Санкт-Петербургский политехнический университет
Институт компьютерных наук и технологий
Кафедра «Компьютерные системы и программные технологии»

КУРСОВОЙ ПРОЕКТ

Разработка игры "Terra Incognita"

по дисциплине «Технологии программирования»

Выполнил студент Козырев Д.В.
гр. 3530901/10001

Преподаватель Алексюк А.О.

Санкт-Петербург
2022

ЗАДАНИЕ

НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

студенту группы 3530901/10001 Козыреву Даниилу Владимировичу

1. Тема проекта: создание игры «Terra Incognita» с графическим интерфейсом.
2. Срок сдачи законченного проекта: 18 июня
3. Исходные данные к проекту: требования к реализовываемому проекту
4. Содержание пояснительной записки (перечень подлежащий разработке вопросов): введение, основная часть (текст программы, описание программы, испытания программы), заключение, список использованных источников.

Дата получения задания: «2» апреля 2020 г.

Руководитель Алексюк А.О.

Задание принял к исполнению Козырев Д.В.

9 апреля 2022г.

СОДЕРЖАНИЕ

1. Цель работы

Создать и протестировать игру “Terra Incognita” с графическим интерфейсом для числа игроков от 2-х до 4-х.

2. Правила игры

Terra Incognita – игра для игроков в компании от 2-х до 4-х игроков. Имеется заранее сгенерированный лабиринт, который неизвестен игрокам и состоящий из клеточек стен, входа и выхода, клеток «кротовых нор» сос воим номером, которые перемещают игрока на клетку «кротовой норы» со следующий номером, а так же имеющий особую клеточку сокровища и пустые (ничем не занятые) клеточки. Игроки по очереди делают ходы, начиная со старта и, делая шаги в одном из 4-х направлений, открывают лабиринт.

Цель игры – найти сокровище и раньше своего противника добраться с ним до выхода.

3. Описание решения

Программа была написана по паттерну **MVC (Model-View-Controller)** для отделения модели (логики, описывающей правила игры) программы от визуальной составляющей, в связи с этим весь код поделен на два пакета: model, controllers; а также 3 fxml-файла, описывающие графический интерфейс приложения. В коде дополнительно был реализован пакет утилит, включающий: дополнительные классы исключений, помогающие при отладке приложения и информировании, когда что-то пошло не так; утилиты для проведения тестов, а также загрузчик ресурсов, необходимый для загрузки всех изображений и лабиринтов.

Класс «Main», содержащий функцию, с которой начинается выполнение программы, служит лишь неоходимой оберткой над классом «App», для корректной работы библиотеки JavaFX, на основе которой построены представление и контроллеры. Класс «App» наследуется от класса библиотеки JavaFX «Application» и

является методом, в котором происходит вся начальная логика приложения и обладающий всем необхидим, к чему должен быть доступ из любого места программного кода.

3.1. Пакет model

Пакет «model» описывает логику правил игры: перемещение, открытие клетки, если та не была еще открыта данным игроком, взаимодействие игрока со стенами, сокровищем, и выходом из лабиринта. Пакет «model» содержит:

- Публичный класс «Game», хранящий в себе сокрытый от игроков лабиринт, и предоставляющий доступ к данным о участвующих игроках.
- Публичный enum «MovementDirection», который описывает 4 возможных направления движения и соответствующий каждому метод изменения координаты (Публичный класс «Point», см пакет «utils»)
- Публичный класс «Player», описывающий состояния игрока и открытую им часть лабиринта. Предоставлеяет метод **move(MovementDirection)**, перемещающий игрока (в случае, если возможно), и возвращающий позиции клеток, которые были открыты игроком за это ход.
- Пакет «tiles», содержащий публичный абстрактный класс «Tile», от которого наследуются 7 классов, реулизующие основные типы клеток. При этом используется шабоон проектирования «**singleton**».
- Пакет «desk», содержащий классы «Desk» и наследуемый от него класс «Labyrinth». «Desk» предоставляет базовый функционал по хранению клеточек в сетке, получению и добавлению клеток в сетку. «Labyrinth» расширяет класс «Desk», добавляет статический метод-генератор объекта класса «Labyrinth», гетеры на получение размеров лабиринта и его стартовой позии, а так же метод валидации лабиринта, проверяющий, что все необходимое есть в лабиринте, иначе выбрасывающий исключение

типа «NoNeededTileException» (см. пакет «utils»).

3.2. Пакет controllers

Пакет «controllers» содержит классы необходимые для удобного обращения со сценами и их контроллерами (классы, задача которых - описать функционал по передачи информации модели и реагированию пользовательского интерфейса на результат действий в модели), в частности:

- Публичный абстрактный класс «BasicController», реализующий интерфейс «Initializable», необходимый для корректной загрузки fxml-файлов и их подключению к котроллерам.
- Публичные классы «StartWindowController», «GameWindowController» и «EndWindowController», привязанные к конкретной сцене, и реализующие задачи контроллеров.
- Публичный класс «StageController», объект которого способен хранить в себе ссылки на сцены и их контроллеры, поэтому можно удобно осуществлять поиск и получение обоих по названию сцены с помощтю методов **getControllerOf(String), loadScene(URL), prepareScene(String), и showScene()**

3.3. Пакет utils

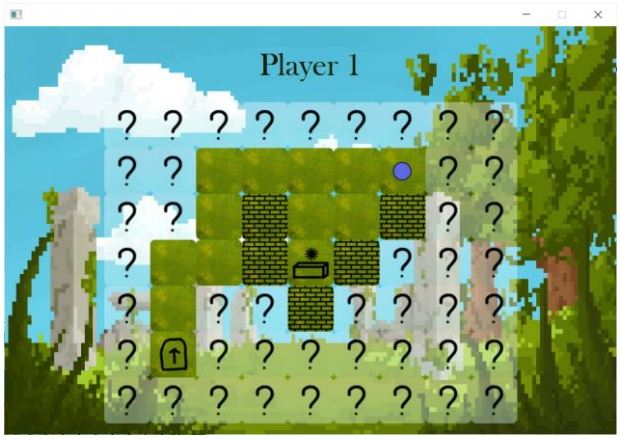
Пакет «utils» предоставляет дополнительный функционал, полезный для всего кода и не относящийся обязательно только к одному классу. Пакет содержит:

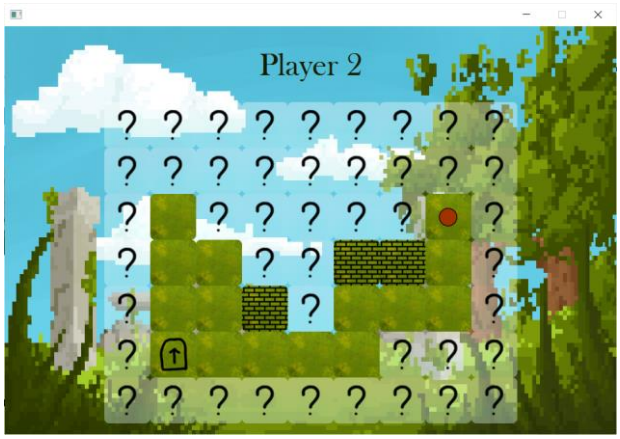
- Пакет «exceptions» содержит ряд классов исключений, дающие более наглядное понимание того, что могло пойти не так в ходе выполнения программы, а так же небольшой класс «ExceptionsUtils», содержащий публичный статический метод, возвращающий весь стек ошибки в виде строки.
- Публичный служебный класс «Point» - удобная обертка над двумя

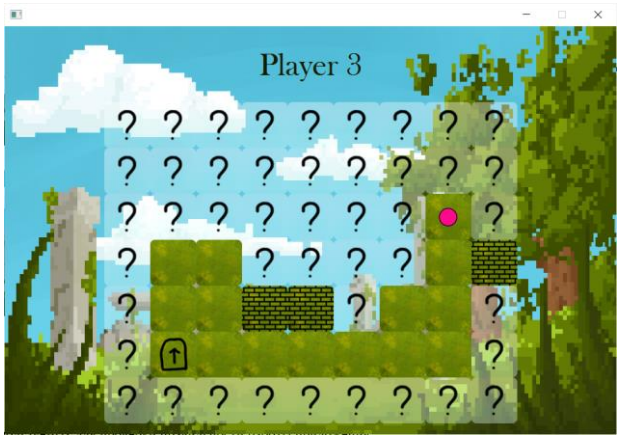
переменными типа **int** представляющая координату.

- Публичный класс «ResourceLoader» имеющий один метод, возвращающий объект класса **InputStream,** который используется для загрузки изображений и лабиринтов.
- Публичный класс «Utils» содержащий служебные методы, которые по своей идеи не могут быть отнесенны ни к одному из классов описанных ранее. Здесь есть методы для загрузки FXML сцены и чтения строк из файла, возвращая список этих строк (**List<String>**), а так же метод, который вызывается при срабатывании любой ошибки **logErrorWithExit(Exception)**. Этот метод в зависимости от того, проводится сейчас тестрирование или нет, выполняет разный фнукционал. Программа используется пользователем и что-то пошло не так – начинается построение дилогового окна с сообщением об ошибки. Программа троходит тестирование – ошибка выбрасывается в специальной обертке «ExceptionWrapper», который просто расширяет класс «RuntimeException».
- Класс «TestUtils» содержит лишь переменную, определяющую, проводятся сейчас тесты или нет.

4. Внешний вид приложения







5. Тестирование программы

С использованием библиотеки JUnit было написано 8 автоматических тестов, используемых для проверки правильности работы приложения. Тесты нацелены на проверку правильной работы модели, и включают в себя тесты на проверку входных данных, проверку изменения позиции (при возможности), при попытки передвижания игрока, проверку работы «кротовых нор» (см. «Правила игры»),

проверку на действительное завершение игры, когда игрок с сокровищем доходит до выхода.

6. Заключение

Было создано приложение с графическим интерфейсом, для игры в «Terra Incognita». Были разработаны автоматические тесты для проверки правильности кода. В ходе выполнения задания я изучил шаблон MVC и научился работать с библиотекой JavaFX.

Исходные файлы приложения лежат в репозитории на GitHub: https://github.com/PurpleLimon/ProgrammingLabSummer2022Task3.

7. СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. https://docs.oracle.com/en/java/javase/18/ – документация языка Java.
2. https://ru.wikipedia.org/wiki/Лабиринт_(игра_на_бумаге) – правила игры.