Лабораторная работа №12

Компоненты. Обработка событий

Компоненты

Компонент (*Component*) — это абстрактный класс, который инкапсулирует все атрибуты визуального интерфейса: обработка ввода с клавиатуры, управление фокусом, взаимодействие с мышью, уведомление о входе/выходе из окна, изменения размеров и положения окон, прорисовка своего собственного графического представления, сохранение текущего текстового шрифта, цветов фона и переднего плана.
Объекты класса Component будут далее называться *компонентами*.
В языке Java для разработки стандартных элементов пользовательского интерфейса (таких, как полосы прокрутки или раскрывающиеся меню) используется Abstract Windowing Toolkit (AWT) или Swing. Состав компонентов из пакета AWT похож на набор средств традиционных графических пользовательских интерфейсов (GUI).

Иерархия компонентов пользовательского интерфейса
В языке Java существует несколько классов для создания иерархической структуры элементов пользовательского интерфейса. Основа структуры - класс Component (Компонент), содержащий базовые методы для рисования на экране и обработки событий от клавиатуры или мыши. Большинство классов являются дочерними по отношению к Component.

На следующем уровне иерархии расположен подкласс **Container.**
Класс Container содержит подклассы **Panel** и **Window.**

Класс Panel предоставляет место для расположения других компонентов. Класс **Applet** (является подклассом класса Panel) - окно для вывода в браузере.

Класс `Window` имеет два подкласса:
`Frame` – для создания форм приложений или кадров в окне браузера.
`Dialog` – **для** создания диалоговых окон.

Компоненты представляют собой базовые блоки пользовательского интерфейса, их можно группировать и объединять в панели. В то же время панели сами являются компонентами, что позволяет "вкладывать" объекты друг в друга, создавая сложную иерархию. Структура AWT не налагает никаких ограничений на сложность интерфейса. Разумеется, глубоко вложенные структуры компонентов, кадров или панелей могут привести к дополнительным проблемам.

Создание компонентов
Компоненты создаются так же, как и другие объекты языка Java. Затем новый компонент следует поместить в панель или контейнер, отвечающий за его отображение на экране. Например, компонент ***`Label`*** («надпись») может быть создан так:

```
Label label1 = new Label ("Hi!!!");
```

Затем он может быть размещен на панели ***`panel1`*** следующим образом:

```
panel1.add(label1);
```

В классе Panel определены базовые функции размещения. По умолчанию компоненты выводятся в порядке поступления слева направо до правой границы окна. Далее происходит переход на новую линию по аналогии с новой строкой текста.
Компонент можно принудительно разместить (методом move()) в нужном месте, задав его координаты (x, y). Метод location() возвращает координаты компонента (в виде объекта Point). Метод locate() по заданным координатам возвращает компонент, расположенный в этом месте.

Кнопки
Из всех компонентов чаще всего применяются кнопки (button). При их создании следует запрограммировать реакцию на нажатие кнопки или другие события, связанные с нею. Конструктор Button имеет единственный параметр — строку, определяющую метку кнопки. Метод getLabel осуществляет доступ к метке кнопки, а с помощью setLabel метку можно изменить.
В приведенном ниже примере, создаются две кнопки пользовательского интерфейса, которые помещаются во фрейм.

```
import java.awt.*;
import javax.swing.JButton;
import javax.swing.JFrame;

public class Buttontest extends JFrame {

public Buttontest()
{
JButton b1,b2;
b1=new JButton("Wide");
add(b1);
b2=new JButton("Tall");
add(b2);
}
}
```

Флажки и переключатели
Флажок (checkbox) — это элемент интерфейса, который может находиться в одном из двух состояний: включен или выключен. Переключатели (radio buttons) — это группа флажков, из которых включенным может быть только один.
Базовый конструктор класса Checkbox - Checkbox(String s) имеет в качестве параметра строку (метку), которая выводится на экран справа от изображения флажка. Для изменения метки можно использовать методы getLabel() и setLabel(). Для установки состояния флажка применяется метод setState(Boolean state). При state=true флажок считается включенным, а при slate=false - выключенным.
Второй конструктор класса Checkbox имеет три аргумента:

```
Checkbox(String s, CheckboxGroup g, Boolean initialState)
```

Строка s определяет имя метки, а initialState — начальное состояние (включен или выключен).

Класс CheckboxGroup служит для группировки нескольких флажков в переключатель. Создание группы можно выполнить следующим образом:

```
CheckboxGroup talk=new CheckboxGroup();
add(new Checkbox("First", talk, false));
add(new Checkbox("Second", talk, true));
```

В примере создается переключатель из двух элементов.
Метод setCurrent() позволяет установить переключатель в нужное состояние. Элементы переключателя могут отображаться на экране в виде ромбиков или кружочков.

Приведем пример программы, которая создает флажки и выводит их на экран.
При передаче в конструктор вместо CheckboxGroup значения null будет создан флажок, а не переключатель.

```
Checkbox c1,c2,c3,c4;
CheckboxGroup g;

g=new CheckboxGroup();
c1=new Checkbox("First", g, false);
add(c1);
c2 = new Checkbox("Second", g, false);
add(c2);
g.setCurrent (c1);
c3 = new Checkbox("Free Checkbox");
add(c3);
c4 = new Checkbox("Also Free Checkbox", null, false);
add(c4);
```

Меню
В классе компонентов Choice имеется функция для создания нескольких вариантов раскрывающегося меню (pull-down menu). Конструктор Choice() не имеет параметров. Для включения нового элемента в меню требуется выполнить оператор addItem(String s). Пример программы:

```
Choice c;

c=new Choice();
c.addItem("Un");
c.addItem("Deux");
c.addItem("Trois");
c.addItem("Quatre");
add(c);
```

Метки и текстовые поля
Класс Labels позволяет вывести метку (надпись) в окне с применением трех различных типов выравнивания. Для ввода данных используется текстовое поле — компонент TextField.
Примеры создания надписей:

```
add(new Label("I'm automatically aligned on the left."));
add(new Label("I'm also left-aligned.", Label.LEFT));
add(new Label("I'm centered.", Label.CENTER));
add(new Label("I'm pushed to the right.", Label.RIGHT));
```

Полезные методы для работы с метками и текстовыми полями - getText() и setText() позволяют получить и преобразовать строку текста. Именно их следует использовать для изменения текста
Вывести текст на экран можно и с помощью метода drawString().
Текстовое поле TextField может содержать только одну строку текста. Для отображения нескольких строк применяют компонент TextArea.
Пример применения четырех различных конструкторов для создания текстового поля:

```
add (new TextField(12));
// Минимум 12 символов.
add (new TextField( "I'm a start up message."));
// Минимальный размер не задан.
add (new TextField( "I'm also a start up .", 28));
// Задан максимальный размер и начальное значение строки.
add (new TextField());
// Пустой конструктор.
```

Выводимые строки располагаются на экране (заданным по умолчанию методом размещения FlowLayout) последовательно слева направо.
Метод FlowLayout() предпочтителен при чередовании компонентов Label и TextField, чтобы они следовали друг за другом.
Последний конструктор не имеет параметров и создает маленький прямоугольник в конце второй строки. Размеры прямоугольника можно позднее изменить методами, заимствованными из суперкласса.
Метод getText() позволяет получить строку текста введенную в текстовое поле. Изменить текст можно с помощью метода setText().

Надписи
Функциональность класса Label сводится к тому, что он знает, как нарисовать объект String — текстовую строку, выровняв ее нужным образом. Шрифт и цвет, которыми отрисовывается строка метки, являются частью базового определения класса Component. Для работы с этими атрибутами предусмотрены пары методов getFont/setFont и getForeground/setForeground. Задать или изменить текст строки после создания объекта с помощью метода setText. Для задания режимов выравнивания в классе Label определены три константы — LEFT, RIGHT и CENTER. Ниже приведен пример, в котором создаются три метки, каждая — со своим режимом выравнивания.

```
import java.awt.*;
import javax.swing.JLabel;
import javax.swing.JFrame;

public class LabelDemo extends JFrame {
```

```
setLayout(null);

Label left = new Label("Left", Label.LEFT);
Label right = new Label("Right", Label.RIGHT);
Label center = new Label("Center", Label.CENTER);

add(left);
add(right);
add(center);
}
```

Размещение компонентов

Обычно программистам приходится указывать точное местоположение объекта на экране и при изменении размера окна пользовательского интерфейса соответствующим образом управлять размерами и расположением объектов. Java имеет специальные функции размещения, управляющие представлением объектов в окне.
Возложение на Java ответственности за размещение компонентов имеет определенные преимущества, поскольку Java старается наилучшим образом использовать свои возможности.
Автоматическое размещение имеет некоторые недостатки. Разработчику может потребоваться более полный контроль над объектами, например, когда на очень маленьких или очень больших экранах результаты будут выглядеть не вполне эстетично. В этом случае можно отменить автоматическое размещение и самостоятельно расположить элементы на экране.

Для решения задач автоматического размещения предусмотрены диспетчеры размещения (layout managers), реализующие интерфейс LayoutManager.
В Java есть несколько предопределенных классов — диспетчеров размещения, описываемых ниже.

FlowLayout
Класс FlowLayout реализует простой стиль размещения, при котором компоненты располагаются, начиная с левого верхнего угла, слева направо и сверху вниз. Если в данную строку не помещается очередной компонент, он располагается в левой позиции новой строки. Справа, слева, сверху и снизу компоненты отделяются друг от друга небольшими промежутками. Ширину этого промежутка можно задать в конструкторе FlowLayout. Каждая строка с компонентами выравнивается по левому или правому краю, либо центрируется в зависимости от того, какая из констант LEFT, RIGHT или CENTER была передана конструктору. Режим выравнивания по умолчанию — CENTER, используемая по умолчанию ширина промежутка — 5 пикселей.

BorderLayout
Класс BorderLayout реализует обычный стиль размещения для окон верхнего уровня, в котором предусмотрено четыре узких компонента фиксированной ширины по краям, и одна большая область в центре, которая может расширяться и сужаться в двух направлениях, занимая все свободное пространство окна. У каждой из этих областей есть строки-имена North (север), South (юг), East(восток) и West (запад), которые соответствуют четырем краям, а Center — центральной области. Ниже приведен пример BorderLayout с компонентом в каждой из названных областей.

```
setLayout(new BorderLayout());
int width = Integer.parseInt(getParameter("width"));
int height = Integer.parseInt(getParameter("height"));
add("North", new Button("This is across the top"));
add("South", new Label("The footer message might go here"));
add("East", new Button("Left"));
add("West", new Button("Right"));
String msg = "The reasonable man adapts " +
"himself to the world;\n" +
"the unreasonable one persists in " +
"trying to adapt the world to himself.\n" +
"Therefore all progress depends " +
"on the unreasonable man.\n\n" +
“George Bernard Shaw\n\n";
add("Center", new TextArea(msg));
```

GridLayout
Класс GridLayout размещает компоненты в простой равномерной сетке. Конструктор этого класса позволяет задавать количество строк и столбцов. Ниже приведен пример, в котором GridLayout используется для создания сетки 4x4, 15 квадратов из 16 заполняются кнопками, помеченными соответствующими индексами.

```
int n = 4;
setLayout(new GridLayout(n, n));
setFont(new Font("Helvetica", Font.BOLD, 24));
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
int k = i * n + j;
if (k > 0)
add(new Button("" + k));
```

Обработка событий

Модель обработки событий Java базируется на концепции слушателя событий. *Слушателем события* является объект, заинтересованный в получении данного события. В объекте, который порождает событие (в *источнике событий),* содержится список слушателей, заинтересованных в получении уведомления о том, что данное событие произошло, а также методы, которые позволяют слушателям добавлять или удалять себя из этого списка. Когда источник порождает событие (или когда объект источника зарегистрирует событие, связанное с вводом информации пользователем), он оповещает все объекты слушателей событий о том, что данное событие произошло.
Источник события оповещает объект слушателя путем вызова специального метода и передачи ему объекта события (экземпляра подкласса EventObject). Для того чтобы источник мог вызвать данный метод, он должен быть реализован для каждого слушателя. Это объясняется тем, что все слушатели событий определенного типа должны реализовывать соответствующий интерфейс. Например, объекты слушателей событий ActionEvent должны реализовывать интерфейс ActionListener. В пакете Java.awt.event определены интерфейсы слушателей для

каждого из определенных в нем типов событий (например, для событий MouseEvent здесь определено два интерфейса слушателей: MouseListener и MouseMotionListener). Все интерфейсы слушателей событий являются расширениями интерфейса java.util.EventListener. В этом интерфейсе не определяется ни один из методов, но он играет роль интерфейса-метки, в котором однозначно определены все слушатели событий как таковые.

В интерфейсе слушателя событий может определяться несколько методов. Например, класс событий, подобный MouseEvent, описывает несколько событий, связанных с мышью, таких как события нажатия и отпускания кнопки мыши. Эти события вызывают различные методы соответствующего слушателя. По установленному соглашению, методам слушателей событий может быть передан один единственный аргумент, являющийся объектом того события, которое соответствует данному слушателю. В этом объекте должна содержаться вся информация, необходимая программе для формирования реакции на данное событие. В таблице 6 приведены определенные в пакете java.awt.event типы событий, соответствующие им слушатели, а также методы, определенные в каждом интерфейсе слушателя.

Таблица. Типы событий, слушатели и методы слушателей в Java

Класс события	*Интерфейс слушателя*	*Методы слушателя*
ActionEvent	ActionListener	actionPerformed()
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged()
ComponentEvent	ComponentListener	componentHidden() componentMoved() componentResized() Cor”componentShown()
ContainerEvent	ContainerListener	componentAdded() componentRemoved()
FocusEvent	FocusListener	focusGained() focusLost ()
ItemEvent	ItemListener	itemStateChanged()
KeyEvent	KeyListener	keyPressed() keyReleased() keyTyped()
MouseEvent	MouseListener	mouseClicked() mouseEntered() mouseExited() mousePressed() mouseReleased()
	MouseMotionListener	mouseDragged() mouseMoved()
TextEvent	TextListener	textValueChanged()
WindowEvent	WindowListener	windowActivated() windowClosed() windowClosing() windowDeactivated() windowDeiconified() windowlconified() windowOpened()

Для каждого интерфейса слушателей событий, содержащего несколько методов, в пакете java.awt.event определен простой класс-адаптер, который обеспечивает пустое тело для каждого из методов соответствующего интерфейса. Когда нужен только один или два таких метода, иногда проще получить подкласс класса-адаптера, чем реализовать интерфейс самостоятельно. При получении подкласса адаптера требуется лишь переопределить те методы, которые нужны, а при прямой реализации интерфейса необходимо определить все методы, в том числе и ненужные в данной программе. Заранее определенные классы-адаптеры называются так же, как и интерфейсы, которые они реализуют, но в этих названиях Listener заменяется на Adapter: MouseAdapter, WindowAdapter и т.д.

Как только реализован интерфейс слушателя или получены подклассы класса-адаптера, необходимо создать экземпляр нового класса, чтобы определить конкретный объект слушателя событий. Затем этот слушатель должен быть зарегистрирован соответствующим источником событий. В программах пакета AWT источником событий всегда является какой-нибудь элемент пакета. В методах регистрации слушателей событий используются стандартные соглашения об именах: если источник событий порождает события типа X, в нем существует метод addXListener () для добавления слушателя и метод removeXListener() для его удаления. Одной из приятных особенностей модели обработки событий Java 1.1 является возможность легко определять типы событий, которые могут порождаться данным элементом. Для этого следует просто просмотреть, какие методы зарегистрированы для его слушателя событий. Например, из описания API для объекта класса Button следует, что он порождает события ActionEvent. В таблице ниже приведен список элементов пакета AWT и событий, которые они порождают.

Таблица. Элементы пакета AWT и порождаемые ими события

Элемент	Порождаемое событие	Значение
Button	ActionEvent	Пользователь нажал кнопку
CheckBox	ItemEvent	Пользователь установил или сбросил флажок
CheckBoxMenuItem	ItemEvent	Пользователь установил или сбросил флажок рядом с пунктом 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

Задание 1.

Ниже приведен пример программы-заготовки для калькулятора. В ней реализована только одна операция сложения целых чисел. Необходимо дополнить ее кнопками-цифрами (в программе вставлены только три кнопки для цифр 1, 2, 3), а также кнопками для выполнения остальных арифметических действий и кнопкой - десятичной точкой для ввода действительных чисел.
Обработать события получения и потери фокуса текстовым окном.

```
package swing;

import java.awt.BorderLayout;

import java.awt.Color;
import java.awt.GridLayout;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JTextField;


public class Calc extends JFrame implements ActionListener{
      float num;
      float buf;
      int code;
      JTextField name1;
      JLabel label2;
      JButton but[];
      boolean point;
      String str="0";
      public Calc() {
            super ("Калькулятор");//название фрейма
            //или заголовок фрейма можно задать еще и так: setTitle("Калькулятор");
            num=0;
          buf=0;
          code=0;
          point=false;

            //при выходе из формы завершить и программу
            setDefaultCloseOperation(EXIT_ON_CLOSE);

            //создана панель, которая нужна для размещения на ней дисплея для ввода чисел
             JPanel panelNorth=new JPanel(new BorderLayout());
            //панель будет расположена на форме прижатой к ее верхнему краю
             add(panelNorth,BorderLayout.NORTH);

      // на панели panelNorth создано текстовое окно для ввода чисел
             name1 = new JTextField(10);
      // текст окно name1 расположено по всей области панели panelNorth
         panelNorth.add(name1,BorderLayout.CENTER);
      // цвет фона текстового окна
         name1.setBackground(Color.yellow);


      // создана еще одна панель, которая нужна для размещения на ней кнопок в виде сетки
         JPanel pn=new JPanel(new GridLayout(3,2,5,0));
      // панель pn будет расположена на форме так
         add(pn,BorderLayout.CENTER);

      // цвет фона панели кнопок
              pn.setBackground(Color.BLUE);

      // всего будет 17 кнопок
              but=new JButton[17];

          // ниже каждая кнопка будет помещена на панель pn в виде сетки

        //создана первая кнопка
            but[0]=new JButton("0");
        // подключение к первой кнопке слушателя
            but[0].addActionListener((ActionListener) this);
         // первая кнопка помещена на панель pn
            pn.add(but[0]);
        //создана вторая кнопка
          but[1]=new JButton("1");
        // подключение ко второй кнопке слушателя
          but[1].addActionListener((ActionListener) this);
        // вторая кнопка помещена на панель pn
```

```
        pn.add(but[1]);

        but[2]=new JButton("2");
        but[2].addActionListener((ActionListener) this);

        pn.add(but[2]);

        but[3]=new JButton("3");
        but[3].addActionListener((ActionListener) this);

        pn.add(but[3]);

        but[11]=new JButton("+");
        but[11].addActionListener((ActionListener) this);
        pn.add(but[11]);

        but[13]=new JButton("=");
        but[13].addActionListener((ActionListener) this);
        pn.add(but[13]);

        but[16]=new JButton("C");
        but[16].addActionListener((ActionListener) this);
        pn.add(but[16]);



    // установить размеры фрейма
          setSize(200,200);
    // сделать фрейм видимым
                setVisible(true);
          }

    // обработка событий щелчков на кнопках

           public void actionPerformed(ActionEvent e){

                num=new Float(str).floatValue();

                if(e.getSource()==but[0])zero();

                if(e.getSource()==but[1])chislo("1");
                if(e.getSource()==but[2])chislo("2");
                if(e.getSource()==but[3])chislo("3");

                if(e.getSource()==but[13])result();
                if(e.getSource()==but[16])sbros();

                if(e.getSource()==but[11])
                      {code=2;buf=num;str="0";}// операция + имеет код 2

          }


    private void zero(){
        if(str!="0")str=str+"0";
        name1.setText(str);
       }

    private void chislo(String s){
          if(str=="0")str=s;
        else str=str+s;
        name1.setText(str);
        }

    private void result()
      {
              num=new Float(str).floatValue();
              if(code==1)num=buf-num;
              if(code==2)num=buf+num;
              if(code==3)num=buf*num;
              if(code==4 && num!=0)num=buf/num;
              else
                {
                str="делить на ноль нельзя";
                //здесь надо вывести это сообщение каким-то образом
                }
                str=str.valueOf(num);
              name1.setText(str);
          }

    private void sbros(){
                    num=0;
                    buf=0;
                    code=0;
                    str="0";
                    point=false;
                    name1.setText(str);
                    }

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