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 21

к лабораторной работе по дисциплине:

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

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | АСУб-19-1 |  |  |  | Казаков С.Э. |
|  | 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.

Задание из 3 лабораторной работы:

a. Дан массив чисел. Необходимо упорядочить его по убыванию.

b. Отсортируйте слова в массиве по алфавиту.

c. Выберите из предложения все слова, которые имеют четный

порядковый номер.

2 Проектирование классов

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Проектирование классов | | | |
| **№** | **Название** | **Назначение** | **Поля (атрибуты)** |
| 1 | public class Main | Главный класс | - |
| 2 | public class MainWindow | Класс c 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 JTextField digits, static JTextField text3 – поля для пользовательского ввода |
| 3 | public class EventListener | Класс для обработки событий | private MainWindow m – получение полей класса MainWindow  private Do d – получение полей класса Do |
| 4 | public class TaskBC | Класс для выполнения заданий B и C | private String text;  String Z;  String WordsB;  String WordsC;  String[] str;  String out; |
| 5 | public class TaskA | Класс для выполнения задания A | private String text;  String Words;  String out; |
| 6 | public class WorkFile | Класс для работы с файлами | static String traceback – вывод сообщения об ошибке; |
| 7 | public class Do | Класс с диалоговым окном | private JLabel label1, static JTextArea label2 – формы с текстом |

**TaskBC**

Методы:

public TaskBC() – конструктор по умолчанию

public TaskBC(String text) – конструктор строки

private String Split(String C) – разделение строки для задания C

public String getTaskC() – вывод задания C

private String Sort(String TaskB) – сортировка для задания B

public String getTaskB() – вывод задания B

**TaskA**

Методы:

public TaskA() – конструктор по умолчанию

public TaskA(String text) – конструктор строки

private int[] Parse(String a) – перевод строки в массив чисел

public String getTaskA() – вывод задания A

private String Sort() – сортировка для задания A

**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, ввод/вывод из файла

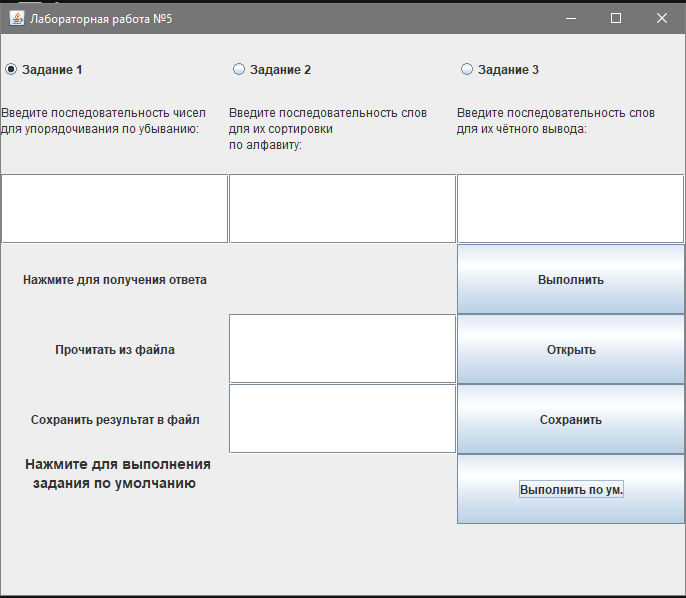


Рисунок 1 – Модель главного окна



Рисунок 2 – Модель диалогового окна

4 Таблица тестов

Таблица 2 – Таблица тестов.

|  |  |  |
| --- | --- | --- |
| **№** | **Словесное описание теста** | **Что должна сделать программа** |
| 1 | Нажатие на кнопку Выполнить | Вызов окна с выполнением |
| 2 | Нажатие на кнопку Сохранить | Сохранение результата файла |
| 3 | Нажатие на кнопку Открыть | Переход к диалоговому окну сохранения |
| 4 | Нажатие на кнопку Выполнить по умолчанию | Вызов окна с выполнением стандартных значений |
| 5 | Попытка ввода слов в задание A | Вывод ошибки |
| 6 | Ошибка ввода имени файла | Вывод ошибки |

5 Результаты тестирования

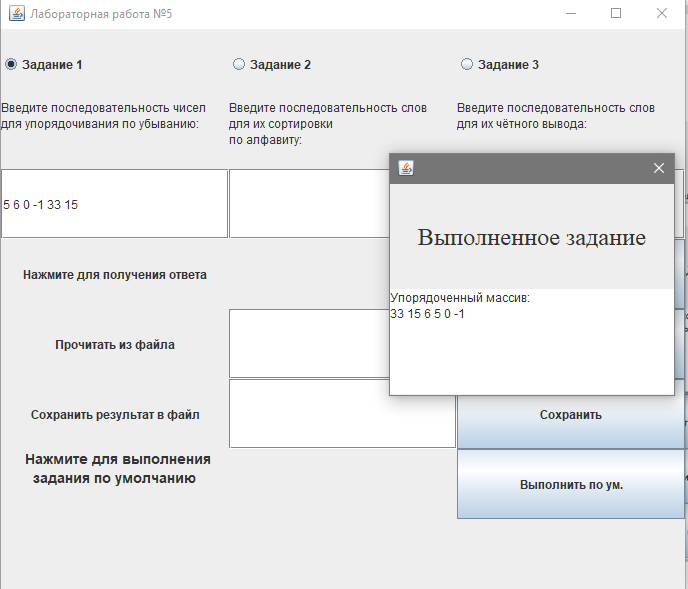


Рисунок 3 – Тест №1

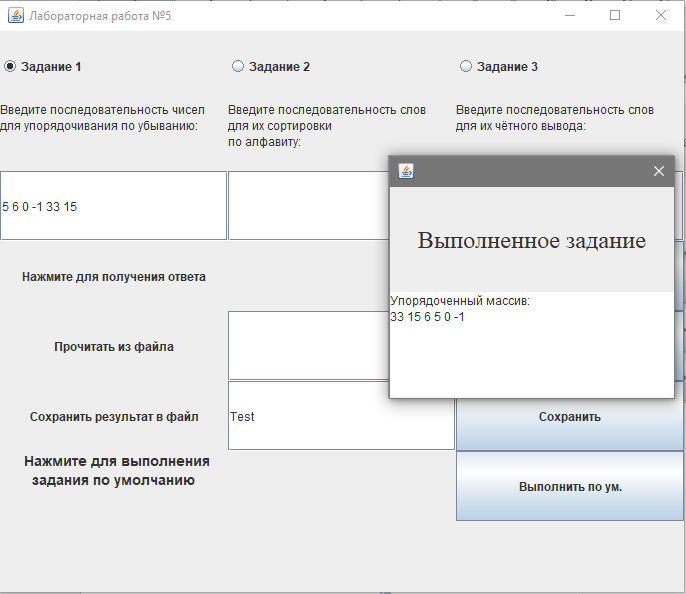


Рисунок 4 – Тест №2

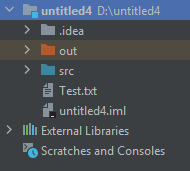


Рисунок 4.1 – Тест №2 (подтверждение создания файла)

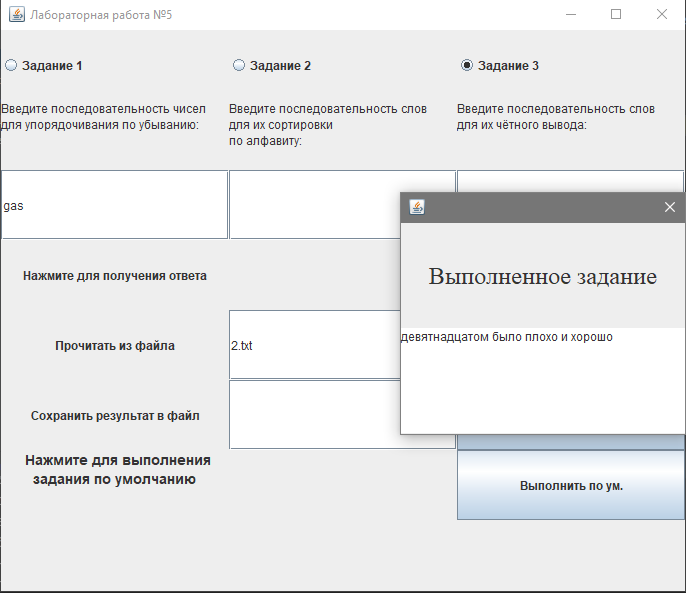
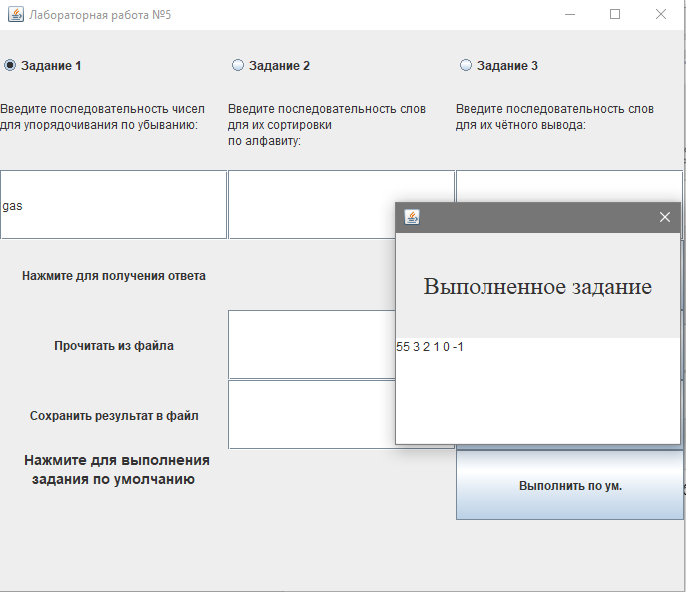


Рисунок 6 – Тест №4

Рисунок 5 – Тест №3



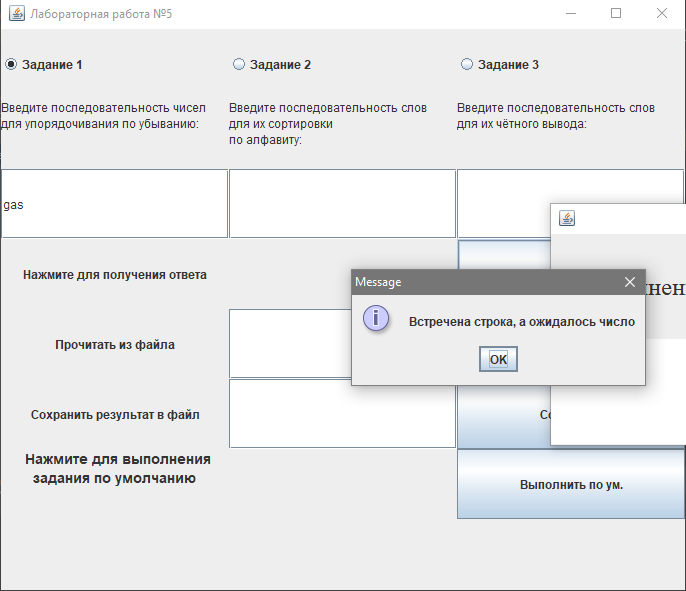


Рисунок 7 – Тест №5

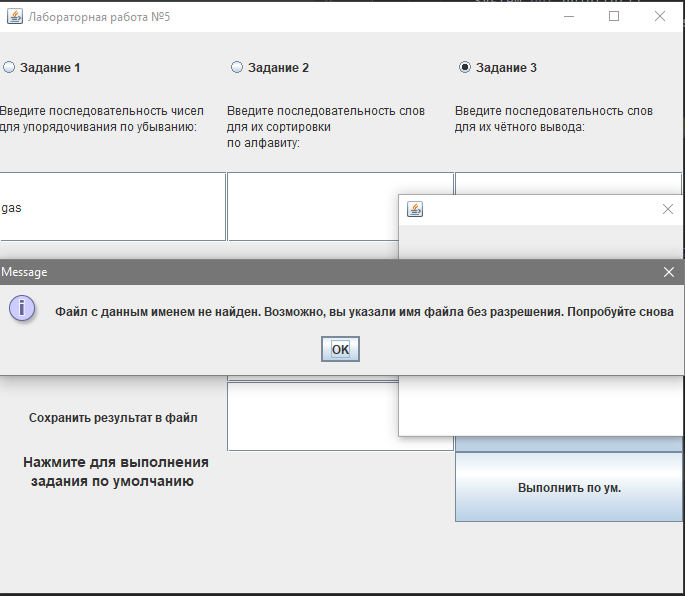


Рисунок 8 – Тест №6

6 Исходный код

MainWindow

package com.company;  
  
import javax.swing.\*;  
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 последовательность чисел\nдля упорядочивания по убыванию:");  
 private JTextArea label2 = new JTextArea("Введите последовательность слов\nдля их сортировки\nпо алфавиту:");  
 private JTextArea label3 = new JTextArea("Введите последовательность слов \nдля их чётного вывода:");  
 private JLabel label4 = new JLabel("Прочитать из файла");  
 private JLabel label6 = new JLabel("Нажмите для получения ответа");  
 private JLabel label7 = new JLabel("Сохранить результат\n в файл");  
 private JLabel label9 = new JLabel("");  
  
  
 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 JTextField *digits* = new JTextField();  
 static JTextField *text3* = new JTextField ();  
  
  
 public MainWindow(){  
 super("Лабораторная работа №5");  
  
  
 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*);  
  
  
 //изначально выбрана первая задача  
 *task1RadioButton*.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(*digits*);  
 container.add(*text*);  
 container.add(*text2*);  
  
 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.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 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");  
 }  
 }  
  
}

Do

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Do extends JDialog {  
 static JTextArea *label2* = new JTextArea();  
  
 public Do(){  
 //создаём контейнер, описывающий структуру окна  
 Container container = this.getContentPane();  
 container.setLayout(new GridLayout(2,1));  
 JLabel label1 = new JLabel("Выполненное задание");  
 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.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.IOException;  
  
public class EventListener implements ActionListener {  
 //поля для получения данных полей  
 public void actionPerformed(ActionEvent e) {  
 Do d = new Do();  
 d.setVisible(true);  
 switch (e.getActionCommand()) {  
 case "Выполнить":  
 if (MainWindow.*task2RadioButton*.isSelected()) { //Если выбрано 2 задание  
 TaskBC stroka = new TaskBC(MainWindow.*text*.getText());  
 d.*label2*.setText(stroka.getTaskB());  
 } else if (MainWindow.*task1RadioButton*.isSelected()) { //Если выбрано 1 задание  
 TaskA stroka = new TaskA(MainWindow.*digits*.getText());  
 d.*label2*.setText(stroka.getTaskA());  
 }  
 else{ //Если выбрано 3 задание  
 TaskBC stroka = new TaskBC(MainWindow.*text2*.getText());  
 d.*label2*.setText(stroka.getTaskC());  
 }  
 break;  
 //если была нажата кнопка "Save"  
 case "Сохранить":  
 WorkFile sv = new WorkFile();  
 if (MainWindow.*task2RadioButton*.isSelected()) {  
 d.setVisible(false);  
 sv.*writeFile*(d.*label2*.getText(), MainWindow.*text1*.getText());  
 if (!sv.*getTrace*().equals("")) {  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null, sv.*getTrace*());  
 sv.*setTrace*("");  
 }  
 } else if (MainWindow.*task3RadioButton*.isSelected()) {  
 try {  
 sv.*writeFile*(d.*label2*.getText(), MainWindow.*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(), MainWindow.*text1*.getText());  
 if (!sv.*getTrace*().equals("")) {  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null, sv.*getTrace*());  
 sv.*setTrace*("");  
 }  
 }  
 break;  
 }  
 //если была нажата кнопка "Open"  
 switch (e.getActionCommand()) {  
 case "Открыть":  
 WorkFile so = new WorkFile();  
  
 if (MainWindow.*task2RadioButton*.isSelected() == true) {  
 try {  
 String path = MainWindow.*text3*.getText();  
 String text;  
 text = WorkFile.*loadSFile*(path);  
 TaskBC stroka = new TaskBC(text);  
 d.*label2*.setText(stroka.getTaskB());  
 if (!so.*getTrace*().equals("")) {  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null, so.*getTrace*());  
 so.*setTrace*("");  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 } else if (MainWindow.*task3RadioButton*.isSelected() == true) {  
 try {  
 String path = MainWindow.*text3*.getText();  
 String text = null;  
 text = WorkFile.*loadSFile*(path);  
 TaskBC stroka = new TaskBC(text);  
 d.*label2*.setText(stroka.getTaskC());  
 if (!so.*getTrace*().equals("")) {  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null, so.*getTrace*());  
 so.*setTrace*("");  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 } else {  
 try {  
 String path = MainWindow.*text3*.getText();  
 String text = null;  
 text = WorkFile.*loadSFile*(path);  
 TaskA stroka = new TaskA(text);  
 d.*label2*.setText(stroka.getTaskA());  
 if (!so.*getTrace*().equals("")) {  
 d.setVisible(false);  
 JOptionPane.*showMessageDialog*(null, so.*getTrace*());  
 so.*setTrace*("");  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }  
 break;  
 }  
  
 switch (e.getActionCommand()) {  
 case "Выполнить по ум.":  
  
 if (MainWindow.*task2RadioButton*.isSelected()) {  
 TaskBC stroka = new TaskBC();  
 d.*label2*.setText(stroka.getTaskB());  
 } else if (MainWindow.*task1RadioButton*.isSelected()) {  
 TaskA stroka = new TaskA();  
 d.*label2*.setText(stroka.getTaskA());  
 } else {  
 TaskBC stroka = new TaskBC();  
 d.*label2*.setText(stroka.getTaskC());  
 }  
 break;  
 }  
 }  
}

TaskBC

package com.company;  
  
import java.lang.String;  
  
public class TaskBC {  
 private String text;  
 String Z;  
 String WordsB;  
 String WordsC;  
 String[] str;  
 String out;  
  
 public TaskBC() {  
 this.text = "one two";  
 }  
  
 public TaskBC(String text) {  
 this.text = text;  
 }  
  
 private String Split(String C) {  
 WordsC = "";  
 String[] words = C.split(" "); //Разделяем строку на слова  
 for (int i = 1; i < words.length; i += 2){  
 WordsC += words[i] + " ";  
 }  
 return WordsC;  
 }  
  
  
 public String getTaskC() {  
 out = Split(text);  
 return out;  
 }  
  
 private String Sort(String TaskB) {  
 str = TaskB.split(" ");  
 System.*out*.println(str.length);  
 for (int i = 0; i < str.length; i++){  
 System.*out*.println(i + ")" + str[i]);  
 }  
 for (int i = 0; i < str.length; i++) {  
 for (int j = 0; j < str.length - 1; j++) {  
 if (str[j].charAt(0) > str[j + 1].charAt(0)) {  
 Z = str[j];  
 str[j] = str[j + 1];  
 str[j + 1] = Z;  
 }  
 }  
 }  
 WordsB = "";  
 for (int i = 0; i < str.length; i++){  
 WordsB += str[i] + " ";  
 }  
 return WordsB;  
 }  
  
 public String getTaskB(){  
 out = Sort(text);  
 return out;  
 }  
}

TaskA

package com.company;  
  
  
import javax.swing.\*;  
  
public class TaskA {  
 private String text;  
 String Words;  
 String out;  
  
 public TaskA(){  
 this.text = "1 0 -1 2 3 55";  
 }  
  
 public TaskA(String text){  
 this.text = text;  
 }  
  
 private int[] Parse(String a) {  
 String[] str = a.split(" ");  
 System.*out*.println(str.length);  
 int[] arr = new int[str.length];  
 for (int i = 0; i < str.length; i++) {  
 System.*out*.println(i);  
 arr[i] = Integer.*parseInt*(str[i]);  
 }  
 return arr;  
 }  
  
 private String Sort(int[] digit){  
 int tmp;  
 for (int i = 0; i < digit.length; i++) {  
 for (int j = 0; j < digit.length - 1; j++) {  
 if (digit[j] < digit[j + 1]) {  
 tmp = digit[j];  
 digit[j] = digit[j + 1];  
 digit[j + 1] = tmp;  
 }  
 }  
 }  
 Words = "";  
 for (int i = 0; i < digit.length; i++){  
 Words += digit[i] + " ";  
 }  
 return Words;  
 }  
 public String getTaskA(){  
 try {  
 int[] digit = Parse(text);  
 out = Sort(digit);  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(null, "Встречена строка, а ожидалось число");  
 out = "";  
 } return out;  
 }  
}

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.