클릭하세요 자바

6부 AWT

- 26장 컴포넌트의 이벤트 처리

최 문 환

26장 컴포넌트의 이벤트 처리

1. 이벤트 처리 모델
2. 이벤트 처리
3. 리스너 인터페이스와 어댑터 클래스

1. 이벤트 처리 모델

이벤트(event)란?
프로그램과 사용자간의 상호작용을 위해서 사용자가 키보드나 마우스 등의 장치를 통해서 응용 프로그램에 어떤 요구를 하는 사건을 말합니다.

1. 이벤트 처리 모델

이벤트 리스너(event listener)는?
이벤트를 기다렸다가 이벤트가 발생하게 되면 전달되어오는 이벤트 소스 객체로 적당한 처리를 하게 되는 객체를 말합니다.

이벤트 소스(event source)란?
버튼(Button)이나 스크롤바(Scrollbar)와 같이 이벤트가 발생한 컴포넌트 객체를 말합니다.

2. 이벤트 처리

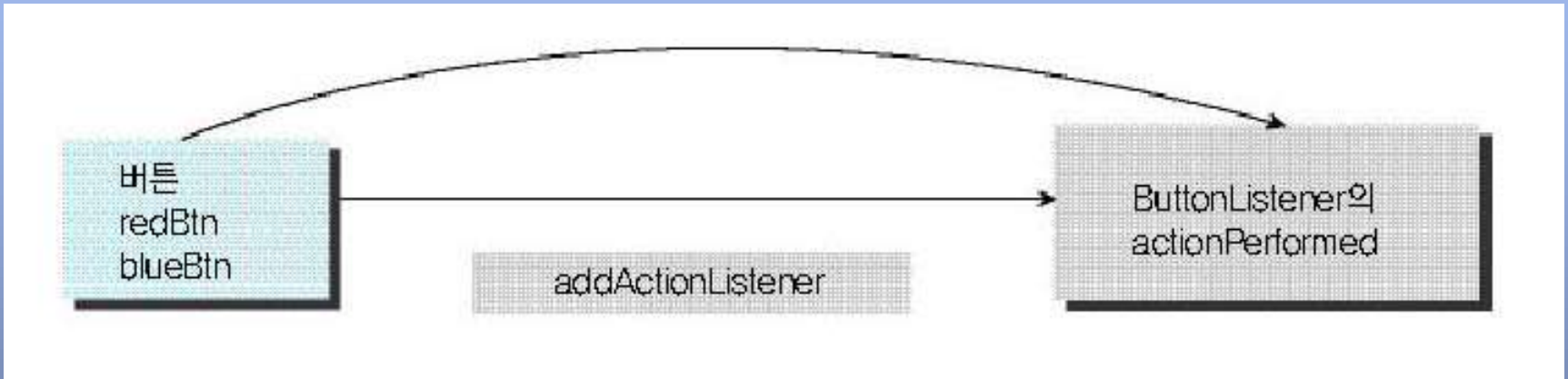
<예제> 버튼에서 발생한 이벤트 처리-[파일 이름 : EventTest01.java]

```
001:import java.awt.*;
002:
003:import java.awt.event.*;
004:
005:class FrameEvent extends Frame {
006:  //이벤트를 받아들인 컴포넌트(버튼) 객체는 FrameEvent의 멤버변수
007:  Button redBtn, blueBtn;
008:
009:  public FrameEvent( ) {
010:    setLayout(new FlowLayout());
011:
012:    //이벤트를 받아들인 컴포넌트 객체(여기서는 버튼 객체) 생성
013:    redBtn = new Button("빨간 색");
014:    blueBtn = new Button("파란 색");
015:    add(redBtn);  //버튼 객체를 프레임에 등록
016:    add(blueBtn); //배치는 배치관리자가 한다.
017:
018:
019:    ButtonListener  handler = new ButtonListener( );
020:
021:    redBtn.addActionListener(handler);
022:    blueBtn.addActionListener(handler);
```

2. 이벤트 처리

```
024:     setSize(300,200);
025:     setVisible(true);
026:
027:     addWindowListener(new WindowAdapter( ){
028:       public void windowClosing(WindowEvent e) {
029:           dispose();
030:           System.exit(0);
031:       }//windowClosing 메서드 끝
032:     }//내부 무명 클래스 정의 끝
033:     );//addWindowListener 메서드 끝
034:   }//FrameEvent  생성자 끝
035:}//FrameEvent  프레임 클래스 끝
036:
037:class ButtonListener implements ActionListener{
038:   public void actionPerformed(ActionEvent e) {
039:     System.out.println("버튼이 눌렸습니다.");
040:   } //actionPerformed 메서드 끝
041:}//ButtonListener 리스너 클래스 끝
042:
043:public class EventTest01{
044:   public static void main(String[] args) {
045:    new FrameEvent( );
046:   }
047:}
```

2. 이벤트 처리


버튼
redBtn
blueBtn
addActionListener
ButtonListener의
actionPerformed

2.1 ActionEvent 클래스

<예제> 버튼이 눌리면 프레임의 배경 색을 변경-[파일 이름 : EventTest02java]

```
001:import java.awt.*;
002://1. 이벤트 클래스 사용하기 위해서 java.awt.event 패키지를 import
003:import java.awt.event.*;
004:
005:class FrameEvent extends Frame {
006:  //이벤트를 받아들인 컴포넌트(버튼) 객체는 FrameEvent의 멤버변수
007:  Button redBtn, blueBtn;
008:
009:  public FrameEvent( ) {
010:    setLayout(new FlowLayout());  //플로우 레이아웃을 배치관리자 지정
011:
012:    //이벤트를 받아들인 컴포넌트 객체(여기서는 버튼 객체) 생성
013:    redBtn = new Button("빨간 색");
014:    blueBtn = new Button("파란 색");
015:    add(redBtn);  //버튼 객체를 프레임에 등록
016:    add(blueBtn); //배치는 배치관리자가 한다.
```

2.1 ActionEvent 클래스

```
018:     //3. 이벤트 리스너 객체를 생성
019:     ButtonListener  handler = new ButtonListener(this);
020:     //4. 이벤트를 받아들인 컴포넌트 객체(여기서는 버튼 객체)에 리스너를 등록
021:     redBtn.addActionListener(handler);
022:     blueBtn.addActionListener(handler);
023:
024:     setSize(300,200);
025:     setVisible(true);
026:
027:     addWindowListener(new WindowAdapter( ){
028:       public void windowClosing(WindowEvent e) {
029:           dispose();
030:           System.exit(0);
031:       }//windowClosing 메서드 끝
032:     }  //클래스 정의 끝
033:     );  //addWindowListener 메서드 끝
034:   }//생성자 끝
035:}//프레임 클래스 끝
```

2.1 ActionEvent 클래스

```
037:class ButtonListener implements  ActionListener{
038:   Frame  frm=null;
039:   public ButtonListener(){
040:   }
041:   public ButtonListener(Frame value){
042:     frm=value;
043:   }
044:   public void actionPerformed(ActionEvent e){
045:     if(e.getActionCommand( )=="빨간 색")
046:       frm.setBackground(Color.red);
047:     else //파란 버튼이 눌리면 프레임의 배경 색을 파란색으로
048:       frm.setBackground(Color.blue);
049:     }
050:};
051:
052:public class EventTest02{
053:   public static void main(String[] args) {
054:    new FrameEvent( );
055:   }
056:}
```

2.3 ActionEvent의 getSource( ) 메서드

<예제> 버튼이 눌리면 프레임의 배경색 변경-[파일 이름 : EventTest03.java]

```
001:import java.awt.*;
002://1. 이벤트 클래스 사용하기 위해서 java.awt.event 패키지를 import
003:import java.awt.event.*;
004:
005://2. 이벤트 소스를 포함하고 있는 클래스 자체가 이벤트 리스너가 되도록 설계
006:class FrameEvent extends Frame implements ActionListener{
007:   //이벤트를 받아들인 컴포넌트(버튼) 객체는 FrameEvent의 멤버
008:   Button redBtn, blueBtn;
009:
010:   public FrameEvent( ) {
011:     setLayout(new FlowLayout());
012:
013:     //이벤트를 받아들인 컴포넌트 객체(여기서는 버튼 객체) 생성
014:     redBtn = new Button("빨간 색");
015:     blueBtn = new Button("파란 색");
016:     add(redBtn);  // 프레임에 등록
017:     add(blueBtn);
018:
019:   //3. 이벤트 소스인 버튼에 이벤트 처리를 할 이벤트 리스너 객체 등록
020:   redBtn.addActionListener(this);//버튼을 멤버로 갖는 FrameEvent 자기 자신이
021:   blueBtn.addActionListener(this);//이벤트 리스너 객체이므로 this로 등록함
022:
023:   setSize(300,200);
024:   setVisible(true);
```

2.3 ActionEvent의 getSource( ) 메서드

```
026:    addWindowListener(new WindowAdapter( ){
027:        public void windowClosing(WindowEvent e) {
028:          dispose();
029:          System.exit(0);
030:        }//windowClosing 메서드 끝
031:      }  //클래스 정의 끝
032:    );    //addWindowListener 메서드 끝
033:  }//생성자 끝
 034:
035:  //4. 버튼이 클릭되면 호출되는 actionPerformed( ) 메서드를 오버라이딩
036:  public void actionPerformed(ActionEvent e) {
037:    //getSource( ) 메서드로 이벤트를 발생시킨 이벤트 소스 객체를 얻어 와서
038:    if(e.getSource( ) == redBtn)          //그 값이 빨간 색  버튼 객체와 동일하면
039:      this.setBackground(Color.red);      //프레임의 배경색을 빨간 색으로 변경
040:    else if(e.getSource( ) == blueBtn)    //그 값이 파란 색  버튼 객체와 동일하면
041:      this.setBackground(Color.blue);     //프레임의 배경색을 파란 색으로 변경
042:  }//actionPerformed메서드 끝
043:}//FrameEvent 클래스 끝
044:
045:public class EventTest03{
046:  public static void main(String[] args) {
047:    new FrameEvent( );
048:  }
049:}
```

3. 리스너 인터페이스와 어댑터 클래스

리스너 인터페이스를 구현해 놓은 클래스에서는 리스너 인터페이스의 추상 메서드를 모두 오버라이딩해서 메서드의 몸체를 구현해야 합니다.

메서드	설명
public void windowActivated(WindowEvent e):	윈도우가 활성화되었을 때 호출
public void windowClosed(WindowEvent e):	윈도우가 닫혔을 때 호출
public void windowClosing(WindowEvent e):	윈도우가 닫히고 있을 때 호출
public void windowDeactivated(WindowEvent e):	윈도우가 비 활성화되었을 때 호출
public void windowDeiconified(WindowEvent e):	윈도우가 정상화되었을 때 호출
public void windowIconified(WindowEvent e):	윈도우가 아이콘화되었을 때 호출
public void windowOpened(WindowEvent e):	윈도우가 열렸을 때 호출

<예제> 프레임 창 닫기 위한 이벤트 처리하기

```
<예제>-프레임 창 닫기 위한 이벤트 처리하기-[파일 이름 : EventTest04.java]
001:import java.awt.*;
002://1. 이벤트 클래스 사용하기 위해서 java.awt.event 패키지를 import
003:import java.awt.event.*;
004://2. 이벤트 소스를 포함하고 있는 클래스 자체가 이벤트 리스너가 되도록 설
005:class ExitEventFrame extends Frame implements WindowListener{
006:  public ExitEventFrame ( ){ //생성자
007:
008:    //이벤트 소스와 리스너 객체 모두 프레임임
009:    this.addWindowListener(this);
010:    setSize(200,200);
011:    setVisible(true);
012:  } //생성자의 끝
013:
014:  public void windowClosing(WindowEvent e){
015:    dispose( );     //프레임 창을 닫는다.
016:    System.exit(0);//프로그램을 종료한다.
017:  }
```

<예제> 프레임 창 닫기 위한 이벤트 처리하기

```
018:  public void windowActivated(WindowEvent e){ }
 019:  public void windowClosed(WindowEvent e){ }
020:  public void windowDeactivated(WindowEvent e){ }
021:  public void windowDeiconified(WindowEvent e){ }
022:  public void windowIconified(WindowEvent e){ }
023:  public void windowOpened(WindowEvent e){ }
024:}
025:public  class EventTest04{
026:  public static void main (String args[]){
027:    new ExitEventFrame();
028:  }
029:}
```


3.1 어댑터 클래스

어댑터 클래스란 리스너 인터페이스를 구현(implements)하여 리스너 인터페이스의 추상 메서드로 정의되어 있는 메서드를 모두 다 정의해 놓은 클래스입니다.

이벤트 처리를 위한 리스너 클래스가 필요하다면 리스너 인터페이스 대신 어댑터 클래스의 상속을 받도록 하면 이미 어댑터 클래스에는 메서드가 정의가 되어 있기 때문에 당장 필요한 메서드만 오버라이딩하면 되기 때문에 보다 간단하게 프로그램을 구현할 수 있습니다.

<예제> 프레임 창 닫기 위한 이벤트 처리를 어댑터 클래스로 하기

```
001:import java.awt.*;
002://이벤트 처리를 위한 리스너 인터페이스를 위해 반드시 기술
003:import java.awt.event.*;
004://Frame의 상속을 받아 ExitFrameEvent 클래스를 새로 설계한다.
005:class ExitFrameEvent extends Frame {
006:  public ExitFrameEvent( ) { //생성자
007:    setSize(200,200);
008:    setVisible(true);
009:   //프레임에 이벤트를 처리하기 위한  WindowAdapter 클래스의 서브 클래스로 객체를 생성하여 등록
010:    addWindowListener( new SubClass( ) );
011:  }//생성자 끝
012:}//ExitFrameEvent 클래스 정의 끝
013://WindowAdapter 클래스를 확장(extends)하여 서브 클래스 정의
```

<예제> 프레임 창 닫기 위한 이벤트 처리를 어댑터 클래스로 하기

```
014:class SubClass extends WindowAdapter{
015:    //[닫기]버튼이 눌렸을 때 호출되는 메서드 오버라이딩
016:   public void windowClosing(WindowEvent e) {
017:     //dispose( );   //자원 해제
018:     System.exit(0); //프로세스를 강제 종료함, 윈도를 종료하기 위한 코딩
019:   }//windowClosing 메서드 끝
020:}//SubClass클래스 정의 끝
021:
022:public class EventTest05{
023:   public static void main(String[] args) {
024:     new ExitFrameEvent( );
025:   }
026:}
```

3.3 내부 무명 클래스

<예제> 프레임 창 닫기 위해 어댑터 클래스를 내부 무명 클래스로 설계하기

```
001:import java.awt.*;
002://이벤트 처리를 위한 리스너 인터페이스를 위해 반드시 기술
003:import java.awt.event.*;
004://Frame의 상속을 받아 ExitFrameEvent 클래스를 새로 설계한다.
005:class ExitFrameEvent extends Frame {
006:  public ExitFrameEvent( ) { //생성자
007:    setSize(200,200);
008:    setVisible(true);
009:    //프레임에 이벤트를 처리하기 위한 리스너 객체 생성과 등록
010:    addWindowListener(new WindowAdapter( ){
011:      //[닫기]버튼이 눌렸을 때 호출되는 메서드 오버라이딩
012:      public void windowClosing(WindowEvent e) {
013:        dispose();
014:        System.exit(0);  //프로세스를 강제 종료함, 윈도를 종료하기 위한 코딩
015:      }//windowClosing 메서드 끝
016:    }  //클래스 정의 끝
017:    );  //addWindowListener 메서드 끝
018:  }//생성자 끝
019:}//클래스 정의 끝
020:
021:public class EventTest06{
022:  public static void main(String[] args) {
023:    new ExitFrameEvent( );
024:  }
025:}
```