이벤트 처리 간단 정리

한성대학교 컴퓨터공학부

신 성

정리

☞ 단순히 코드만 따라치는 것은 의미가 없습니다.

전체적인 큰 흐름으로 먼저 이해하시고,

이후 각 부분부분 명확히 이해하고 넘어가시면
실력 향상에 큰 도움이 됩니다.

정리

1. 버튼 등의 위젯을 클릭하는 Action 이벤트 등이 발생했을 때 실행할 코드 작성
   - ActionListener 인터페이스 안의 actionPerformed() 함수에 작성 (인터페이스 구현 클래스 작성)
     즉, 버튼을 클릭했을 때 실행할 코드가 들어있는 클래스 작성

2. 방금 전에 만든 클래스 이용해서 객체 생성

3. 방금 전에 만든 (버튼을 클릭했을 때 실행할 코드가 들어있는) 객체를
   적용하고 싶은 원하는 버튼에 등록(저장)

- 그럼 사용자가 버튼을 클릭했을 때
  방금 등록한 코드를 자동으로 실행(자동으로 actionPerformed() 함수를 호출해 줌)
- 추가적으로 함수를 호출할 때 (이벤트에 대한 정보를 담고 있는) '이벤트 객체' 전달

이벤트의 종류와 각 이벤트 소스 정리

이벤트 객체	이벤트 소스	이벤트가 발생하는 경우
ActionEvent	JButton	마우스나 <Enter> 키로 버튼 선택
	JMenuItem	메뉴 아이템 선택
	JTextField	텍스트 입력 중 <Enter> 키 입력
ItemEvent	JCheckBox	체크박스의 선택 혹은 해제
	JRadioButton	라디오 버튼의 선택 상태가 변할 때
	JCheckBoxMenuItem	체크박스 메뉴 아이템의 선택 혹은 해제
ListSelectionEvent	JList	리스트에 선택된 아이템이 변경될 때
KeyEvent	Component	키가 눌러지거나 눌러진 키가 떼어질 때
MouseEvent	Component	마우스 버튼이 눌러지거나 떼어질 때, 마우스 버튼이 클릭될 때, 컴포넌트 위에 마우스가 올라갈 때, 올라간 마우스가 내려올 때, 마우스가 드래그될 때, 마우스가 단순히 움직일 때
FocusEvent	Component	컴포넌트가 포커스를 받거나 잃을 때
TextEvent	TextField	텍스트 변경
	TextArea	텍스트 변경
WindowEvent	Window	Window를 상속받는 모든 컴포넌트에 대해 윈도우 활성화, 비활성화, 아이콘화, 아이콘에서 복구, 윈도우 열기, 윈도우 닫기, 윈도우 종료

이벤트 소스 : Component

=> Component를 상속받은 모든 스윙 컴포넌트에 대해 이벤트 발생 가능

이벤트 종류별 호출되는 메소드 정리(인터페이스 및 추상메소드)

이벤트 종류	리스너 인터페이스	리스너의 추상 메소드	메소드가 호출되는 경우
Action	ActionListener	void actionPerformed(ActionEvent)	Action 이벤트가 발생하는 경우
Item	ItemListener	void itemStateChanged(ItemEvent)	Item 이벤트가 발생하는 경우
Key	KeyListener	void keyPressed(KeyEvent)	모든 키에 대해 키가 눌러질 때
		void keyReleased(KeyEvent)	모든 키에 대해 눌러진 키가 떼어질 때
		void keyTyped(KeyEvent)	유니코드 키가 입력될 때
Mouse	MouseListener	void mousePressed(MouseEvent)	마우스 버튼이 눌러질 때
		void mouseReleased(MouseEvent)	눌러진 마우스 버튼이 떼어질 때
		void mouseClicked(MouseEvent)	마우스 버튼이 클릭될 때
		void mouseEntered(MouseEvent)	마우스가 컴포넌트 위에 올라올 때
		void mouseExited(MouseEvent)	컴포넌트 위에 올라온 마우스가 컴포넌트를 벗어날 때
Mouse	MouseMotionListener	void mouseDragged(MouseEvent)	마우스를 컴포넌트 위에서 드래그할 때
		void mouseMoved(MouseEvent)	마우스가 컴포넌트 위에서 움직일 때
Focus	FocusListener	void focusGained(FocusEvent)	컴포넌트가 포커스를 받을 때
		void focusLost(FocusEvent)	컴포넌트가 포커스를 잃을 때
Text	TextListener	void textValueChanged(TextEvent)	텍스트가 변경될 때
Window	WindowListener	void windowOpened(WindowEvent)	윈도우가 생성되어 처음으로 보이게 될 때
		void windowClosing(WindowEvent)	윈도우의 시스템 메뉴에서 윈도우 닫기를 시도할 때
		void windowIconfied(WindowEvent)	윈도우가 아이콘화될 때
		void windowDeiconfied(WindowEvent)	아이콘 상태에서 원래 상태로 복귀할 때
		void windowClosed(WindowEvent)	윈도우가 닫혔을 때
		void windowActivated(WindowEvent)	윈도우가 활성화될 때
		void windowDeactivated(WindowEvent)	윈도우가 비활성화될 때
ListSelection	ListSelectionListener	void valueChanged(ListSelectionEvent)	Jlist에 선택된 아이템이 변경될 때

이벤트 리스너 작성 방법

- **버튼을 클릭했을 때 실행되는 코드(호출되는 함수)가 들어있는 클래스 작성 방법**

1. 독립 클래스로 작성
   - 이벤트 리스너를 완전한 클래스로 작성
   - 이벤트 리스너를 여러 곳에서 사용할 때 적합

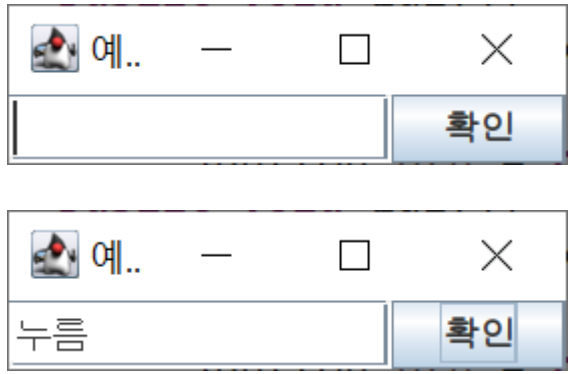
2. 내부 클래스(inner class)로 작성
   - 클래스 안에 멤버처럼 클래스 작성
   - 이벤트 리스너를 특정 클래스에서만 사용할 때 적합

3. 익명 클래스(anonymous class)로 작성
   - 클래스의 이름 없이 간단히 리스너 작성
   - 이벤트 리스너를 특정 클래스에서만 사용할 때
     클래스 조차 만들 필요 없이 리스너 코드가 간단한 경우에 적합

4. 현재 클래스에 리스너 인터페이스를 구현
   - 현재 클래스에 간단히 리스너의 추상 메소드만 구현 가능
   - 이벤트 리스너를 특정 클래스에서만 사용할 때
     클래스 조차 만들 필요 없이 리스너 코드가 간단한 경우에 적합

올려드린 ex2.txt

```
public class Test {
	JFrame mf;
	JTextField text;

	public Test() {
		mf = new JFrame();
		mf.setTitle("예제");
		mf.setSize(200, 60);
		mf.setLocation(500, 300);
		mf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		gui();

		mf.setVisible(true);
	}

	public void gui(){
		text = new JTextField();

		JButton btn = new JButton("확인");

		mf.add(text, BorderLayout.CENTER);
		mf.add(btn, BorderLayout.EAST);

		btn.addActionListener(new MyActionListener(text));
	}

	public static void main(String[] args) {
	    Test test = new Test();
	    test.gui();
	}
}
```



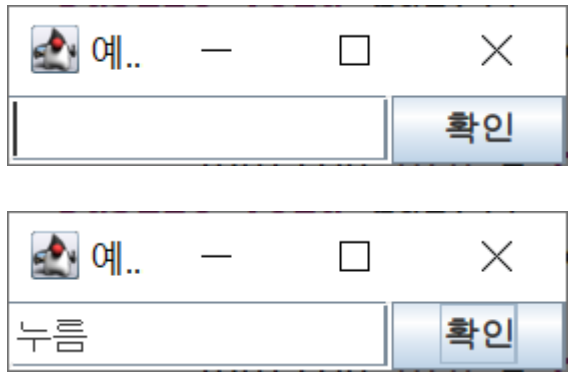
1번 방법

```
class MyActionListener implements ActionListener {

    JTextField t;

    MyActionListener(JTextField t){
        this.t=t;
    }

    public void actionPerformed(ActionEvent e) {
            t.setText("누름");
    }
}
```

올려드린 ex3.txt

```
public class Test {
	JFrame mf;
	JTextField text;

	public Test() {
		mf = new JFrame();
		mf.setTitle("예제");
		mf.setSize(200, 60);
		mf.setLocation(500, 300);
		mf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		gui();

		mf.setVisible(true);
	}

	public void gui(){
		text = new JTextField();

		JButton btn = new JButton("확인");

		mf.add(text, BorderLayout.CENTER);
		mf.add(btn, BorderLayout.EAST);

		btn.addActionListener(new MyActionListener());
	}

	public static void main(String[] args) {
	    Test test = new Test();
	    test.gui();
	}

	class MyActionListener implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			text.setText("누름");
		}
	}
}
```

내부 클래스 이용



2번 방법

[설명] 내부 클래스

한 클래스 정의 안에 정의된 또다른 클래스

- 내부 클래스의 메서드에서 외부 클래스의 멤버에 직접 접근이 가능
- 선언된 위치에 따라 변수와 동일한 유효 범위와 접근성을 가짐
- 코드의 복잡성을 줄일 수 있음

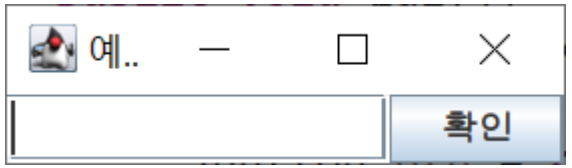
장점
- 외부 클래스 멤버를 직접 접근 가능
- 이벤트 리스너를 특정 클래스에서만 사용할 때 적합

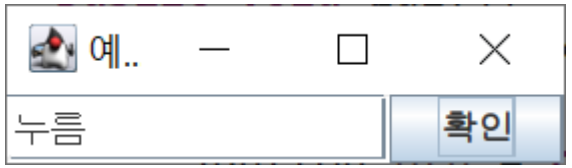
단점
- 리스너 구현만 필요한 경우에는 여전히 별도의 클래스 선언 필요

올려드린 ex4.txt

```
public class Test {
	JFrame mf;
	JTextField text;

	public Test() {
		mf = new JFrame();
		mf.setTitle("예제");
		mf.setSize(200, 60);
		mf.setLocation(500, 300);
		mf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		gui();

		mf.setVisible(true);
	}

	public void gui(){
		text = new JTextField();

		JButton btn = new JButton("확인");

		mf.add(text, BorderLayout.CENTER);
		mf.add(btn, BorderLayout.EAST);

		btn.addActionListener(new ActionListener(){
		public void actionPerformed(ActionEvent e) {
			text.setText("누름");
		}
	});
	}

	public static void main(String[] args) {
	    Test test = new Test();
	    test.gui();
	}

}
```

익명(무명) 내부 클래스 임시 객체 이용





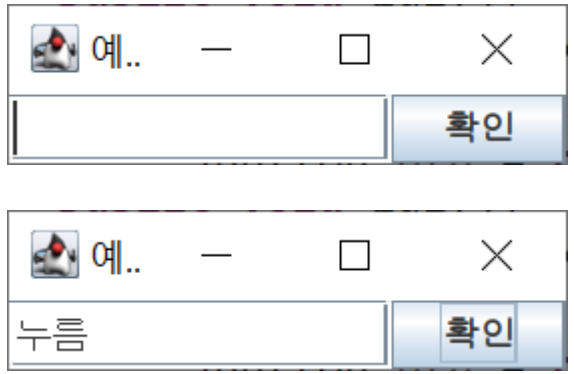
3번 방법

- (무명) 클래스를 만들면서 바로 객체 생성 후 등록

```java
public class Test implements ActionListener {
	JFrame mf;
	JTextField text;

	public Test() {
		mf = new JFrame();
		mf.setTitle("예제");
		mf.setSize(200, 60);
		mf.setLocation(500, 300);
		mf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		gui();

		mf.setVisible(true);
	}

	public void gui(){
		text = new JTextField();

		JButton btn = new JButton("확인");

		mf.add(text, BorderLayout.CENTER);
		mf.add(btn, BorderLayout.EAST);

		btn.addActionListener(this);
	}

	public static void main(String[] args) {
	    Test test = new Test();
	    test.gui();
	}

	public void actionPerformed(ActionEvent e) {
	    text.setText("누름");
	}
}
```

올려드린 ex5.txt

현재 클래스에 리스너 인터페이스를 구현



4번 방법

- 심플하게 추상 메소드만 구현 가능

Q&A

담당교수 : 신성

E-mail : sihns@hansung.ac.kr

연구실 : 우촌관 702호

휴대폰 번호 : 010-8873-8353