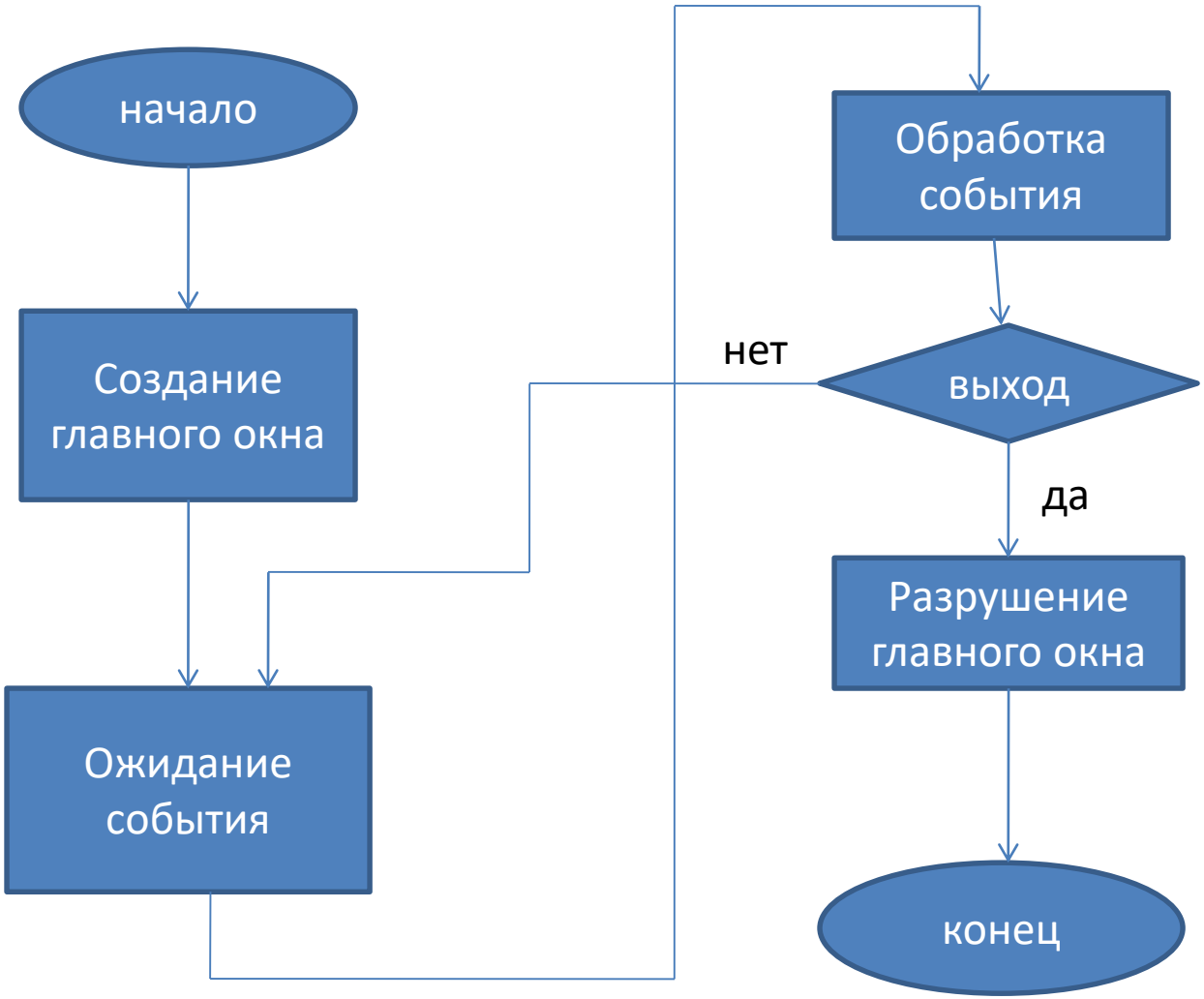
Схема работы GUI - приложения



Основные понятия обработки событий

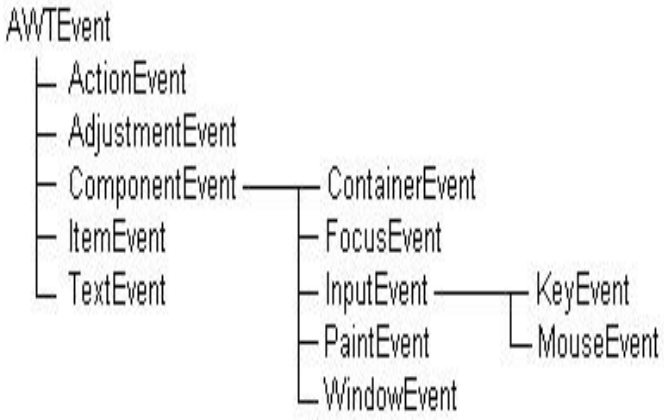
Событие - это объект, описывающий изменение состояния источника событий (нажатие клавиши, движение мыши, нажатие кнопки, изменение компонента …).

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 абстрактный класс, в котором описан интерфейс определенного слушателя. Если слушатель называется XXXListener, то имя адаптера выглядит как XXXAdapter (позволяет переопределять не все методы слушателя, а только те, которые нужны).

Иерархия событий AWT



Узнать, в каком объекте произошло событие, можно методом getSource() класса Eventobject.
Этот метод возвращает тип object.

События типа ComponentEvent, KeyEvent, MouseEvent могут быть привязаны ко всем компонентам.
События типа ContainerEvent — только в контейнерах: Container, Dialog, FileDialog, Frame, Panel, ScrollPane, Window.
События типа WindowEvent возникают ТОЛЬКО в окнах: Frame, Dialog, FileDialog, Window.
События типа TextEvent генерируются только в контейнерах Textcomponent, TextArea, TextField.
События типа ActionEvent проявляются только в контейнерах Button, List, TextField.
События типа ItemEvent возникают только в контейнерах Checkbox, Choice, List.
События типа AdjustmentEvent возникают только в контейнере Scrollbar.

События, связанные с визуальными компонентами AWT

Компонент	Событие	Описание
Button	ActionEvent	Пользователь нажал кнопку
CheckBox	ItemEvent	Пользователь установил или сбросил флажок
CheckBoxMenuItem	ItemEvent	Пользователь установил или сбросил флажок в пункте меню
Choice	ItemEvent	Пользователь выбрал элемент списка или отменил его выбор
Component	ComponentEvent	Элемент либо перемещен, либо он стал скрытым, либо видимым
	FocusEvent	Элемент получил или потерял фокус ввода
	KeyEvent	Пользователь нажал или отпустил клавишу
	MouseEvent	Пользователь нажал или отпустил кнопку мыши, либо курсор мыши вошел или покинул область, занимаемую элементом, либо пользователь просто переместил мышь или переместил мышь при нажатой кнопке мыши
Container	ContainerEvent	Элемент добавлен в контейнер или удален из него
List	ActionEvent	Пользователь выполнил двойной щелчок мыши на элементе списка
	ItemEvent	Пользователь выбрал элемент списка или отменил выбор
MenuItem	ActionEvent	Пользователь выбрал пункт меню
Scrollbar	AdjustmentEvent	Пользователь осуществил прокрутку
TextComponent	TextEvent	Пользователь внес изменения в текст элемента
TextField	ActionEvent	Пользователь закончил редактирование текста элемента
Window	WindowEvent	Окно было открыто, закрыто, представлено в виде пиктограммы, восстановлено или требует восстановления

Методы интерфейсов слушателей

Класс события	Интерфейс слушателя	Абстрактные методы
ActionEvent (событие, определяемое компонентом, например нажатие кнопки)	ActionListener	actionPerformed(ActionEvent e) генерация события
AdjustmentEvent (событие при изменении полосы прокрутки)	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e) при изменении состояния полосы прокрутки
ComponentEvent (генерируется, когда позиция, размер или видимость компонента изменяются)	ComponentListener	componentResized(ComponentEvent e) размер компонента изменился
		componentMoved(ComponentEvent e) позиция компонента изменилась
		componentShown(ComponentEvent e) компонент был сделан видимым
		componentHidden(ComponentEvent e) компонент был сделан невидимым
ContainerEvent (генерируется при добавлении или удалении элемента из контейнера)	ContainerListener	componentAdded(ContainerEvent e) компонент добавлен
		componentRemoved(ContainerEvent e) компонент удален
FocusEvent (генерируется при получении или потери фокуса ввода компонентом)	FocusListener	focusGained(FocusEvent e) при получении фокуса компонентом
		focusLost(FocusEvent e) при потере фокуса компонентом
ItemEvent (событие выбора или отменены выбора элемента. Например, изменение состояния кнопки-флажка, выбор элемента меню или списка)	ItemListener	itemStateChanged(ItemEvent e) при смене состояния JCheckBox

Методы интерфейсов слушателей

Класс события	Интерфейс слушателя	Абстрактные методы
KeyEvent (событие ввода с клавиатуры)	KeyListener	keyPressed(KeyEvent e) Вызывается при нажатии пользователем на любую клавишу
		keyReleased(KeyEvent e) Вызывается после того, как пользователь нажмет и отпустит любую клавишу
		keyTyped(KeyEvent e) Срабатывает каждый раз, когда пользователь вводит символы Unicode
MouseEvent (события мыши)	MouseListener (нажатия/отжатия кнопок и входа/ухода курсора мыши с области компонента)	mouseClicked(MouseEvent e) выполнен щелчок мышкой на наблюдаемом объекте
		mousePressed(MouseEvent e) кнопка мыши нажата в момент, когда курсор находится над наблюдаемым объектом
		mouseReleased(MouseEvent e) кнопка мыши отпущена в момент, когда курсор находится над наблюдаемым объектом
		mouseEntered(MouseEvent e) курсор мыши вошел в область наблюдаемого объекта
		mouseExited(MouseEvent e) курсор мыши вышел из области наблюдаемого объекта
	MouseMotionListener (движение курсора мыши или перетаскивании мышью)	mouseDragged(MouseEvent e) Перемещение курсора мыши
		mouseMoved(MouseEvent e) пользователь нажимает клавишу мыши, перемещает курсор и затем отпускает клавишу мыши

Методы интерфейсов слушателей

Класс события	Интерфейс слушателя	Абстрактные методы
TextEvent (генерируется при изменении текстового поля)	TextListener	textValueChanged(TextEvent e) значение текста изменилось
WindowEvent (события окна, активация и свертывание)	WindowListener	windowOpened(WindowEvent e) При открытии ока
		windowClosing(WindowEvent e) когда пользователь пытается закрыть окно
		windowClosed(WindowEvent e) когда окно закрылось
		windowIconified(WindowEvent e) окно изменяется от нормального до минимизированного состояния
		windowDeiconified(WindowEvent e) окно изменяется от минимизированного до нормального состояния
		windowActivated(WindowEvent e) Окно становится активным
		windowDeactivated(WindowEvent e) Окно больше не является активным

Механизм обработки событий

Подход к обработке событий основан на модели делегирования событий, т.е. логика приложения, обрабатывающего события, четко отделена от логики пользовательского интерфейса, извещающего об этом событии.

- При наступлении события источник посылает объекты события всем зарегистрированным обработчикам (слушателям).
- Обработчики используют информацию, инкапсулированную в объекте события, для того, чтобы решить, как реагировать на это событие.
- После обработки события слушатель возвращает управление.

Порядок написания обработчиков

1. Определить класс (слушателя 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 (addXXX).

Обработка событий

- В основном классе
- Во внешнем классе
- Во вложенном классе
- В анонимном классе

Подключение нескольких обработчиков

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

public class TestButton {
private JFrame f;
private JButton b,b1;
public TestButton() {
f = new JFrame("Test");
f.setLayout( new FlowLayout( FlowLayout.CENTER ));
b = new JButton("Кнопка1");b1 = new JButton("Кнопка2");
b.setActionCommand("Кнопка1 нажата"); b1.setActionCommand("Кнопка2 нажата"); // Установка названия команды
}
public void launchFrame() {
b.addActionListener(new ButtonHandler1()); // Подключение Обработчик 1 к кнопке 1
b.addActionListener(new ButtonHandler2()); b1.addActionListener(new ButtonHandler2()); // Подключение Обработчик 2 к кнопке 1 и 2
f.add(b); f.add(b1);
f.pack();
f.setVisible(true);
}
public static void main(String args[]) {
TestButton guiApp = new TestButton();
guiApp.launchFrame();

}
}
class ButtonHandler1 implements ActionListener {// Обработчик 1
public void actionPerformed(ActionEvent e) {
      System.out.println("Обработчик 1 " + e.getActionCommand());

}
}

class ButtonHandler2 implements ActionListener {// Обработчик 2
public void actionPerformed(ActionEvent e) {
      System.out.println("Обработчик 2 " + e.getActionCommand());

}
}Swing8
```

Адаптер для окна

```
import java.awt.*;
import java.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, Обработчик находится в анонимном
      классе */
 addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent ev){
        System.out.println("Завершение приложения ");
        System.exit (0);
      }
    } ) ;
  }
  public static void main(String[] args){
    new SimpleFrame(" Моя программа");
  }
}
```

Swig24

Обработка события ввода с клавиатуры

```
import javax.swing.*;
import java.awt.event.*;
public class FirstEvents extends JFrame {
FirstEvents() {
super("FirstEvents");  //  название окна
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();
}
}
// обработчик находится во внешнем классе, 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
int code =k .getKeyCode();
if (code == KeyEvent.VK_DOWN) System.out.println(“стрелка вниз”);
}
}
```

Swing1

Примеры диалоговых окон

```
import javax.swing.*;
import java.awt.event.*;
public class MessageDialogs extends JFrame {
// этот значок выведем в одном из сообщений
private ImageIcon icon = new ImageIcon("3266.png");
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,
“Вывод текста", "Свой заголовок",
JOptionPane.INFORMATION_MESSAGE, icon);
}
});
// выведем окно на экран
JPanel contents = new JPanel();
contents.add(message1); // добавляем кнопки
contents.add(message2);
contents.add(message3);
setContentPane(contents);
setSize(400, 200);
setVisible(true);
}
public static void main(String[] args) {
new MessageDialogs();
}
}
```

Обработчики находятся в анонимном классе

Создание меню

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class MenuExp extends JFrame {
  public MenuExp() {
        setTitle("Menu Example");
      setSize(150, 150);
// Создаем контейнер  для меню
      JMenuBar menuBar = new JMenuBar();
      setJMenuBar(menuBar); // Добавляем контейнер в окно
      // Создаем выпадающие пункты меню и добавляем их
       в контейнер
       JMenu fileMenu = new JMenu("File");
      JMenu editMenu = new JMenu("Edit");
      menuBar.add(fileMenu);
      menuBar.add(editMenu);
//Создание  пунктов меню
      JMenuItem newAction = new JMenuItem("New");
      JMenuItem openAction = new JMenuItem("Open");
      JMenuItem exitAction = new JMenuItem("Exit");
     JCheckBoxMenuItem checkAction =
new JCheckBoxMenuItem("Check Action");
// добавляем в меню файл пункты меню
 fileMenu.add(newAction);
 fileMenu.add(openAction);
fileMenu.add(checkAction);
// добавляем разделительную линию
      fileMenu.addSeparator();
      fileMenu.add(exitAction);
  // обработчик для пункта меню new
      newAction.addActionListener(
new ActionListener() {
  public void actionPerformed(ActionEvent arg0)
{
        System.out.println("Выбрали меню new ");
        }     }          );
         }
  public static void main(String[] args) {
     MenuExp me = new MenuExp();
      me.setVisible(true);
  } }
```

Swing13

Панель вывода с прокруткой

```
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"); // многострочное поле для вывода сообщений
c.add(new JScrollPane(info));  //Добавляем поле с прокруткой
// привязываем к нашей кнопке слушателей событий ,  слушатели описаны как внутренние классы
button.addActionListener(new ActionL());
// выводим окно на экран
setSize(200, 200);
setVisible(true);
}
JTextArea info;
class ActionL implements ActionListener { // обработчик кнопки
public void actionPerformed(ActionEvent e) {
info.append("Получено сообщение от кнопки- "+ e.getActionCommand() + "\n");
} }
public static void main(String[] args) {
new ButtonEvents();
}}
```

Swing17

Программный вызов события

```
public class TestFrame extends JFrame
{
  private JButton    button1;
  private JButton    button2;
  private JButton    button3;
     public TestFrame() {
        super("Test frame");
        createGUI();
    }
     public void createGUI() {
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout());
         button1 = new JButton("Button 1");
        button1.setActionCommand("Button 1 нажата!");
        panel.add(button1);
         button2 = new JButton("Button 2");
        button2.setActionCommand("Button 2 нажата");
        panel.add(button2);
        button3 = new JButton("Button 3");
        button3.setActionCommand("Button 3 нажата");
        panel.add(button3);

        ActionListener actionListener = new TestActionListener();

        button1.addActionListener(actionListener);
        button2.addActionListener(actionListener);
        button3.addActionListener(actionListener);
         getContentPane().add(panel);
// установка размера окна
        setPreferredSize(new Dimension(320, 100));
    }
```

```
public class TestActionListener implements ActionListener {
       public void actionPerformed(ActionEvent e) {
//кнопка, где произошло событие
          JButton button = (JButton) e.getSource();
          System.out.println (button.getText() + ", " +
                              e.getActionCommand());
          if (e.getSource() == button3) {
    // создание события -нажатие мышью на кнопку 2 и
запись строки
             ActionEvent e1 = new ActionEvent(button2,
Event.MOUSE_DOWN,"Button 2 нажата программно!");
             ActionListener[] listeners;
//  Возвращает массив всех слушателей для кнопки 2
listeners = button2.getActionListeners();
// генерируем событие
             listeners[0].actionPerformed(e1);
          }
       }
    }

     public static void main(String[] args) {
             TestFrame frame = new TestFrame();
              frame.pack();
// центрирование окна
              frame.setLocat ionRelativeTo(null);
              frame.setVisible(true);
     }
}
```

Swing26

Переключатели JRadioButton

```
import javax.swing.*;
import java.awt.*; import java.awt.event.*;
public class RadioButtons extends JFrame {
public RadioButtons() {
super("RadioButtons");
Container c = getContentPane(); // получаем панель содержимого
c.setLayout(new FlowLayout()); // используем последовательное расположение
JRadioButton r1 = new JRadioButton(“Сброс"); // 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,1 столбец, 0 по горизонтали и 5 по вертикали
panel.setBorder(BorderFactory.createTitledBorder("Внешний вид")); // Создаем рамку с заголовком
ButtonGroup bg = new ButtonGroup(); //создание группы для переключателей
String[] names = { “Java", “Windows    ", “Linux" };
 for (int i=0; i < names.length; i++) { // 3 переключателя
JRadioButton radio = new JRadioButton(names[i]);
panel.add(radio);
bg.add(radio); }
// добавляем все в контейнер
c.add(r1);
c.add(panel);
r1.addItemListener(new ItemListener() {    public void itemStateChanged(ItemEvent e){
    if (e.getStateChange()  == ItemEvent.SELECTED)        { System.out.println(" Сброс "); }else { System.out.println("
       Установлен "); }     } });
pack(); // устанавливаем минимальный размер окна, достаточный для отображения всех компонентов
setVisible(true); // выводим окно на экран
}
public static void main(String[] args) {
new RadioButtons();
} }
```

```
radio.setActionCommand(names[i]);
radio.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
   System.out.println(evt.getActionCommand());
   }
    }
```

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 минимальным количеством символов
smallField = new JTextField(2);
bigField = new JTextField("Текст поля", 25); // поле с начальным текстом
// дополнительные настройки
bigField.setFont(new Font("Dialog", 1, 16)); // выбор шрифта (имя, стиль, размер)
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 smallField
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();
}
}
```

Многострочные текстовые поля

```
import javax.swing.*;
import java.awt.Font;
public class UsingTextArea extends JFrame {
public UsingTextArea() {
super("UsingTextArea");
// создаем пару многострочных полей 5 строк и 10 символов в строке
JTextArea area1 = new JTextArea("Многострочное поле", 5, 10);
// нестандартный шрифт и табуляция,автоматическая прокрутка
area1.setFont(new Font(" Times New Roman ", 0, 18)); // шрифт стиль размер
area1.setTabSize(10); // по умолчанию 8
JTextArea area2 = new JTextArea(15, 10);
area2.setLineWrap(true); //  автоматический перенос слов
area2.setWrapStyleWord(true); //слова, не умещающиеся в строке, будут целиком переноситься на строку новую
// 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");
// 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(«1.png"));
adjustLabel(l2);
l2.setHorizontalAlignment(LEFT);  // значок слева
contents.add(l2);
// надпись с нестандартным выравниванием
JLabel l3 = new JLabel("Текст и значок", new ImageIcon(«1.png"), 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(520, 500);
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();
}
}
```

Swng22

Всплывающие подсказки

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ToolTips extends JFrame {
public ToolTips() {
super("ToolTips");
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 подсказок кнопки2 (10 ,10 пикселей от верхнего левого угла)
}
public String getToolTipText(MouseEvent e) {    // условие  вывода первой подсказки кнопки 2
if ( e.getY() > 15 ) {
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();                                                    Swing12
}}
```

Редактируемая таблица

```
import javax.swing.*;
public class TableDefaultEditing extends JFrame {
private String[] columns = { "Имя", "Любимый Цвет" }; // заголовки столбцов
// данные для таблицы
private String[][] data = {
{ "Иван", "Зеленый" },
{ "Александр", "Красный"},
{ "Петр", "Синий"}
};
public TableDefaultEditing() {
super("TableDefaultEditing");
setDefaultCloseOperation(EXIT_ON_CLOSE);
JTable table = new JTable(data, columns); // создаем таблицу
// настраиваем стандартный редактор
JComboBox combo = new JComboBox(new String[] { "Зеленый", "Желтый", "Белый"});
// Редактор для таблицы использует поле комбинированного списка
DefaultCellEditor editor = new DefaultCellEditor(combo);
// Используется модель с настройкой столбцов TableColumnModel
table.getColumnModel().getColumn(1).setCellEditor(editor); // подсоединение редактора к колонке 1
// 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");
// 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);
// удаляем выделенную строку
dtm.removeRow(table.getSelectedRow());
}
});
// 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();
}
}
```