Основные понятия обработки событий

Событие - это объект, описывающий изменение состояния источника событий (нажатие клавиши, движение мыши, нажатие кнопки, изменение компонента …).

Источник события - это объект (субъект), за которым ведется наблюдение, и извещающий о событии (кнопки, текстовые поля, списки).

Слушатель (наблюдатель)событий - это объект, определяющий методы, которые следует вызывать субъекту, для того чтобы сообщить о своих изменениях. Каждому типу событий соответствует свой слушатель с определенным интерфейсом.

Адаптер – это класс, в котором описан интерфейс определенного слушателя. Если слушатель называется XXXListener, то имя адаптера выглядит как XXXAdapter.

Классы событий

ActionEvent - событие, определяемое компонентом, например нажатие кнопки;

ComponentEvent - генерируется, когда позиция, размер или видимость компонента изменяются;

ContainerEvent - генерируется при добавлении или удалении элемента из контейнера;

FocusEvent - генерируется при получении или потери фокуса ввода компонентом;

ItemEvent - событие выбора или отменены выбора элемента. Например, изменение состояния кнопки-флажка, выбор элемента меню или списка;

KeyEvent - событие ввода с клавиатуры;

MouseEvent - события мыши;

WindowEvent - события окна, как активация и свертывание.

Интерфейсы обработки событий

ActionListener - интерфейс обработки события ActionEvent;

ItemListener - интерфейс обработки события ItemEvent;

KeyListener - интерфейс обработки события KeyEvent;

MouseListener - интерфейс обработки события MouseEvent, для нажатия/отжатия кнопок и входа/ухода курсора мыши с области компонента;

MouseMotionListener - интерфейс обработки события MouseEvent, для движение курсора мыши или перетаскивании мышью;

MouseWheelListener - интерфейс обработки события MouseEvent, для прокрутки колеса

Порядок написания обработчиков

1. Определить класс (слушатель или адаптер), который отвечает за обработку события. Этот класс должен наследовать интерфейс, отвечающий за обработку данного события.
2. Написать реализацию методов интерфейса.
3. Зарегистрировать слушателя в классе, где может происходить это событие.

Окно с кнопкой закрытия

```
import java.awt.*;
import j ava.awt.event.*;
class SimpleFrame extends Frame{
  SimpleFrame(String s){
    super (s); //название окна
    setSize(400, 150);
    setVisible(true);
   /* добавление слушателя, Адаптер — это класс, который реализует
интерфейс определенного слушателя */
 addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent ev){
        System.exit (0);
      }
    } ) ;
  }
  public static void main(String[] args){
    new SimpleFrame(" Моя программа");
  }
}
Swing21
```

Обработка события ввода с клавиатуры

```
import javax.swing.*;
import java.awt.event.*;
public class FirstEvents extends JFrame {
FirstEvents() {
super("FirstEvents");  //  название окна
setDefaultCloseOperation(EXIT_ON_CLOSE); // при закрытии окна – выход
addKeyListener(new KeyL()); // регистрируем слушателя ввода с клавиатуры
setSize(200, 200); // выводим окно на экран
setVisible(true);
}
public static void main(String[] args) {
new FirstEvents();
}
}
// этот класс будет получать извещения о событиях клавиатуры
class KeyL implements KeyListener {
public void keyTyped(KeyEvent k) {// пользователь нажал и отпустил клавишу с печатным символом
char ch =k . getKeyChar(); System.out.println(ch);
}
public void keyPressed(KeyEvent k) {// нажатие клавиши
int code =k .getKeyCode();
if (code == KeyEvent.VK_LEFT) System.out.println(“Стрелка влево”);
}
public void keyReleased(KeyEvent k) {// отпускание нажатой клавиши
I
int code =k .getKeyCode();
f (code == KeyEvent.VK_DOWN) System.out.println(“DOWN”);
}
} Swing1
```

Пример диалогового окна

```
import javax.swing.*;
import java.awt.event.*;
public class MessageDialogs extends JFrame {
// этот значок выведем в одном из сообщений
private ImageIcon icon = new ImageIcon("question.gif");
public MessageDialogs() {
super("MessageDialogs");
setDefaultCloseOperation(EXIT_ON_CLOSE);
// кнопки, после щелчков на которых  выводятся сообщения
JButton message1 = new JButton("2 параметра");
message1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(
MessageDialogs.this, "<html><h2>Привет!<br>на HTML !");
}
});
JButton message2 = new JButton("4 параметра");
message2.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(MessageDialogs.this,new String[] {
"Сообщение может быть",
"записано массивом!" }, "Свой заголовок", JOptionPane.ERROR_MESSAGE);
}
});
JButton message3 = new JButton("5 параметров");
message3.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(MessageDialogs.this,
"Настроено все что можно", "Свой заголовок",
JOptionPane.INFORMATION_MESSAGE, icon);
}
});
// выведем окно на экран
JPanel contents = new JPanel();
contents.add(message1);
contents.add(message2);
contents.add(message3);
setContentPane(contents);
setSize(400, 200);
setVisible(true);
}
public static void main(String[] args) {
new MessageDialogs();
}
}
```

Swing7

Обработка событий от кнопок

```
import javax.swing.*; import javax.swing.event.*; import java.awt.*; import java.awt.event.*;
public class ButtonEvents extends JFrame {
public ButtonEvents() {
super("ButtonEvents");
setDefaultCloseOperation( EXIT_ON_CLOSE );
// получаем панель содержимого
Container c = getContentPane();
JButton button = new JButton("Нажмите меня!"); // создаем кнопку и помещаем ее на север
    окна
c.add(button, "North");
info = new JTextArea("Пока событий не было\n"); // поле для вывода сообщений о событиях
c.add(new JScrollPane(info));  //Добавляем прокрутку
// привязываем к нашей кнопке слушателей событий ,  слушатели описаны как
    внутренние классы
button.addActionListener(new ActionL());
// выводим окно на экран
setSize(400, 300);
setVisible(true);
}
JTextArea info;
class ActionL implements ActionListener {
public void actionPerformed(ActionEvent e) {
info.append("Получено сообщение о нажатии кнопки! От - "+ e.getActionCommand() + "\n");
} }
public static void main(String[] args) {
new ButtonEvents();
}}
```

Переключатели JRadioButton

```
import javax.swing.*;
import java.awt.*;
public class RadioButtons extends JFrame {
public RadioButtons() {
super("RadioButtons");
setDefaultCloseOperation( EXIT_ON_CLOSE );
Container c = getContentPane(); // получаем панель содержимого
c.setLayout(new FlowLayout()); // используем последовательное расположение
JRadioButton r1 = new JRadioButton("Сам по себе"); // отдельный переключатель
// группа связанных переключателей в своей собственной панели
JPanel panel = new JPanel(new GridLayout(0, 1, 0, 5)); // табличное расположение 0 – произвольное число строк
panel.setBorder(BorderFactory.createTitledBorder("Внешний вид")); // Создаем рамку с заголовком
ButtonGroup bg = new ButtonGroup();
String[] names = { “Java", “Windows    ", “Linux" };
for (int i=0; i < names.length; i++) {
JRadioButton radio = new JRadioButton(names[i]);
panel.add(radio);
bg.add(radio);
}
// добавляем все в контейнер
c.add(r1);
c.add(panel);
pack(); // устанавливаем оптимальный размер
setVisible(true); // выводим окно на экран
}
public static void main(String[] args) {
new RadioButtons();
}
}
```

Swing6

Текстовые поля

```
import javax.swing.*;
import java.awt.Font;
import java.awt.event.*;
public class UsingTextFields extends JFrame {
// наши поля
JTextField smallField, bigField;
public UsingTextFields() {
super("UsingTextFields");
setDefaultCloseOperation(EXIT_ON_CLOSE);
// создаем текстовые поля с максимальным количеством символов
smallField = new JTextField(10);
bigField = new JTextField("Текст поля", 25); // поле с начальным текстом
// дополнительные настройки
bigField.setFont(new Font("Dialog", Font.PLAIN, 16));
bigField.setHorizontalAlignment(JTextField.RIGHT);  // выравнивание по правому краю
// слушатель окончания ввода в поле smallField
smallField.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
// показываем введенный текст в диалоговом окне
JOptionPane.showMessageDialog(
UsingTextFields.this, "Ваше слово: " + smallField.getText());
}});
// поле с паролем
JPasswordField password = new JPasswordField(15);
password.setEchoChar('$'); //  отображаемый символ $
// добавляем поля в окно и выводим его на экран
JPanel contents = new JPanel();
contents.add(smallField);
contents.add(bigField);
contents.add(password);
setContentPane(contents); // добавление панели в окно
setSize(400, 300);
setVisible(true);
}
public static void main(String[] args) {
new UsingTextFields();
}
} Swing10
```

Многострочные текстовые поля

```
import javax.swing.*;
import java.awt.Font;
public class UsingTextArea extends JFrame {
public UsingTextArea() {
super("UsingTextArea");
setDefaultCloseOperation(EXIT_ON_CLOSE);
// создаем пару многострочных полей
JTextArea area1 = new JTextArea("Многострочное поле", 5, 10);
// нестандартный шрифт и табуляция
area1.setFont(new Font(" Times New Roman ", Font.PLAIN, 14));
area1.setTabSize(10); // по умолчанию 8
JTextArea area2 = new JTextArea(15, 10);
// параметры переноса слов
area2.setLineWrap(true);
area2.setWrapStyleWord(true);
// добавим поля в окно
JPanel contents = new JPanel();
contents.add(new JScrollPane(area1));
contents.add(new JScrollPane(area2));
setContentPane(contents);
// выводим окно на экран
setSize(400, 300);
setVisible(true);
}
public static void main(String[] args) {
new UsingTextArea();
}
} Swing11
```

Вывод вспомогательной информации

```
import java.awt.*;
import javax.swing.*;
public class Labels extends JFrame implements SwingConstants {
public Labels() {
super("Labels");
// при закрытии окна заканчиваем работу
setDefaultCloseOperation(EXIT_ON_CLOSE);
// самая простая надпись
JPanel contents = new JPanel();
JLabel l1 = new JLabel("Ваше имя:");
JTextField name = new JTextField(20);
contents.add(l1);
contents.add(name);
// надпись со значком
JLabel l2 = new JLabel(new ImageIcon("monkey.gif"));
adjustLabel(l2);
l2.setHorizontalAlignment(LEFT);  // значок слева
contents.add(l2);
// надпись с нестандартным выравниванием
JLabel l3 = new JLabel("Текст и значок", new ImageIcon("bulb.gif"), RIGHT);  //значок справа
adjustLabel(l3);
l3.setVerticalTextPosition(BOTTOM); //  текст  снизу от значка
l3.setHorizontalTextPosition(LEFT);   //  текст слева от значка
contents.add(l3);
// вывод окна на экран
setContentPane(contents);
setSize(320, 300);
setVisible(true);
}
// метод производит специальную настройку надписи
private void adjustLabel(JLabel l) {
l.setOpaque(true); //  непрозрачность
l.setBackground(Color.red); // цвет фона
l.setPreferredSize(new Dimension(250, 100));
}
public static void main(String[] args) {
new Labels();
}
}
```

Всплывающие подсказки

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ToolTips extends JFrame {
public ToolTips() {
super("ToolTips");
setDefaultCloseOperation(EXIT_ON_CLOSE);
// добавим несколько кнопок с подсказками
JButton b1 = new JButton("Один");
b1.setToolTipText("Это первая кнопка");  // задание подсказки
JButton b2 = new JButton() {
public Point getToolTipLocation(MouseEvent e) {
return new Point(10, 10);                                    // место вывода подсказки кнопки 1 ,10 пикселей от верхнего левого угла
}
public String getToolTipText(MouseEvent e) {          // место вывода первой подсказки кнопки 2
if ( e.getY() > 10 ) {
return "Нижняя часть кнопки!";
}
return super.getToolTipText(e);
}};
b2.setText("Два");
b2.setToolTipText("<html><h3>Это вторая кнопка.<ul>" +                              // задание второй подсказки
"Она:<li>Ничего не делает<li>Но ее можно нажать!");
JPanel contents = new JPanel();
contents.add(b1);
contents.add(b2);
// выводим окно на экран
setContentPane(contents);
setSize(400, 150);
setVisible(true);
}
public static void main(String[] args) {
new ToolTips();
}
}
```

Редактируемая таблица

```
import javax.swing.*;
public class TableDefaultEditing extends JFrame {
// название столбцов
private String[] columns = { "Имя", "Любимый Цвет" };
// данные для таблицы
private String[][] data = {
{ "Иван", "Зеленый" },
{ "Александр", "Красный"},
{ "Петр", "Синий"}
};
public TableDefaultEditing() {
super("TableDefaultEditing");
setDefaultCloseOperation(EXIT_ON_CLOSE);
// создаем таблицу
JTable table = new JTable(data, columns);
// настраиваем стандартный редактор
JComboBox combo = new JComboBox(new String[] { "Зеленый", "Желтый", "Белый"});
DefaultCellEditor editor = new DefaultCellEditor(combo);
table.getColumnModel().getColumn(1).setCellEditor(editor);
// выводим окно на экран
getContentPane().add(new JScrollPane(table));
setSize(400, 300);
setVisible(true);
}
public static void main(String[] args) {
new TableDefaultEditing();
}
}
```

Использование стандартной модели таблицы

```
import javax.swing.*;
import javax.swing.table.*;
import java.awt.event.*;
public class UsingDefaultTableModel extends JFrame {
// наша модель
private DefaultTableModel dtm;
public UsingDefaultTableModel() {
super("UsingDefaultTableModel");
setDefaultCloseOperation(EXIT_ON_CLOSE);
// создаем стандартную модель
dtm = new DefaultTableModel();
// задаем названия стоблцов
dtm.setColumnIdentifiers(new String[] {"Номер", "Товар", "Цена"});
// наполняем модель данными
dtm.addRow(new String[] {"№1", "Блокнот", "5.5"});
dtm.addRow(new String[] {"№2", "Телефон", "175"});
dtm.addRow(new String[] {"№3", "Карандаш", "1.2"});
// передаем модель в таблицу
JTable table = new JTable(dtm);
// данные могут меняться динамически
JButton add = new JButton("Добавить");
add.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
// добавляем новые данные
dtm.addRow(new String[] {"?", "Новинка!", "Супер Цена!"});
}
});
JButton remove = new JButton("Удалить");
remove.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
// удаляем последнюю строку (отсчет с нуля)
dtm.removeRow(dtm.getRowCount() - 1);
}
});
// добавляем кнопки и таблицу в панель содержимого
getContentPane().add(new JScrollPane(table));
JPanel buttons = new JPanel();
buttons.add(add);
buttons.add(remove);
getContentPane().add(buttons, "South");
// выводим окно на экран
setSize(300, 300);
setVisible(true);
}
public static void main(String[] args) {
new UsingDefaultTableModel();
}
}
```

```
import java.awt.*;
import java.awt.event.*;
class HelloWorldFrame extends Frame{
  HelloWorldFrame(String s){
    super(s);
  }
  public void paint(Graphics g){
    g.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 30));
    g.drawstring("Hello, XXI century World!", 20, 100);
  }
  public static void main(String[] args){
    Frame f = new HelloWorldFrame("Здравствуй, мир XXI века!");
    f.setSize(400, 150);
    f.setvisible(true);
    f.addWindowListener(new WindowAdapter(){
      public void windowciosing(WindowEvent ev)(
        System.exit(0);
      }
    });
  }
}
```

```
import java.io.*; //импортирование пакета ввода-вывода
        import java.awt.*; //импортирование пакета awt
        import java.awt.event.*; //импортирование пакета поддержки событий
        public class GUISample extends Frame{ //объявление класса GUISample
        Button b1 = new Button("Add"); //создание кнопки с надписью "Add"
        Choice ch1=new Choice(); //создание раскрывающегося списка
        TextField tf1 = new TextField(); //создание текстового поля (строки
        //ввода)
        Label label1 = new Label("Enter your text here:"); //создание текстовой
        //строки
        List l1 = new List(); //создание списка
        public GUISample(){ //объявление конструктора класса
        setLayout(null); //отключение менеджера компоновки
        setSize(600,400); //установка размеров фрейма
        setTitle("This is my Frame"); //установка заголовка фрейма
        setBackground(Color.cyan); //установка цвета заднего фона фрейма
        add(b1); //добавление кнопки к окну
        b1.setBounds(220,200,84,24); //установка размеров кнопки
        b1.setForeground(Color.black); //установка цвета переднего фона кнопки
        b1.setBackground(Color.magenta); //установка цвета заднего фона кнопки
        add(ch1); //добавление раскрывающегося списка к окну
        ch1.setBounds(50,120,120,20); //установка размеров раскрывающегося
        //списка
        add(tf1); //добавление текстового поля к окну
        tf1.setBounds(200,80,120,20); //установка размеров текстового поля
        add(label1); //добавление текстовой строки к окну
        label1.setBounds(200,55,120,20); //установка размеров текстовой строки
        add(l1); //добавление списка к окну
        l1.setBackground(Color.white); //установка цвета заднего фона списка
        l1.setBounds(350,120,200,216); //установка размеров списка
        /*регистрация блока прослушивания событий типа WindowEvent*/
        addWindowListener(new WindowClose());
        /*регистрация блока прослушивания событий типа ActionEvent*/
        b1.addActionListener(new ButtonAdd());
        }
        /*объявление класса-адаптера для обработки Window-событий*/
        class WindowClose extends WindowAdapter {
        /*метод, который вызывается при закрытии окна*/
        public void windowClosing(WindowEvent we) {
        setVisible(false); //фрейм-окно становится невидимым
        }
        }
        /*объявление класса для обработки Action-событий (класс ButtonAdd реализует интерфейс ActionListener)*/
        class ButtonAdd implements ActionListener {
        /*реализация метода, который вызывается при наступлении action-события*/
        public void actionPerformed(ActionEvent event) {
        /*добавление текста из текстового поля в раскрывающийся список*/
        ch1.add(tf1.getText());
        /*добавление текста из текстового поля в список*/
        l1.add(tf1.getText(),2);
        }
        }
        static public void main(String args[]){ //объявление метода main()
        GUISample MyFrame=new GUISample(); //создание экземпляра класса GUISample
        MyFrame.setVisible(true); //выведение окна на экран дисплея
        } }
```