ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Курсовая работа

по дисциплине
«Объектно-ориентированное программирование»

Выполнил:
студент группы в5130904/20322
Иванов Н.А.

Проверил:
преподаватель Маслаков А.П.

Санкт-Петербург
2024

Содержание

1. Задание....3
2. Ход работы....3
2.1. Выбор технологий....3
2.2. Диаграммы классов....4
2.2.1. Лабораторная работа 1....4
2.2.2. Лабораторная работа 2....4
2.2.3. Лабораторная работа 3....5
2.2.4. Лабораторная работа 4....5
2.2.5. Курсовая работа....5
2.3. Разработка приложения....6
3. Вывод....12
Приложение А. Исходный код лабораторной работы 1....13
А.1. Класс «Main»....13
А.2. Класс «Hero»....15
А.3. Интерфейс «IMovingStrategy»....16
А.4. Класс «IMovingStrategyNone»....16
А.5. Класс «IMovingStrategyWalking»....17
А.6. Класс «IMovingStrategyRunning»....17
А.7. Класс «IMovingStrategyOnHorse»....17
А.8. Класс «IMovingStrategyFlying»....18
Приложение Б. Исходный код лабораторной работы 2....19
Б.1. Класс «Main»....19
Б.2. Класс «MyAnnotation»....20
Б.3. Класс «MyClass»....21
Приложение В. Исходный код лабораторной работы 3....23
В.1. Класс «Main»....23
В.2. Класс «Dictionary»....24
В.3. Класс «FileReadException»....26
В.4. Класс «InvalidFileFormatException»....26
Приложение Г. Исходный код лабораторной работы 4....27
Г.1. Класс «Main»....27
Приложение Д. Исходный код курсовой работы....30
Д.1. Класс «Main»....30
Д.2. Класс «MainWindowController»....30
Д.3. Файл разметки «MainWindow.fxml»....34

1. Задание

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.

Блок практических заданий 1.1-1.4 призван сформировать у студента понимание особенностей хранения, умение настраивать и поддерживать данные.

2. Ход работы

2.1. Выбор технологий

В качестве технологии для создания графического приложения курсовой работы, был выбран изученный в ходе лекция фреймворк JavaFX. Разработка велась в IntelliJ IDEA с использованием Scene Builder для удобства настройки FXML-разметки.

2.2. Диаграммы классов

2.2.1. Лабораторная работа 1

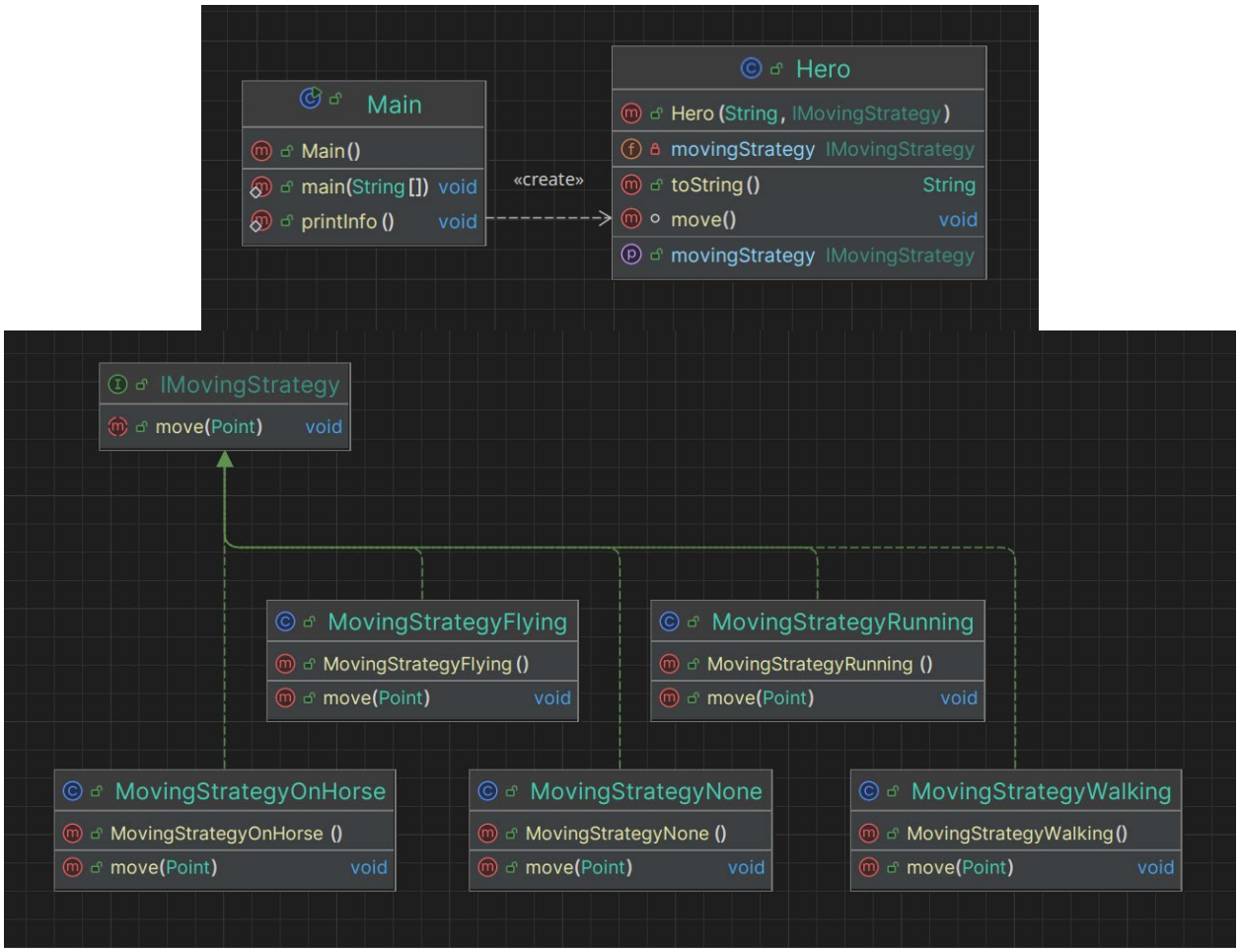

Рисунок 1 – Диграмма классов 1 лабораторной работы 1

2.2.2. Лабораторная работа 2

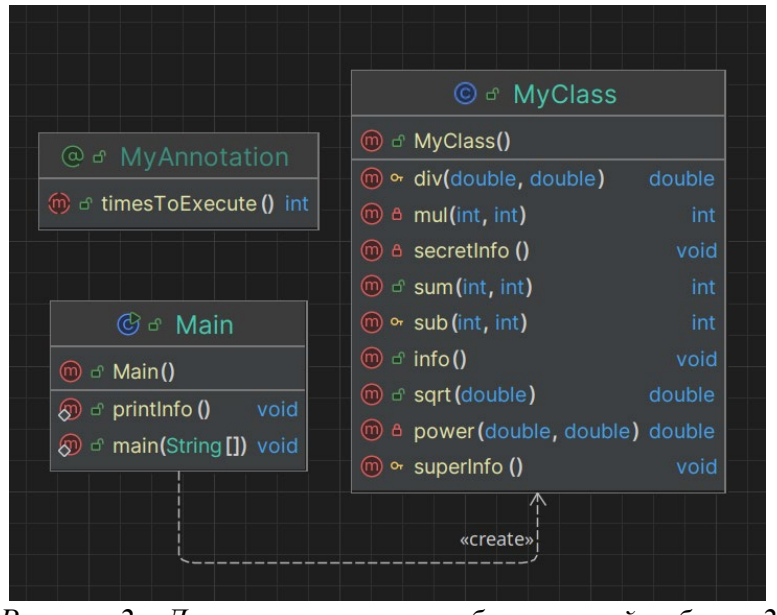

Рисунок 2 – Диаграмма классов лабораторной работы 2

2.2.3. Лабораторная работа 3

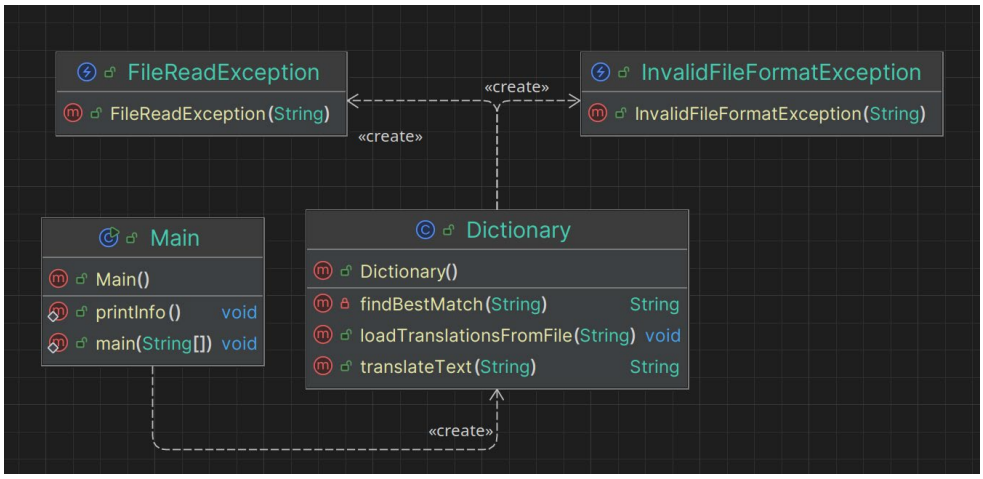

Рисунок 3 – Диаграмма классов лабораторной работы 3

2.2.4. Лабораторная работа 4

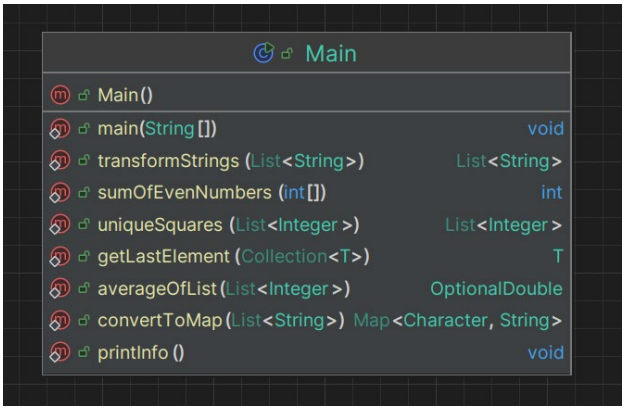

Рисунок 4 – Диаграмма классов лабораторной работы 4

2.2.5. Курсовая работа

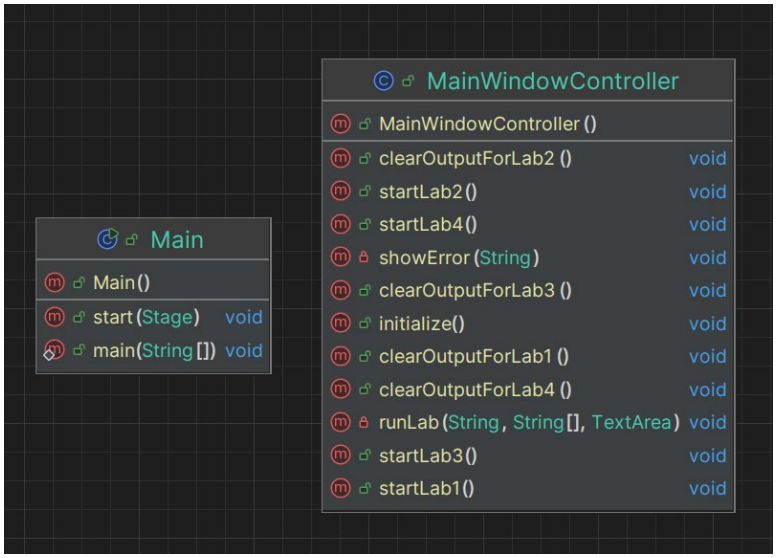

Рисунок 5 – Диаграмма классов курсовой работы

2.3. Разработка приложения

Для каждой из лабораторных работ отведена отдельная вкладка внутри приложения. На каждой из вкладок приведено поле вывода информации о работе, кнопка для запуска выполнения программы, и поле вывода информации о выполнении программы (*рисунок 6*).

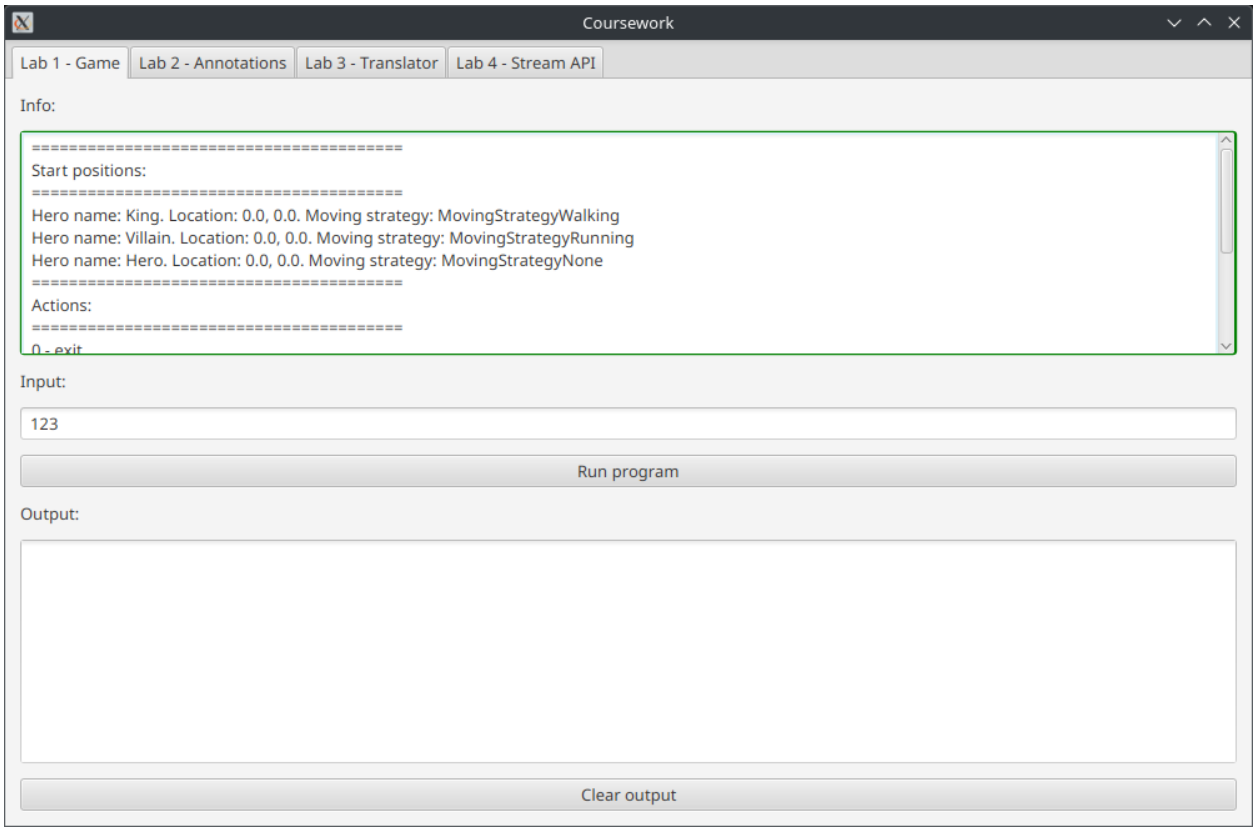

Рисунок 6 – Вид программы

Вызов лабораторных работ происходит посредством вызова их скомпилированного файла. Данная логика описана в методе «runLab»:

```
private void runLab(String className, String[] args, TextArea outputArea) {
    // Prepare the command to execute the class in a separate JVM
    List<String> command = new ArrayList<>();
    command.add("/home/nikolai/.jdks/openjdk-23/bin/java");
    command.add("-classpath");
    // Set the correct classpath where lab_x.Main classes are located
    command.add("./target/classes");
    // Class to be run
    command.add(className);
    Collections.addAll(command, args);

    // Start the process using ProcessBuilder
    ProcessBuilder processBuilder = new ProcessBuilder(command);
    Process process;
    try {
        process = processBuilder.start();
```

```
    } catch (IOException e) {
        showError("Error while starting lab process: " + e.getMessage());
        return;
    }

    // We will print both standard and error output in the same text
    StringBuilder outputTextBuilder = new StringBuilder();

    BufferedReader outReader = new BufferedReader(new
InputStreamReader(process.getInputStream()));
    BufferedReader errReader = new BufferedReader(new
InputStreamReader(process.getErrorStream()));

    try {
        String outLine;
        String errLine;
        do {
            // Capture the process's standard output
            outLine = outReader.readLine();
            if (outLine != null) {
                outputTextBuilder.append(outLine).append("\n");
            }
            // Capture any errors from the process's standard error
            errLine = errReader.readLine();
            if (errLine != null) {
                outputTextBuilder.append(errLine).append("\n");
            }
        } while (outLine != null || errLine != null);
    } catch (IOException e) {
        showError("Error while reading lab process standard output: " +
e.getMessage());
        return;
    }
    // Wait for the process to exit
    int exitCode;
    try {
        exitCode = process.waitFor();
    } catch (InterruptedException e) {
        showError("Error while waiting for lab process to exit: " +
e.getMessage());
        return;
    }
    // If exit code is not zero, change border color of the output box
    if (exitCode == 0) {
        outputArea.setStyle("-fx-background-color: green;");
    } else {
        outputArea.setStyle("-fx-background-color: red;");
    }

    outputArea.setText(outputTextBuilder.toString());
}
```

Полный код классов курсовой работы приведён в *приложении Д*.

После вызова лабораторной работы, её стандартный вывод и вывод ошибок перехватываются и выводятся в соответствующее поле вывода. Если произошла ошибка, поле вывода будет подсвечено красным цветом (*рисунок 7*). Если же выполнение прошло успешно – подсветка будет зелёной.

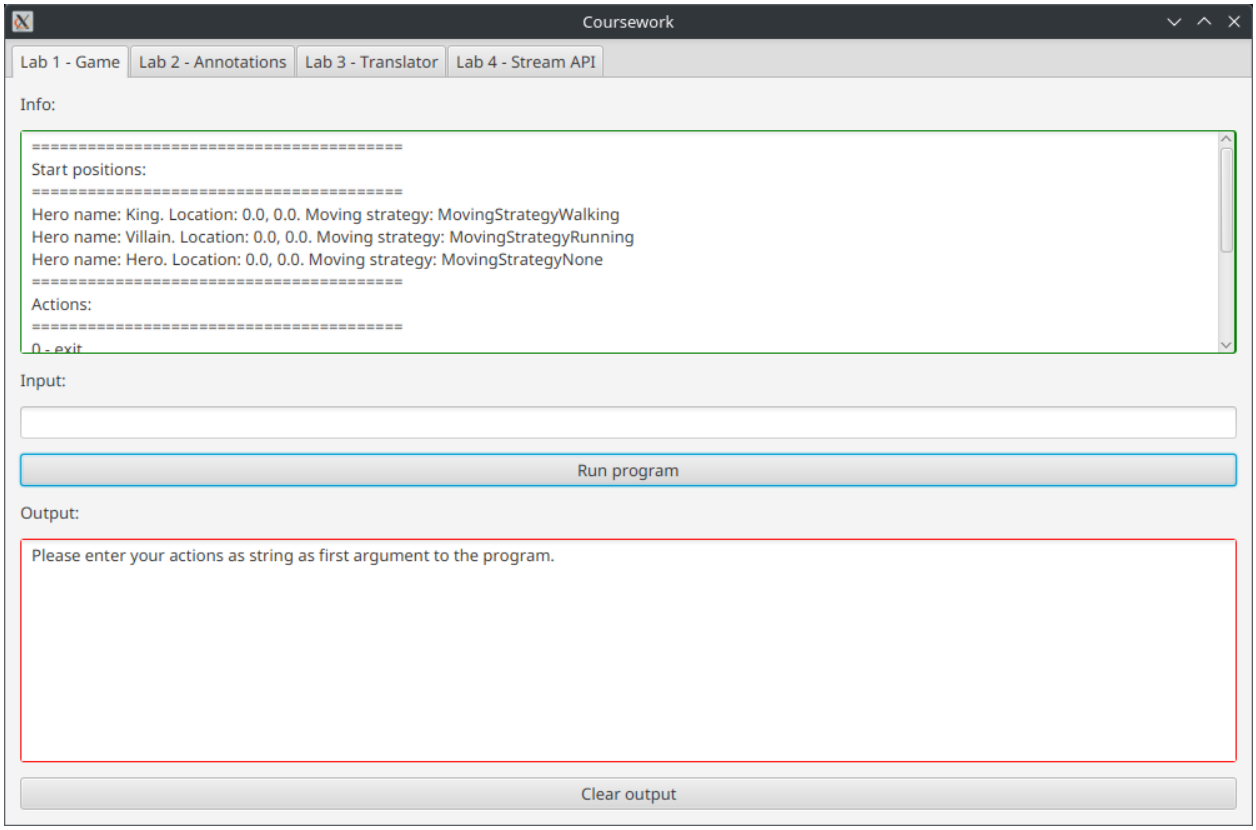

Рисунок 7 – Подсветка поля вывода в случае возникновения ошибки внутри лабораторной работы

Для первой лабораторной работы было также добавлено поле для ввода действий. После нажатия на кнопку запуска лабораторной, указанные действия будут переданы в программа первым аргументом, где он будет разбит на нужные действия:

```
public static void main(String[] args) {
    if (Arrays.stream(args).toList().contains("--info")) {
        printInfo();
        System.exit(0);
    }

    if (args.length == 0) {
        System.out.println("Please enter your actions as string as first argument
to the program.");
        System.exit(1);
    }

    String moves = args[0];
```

```
    if (moves.isEmpty()) {
        System.out.println("Please enter your actions as string as first argument
to the program.");
        System.exit(1);
    }

    System.out.println("========================================");

    for_cycle:
    for (int i = 0; i < moves.length(); i++) {
        System.out.print("Your turn. Choose action: ");
        char command = moves.charAt(i);
        System.out.println(command);

        switch (command) {
            // ...
        }

        // ...
```

Пример вывода для лабораторной работы 1 приведён на *рисунке 8*.

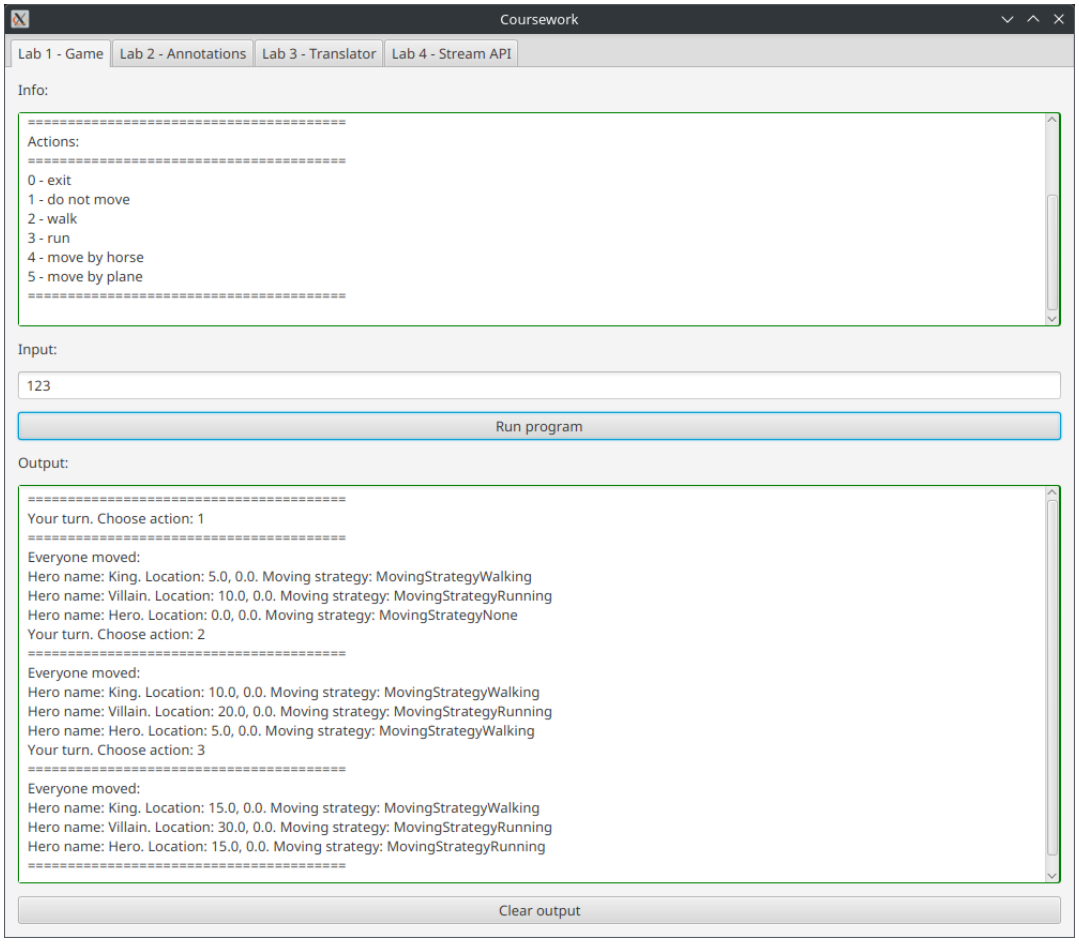

Рисунок 8 – Вывод лабораторной работы 1

Так как в лабораторной работе 3 количество вызовов для методов указываются в аннотации на этапе компиляции, дополнительных полей ввода

тут добавлено не было. Пример вывода для лабораторной работы 2 приведён на *рисунке 9*.

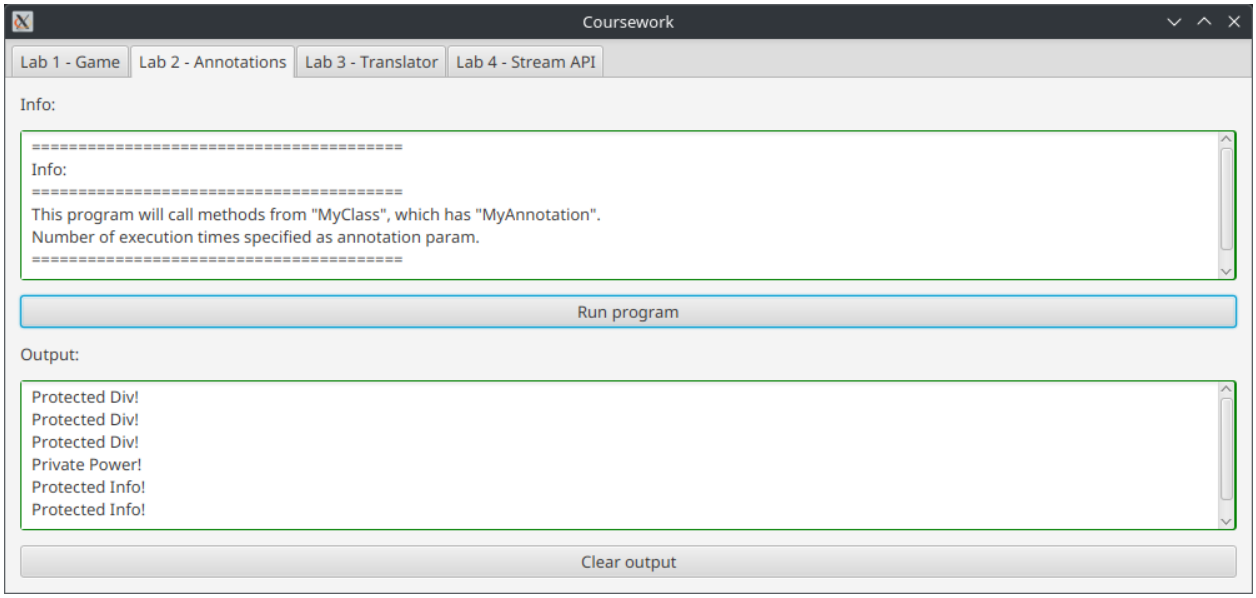

Рисунок 9 – Вывод лабораторной работы 2

Для лабораторной работы 3 было добавлено поле для ввода путей к файлам словарей, а также поле для ввода текста для перевода. Пример вывода для лабораторной работы 3 приведён на *рисунке 10*.

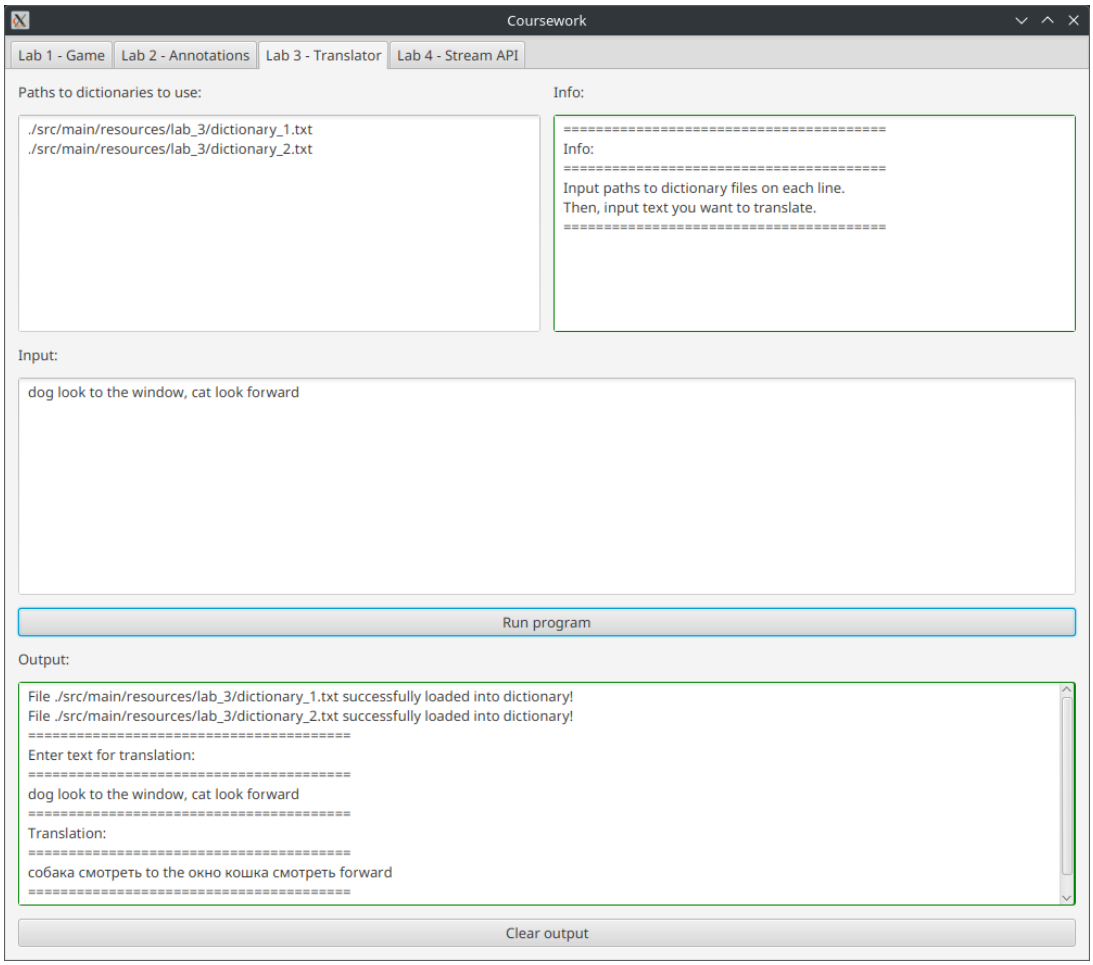

Рисунок 10 – Вывод лабораторной работы 3

Для последней лабораторной работы были добавлены поля для ввода соответствующих списков с данными. Введённые списки передаются в лабораторную работу строчками, где обрабатываются через операцию «split»:

```
List<String> strings = Arrays.stream(args[1].split("\\s*,\\s*")).toList();
```

Для целочисленных список добавлена обработка исключения при преобразовании чисел:

```
List<Integer> numbers;
try {
    numbers =
Arrays.stream(args[0].split("\\s*,\\s*")).map(Integer::parseInt).toList();
} catch (NumberFormatException e) {
    System.out.println("Failed to convert string \"" + args[0] + "\" to list of
integers!");
    System.exit(1);
    return;
}
```

Пример вывода для лабораторной работы 4 приведён на *рисунке 11*.

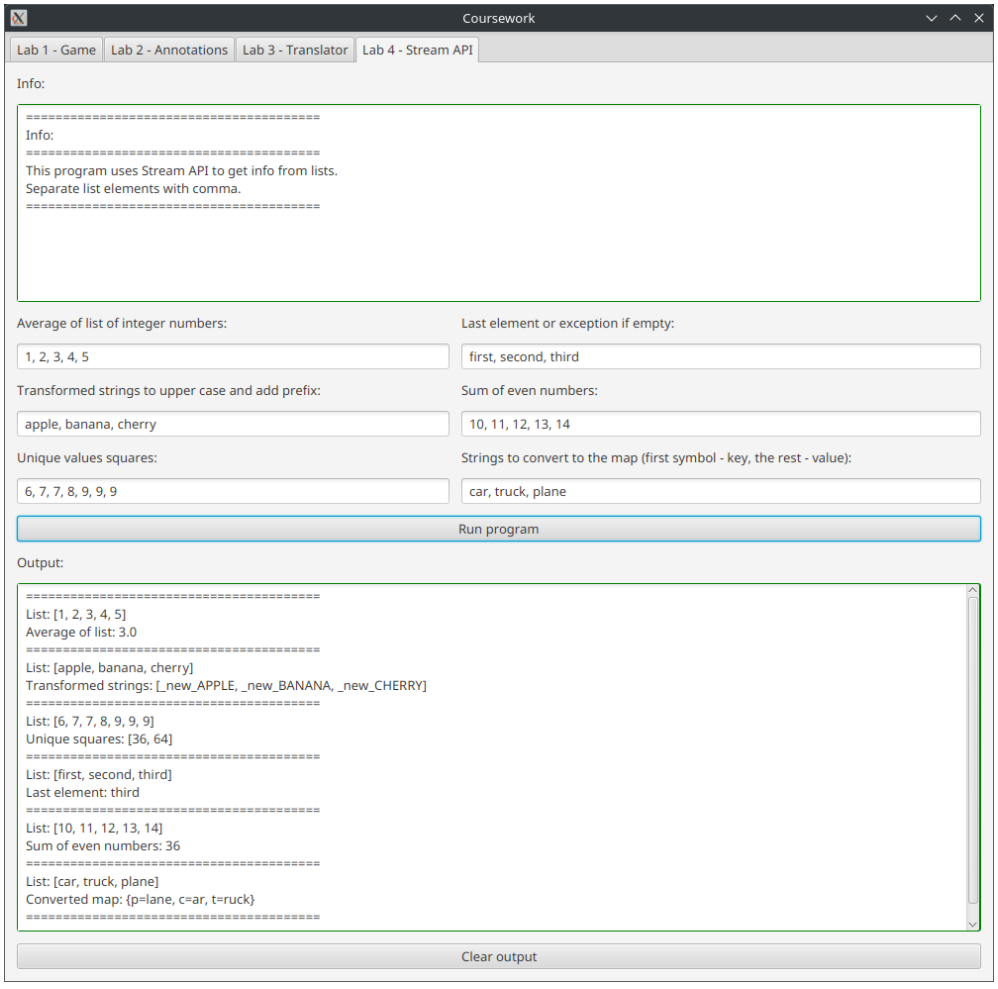

Рисунок 11 – Вывод лабораторной работы 4

Для завершения графического приложения достаточно закрыть его окно.

3. Вывод

В ходе курсовой работы было разработано приложение с графическим интерфейсом для лабораторных раб 1-4, которое позволяет вызывать каждую из них с указанными параметрами, которые можно указать через поля ввода. Также, в графическом интерфейсе отображается вывод вызываемых программ, защищённый от изменений.

Сами работы были модифицированы для получения входных данных через параметры главного метода.

Для лабораторной работы 3 предусмотрен выбор файлов словаря и текста для перевода, возможность ручного ввода текста.

Для лабораторной работы 4 предусмотрен ввод входных данных для методов.

По итогу выполнения курсовой работы было сформировано понимание особенностей хранения, умение настраивать и поддерживать данные, а также были закреплены знания по изученному фреймворку JavaFX для создания графических приложений на языке программирования Java.

Приложение А.

Исходный код лабораторной работы 1

А.1. Класс «Main»

```
package lab_1;

import lab_1.moving_strategies.*;

import java.util.Arrays;

public class Main {
    private static final Hero king = new Hero("King", new
MovingStrategyWalking());
    private static final Hero villain = new Hero("Villain", new
MovingStrategyRunning());
    private static final Hero hero = new Hero("Hero", new MovingStrategyNone());

    public static void printInfo () {
        System.out.println("========================================");
        System.out.println("Start positions:");
        System.out.println("========================================");
        System.out.println(king);
        System.out.println(villain);
        System.out.println(hero);

        System.out.println("========================================");
        System.out.println("Actions:");
        System.out.println("========================================");
        System.out.println("0 - exit");
        System.out.println("1 - do not move");
        System.out.println("2 - walk");
        System.out.println("3 - run");
        System.out.println("4 - move by horse");
        System.out.println("5 - move by plane");
        System.out.println("========================================");
    }

    public static void main(String[] args) {
        if (Arrays.stream(args).toList().contains("--info")) {
            printInfo();
            System.exit(0);
        }

        if (args.length == 0) {
            System.out.println("Please enter your actions as string as first
argument to the program.");
            System.exit(1);
        }
```

```
        String moves = args[0];

        if (moves.isEmpty()) {
            System.out.println("Please enter your actions as string as first
argument to the program.");
            System.exit(1);
        }

        System.out.println("========================================");

        for_cycle:
        for (int i = 0; i < moves.length(); i++) {
            System.out.print("Your turn. Choose action: ");
            char command = moves.charAt(i);
            System.out.println(command);

            switch (command) {
                case '0':
                    break for_cycle;
                case '1':
                    hero.setMovingStrategy(new MovingStrategyNone());
                    break;
                case '2':
                    hero.setMovingStrategy(new MovingStrategyWalking());
                    break;
                case '3':
                    hero.setMovingStrategy(new MovingStrategyRunning());
                    break;
                case '4':
                    hero.setMovingStrategy(new MovingStrategyOnHorse());
                    break;
                case '5':
                    hero.setMovingStrategy(new MovingStrategyFlying());
                    break;
                default:
                    System.out.println("Unknown action: " + command + ".");
                    System.exit(1);
            }

            System.out.println("========================================");
            System.out.println("Everyone moved:");

            king.move();
            villain.move();
            hero.move();

            System.out.println(king);
            System.out.println(villain);
            System.out.println(hero);
        }
```

```
        System.out.println("=======================================");
    }
}
```

A.2. Класс «Hero»

```
package lab_1;

import lab_1.moving_strategies.IMovingStrategy;

import java.awt.*;

/**
 * Герой
 */
public class Hero {
    /**
     * Имя героя
     */
    private final String name;

    /**
     * Расположение героя
     */
    private final Point location;

    /**
     * Стратегия передвижения героя
     */
    private IMovingStrategy movingStrategy;

    /**
     * Создаёт героя
     * @param name Имя героя
     * @param movingStrategy Стратегия передвижения героя
     */
    public Hero(String name, IMovingStrategy movingStrategy) {
        this.name = name;
        this.location = new Point(0, 0);
        this.movingStrategy = movingStrategy;
    }

    @Override
    public String toString() {
        return "Hero name: " + name + ". Location: " + location.getX() + ", " +
location.getY() + ". Moving strategy: " +
movingStrategy.getClass().getSimpleName();
    }
```

```
    /**
     * Возвращает текущую стратегию передвижения героя
     * @return Стратегия передвижения героя
     */
    public IMovingStrategy getMovingStrategy() {
        return movingStrategy;
    }

    /**
     * Меняет стратегию передвижения героя
     * @param movingStrategy Стратегия передвижения
     */
    public void setMovingStrategy(IMovingStrategy movingStrategy) {
        this.movingStrategy = movingStrategy;
    }

    /**
     * Двигает героя
     */
    void move() {
        this.movingStrategy.move(this.location);
    }
}
```

A.3. Интерфейс «IMovingStrategy»

```
package lab_1.moving_strategies;

import java.awt.*;

/**
 * Стратегия передвижения
 */
public interface IMovingStrategy {
    void move(Point location);
}
```

A.4. Класс «IMovingStrategyNone»

```
package lab_1.moving_strategies;

import java.awt.*;

/**
 * Стоять на месте
 */
public class MovingStrategyNone implements IMovingStrategy {
    @Override
    public void move(Point location) {
        // Nothing
```

```
    }
}
```

A.5. Класс «IMovingStrategyWalking»

```
package lab_1.moving_strategies;

import java.awt.*;

/**
 * Передвижение пешком
 */
public class MovingStrategyWalking implements IMovingStrategy {
    @Override
    public void move(Point location) {
        location.setLocation(location.getX() + 5, location.getY());
    }
}
```

A.6. Класс «IMovingStrategyRunning»

```
package lab_1.moving_strategies;

import java.awt.*;

/**
 * Передвижение бегом
 */
public class MovingStrategyRunning implements IMovingStrategy {
    @Override
    public void move(Point location) {
        location.setLocation(location.getX() + 10, location.getY());
    }
}
```

A.7. Класс «IMovingStrategyOnHorse»

```
package lab_1.moving_strategies;

import java.awt.*;
/**
 * Передвижение на лошади
 */
public class MovingStrategyOnHorse implements IMovingStrategy {
    @Override
    public void move(Point location) {
        location.setLocation(location.getX() + 30, location.getY());
    }
}
```

A.8. Класс «IMovingStrategyFlying»

```java
package lab_1.moving_strategies;

import java.awt.*;

/**
 * Передвижение на самолёте
 */
public class MovingStrategyFlying implements IMovingStrategy {
    @Override
    public void move(Point location) {
        location.setLocation(location.getX() + 1000, location.getY());
    }
}
```

Приложение Б.

Исходный код лабораторной работы 2

Б.1. Класс «Main»

```
package lab_2;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.util.Arrays;

public class Main {
    public static void printInfo () {
        System.out.println("========================================");
        System.out.println("Info:");
        System.out.println("========================================");
        System.out.println("This program will call methods from \"MyClass\",
which has \"MyAnnotation\".");
        System.out.println("Number of execution times specified as annotation
param.");
        System.out.println("========================================");
    }

    public static void main(String[] args) {
        if (Arrays.stream(args).toList().contains("--info")) {
            printInfo();
            System.exit(0);
        }

        // Create object instance
        MyClass obj = new MyClass();

        // Get all object methods (private included)
        Method[] methods = MyClass.class.getDeclaredMethods();

        for (Method method : methods) {
            // If method has not our annotation - skip it
            if (!method.isAnnotationPresent(MyAnnotation.class)) {
                continue;
            }

            // Skip, if method not protected or private
            if (!Modifier.isProtected(method.getModifiers()) &&
!Modifier.isPrivate(method.getModifiers())) {
                continue;
            }

            // Make method callable
```

```
            method.setAccessible(true);

            // Get annotation parameter
            int timesToExecute =
method.getAnnotation(MyAnnotation.class).timesToExecute();

            // Call the method specified number of times
            for (int executionId = 0; executionId < timesToExecute;
executionId++) {
                Class<?>[] parameterTypes = method.getParameterTypes();

                // Generate parameters based on their type
                Object[] parameters = new Object[parameterTypes.length];
                for (int parameterId = 0; parameterId < parameterTypes.length;
parameterId++) {
                    Class<?> parameterType = parameterTypes[parameterId];
                    if (parameterType == int.class) {
                        parameters[parameterId] = 5;
                    } else if (parameterType == double.class) {
                        parameters[parameterId] = 7.4;
                    } else if (parameterType == float.class) {
                        parameters[parameterId] = 8.3f;
                    } else if (parameterType == boolean.class) {
                        parameters[parameterId] = true;
                    } else if (parameterType == String.class) {
                        parameters[parameterId] = "Lalala";
                    } else {
                        throw new RuntimeException("Parameter type " +
parameterType.getName() + " not supported!");
                    }
                }

                // Method call
                try {
                    method.invoke(obj, parameters);
                } catch (IllegalAccessException | InvocationTargetException e) {
                    throw new RuntimeException(e);
                }
            }
        }
    }
}
```

Б.2. Класс «MyAnnotation»

```
package lab_2;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
```

```java
import java.lang.annotation.Target;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface MyAnnotation {
    int timesToExecute() default 0;
}
```

Б.3. Класс «MyClass»

```java
package lab_2;

public class MyClass {
    public int sum(int a, int b) {
        System.out.println("Public Sum!");
        return a + b;
    }

    @MyAnnotation(timesToExecute = 2)
    public double sqrt(double a) {
        System.out.println("Public Sqrt!");
        return Math.sqrt(a);
    }

    public void info() {
        System.out.println("Hello, World!");
    }

    protected int sub(int a, int b) {
        System.out.println("Protected Sub!");
        return a - b;
    }

    @MyAnnotation(timesToExecute = 3)
    protected double div(double a, double b) {
        System.out.println("Protected Div!");
        return a / b;
    }

    @MyAnnotation(timesToExecute = 2)
    protected void superInfo() {
        System.out.println("Protected Info!");
    }

    @MyAnnotation(timesToExecute = 0)
    private int mul(int a, int b) {
        System.out.println("Private Mul!");
        return a * b;
    }
```

```
    @MyAnnotation(timesToExecute = 1)
    private double power(double a, double b) {
        System.out.println("Private Power!");
        return Math.pow(a, b);
    }

    private void secretInfo() {
        System.out.println("Private Info!");
    }
}
```

Приложение В.

Исходный код лабораторной работы 3

В.1. Класс «Main»

```
package lab_3;

import lab_3.exceptions.FileReadException;
import lab_3.exceptions.InvalidFileFormatException;

import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void printInfo () {
        System.out.println("========================================");
        System.out.println("Info:");
        System.out.println("========================================");
        System.out.println("Input paths to dictionary files on each line.");
        System.out.println("Then, input text you want to translate.");
        System.out.println("========================================");
    }

    public static void main(String[] args) {
        if (Arrays.stream(args).toList().contains("--info")) {
            printInfo();
            System.exit(0);
        }

        if (args.length == 0) {
            System.out.println("Please enter your text for translation as first
argument to the program.");
            System.exit(1);
        } else if (args.length == 1) {
            System.out.println("Please specify dictionaries paths as arguments to
the program after your text.");
            System.exit(1);
        }

        Dictionary translator = new Dictionary();
        for (int dictionary_id = 1; dictionary_id < args.length; dictionary_id++)
{
            String dictionaryFilePath = args[dictionary_id];
            // Load file
            try {
                translator.loadTranslationsFromFile(dictionaryFilePath);
            } catch (InvalidFileFormatException | FileReadException e) {
                System.out.println(e.getMessage());
                System.exit(1);
```

```
            }
            System.out.println("File " + dictionaryFilePath + " successfully
loaded into dictionary!");
        }

        // Get input from user
        System.out.println("========================================");
        System.out.println("Enter text for translation:");
        System.out.println("========================================");

        String input = args[0];
        System.out.println(input);

        // Translate and print it
        System.out.println("========================================");
        System.out.println("Translation:");
        System.out.println("========================================");
        System.out.println(translator.translateText(input));
        System.out.println("========================================");
    }
}
```

B.2. Класс «Dictionary»

```
package lab_3;

import lab_3.exceptions.FileReadException;
import lab_3.exceptions.InvalidFileFormatException;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.Paths;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Map;

// Class to translate text with specified dictionary
public class Dictionary {
    // Dictionary itself, containing words/phrases to translations
    private final Map<String, String> dictionary = new HashMap<>();

    public void loadTranslationsFromFile(String filePath) throws
InvalidFileFormatException, FileReadException {
        // Try reading the file from the provided file path
        try (BufferedReader reader = new BufferedReader(new
FileReader(Paths.get(filePath).toFile()))) {
            String line;
            while ((line = reader.readLine()) != null) {
                // Split line into two words
```

```
                String[] parts = line.split("\\|");
                if (parts.length != 2) {
                    throw new InvalidFileFormatException("Invalid file format: "
+ line);
                }
                // Add entry to the dictionary (and ignore case)
                dictionary.put(parts[0].trim().toLowerCase(),
parts[1].trim().toLowerCase());
            }
        } catch (IOException e) {
            throw new FileReadException("Error reading file: " + e.getMessage());
        }
    }

    // Translate input text based on dictionary
    public String translateText(String input) {
        String[] words = input.split("\\s+");

        StringBuilder result = new StringBuilder();

        for (int wordId = 0; wordId < words.length; wordId++) {
            // Ignore case
            String word = words[wordId].toLowerCase();

            String bestMatch = findBestMatch(word);
            if (bestMatch != null) {
                // If match found - add it to the result
                result.append(dictionary.get(bestMatch));
            } else {
                // If no match found - just add the word as it was
                result.append(words[wordId]);
            }

            // Add space between words
            if (wordId < words.length - 1) {
                result.append(" ");
            }
        }

        return result.toString();
    }

    // Find the best matching translation in the dictionary
    private String findBestMatch(String input) {
        // Sort and use match with max length
        return dictionary.keySet().stream()
                .filter(input::startsWith)
                .max(Comparator.comparingInt(String::length))
                .orElse(null);
    }
}
```

В.3. Класс «FileReadException»

```
package lab_3.exceptions;

// Custom exception for file read errors
public class FileReadException extends Exception {
    public FileReadException(String message) {
        super(message);
    }
}
```

В.4. Класс «InvalidFileFormatException»

```
package lab_3.exceptions;

// Custom exception for invalid file format
public class InvalidFileFormatException extends Exception {
    public InvalidFileFormatException(String message) {
        super(message);
    }
}
```

Приложение Г.

Исходный код лабораторной работы 4

Г.1. Класс «Main»

```
package lab_4;

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

public class Main {
    public static void printInfo () {
        System.out.println("========================================");
        System.out.println("Info:");
        System.out.println("========================================");
        System.out.println("This program uses Stream API to get info from
lists.");
        System.out.println("Separate list elements with comma.");
        System.out.println("========================================");
    }

    public static void main(String[] args) {
        if (Arrays.stream(args).toList().contains("--info")) {
            printInfo();
            System.exit(0);
        }

        if (args.length < 6) {
            System.out.println("You must specify all the arrays");
            System.exit(1);
        }


        // Test averageOfList method
        System.out.println("========================================");
        List<Integer> numbers;
        try {
            numbers =
Arrays.stream(args[0].split("\\s*,\\s*")).map(Integer::parseInt).toList();
        } catch (NumberFormatException e) {
            System.out.println("Failed to convert string \"" + args[0] + "\" to
list of integers!");
            System.exit(1);
            return;
        }
        System.out.println("List: " + numbers);
        System.out.println("Average of list: " +
averageOfList(numbers).orElse(0.0));
```

```
        // Test transformStrings method
        System.out.println("=======================================");
        List<String> strings =
Arrays.stream(args[1].split("\\s*,\\s*")).toList();
        System.out.println("List: " + strings);
        System.out.println("Transformed strings: " + transformStrings(strings));

        // Test uniqueSquares method
        System.out.println("=======================================");
        List<Integer> duplicateNumbers;
        try {
            duplicateNumbers =
Arrays.stream(args[2].split("\\s*,\\s*")).map(Integer::parseInt).toList();
        } catch (NumberFormatException e) {
            System.out.println("Failed to convert string \"" + args[2] + "\" to
list of integers!");
            System.exit(1);
            return;
        }
        System.out.println("List: " + duplicateNumbers);
        System.out.println("Unique squares: " + uniqueSquares(duplicateNumbers));

        // Test getLastElement method
        System.out.println("=======================================");
        List<String> moreStrings =
Arrays.stream(args[3].split("\\s*,\\s*")).toList();
        System.out.println("List: " + moreStrings);
        System.out.println("Last element: " + getLastElement(moreStrings));

        // Test sumOfEvenNumbers method
        System.out.println("=======================================");
        int[] numArray;
        try {
            numArray =
Arrays.stream(args[4].split("\\s*,\\s*")).mapToInt(Integer::parseInt).toArray();
        } catch (NumberFormatException e) {
            System.out.println("Failed to convert string \"" + args[4] + "\" to
array of integers!");
            System.exit(1);
            return;
        }
        System.out.println("List: " + Arrays.toString(numArray));
        System.out.println("Sum of even numbers: " + sumOfEvenNumbers(numArray));

        // Test convertToMap method
        System.out.println("=======================================");
        List<String> stringList =
Arrays.stream(args[5].split("\\s*,\\s*")).toList();
        System.out.println("List: " + stringList);
        System.out.println("Converted map: " + convertToMap(stringList));
```

```
        System.out.println("========================================");
    }

    // Method to return the average of a list of integers
    public static OptionalDouble averageOfList(List<Integer> numbers) {
        return numbers.stream()
                .mapToInt(Integer::intValue)
                .average();
    }

    // Method to convert all strings in the list to uppercase and add "_new_"
prefix
    public static List<String> transformStrings(List<String> strings) {
        return strings.stream()
                .map(s -> "_new_" + s.toUpperCase())
                .collect(Collectors.toList());
    }

    // Method to return a list of squares of elements that appear only once in
the list
    public static List<Integer> uniqueSquares(List<Integer> numbers) {
        return numbers.stream()
                .filter(n -> Collections.frequency(numbers, n) == 1)
                .map(n -> n * n)
                .collect(Collectors.toList());
    }

    // Method to return the last element of a collection or throw an exception if
empty
    public static <T> T getLastElement(Collection<T> collection) {
        return collection.stream()
                .reduce((first, second) -> second)
                .orElseThrow(() -> new NoSuchElementException("Collection is
empty"));
    }
    // Method to return the sum of all even numbers in the array or 0 if none
exist
    public static int sumOfEvenNumbers(int[] numbers) {
        return Arrays.stream(numbers)
                .filter(n -> n % 2 == 0)
                .sum();
    }
    // Method to convert a list of strings to a Map where the first character is
the key and the rest is the value
    public static Map<Character, String> convertToMap(List<String> strings) {
        return strings.stream()
                .filter(s -> s.length() > 0)
                .collect(Collectors.toMap(s -> s.charAt(0), s ->
s.substring(1)));
    }
}
```

Приложение Д.

Исходный код курсовой работы

Д.1. Класс «Main»

```
package coursework;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.File;
import java.io.IOException;
import java.net.URL;

public class Main extends Application {
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage stage) throws IOException {
        // Устанавливаем размеры окна
        stage.setWidth(1000);
        stage.setHeight(800);

        // Устанавливаем сцену с FXML-разметкой
        URL url = new
File("./src/main/resources/coursework/MainWindow.fxml").toURI().toURL();
        Parent root = FXMLLoader.load(url);
        Scene scene = new Scene(root, stage.getWidth(), stage.getHeight());
        stage.setScene(scene);

        // Устанавливаем название окна
        stage.setTitle("Coursework");

        // Показываем окно
        stage.show();
    }
}
```

Д.2. Класс «MainWindowController»

```
package coursework;

import javafx.fxml.FXML;
import javafx.scene.control.Alert;
```

```
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;

import java.io.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class MainWindowController {
    @FXML
    private TextArea lab1Info, lab2Info, lab3Info, lab4Info, lab1Output,
lab2Output, lab3Output, lab4Output, lab3Input, lab3DictionariesList;
    @FXML
    private TextField lab1Input, lab4InputAverageOfList,
lab4InputTransformedStrings, lab4InputUniqueSquares, lab4InputLastElement,
lab4InputSumOfEvenNumbers, lab4InputConvertedMap;

    @FXML
    public void initialize() {
        // Set lab information by calling it with "--info" argument
        runLab("lab_1.Main", new String[]{"--info"}, lab1Info);
        runLab("lab_2.Main", new String[]{"--info"}, lab2Info);
        runLab("lab_3.Main", new String[]{"--info"}, lab3Info);
        runLab("lab_4.Main", new String[]{"--info"}, lab4Info);
    }

    public void startLab1() {
        runLab("lab_1.Main", new String[]{lab1Input.getText()}, lab1Output);
    }

    public void startLab2() {
        runLab("lab_2.Main", new String[]{}, lab2Output);
    }

    public void startLab3() {
        ArrayList<String> options = new ArrayList<>();
        options.add(lab3Input.getText());

        // Split text into list and ignore empty lines too
        options.addAll(Arrays.stream(lab3DictionariesList.getText().split("\n")).
filter(s -> !s.trim().isEmpty()).toList());

        runLab("lab_3.Main", options.toArray(new String[0]), lab3Output);
    }

    public void startLab4() {
        runLab("lab_4.Main", new String[]{
                // We pass all values as string and then will convert them to
arrays inside lab 4
                lab4InputAverageOfList.getText(),
```

```
                lab4InputTransformedStrings.getText(),
                lab4InputUniqueSquares.getText(),
                lab4InputLastElement.getText(),
                lab4InputSumOfEvenNumbers.getText(),
                lab4InputConvertedMap.getText(),
        }, lab4Output);
    }

    public void clearOutputForLab1() {
        lab1Output.clear();
    }

    public void clearOutputForLab2() {
        lab2Output.clear();
    }

    public void clearOutputForLab3() {
        lab3Output.clear();
    }

    public void clearOutputForLab4() {
        lab4Output.clear();
    }

    private void runLab(String className, String[] args, TextArea outputArea) {
        // Prepare the command to execute the class in a separate JVM
        List<String> command = new ArrayList<>();
        command.add("/home/nikolai/.jdks/openjdk-23/bin/java");
        command.add("-classpath");
        // Set the correct classpath where lab_x.Main classes are located
        command.add("./target/classes");
        // Class to be run
        command.add(className);
        Collections.addAll(command, args);

        // Start the process using ProcessBuilder
        ProcessBuilder processBuilder = new ProcessBuilder(command);
        Process process;
        try {
            process = processBuilder.start();
        } catch (IOException e) {
            showError("Error while starting lab process: " + e.getMessage());
            return;
        }

        // We will print both standard and error output in the same text
        StringBuilder outputTextBuilder = new StringBuilder();

        BufferedReader outReader = new BufferedReader(new
InputStreamReader(process.getInputStream()));
```

```
        BufferedReader errReader = new BufferedReader(new
InputStreamReader(process.getErrorStream()));

        try {
            String outLine;
            String errLine;
            do {
                // Capture the process's standard output
                outLine = outReader.readLine();
                if (outLine != null) {
                    outputTextBuilder.append(outLine).append("\n");
                }

                // Capture any errors from the process's standard error
                errLine = errReader.readLine();
                if (errLine != null) {
                    outputTextBuilder.append(errLine).append("\n");
                }
            } while (outLine != null || errLine != null);
        } catch (IOException e) {
            showError("Error while reading lab process standard output: " +
e.getMessage());
            return;
        }

        // Wait for the process to exit
        int exitCode;
        try {
            exitCode = process.waitFor();
        } catch (InterruptedException e) {
            showError("Error while waiting for lab process to exit: " +
e.getMessage());
            return;
        }

        // If exit code is not zero, change border color of the output box
        if (exitCode == 0) {
            outputArea.setStyle("-fx-background-color: green;");
        } else {
            outputArea.setStyle("-fx-background-color: red;");
        }

        outputArea.setText(outputTextBuilder.toString());
    }

    private void showError(String errorMessage) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(null);
        alert.setContentText(errorMessage);
        alert.showAndWait();
```

```
    }
}
```

Д.3. Файл разметки «MainWindow.fxml»

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<TabPane prefHeight="800.0" prefWidth="1000.0" tabClosingPolicy="UNAVAILABLE"
xmlns="http://javafx.com/javafx/17.0.2-ea" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="coursework.MainWindowController">
    <Tab text="Lab 1 - Game">
        <VBox spacing="12.0">
            <padding>
                <Insets bottom="12" left="12" right="12" top="12" />
            </padding>
            <Label text="Info:" />
            <TextArea fx:id="lab1Info" editable="false" prefHeight="200.0" />
            <Label text="Input:" />
            <TextField fx:id="lab1Input" text="123" />
            <Button maxWidth="Infinity" onAction="#startLab1" text="Run program"
/>
            <Label text="Output:" />
            <TextArea fx:id="lab1Output" editable="false" VBox.vgrow="ALWAYS" />
            <Button maxWidth="Infinity" onAction="#clearOutputForLab1"
text="Clear output" />
        </VBox>
    </Tab>
    <Tab text="Lab 2 - Annotations">
        <VBox spacing="12.0">
            <padding>
                <Insets bottom="12" left="12" right="12" top="12" />
            </padding>
            <Label text="Info:" />
            <TextArea fx:id="lab2Info" editable="false" prefHeight="200.0" />
            <Button maxWidth="Infinity" onAction="#startLab2" text="Run program"
/>
            <Label text="Output:" />
            <TextArea fx:id="lab2Output" editable="false" VBox.vgrow="ALWAYS" />
            <Button maxWidth="Infinity" onAction="#clearOutputForLab2"
text="Clear output" />
        </VBox>
    </Tab>
    <Tab text="Lab 3 - Translator">
        <VBox spacing="12.0">
            <padding>
                <Insets bottom="12" left="12" right="12" top="12" />
```

```
            </padding>
            <HBox spacing="12.0">
                <VBox spacing="12.0" HBox.hgrow="ALWAYS">
                    <Label text="Paths to dictionaries to use:" />
                    <TextArea fx:id="lab3DictionariesList"
text="./src/main/resources/lab_3/dictionary_1.txt&#10;./src/main/resources/lab_3/
dictionary_2.txt" />
                </VBox>
                <VBox spacing="12.0" HBox.hgrow="ALWAYS">
                    <Label text="Info:" />
                    <TextArea fx:id="lab3Info" editable="false" />
                </VBox>
            </HBox>
            <Label text="Input:" />
            <TextArea fx:id="lab3Input" prefHeight="200.0" text="dog look to the
window, cat look forward" />
            <Button maxWidth="Infinity" onAction="#startLab3" text="Run program"
/>
            <Label text="Output:" />
            <TextArea fx:id="lab3Output" editable="false" VBox.vgrow="ALWAYS" />
            <Button maxWidth="Infinity" onAction="#clearOutputForLab3"
text="Clear output" />
        </VBox>
    </Tab>
    <Tab text="Lab 4 - Stream API">
        <VBox spacing="12.0">
            <padding>
                <Insets bottom="12" left="12" right="12" top="12" />
            </padding>
            <Label text="Info:" />
            <TextArea fx:id="lab4Info" editable="false" prefHeight="200.0" />
            <HBox spacing="12.0">
                <VBox spacing="12.0" HBox.hgrow="ALWAYS">
                    <Label text="Average of list of integer numbers:" />
                    <TextField fx:id="lab4InputAverageOfList" text="1, 2, 3, 4,
5" />
                    <Label text="Transformed strings to upper case and add
prefix:" />
                    <TextField fx:id="lab4InputTransformedStrings" text="apple,
banana, cherry" />
                    <Label text="Unique values squares:" />
                    <TextField fx:id="lab4InputUniqueSquares" text="6, 7, 7, 8,
9, 9, 9" />
                </VBox>
                <VBox spacing="12.0" HBox.hgrow="ALWAYS">
                    <Label text="Last element or exception if empty:" />
                    <TextField fx:id="lab4InputLastElement" text="first, second,
third" />
                    <Label text="Sum of even numbers:" />
                    <TextField fx:id="lab4InputSumOfEvenNumbers" text="10, 11,
12, 13, 14" />
```

```
                    <Label text="Strings to convert to the map (first symbol -
key, the rest - value):" />
                    <TextField fx:id="lab4InputConvertedMap" text="car, truck,
plane" />
                </VBox>
            </HBox>
            <Button maxWidth="Infinity" onAction="#startLab4" text="Run program"
/>
            <Label text="Output:" />
            <TextArea fx:id="lab4Output" editable="false" VBox.vgrow="ALWAYS" />
            <Button maxWidth="Infinity" onAction="#clearOutputForLab4"
text="Clear output" />
        </VBox>
    </Tab>
</TabPane>
```