Лабораторная работа №8. Модель обработки событий в Java.

Требование к отчету.

Отчет должен содержать:

- Титульный лист согласно образца с указанным номером и названием лабораторной работы.
- Оглавление с указанием номера страницы для каждого раздела.
- Цель работы.
- Теоретическое обоснование. Пишется самостоятельно и должно охватывать вопросы, затрагиваемые в лабораторной работе.
- Задание.
- Документированные листинги программ. После каждого листинга программы должен приводится скриншот с результатом её работы.
- Выводы.

Теоретическое обоснование.

Обработка событий в языке Java организована с использованием специальных классов-слушателей (listeners). Т.е., если мы хотим обработать нажатие на кнопку, то мы регистрируем для этой кнопки объектслушатель, в одном из методов которого прописываем действия на произошедшее событие. Причем сделать это можно как минимум двумя способами: 1 – с помощью анонимного вложенного класса; 2 – с помощью реализации нужного интерфейса. Разберем оба эти способа.

1. Обработка события с помощью анонимного вложенного класса. Создадим форму с двумя текстовыми полями и кнопкой, и по нажатию кнопки текст из одного поля будет копироваться во второе (рис. 1).

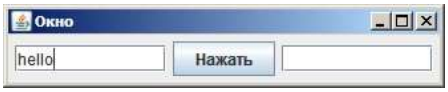

Рис. 1. Обработка события

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent; //класс для обработки
событий import java.awt.event.ActionListener; //класс-
интерфейс слушателя import javax.swing.*;

public class myAction {
  public static JFrame myFrame;//объявляем окно приложения
public static JButton myButton; //объявляем кнопку   public
static JTextField myText1; //объявляем текстовое поле
public static JTextField myText2; //объявляем другое
текстовое поле

  public static void main(String[] args){
    initWindow();//вызываем функцию инициализации нашего приложения, которую написали ниже
  }

  private static void initWindow(){ //функция для инициализации
программы    myFrame=new JFrame("Окно"); //создаем окно программы
myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//см. Лаб.раб
№4    myButton=new JButton("Нажать"); //создаем кнопку
myText1=new JTextField();  //создаем первое текстовое поле
```

```
myText1.setColumns(10); //задаем ему ширину в 10 символов
myText2=new JTextField(); //создаем второе текстовое поле
myText2.setColumns(10); //задаем ему ширину в 10 символов
JPanel myPanel=new JPanel(); //создаем панель
       myPanel.add(myText1); //ставим на панель первое текстовое окно
myPanel.add(myButton); //ставим на панель кнопку
       myPanel.add(myText2); //ставим на панель второе текстовое окно
myButton.addActionListener( new ActionListener() { //добавляем к кнопке слушатель
событий
                                  //и тут же его и создаем в виде анонимного вложенного класса
          public void actionPerformed(ActionEvent e) {//описываем процедуру обработки
события                myText2.setText(myText1.getText()); //копируем текст из первого
поля во второе
             }
       }); //!!!не забудьте закрыть нужные скобки и поставить точку с запятой
myFrame.add(myPanel,BorderLayout.NORTH);//добавляем панель в верхний регион
окна   myFrame.pack(); //упаковываем
myFrame.setMinimumSize(myFrame.getSize());//задаем минимальный размер
myFrame.setVisible(true); //показываем
  }
}
```

2. Обработка события с помощью реализации нужного интерфейса. Для этого свой класс создается с реализацией нужного класса-интерфейса (например, `ActionListener` – обработка нажатия кнопок). При этом, в созданном классе нужно переопределить метод `actionPerformed`, в котором прописать нужные действия. Создадим форму (рис.2)с тремя кнопками

Обработка и одним текстовым полем, в которое будет выводиться цифра, написанная на нажатой кнопке события (способ 2).

Рисунок 2

```
import
java.awt.BorderLayout;
import
java.awt.event.ActionEvent;
import
java.awt.event.ActionListen
er; import javax.swing.*;

public class myAction2 extends JFrame implements ActionListener{ //наш класс наследуется
от JFrame
                    //и реализует интерфейс
ActionListener   public JFrame myFrame;   public
JButton myButton;    public JTextField myText1;
public JButton myButton2;    public JButton
myButton3;
  public static void main(String[] args) {    new
myAction2(); //создаем безымянный объект нашего класса
  }

  public  myAction2(){ //конструктор нашего класса
       myFrame=new JFrame("Окно");
```

```
myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOS
E);    myButton=new JButton("1"); //создаем кнопки
myButton2=new JButton("2");      myButton3=new
JButton("3");       myText1=new JTextField();
JPanel myPanel=new JPanel();
myPanel.add(myButton);     myPanel.add(myButton2);
myPanel.add(myButton3);
       myButton.addActionListener(this);//регистрируем слушатель для кнопок – метод
нашего класса,      myButton2.addActionListener(this); // для этого используем указатель
this – указатель на       myButton3.addActionListener(this); // текущий класс –
myAction2, т.е. для обработки нажатия         //на любую из трех кнопок будет
использоваться метод actionPerformed, определенный ниже
myFrame.add(myText1,BorderLayout.NORTH);
       myFrame.add(myPanel,BorderLayout.CENTER);
myFrame.pack();
myFrame.setMinimumSize(myFrame.getSize());
       myFrame.setVisible(true);
  }

  public void actionPerformed(ActionEvent e) {//переопределяем метод actionPerformed из
интерфейса  myText1.setText(e.getActionCommand()); // ActionListener
  }
}
```

Кроме классов `ActionEvent` и `ActionListener`, существуют другие классы и соответствующие интерфейсы для обработки событий:

Класс события (который подключается через `import java.awt.event.*`)	*Интерфейс слушателя* (через `implements`)	*Методы слушателя (которые нужно переопределить для реализации обработки соответствующего события)*	*Описание метода, т.е. что с его помощью можно обработать*
`ActionEvent`	`ActionListener`	`actionPerformed()`	Нажатие на кнопку, нажатие клавиши Enter при вводе текста (для компонентов типа `JButton`, `JList`, `JTextField`)
`AdjustmentEvent`	`AdjustmentListener`	`adjustmentValueChanged()`	Изменение значения для компонента типа `JScrollbar`
`ComponentEvent` (события любого компонента)	`ComponentListener`	`componentHidden()`	Компонент стал невидимым
		`componentMoved()`	Компонент переместился
		`componentResized()`	Компонент изменил размер
		`componentShown()`	Компонент стал виден
`ContainerEvent` (события контейнера, т.е. объекта типа `JFrame`, `JPanel`, `JScrollPane`, `JDialog`, `JFileDialog` …)	`ContainerListener`	`componentAdded()`	Добавлен объект в контейнер
		`componentRemoved()`	Убран объект из контейнера
`FocusEvent` (события фокуса)	`FocusListener`	`focusGained()`	Объект получает фокус
		`focusLost()`	Объект теряет фокус
`ItemEvent`	`ItemListener`	`itemStateChanged()`	Изменение состояния элемента в списке (для `JCheckbox`, `JList`, `JComboBox` и др., в которых есть объекты типа Item)

KeyEvent (события клавиатуры)	KeyListener	keyPressed()	Была нажата кнопка клавиатуры
		keyReleased()	Была отпущена кнопка клавиатуры
		keyTyped()	Была нажата и отпущена кнопка клавиатуры
MouseEvent (события мыши)	MouseListener	mouseClicked()	Была нажата и отпущена кнопка мыши на указанном объекте
		mouseEntered()	Курсор мыши переместился на объект
		mouseExited()	Курсор мыши покинул область объекта
		mousePressed()	Была нажата кнопка мыши на указанном объекте
		mouseReleased()	Была отпущена кнопка мыши на указанном объекте
	MouseMotionListener	mouseDragged()	Во время перетаскивания (т.е. когда зажата клавиша мыши и курсор перемещается)
		mouseMoved()	Курсор мыши переместился в пределах объекта
TextEvent (события текста)	TextListener	textValueChanged()	(для компонентов типа JTextArea, JTextField)
WindowEvent (события окна приложения)	WindowListener	windowActivated()	Окно становится активным
		windowClosed()	Окно закрылось
		windowClosing()	Окно закрывается (только нажат крестик закрытия, например)
		windowDeactivated()	Окно перестало быть активным
		windowDeiconified()	Окно восстановлено из свернутого состояния
		windowIconified()	Окно минимизировано (свернуто)
		windowOpened()	Окно появилось (впервые стало видимым)

Кроме соответствующих интерфейсов слушателей есть и специальные классы-адаптеры, отличающиеся от слушателей тем, что можно переопределить не все методы, а только нужные. Например, напишем программу, в которой при наведении курсора мыши на объект будет выводиться имя класса этого объекта:

Рис. 3. При наведении мыши выводится название класса объекта.

1-й способ, с созданием собственного класса-слушателя для обработки события с **реализацией интерфейса `MouseListener`**.

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


public class myAction2 extends JFrame{  public JFrame myFrame;  public JButton myButton;
  public JTextField
myText1;    public
```

```
JCheckBox myCheck;
public JRadioButton
myRButton;
  public static JLabel myLabel; //static нужно, чтобы этот объект был доступен для других
классов

  public static void main(String[]
args) {      new myAction2();
  }

  public   myAction2(){   myFrame=new JFrame("Окно");
myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOS
E);    myButton=new  JButton("1");  //создаем  разные
объекты
      myCheck=new
JCheckBox("2");
myRButton=new
JRadioButton("3");
myText1=new JTextField();
myLabel=new JLabel(" ");
JPanel myPanel=new JPanel();
myPanel.add(myButton);
myPanel.add(myCheck);
myPanel.add(myRButton);
      myButton.addMouseListener(new MyMouseListener());//регистрируем для всех
объектов     myCheck.addMouseListener(new MyMouseListener()); //в качестве обработчика
событий мыши  myRButton.addMouseListener(new MyMouseListener());//безымянный объект
типа MyMouseListener myText1.addMouseListener(new MyMouseListener());
myFrame.add(myText1,BorderLayout.NORTH); myFrame.add(myPanel,BorderLayout.CENTER);
      myFrame.add(myLabel,BorderLayout.SOUTH);//сюда будем выводить информацию
      myFrame.setMinimumSize(new Dimension(200,100));//минимальный размер делаем 200 на
      100
      myFrame.setVisible(true);
  }
}

class MyMouseListener implements MouseListener{//создаем свой класс-слушатель,
реализующий                              //интерфейс MouseListener и поэтому нужно
переопределить все методы
     public void mouseClicked(MouseEvent e) { } // интерфейса MouseListener, даже если нам
нужен
                                                  // всего один из них
      public void mouseEntered(MouseEvent e) {
   //обращаемся к объекту myLabel класса myAction2, и устанавливаем на него текст, который
      //состоит из названия класса компонента, который вызвал обрабатываемое событие
      myAction2.myLabel.setText(e.getComponent().getClass().toString());
      }
                                 //ненужные нам методы просто оставляем пустыми
      public void mouseExited(MouseEvent e) { }

      public void mousePressed(MouseEvent e) {      }

      public void mouseReleased(MouseEvent e)
{     } }
```

Из примера видно, что неудобством данного способа является необходимость переопределять все методы реализуемого интерфейса, даже если нам реально нужен всего один.

2-й способ, с созданием собственного класса-слушателя для обработки события с **наследованием от класса `MouseAdapter`**.

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

public class myAction2
extends JFrame{   public
JFrame myFrame;   public
JButton myButton;    public
JTextField myText1;
public JCheckBox myCheck;
public JRadioButton
myRButton;    public static
JLabel myLabel;

  public static void main(String[]
args) {      new myAction2();
  }

  public   myAction2(){   myFrame=new JFrame("Окно");
myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOS
E);
       myButton=new JButton("1");
myCheck=new JCheckBox("2");      myRButton=new
JRadioButton("3");         myText1=new
JTextField();       myLabel=new JLabel(" ");
JPanel myPanel=new JPanel();
myPanel.add(myButton);
myPanel.add(myCheck);
myPanel.add(myRButton);
myButton.addMouseListener(new
MyMouseAdapter());
myCheck.addMouseListener(new
MyMouseAdapter());
myRButton.addMouseListener(new
MyMouseAdapter());
myText1.addMouseListener(new
MyMouseAdapter());
myFrame.add(myText1,BorderLayout.NORTH);
myFrame.add(myPanel,BorderLayout.CENTER);
myFrame.add(myLabel,BorderLayout.SOUTH);
myFrame.setMinimumSize(new
Dimension(200,100));
       myFrame.setVisible(true);
  }
}

class MyMouseAdapter extends MouseAdapter{ //создаем свой класс-слушатель, наследуя его
от                  //MouseAdapter, что позволяет нам переопределить только нужный нам
метод   public void mouseEntered(MouseEvent e) {
       myAction2.myLabel.setText(e.getComponent().getClass().toString());
  }
}
```

Классы-адаптеры есть для всех интерфейсов слушателей, в которых методов больше одного, в названии соответствующего интерфейса достаточно поменять `Listener` на `Adapter`.

Задание.

Задачи: I. Обязательная задача для всех:

Найти описание следующих классов-слушателей и выписать основное:

`ChangeListener`, `MouseWheelListener`.

II. **Индивидуальное задание.** К созданному интерфейсу из предыдущей лаб. раб. добавить функциональность – как минимум 3 различных слушателя (например, при нажатии кнопки выводить текст в текстовое поле, при наведении курсора мыши выводить соответствующее сообщение, при изменении ползунка типа `JSlider` выводить его значение, при нажатии клавиши Enter после набора текста перемещать его куда-нибудь и т.д.). У студентов с одним вариантом интерфейса не должны быть одинаковые слушатели!