Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Кафедра Автоматизированных систем

ОТЧЕТ

Вариант 16

к лабораторной работе по дисциплине:

|  |
| --- |
| Объектно–ориентированное программирование |
| Разработка приложения с оконным графическим |
| Пользовательским интерфейсом. |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | АСУб-19-1 |  |  |  | Гамаюнова А.С. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В, |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск 2020 г.

Содержание

1 Постановка задачи3

2 Проектирование классов4

3 Проектирование графического интерфейса пользователя 7

4 Таблица тестов8

5 Результаты тестирования9

6 Исходный код14

Список использованных источников24

1 Постановка задачи

1. Получение общего представления о шаблоне программирования Модель/Вид/Контроллер и его практического применения.
2. Получение практического опыта работы с базовыми графическими библиотеками java.

Для выполнения задания необходимо разработать графический пользовательский интерфейс средствами java для работы с классами реализованными при выполнении либо 3-ей, либо 4-ой лабораторной работы.

Графический пользовательский интерфейс должен предоставлять возможности работы со всеми возможными функциями программы (ввод и вывод данных в соответствующей форме, изменение условий выполнения задания, работа с файлами и т.д.). Состав используемых компонентов определяется самостоятельно, но для них должны быть предусмотрены соответствующие планировщики раскладки компонентов в основных контейнерах.

Классы, реализующие графический пользовательский интерфейс, должны содержать только обращения к методам классов для выполнения заданий. Для этого может понадобиться модифицировать программный код из предыдущих лабораторных работ.

2 Проектирование классов

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Проектирование классов | | | |
| **№** | **Название** | **Назначение** | **Поля (атрибуты)** |
| 1 | public class Array | Класс для работы с массивом | private int amount – количество элементов в массиве  private int[] arr - массив |
| 2 | public class WorkString | Класс для работы со строкой | private String text – строка |
| 3 | public class WorkFile | Класс для работы с вводом/выводом файлов | static String traceback – вывод сообщения об ошибке; |
| 4 | public class main | Класс для запуска | - |
| 5 | public class MainWindow | Класс с главным окном | static JRadioButton task1RadioButton, static JRadioButton task2RadioButton, static JRadioButton task3RadioButton – кнопки для выбора задания  private JTextArea label1, private JTextArea label2, private JTextArea label3 – текстовые формы с заданиями  private JLabel label4, private JLabel label6, private JLabel label7, private JLabel label9, private JTextArea – текстовые формы для выбора  static JButton buttonTask, static JButton buttonOpen, static JButton buttonSave, static JButton buttonUM – кнопки с действиями  static JTextField text,static JTextField text2,  static JTextField text1, static SpinnerModel value1, static JTextField text3 – поля для пользовательского ввода |
| 6 | public class EventListener | Класс для обработки событий | private MainWindow m – получение полей класса MainWindiw  private Do d – получение полей класса Do |
| 7 | public class Do | Класс с диалоговым окном | private JLabel label1, static JTextArea label2 – формочки с текстом |

**Array:**

Методы:

public Array() – конструктор по умолчанию

public Array(int amount) – конструктор с заданным параметром

private int[] createArr() – создание массива

public String getArr() – получение массива

public Array(Array a) – конструктор копирования

public String getForSave() – получение массива для сохранения

public void setArr(int chislo)- установка массива с одним определенным элементом

**WorkString:**

Методы:

public WorkString() – конструктор по умолчанию

public WorkString(String text) – конструктор с заданным параметром

public WorkString(WorkString s) – конструктор копирования

private String createNew\_str() – удаление слов, начинающихся и заканчивающихся на разные буквы

public String getNew\_str() – вывод строки

public String getForSaveNew() – вывод строки для сохранения

private int countTotal() – подсчет знаков препинания

public String getTotal – вывод количества знаков препинания

public int getForSaveTotal() – вывод количества знаков препинания для сохранения

public void setStr(String znak) – изменение копии

**Do:**

Методы:

public Do() – прорисовка диалогового окна

**WorkFile:**

Методы:

public static String getTrace() – получение сообщения об ошибки

public static void setTrace(String traceback) – изменение текста об ошибке

public static int loadMFile (String name) throws IOException – открытие файла с числом

public static String loadSFile (String name) throws IOException – открытие файла со строкой

public static void writeFile(String name, String filename) – запись строки в файл

public static void writeByte(String name, String filename) – запись числа в файл

**Main:**

Методы:

public static void main(String[] args) throws Exception – вызов главного окна

**MainWindow:**

Методы:

public MainWindow()- прорисовка главного окна

**EventListener:**

Методы:

public void actionPerformed(ActionEvent e)

3 Проектирование графического интерфейса пользователя

Для удобства пользователя используется табличное проектирование, имеется возможность выбора задания, ввода/вывода информации, работы с копией, ввод/вывод из файла



Рисунок 1 – Модель главного окна



Рисунок 2 – Модель диалогового окна

4 Таблица тестов

Таблица 2 – Таблица тестов.

|  |  |  |
| --- | --- | --- |
| **№** | **Словесное описание теста** | **Что должна сделать программа** |
| 1 | Нажать на кнопку Выполнить | Вызов окна с выполнением |
| 2 | Нажать на кнопку Сохранить | Сохранение файла |
| 3 | Нажать на кнопку Открыть | Переход к диалоговому окну сохранения |
| 4 | Нажать на кнопку Выполнить по умолчанию | Переход к диалоговому окну открытия |
| 5 | Нажать на кнопку С копией | Удаление организации |
| 6 | Нажать на Крестик | Завершение программы |
| 7 | Ошибка с копией | Вывод ошибки |
| 8 | Ошибка ввода имени файла | Вывод ошибки |

5 Результаты тестирования

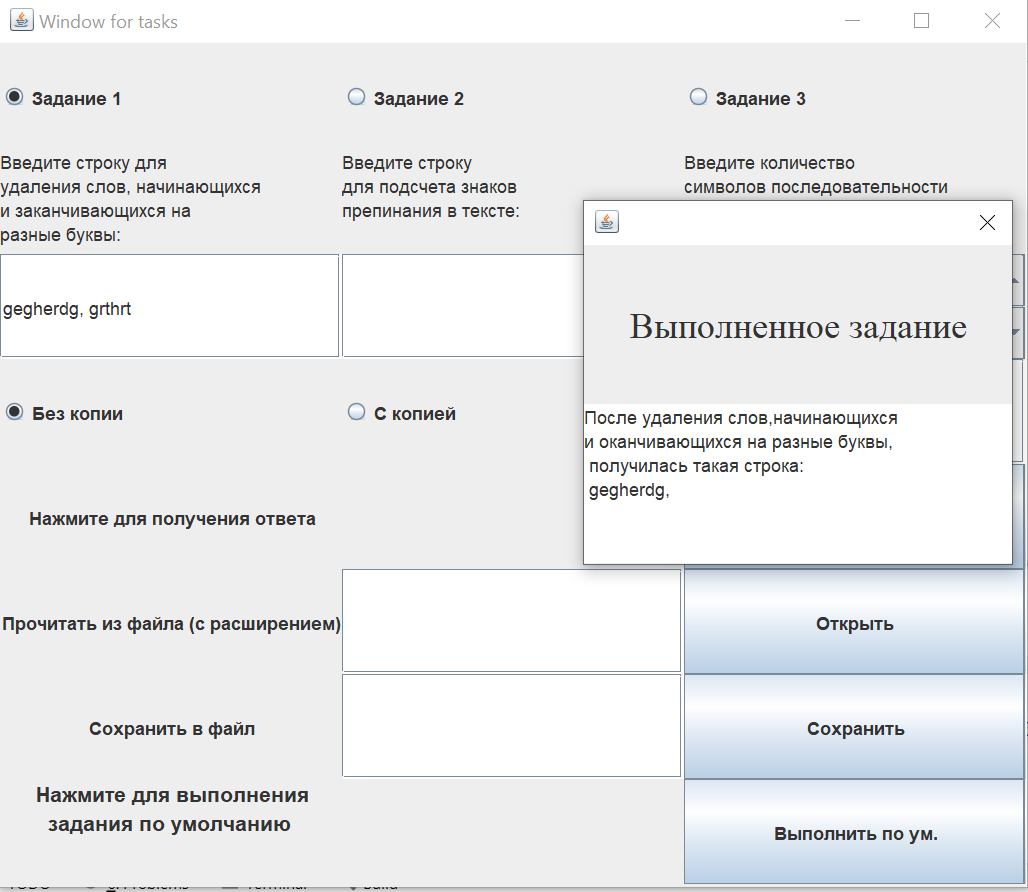


Рисунок 3 – Тест №1

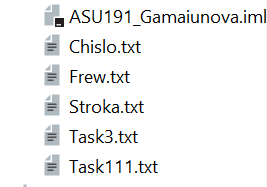
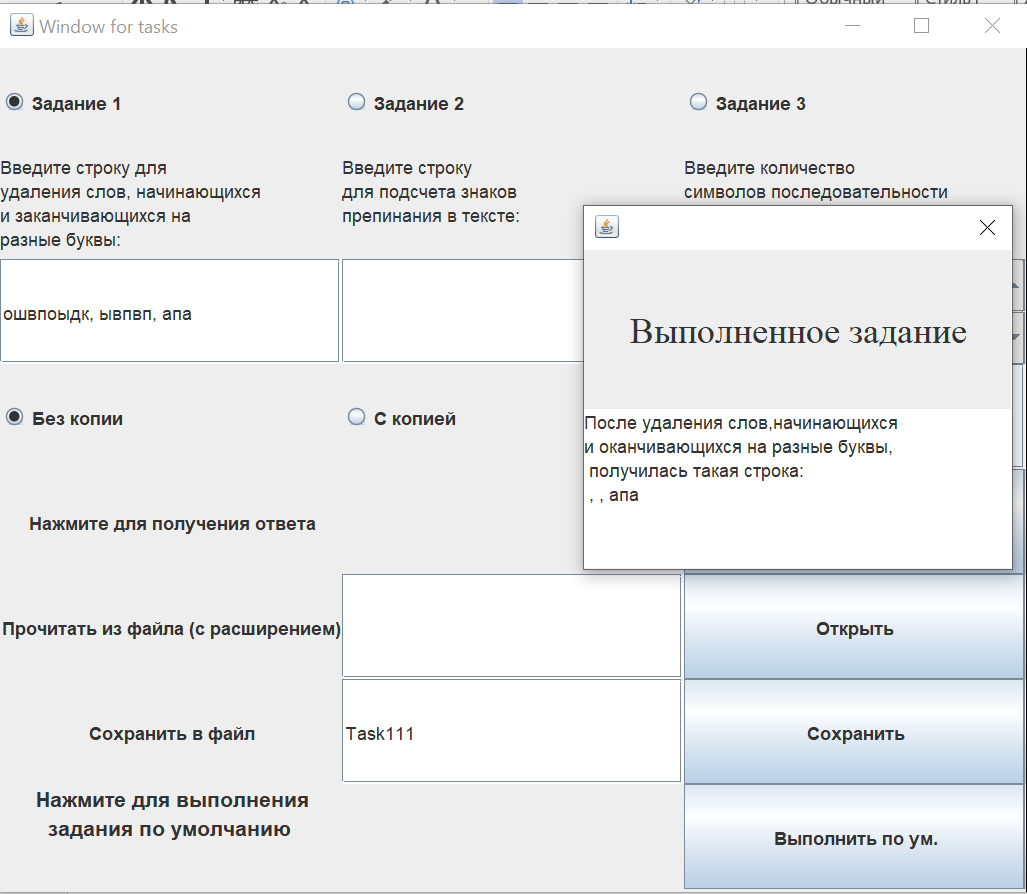


Рисунок 4.1 – Тест №2 (подтверждение создания файла)

Рисунок 4 – Тест №2

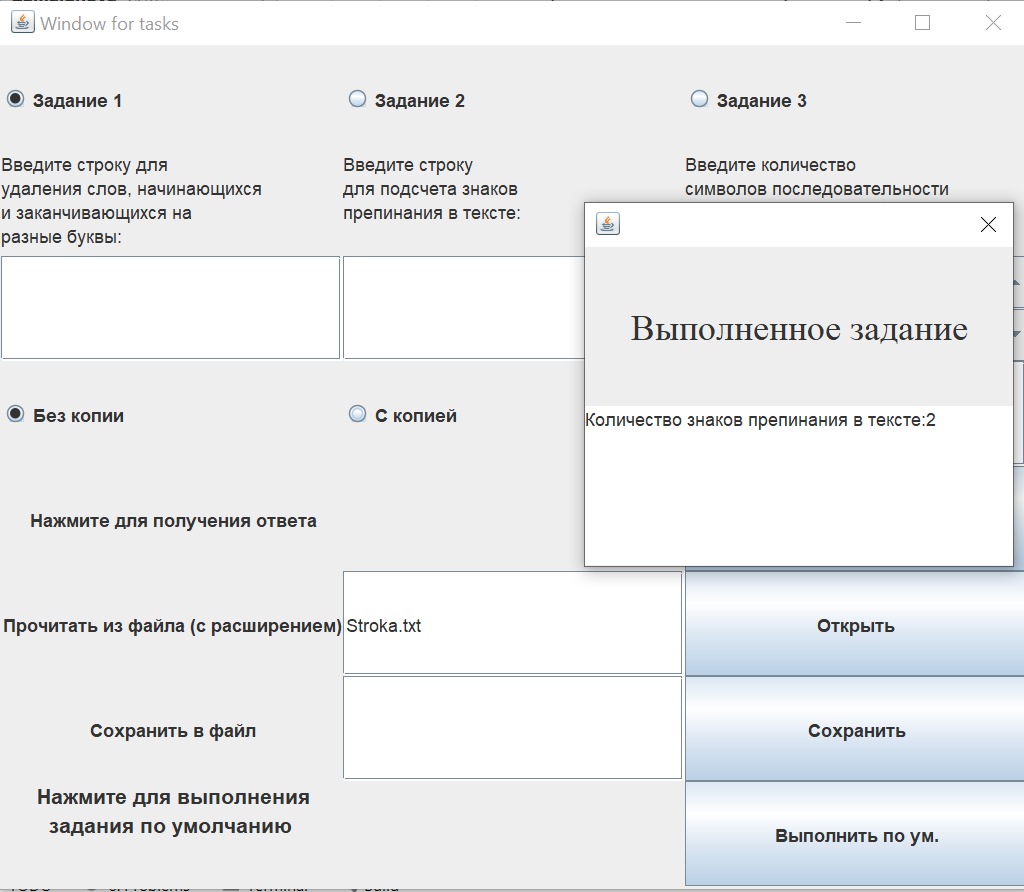
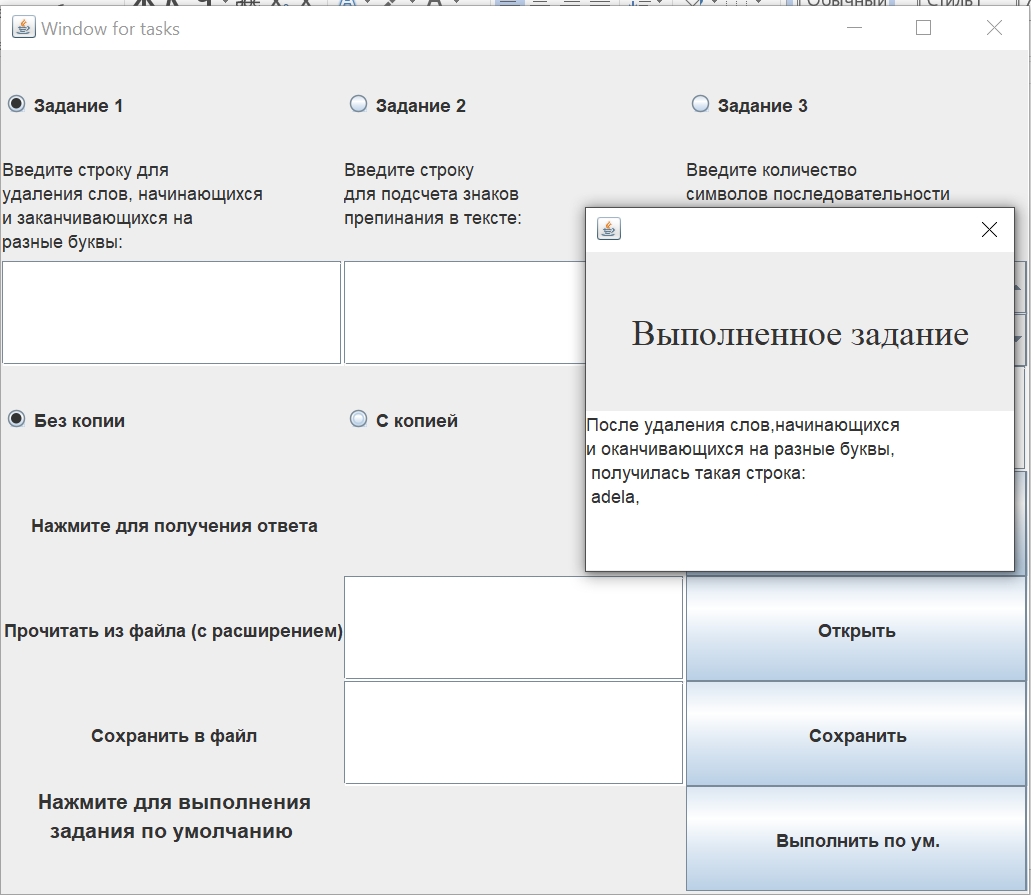


Рисунок 6 – Тест №4

Рисунок 5 – Тест №3

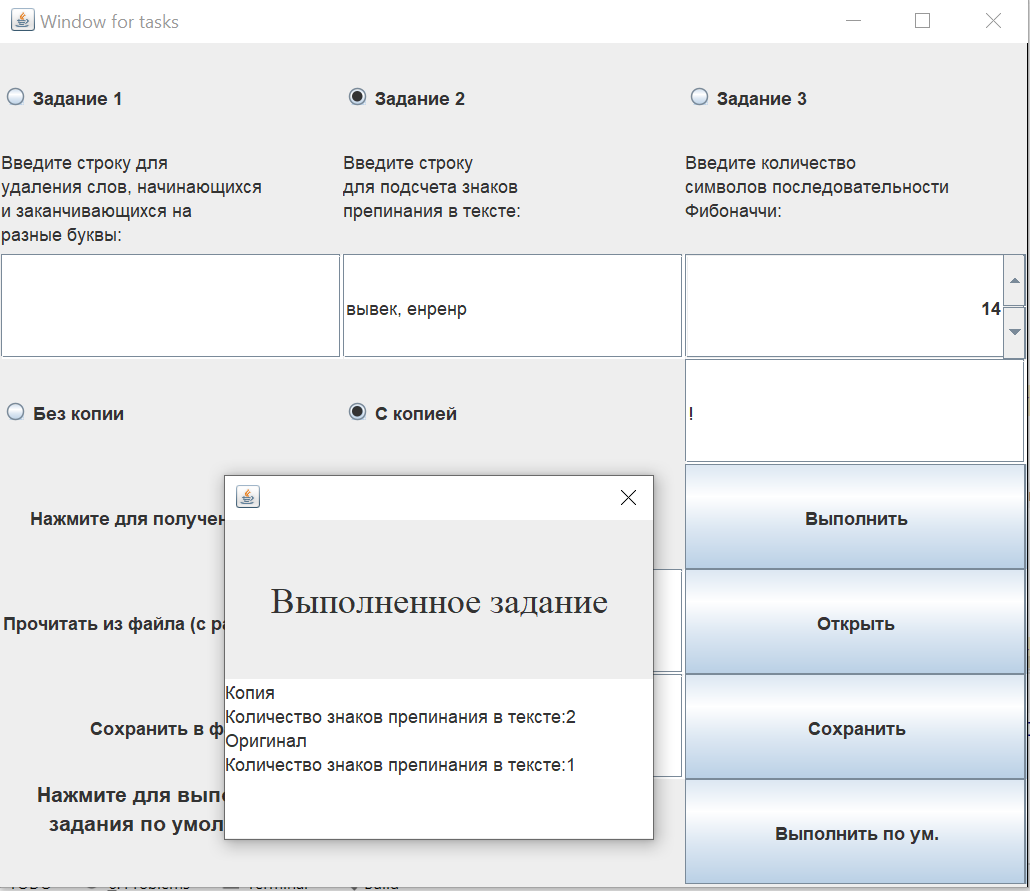
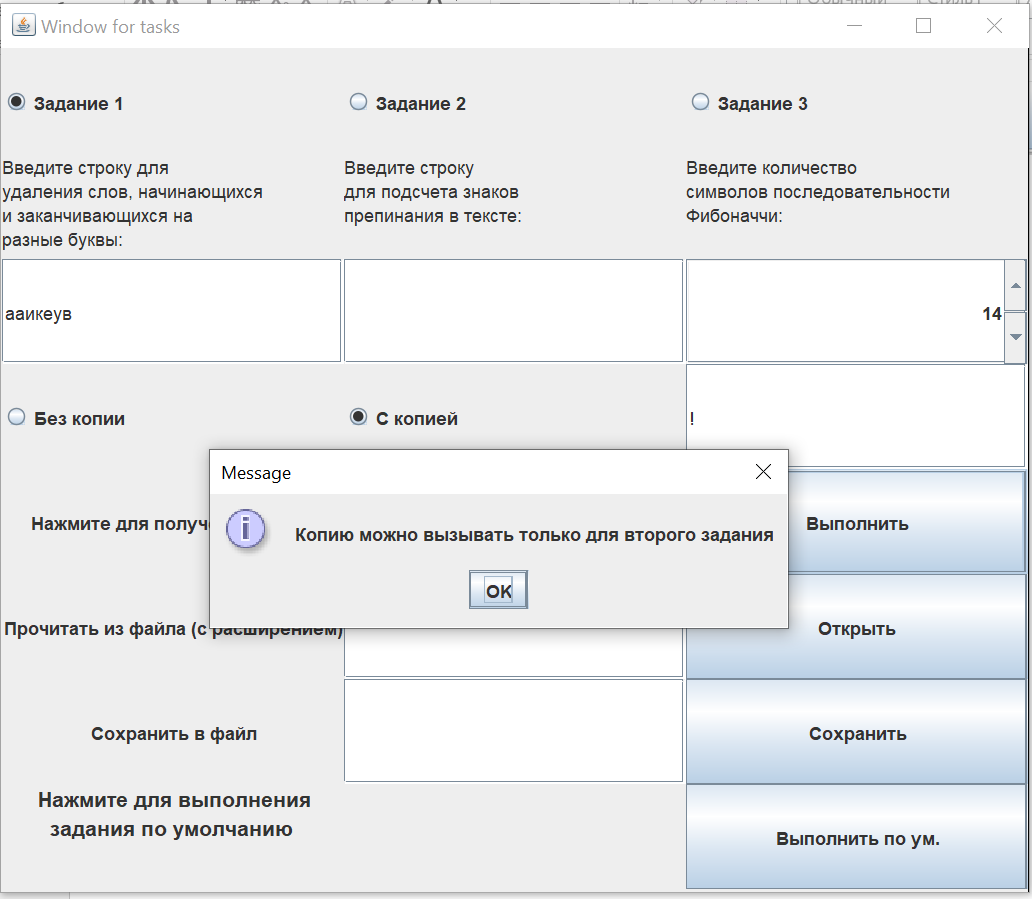


Рисунок 8 – Тест №7

Рисунок 7 – Тест №5

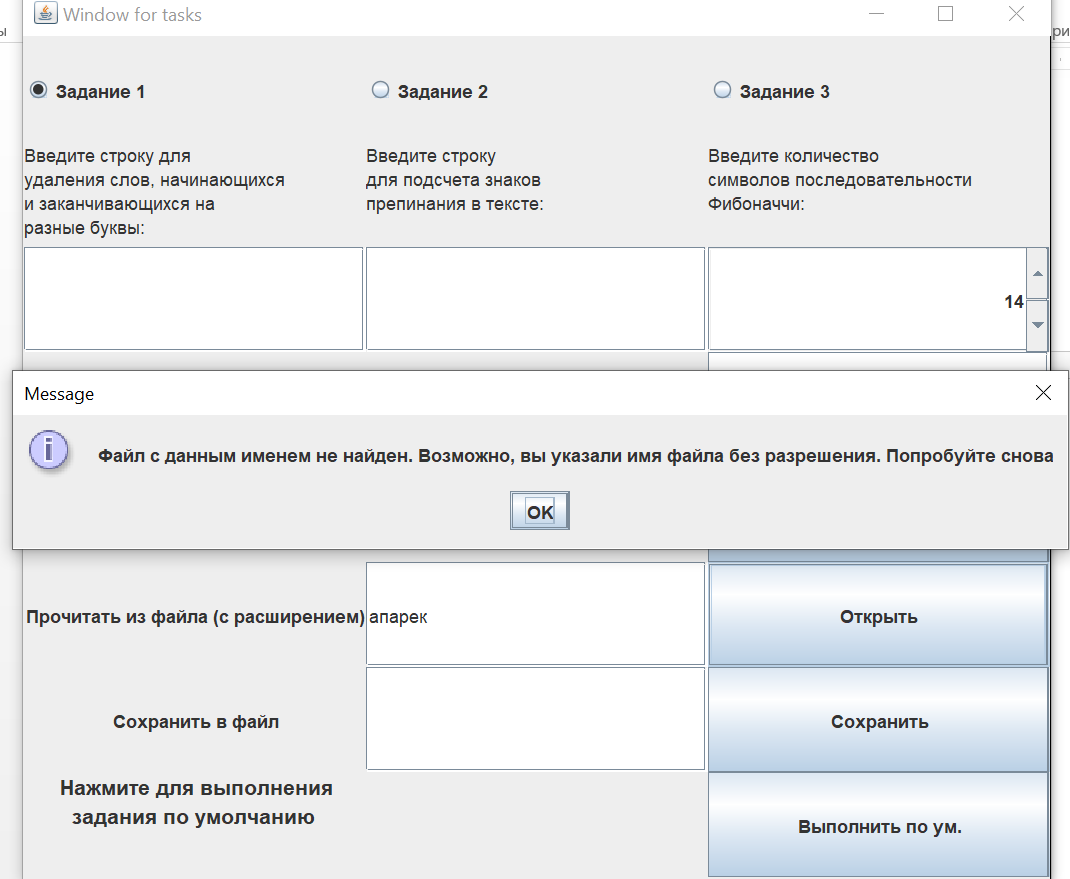


Рисунок 9 – Тест №8

6 Исходный код

MainWindow

package com.company;  
  
import javax.swing.\*;  
import javax.swing.text.JTextComponent;  
import java.awt.\*;  
  
public class MainWindow extends JFrame {  
 static JRadioButton *task1RadioButton*=new JRadioButton("Задание 1");  
 static JRadioButton *task2RadioButton* = new JRadioButton("Задание 2");  
 static JRadioButton *task3RadioButton*=new JRadioButton("Задание 3");  
  
 private JTextArea label1 = new JTextArea("Введите строку для \nудаления слов, начинающихся \nи заканчивающихся на \nразные буквы:");  
 private JTextArea label2 = new JTextArea("Введите строку\nдля подсчета знаков \nпрепинания в тексте:");  
 private JTextArea label3 = new JTextArea("Введите количество \nсимволов последовательности \nФибоначчи:");  
 private JLabel label4 = new JLabel("Прочитать из файла (с расширением)");  
 private JLabel label6 = new JLabel("Нажмите для получения ответа");  
 private JLabel label7 = new JLabel("Сохранить в файл\n");  
 private JLabel label9 = new JLabel("");  
  
 static JRadioButton *copRadioButton*=new JRadioButton("С копией");  
 static JRadioButton *nocopRadioButton* = new JRadioButton("Без копии");  
 static JTextField *copt* = new JTextField ();  
  
 static JButton *buttonTask*=new JButton("Выполнить");  
 static JButton *buttonOpen* = new JButton("Открыть");  
 static JButton *buttonSave* = new JButton("Сохранить");  
  
 private JTextArea label8 = new JTextArea(" Нажмите для выполнения\n задания по умолчанию");  
 private JLabel label10 = new JLabel("");  
 static JButton *buttonUM*=new JButton("Выполнить по ум.");  
 static JTextField *text* = new JTextField ();  
 static JTextField *text2* = new JTextField ();  
 static JTextField *text1* = new JTextField ();  
 static SpinnerModel *value1* = new SpinnerNumberModel(10, 10, 49, 1);  
 static JSpinner *spin2* = new JSpinner(*value1*);  
 static JTextField *text3* = new JTextField ();  
  
  
 public MainWindow(){  
 super("Window for tasks");  
  
  
 setBounds(100,100,700,600);  
 setVisible(true);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 *//создаём контейнер* Container container = this.getContentPane();  
 container.setLayout(new GridLayout(8,3));  
 *//Container container1 = this.getContentPane();  
 //container1.setLayout(new BoxLayout(getContentPane(), BoxLayout.Y\_AXIS));  
  
 //группируем задания* ButtonGroup group = new ButtonGroup();  
 group.add(*task1RadioButton*);  
 group.add(*task2RadioButton*);  
 group.add(*task3RadioButton*);  
  
 *//группируем копийные элементы* ButtonGroup group2 = new ButtonGroup();  
 group2.add(*copRadioButton*);  
 group2.add(*nocopRadioButton*);  
  
 *//изначально выбрана первая задача и выполнение без копии  
 task1RadioButton*.setSelected(true);  
 *nocopRadioButton*.setSelected(true);  
  
 *//делаем недоступными JTextArea  
 //label5.setEditable(false);* label1.setEditable(false);  
 label2.setEditable(false);  
 label3.setEditable(false);  
 label8.setEditable(false);  
  
 *//устанавливаем им текст* label1.setBackground(null);  
 label2.setBackground(null);  
 label3.setBackground(null);  
 label8.setBackground(null);  
  
 Font font = new Font("Baskerville", Font.*BOLD*, 14);  
 label8.setFont(font);  
  
 *//и выравнивание* label4.setHorizontalAlignment(SwingConstants.*CENTER*);  
 label6.setHorizontalAlignment(SwingConstants.*CENTER*);  
 label7.setHorizontalAlignment(SwingConstants.*CENTER*);  
  
 *// добавляем в контейнер поочерёдно блоки* container.add(*task1RadioButton*);  
 container.add(*task2RadioButton*);  
 container.add(*task3RadioButton*);  
  
 container.add(label1);  
 container.add(label2);  
 container.add(label3);  
  
 container.add(*text*);  
 container.add(*text2*);  
 container.add(*spin2*);  
  
 container.add(*nocopRadioButton*);  
 container.add(*copRadioButton*);  
 container.add(*copt*);  
  
 container.add(label6);  
 container.add(label9);  
 *//container.add(label5);* container.add(*buttonTask*);  
  
 container.add(label4);  
 container.add(*text3*);  
 container.add(*buttonOpen*);  
  
 container.add(label7);  
 container.add(*text1*);  
 container.add(*buttonSave*);  
  
 container.add(label8);  
 container.add(label10);  
 container.add(*buttonUM*);  
 *//создаём слушателей для обработки нажатия кнопок  
 buttonTask*.addActionListener(new EventListener());  
 *buttonSave*.addActionListener(new EventListener());  
 *buttonOpen*.addActionListener(new EventListener());  
 *buttonUM*.addActionListener(new EventListener());  
 }  
  
}

main

package com.company;  
  
public class main {  
 public static void main(String[] args) throws Exception {  
 new MainWindow();  
 }  
}

WorkFile

package com.company;  
  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.io.\*;  
import java.util.Scanner;  
  
public class WorkFile {  
 static String *traceback* = "";  
 public static String getTrace() {  
 return *traceback*;  
 }  
  
 public static void setTrace(String traceback) {  
 WorkFile.*traceback* = traceback;  
 }  
 public WorkFile() {  
 }  
  
 public static int loadMFile (String name) throws IOException {  
 try{  
 File file = new File(name);  
 Scanner scanner = new Scanner(file);  
 int out = Integer.*parseInt*(scanner.nextLine());  
 scanner.close();  
 return out;  
 }  
 catch(FileNotFoundException e){  
 *setTrace*("Файл с данным именем не найден. Возможно, вы указали имя файла без разрешения. Попробуйте снова");  
 }  
 return 0;  
 }  
  
 public static String loadSFile (String name) throws IOException {  
 try {  
 File file = new File(name);  
 Scanner scanner = new Scanner(file);  
 String output = scanner.nextLine();  
 scanner.close();  
 return output;  
 }  
 catch(FileNotFoundException e){  
 *setTrace*("Файл с данным именем не найден. Возможно, вы указали имя файла без разрешения. Попробуйте снова");  
 }  
 return "";  
 }  
  
 public static void writeFile(String name, String filename) {  
 try {  
 String fn = filename + ".txt";  
 File file = new File(fn);  
 PrintWriter pw = new PrintWriter(file);  
 pw.println(name);  
 pw.close();  
 }catch (FileNotFoundException e) {  
 *setTrace*("Возникла ошибка. Попробуйте снова");  
 }  
 }  
 public static void writeByte(String name, String filename) {  
 String fn = filename + ".txt";  
 OutputStream os = null;  
 name = name.substring(name.indexOf("\n")+1,name.length()-1);  
 try {  
 os = new FileOutputStream(new File(fn));  
 os.write(name.getBytes(), 0, name.length());  
 } catch (IOException e) {  
 *setTrace*("Возникла ошибка. Попробуйте снова");  
 } finally {  
 try {  
 os.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
  
}

Do

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Do extends JDialog {  
 private JLabel label1 = new JLabel("Выполненное задание");  
 static JTextArea *label2* = new JTextArea();  
  
 public Do(){  
 *//создаём контейнер, описывающий структуру окна* Container container = this.getContentPane();  
 container.setLayout(new GridLayout(2,1));  
 label1.setHorizontalAlignment((int) *CENTER\_ALIGNMENT*);  
 label1.setFont(new Font("Serif", Font.*PLAIN*, 24));  
 *label2*.setEditable(false);  
  
 container.add(label1);  
 container.add(*label2*);  
 *//Делаем невидимым окно* setVisible(false);  
 *//Устанавливаем размеры* setSize(300, 250);  
 *//Отображаем по центру* setLocationRelativeTo(null);  
 }  
}

EventListener

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.IOException;  
  
public class EventListener implements ActionListener {  
 *//поля для получения данных полей* private MainWindow m;  
 private Do d;  
 public void actionPerformed(ActionEvent e) {  
 d = new Do();  
 d.setVisible(true);  
 if (e.getActionCommand().equals("Выполнить")) {  
  
 if (m.*task2RadioButton*.isSelected()) {  
 WorkString stroka = new WorkString(m.*text2*.getText());  
 if (m.*copRadioButton*.isSelected()){  
 WorkString stroka1 = new WorkString(stroka);  
 String znak = m.*copt*.getText();  
 stroka1.setStr(znak);  
 *//Работа с конструктором копирования* d.*label2*.setText("Копия\n" + stroka1.getTotal()+"\nОригинал\n" + stroka.getTotal());  
 }  
 else{  
 d.*label2*.setText(stroka.getTotal());  
 }  
 }  
 else if (m.*task3RadioButton*.isSelected()) {  
 Array array = new Array((int) m.*spin2*.getValue());  
 if (m.*copRadioButton*.isSelected()) {  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null, "Копию можно вызывать только для второго задания");  
 }  
 else{  
 d.*label2*.setText("Числа Фибоначчи\n" + array.getForSave());  
 }  
 } else {  
 if (m.*copRadioButton*.isSelected()) {  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null, "Копию можно вызывать только для второго задания");  
 }  
 else {  
 WorkString stroka = new WorkString(m.*text*.getText());  
 d.*label2*.setText(stroka.getNew\_str());  
 }  
 }  
 } else  
 *//если была нажата кнопка "Save"* if (e.getActionCommand().equals("Сохранить")) {  
 WorkFile sv = new WorkFile();  
 if (m.*task2RadioButton*.isSelected()) {  
 d.setVisible(false);  
 sv.*writeFile*(d.*label2*.getText(), m.*text1*.getText());  
 if (!sv.*getTrace*().equals("")) {  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null, sv.*getTrace*());  
 sv.*setTrace*("");  
 }  
 } else if (m.*task3RadioButton*.isSelected()) {  
 try {  
 sv.*writeByte*(d.*label2*.getText(), m.*text1*.getText());  
 if (!sv.*getTrace*().equals("")){  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null,sv.*getTrace*());  
 sv.*setTrace*("");  
 }  
 } catch (Exception exception) {  
 exception.printStackTrace();  
 }  
 } else {  
 sv.*writeFile*(d.*label2*.getText(), m.*text1*.getText());  
 if (!sv.*getTrace*().equals("")){  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null,sv.*getTrace*());  
 sv.*setTrace*("");  
 }  
  
 }  
 }  
 *//если была нажата кнопка "Open"* if (e.getActionCommand().equals("Открыть")) {  
 WorkFile so = new WorkFile();  
  
 if (m.*task1RadioButton*.isSelected() == true) {  
 try {  
 String path = m.*text3*.getText();  
 String text;  
 text = WorkFile.*loadSFile*(path);  
 WorkString stroka = new WorkString(text);  
 d.*label2*.setText(stroka.getTotal());  
 if (!so.*getTrace*().equals("")){  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null,so.*getTrace*());  
 so.*setTrace*("");  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 } else if (m.*task2RadioButton*.isSelected() == true) {  
 try {  
 String path = m.*text3*.getText();  
 String text = null;  
 text = WorkFile.*loadSFile*(path);  
 WorkString stroka = new WorkString(text);  
 d.*label2*.setText(stroka.getNew\_str());  
 if (!so.*getTrace*().equals("")){  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null,so.*getTrace*());  
 so.*setTrace*("");  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 } else {  
 Array massiv;  
 try {  
 String path = m.*text3*.getText();  
 int amount = WorkFile.*loadMFile*(path);  
 massiv = new Array(amount);  
 d.*label2*.setText("Числа Фибоначчи\n" + massiv.getForSave());  
 if (!so.*getTrace*().equals("")){  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null,so.*getTrace*());  
 so.*setTrace*("");  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }  
 }  
  
 if (e.getActionCommand().equals("Выполнить по ум.")) {  
  
 if (m.*task2RadioButton*.isSelected()) {  
 WorkString stroka = new WorkString();  
 d.*label2*.setText(stroka.getTotal());  
 }  
 else if (m.*task3RadioButton*.isSelected()) {  
 Array array = new Array();  
 d.*label2*.setText("Числа Фибоначчи\n" + array.getForSave());  
 } else {  
 WorkString stroka = new WorkString();  
 d.*label2*.setText(stroka.getNew\_str());  
 }  
 }  
 }  
 }

Array

package com.company;  
  
import java.util.Random;  
  
public class Array {  
 private int amount;  
 private int[] arr;  
  
 public Array(){  
 this.amount = new Random().nextInt(41) + 10;  
 this.arr = createArr();  
 }  
  
 public Array(int amount){  
 this.amount = amount;  
 this.arr = createArr();  
 }  
  
 private int[] createArr(){  
 this.arr = new int[this.amount];  
  
 this.arr[0] = this.arr[1] = 1;  
  
 this.amount = this.amount - 2; *//так как 2 элемента задано* int i = 0;  
  
 while (i < this.amount) {  
 this.arr[i + 2] = this.arr[i + 1] + this.arr[i];  
  
 i++;  
 }  
 return this.arr;  
 }  
  
 public String getArr(){  
 int i = 0;  
  
 while (i < 10) {  
 int element = this.arr[i];  
  
 if (element % 2 == 0) {  
 System.*out*.print("$" + this.arr[i] + " ");  
 } else {  
 System.*out*.print(this.arr[i] + " ");  
 }  
  
 i++;  
 }  
 return null;  
 }  
 public Array(Array a){  
 this.amount = a.amount;  
 this.arr = a.arr;  
  
 }  
 public String getForSave(){  
 int i = 0;  
 String mass = "";  
  
 while (i < 10) {  
 int element = this.arr[i];  
  
 if (element % 2 == 0) {  
 mass = mass + "$" + Integer.*toString*(this.arr[i]) + " ";  
 } else {  
 mass = mass + Integer.*toString*(this.arr[i]) + " ";  
 }  
  
 i++;  
 }  
 return mass;  
 }  
 public void setArr(int chislo){  
 this.arr[0] = chislo;  
 }  
  
}

WorkString

package com.company;  
  
public class WorkString {  
 private String text;  
  
 WorkString() {  
 this.text = "adela, frfe";  
 }  
  
 WorkString(String text) {  
 this.text = text;  
 }  
  
 WorkString(WorkString s){  
 this(s.text);  
 }  
  
 private String createNew\_str() {  
 String[] word\_check = this.text.split("[,. ]+"); *//Разделяем строку на слова* for (int i = 0; i < word\_check.length; i++) {  
 *//Проверка первой и последней буквы* String wch = word\_check[i];  
 if (wch.toLowerCase().charAt(0) != wch.toLowerCase().charAt(wch.length() - 1))  
 this.text = this.text.replace(word\_check[i], "");  
 }  
 return this.text;  
 }  
  
 public String getNew\_str() {  
 return "После удаления слов,начинающихся\nи оканчивающихся на разные буквы,\n получилась такая строка:\n " + createNew\_str();  
 }  
  
 public String getForSaveNew(){  
 return createNew\_str();  
 }  
  
 private int countTotal() {  
 int before = this.text.length(); *//Подсчитываем длину текста* int after = this.text.replaceAll("[,.;!:?]", "").length(); *//Подсчитываем длину текста без знаков препинания* return before - after;  
 }  
  
 public String getTotal(){  
 String st;  
 if (countTotal() > 0)  
 st = "Количество знаков препинания в тексте:" + Integer.*toString*(countTotal());  
 else  
 st = "Увы, в заданном тексте мне не удалось найти знаки препинания";  
 return st;  
 }  
 public int getForSaveTotal(){  
 return countTotal();  
 }  
  
 public String choiceOf(int ch){  
 String save = "";  
 String k = Integer.*toString*(getForSaveTotal());;  
 if (ch == 1)  
 save = k;  
 if (ch == 2)  
 save = getForSaveNew();  
 return save;  
 }  
  
 public void setStr(String znak){  
 if (znak.equals("!") || znak.equals(".") || znak.equals("?") || znak.equals(",") || znak.equals(";"))  
 this.text += znak;  
 }  
 public String getStr(){  
 return this.text;  
 }  
}

Список использованных источников

1. Аршинский В.Л. Объектно-ориентированное программирование. Метод. указания по выполнению лабораторных работ / В.Л. Аршинский. – Иркутск: Изд-во ИРНИТУ, 2015. – 23 c.