Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Курсовой проект

Разработка приложения с графическим интерфейсом для заданий 1-4.

По дисциплине: «Объектно-ориентированное программирование»

Выполнил
студент гр. в5130904/20021 Непомнящий Д. С.

Преподаватель Маслаков А. П.

Санкт-Петербург
2024

Оглавление

Введение..... 3

Постановка задачи 4

Файловая структура проекта..... 5

Исходный код 7

Скриншоты программы 31

Вывод..... 36

Введение

Для разработки на языке программирования Java нам потребуется специальный комплект инструментов, который называется JDK или Java Development Kit. Однако стоит отметить, что существуют разные реализации JDK, хотя все они используют один и тот же язык - Java. Две наиболее популярных реализации - Oracle JDK и OpenJDK.

Oracle JDK всецело развивается компанией Oracle. OpenJDK же представляет открытый проект, который развивается сообществом Java-разработчиков, а также рядом компаний, в том числе Oracle, Red Hat и рядом других.

Наибольшие различия с точки зрения лицензирования и поддержки. Согласно лицензии Oracle JDK можно использовать бесплатно для персональных нужд, а также для разработки, тестирования и демонстрации приложений. В остальных случаях (например, для получения поддержки) необходима коммерческая лицензия в виде подписки. А OpenJDK полностью бесплатна, поэтому он был выбран, как предпочтительный.

Также в работе использовался Java Swing. Он представляет собой фреймворк для создания графических пользовательских интерфейсов (GUI) в приложениях, разработанных на языке Java. Этот фреймворк обеспечивает разработчиков средствами для создания множества элементов интерфейса, таких как окна, кнопки, меню, панели и другие компоненты, позволяя создавать богатые и интерактивные приложения.

Постановка задачи

Разработать приложение с графическим интерфейсом для заданий 1–4.

Для этого приложения должна быть реализована возможность выбора из списка любого приложения, ввод входных данных и его выполнение.

Модифицировать задания 1–4 так, чтобы весь вывод происходил в текстовых областях, защищённых от редактирования.

Предусмотреть для заданий:

- 3 - выбор файлов словаря и текста для перевода, возможность ручного ввода текста
- 4 - ввод входных данных для методов

Файловая структура проекта

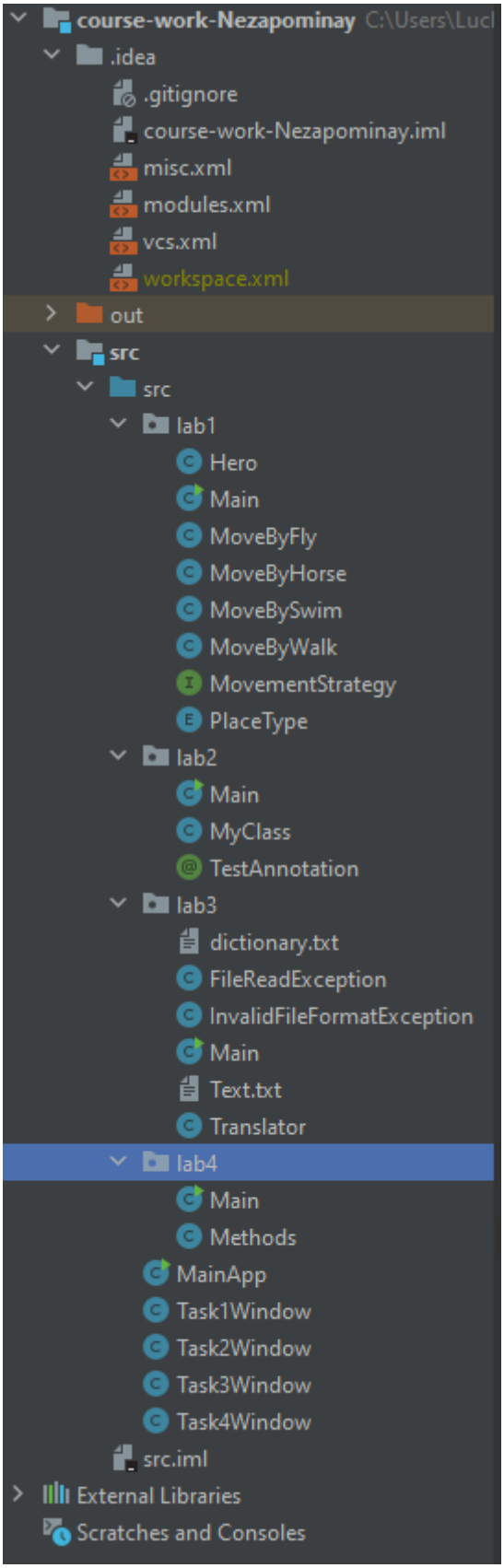

Рисунок 1 – файловая структура проекта.

На рисунке 1 мы видим исходные программы в папках lab1-lab4, а также в корневой папке src расположились файлы реализации с помощью графического интерфейса.

Исходный код

Код файла .\src\lab1\Hero.class

```
package lab1;

public class Hero {

    String name = "";

    MovementStrategy movement;

    private final PlaceType placeFrom;

    private final PlaceType placeIn;

    public Hero(String name, MovementStrategy movement, PlaceType placeFrom, PlaceType placeIn) {

        this.placeFrom = placeFrom;

        this.placeIn = placeIn;

        this.name = name;

        this.movement = movement;

    }

    public void move() {

        System.out.println("Герой по имени " + name);

        movement.move();

        System.out.println("Из " + placeFrom + " в " + placeIn);

    }

    public void setMovement(MovementStrategy movement) {

        this.movement = movement;

    }

    public String getName() {

        return name;

    }

    public PlaceType getPlaceFrom() {

        return placeFrom;

    }

    public PlaceType getPlaceIn() {

        return placeIn;

    }
```

```
    public String getMovementDescription() {
        return movement.getDescription();
    }
}
```

Код файла .\ src\lab1\MoveByFly.class

```
package lab1;
public class MoveByFly implements MovementStrategy{
    @Override
    public void move() {
        System.out.println("Летит");
    }
    public String getDescription() {
        return "Летит";
    }
}
```

Код файла .\ src\lab1\MoveByHorse.class

```
package lab1;
public class MoveByHorse implements MovementStrategy{

    @Override
    public void move() {
        System.out.println("Едет на лошади");
    }
    public String getDescription() {
        return "Едет на лошади";
    }
}
```

Код файла .\ src\lab1\MoveBySwim.class

```
package lab1;
public class MoveBySwim implements MovementStrategy{
    @Override
    public void move() {
        System.out.println("Плывёт");
```

```
    }
    public String getDescription() {
        return "Плывёт";
    }
}
```

Код файла .\ src\lab1\MoveByWalk.class

```
package lab1;
public class MoveByWalk implements MovementStrategy{
    @Override
    public void move() {
        System.out.println("Идёт пешком");
    }
    public String getDescription() {
        return "Идет пешком";
    }
}
```

Код файла .\ src\lab1\MovementStrategy.interface

```
package lab1;
public interface MovementStrategy {
    void move();
    String getDescription();
}
```

Код файла .\ src\lab1\PlaceType.enum

```
package lab1;
public enum PlaceType {
    Town, Camp, River, Tower
}
```

Код файла .\ src\lab2\MyClass.class

```
package lab2;
public class MyClass {
    @TestAnnotation(1)
    public void methodPublic1(int a, int b)
    {
```

```
        System.out.println("Первый публичный метод с параметрами: " + a
+ " " + b);
    }
    public void methodPublic2(int a, int b)
    {
        System.out.println("Второй публичный метод с параметрами: " + a
+ " " + b);
    }
    private void methodPrivate1(int a, int b)
    {
        System.out.println("Первый приватный метод с параметрами: " + a
+ " " + b);
    }

    @TestAnnotation(2)
    private void methodPrivate2(int a, double b)
    {
        System.out.println("Второй приватный метод с параметрами: " + a
+ " " + b);
    }
    @TestAnnotation(1)
    protected void methodProtected1(int a, int b)
    {
        System.out.println("Первый защищённый метод с параметрами: " +
a + " " + b);
    }
    @TestAnnotation(5)
    protected void methodProtected2(boolean a, int b)
    {
        System.out.println("Второй защищённый метод с параметрами: " +
a + " " + b);
    }
}
```

Код файла .\ src\lab2\TestAnnotation.annotation

```
package lab2;
```

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface TestAnnotation {
    int value();
}
```

Код файла .\ src\lab3\Translator.class

```
package lab3;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Paths;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
public class Translator {
    private Map<String, String> dictionary = new LinkedHashMap<>();
    private boolean isDictionaryLoaded = false;
    public void loadDictionary(String filePath) throws
InvalidFileFormatException, FileReadException {
        dictionary.clear();
        try {
            List<String> lines =
Files.readAllLines(Paths.get(filePath));
            for (String line : lines) {
                String[] parts = line.split("\\|");
                if (parts.length != 2) {
                    throw new InvalidFileFormatException("Неверный
формат строки в словаре: " + line);
                }
```

```
                dictionary.put(parts[0].trim().toLowerCase(),
parts[1].trim());
            }
            isDictionaryLoaded = true;
        } catch (NoSuchFileException e) {
            throw new FileReadException("Файл не найден: " + filePath);
        } catch (IOException e) {
            throw new FileReadException("Ошибка при чтении файла: " +
filePath);
        }
    }
    public boolean isDictionaryLoaded() {
        return isDictionaryLoaded;
    }
    public String translate(String text) {
        if (!isDictionaryLoaded) {
            return "";
        }
        String[] words = text.split("\\s+");
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < words.length; i++) {
            String longestMatch = null;
            String translation = words[i];
            for (String key : dictionary.keySet()) {
                String[] keyWords = key.split("\\s+");
                if (matches(words, i, keyWords)) {
                    if (longestMatch == null || keyWords.length >
longestMatch.split("\\s+").length) {
                        longestMatch = key;
                    }
                }
            }
            if (longestMatch != null) {
                translation = dictionary.get(longestMatch);
```

```
                i += longestMatch.split("\\s+").length - 1;
            }
            result.append(translation).append(" ");
        }
        return result.toString().trim();
    }
    private boolean matches(String[] words, int start, String[]
keyWords) {
        if (start + keyWords.length > words.length) return false;
        for (int j = 0; j < keyWords.length; j++) {
            if (!words[start + j].equalsIgnoreCase(keyWords[j])) {
                return false;
            }
        }
        return true;
    }
}
```

Код файла .\ src\lab3\FileReadException.class

```
package lab3;
public class FileReadException extends Exception {
    public FileReadException(String message) {
        super(message);
    }
}
```

Код файла .\ src\lab3\InvalidFileFormatException.class

```
package lab3;
public class InvalidFileFormatException extends Exception {
    public InvalidFileFormatException(String message) {
        super(message);
    }
}
```

Код файла .\ src\lab4\Methods.class

```
package lab4;
```

```
import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class Methods {
    public static Map<Character, String> streamTransformToMap(List<String> list) {
        return list.stream()
                .collect(Collectors.toMap(str -> str.charAt(0), str -> str.substring(1)));
    }
    public static int streamEvenNumbers(int[] numbers) {
        return Arrays.stream(numbers)
                .filter(x -> x % 2 == 0)
                .reduce(0, Integer::sum);
    }
    public static <T> String streamLastElement(Collection<T> collection) {
        return collection.stream()
                .reduce((x, y) -> y)
                .map(Object::toString)
                .orElseThrow(() -> new NoSuchElementException("Ни одного элемента не найдено"));
    }

    public static String streamToUpper(List<String> words) {
                return words.stream()
                .map(str -> str.toUpperCase())
                .map(str -> str = "_new_" + str)
                .collect(Collectors.joining("\n"));
    }

    public static double streamAverage(List<Integer> numbers) {
        return numbers.stream().mapToDouble(Integer::doubleValue)
                .reduce(0, (x, y) -> x + y) / numbers.stream().count();
```

```
    }
    public static String streamUniqueSquare(List<Integer> numbers) {
        return numbers.stream()
                .filter(n -> Collections.frequency(numbers, n) == 1)
                .map(n -> n * n)
                .map(String::valueOf)
                .collect(Collectors.joining("\n"));
    }
}
```

Код файла .\src\Task1.class

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import lab1.Hero;
import lab1.PlaceType;
import  lab1.MoveByWalk;
import  lab1.MoveByFly;
import  lab1.MoveBySwim;
import  lab1.MoveByHorse;

public class Task1Window extends JFrame {
    private Hero hero;
    private JComboBox<String> movementSelector;
    private JTextArea outputArea;

    public Task1Window() {
        setTitle("Задание 1: Герой");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(400, 300);
        setLayout(new BorderLayout());
```

```
        hero = new Hero("Иван", new MoveByFly(), PlaceType.Camp,
PlaceType.Tower);

        String[] movements = {"Пешком", "На лошади", "Лететь", "Плыть"};
        movementSelector = new JComboBox<>(movements);
        add(movementSelector, BorderLayout.NORTH);

        outputArea = new JTextArea();
        outputArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(outputArea);
        add(scrollPane, BorderLayout.CENTER);


        JButton moveButton = new JButton("Переместить героя");
        add(moveButton, BorderLayout.SOUTH);

        moveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int                    selectedIndex                    =
movementSelector.getSelectedIndex();
                switch (selectedIndex) {
                    case 0:
                        hero.setMovement(new MoveByWalk());
                        break;
                    case 1:
                        hero.setMovement(new MoveByHorse());
                        break;
                    case 2:
                        hero.setMovement(new MoveByFly());
                        break;
                    case 3:
                        hero.setMovement(new MoveBySwim());
                        break;
                }
```

```
                updateOutput();
            }
        });

        setLocationRelativeTo(null);
        setVisible(true);
    }

    private void updateOutput() {
        StringBuilder output = new StringBuilder();
        output.append("Герой по имени ").append(hero.getName()).append("\n");
        output.append(hero.getMovementDescription()).append("\n");
        output.append("Из ").append(hero.getPlaceFrom()).append(" в ").append(hero.getPlaceIn()).append("\n");
        outputArea.setText(output.toString());
    }
}
```

Код файла .\src\Task2.class

```
import javax.swing.*;
import java.awt.*;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import lab2.MyClass;
import lab2.TestAnnotation;

public class Task2Window extends JFrame {
    private JTextArea outputArea;

    public Task2Window() {
        setTitle("Задание 2: Вызов аннотированных методов");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(600, 400);
```

```
        setLayout(new BorderLayout());

        outputArea = new JTextArea();
        outputArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(outputArea);
        add(scrollPane, BorderLayout.CENTER);

        JButton executeButton = new JButton("Выполнить");
        executeButton.addActionListener(e -> executeAnnotatedMethods());
        add(executeButton, BorderLayout.SOUTH);

        setLocationRelativeTo(null);
        setVisible(true);
    }

    private void executeAnnotatedMethods() {
        outputArea.setText("");
        MyClass myClass = new MyClass();

        Method[] methods = MyClass.class.getDeclaredMethods();

        for (Method method : methods) {
            if (method.isAnnotationPresent(TestAnnotation.class)) {
                TestAnnotation annotation = method.getAnnotation(TestAnnotation.class);
                int repeatCount = annotation.value();
                int modifiers = method.getModifiers();

                if (Modifier.isProtected(modifiers) || Modifier.isPrivate(modifiers)) {
                    method.setAccessible(true);

                    for (int i = 0; i < repeatCount; i++) {
                        try {
```

```
                            Object[]            parameters            =
generateDefaultParameters(method);

                            outputArea.append("Вызов    метода:    "    +
method.getName() + "\n");
                            method.invoke(myClass, parameters);

                        }      catch      (IllegalAccessException      |
InvocationTargetException ex) {
                            outputArea.append("Ошибка     при     вызове
метода: " + ex.getMessage() + "\n");
                        }
                    }
                }
            }
        }

    private Object[] generateDefaultParameters(Method method) {
        Class<?>[] paramTypes = method.getParameterTypes();
        Object[] parameters = new Object[paramTypes.length];

        for (int i = 0; i < paramTypes.length; i++) {
            if (paramTypes[i] == int.class) {
                parameters[i] = 0;
            } else if (paramTypes[i] == double.class) {
                parameters[i] = 0.0;
            } else if (paramTypes[i] == boolean.class) {
                parameters[i] = false;
            } else {
                parameters[i] = null;
            }
        }

        return parameters;
```

```
    }
}
```

Код файла .\src\Task3.class

```
import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;

import lab3.Translator;
import lab3.FileReadException;
import lab3.InvalidFileFormatException;

public class Task3Window extends JFrame {
    private JTextField dictionaryPathField;
    private JTextField textPathField;
    private JTextArea inputTextArea;
    private JTextArea outputTextArea;
    private Translator translator;

    public Task3Window() {
        setTitle("Задание 3: Перевод текста");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(600, 600);
        setLayout(new BorderLayout());

        translator = new Translator();

        JPanel topPanel = new JPanel(new GridLayout(2, 1));

        JPanel dictionaryPanel = new JPanel(new BorderLayout());
        dictionaryPathField = new JTextField();
        dictionaryPathField.setEditable(false);
        JButton chooseDictionaryButton = new JButton("Выбрать словарь");
```

```
        chooseDictionaryButton.addActionListener(e                    ->
chooseDictionaryFile());
        dictionaryPanel.add(new          JLabel("Словарь:          "),
BorderLayout.WEST);
        dictionaryPanel.add(dictionaryPathField, BorderLayout.CENTER);
        dictionaryPanel.add(chooseDictionaryButton, BorderLayout.EAST);
        topPanel.add(dictionaryPanel);

        JPanel textPanel = new JPanel(new BorderLayout());
        textPathField = new JTextField();
        textPathField.setEditable(false);
        JButton chooseTextButton = new JButton("Выбрать текст");
        chooseTextButton.addActionListener(e -> chooseTextFile());
        textPanel.add(new JLabel("Текст: "), BorderLayout.WEST);
        textPanel.add(textPathField, BorderLayout.CENTER);
        textPanel.add(chooseTextButton, BorderLayout.EAST);
        topPanel.add(textPanel);

        add(topPanel, BorderLayout.NORTH);

        JPanel centerPanel = new JPanel(new GridLayout(2, 1, 5, 5));
        inputTextArea = new JTextArea();
        inputTextArea.setLineWrap(true);
        inputTextArea.setWrapStyleWord(true);
        JScrollPane inputScroll = new JScrollPane(inputTextArea);

inputScroll.setBorder(BorderFactory.createTitledBorder("Введите   текст
для перевода"));

        outputTextArea = new JTextArea();
        outputTextArea.setEditable(false);
        outputTextArea.setLineWrap(true);
        outputTextArea.setWrapStyleWord(true);
        JScrollPane outputScroll = new JScrollPane(outputTextArea);
```

```
outputScroll.setBorder(BorderFactory.createTitledBorder("Переведённый
текст"));

        centerPanel.add(inputScroll);
        centerPanel.add(outputScroll);
        add(centerPanel, BorderLayout.CENTER);

        JButton translateButton = new JButton("Перевести");
        translateButton.addActionListener(e -> translateText());
        add(translateButton, BorderLayout.SOUTH);

        setLocationRelativeTo(null);
        setVisible(true);
    }

    private void chooseDictionaryFile() {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Выберите файл словаря");

        int result = fileChooser.showOpenDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();

dictionaryPathField.setText(selectedFile.getAbsolutePath());

            try {

translator.loadDictionary(selectedFile.getAbsolutePath());
                JOptionPane.showMessageDialog(this, "Словарь успешно
загружен.", "Успех", JOptionPane.INFORMATION_MESSAGE);
            } catch (InvalidFileFormatException | FileReadException e)
{
                JOptionPane.showMessageDialog(this, "Ошибка: " +
e.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
```

```
        }
    }

    private void chooseTextFile() {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Выберите файл текста");

        int result = fileChooser.showOpenDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            textPathField.setText(selectedFile.getAbsolutePath());

            try {
                String text = Files.readString(selectedFile.toPath());
                inputTextArea.setText(text);
                JOptionPane.showMessageDialog(this, "Текст успешно загружен.", "Успех", JOptionPane.INFORMATION_MESSAGE);
            } catch (IOException e) {
                JOptionPane.showMessageDialog(this, "Ошибка при чтении файла: " + e.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    private void translateText() {
        String inputText = inputTextArea.getText().trim();
        if (inputText.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Введите текст для перевода или загрузите файл текста.", "Ошибка", JOptionPane.WARNING_MESSAGE);
            return;
        }

        if (!translator.isDictionaryLoaded()) {
            JOptionPane.showMessageDialog(this, "Словарь не загружен.", "Ошибка", JOptionPane.WARNING_MESSAGE);
```

```
            return;
        }

        String translatedText = translator.translate(inputText);
        outputTextArea.setText(translatedText);
    }
}
```

Код файла .\src\Task4.class

```
import javax.swing.*;
import java.awt.*;
import java.util.*;
import java.util.List;
import java.util.stream.Collectors;
import lab4.Methods;

public class Task4Window extends JFrame {

    private final JComboBox<String> methodSelector;
    private final JPanel inputPanel;
    private final JTextArea outputArea;

    public Task4Window() {
        setTitle("Задание 4: Вызов методов");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(600, 500);
        setLayout(new BorderLayout());

        methodSelector = new JComboBox<>(new String[]{
                "streamAverage",
                "streamToUpper",
                "streamUniqueSquare",
                "streamLastElement",
                "streamEvenNumbers",
                "streamTransformToMap"
```

```
        });

        inputPanel = new JPanel(new GridLayout(0, 2, 10, 10));

        JButton executeButton = new JButton("Выполнить");
        executeButton.addActionListener(e -> executeMethod());

        outputArea = new JTextArea();
        outputArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(outputArea);

        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.add(new JLabel("Выберите метод:"), BorderLayout.WEST);
        topPanel.add(methodSelector, BorderLayout.CENTER);

        JPanel centerPanel = new JPanel(new BorderLayout());
        centerPanel.add(inputPanel, BorderLayout.NORTH);
        centerPanel.add(executeButton, BorderLayout.SOUTH);

        add(topPanel, BorderLayout.NORTH);
        add(centerPanel, BorderLayout.CENTER);
        add(scrollPane, BorderLayout.SOUTH);

        methodSelector.addActionListener(e -> updateInputFields());
        updateInputFields();

        setVisible(true);
    }

    private void updateInputFields() {
        inputPanel.removeAll();

        String selectedMethod = (String)
methodSelector.getSelectedItem();
```

```
        switch (selectedMethod) {
            case "streamAverage":
                inputPanel.add(new JLabel("Введите список чисел (через
запятую):"));
                inputPanel.add(new JTextField());
                break;
            case "streamToUpper":
                inputPanel.add(new JLabel("Введите список слов (через
запятую):"));
                inputPanel.add(new JTextField());
                break;
            case "streamUniqueSquare":
                inputPanel.add(new JLabel("Введите список чисел (через
запятую):"));
                inputPanel.add(new JTextField());
                break;
            case "streamLastElement":
                inputPanel.add(new JLabel("Введите список строк (через
запятую):"));
                inputPanel.add(new JTextField());
                break;
            case "streamEvenNumbers":
                inputPanel.add(new JLabel("Введите массив чисел (через
запятую):"));
                inputPanel.add(new JTextField());
                break;
            case "streamTransformToMap":
                inputPanel.add(new JLabel("Введите список строк (через
запятую):"));
                inputPanel.add(new JTextField());
                break;
        }

        inputPanel.revalidate();
        inputPanel.repaint();
    }
```

```
    private void executeMethod() {
        String selectedMethod = (String) methodSelector.getSelectedItem();
        Component[] components = inputPanel.getComponents();
        String input = "";

        for (Component comp : components) {
            if (comp instanceof JTextField) {
                input = ((JTextField) comp).getText();
                break;
            }
        }

        try {
            String result;

            switch (selectedMethod) {
                case "streamAverage":
                    List<Integer> numbers = Arrays.stream(input.split(","))
                            .map(String::trim)
                            .map(Integer::parseInt)
                            .collect(Collectors.toList());
                    result = "Среднее значение: " + Methods.streamAverage(numbers);
                    break;
                case "streamToUpper":
                    List<String> words = Arrays.stream(input.split(","))
                            .map(String::trim)
                            .collect(Collectors.toList());
                    result = "Слова в верхнем регистре:\n" + Methods.streamToUpper(words);
                    break;
                case "streamUniqueSquare":
```

```
                    List<Integer>            uniqueNumbers            =
Arrays.stream(input.split(","))
                            .map(String::trim)
                            .map(Integer::parseInt)
                            .collect(Collectors.toList());
                    result   =   "Квадраты   уникальных   чисел:\n"   +
Methods.streamUniqueSquare(uniqueNumbers);
                    break;
                case "streamLastElement":
                    List<String>               elements                =
Arrays.stream(input.split(","))
                            .map(String::trim)
                            .collect(Collectors.toList());
                    result     =     "Последний     элемент:     "     +
Methods.streamLastElement(elements);
                    break;
                case "streamEvenNumbers":
                    int[] array = Arrays.stream(input.split(","))
                            .map(String::trim)
                            .mapToInt(Integer::parseInt)
                            .toArray();
                    result    =    "Сумма    чётных    чисел:    "    +
Methods.streamEvenNumbers(array);
                    break;
                case "streamTransformToMap":
                    List<String>              listForMap              =
Arrays.stream(input.split(","))
                            .map(String::trim)
                            .collect(Collectors.toList());
                    result    =    "Преобразование    в    Map:\n"    +
Methods.streamTransformToMap(listForMap);
                    break;
                default:
                    result = "Неизвестный метод.";
                    break;
            }
```

```
            outputArea.setText(result);
        } catch (Exception e) {
            outputArea.setText("Ошибка при выполнении метода: " +
e.getMessage());
        }
    }
}
```

Код файла .\src\MainApp.class

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

public class MainApp extends JFrame {
    private JComboBox<String> appSelector;
    private JButton runButton;

    public MainApp() {
        setTitle("Выбор приложения");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 200);
        setLayout(new BorderLayout());

        String[] apps = {"Задание 1", "Задание 2", "Задание 3", "Задание
4"};
        appSelector = new JComboBox<>(apps);
        add(appSelector, BorderLayout.CENTER);

        runButton = new JButton("Запустить");
        add(runButton, BorderLayout.SOUTH);

        runButton.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) {
                String selectedApp = (String) appSelector.getSelectedItem();
                launchApp(selectedApp);
            }
        });

        setLocationRelativeTo(null);
        setVisible(true);
    }

    private void launchApp(String appName) {
        switch (appName) {
            case "Задание 1":
                new Task1Window();
                break;
            case "Задание 2":
                new Task2Window();
                break;
            case "Задание 3":
                new Task3Window();
                break;
            case "Задание 4":
                new Task4Window();
                break;
            default:
                JOptionPane.showMessageDialog(this, "Неизвестное задание");
        }
    }
    public static void main(String[] args) {
        new MainApp();
    }
}
```

Скриншоты программы

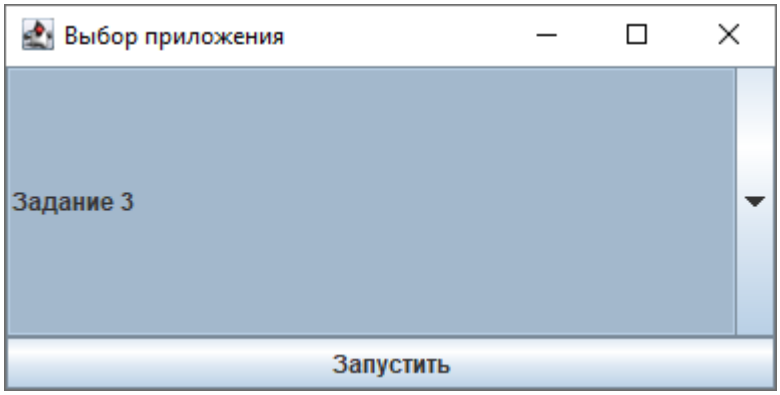

Рисунок 2 – запуск выбранного задания

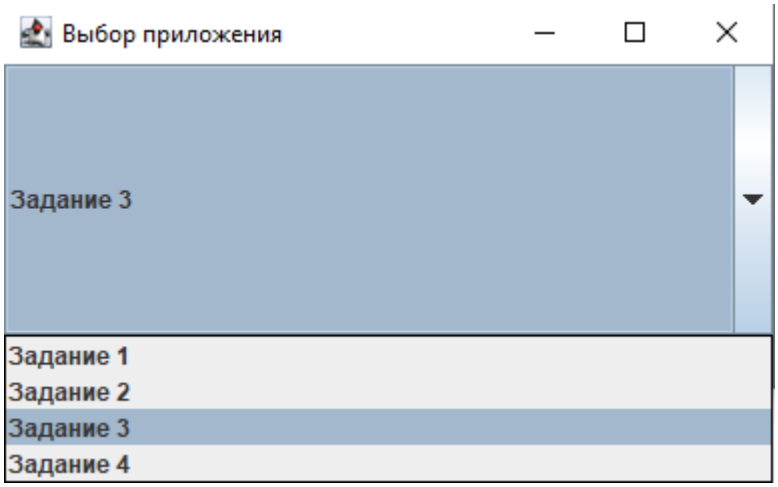

Рисунок 3 – меню выбора задания

Запуск первого задания

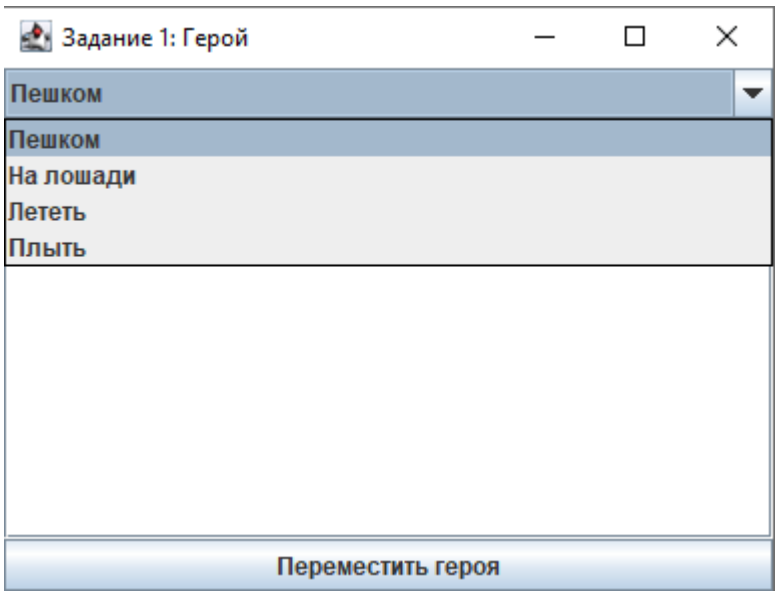

Рисунок 4 – выбор стратегии перемещения

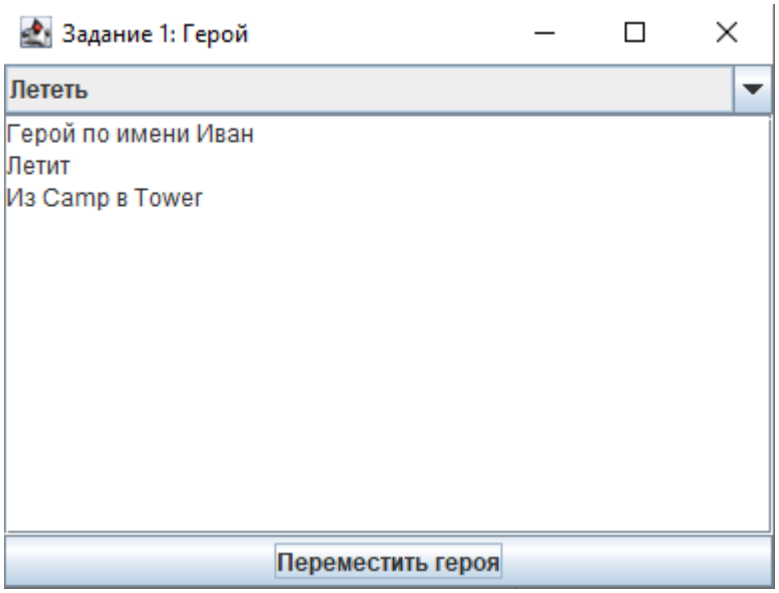

Рисунок 5 – результат с выбранной стратегией

Запуск второго задания

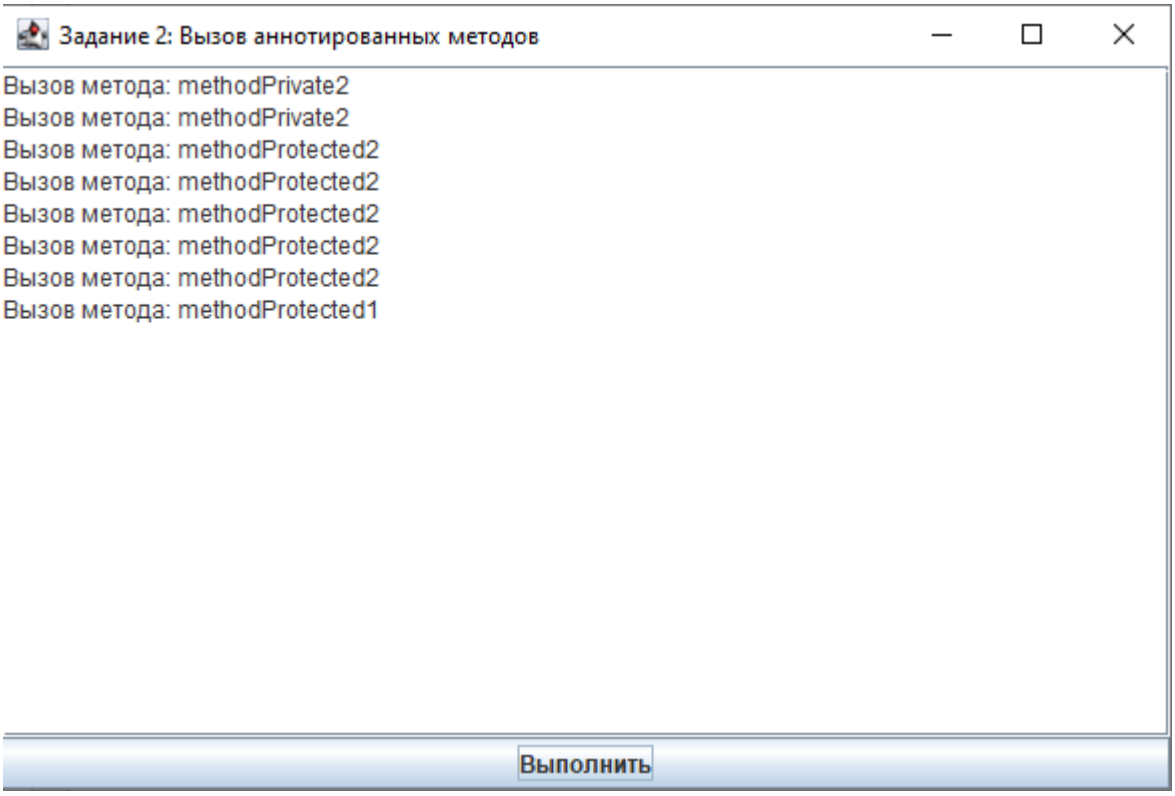

Рисунок 6 – результат выполнения программы второго задания

Запуск третьего задания

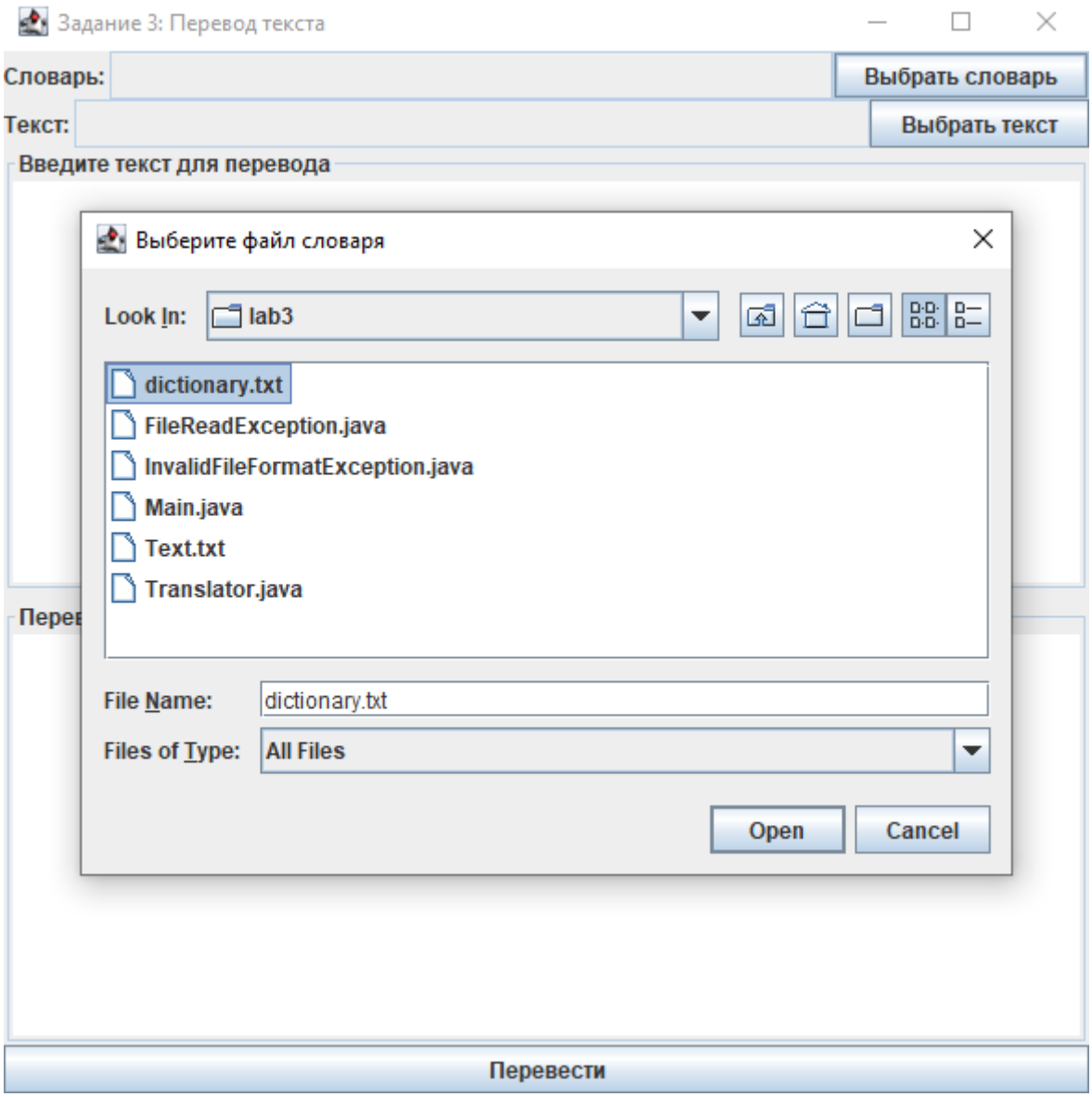

Рисунок 7 – выбор словаря

Выбор текста для перевода по аналогии

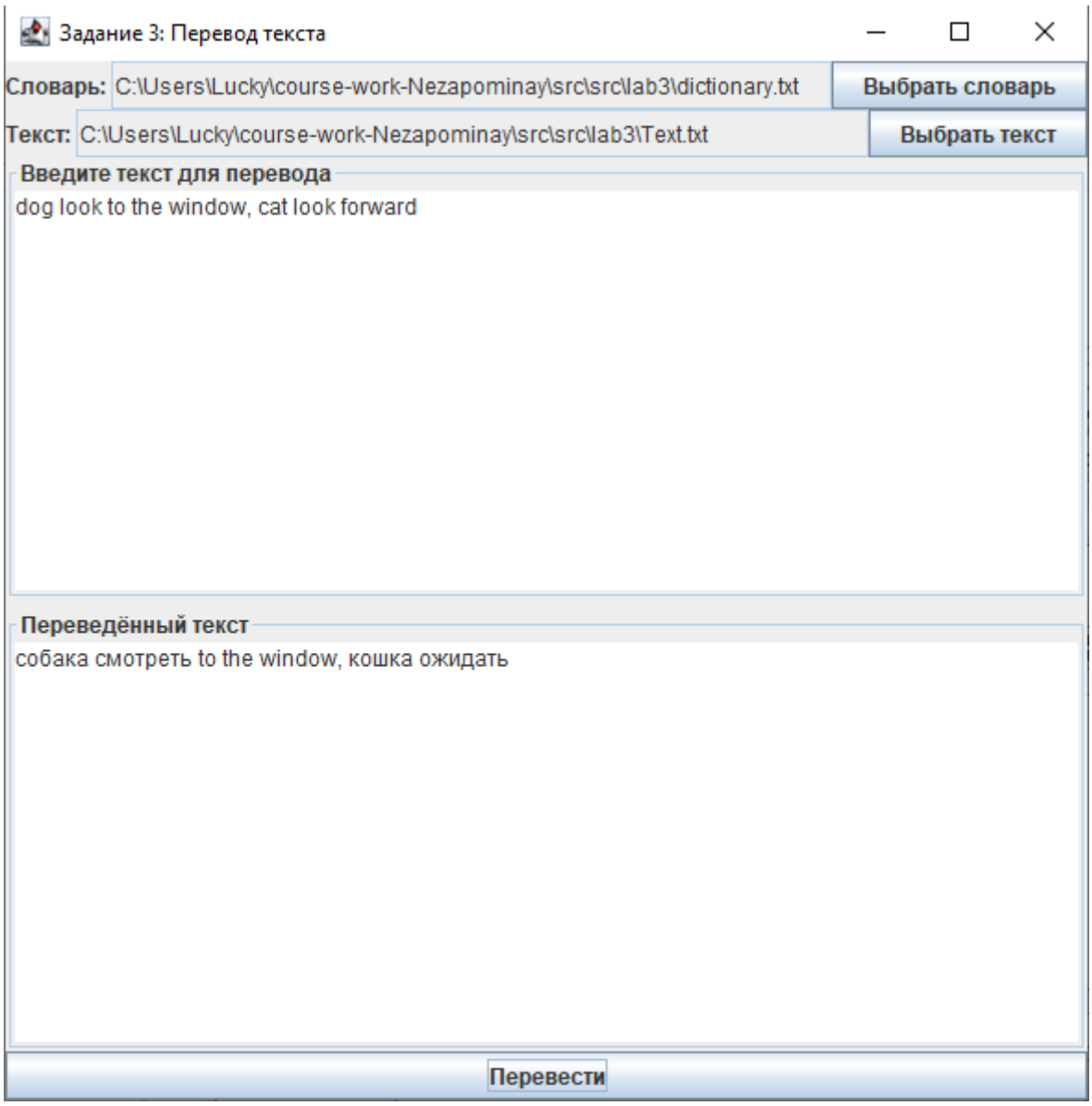

Рисунок 8 – результат выполнения

Запуск четвертого задания и выбор одного из методов для выполнения

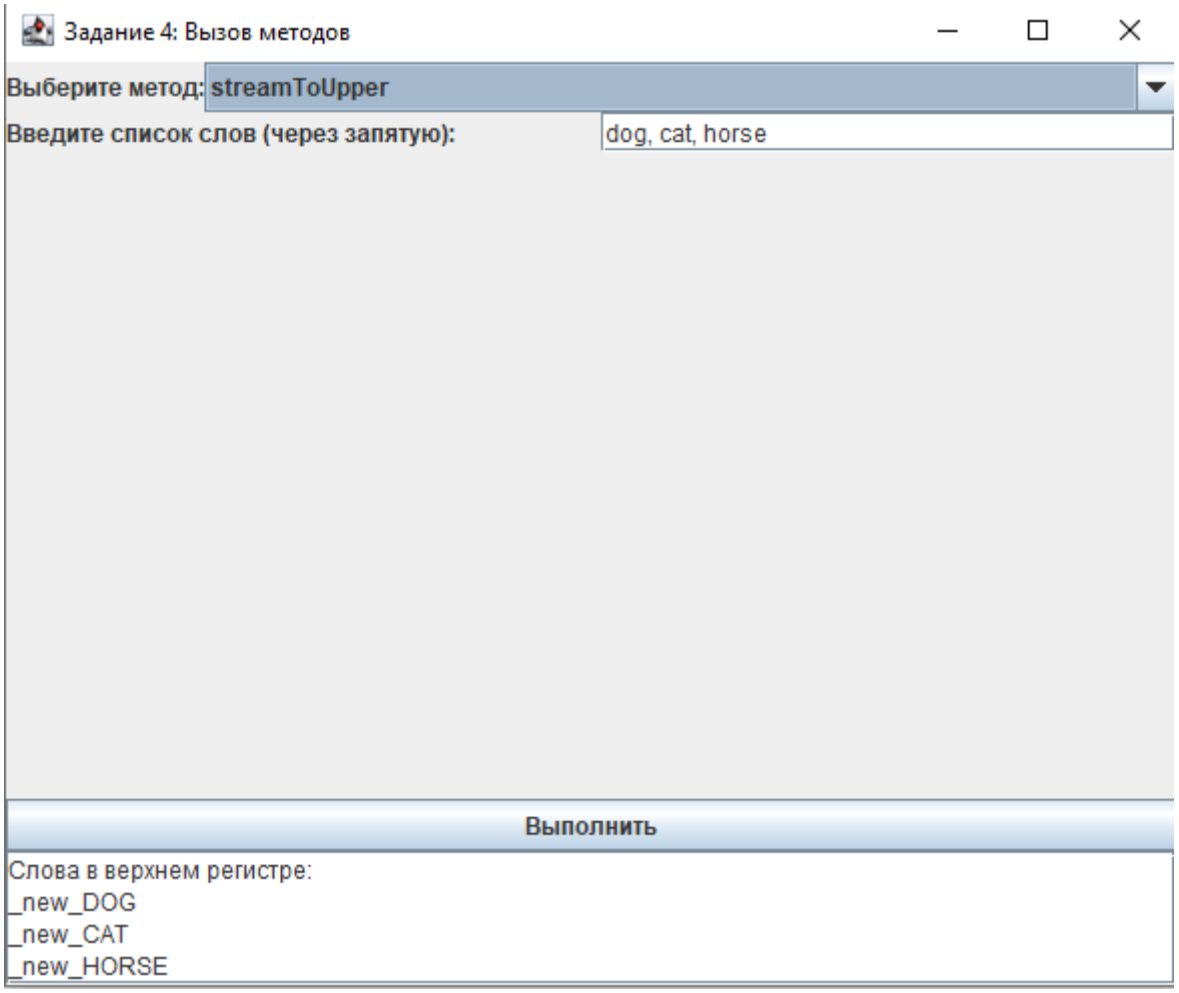

Рисунок 9 – результат выполнения

Вывод

На основе выполненной работы, было разработано приложение с графическим интерфейсом для выполнения лабораторных работа 1-4.

Целью данной разработки было предоставление пользователю удобного и интуитивно понятного средства для запуска лабораторной работы, включая выбор задания, ввод входных данных и получение результатов.

В процессе разработки приложения были достигнуты следующие ключевые цели:

- создание графического интерфейса, включая элементы управления, такие как выпадающий список, поля ввода и текстовые области
- возможность выбора задания
- ввод входных данных, в зависимости от выбранной лабораторной работы, приложение предоставляет соответствующие поля для ввода данных.
- выполнение задания и отображение результатов в текстовых областях. Текстовые области защищены от редактирования, обеспечивая надёжный вывод результатов.