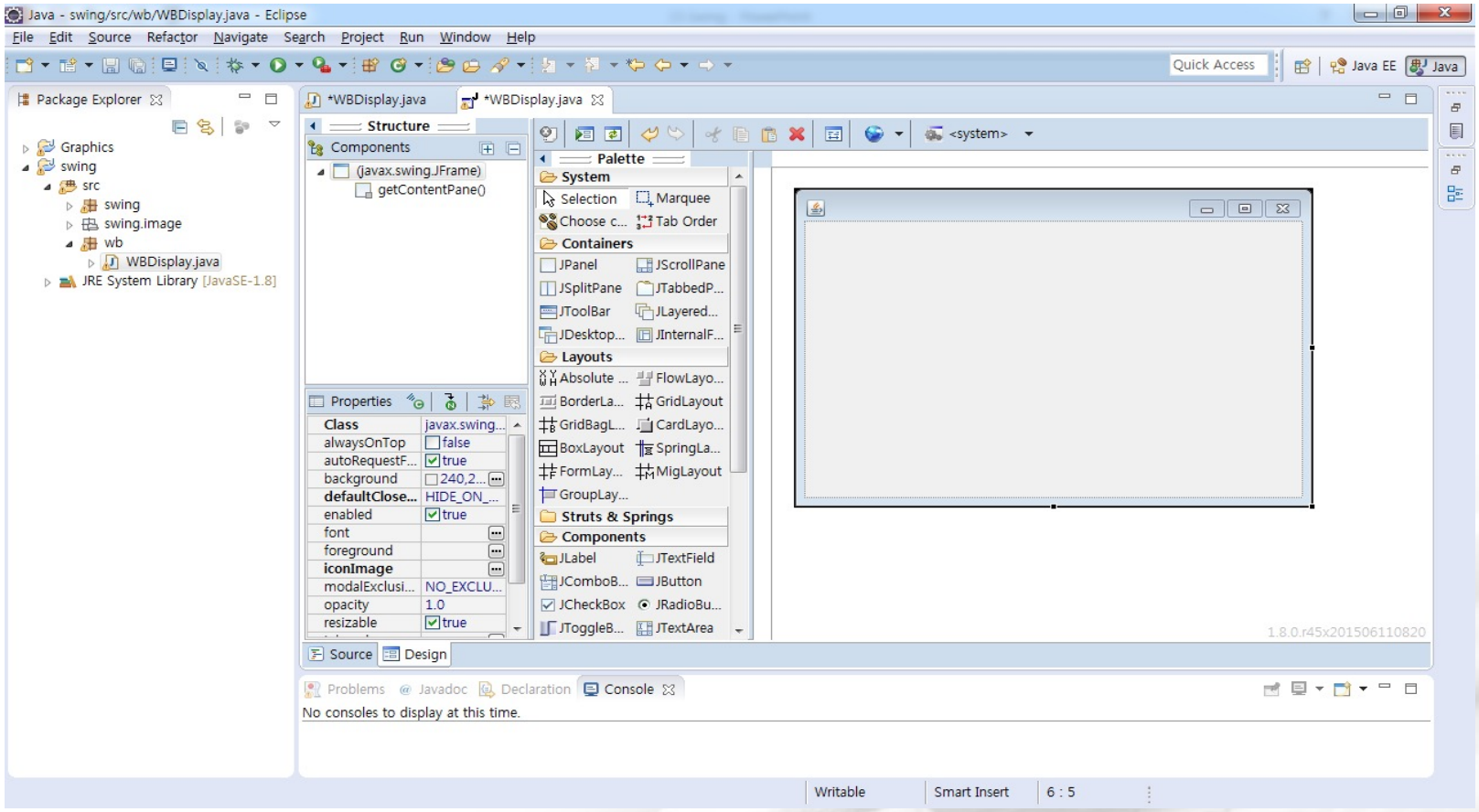
WindowBuilder

- 프로젝트 생성
- JFrame으로 부터 상속받는 클래스 생성(WBDisplay)
- 클래스를 선택하고 마우스 오른쪽을 클릭해서 [Open With] – [WindowBuilder Editor] 선택



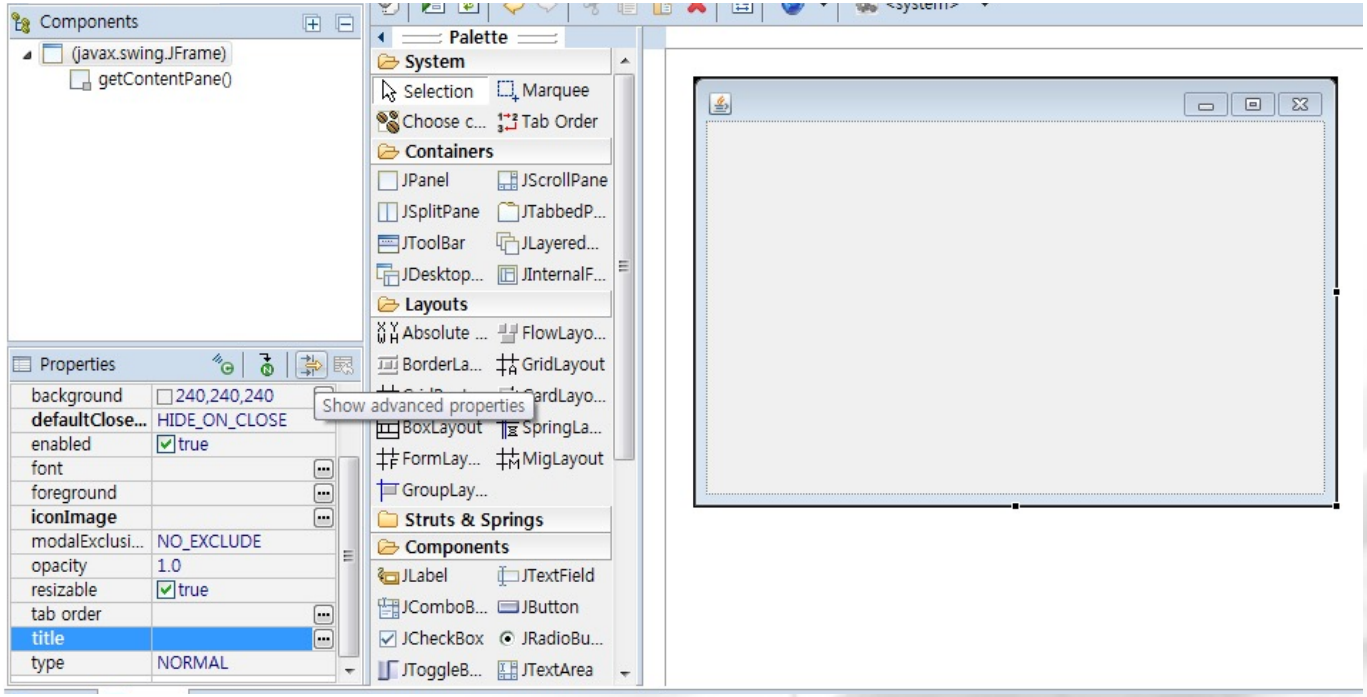
WindowBuilder

❖ 하단에서 Design 탭을 선택하면 디자인 할 수 있는 화면으로 변경됩니다.



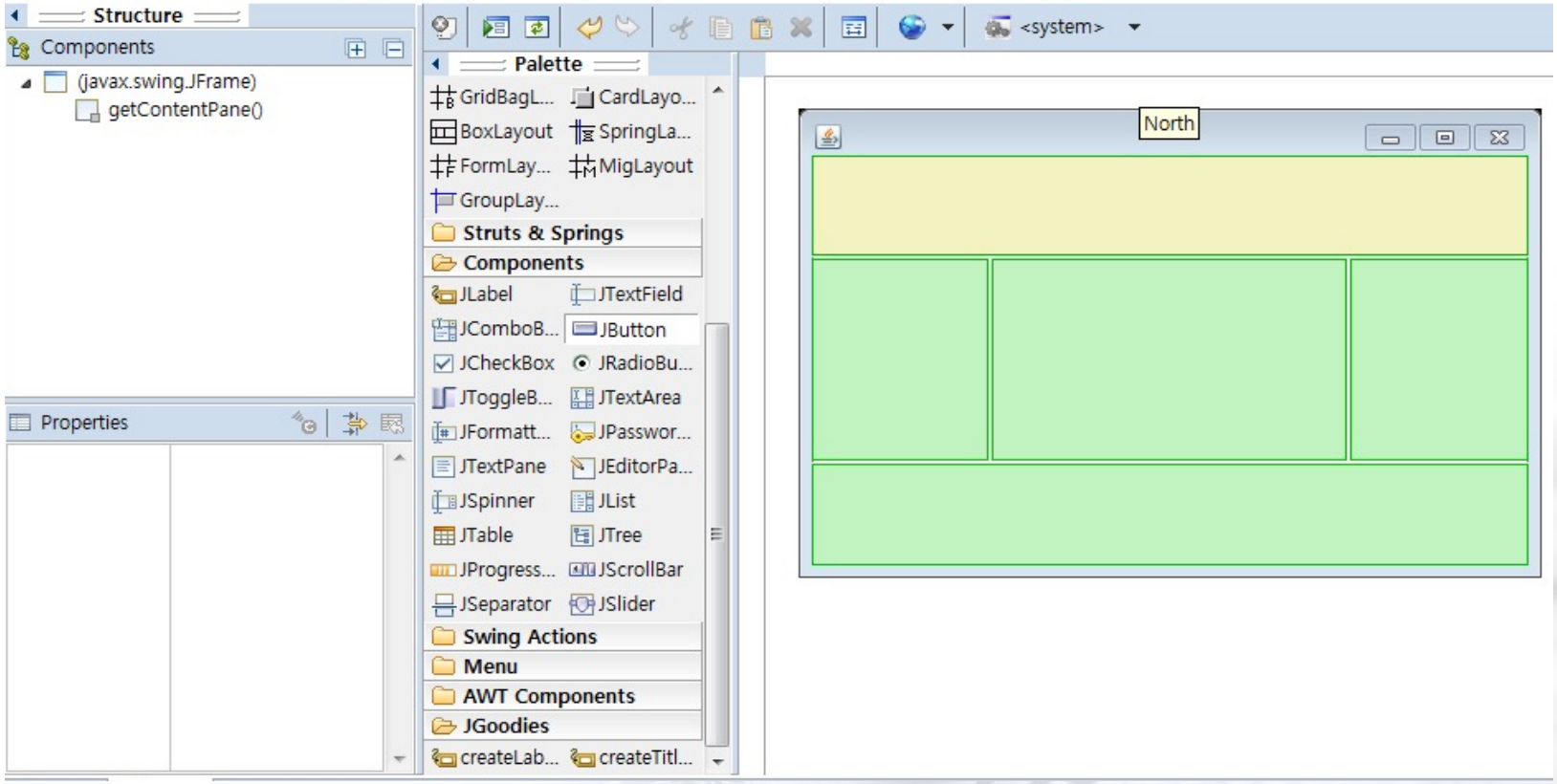
WindowBuilder

- Component의 속성을 변경하고자 하는 경우에는 Components 창이나 화면에 보이는 GUI 창에서 Component를 선택하고 Properties 창에서 변경하면 되는데 안보이는 속성이 있으면 Properties 창에서 [Show Advanced Properties]를 누르면 확인이 가능



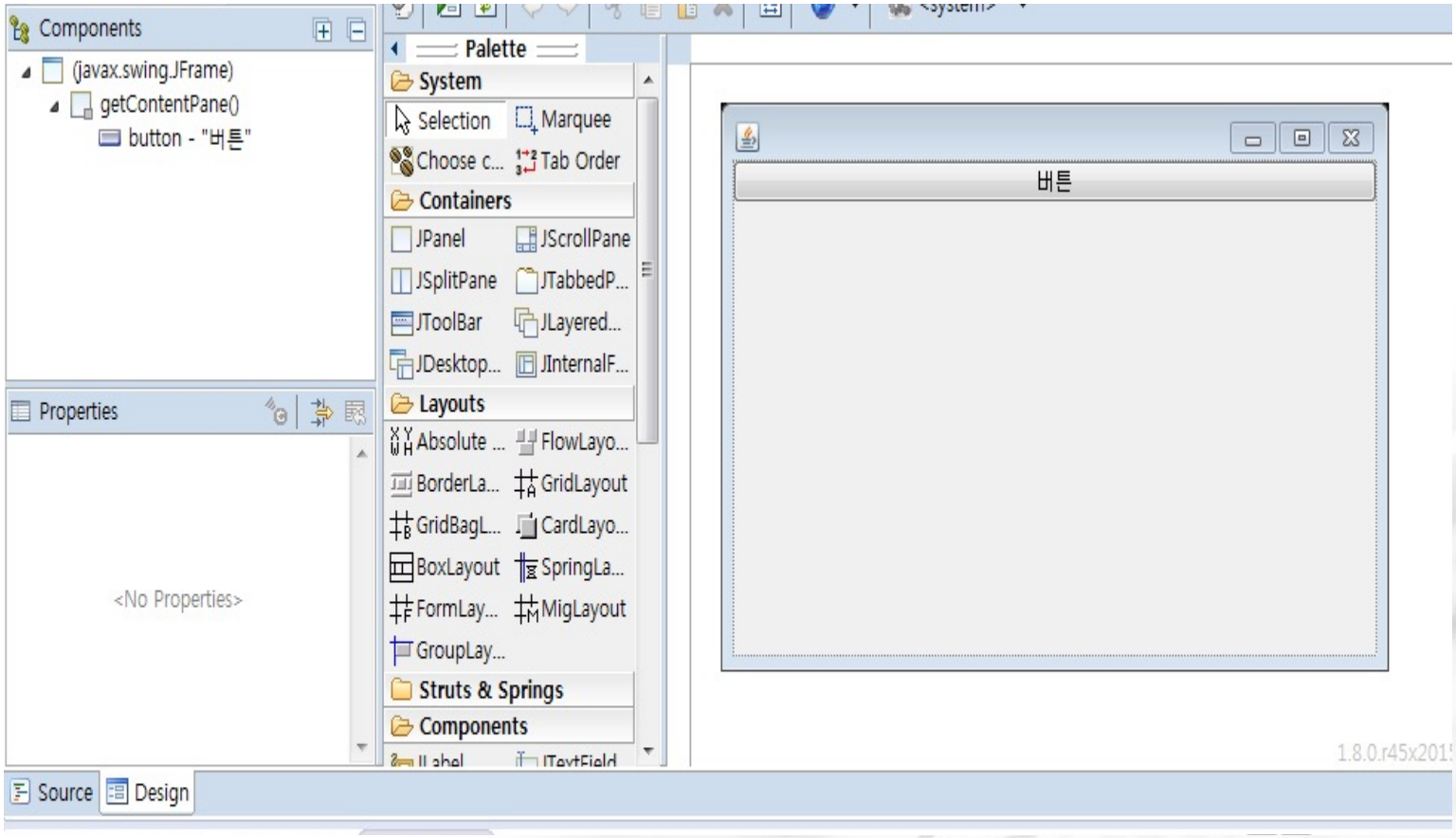
WindowBuilder

- Component를 추가하고자 하는 경우에는 Palette에서 선택한 후 GUI 화면에서 원하는 위치에 드랍



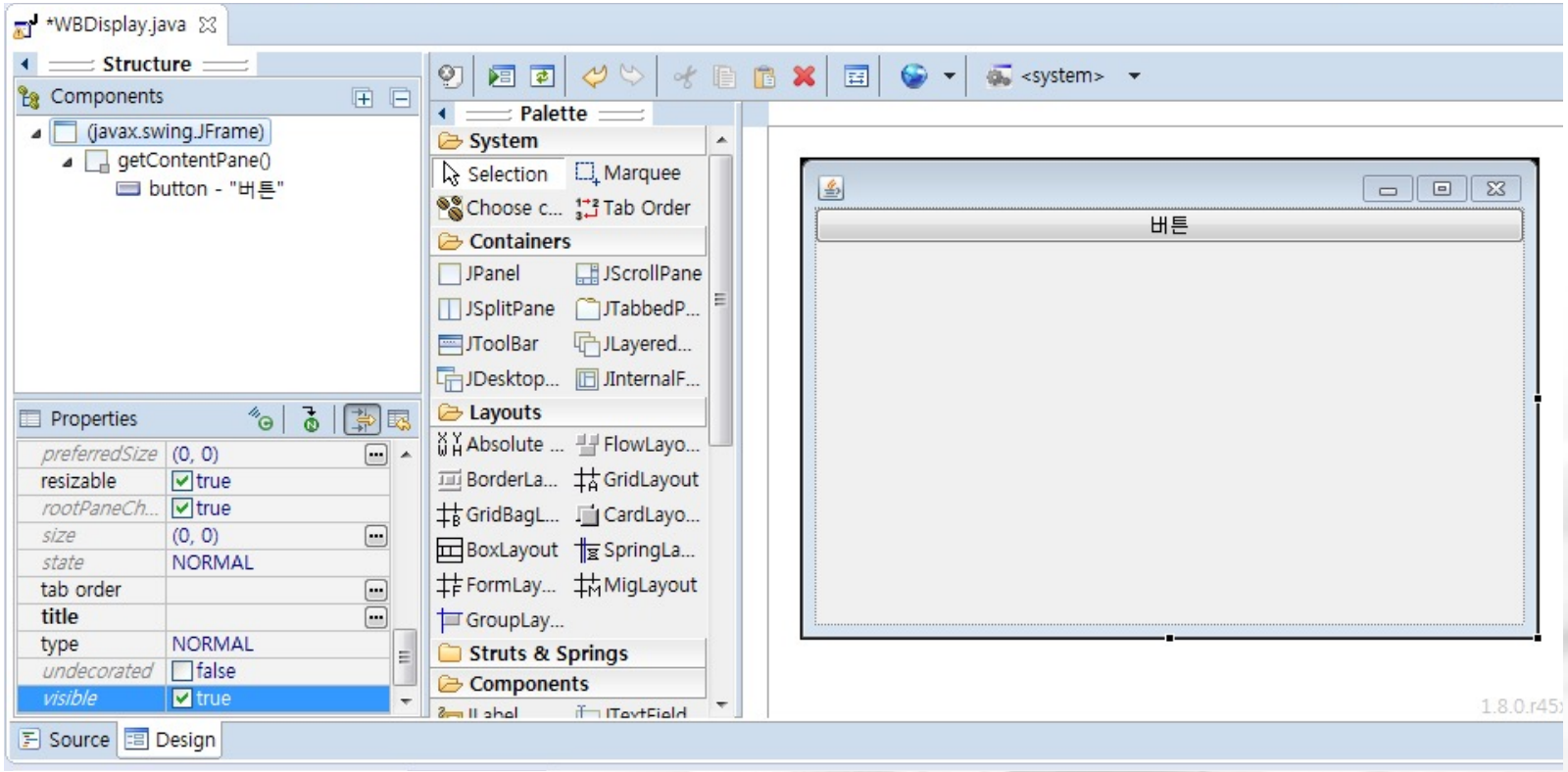
WindowBuilder

❖ 버튼을 북쪽에 배치



WindowBuilder

- JFrame을 선택하고 visible 속성을 true로 변경하고 size를 입력



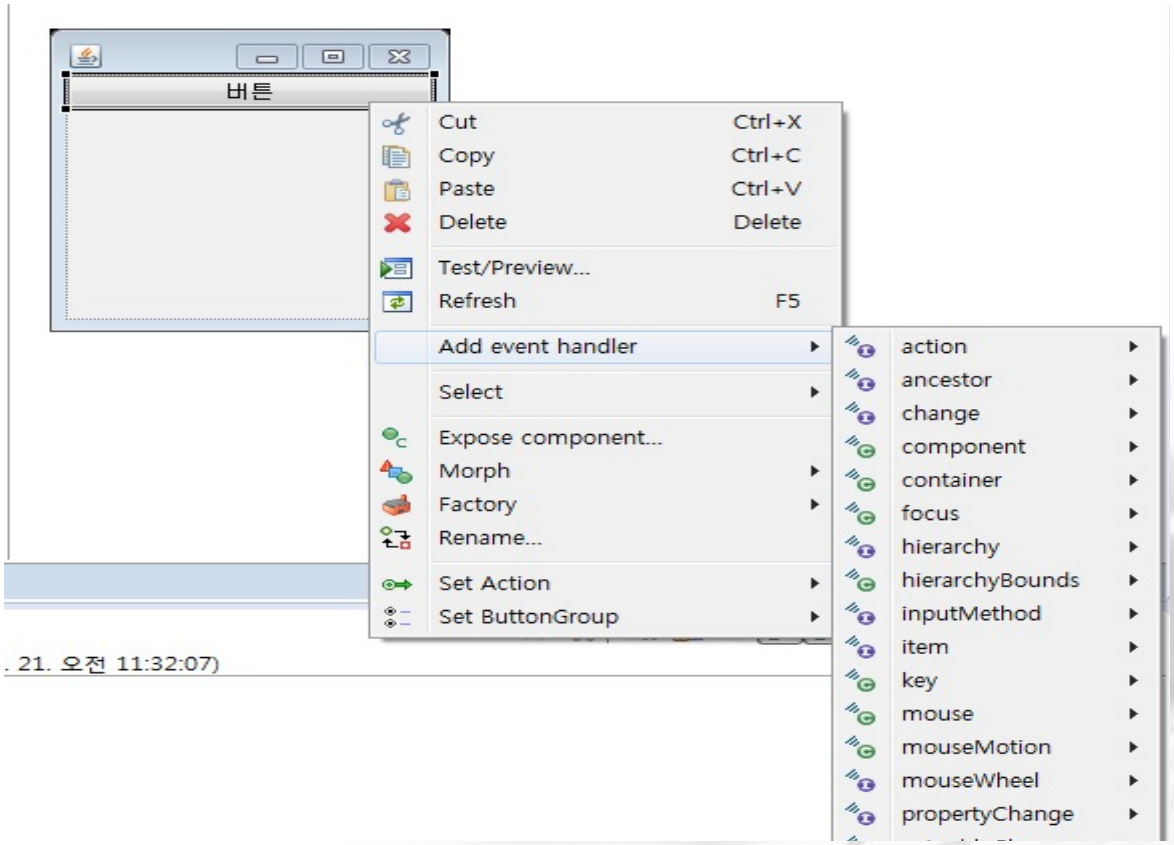
WindowBuilder

- Main 클래스를 생성하고 이전에 만든 클래스의 인스턴스 생성

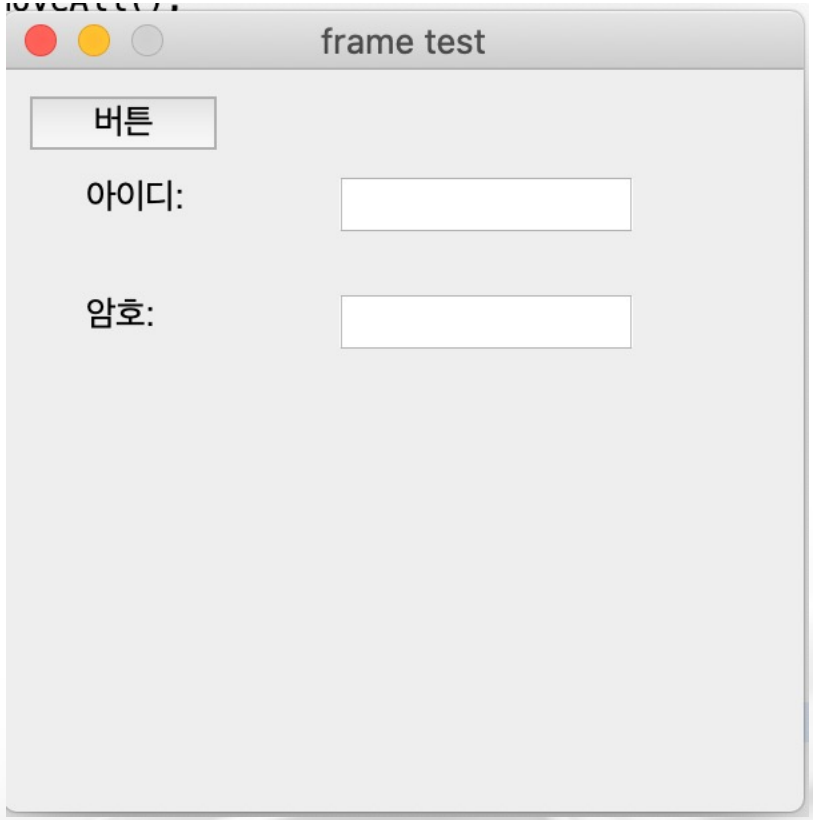
```
public class Main {
        public static void main(String[] args) {
                new WBDisplay();
        }
}
```

WindowBuilder

- 이벤트 처리: 이벤트 처리를 수행 할 Component를 선택하고 마우스 오른쪽 버튼을 누르고 [Add event handler]에서 처리할 이벤트 핸들러 메서드를 선택하고 작성



화면 전환



화면 전환

❖ 첫번째 화면(JPanel01)

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


class JPanel01 extends JPanel {  // 1번째 패널

        private JButton jButton1;
        private JScrollPane jScrollPane1;
        private JTextArea jTextArea1;
        private JPanelTest win;

        public JPanel01(JPanelTest win){
                this.win = win;
                setLayout(null);

                jButton1 = new JButton("버튼");
                jButton1.setSize(70,20);
                jButton1.setLocation(10, 10);
                add(jButton1);
```

화면 전환

❖ 첫번째 화면(JPanel01.java)

```
                jTextArea1 = new JTextArea();

                jScrollPane1 = new JScrollPane(jTextArea1);
                jScrollPane1.setSize(200,150);
                jScrollPane1.setLocation(10,40);
                add(jScrollPane1);

                jButton1.addActionListener(new MyActionListener());
        }

        class MyActionListener implements ActionListener {    // 버튼 키 눌리면 패널 2번 호출
                @Override
                public void actionPerformed(ActionEvent e) {
                        win.change("panel02");
                }
        }
}
```

화면 전환

❖ 두번째 화면(JPanel02)

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


class JPanel02 extends JPanel{          // 2번째 패널
    private JTextField textField;
    private JPasswordField passwordField;
    private JPanelTest win;

    public JPanel02(JPanelTest win) {
        setLayout(null);
        this.win = win;
        JLabel lblLbl = new JLabel("아이디:");
        lblLbl.setBounds(31, 40, 67, 15);
        add(lblLbl);

        textField = new JTextField();
        textField.setBounds(123, 40, 116, 21);
        add(textField);
        textField.setColumns(10);
```

화면 전환

❖ 두번째 화면(JPanel02)

```
        JLabel lblLbl_1 = new JLabel("암호:");
        lblLbl_1.setBounds(31, 84, 67, 15);
        add(lblLbl_1);

        passwordField = new JPasswordField();
        passwordField.setBounds(123, 84, 116, 21);
        add(passwordField);

        JButton btn = new JButton("버튼");
        btn.setSize(70,20);
        btn.setLocation(10,10);
        add(btn);
        btn.addActionListener(new MyActionListener());
    }
    class MyActionListener implements ActionListener {          // 버튼 키 눌리면 패널 1번 호출
        @Override
        public void actionPerformed(ActionEvent e) {
            win.change("panel01");
        }
    }
}
```

화면 전환

❖ 메인 화면(JPanelTest)

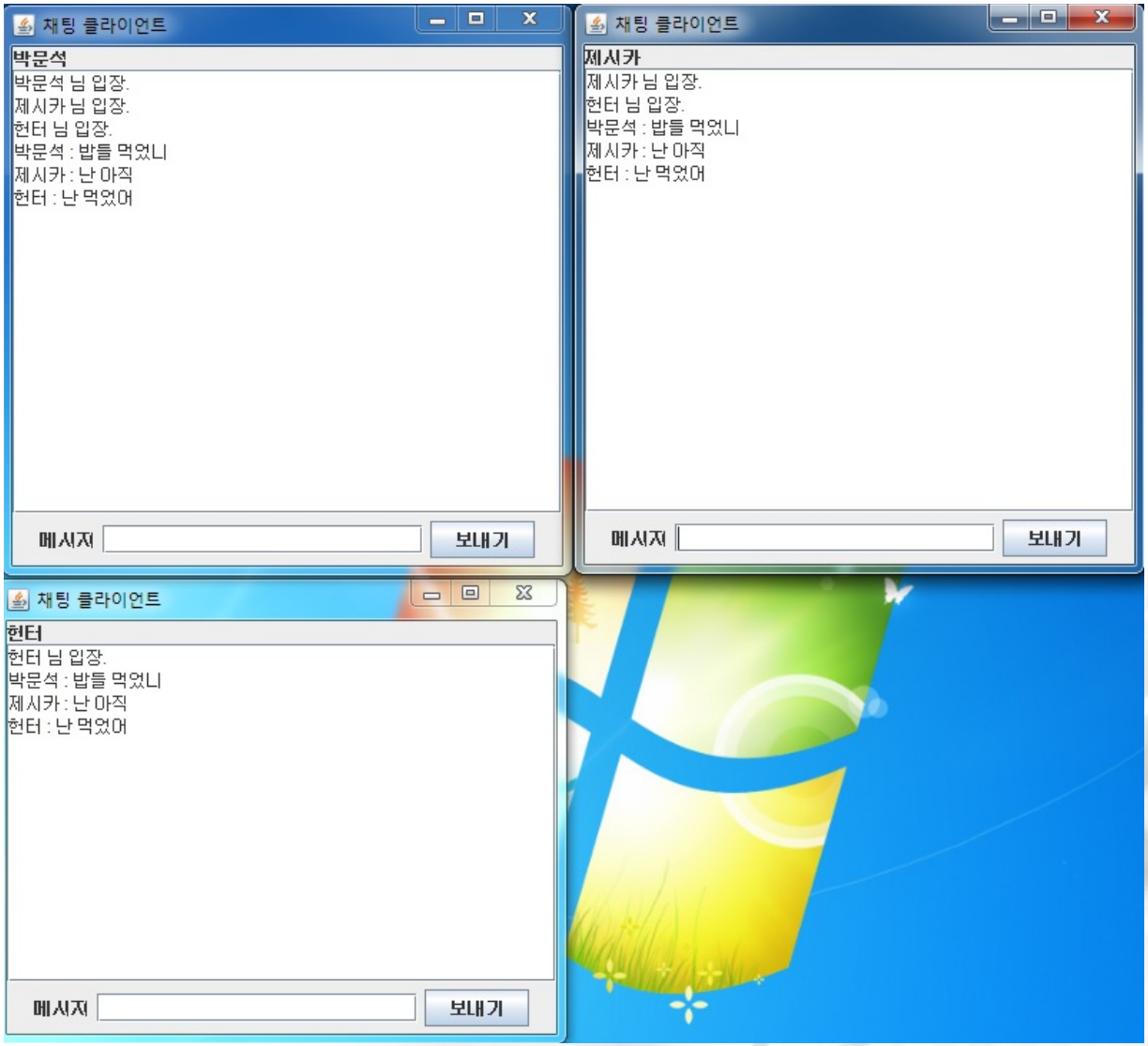
```
import javax.swing.JFrame;

class JPanelTest extends JFrame{

    public JPanel01 jpanel01 = null;
    public JPanel02 jpanel02 = null;

    public void change(String panelName){        // 패널 1번과 2번 변경 후 재설정

        if(panelName.equals("panel01")){
            getContentPane().removeAll();
            getContentPane().add(jpanel01);
            revalidate();
            repaint();
        }else {
            getContentPane().removeAll();
            getContentPane().add(jpanel02);
            revalidate();
            repaint();
        }
    }
```

화면 전환

❖ 메인 화면(JPanelTest)

```
    public static void main(String[] args) {

        JPanelTest win = new JPanelTest();

        win.setTitle("frame test");
        win.jpanel01 = new JPanel01(win);
        win.jpanel02 = new JPanel02(win);

        win.add(win.jpanel01);
        win.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        win.setSize(300,300);
        win.setResizable(false);
        win.setVisible(true);
    }
}
```

Multi Chatting



Multi Chatting

❖ **채팅에 사용할 스레드 클래스(ChatHandler.java)**

```
//스레드 핸들러 : ChatHandler클래스
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;

public class ChatHandler extends Thread {
        Socket client; //Socket 인스턴스
        BufferedReader input; //입력Stream
        PrintWriter output; //출력Stream
        GUIChatServer server; //서버 인스턴스

        //생성자, Socket인스턴스로부터 입출력 Stream을 얻어냄
        public ChatHandler(GUIChatServer server, Socket client) throws IOException {
                this.server = server;
                this.client = client;

                input = new BufferedReader(
                                new InputStreamReader(client.getInputStream()));
                output = new PrintWriter(
                                new OutputStreamWriter(client.getOutputStream()));
        }
```

Multi Chatting

❖ **채팅에 사용할 스레드 클래스(ChatHandler.java)**

```
//클라이언트가 보낸 메시지를 읽는 메서드
public void run() {
        String name = "";
        try {
                //클라이언트가 보낸메시지 읽음
                name = input.readLine();
                //클라이언트가 보낸메시지를 모든 클라이언트에게 중계(보냄)
                //모든 사용자에게 메시지를 중계하는 broadcast()메서드 호출
                broadcast (name + " 님 입장.");
                //무한히 클라이언트가 보낸 메시지를 받을 수 있도록 무한루프로
처리
                while (true) {
                        //클라이언트가 보낸 메시지 읽음
                        String msg = input.readLine();

                        //모든 사용자에게 메시지를 중계하는 broadcast()메서
드 호출
                        broadcast(name + " : " + msg);
                }
        }
```

Multi Chatting

❖ **채팅에 사용할 스레드 클래스(ChatHandler.java)**

```
                catch (Exception ex) {
                        System.out.println (name + " 님 퇴장, "
                                        + "IP: " + client.getInetAddress());

                } finally {//채팅창의 닫기단추를 클릭하면 수행
                        server.handlers.removeElement(this);
                        broadcast (name + " 님 퇴장");
                        try {//리소스 해제
                                input.close();
                                output.close();
                                client.close();
                        } catch (IOException ex) {
                                ex.printStackTrace();
                        }
                }
        }
```

Multi Chatting

❖ **채팅에 사용할 스레드 클래스(ChatHandler.java)**

```
//현재 접속한 모든 클라이언트에게 메시지를 보냄
protected void broadcast(String message) {
        //모든 사용자들에게 메시지를 중계하는 동안
        //벡터에서 클라이언트의 추가와 제거가 안됨
        synchronized (server.handlers) {
                int n = server.handlers.size();
                for(int i=0; i < n; i++) {
                        //클라이언트 하나를 얻어냄
                        ChatHandler handler = server.handlers.elementAt(i);
                        try{
        //메시지를 보내는 동안 출력Stream을 다른곳에서 사용못하게 함
                                synchronized (handler.output) {
                                //클라이언트에게 메시지 보냄
                                        handler.output.println(message);
                                }
                                handler.output.flush ();
                        }catch (Exception ex) {
                                ex.printStackTrace();
                        }
                }
        }
}
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
//클라이언트 : ChatClient클래스

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.Socket;
import javax.swing.*;

public class GUIChatClient extends JFrame implements Runnable, ActionListener{
        private static final long serialVersionUID = 1L;

        BufferedReader input; //입력Stream
        PrintWriter output; //출력Stream
        Thread handler; //ChatHandler와 메시지를 주고 받기 위한 쓰레드
        Container c;
        JTextArea display; //채팅창에서 대화를 표시
        JTextField id; //사용자 id
        JTextField inData; //사용자가 메시지를 입력하는 필드
        JLabel displayId; //채팅창에 id표시하는 레이블
        JButton      send; //[보내기]버튼
        CardLayout window;
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
//생성자로 채팅창의 UI를 구성
public GUIChatClient(){
        super("채팅 클라이언트");

        //컨테이너에 레이아웃 매니저 지정
        c = getContentPane();
        window = new CardLayout();
        c.setLayout(window);

        //로그인 창을 구성
        JPanel login = new JPanel(new BorderLayout());
        JPanel bottom = new JPanel();
        JLabel idLable = new JLabel("로그인:");
        //아이디 입력필드의 생성과 리스너 등록
        id = new JTextField(15);
        id.addActionListener(this);
        //로그인창의 Component배치
        bottom.add(idLable);
        bottom.add(id);
        login.add("South",bottom);
        c.add("login", login);
        //채팅창을 구성
        JPanel chat = new JPanel(new BorderLayout());
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
//채팅창의 대화 표시 텍스트에리어 생성 및 스크롤바 추가, 배치
display = new JTextArea (10,30);
JScrollPane s = new JScrollPane(display);
chat.add ("Center", s);//패널에 추가
display.setEditable(false); //대화표시 화면에 임의로 입력 금지
//채팅창의 메시지입력과[보내기]버튼 생성및 배치
JPanel mess = new JPanel();
mess.add(new JLabel("메시지"));
//메시지 입력필드의 생성과 리스너 등록, 배치
inData = new JTextField(20);
mess.add(inData);
inData.addActionListener(this);
//[보내기]버튼의 생성과 리스너 등록, 배치
send = new JButton("보내기");
mess.add(send);
send.addActionListener(this);
//채팅창의 Component 배치
chat.add("South",mess);
displayId = new JLabel();
chat.add("North",displayId);
c.add("chat", chat);
window.show(c, "login");
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
                setSize(400, 400);
                setVisible(true);
        }
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
//ChatHandler와 메시지를 주고받은 일을하는 쓰레드 생성 후 실행시킴
public void clientStart() {
        handler = new Thread(this);
        handler.start ();
}

//쓰레드를 실행시 자동으로 실행
public void run() {
        try {
                //Socket인스턴스 생성
                Socket s = new Socket ("127.0.0.1", 5000);
                //입출력Stream 얻어냄
                input = new BufferedReader(
                                new
InputStreamReader(s.getInputStream()));
                output = new PrintWriter(
                                new
OutputStreamWriter(s.getOutputStream()));
                //쓰레드 핸들러가 보낸 메시지를  받는 execute()메서드
                execute();
        } catch (IOException ex) {
                ex.printStackTrace (System.out);
        }
}
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
//쓰레드 핸들러인 ChatHandler가 보낸 메시지를 받아서 대화화면에 표시
public void execute() {
        try {
                while(true) {//무한히 메시지를 받음
                        //받은 메시지를 대화화면에 표시
                        String line = input.readLine();
                        display.append (line + "₩n");
                }
        }catch (IOException ex) {
                ex.printStackTrace (System.out);
        }finally{//메시지 받는 것이 중단될때 수행
                stop ();
        }
}
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
//서버와의 연결을 중단시 사용하는 메서드
public void stop () {
        if (handler != null) {
                try {
                        if(output != null) {
                                input.close();
                                output.close();
                        }
                }catch (IOException ioe) {
                        ioe.printStackTrace();
                }
        }
        //사용자 쓰레드 제거
        handler = null;
}
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
//사용자의 아이디 처리와 쓰레드 핸들러로 메시지를 보내는 것을 처리
public void actionPerformed (ActionEvent e) {
        //이벤트가 발생한 Component 얻어냄
        Component event = (Component) e.getSource();
        //이벤트가 발생한 Component가 아이디 입력필드이면 수행
        if(event == id) {
                //아이디를 얻어내 채팅창의 제일위에 표시
                String name = id.getText().trim();
                displayId.setText(name);
                //아이디를 입력하지 않으면 실행중단.
                if(name == null || name.length() == 0) {
                        return;
                }
                //쓰레드 핸들러인 ChatHandler로 메시지 보냄
                output.println(name);
                output.flush();
                //채팅창이 표시되도록함
                window.show(c, "chat");
                inData.requestFocus();
                //이벤트의 발생이 메시지 입력 필드나 [보내기]버튼이면 수행
        }
```

Multi Chatting

❖ **클라이언트 클래스(ChatClient.java)**

```
                else if(event == inData || event == send) {
                        //쓰레드 핸들러인 ChatHandler로 메시지 보냄
                        output.println(inData.getText());
                        output.flush();
                        inData.setText("");
                }
        }
}
```

Multi Chatting

❖ **클라이언트 메인 클래스(ChatClientMain.java)**

```
public class ChatClientMain {
        public static void main(String[] args) {
                GUIChatClient client  = new GUIChatClient();
                Thread th = new Thread(client);
                th.start();
        }
}
```

Multi Chatting

❖ **서버 클래스(ChatServer.java)**

```
//서버: ChatServer클래스
import java.io.EOFException;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Vector;

public class GUIChatServer {
        //클라이언트를 저장하기 위한 벡터, 클라이언트는 쓰레드인 ChatHandler인스턴스로 저장
        Vector<ChatHandler> handlers;

        public GUIChatServer (int port) {
                try {
                                //서버 Socket 인스턴스생성 , 사용자는 최대 50까지 받을 수 있음
                                ServerSocket server = new ServerSocket (port);
                //클라이언트를 관리하는 벡터생성
                                handlers = new Vector<ChatHandler>();

                                System.out.println("채팅 서버 준비완료");
```

Multi Chatting

❖ **서버 클래스(ChatServer.java)**

```
                                        //무한히 클라이언트를 받을 수 있도록 무한루프로 처리
                                        while (true) {
                                                //Socket인스턴스 얻어냄
                    Socket client = server.accept ();
                    System.out.println ("접속한 클라이언트 IP: " + client.getInetAddress());
                    //클라이언트당 1개씩 ChatHandler인스턴스 생성
                    ChatHandler ch = new ChatHandler(this, client);
                    //클라이언트 관리 벡터에 접속한 클라이언트 추가
                    handlers.addElement(ch);
                    //ChatHandler클래스의 run()메서드가 호출됨
                    ch.start ();
                }
            }catch(EOFException eofe){
                eofe.printStackTrace();
            }catch(IOException ioe){
                ioe.printStackTrace();
            }
        }

        public static void main(String[] args) {
                new GUIChatServer(5000);
        }
}
```