САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И КИБЕРБЕЗОПАСНОСТИ

Дисциплина: Объектно-ориентированное программирование (Java)

Курсовая работа:
Приложение на Java — Паттерн «Стратегия», аннотации и рефлексия, переводчик, Stream API; GUI на Swing

Студент: *Момотова Алиса*, группа 30322

Руководитель: *Алексей Маслаков*

Санкт-Петербург — 2025

Содержание

Аннотация

В работе реализовано десктопное приложение на Java (Swing), объединяющее четыре самостоятельных задания:

1. Паттерн «Стратегия» для класса `Hero` (пешком/лошадь/полёт).
2. Аннотация `@Repeat(int)` и рефлексивный вызов аннотированных `protected`/`private` методов произвольной сигнатуры.
3. Переводчик с чтением словаря из файла формата `левая | перевод`, пользовательские исключения `InvalidFileFormatException`, `FileReadException`, правило «самого длинного совпадения».
4. Набор задач на Stream API.

Все подсистемы подключены к единому GUI (`JTabbedPane`); вывод ведётся в нередактируемые текстовые области. Для задания 3 предусмотрен выбор файлов словаря/текста; для задания 4 — ввод данных.

1 Введение

Цель — разработать модульное Java-приложение, демонстрирующее шаблоны проектирования, аннотации и рефлексию, обработку исключений, Stream API и интеграцию подсистем в единый GUI. Стек: Java 17, Gradle, Swing. Пакеты: `gui`, `hero`, `lab2`, `translator`, `streamApi` (или `org.example`).

2 Постановка задачи

2.1 Задание 1: Паттерн «Стратегия»

Класс `Hero` перемещается по стратегиям `MoveStrategy` (`Walk`, `Horse`, `Fly`). Метод `move` возвращает описание действия.

2.2 Задание 2: Аннотация и рефлексия

Определить `@Repeat(int)`; создать класс с публичными/защищёнными/приватными методами и аннотировать часть из них. Из другого класса вызвать *только* `protected`/`private` аннотированные методы требуемое число раз, подставляя дефолтные аргументы.

2.3 Задание 3: Переводчик

Словарь UTF-8: `левая | перевод`. Перевод построчно, регистр игнорируется; при множестве кандидатов — берётся *самый длинный* слева. Исключения: `InvalidFileFormatExceptio` (формат), `FileReadException` (файл).

2.4 Задание 4: Stream API

Методы: среднее `List<Integer>`; строки в верхний регистр с префиксом `_new_`; квадраты элементов, встречающихся ровно один раз; последний элемент коллекции (или исключение); сумма чётных чисел массива; преобразование списка строк в `Map<Character,String>`.

3 Архитектура и диаграмма классов

Пакеты

- **gui**: `App` — точка входа (единственный `main`).
- **hero**: `Hero`, `Position`, `MoveStrategy`, `Walk`, `Horse`, `Fly`.
- **lab2**: `Repeat` (аннотация), `SampleService`, `Invoker`.

- **translator**: `Dictionary`, `DictionaryLoader`, `Translator`, `InvalidFileFormatException`, `FileReadException`.
- **streamApi/org.example**: `StreamTasks`.

Диаграмма классов

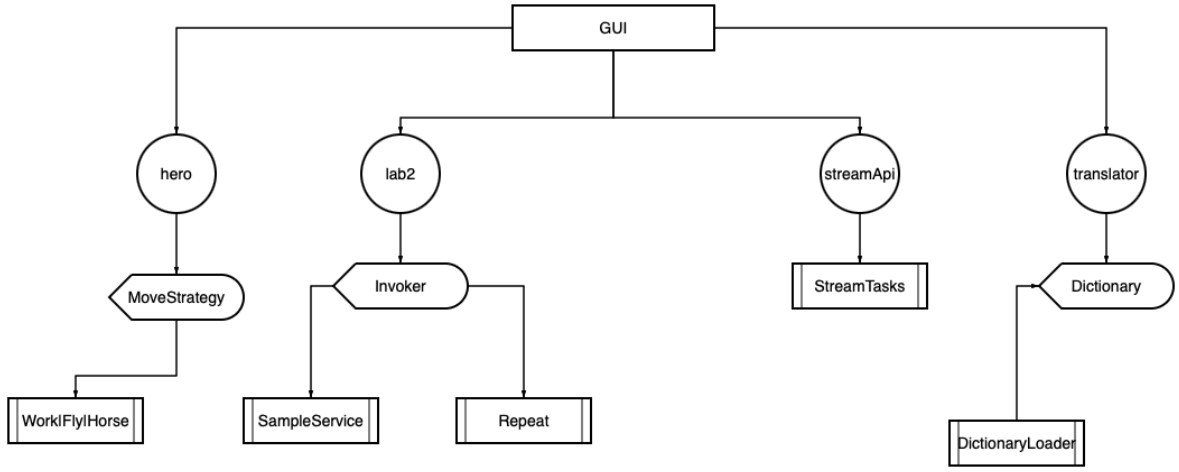

Рис. 1: Enter Caption

4 Реализация

4.1 Задание 1: «Стратегия»

Идея. `Hero` зависит от абстракции `MoveStrategy`. Переключение через `setStrategy`. `move` делегирует стратегии и возвращает строку для GUI.

4.2 Задание 2: Аннотация и рефлексия

Аннотация. `@Repeat(int)`: `RetentionPolicy.RUNTIME`, `Target(ElementType.METHOD)`.

Инвокер. `Invoker` находит методы с `@Repeat`, оставляет `protected`/`private`, делает `setAccessible(true)`, генерирует дефолтные аргументы и вызывает нужное число раз, логируя результаты.

4.3 Задание 3: Переводчик

Формат. «левая | перевод», строки с # игнорируются. Ошибки формата/чтения — пользовательские исключения.

Алгоритм. Токенизация по пробелам, на каждой позиции — поиск самого длинного совпадения по ключам словаря (регистронезависимо).

4.4 Задание 4: Stream API

Методы реализованы стандартными операциями потоков: `map/mapToInt/filter/reduce/group`

5 Графический интерфейс (Swing)

Главное окно — `JFrame` + `JTabbedPane`. Для каждой задачи отдельная вкладка; все результаты — в `JTextArea` с `setEditable(false)`. Для переводчика — выбор словаря/текста (`JFileChooser`); для Stream API — поля ввода.

6 Сборка и запуск

6.1 Структура проекта

```
coursework-java/
 app/
    build.gradle
    src/main/java/
```

```
gui/          (App.java - единственный main)
hero/         (Hero, MoveStrategy, Walk, Horse, Fly, Position)
lab2/         (Repeat, SampleService, Invoker)
translator/   (Dictionary, DictionaryLoader, Translator, Exceptions)
streamApi/    (StreamTasks)  # или org/example/StreamTasks
```

6.2 Gradle

```
plugins { id 'application' }
application { mainClass = 'gui.App' }
```

Сборка и запуск:

```
cd coursework-java/app
./gradlew clean build
./gradlew run
```

7 Перечень выполненных работ

1. Реализован паттерн «Стратегия» для перемещения героя.
2. Создана аннотация `@Repeat` и рефлексивный инвокер.
3. Разработан переводчик с пользовательскими исключениями и правилом длиннейшего совпадения.
4. Имплементированы шесть методов на Stream API.
5. Создан общий GUI (Swing) с вкладками и вводом/выводом.
6. Подготовлены инструкции по сборке/запуску Gradle-проекта.
7. Построена UML-диаграмма.

8 Заключение

Показано применение интерфейсов и шаблонов, аннотаций и рефлексии, потоковых операций, исключений и GUI на Swing. Архитектура модульна, что упрощает демонстрацию и дальнейшее расширение.