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

КУРСОВАЯ РАБОТА

Дисциплина: Объектно-ориентированное программирование

Тема: «Разработка GUI приложения на языке программирования Java»

Выполнил студент гр. в5130904/30321 _____ В. В. Краснов
(подпись)

Руководитель _____ А. П. Маслаков
(подпись)

“12” ноября 2025 г.

Санкт-Петербург
2025

Оглавление

Постановка задачи

Цель: разработать приложение с графическим интерфейсом для заданий из лабораторных работ 1–4:

1. В компьютерной игре герой (класс Hero) может перемещаться между двумя точками (метод move) различными способами: идти пешком, ехать на лошади, лететь и т. п. Реализовать классы, позволяющие пользователю выбирать и менять в ходе выполнения программы способ перемещения героя, используя паттерн “стратегия” (strategy).
2. Написать аннотацию с целочисленным параметром. Создать класс, содержащий публичные, защищенные и приватные методы (2–3 каждого вида) с параметрами, аннотировать любые из них. Вызвать из другого класса все аннотированные защищенные и приватные методы столько раз, сколько указано в параметре аннотации. Вызывающий методы код не должен зависеть от количества и типов параметров этих методов.
3. Реализовать программу-переводчик. Использовать словарь из файла, записанного в следующем формате: слово или выражение | перевод.
   Перевод осуществляется по следующим правилам:
   - регистр букв игнорируется
   - если искомого слова нет в словаре – выводится без перевода
   - если есть несколько подходящих вариантов, выбирается вариант с максимальной длиной левой части.
   Например
   Словарь:
   look | смотреть
   look forward | ожидать
   Текст: dog look to the window, dog look forward
   Перевод: dog смотреть to the window, dog ожидать
4. С использованием только Stream API реализовать следующие методы:
   - метод, возвращающий среднее значение списка целых чисел;

- метод, приводящий все строки в списке в верхний регистр и добавляющий к ним префикс «_new_»;
- метод, возвращающий список квадратов всех встречающихся только один раз элементов списка;
- метод, принимающий на вход коллекцию и возвращающий ее последний элемент или кидающий исключение, если коллекция пуста;
- метод, принимающий на вход массив целых чисел, возвращающий сумму чётных чисел или 0, если чётных чисел нет;
- метод, преобразовывающий все строки в списке в Map, где первый символ – ключ, оставшиеся – значение.

Определены следующие задачи:

1. Создать для Java Fx разметку UI элементов, для предоставления необходимого визуального представления поставленных заданий.
2. Создать необходимые контроллеры Java Fx для взаимодействия с UI элементами.
3. Реализовать возможность выбора задания, ввод входных данных и выполнение задания.
4. Вывод заданий 1–4 так должен происходить в текстовые области, защищённых от редактирования.
5. Привести реализованную диаграмму всех используемых классов.
6. Оформить перечень выполненных работ.

Описание реализации

Приложение реализовано на языке Java с использованием фрейворка разработки пользовательского интерфейса Java Fx. При работе с Java Fx были использованы файлы разметки представления *.fxml для описания пользовательского интерфейса, и контроллеры – классы, реализующие взаимодействие с элементами управления классов с реализованными заданиями. Каждый контроллер связан с соответствующим ему представлением *.fxml.

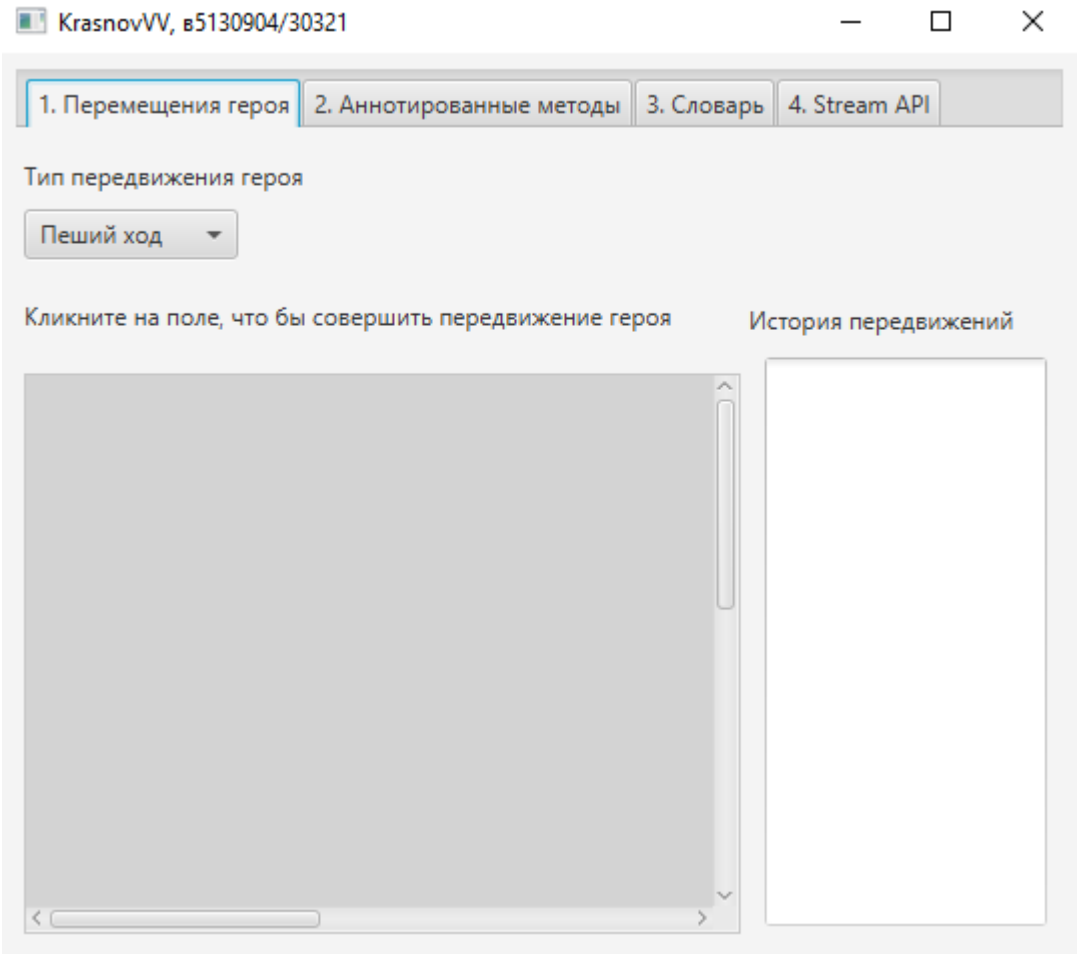
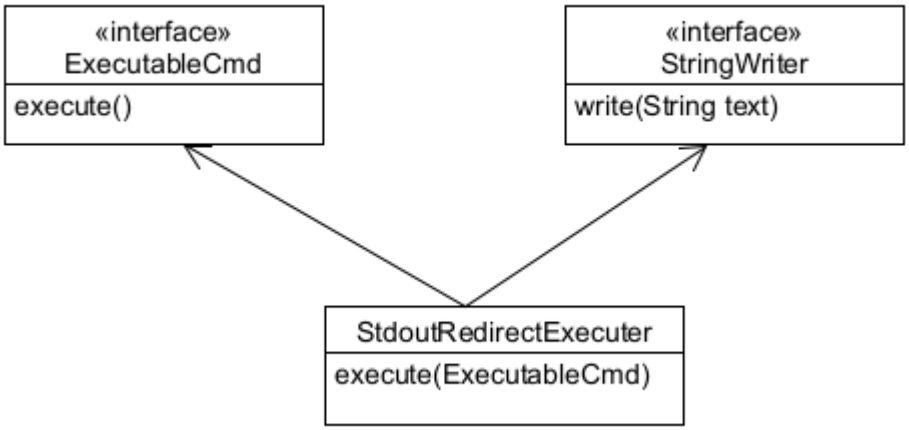


Рисунок 1. Общий вид приложения

Переключение между заданиями реализовано с помощью UI элементов Tab вкладок, каждая вкладка имеет соответствующую подпись. Для перехода к конкретному заданию необходимо выбрать соответствующую вкладку.

При реализации задания 2 и 4 использовался подход, реализующий переопределение вывода стандартного потока stdout.



Компоненту, которому нужно перенаправить вывод стандартного потока stdout, необходимо создать класс StdoutRedirectExecuter и предоставить ему при инстанцировании реализацию интерфейса StringWriter, в котором и будет определено каким образом поступить с выводом в stdout. Далее производится вызов метода execute созданного экземпляра класса StdoutRedirectExecuter с указанием команды ExecutableCmd. Во время выполнения команды вывод в поток stdout будет перенаправлен в соответствии с логикой, предоставленной в StringWriter.

Задание 1. Передвижения героя

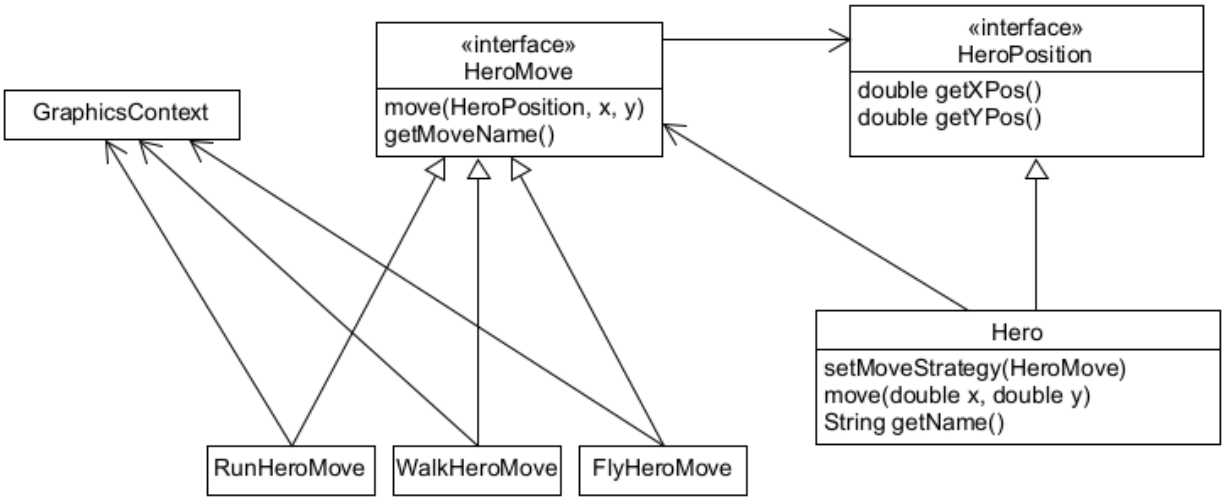


Рисунок 2. Диаграмма классов, задание 1

При реализации первого задания был применен паттерн Стратегия, позволяющий без изменения реализации класса героя изменить его поведение, передав в класс необходимую реализацию (интерфейс HeroMove).

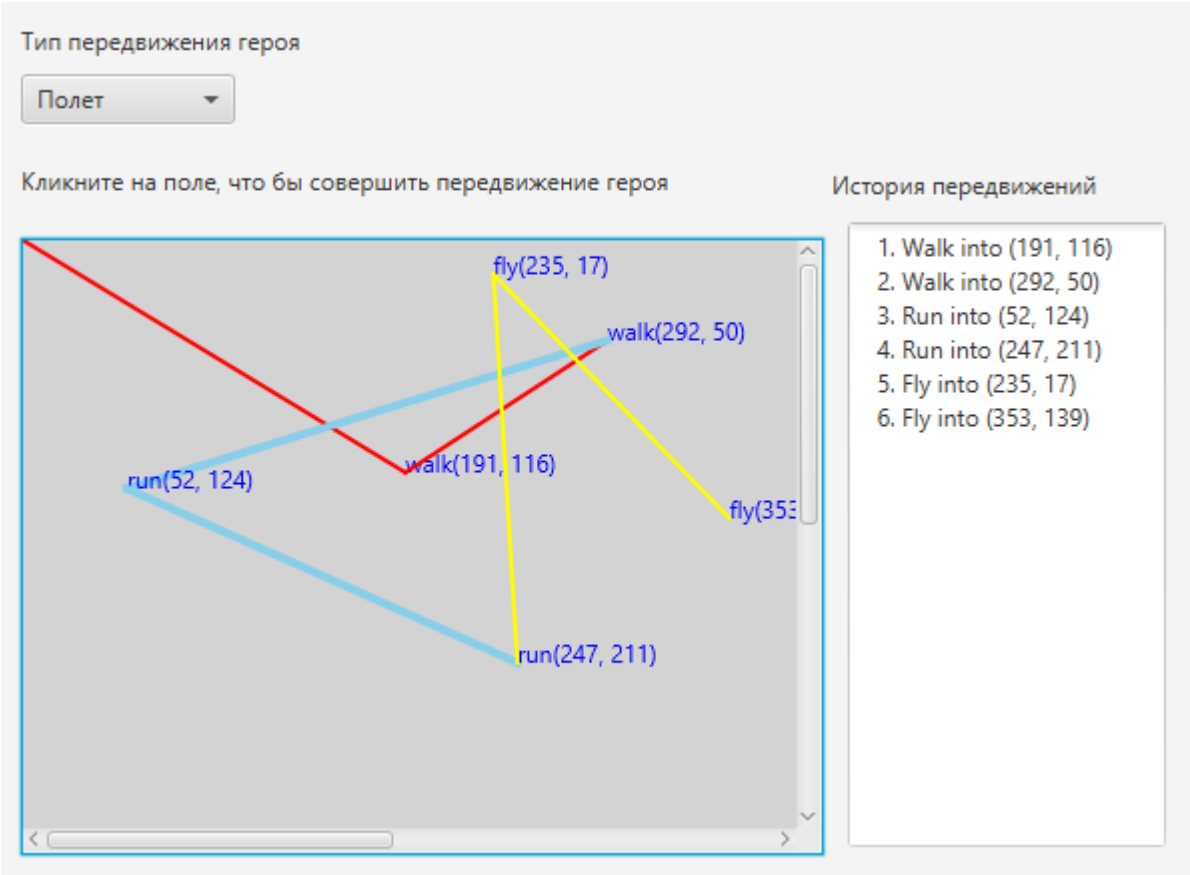


Рисунок 3. Вкладка «Задание 1»

Для осуществления перемещения героя приложение предлагает выбрать тип перемещения в выпадающем списке и мышкой кликнуть по полю в то место, куда необходимо сделать перемещение. В зависимости от типа перемещения на поле будет отрисована линия с соответствующим цветом и толщиной, а также новое перемещение будет добавлено в Историю перемещений. Начальная позиция герой – (0, 0).

Задание 2. Аннотированные методы

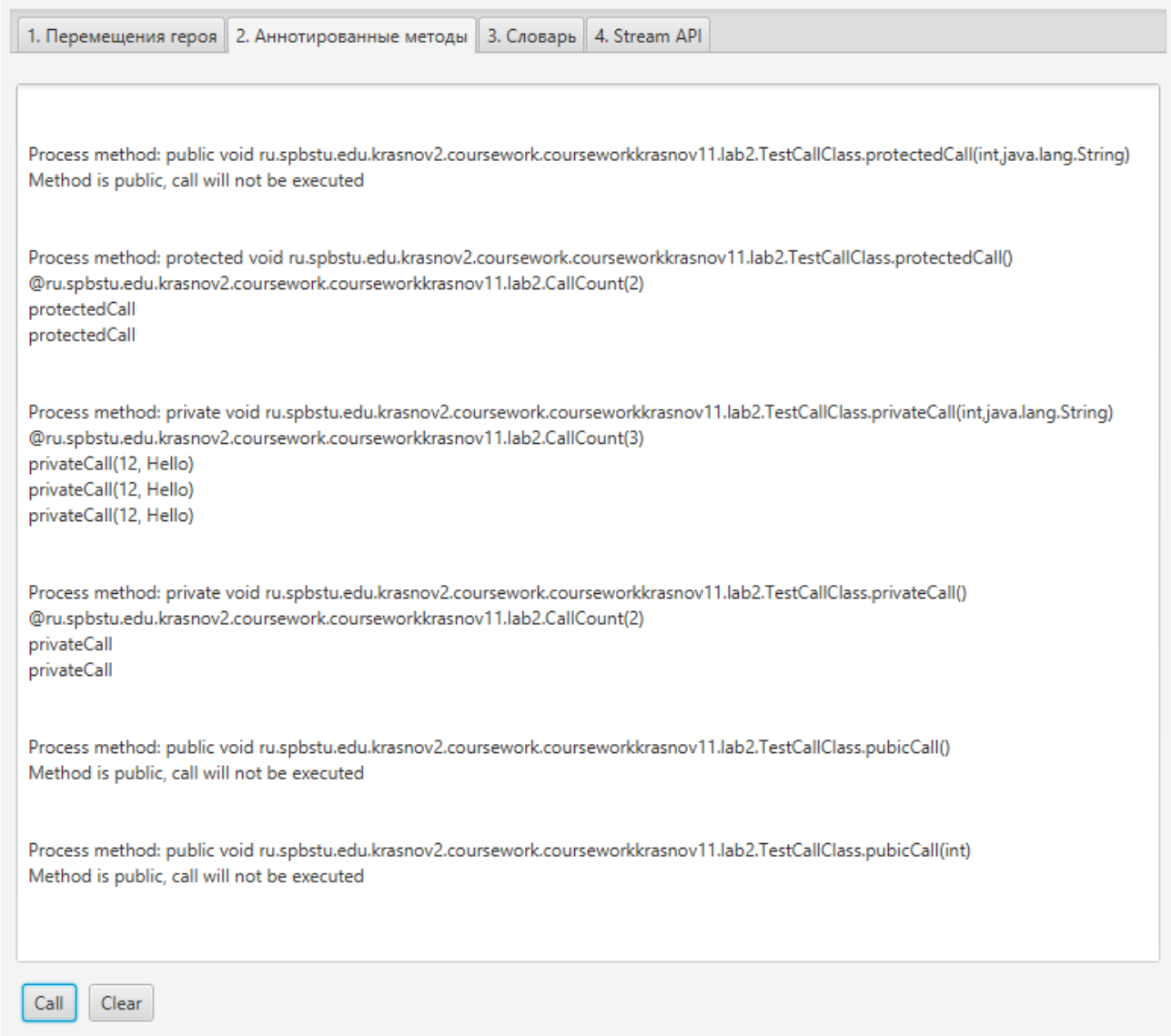


Рисунок 4. Вкладка «Задание 2»

В рамках второго задания был создан класс Caller и создана аннотация CallCount, с помощью которой можно помечать методы, которые необходимо вызвать классу Caller, в соответствии с заданием 3.

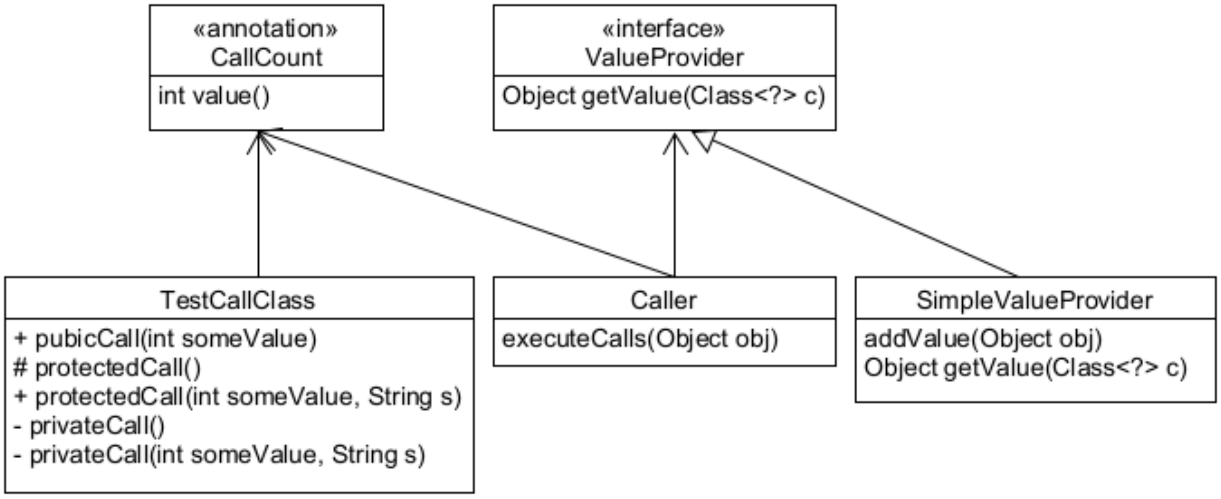


Рисунок 5. Диаграмма классов задания 2

Для вызовов методов переданного объекта классу Caller необходимы типизированные значения для передачи в параметры методов. Для получения таких значений создан интерфейс ValueProvider и его простая реализация SimpleValueProvider. Caller получает необходимые для вызова значения, используя этот интерфейс, и делает столько вызовов, сколько указано в аннотации CallCount.

В этом задании используется перенаправление вывода потока stdout (класс StdoutRedirectExecuter) в UI элемент, представляющий текстовую область (TextArea Java Fx).

Задание 3. Переводчик

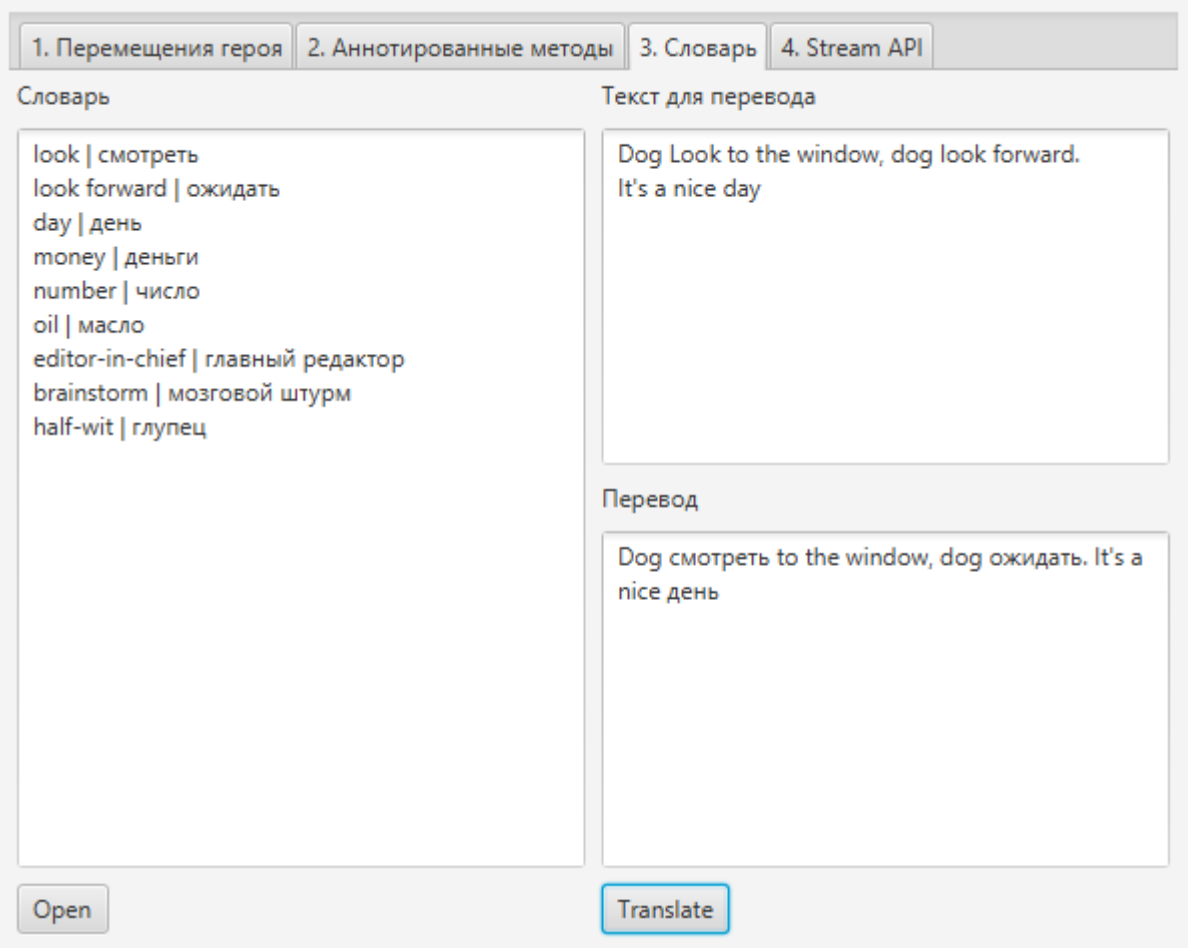


Рисунок 6. Вкладка «Задание 3»

В задании 3 создан класс WordDictionary, который читает данные словаря из стандартного класса Reader, который может представлять собой, например, чтение из файла. Для разбора считанного текста используется стандартный класс Scanner. Класс WordDictionary предоставляет метод translate для осуществления перевода, в соответствии с условиями задания 3.

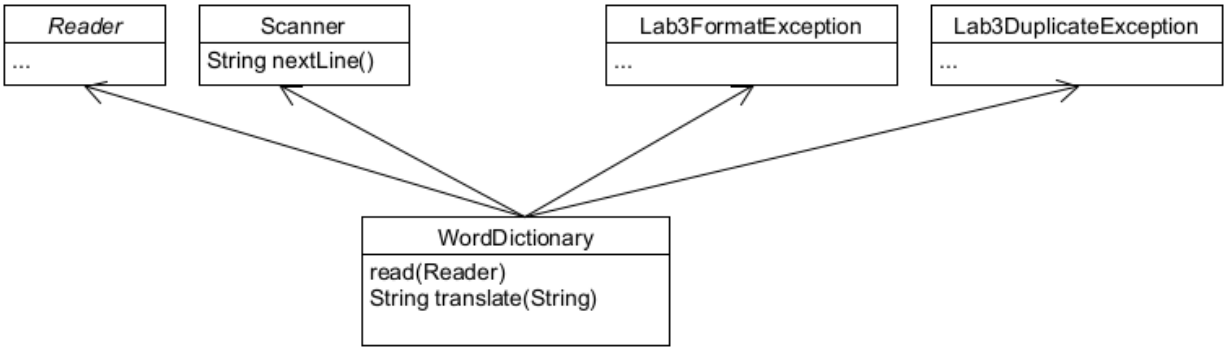


Рисунок 7. Диаграмма классов задания 3

Для возможности детектирования ошибок при чтении файла, помимо возможных стандартных исключений, используются дополнительные созданные в рамках задания 3 Lab3FormatException и Lab3DuplicateException.

Задание 4. Stream API

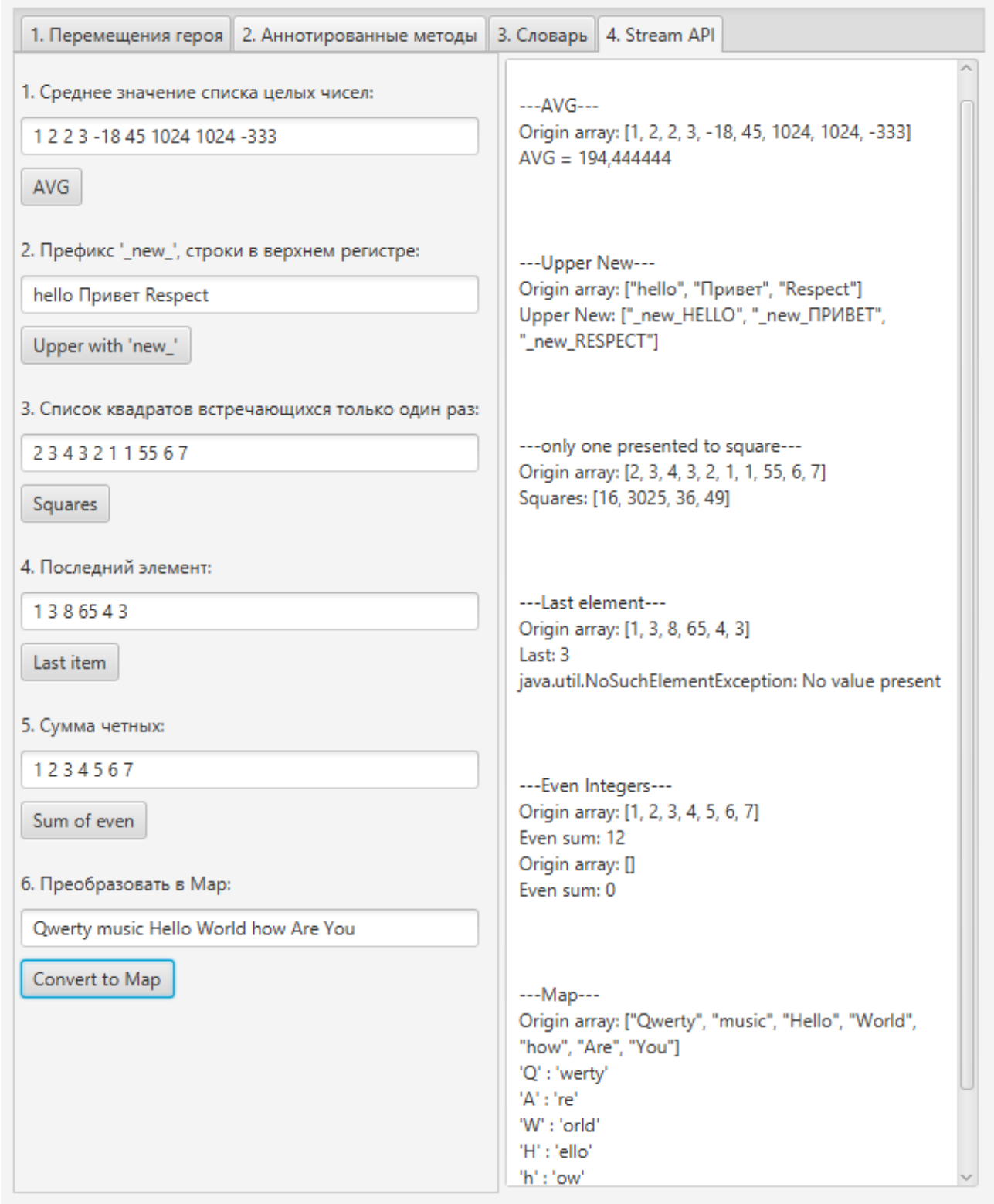


Рисунок 8. Вкладка «Задание 4»

Вся логика выполнения задания 4 описана напрямую в контроллере Lab4Controller. Используются коллекции List<>, ArrayList<>, Map<> и другие. Для непосредственного выполнения задания используются класс Stream и соответствующие методы работы с ним (Stream API).

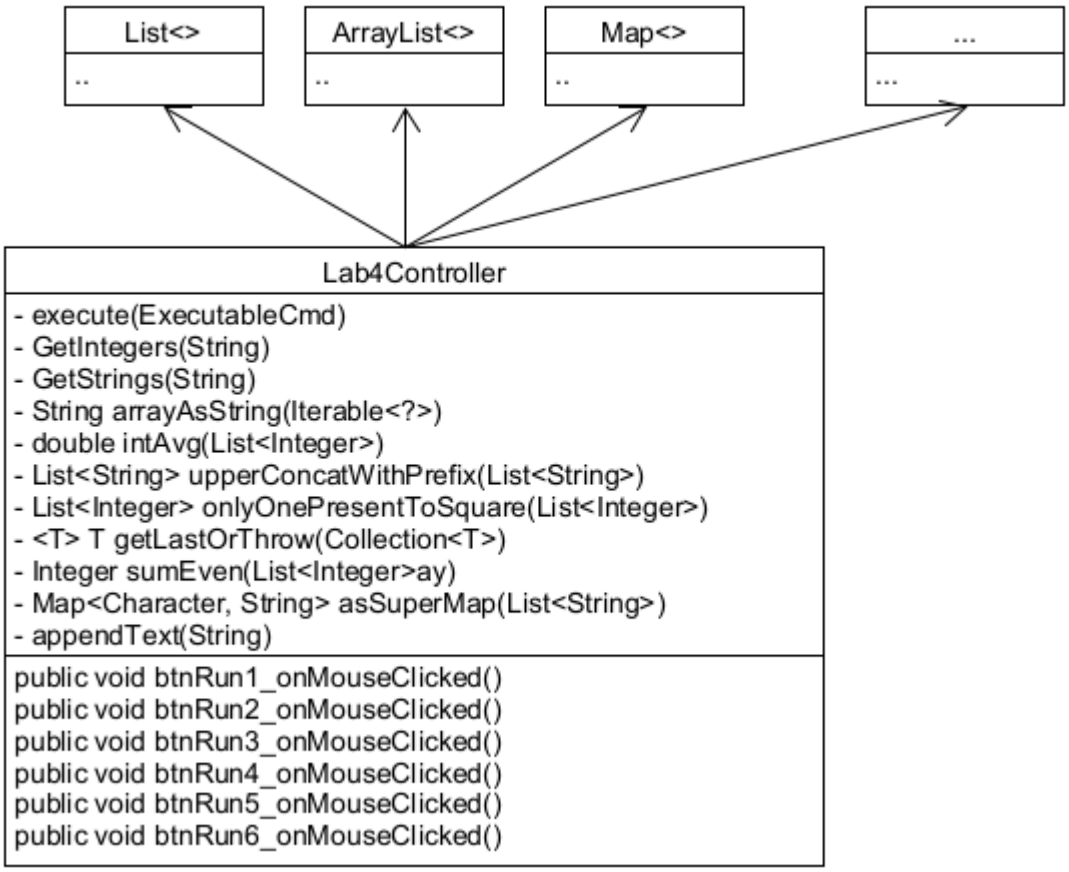


Рисунок 9. Диаграмма классов задания 4

В этом задании используется перенаправление вывода потока stdout (класс StdoutRedirectExecuter) в UI элемент, представляющий текстовую область (TextArea Java Fx).

Выводы

В ходе выполнения курсовой работы реализовано приложение на языке Java с использованием UI фреймворка Java Fx. Выполнены модификации четырех лабораторных работ для корректной работы в окружении графического интерфейса. Получены навыки работы с файлами представления *.fxml и контроллерами фреймворка Java Fx.

Задания выполнены, были учтены дополнительные требования к каждому заданию. Поставленные задачи по курсовой работе были выполнены, цель курсовой работы достигнута.