



java 18강 - AWT, Event

양 명 속

[now4ever7@gmail.com]



목차

- event 처리



이벤트 처리(Event Handling)



이벤트(Event)란

■ 이벤트(Event)란

- 사용자나 프로그램 코드에 의해서 발생할 수 있는 사건
 - 예) 사용자가 마우스를 움직이거나 클릭할 때,
 - 키보드를 눌렀을 때, Frame의 크기를 변경할 때 이벤트 발생

종류	설명
이벤트 소스 (Event Source, 이벤트 발생지)	이벤트가 발생한 컴포넌트 . 사용자가 Button을 눌렀을 때 이벤트가 발생하고, Button은 이 이벤트의 이벤트 소스가 됨
이벤트 핸들러 (Event Handler, 이벤트 처리기)	이벤트가 발생했을 때 실행될 코드를 구현해 놓은 클래스
이벤트 리스너 (Event Listener, 이벤트 감지기)	이벤트를 감지하고 처리함. 이벤트 핸들러를 이벤트 리스너로 이벤트 소스에 연결 해야 이벤트가 발생했을 때 이벤트가 처리됨
이벤트(Event)	ActionEvent, MouseEvent, KeyEvent



이벤트(Event)란

- 이벤트 처리(Event Handling)
 - 이벤트에 대한 수행코드를 작성하여 이벤트소스에 이벤트 리스너(Event Listener)로 등록하는 것
 - AWT 프로그램 실행 중에 사용자의 어떠한 동작에 의해서 이벤트가 발생했을 때, 이에 대해 적절한 작업이 수행되도록 하는 것
 - 이벤트가 발생하더라도 그에 대한 이벤트 처리를 하지 않으면, 아무 일도 일어나지 않음

EventHandler 클래스는 WindowListener 인터페이스를 구현해야 하므로 WindowListener 인터페이스에 정의되어 있는 모든 추상 메서드의 몸통(body)부분을 만들어 주어야 함

이벤트 처리 방법

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

Frame의 닫기 버튼을 눌렀을 때 Frame이 닫히도록 하기

```
class FrameTest3 {
    public static void main(String args[]) {
        Frame f = new Frame("Login"); // Frame객체를 생성한다.
        f.setSize(300, 200);           // Frame의 크기를 설정한다.

        // EventHandler클래스의 객체를 생성해서 Frame의 WindowListener로 등록한다.
        f.addWindowListener(new EventHandler());
        f.setVisible(true);             // 생성한 Frame을 화면에 보이도록 한다.
    }
}

class EventHandler implements WindowListener{
    public void windowOpened(WindowEvent e) {}
    public void windowClosing(WindowEvent e) { // Frame의 닫기 버튼을 눌렀을 때 호출된다.
        e.getWindow().setVisible(false);      // Frame을 화면에서 보이지 않도록 하고
        e.getWindow().dispose();              // 메모리에서 제거한다.
        System.exit(0);                       // 프로그램을 종료한다.
    }
    public void windowClosed(WindowEvent e) {} // 아무내용도 없는 메서드 구현
    public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
    public void windowActivated(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
}
```



이벤트 처리 방법

- 이벤트 처리 방법

- 1. 이벤트 메서드 중에서 필요한 것을 찾는다

```
windowClosing(WindowEvent e)
```

- 2. 선택한 메서드가 속해있는 인터페이스를 구현하는 클래스를 작성한다

```
class EventHandler implements WindowListener
{
    public void windowClosing(WindowEvent e) {코드작성}
    ...
}
```

- 3. 위에서 구현한 클래스의 인스턴스를 생성해서 이벤트 소스에 Listener로 등록한다

```
f.addWindowListener(new EventHandler());
```

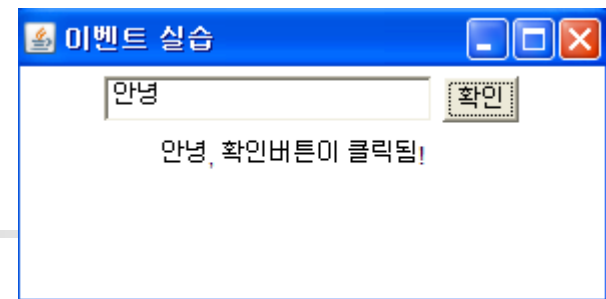
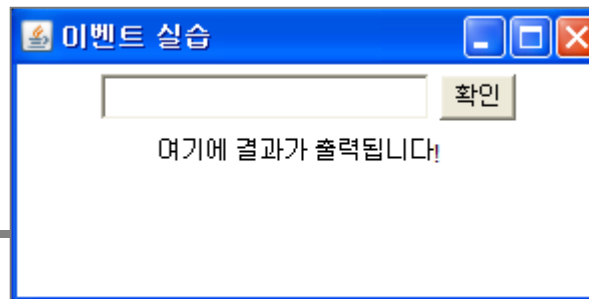


이벤트 처리 방법

- Frame의 닫기 버튼을 눌렀을 때, 그 이후의 진행 과정

- 사용자가 Frame의 닫기 버튼을 누르면,
 1. WindowEvent가 발생하고 (WindowEvent의 인스턴스가 생성됨)
 2. Frame에 WindowListener 로 등록되어 있는 이벤트 핸들러의 windowClosing 메서드를 호출한다
 - 이 메서드 내에서는 이벤트 발생시 생성된 WindowEvent 인스턴스의 참조를 사용할 수 있어서 WindowEvent 인스턴스의 메서드들을 사용할 수 있다

예제



```
import java.awt.*;
import java.awt.event.*;
//1. 이벤트소스: Button
//2. 이벤트: ActionEvent
//3. 이벤트 핸들러: ~Listener를 상속받는 클래스 => ActionListener
public class EventTest extends Frame
{
    Button bt;
    TextField tf;
    Label lb;
    public EventTest(){
        super("이벤트 실습");
        setLayout(new FlowLayout());
        tf = new TextField(20);
        bt=new Button("확인");
        lb=new Label("여기에 결과가 출력됩니다!");

        add(tf);add(bt); add(lb);
        //리스너 부착
        bt.addActionListener(new EventHandler());
    }
}
```



예제

```
class EventHandler implements ActionListener
{
    //추상메소드 오버라이딩-----
    public void actionPerformed(ActionEvent e){
        //이벤트 처리 코드
        String cmd=e.getActionCommand();//이벤트소스[버튼]의 라벨(문자열)을 돌려
준다.
        lb.setText(tf.getText() + ", "+cmd + "버튼이 클릭됨!");
    }
}

public static void main(String args[]) {
    EventTest0 f = new EventTest0();
    f.setSize(300, 150);
    f.setVisible(true);
}
}
}
```

Event의 종류와 관련 인터페이스

이벤트	인터페이스	메서드
ActionEvent	ActionListener	actionPerformed(ActionEvent e)
ComponentEvent	ComponentListener	componentMoved(ComponentEvent e)
		componentHidden(ComponentEvent e)
		componentResized(ComponentEvent e)
		componentShown(ComponentEvent e)
MouseEvent	MouseMotionListener	mouseDragged(MouseEvent e)
		mouseMoved(MouseEvent e)
	MouseListener	mousePressed(MouseEvent e)
		mouseReleased(MouseEvent e)
		mouseEntered(MouseEvent e)
		mouseExited(MouseEvent e)
		mouseClicked(MouseEvent e)



Event의 종류와 관련 인터페이스

이벤트	인터페이스	메서드
MouseEvent	MouseListener	mouseWheelMoved(MouseEvent e)
KeyEvent	KeyListener	keyPressed(KeyEvent e)
		keyReleased(KeyEvent e)
		keyTyped(KeyEvent e)
TextEvent	TextListener	textValueChanged(TextEvent e)
FocusEvent	FocusListener	focusGained(FocusEvent e)
		focusLost(FocusEvent e)
ItemEvent	ItemListener	itemStateChanged(ItemEvent e)
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)
ContainerEvent	ContainerListener	componentAdded(ContainerEvent e)
		componentRemoved(ContainerEvent e)



Event의 종류와 관련 인터페이스

이벤트	인터페이스	메서드
WindowEvent	WindowListener	<code>windowClosing(WindowEvent e)</code>
		<code>windowOpened(WindowEvent e)</code>
		<code>windowIconified(WindowEvent e)</code>
		<code>windowDeiconified(WindowEvent e)</code>
		<code>windowClosed(WindowEvent e)</code>
		<code>windowActivated(WindowEvent e)</code>
		<code>windowDeactivated(WindowEvent e)</code>
	WindowFocusListener	<code>windowGainedFocus(WindowEvent e)</code>
		<code>windowLostFocus(WindowEvent e)</code>
	WindowStateListener	<code>windowStateChanged(WindowEvent e)</code>

EventListener를 Component에 추가/제거하는 메서드

메서드	호출 시기
<code>actionPerformed(ActionEvent e)</code>	Button을 클릭했을 때, Menu를 클릭했을 때, TextField에서 Enter키를 눌렀을 때, List의 item 하나를 선택하여 더블클릭했을 때
<code>componentMoved(ComponentEvent e)</code>	컴포넌트가 이동되었을 때
<code>componentHidden(ComponentEvent e)</code>	컴포넌트가 화면에 보이지 않게 되었을 때
<code>componentResized(ComponentEvent e)</code>	컴포넌트의 크기가 변경되었을 때
<code>componentShown(ComponentEvent e)</code>	컴포넌트가 화면에 보여질 때
<code>mouseDragged(MouseEvent e)</code>	마우스 버튼을 누른 채로 마우스를 움직였을 때
<code>mouseMoved(MouseEvent e)</code>	마우스 포인터를 이동시킬 때
<code>mousePressed(MouseEvent e)</code>	마우스 버튼을 눌렀을 때
<code>mouseReleased(MouseEvent e)</code>	마우스 버튼을 떼었을 때
<code>mouseEntered(MouseEvent e)</code>	마우스 포인터가 이벤트 소스의 영역 안으로 들어왔을 때
<code>mouseExited(MouseEvent e)</code>	마우스 포인터가 이벤트 소스의 영역 안에서 밖으로 이동할 때
<code>mouseClicked(MouseEvent e)</code>	마우스 버튼을 눌렀다가 떼었을 때

EventListener를 Component에 추가/제거하는 메서드

메서드	호출 시기
<code>mouseWheelMoved(MouseWheelEvent e)</code>	마우스의 휠을 움직였을 때
<code>keyPressed(KeyEvent e)</code>	키보드의 키를 눌렀을 때
<code>keyReleased(KeyEvent e)</code>	키보드의 키를 떼었을 때
<code>keyTyped(KeyEvent e)</code>	키보드의 키를 눌렀다 떼었을 때
<code>textValueChanged(TextEvent e)</code>	TextField나 TextArea의 내용이 변경되었을 때
<code>focusGained(FocusEvent e)</code>	이벤트 소스로 focus가 이동했을 때
<code>focusLost(FocusEvent e)</code>	이벤트 소스가 갖고 있던 focus가 다른 컴포넌트로 이동했을 때
<code>itemStateChanged(ItemEvent e)</code>	Checkbox, CheckboxMenuItem, List, Choice의 status가 변경되었을 때 (selected <-> unselected)
<code>adjustmentValueChanged(AdjustmentEvent e)</code>	Scrollbar의 값이 변경되었을 때
<code>componentAdded(ContainerEvent e)</code>	컨테이너에 컴포넌트가 추가되었을 때
<code>componentRemoved(ContainerEvent e)</code>	컨테이너에 컴포넌트가 제거되었을 때

EventListener를 Component에 추가/제거하는 메서드

메서드	호출 시기
windowOpened(WindowEvent e)	윈도우가 열렸을 때
windowClosing(WindowEvent e)	윈도우가 닫힐 때(닫기 버튼을 눌렀을 때)
windowClosed(WindowEvent e)	윈도우가 닫혔을 때 (dispose())가 호출되었을 때)
windowIconified(WindowEvent e)	윈도우가 최소화(아이콘화)되었을 때
windowDeiconified(WindowEvent e)	윈도우가 최소화상태에서 다시 원래 크기로 되었을 때
windowActivated(WindowEvent e)	윈도우가 활성화되었을 때
windowDeactivated(WindowEvent e)	윈도우가 비활성화되었을 때
windowGainedFocus(WindowEvent e)	윈도우가 포커스를 얻을 때
windowLostFocus(WindowEvent e)	윈도우가 포커스를 잃었을 때
windowStateChanged(WindowEvent e)	윈도우의 상태가 변했을 때

윈도우 => 컨테이너인 Window와 그 자손들(Frame, Dialog, FileDialog)을 뜻함



ActionEvent

■ ActionEvent

- 고수준 이벤트로 컴포넌트에 정의된 **동작이 수행되었을 때 발생함**
- 예) 버튼을 마우스로 클릭하거나, 마우스가 포커스를 가졌을 때 스페이스바를 입력하면 눌러짐
 - 이 때 MouseEvent나 KeyEvent 가 발생하는 것이 아니라 ActionEvent가 발생함
 - 실제로는 MouseEvent나 KeyEvent 가 발생하지만 최종적으로는 ActionEvent가 발생한 것으로 처리됨
 - 그래서 ActionListener의 actionPerformed(ActionEvent ae)에 버튼이 눌러졌을 때 수행될 코드를 작성해야 함

각 Event의 주요 메서드

메서드	호출 시기
ItemEvent	
Object <code>getItem()</code>	이벤트가 발생한 item을 반환함
ItemSelectable <code>getItemSelectable()</code>	이벤트 소스를 반환함
int <code>getStateChange()</code>	item의 상태(선택/선택해제상태)를 반환함
KeyEvent	
char <code>getKeyChar()</code>	눌러진 키를 반환
int <code>getKeyCode()</code>	눌러진 키의 코드를 반환
static String <code>getKeyModifiersText(int modifiers)</code>	눌러진 특수키(Alt, Ctrl)나 특수키 조합을 설명하는 문자열을 반환
static String <code>getKeyText(int keyCode)</code>	눌러진 키를 HOME, F1과 같이 키를 설명하는 문자열을 반환
MouseEvent	
int <code>getClickCount()</code>	마우스를 클릭한 회수를 반환
Point <code>getPoint()</code>	이벤트가 일어난 위치를 반환
int <code>getX()</code>	이벤트가 일어난 위치의 x좌표를 반환
int <code>getY()</code>	이벤트가 일어난 위치의 y좌표를 반환
모든 이벤트 공통	
Object <code>getSource()</code>	이벤트 소스를 반환함(EventObject에 정의)



이벤트 리스너를 컴포넌트에 추가/제거

- 이벤트 소스(컴포넌트)에 Listener를 추가 또는 제거하는 메서드
 - Listener의 이름에 add 또는 remove 를 붙여서 만들어 줌
 - ActionListener
 - void **addActionListener**(ActionListener l)
 - void **removeActionListener**(ActionListener l)



리스너를 추가할 수 있는 컴포넌트

Listener	이벤트 소스
ActionListener	Button, List, MenuItem, TextField
AdjustmentListener	Scrollbar
ComponentListener	Component
ContainerListener	Container
FocusListener	Component
ItemListener	Checkbox, CheckboxMenuItem, List, Choice
KeyListener	Component
MouseListener	Component
MouseMotionListener	Component
TextListener	TextField, TextArea
WindowListener	Window
WindowFocusListener	Window
WindowStateListener	Window



Adapter 클래스

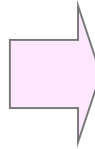
■ Adapter 클래스

- 이벤트 핸들러를 작성할 때, 해당 이벤트 리스너에 정의된 모든 추상 메서드를 구현해야 한다는 불편한 점을 없애기 위해 고안된 것
- 이벤트 처리에 필요한 메서드만 작성하면 됨
- 이벤트 리스너를 직접 구현하는 대신에 Adapter 클래스를 상속받아서 원하는 메서드만 작성(오버라이딩)하면 됨
- Adapter 클래스는 이벤트 리스너에 정의된 메서드를 아무 내용도 없이 구현해 놓았음

Adapter 클래스

이벤트 핸들러를 작성할 때 이벤트 리스너를 구현한 것

```
class KeyHandler implements KeyListener
{
    public void keyPressed(KeyEvent e)
    {
        System.out.println(e.getKeyChar());
    }
    public void keyReleased(KeyEvent e){}
    public void keyTyped(KeyEvent e){}
}
```



Adapter 클래스를 상속받은 것

```
class KeyHandler extends KeyAdapter
{
    public void keyPressed(KeyEvent e)
    {
        System.out.println(e.getKeyChar());
    }
}
```

```
interface KeyListener
{
    public void keyPressed(KeyEvent e);
    public void keyReleased(KeyEvent e);
    public void keyTyped(KeyEvent e);
}
class KeyAdapter implements KeyListener
{
    public void keyPressed(KeyEvent e){}
    public void keyReleased(KeyEvent e){}
    public void keyTyped(KeyEvent e){}
}
```

Adapter클래스	이벤트 리스너(interface)
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
FocusAdapter	FocusListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
WindowAdapter	WindowListener



예제-Adapter 클래스 이용

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

public class AdapterTest1 extends Frame{
    public AdapterTest1(){
        super("Adapter Test");

        addWindowListener(new EventHandler());

        /*익명 클래스 이용
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e) {
                e.getWindow().setVisible(false);
                e.getWindow().dispose();
                System.exit(0);
            }
        });*/
    }
}
```



예제-Adapter 클래스 이용

// 내부 클래스 이용

```
class EventHandler extends WindowAdapter
{
    public void windowClosing(WindowEvent e) {
        e.getWindow().setVisible(false);
        e.getWindow().dispose();
        System.exit(0);
    }
}
```

```
public static void main(String args[]) {
    AdapterTest1 f = new AdapterTest1();
    f.setSize(300, 200);
    f.setVisible(true);
}
}
```


예제

```
import java.awt.*;
import java.awt.event.*;
class TextFieldTest2 extends Frame {
    Label lid;
    Label lpwd;
    TextField tfld;
    TextField tfPwd;
    Button ok;
```

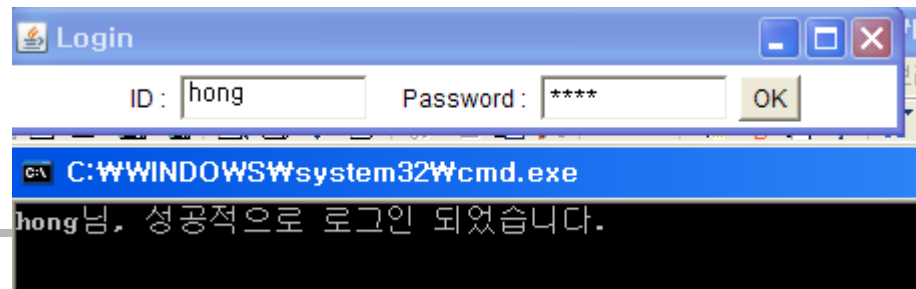
```
    TextFieldTest2(String title) {
        super(title); // Frame(String title)을 호출한다.
```

```
        lid = new Label("ID :", Label.RIGHT); // Label의 text정렬을 오른쪽으로.
        lpwd = new Label("Password :", Label.RIGHT);
```

```
        // 약 10개의 글자를 입력할 수 있는 TextField 생성.
        tfld = new TextField(10);
        tfPwd = new TextField(10);
        tfPwd.setEchoChar('*'); // 입력한 값 대신 '*'이 보이게 한다.
```

```
        ok = new Button("OK");
        // OK버튼과 TextField에 이벤트처리를 위한 Listener를 추가해준다.
        tfld.addActionListener(new EventHandler());
        tfPwd.addActionListener(new EventHandler());
        ok.addActionListener(new EventHandler());
```

버튼 클릭하거나 TextField에서 Enter 키를 눌러도 ActionEvent 처리



예제

```
setLayout(new FlowLayout()); // LayoutManager를 FlowLayout으로
add(lid); // 생성한 Component들을 Frame에 포함시킨다.
add(tfld);
add(lpwd);
add(tfPwd);
add(ok);
setSize(450, 65);
setVisible(true); // Frame이 화면에 보이게 한다.
}

public static void main(String args[]) {
    TextFieldTest2 f = new TextFieldTest2("Login");
}

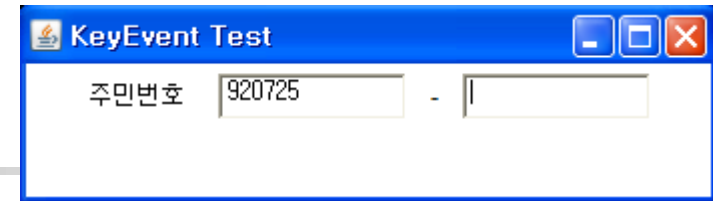
class EventHandler implements ActionListener {
    public void actionPerformed(ActionEvent e){
        String id = tfld.getText(); // tfld에 입력되어있는 text를 얻어온다.
        String password = tfPwd.getText();
        if (!id.equals("hong")){
            System.out.println("입력하신 id가 유효하지 않습니다. 다시 입력해
주세요.");
            // id를 다시 입력할 수 있도록, focus를 tfld로 옮긴다.
            tfld.requestFocus();
            tfld.selectAll(); // tfld에 입력된 text가 선택되게 한다.
        } else if (!password.equals("1234")) {
            System.out.println("입력하신 비밀번호가 틀렸습니다. 다시 입력해 주
시기 바랍니다.");
        }
    }
}
```



예제

```
tfPwd.requestFocus();
tfPwd.selectAll();
    } else {
        System.out.println( id + "님, 성공적으로 로그인 되었습니다.");
    }
}
} // class EventHandler
}
```

예제 1



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

```
class KeyEventTest extends Frame implements KeyListener {
    private FlowLayout fl = new FlowLayout();
    private Label lb1 = new Label("주민번호");
    private TextField tf = new TextField(10);
    private Label lb = new Label("-", Label.CENTER);
    private TextField tf1 = new TextField(10);

    public KeyEventTest() {
        super("KeyEvent Test");
        this.init();
        this.start();
        this.setSize(350, 100);
        this.setVisible(true);
    }

    public void init() {
        this.setLayout(fl);
        this.add(lb1);
        this.add(tf);
        this.add(lb);
        this.add(tf1);
    }
}
```



예제 1

```
public void start() {  
    tf.addKeyListener(this);  
}  
  
public void keyPressed(KeyEvent e) {  
  
    public void keyReleased(KeyEvent e) {  
        String str = tf.getText().trim();  
        if (str.length() == 6) {  
            tf1.requestFocus();  
        }  
    }  
  
    public void keyTyped(KeyEvent e) {  
  
    public static void main(String[] ar) {  
        KeyEventTest es = new KeyEventTest();  
    }  
}
```

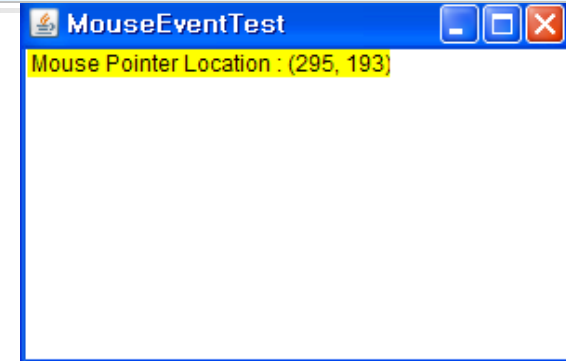
예제2

- Frame의 위에서 **마우스 포인터를 움직일 때**, MouseEvent가 발생하며 MouseEvent의 **mouseMoved** 메서드가 호출되며 이벤트 소스는 **Frame**
- MouseEvent에는 **getX(), getY()**가 있는데, 마우스 포인터의 좌표를 알 수 있음

```
import java.awt.*;
import java.awt.event.*;
class MouseEventTest extends Frame {
    Label location;
    MouseEventTest(String title) {
        super(title); // Frame(String title)을 호출한다.
        location = new Label("Mouse Pointer Location : ");
        location.setSize(195, 15);
        location.setLocation(5,30);
        location.setBackground(Color.yellow); // Label의 배경색을 노란색으로 한다.
        add(location);

        // EventHandler의 인스턴스를 Frame의 Listener로 등록한다.
        addMouseMotionListener(new EventHandler());

        setSize(300, 200);
        setLayout(null);
        setVisible(true);
    }
    public static void main(String args[]) {
        MouseEventTest mainWin = new MouseEventTest("MouseEventTest");
    } // main메서드의 끝
    class EventHandler implements MouseMotionListener {
        public void mouseDragged(MouseEvent e) {}
        public void mouseMoved(MouseEvent e) {
            location.setText("Mouse Pointer Location : (" + e.getX() + ", " + e.getY()+ ")");
        }
    } // EventHandler클래스의 끝
}
```



예제 3

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

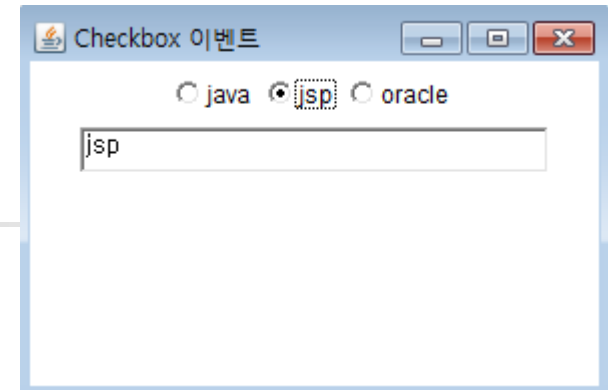
```
class Test extends Frame {  
    CheckboxGroup group;  
    Checkbox cb1;  
    Checkbox cb2;  
    Checkbox cb3;  
    TextField tf;
```

```
    Test(String title) {  
        super(title);
```

```
        group = new CheckboxGroup();  
        cb1 = new Checkbox("java", group, false);  
        cb2 = new Checkbox("jsp", group, false);  
        cb3 = new Checkbox("oracle", group, false);
```

```
        tf = new TextField(30);
```

```
        cb1.addItemListener(new EventHandler());  
        cb2.addItemListener(new EventHandler());  
        cb3.addItemListener(new EventHandler());
```





예제3

```
        setLayout(new FlowLayout());
        add(cb1);
        add(cb2);
        add(cb3);
        add(tf);

        setSize(300, 200);
        setVisible(true);
    }

    public static void main(String args[]) {
        Test mainWin = new Test("Checkbox 이벤트");
    }    // main메서드의 끝

    class EventHandler implements ItemListener {
        public void itemStateChanged(ItemEvent e) {
            Checkbox cb = (Checkbox)e.getSource();
            String str = cb.getLabel();

            tf.setText(str);
        }
    } // class EventHandler
}
```


예제 4

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

```
class ChoiceEventTest extends Frame implements ItemListener {
```

```
    FlowLayout fl;
```

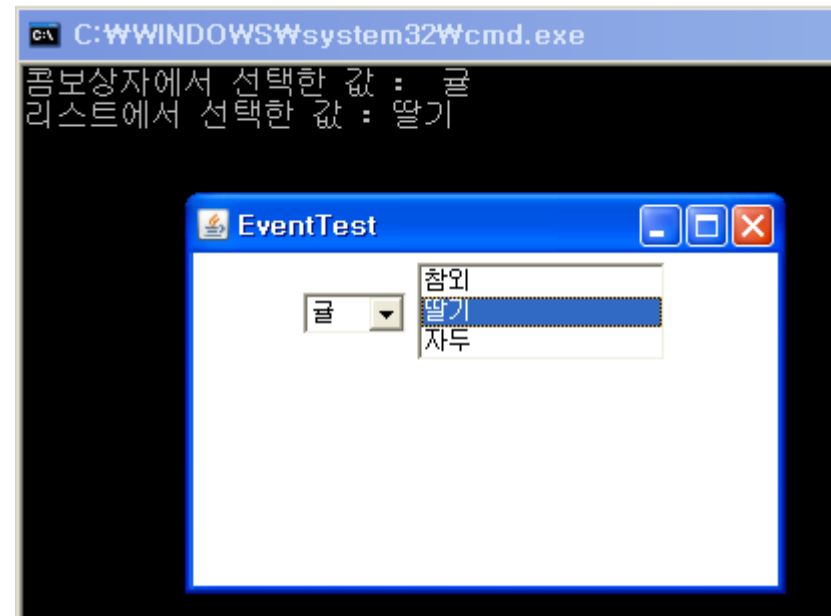
```
    Choice ch;
```

```
    List li;
```

```
    public ChoiceEventTest() {  
        super("EventTest");  
        this.init();  
        this.start();  
        this.setSize(300, 200);  
        this.setVisible(true);  
    }
```

```
    public void init() {  
        fl = new FlowLayout();  
        ch = new Choice();  
        li = new List(3, false);
```

```
        this.setLayout(fl);  
        ch.add("사과");  
        ch.add("귤");  
        ch.add("포도");  
        this.add(ch);
```





예제 4

```
        li.add("참외");
        li.add("딸기");
        li.add("자두");
        this.add(li);
    }

    public void start() {
        ch.addItemListener(this);
        li.addItemListener(this);
    }

    public void itemStateChanged(ItemEvent e) {
        if (e.getSource() == ch) {
            String str = ch.getSelectedItem();
            System.out.println("콤보상자에서 선택한 값 : " + str);
        } else if (e.getSource() == li) {
            String str = li.getSelectedItem();
            System.out.println("리스트에서 선택한 값 : " + str);
        }
    }

    public static void main(String[] ar) {
        ChoiceEventTest f = new ChoiceEventTest();
    }
}
```

예제5

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

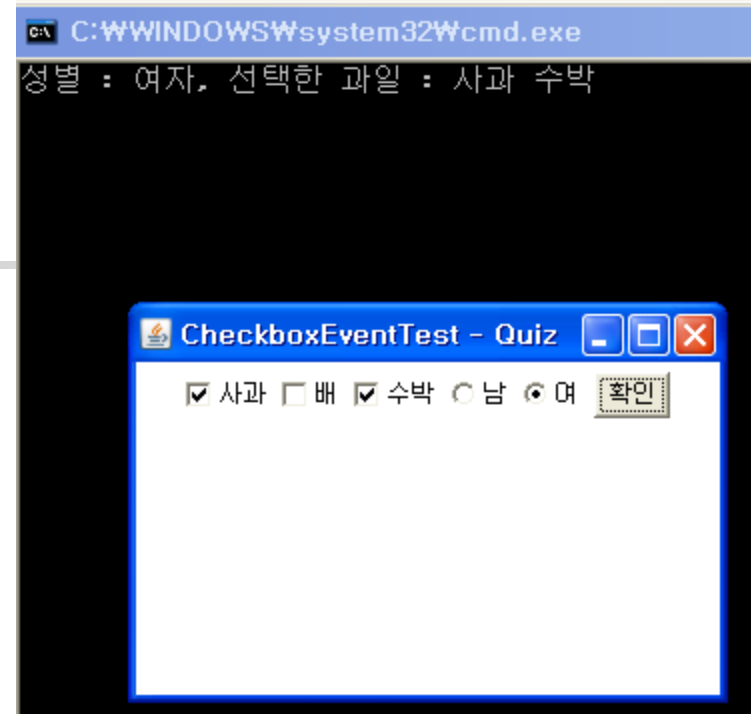
class CheckboxEventTest1 extends Frame {
    Checkbox cb1, cb2, cb3;
    Checkbox cbM, cbF;
    CheckboxGroup group;
    Button bt;

    CheckboxEventTest1(String title) {
        super(title);
        setSize(300, 200);
        setLayout(new FlowLayout());

        cb1 = new Checkbox("사과");
        cb2 = new Checkbox("배");
        cb3 = new Checkbox("수박");

        group = new CheckboxGroup();
        cbM = new Checkbox("남", group, true);
        cbF = new Checkbox("여", group, false);

        bt = new Button("확인");
        bt.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```



예제5

- Checkbox의 `getState()` 를 이용해서 Checkbox의 상태 (선택되었는지, 선택되지 않았는지) 을 알 수 있음

```
String str="";
if(cbM.getState()){
    str = "남자";
}
else if (cbF.getState()){
    str = "여자";
}

String str2="";
if(cb1.getState())    str2 += cb1.getLabel() + " ";
if(cb2.getState())    str2 += cb2.getLabel() + " ";
if(cb3.getState())    str2 += cb3.getLabel() + " ";
System.out.println("성별 : " + str + ", 선택한 과일 : " + str2);
    }
});

add(cb1); add(cb2); add(cb3);
add(cbM); add(cbF);
add(bt); //add(lb);

setVisible(true);
}

public static void main(String args[]) {
    CheckboxEventTest1 mainWin = new CheckboxEventTest1("CheckboxEventTest - Quiz");
}
}
```

예제 6

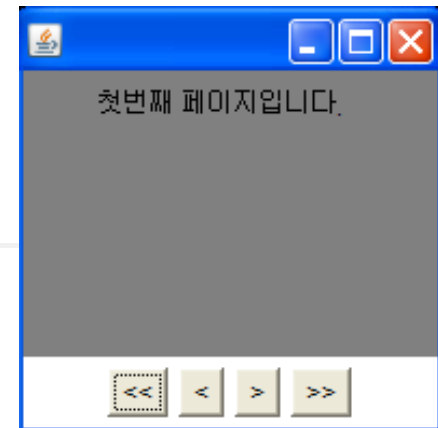
```
import java.awt.*;
import java.awt.event.*;

class CardLayoutEventTest extends Frame {
    Button first, prev, next, last;
    Panel buttons;
    Panel slide;
    Panel card1, card2, card3, card4, card5; // slide에 포함될 Panel들
    CardLayout cardLayout;

    CardLayoutEventTest(String title) {
        // 화면을 담을 Panel을 담는다.
        slide = new Panel();
        cardLayout = new CardLayout();
        slide.setLayout(cardLayout);

        // 버튼을 담을 Panel을 만든다.
        buttons = new Panel();
        buttons.setLayout(new FlowLayout());

        first = new Button("<<");
        prev = new Button("<");
        next = new Button(">");
        last = new Button(">>");
        buttons.add(first);
        buttons.add(prev);
        buttons.add(next);
        buttons.add(last);
    }
}
```



- **CardLayout** 을 이용해서 5개의 **Panel**을 슬라이드 형식으로 화면에 보여주는 예제
- **Frame**의 **Center**에 **CardLayout**이 레이아웃 매니저인 **slide** 를 넣고, **South** 에는 버튼이 담긴 **Panel**을 넣기
- **slide** 에 5개의 **Panel**이 추가, 버튼에 의해 5개의 **Panel** 중 하나가 보여짐



예제 6

```
// 버튼에 이벤트 리스너를 추가한다.
first.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        // CardLayout의 첫번째 slide(Panel)이 보여지도록 한다.
        cardLayout.first(slide);
    }
});
prev.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        // 현재 slide의 이전 slide(Panel)이 보여지도록 한다.
        cardLayout.previous(slide);
    }
});
next.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        // 현재 slide의 다음 slide(Panel)이 보여지도록 한다.
        cardLayout.next(slide);
    }
});
last.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        // CardLayout의 마지막 slide(Panel)이 보여지도록 한다.
        cardLayout.last(slide);
    }
});
card1= new Panel();
card1.setBackground(Color.gray);
card1.add(new Label("첫번째 페이지입니다.));
card2= new Panel();
card2.add(new Label("두번째 페이지입니다.));
card2.setBackground(Color.orange);
```



예제 6

```
card3= new Panel();
card3.add(new Label("세번째 페이지입니다.));
card3.setBackground(Color.blue);
card4= new Panel();
card4.add(new Label("네번째 페이지입니다.));
card4.setBackground(Color.cyan);
card5= new Panel();
card5.add(new Label("다섯번째 페이지입니다.));
card5.setBackground(Color.pink);
```

```
// slide(Panel)에 card1(Panel)을 "1"이란 이름으로 추가한다.
slide.add(card1, "1");
slide.add(card2, "2");
slide.add(card3, "3");
slide.add(card4, "4");
slide.add(card5, "5");
```

```
add("South", buttons);
add("Center", slide);
```

```
setSize(200, 200);
setLocation(200, 200);
setVisible(true);
```

• slide(Panel)에 추가된 Component 중 이름이 "1" 인 것을 보여줌

```
cardLayout.show(slide, "1"); // 첫번째 슬라이드(card1)가 나타나게 한다.
```

```
}
public static void main(String args[]) {
    CardLayoutEventTest mainWin = new CardLayoutEventTest("CardLayoutTest");
}
```

예제 7

```
import java.awt.*;
import java.awt.event.*;
class ChatWin extends Frame {
    String nickname = "";

    TextArea ta = new TextArea();
    Panel p = new Panel();
    TextField tf = new TextField();

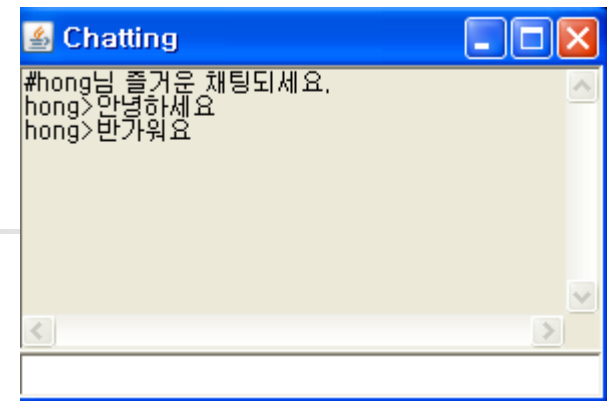
    ChatWin() {
        this("guest");
    }
    ChatWin(String nickname) {
        super("Chatting");
        this.nickname = nickname;

        setBounds(100, 100, 300, 200);

        p.setLayout(new BorderLayout());
        p.add(tf, "Center");

        add(ta, "Center");
        add(p, "South");

        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        EventHandler handler = new EventHandler();
        tf.addActionListener(handler);
    }
}
```





예제 7

```
ta.setText("#" + nickname + "님 즐거운 채팅되세요.");
ta.setEditable(false);

setVisible(true);
tf.requestFocus();
}

public static void main(String[] args) {
    if(args.length != 1) {
        System.out.println("USAGE : java ChatWin NICKNAME");
        System.exit(0);
    }

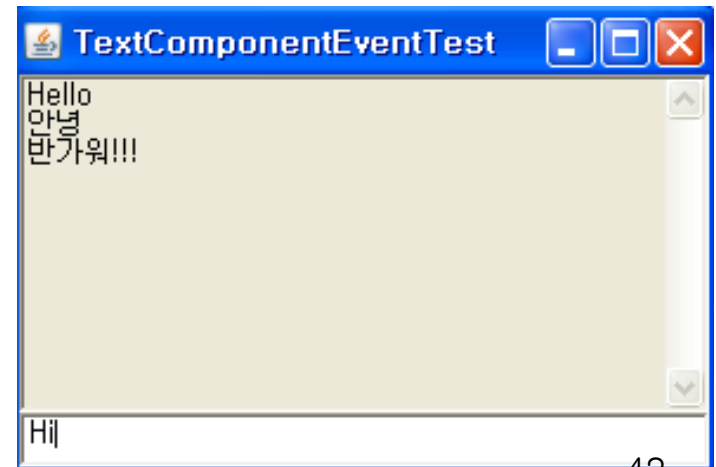
    new ChatWin(args[0]);
} // main

class EventHandler implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
        String msg = tf.getText();
        if("").equals(msg)) return;

        ta.append("WrWn" + nickname + ">" + msg);
        tf.setText("");
    }
} // class EventHandler
} // class
```

실습1

- TextField에서 Enter를 치면, `ActionEvent` 발생 - `actionPerformed()`
 - TextField에 입력된 text를 TextArea에 추가한다.
 - TextField의 text를 지우고, TextField에 포커스가 오게 한다.
- TextArea에 추가하는 메서드
 - `public void append(String str)`
- TextArea의 text를 편집하지 못하게 하는 메서드
 - `public void setEditable(boolean b)`



실습2

버튼 클릭시 `ActionEvent` 발생 - `actionPerformed()`

CheckboxEventTest - Quiz

1. 다음중 `ActionEvent`의 `actionPerformed`메서드가 호출되는 경우는? (모두 고르세요.)

- ☒ Button을 눌렀을때
- ☒ TextField에서 Enter키를 눌렀을때
- ☒ MenuItem을 클릭했을때
- ☒ List에서 더블클릭으로 item을 선택했을때

2. Frame의 기본 `LayoutManager`는? (하나만 고르세요.)

- ☐ FlowLayout
- ☐ GridLayout
- ☒ BorderLayout
- ☐ CardLayout

이 버튼을 누르시면 결과를 알 수 있습니다.

* 결과 : 당신의 점수는 100.0점 입니다.

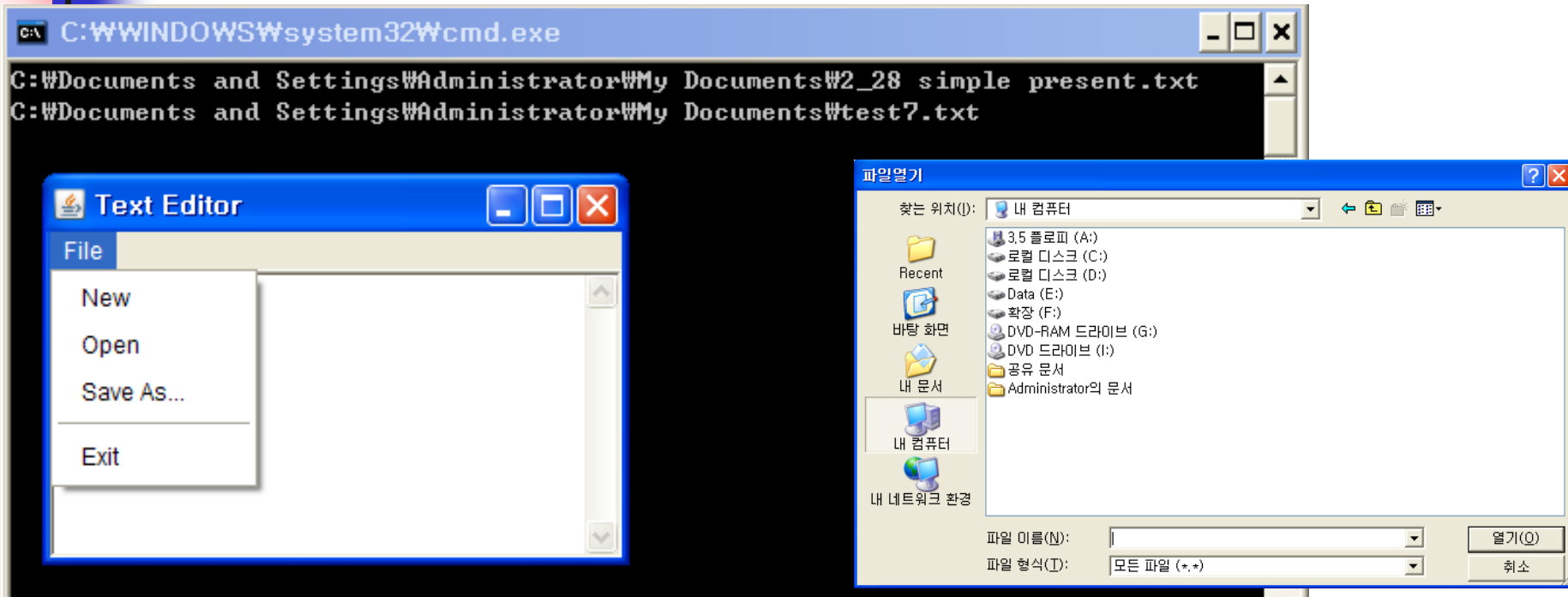
각 12.5점

50점

- 버튼 클릭시 점수를 계산하여 `Label`에 출력하기
- `Checkbox`의 `getState()` 를 이용해서 `Checkbox`의 상태 (선택되었는지, 선택되지 않았는지) 을 알 수 있음

실습3

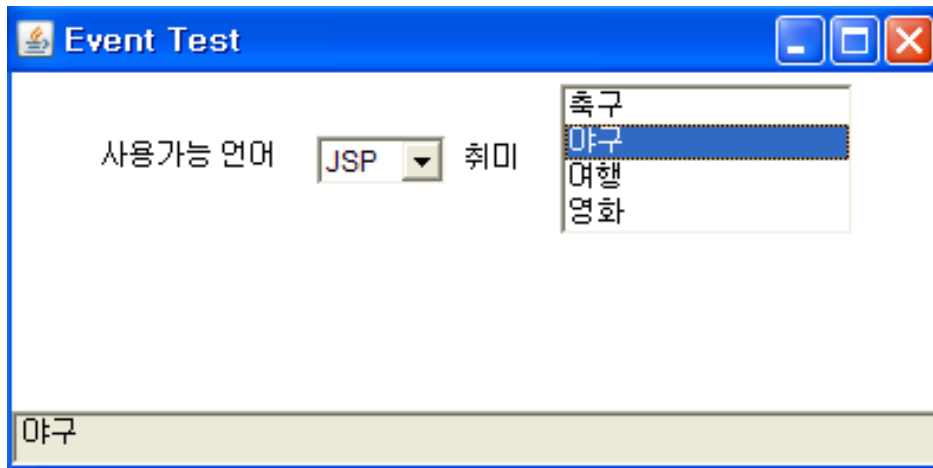
메뉴 클릭시 `ActionEvent` 발생 - `actionPerformed()`



- New 메뉴 클릭 - TextArea의 내용을 지운다
- Open - OpenFileDialog 이용하여 열기대화상자 띄우기, 선택한 파일 화면 출력
- SaveAs - OpenFileDialog 이용하여 저장 대화상자 띄우기, 저장 파일명 화면 출력
- Exit - 프로그램 종료

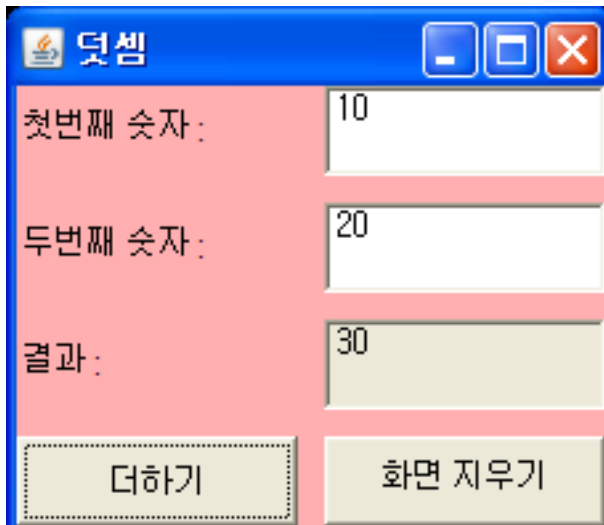
실습4

- Choice와 List에서 항목을 선택하면 TextField에 선택한 항목을 출력하기
 - TextField의 text를 편집하지 못하도록 설정



실습5

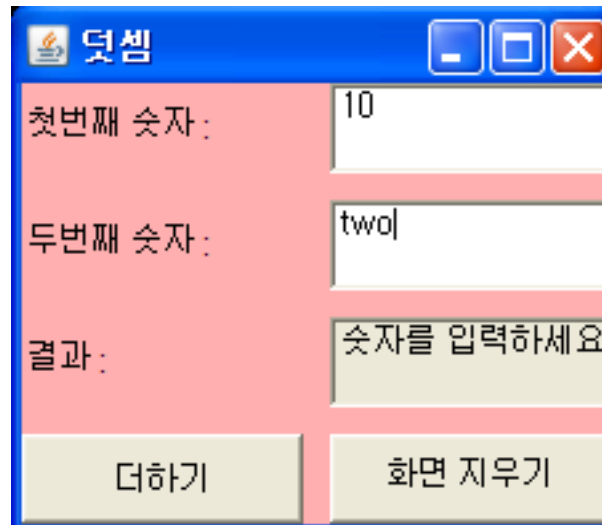
- 더하기 버튼을 클릭하면 두 수를 더한 후 결과 TextField에 보여준다
- 화면 지우기 버튼을 클릭하면 모든 TextField 의 내용을 지운다
- 숫자를 입력하지 않은 경우에는 Exception 처리를 하여 결과 TextField에 에러 메시지를 보여준다.



덧셈

첫번째 숫자 :	10
두번째 숫자 :	20
결과 :	30

더하기 화면 지우기



덧셈

첫번째 숫자 :	10
두번째 숫자 :	two
결과 :	숫자를 입력하세요

더하기 화면 지우기



이벤트 핸들러

- 이벤트 핸들러 작성 방법

- 1) 이벤트 소스를 가진 클래스가 핸들러가 되는 경우
- 2) 이너클래스로 이벤트 핸들러를 구성하는 경우
- 3) 익명 클래스로 이벤트 핸들러를 구성하는 경우
- 4) 외부 클래스로 별도의 이벤트 핸들러를 구성하는 경우

- 이벤트 핸들러 작성

- 1) ~Listener 인터페이스를 상속받아 구현
- 2) ~Adapter를 상속받아 구현



이너클래스로 이벤트 핸들러를 구성

```
import java.awt.*;
import java.awt.event.*;
class MyFrame extends Frame{
    Button bt;
    MyFrame(String title){
        super(title);
        this.setLayout(new FlowLayout());
        bt=new Button("닫기");

        this.add(bt);
        //리스너 부착
        bt.addActionListener(new EventHandler());
    }

    //내부 클래스 이용
    class EventHandler implements ActionListener{
        public void actionPerformed(ActionEvent e){
            System.exit(0);
        }
    } //EventHandler class
} //class

public class EventTest01{
    public static void main(String args[]) {
        MyFrame f = new MyFrame("이벤트 실습");
        f.setSize(200, 100);
        f.setVisible(true);
    }
} //
```




익명 클래스로 이벤트 핸들러를 구성

```
import java.awt.*;
import java.awt.event.*;
class MyFrame extends Frame{
    Button bt;
    MyFrame(String title){
        super(title);
        this.setLayout(new FlowLayout());
        bt=new Button("닫기");
        this.add(bt);

        //익명 클래스 이용
        bt.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                System.exit(0);
            }
        });
    }
}
}

public class EventTest03{
    public static void main(String args[]) {
        MyFrame f = new MyFrame("이벤트 실습");
        f.setSize(200, 100);
        f.setVisible(true);
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
class MyFrame extends Frame implements ActionListener{
    Button bt;
    MyFrame(String title){
        super(title);
        this.setLayout(new FlowLayout());
        bt=new Button("닫기");
        this.add(bt);
        //리스너 부착
        bt.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e){
        System.exit(0);
    }
}

//class

public class EventTest02{
    public static void main(String args[]) {
        MyFrame f = new MyFrame("이벤트 실습");
        f.setSize(200, 100);
        f.setVisible(true);
    }
}

//
```