Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Локтионова Оксана Геннадьевна
Должность: проректор по учебной работе
Дата подписания: 08.09.2026 03:52:36
Уникальный программный ключ:
0b817ca911e6668abb13a5d426d39e5f1c11eabbf73e943df4a4851fda56d089

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

Утверждаю
Проректор по учебной работе
О.Г. Локтионова
«25» 09 2023 г

ЯЗЫК JAVA

Методические рекомендации по выполнению самостоятельных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр)

Курск 2023

УДК 621.(076.1)

Составители: А.А.Кузьмин

Рецензент:
Кандидат технических наук, доцент *Т.Н. Конаныхина*

Язык Java: методические рекомендации по выполнению самостоятельных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр) / Юго-Зап. гос. ун-т; сост.: А.А.Кузьмин. - Курск, 2023. - 32 с.

Содержат методические рекомендации к проведению самостоятельных работ по дисциплине «Язык Java». Методические указания по структуре, содержанию и стилю изложения материала соответствуют методическим и научным требования, предъявляемым к учебным и методическим пособиям.

Предназначены для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр)

Текст печатается в авторской редакции

Подписано в печать 25.09.23 Формат 60x84 1/16
Усо.печ.л. 1,9 . Уч.-изд.л. 1,9 .Тираж 30 экз. Заказ: 1075 .Бесплатно.
Юго-Западный государственный университет.
305040. г. Курск, ул. 50 лет Октября, 94.

Самостоятельная работа №1

Введение в разработку графического интерфейса пользователя

1. Цель работы: Введение в программирование графического интерфейса. Операции ввода-вывода и взаимодействия с пользователем с использованием класса JoptionPane, статические методы JoptionPane.

2 Краткие теоретические сведения

Графический пользовательский интерфейс (GUI) представляет собой удобный для пользователя механизм взаимодействия с приложением. Графический интерфейс (произносится как «GOO-ее») дает приложению отличительный «внешний вид». Графические интерфейсы создаются из компонентов GUI. Иногда они называются элементами управления или виджетами - для оконных гаджетов. Компонент GUI - это объект, с которым пользователь взаимодействует с помощью мыши, клавиатуры или другой формы ввода. Многие IDE (программы, которые позволяют графический интерфейс проектирования) предоставляют инструменты проектирования графического интерфейса, с помощью которых вы можете указать точный размер и местоположение компонента визуально. IDE генерирует для вас код GUI. Хотя это значительно упрощает создание графических интерфейсов, вы не можете понять все свойства и события компонентов. По этой причине мы написали GUI-код руками.

При разработке Java-программы важно выбрать соответствующие компоненты графического интерфейса пользователя (GUI) Java. Есть два основных набора компонентов, с которыми вы, скорее всего, будете строить свои Java-программы. Эти две группы компонентов называются абстрактным инструментарием окна (AWT) и Swing. Обе эти группы компонентов являются частью Java Foundation Classes (JFC). Оба пакета содержат много классов, которые позволяют создавать графические интерфейсы в ваших программах. Чтобы использовать его, вы должны открыть соответствующие пакеты: import javax.swing. *; Для компонентов siwng и импорта java.awt. *; Для компонентов awt.

Какие компоненты лучше? Ниже мы обобщили оба пакета для описания преимуществ и недостатков каждого из них.

Обзор AWT

AWT означает абстрактное окно ToolKit. Инструмент Abstract Window Toolkit поддерживает графическое программирование на Java. Это портативная графическая библиотека для автономных приложений и / или апплетов. Инструментарий Abstract Window Toolkit обеспечивает соединение между вашим приложением и собственным GUI. AWT обеспечивает высокий уровень абстракции для вашей Java-программы, поскольку он скрывает вас от базовых деталей графического интерфейса, в котором будет работать ваша программа. AWT включает в себя: богатый набор компонентов пользовательского интерфейса; Надежная модель обработки событий; Графические и визуальные инструменты.

Swing реализует набор компонентов GUI, которые основаны на технологии AWT и обеспечивают удобный внешний вид. Возможности Swing включают в себя: все функции AWT, 100% чистые Java сертифицированные версии существующего набора компонентов AWT (Button, Scrollbar, Label и т.д.); Богатый набор компонентов более высокого уровня (например, древовидный вид, окно списка и панели с вкладками). Компоненты Swing не зависят от операционной системы для обеспечения их функциональности. Таким образом, эти компоненты часто называют «легкими» компонентами.

В общем, компоненты AWT подходят для простой разработки или разработки апплетов, ориентированных на определенную платформу (например, программа Java будет работать только на одной платформе).

Для большинства других разработок Java GUI вы захотите использовать компоненты Swing. Также обратите внимание, что компоненты Oracle NetBeans IDE и Borland с добавленной стоимостью, включенные в JBuilder, например, dbSwing и JBCL, основаны на компонентах Swing, поэтому, если вы хотите использовать эти компоненты, вы захотите основать свою разработку на Swing.

JOptionPane (пишется как есть) является частью java swing library. Для этого требуется оператор импорта в верхней части программы. Вот что такое утверждение:

```
import javax.swing.JOptionPane;
or
import javax.swing.*;
```

Статические методы в классе JOptionPane позволяют легко создавать модальные диалоги для отображения сообщений JOptionPane.showMessageDialog, чтобы рассказать пользователю о чем-то, что произошло;

JOptionPane.showConfirmDialog, чтобы задать подтверждающий вопрос, например yes / no / cancel.

JOptionPane.showInputDialog, чтобы ввести некоторую информацию, такую как текст или цифры;

JOptionPane.showOptionDialog, чтобы выбрать один из множества кнопок;

Каждый из этих методов возвращает int, определяющую, какая кнопка была нажата, или String, определяющая выбранную опцию.

Подробное описание всех статических методов класса JOptionPane, расположенного ниже:

JOptionPane.showMessageDialog (parentComponent, message); Отображает модальное диалоговое окно с одной кнопкой, которая помечена как «OK». Вы можете легко указать сообщение, значок и название, отображаемое диалоговым окном. В простейшем случае метод принимает два параметра: parentComponent (или null вместо него) и String constant «message», который является отображаемым текстом. Первый параметр «parentComponent» определяет компонент, который должен быть родителем этого диалогового окна. Он используется двумя способами: кадр, который содержит его, используется в качестве родителя кадра для диалогового окна, а его координаты экрана используются при размещении диалогового окна. В общем случае диалоговое окно размещается чуть ниже компонента. Этот параметр может быть нулевым, и в этом случае в качестве родителя используется кадр по умолчанию, и диалог будет центрироваться на экране.

Пример: кода

```
JOptionPane.showMessageDialog(
null, "This is text Message \n to be displayed");
```

Дает вам сообщение, представленное ниже:

Вы можете выделить текст, используя одну строку или много строк, используя \ n разделитель. Также вы можете настроить заголовок сообщения и значок, который будет представлен на нем. В следующем примере используется полный синтаксис метода:

```
JOptionPane.showMessageDialog(
null,"My                                    message","My
title",JOptionPane.WARNING_MESSAGE);
```

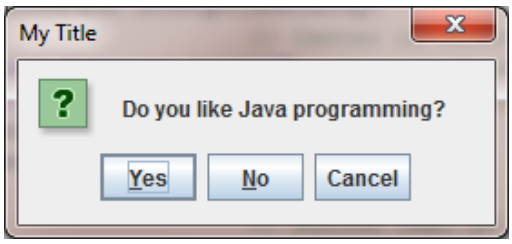
Создает сообщение с текстом «My message», заголовок «My title» и значок «!». Согласно параметру

Таблица 6.1. Связь между последним параметром showMessageDialog и видом значка

Parameter	**Icon**
JOptionPane.ERROR_MESSAGE	
JOptionPane.WARNING_MESSAGE	
JOptionPane.INFORMATION_MESSAGE	
JOptionPane.QUESTION_MESSAGE	
JOptionPane. PLAIN_MESSAGE	none

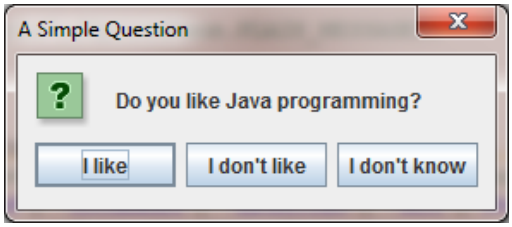
Метод JOptionPane.showConfirmDialog может использоваться для запроса некоторого вопроса от пользователя. В приведенном ниже коде показан простой пример:

```
int n = JOptionPane.showConfirmDialog(null,
"Do you like Java programming?","My Title",
JOptionPane.YES_NO_CANCEL_OPTION);
```



Код создает диалог, пользователь может нажать кнопку «Да», выход этого метода (переменная «n») в этом случае будет равен нулю. Если пользователь нажимает «Нет», значение n будет равно 1. Если пользователь нажимает «Отмена», значение «n» будет равно 2, в противном случае, если пользователь нажимает клавишу «ESC» или нажимает X, n будет равным -1, который указывает на ситуацию, когда ни одна из кнопок не нажата. Последний параметр метода содержит стандартный набор кнопок (например, да, нет), но если вы хотите, вы можете создать несколько нестандартных кнопок, как в примере ниже:

```
String[] buttons = {"I like", "I don't like", "I don't know"};
int n = JOptionPane.showOptionDialog(null,
"Do you like Java programming?",//Message
"A Simple Question",//Title
JOptionPane.YES_NO_OPTION,//Will be ignored
JOptionPane.QUESTION_MESSAGE,//Icon
null,      //no custom icon
buttons,   //the titles of buttons
buttons[0]); //default button title
```

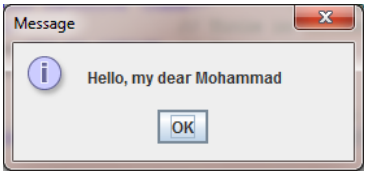


Как вы можете видеть, текст внутри кнопок генерируется в соответствии с данными строковых массивов «Кнопки». По аналогичному методу вы можете создать любое количество кнопок, содержащих любой текст. Выходной сигнал метода будет присвоен переменной «n», которая содержит число нажатой кнопки (отсчитывается с нуля) или -1, если ни одна из кнопок не нажата.

Метод JOptionPane.showInputDialog может использоваться для чтения любых данных от пользователя. Это может быть набор символов или цифр. Следующий пример иллюстрирует простую программу, которая сначала спрашивает пользователя о его имени, а затем показывает приветственное сообщение:

```
import javax.swing.*;
public class Test
{
static public void main(String[]  args)
 {String   st=JOptionPane.showInputDialog("What   is
your name?");
  JOptionPane.showMessageDialog(null,"Hello  my  dear
"+st);
 }
}
```

Результат этой программы для ввода «Mohammad» представлен ниже:



Несколько раз вы можете дать пользователю сделать выбор из многих вопросов. В этом случае вы можете использовать другой синтаксис метода showMessageDialog. Следующая программа показывает пользователю список имен, затем просит выбрать любое любимое имя, а затем показывает, какое имя выбрано.

```
import javax.swing.*;
public class Test
{static public void main(String[]  args)
 {
  String[]  questions  =  {"Mohammad",  "Abdallah",
"Ahmad", "Hakeem"};
  //questions is array of String
  String st;//st is String variable
  st=(String)JOptionPane.showInputDialog(null,
```

```
  "Choose your favorite name",//Text of dialog
  "Customized Dialog",//Title
  JOptionPane.ERROR_MESSAGE,//Icon
  null,//no custom icon
  questions,//array of String
  questions[0]);//Default question
  JOptionPane.showMessageDialog(null,st);
 }
}
```

Результат этой программы представлен ниже:

Последним важным методом является JOptionPane.showOptionDialog, наиболее универсальный по сравнению с предыдущими методами. Этот метод отображает модальный диалог с указанными кнопками, значками, сообщением, заголовком и т. Д. С помощью этого метода вы можете изменить текст, который появляется на кнопках стандартных диалогов. Вы также можете выполнять многие другие виды настройки.

Следующий пример иллюстрирует этот метод

```
String[] options = {"Yes, please","No, thanks","I don't like Java!!!"};
int n = JOptionPane.showOptionDialog(null,
"Do you want new books about Java?",//Message
"A Silly Question",//Title
JOptionPane.YES_NO_CANCEL_OPTION,//default buttons ignored
JOptionPane.QUESTION_MESSAGE,//Icon
null,//Custom icon ignored
options,//Array of String
options[2]);//Default answer
```

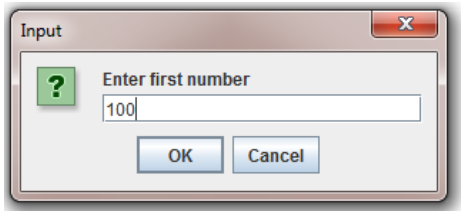
Результат этой программы представлен ниже:

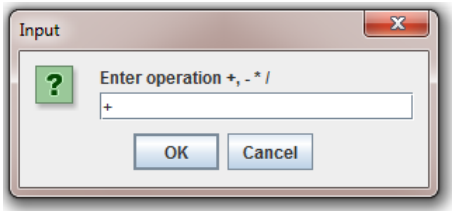
Согласно клику пользователя значение переменной «n» будет равно 0,1,2 или -1, если ни одна из кнопок не выбрана.

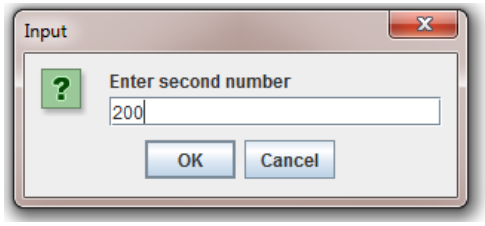
Следующая страница руководства представляет собой задание для этой лабораторной работы.

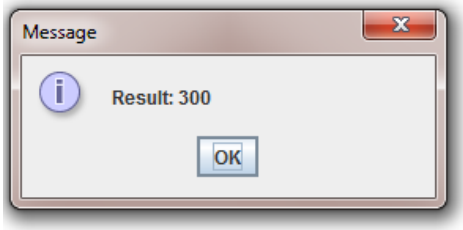
3 Задание для работы.

Сделать простой калькулятор Java. Калькулятор - это программа, которая позволяет пользователю вводить первое число, затем математическую операцию (+, -, *, /), затем второй оператор затем вычисляет и показывает результат:









Подсказка: преобразование из String в int

```
String st="5";
int i=Integer.ValueOf(st);
```

Преобразование из int в String

```
int i=123;
String st1=String.valueOf(i);
```

4 Содержание отчета

1. Титульный лист
2. Задание на работу
3. Листинг программы
4. Скриншот работы приложения

Самостоятельная работа №2

Компоненты Java и модель делегирования событий

1 Цель работы: Создание приложений форм с использованием компонентов JAVA. Обработка событий и событий с использованием интерфейсов Java.

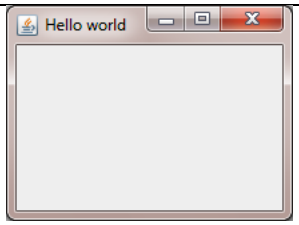
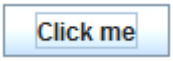
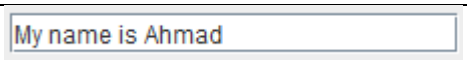
2 Краткие теоретические сведения

Java включает библиотеки, обеспечивающие многоплатформенную поддержку объектов графического интерфейса пользователя. «Мультиплатформенный» аспект заключается в том, что вы можете написать программу для разных операционных систем, таких как Linux, Unix, Mac OS, и у вас появятся одни и те же графические объекты, когда программа будет запущена в UNIX или Windows или Mac OS. Компоненты GUI Java включают метки, текстовые поля, текстовые области, кнопки, всплывающие меню и т.д. В SWING Toolkit также входят контейнеры, которые могут включать эти компоненты. Контейнеры включают фреймы (окна или формы), холсты (которые используются для рисования) и панели (которые используются для группировки компонентов). Панели, кнопки и другие компоненты могут быть размещены либо непосредственно в кадрах, либо в панелях внутри фреймов. Эти компоненты GUI автоматически рисуются всякий раз, когда отображается окно, в котором они находятся. Таким образом, нам не нужно явно вводить команды для их рисования. Действия с этими компонентами GUI обрабатываются с использованием модели событий Java. Когда пользователь взаимодействует с компонентом (щелкает по кнопке, вводит в поле, выбирает из всплывающего меню и т.д.), Событие генерируется компонентом, с которым вы взаимодействуете. Для каждого компонента вашей программы программист должен указывать один или несколько объектов для «прослушивания» событий из этого компонента. Если обнаружено какое-либо событие, код, связанный с этим событием, должен выполняться автоматически. Название кода, которое будет выполнено в случае какого-либо события, - это «обработчик событий».

Остальная часть этой лабораторной работы обсуждает некоторые компоненты GUI, которые позволяют разработчикам приложений

создавать полную программу. В таблице 1 показаны несколько базовых компонентов GUI Swing, которые мы обсуждаем сегодня.

Таблица 1. Список компонентов графического интерфейса пользователя

Название компонента	**Описание**	**Внешний вид**
JFrame	Это класс-контейнер, база программы, окно, содержащее все остальные компоненты программы. В .NET (Microdoft Visual Studio) аналогичный класс имеет имя «Форма».	
JButton	JButton - это класс в пакете javax.swing, который представляет кнопки на экране. Этот компонент запускает событие при щелчке мышью. Аналогичный компонент в .NET имеет имя «Button».	
JLabel	Отображает неотредактируемый текст и / или значки. В .NET подобный класс имеет имя "Label"	
JTextField	Обычно получает вход от пользователя, строку текста. В .NET подобный компонент имеет имя "TextBox"	

Компонент JFrame является базой программы, он выглядит как окно, в котором расположены кнопки, метки и другие компоненты. Размер JFrame, цвет, местоположение можно контролировать с помощью

свойств и методов JFrame. Свойства - это некоторые переменные внутри класса, которые отвечают за его внешнее представление. Методы - это команды, которые должны выполняться объектом класса. Наиболее популярными методами класса JFrame являются:

Show(); - показать форму на экране;

GetWidth (); - этот метод возвращает ширину формы, количество пикселей;

SetSize (int, int); - настроить новый размер формы. Два целочисленных параметра описывают соответственно ширину и высоту формы;

GetHeight (); - этот метод возвращает фактическую ширину формы, представленной в пикселях;

SetLocation (int, int); - установить новое положение формы на экране, два целочисленных параметра - X и Y формы;

SetTitle (string); - этот метод изменит название формы, как показано в таблице 1, название «Hello world».

Наиболее популярные свойства и методы компонентов JButton, JLabel, JTextFiel:

GetText (); - метод, возвращающий текст, отображаемый компонентом;

SetText ("String object"); Определяет текст, который будет отображаться компонентом;

SetCursor (Cursor cursor); Определяет форму курсора, когда указатель мыши будет находиться в границах компонента;

setLocation (x, y); Определяет новое положение компонента в форме согласно параметрам x и y. Этот метод действителен, если ранее не были определены макеты. Тема о макетах (распределение компонентов в форме) будет поднята в следующей лабораторной работе.

getLocation (); Возвращает точку (переменная с типом данных «point»), которая содержит информацию о местоположении компонента, x и y

Все другие методы будут рассмотрены в будущем.

Теперь мы можем создать наше первое графическое приложение, как уже упоминалось ранее, JAVA-программа является дочерней (подклассом) класса JFrame, чтобы создать форму, которую вы должны создать подкласс класса JFrame. Следующий пример иллюстрирует создание окна вашей программы:

```
import java.awt.*;
```

```
import javax.swing.*;

class MyFrame extends JFrame
{
 public MyFrame()
 { //Default constructor of your class
   //you can place any code here
 }
}

public class Example
{ //The class-driver
public static void main( String[] args )
 {
  MyFrame f = new MyFrame(); // create object of the
class
  f.setDefaultCloseOperation(   JFrame.EXIT_ON_CLOSE
);
  f.setSize(250,150);//Set size of the frame
  f.setVisible( true ); // display frame
 } // end main
}
```

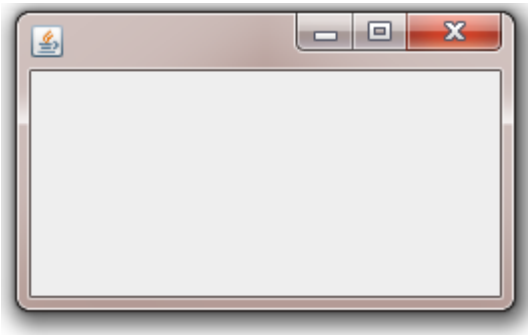


Рисунок 1. Форма (окно), созданная кодом

Во-первых, в программе мы создали класс «MyFrame», используя наследование от класса «JFrame». Затем в основном методе класса «Example» был создан объект «f» класса «MyFrame», этот объект является нашей программой. Вы можете управлять размером формы, используя метод setSize (ширина, высота), заголовок, используя метод

setTitle («String object»), положение в соответствии с набором командLocation (x, y) и многими другими функциями вашей программы. инструкция

```
f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
```

Требуется, чтобы виртуальная машина Java закрывала программу в случае нажатия X формы или нажатия клавиш ALT + F4.

Потому что любая профессиональная программа содержит некоторые другие компоненты, выделенные в Форме. В следующем примере показано, как создать программу, содержащую форму (класс JFrame), две кнопки (JButton), метку (JLabel) и поле для ввода текста (JTextField). (JButton), label (JLabel) and field to type a text (JTextField).

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

class MyFrame extends JFrame
{
    JButton button1=new JButton("Click me");//1
    JButton button2=new JButton("Hello world");//2
    JLabel label1=new JLabel("This is a label");//3
    JTextField text1=new JTextField("Type text
here",20);//4
 public MyFrame()
 {  super("Hello world");//5
    setLayout(new FlowLayout());//6
    this.add(label1);//7
    this.add(text1);//8
    this.add(button1);//9
    this.add(button2);//10
 }
}

public class Example
```

```
{ //The class-driver
public static void main( String[] args )
{
  MyFrame f = new MyFrame(); // create object of the
class
  f.setDefaultCloseOperation(   JFrame.EXIT_ON_CLOSE
);
  f.setSize(250,150);//Set size of the frame
  f.setVisible( true ); // display frame

 } // end main
}
```

Рисунок 2. Графический интерфейс, созданный программой

Все важные строки программы нумеруются с использованием комментариев. Строки 1-4 иллюстрируют определение и создание кнопок, меток и текстовых полей. Строка 5, расположенная внутри конструктора класса, вы можете увидеть инструкцию super («Hello words»), которая используется здесь для определения названия формы («Hello world»). Строка 6 setLayout (новый FlowLayout ()) создает раскладку кадра. Макет - это метод, описывающий, как компоненты будут распределены по форме. Планирование потока означает, что все компоненты распределены слева направо и сверху вниз. Детали макетов будут обсуждаться в следующей лабораторной работе. Строки 7-10 иллюстрируют инструкции по добавлению компонентов в форму:
This.add («Название компонента»); Распределение компонентов, показанных на рисунке 7.2.
Теперь пользователи вашей программы могут вводить текст внутри текстового поля или кнопки, но никаких действий, связанных с нажатием кнопки, не происходит. Другими словами у нас нет

обработчиков событий для нажатия кнопки. Это означает, что очень важной частью графического интерфейса является программирование обработчиков событий.

Событие - это действие, обнаруженное программой. Существует много разных действий, например, нажатие кнопки мыши или перемещение мыши внутри формы или ввод текста с клавиатуры. Иногда некоторые события требуют выполнения некоторого Java-кода. В этом случае код должен быть выделен в специальной функции, которая имеет имя «Обработчик событий». Как правило, в Java-программах существует два способа создания обработчика событий: использование интерфейсов и использование адаптеров классов. Сегодня мы покажем вам, как обрабатывать события с помощью интерфейсов. Интерфейс в Java - это класс, который содержит только конечные переменные и абстрактные методы. В Java существует много интерфейсов, предназначенных для обработки событий, потому что существует много групп событий. Ниже мы покажем вам наиболее популярные интерфейсы, которые могут быть полезны для вас в будущем.

Интерфейс **ActionListener** находится в пакете java.awt.event. Этот интерфейс позволяет обнаруживать действия, связанные с нажатием кнопки. Класс, который заинтересован в обработке события события, реализует этот интерфейс, а объект, созданный с помощью этого класса, регистрируется компонентом с использованием метода «addActionListener» компонента. Когда происходит событие действия, вызывается метод «actionPerformed» этого объекта:

Public void actionPerformed (ActionEvent e);

Интерфейс **KeyListener** расположен в пакете java.awt.event. *. Этот интерфейс предназначен для приема нажатий клавиш. Интерфейс содержит четыре метода, которые должны быть реализованы внутри вашего класса.

Void keyTyped (KeyEvent e) Вызывается при вводе ключа

Void keyPressed (KeyEvent e) Вызывается при нажатии клавиши.

Void keyReleased (KeyEvent e) Вызывается, когда ключ был освобожден.

Интерфейс **MouseListener.** Этот интерфейс прослушивателя для получения «интересных» событий мыши (нажмите, отпустите, нажмите, введите и выйдите) на компонент. Ваш класс должен реализовать 5 методов, если вы хотите использовать этот интерфейс:

Void mouseClicked (MouseEvent e); Вызывается, когда кнопка мыши была нажата (нажата и отпущена) на компоненте.

Void mousePressed (MouseEvent e); Вызывается, когда мышь нажата и удерживается

Void mouseReleased (MouseEvent e); Вызывается, когда мышь выпущена

Void mouseEntered (MouseEvent e); - Вызывается, когда мышь входит в компонент.

Void mouseExited (MouseEvent e); - Вызывается, когда мышь выходит из компонента.

Интерфейс **MouseMotionListener.** Этот интерфейс для приема событий движения мыши на компоненте. Для кликов и других событий мыши используйте интерфейс «MouseListener». Затем объект-слушатель, созданный из этого класса, регистрируется компонентом с использованием метода «addMouseMotionListener» компонента. Событие движения мыши генерируется, когда мышь перемещается или перетаскивается. (Многие такие события будут сгенерированы). Когда происходит событие движения мыши, вызывается соответствующий метод в объекте прослушивателя, и ему передается «MouseEvent». Этот интерфейс содержит два метода:

Void mouseDragged (MouseEvent e); Вызывается, когда кнопка мыши нажата на компоненте, а затем перетаскивается.

Void mouseMoved (MouseEvent e); Вызывается, когда курсор мыши перемещен на компонент, но кнопки не были нажаты.

Следующий пример иллюстрирует программу, которая позволяет нажимать кнопки1 и кнопку2. Если нажата кнопка1, в окне отображается сообщение «Button1 clicked», если кнопка2 нажата, на метке появится сообщение «Button2 clicked».

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

class MyFrame extends JFrame implements
ActionListener
{
 JButton button1=new JButton("Click me");//1
 JButton button2=new JButton("Hello world");//2
 JLabel label1=new JLabel("This is a label");//3
 JTextField text1=new JTextField("Type text
here",20);//4
 public MyFrame()
 {super("Hello world");//5
  setLayout(new FlowLayout());//6
  this.add(label1);//7
  this.add(text1);//8
  this.add(button1);//9
  this.add(button2);//10
  button1.addActionListener(this);//11
  button2.addActionListener(this);//12
 }
public void actionPerformed(ActionEvent event) //13
{if (event.getSource()==button1)//14
   label1.setText("Button1 clicked");//15
 if (event.getSource()==button2)//16
   label1.setText("Button2 was clicked");//17
 text1.setText(event.getActionCommand());//18
}
}

public class Example
{ //The class-driver
public static void main( String[] args )
 {
    MyFrame f = new MyFrame(); // create object of
the class
    f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE
);
    f.setSize(250,150);//Set size of the frame
    f.setVisible( true ); // display frame

 } // end main
}
```

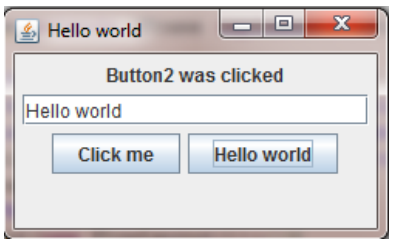


Рисунок 3. Внешний вид программы и выполнение обработчиков событий

В приведенной выше программе содержится обработчик событий, связанный с интерфейсом «ActionListener». Внутри интерфейса есть один метод actionPerformed, который будет выполняться автоматически, если кнопка будет нажата мышью. Строки № 11 и 12 предоставляют регистрацию обработчика событий для кнопок 1 и 2 кнопок. Внутри обработчика событий с именем actionPerformed вы должны распознать объект, который генерирует это событие. Это может быть кнопка1 или кнопка2. Обработчик событий содержит параметр «событие», поле «getSource ()» этого параметра возвращает объект, который выполняет это событие. Строки 14 и 16 сравнивают «getSource ()» с кнопкой 1 и кнопкой2, если условие становится равным «true», объект распознается. Кроме того, вы можете использовать поле getActionCommand (), чтобы получить заголовок нажатой кнопки.

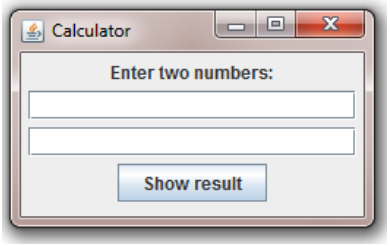
3 Задание для работы.

1. В приведенном ниже коде представлена простая Java-игра под названием «виртуальная лягушка». Форма пуста, но если вы попытаетесь щелкнуть ее мышью, она перейдет в случайную позицию. Класс «MyFrame» реализует интерфейс MouseMotionListener, который позволяет обнаруживать движение мыши. Интерфейс содержит два метода mouseDragged и mouseMoved, оба метода должны быть реализованы в классе, даже если вы не используете какой-либо метод. Протестируйте эту программу с помощью Eclips.

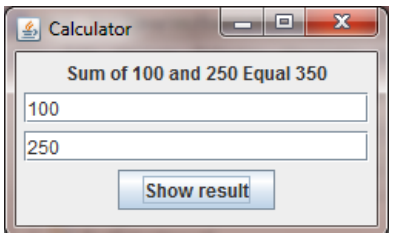
```
import javax.swing.*;
import java.awt.event.*;

class MyFrame extends JFrame implements
MouseMotionListener
```

```
{
 public MyFrame()
 {    super("Virtual frog");//5
      this.addMouseMotionListener(this);
 }
public void mouseDragged(MouseEvent arg0) {
}

public void mouseMoved(MouseEvent arg0) {
double x=Math.random()*800;
double y=Math.random()*600;
this.setLocation((int)x, (int)y);
}
}

public class Example
{public static void main( String[] args )
 {
     MyFrame f = new MyFrame(); // create object of
the class
     f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE
);
     f.setSize(250,150);//Set size of the frame
     f.setVisible( true ); // display frame

 } // end main
}
```

2. Создайте простой калькулятор Java, графический интерфейс программы, представленный ниже.



В программе есть метка, которая указывает сообщение «Введите два номера», два поля (JTextField) и кнопку «Показать результат». Пользователь может набирать числа в текстовых полях, затем нажимает кнопку, программа вычисляет сумму и показывает результат в метке, см. Рисунок ниже:



4 Содержание отчета

1. Титульный лист
2. Задание на работу
3. Листинг программы
4. Скриншот работы приложения

Самостоятельная работа №3

Графический дизайн пользовательского интерфейса и адаптеры класса

1. Цель работы: Создание обработчиков событий с адаптерами классов; Отношения между адаптерами классов и интерфейсами. Обзор компонентов JRadioButton и JCheckBox.

2 Краткие теоретические сведения

Во время последней работы обработчики событий в вашей программе были созданы с использованием интерфейсов Java. Любой интерфейс - это класс, содержащий только абстрактные методы; Это означает, что вы должны реализовать все методы, объявленные в интерфейсе, если вы хотите использовать этот интерфейс в своем классе. Этот способ создания обработчиков событий имеет недостаток: если вы хотите использовать только один метод некоторого интерфейса, вы должны реализовать все методы этого интерфейса. Например, вы хотите создать фрейм с меткой, если пользователь нажимает рамку в некотором положении, метка укажет положение мыши (x и y) внутри этого фрейма. Пример (1) ниже иллюстрирует эту программу:

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

class MyFrame extends JFrame implements MouseListener
{
 JLabel label1=new JLabel("");//1

 public MyFrame() //2
 { setLayout(new FlowLayout());//3
  this.add(label1);//3
  this.addMouseListener(this);//4
 }

public void mouseClicked(MouseEvent e) {
  int x=e.getX();//5
  int y=e.getX();
  label1.setText("Frame clicked! "+"x="+x+" y="+y);
}
```

```
public void mouseEntered(MouseEvent e) {}//6
public void mouseExited(MouseEvent e)  {}//7
public void mousePressed(MouseEvent e) {}//8
public void mouseReleased(MouseEvent e){}//9
}

public class Example
{ //The class-driver
public static void main( String[] args )
 {
    MyFrame f = new MyFrame(); // create object of
the class
    f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE
);
    f.setSize(250,150);//Set size of the frame
    f.setVisible( true ); // display frame

 } // end main
}
```

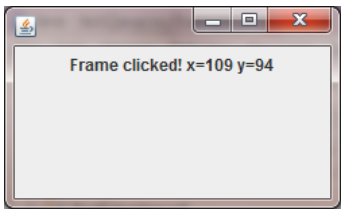


Рисунок 1. GUI примера (1), описанного выше

Основной кадр программы, определенный в классе «Myframe», используя наследование от суперкласса «JFrame» и наследование от интерфейса MouseListener. Интерфейс MouseListener, ответственный за события, связанные с кнопками мыши и положением мыши в сторону компонентов, этот интерфейс содержит 5 методов:

MouseClicked - это событие появляется в любое время, когда пользователь нажимает на какой-либо компонент с помощью кнопки мыши

MouseEntered - щелкнуть компонент во-первых, вы должны перемещать мышь внутри компонента (в нашем случае компонент представляет собой фрейм «MyFrame»). Событие mouseEntered, сгенерированное в тот момент, когда мышь появляется внутри компонента

MouseExited - обратный случай mouseEntered. Событие mouseExited генерируется автоматически каждый раз, когда мышь покидает компонент (появляется вне компонента).

MousePressed - событие, которое будет генерироваться в любое время, когда вы нажимаете кнопку мыши (и удерживайте ее)

MouseReleased - напротив mousePressed event. Это событие генерируется, когда пользователь отпускает кнопку мыши. Например, одно событие mouseClicked, предсказанное событиями mousePressed и mouseReleased.

Для обработки события «mouseClicked» мы использовали интерфейс MouseListener, который содержит 5 событий. Даже если вам нужно только одно событие (метод «mouseClicked»), вы должны реализовать все методы используемого интерфейса, см. Строки 6-9 примера.

Многие интерфейсы event-listener, такие как MouseListener и MouseMotionListener, содержат несколько методов. Не всегда желательно объявлять каждый метод в интерфейсе event-listener. Например, для приложения может понадобиться только обработчик mouseClicked из MouseListener или обработчик mouseDragged из MouseMotionListener. Интерфейс WindowListener задает семь методов обработки событий окна. Для многих интерфейсов слушателя, которые имеют несколько методов, пакеты java.awt.event и javax.swing.event предоставляют классы адаптера событий-слушателей. Класс адаптера реализует интерфейс и предоставляет стандартную реализацию (с пустым телом метода) каждого метода в интерфейсе.

В таблице 1 показаны несколько классов адаптера java.awt.event и интерфейсы, которые они реализуют.

Таблица 1. Интерфейсы и класс-адаптеры

N	Название интерфейса	Название адаптера	Описание событий
1	ComponentListener	ComponentAdapter	События компонента. Способ регистрации: addcomponentListener () или добавить компонентный адаптер. Интерфейс и адаптер класса содержат набор методов:componentHidden();componentMoved(); componentResized();componentShown();
2	ContainerListener	ContainerAdapter	События, связанные с добавлением и удалением компонентов в / из класса контейнера. Способ регистрации:addContainerListener() или add ContainerAdapter. Contains 2 methods: componentAdded() и componentRemoved()
3	FocusListener	FocusAdapter	События, связанные с фокусом (когда компонент контролирует интерфейс пользователя). Способ регистрации:addFocusListener() или addFocusAdapter()
4	KeyListener	KeyAdapter	Ответственный за нажатие и отпускание клавиш. Способ регистрации:addKeyListener или addKeyAdapter. Contains 3 methods: keyPressed(); keyTyped(); keyReleased().
5	MouseListener	MouseAdapter	Некоторые события мыши, интерфейс (и класс адаптера) содержат 5 методов:mouseClicked();mouseEntered();mouseExited() mousePressed();mouseReleased()
6	MouseMotionListener	MouseMotionAdapter	События, связанные с движением мыши. Этот класс и интерфейс содержат 2 метода:mouseDragged() и mouseMoved()
7	WindowListener	WindowAdapter	События из окна. Метод регистрации addWindowListener или добавьте WindowAdapter. Содержит 7 событий:windowActivated(); windowClosed(); windowClosing(); windowDeactivated(); windowDeiconofied(); windowIconofied(); windowOpened().

Вы можете расширить класс адаптера, чтобы наследовать реализацию по умолчанию каждого метода и впоследствии переопределять только методы, необходимые для обработки событий.

Пример ниже (2) - пример (1), где обработка событий реализована с использованием класса адаптера «MouseAdapter» вместо интерфейса «MouseListener»:

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

class MyFrame extends JFrame
{
 class MyClick extends MouseAdapter//inner class
  {
    public void mouseClicked(MouseEvent e) {
    int x=e.getX();//5
    int y=e.getY();
    label1.setText("Frame     clicked!     "+"x="+x+"
y="+y);
   }
}
 JLabel label1=new JLabel("");//1

 public MyFrame() //2
 { setLayout(new FlowLayout());//3
  this.add(label1);//4
  this.addMouseListener(new MyClick());//5
 }
}

public class Example
{ //The class-driver
public static void main( String[] args )
 {
```

```
    MyFrame f = new MyFrame(); // create object of
the class
    f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE
);
    f.setSize(250,150);//Set size of the frame
    f.setVisible( true ); // display frame

 } // end main
}
```

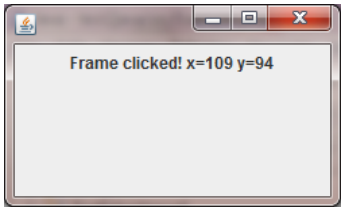


Рисунок 2. GUI примера (2), описанного выше

Теперь вы можете видеть, что код примера 2 простой и короткий по сравнению с примером 1, что является причиной использования классов адаптеров вместо интерфейсов.

Компоненты Swing GUI содержат два типа кнопок состояния: классы JCheckBox и JRadioButton. JRadioButton отличается от JCheckBox тем, что обычно несколько JRadioButtons сгруппированы вместе и являются взаимоисключающими - только в одной группе можно выбрать в любое время, точно так же, как кнопки на автомобильном радио. Элементы данных и методы классов JCheckBox и JRadioButton схожи, поэтому мы обсудим здесь класс JRadioButton.

Радио кнопки (объявленные с классом JRadioButton) аналогичны флажкам, поскольку они имеют два состояния - выбраны и не выбраны (также называемые отмененными). Однако радиокнопки обычно отображаются как группа, в которой может быть выбрана только одна кнопка за раз (см. Вывод на рисунке 8.2). Выбор другого переключателя заставляет все остальные отменять выбор. Радио-кнопки используются для представления взаимоисключающих опций (т. Е. Одновременно невозможно выбрать несколько параметров в группе). Логическая связь между переключателями поддерживается объектом ButtonGroup (пакет javax.swing), который сам по себе не является компонентом GUI. Объект

ButtonGroup организует группу кнопок и сам не отображается в пользовательском интерфейсе. Скорее, отдельные объекты JRadioButton из группы отображаются в графическом интерфейсе. В приложении ниже (пример 3) используются переключатели, допускающие только один ответ на вопрос:

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

class MyFrame extends JFrame //1
{
  class MyClick extends MouseAdapter
  {
   public void mouseClicked(MouseEvent e) {
   if (rb1.isSelected())
    JOptionPane.showMessageDialog(null,  "You  said  'Yes'");
//2
   if (rb2.isSelected())
    JOptionPane.showMessageDialog(null,  "You  said  'No!'");
//3
   if (rb3.isSelected())
    JOptionPane.showMessageDialog(null,       "You       said
"+rb3.getText());//4
   }
}
JLabel  label1=new  JLabel("Balqa  is  very  good  university  in
Amman?");//5
JRadioButton rb1=new JRadioButton("Yes");//6
JRadioButton rb2=new JRadioButton("No");//7
JRadioButton rb3=new JRadioButton("I don't know");//8
ButtonGroup bg=new ButtonGroup();//9
JButton but1=new JButton("My answer");//10

 public MyFrame() //constructor of MyFrame
 { setLayout(new FlowLayout());//11
  this.add(label1);//12
  this.add(rb1);this.add(rb2);//13
  this.add(rb3);//14
  this.add(but1);//15
  bg.add(rb1);bg.add(rb2);bg.add(rb3);//16
```

```
  but1.addMouseListener(new MyClick());////17
 }
}

public class Example
{ //The class-driver
public static void main( String[] args )
 {
     MyFrame f = new MyFrame(); // create object of the class
     f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
     f.setSize(250,150);//Set size of the frame
     f.setVisible( true ); // display frame

 } // end main
}
```

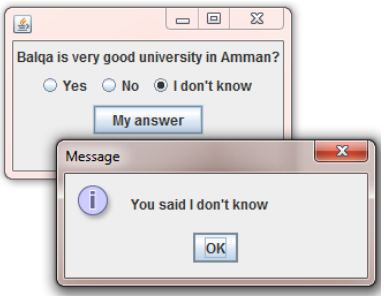
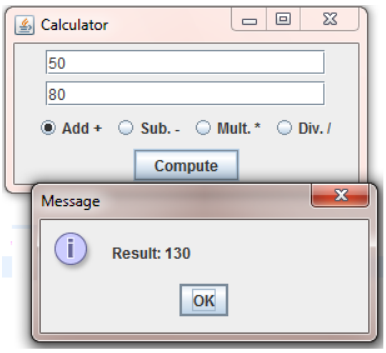


Рисунок 3. GUI примера (3)

3. Задание для работы:

Создайте простой калькулятор Java, графический интерфейс программы, представленный ниже:



GUI содержит 2 объекта JTextField, которые представляют собой переключатели номер 1 и номер 2, 4, каждый переключатель обозначает некоторую арифметическую операцию и кнопку для вычисления результата. По выбору пользователя программа показывает сумму числа

1 и числа 2 или продукт или любой другой результат операции. Используйте адаптер класса MouseAdapter для обработки evemt «щелчок» кнопки.

4 Содержание отчета

1. Титульный лист
2. Задание на работу
3. Листинг программы
4. Скриншот работы приложения

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Программирование REST-сервисов на языке Java [Электронный ресурс] : методиче-ские указания по выполнению лабораторной работы по курсу «Интернет-технологии» для сту-дентов, обучающихся по направлению подготовки 230400.62 «Информационные системы и тех-нологии» / Юго-Западный государственный университет, Кафедра информационных систем и технологий ; ЮЗГУ ; сост. М. В. Бородин. - Курск : ЮЗГУ, 2013. - 18 с.

2. Программирование Web-сервисов на языке Java [Электронный ресурс] : методические указания по выполнению лабораторной работы по курсу «Интернет-технологии» для студентов, обучающихся по направлению подготовки 230400.62 «Информационные системы и технологии» / ЮЗГУ ; сост.: Е. А. Титенко, М. В. Бородин. - Курск : ЮЗГУ, 2013. - 22 с.