1

Санкт-Петербургский политехнический университет Институт компьютерных наук и технологий

**Кафедра «Компьютерные системы и программные технологии»**

**КУРСОВОЙ ПРОЕКТ**

**Разработка игры "Reversi"**

по дисциплине «Технологии программирования»

Выполнил студент гр. 5130901/20002

Преподаватель

Ляпина С.В.

Степанов Д.С.

3 октября 2023 г.

Санкт-Петербург

2023

2

Санкт-Петербургский политехнический университет

**ЗАДАНИЕ**

**НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА**

студенту группы 5130901/20002 Ляпиной Светлане Владимировне

1. Тема проекта: создание игры «Шашки» с графическим интерфейсом для игры двух человек.

2. Срок сдачи законченного проекта: 3 октября

3. Исходные данные к проекту: требования к реализовываемому проекту

4. Содержание пояснительной записки: введение с описанием правил игры, основная часть (технологии JavaFX, MVC и их применение в приложении), описание автоматических тестов для бизнес-логики, заключение, список используемых источников.

Дата получения задания: «23» мая 2023 г.

Руководитель

Задание принял к исполнению

Степанов Д.С..

Ляпина С.В.

23 мая 2023 г.

3

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ ..........................................................................................................4 ОПИСАНИЕ ПРЕДЛОЖЕННОГО РЕШЕНИЯ..................................................5 ТЕСТИРОВАНИЕ ПРОГРАММЫ ......................................................................7 ЗАКЛЮЧЕНИЕ....................................................................................................8 СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ..................................................9

4

**ВВЕДЕНИЕ**

Цель работы: создать и протестировать игру «Шашки» с графическим

интерфейсом для игры двух человек.

Правила игры:

Шашки *—* логическая настольная игра для двух игроков, заключающаяся в передвижении определённым образом фишек-шашек по клеткам шашечной доски. Во время партии каждому игроку принадлежат шашки одного цвета: чёрного или белого (иногда других цветов, один из которых считается тёмным, а другой — светлым). Цель игры — взять все шашки соперника или лишить их возможности хода (запереть). Существует несколько вариантов шашек, различающихся правилами и размерами игрового поля.

*Основные правила игры, которые действуют во всех вариантах шашек:*

* Все шашки, участвующие в партии, выставляются перед началом игры на доску. Далее они передвигаются по полям доски и могут быть сняты с неё в случае боя шашкой противника.
* Брать шашку, находящуюся под боем, обязательно. Исключение: «Сибирские шашки».
* Существует только два вида шашек: простые и дамки. В начале партии все шашки простые. Простая шашка может превратиться в дамку, если достигнет последнего противоположного горизонтального ряда доски (дамочного поля).
* Простые шашки ходят только вперёд на следующее поле. Дамки могут ходить и вперёд, и назад

5

**ОПИСАНИЕ ПРЕДЛОЖЕННОГО РЕШЕНИЯ**

Для создания графического пользовательского интерфейса (GUI)

использовалась библиотека JavaFX, которая обладает большим числом заготовок элементов интерфейса. В частности были использованы следующие элементы: кнопка (Button), текст (Text), ссылка (Hyperlink), холст (Scene),, а также вертикальные и горизонтальные ряды (VBox и HBox). Взаимодействие пользователя с графическим элементом описывается в событиях для этого элемента. Существует множество различных событий. Например, на нажатие, на наведение или на уход мыши.

Программа была написана с использованием концепция MVC (model-view-controller) для отделения бизнес-логики от визуализации, поэтому весь код разбит на семь классов.

Класс `Tile` представляет плитку на доске для игры в шашки. Он наследуется от класса `Rectangle` из пакета `javafx.scene.shape` и расширяет его функциональность.

У класса `Tile` есть следующие методы:

- Метод `hasPiece()` проверяет, содержит ли плитка фигуру. Он возвращает `true`, если на плитке есть фигура, и `false` в противном случае.

- Метод `getPiece()` возвращает фигуру, которая находится на плитке.

- Метод `setPiece(Piece piece)` устанавливает фигуру на плитку.

- Конструктор `Tile(boolean light, int x, int y)` создает объект `Tile`. Он принимает параметры `light` (логическое значение, указывающее, будет ли плитка светлой) и `x`, `y` (целочисленные значения, указывающие позицию плитки на доске). В конструкторе устанавливаются ширина и высота плитки, ее положение на доске и цвет (светлый или темный, в зависимости от значения `light`).содержится main функция программы, задание параметров окна и расположения внутри него всех элементов интерфейса, а также прописаны события на действия мыши.

Класс `PieceType` определяет два значения: `BLACK` и `WHITE`, которые представляют цвет шашек.

Класс `Piece` представляет фигуру шашки на доске. Он наследуется от

6

класса `StackPane` и содержит методы и свойства, которые определяют поведение и внешний вид фигуры.

Класс `MoveType` определяет три значения: `NONE`, `NORMAL` и `KILL`, которые представляют типы ходов в игре шашек.

- Значение `NONE`: указывает, что ход невозможен. Это может быть, например, когда пытаемся переместить фигуру на занятую клетку или находящуюся за пределами доски.

- Значение `NORMAL`: указывает, что ход является обычным перемещением фигуры с одной клетки на другую. В этом случае нет никакого убийства фигур противника.

- Значение `KILL`: указывает, что ход является убийством фигуры противника. В этом случае перемещаемая фигура перемещается через одну.

Класс `MoveResult` представляет результат хода в игре в шашки.

Класс `HelloApplication` наследуется от класса `Application` и содержит основную логику игры.

У класса есть константы, такие как `TILE\_SIZE`, `WIDTH` и `HEIGHT`, которые определяют размеры доски и плиток на ней.

Поля `board`, `tileGroup` и `pieceGroup` представляют доску, группу плиток и группу фигур соответственно.

Метод `createContent()` инициализирует доску, плитки и фигуры шашек на доске.

Метод `tryMove()` проверяет возможность выполнения хода шашкой и возвращает результат хода в виде объекта класса `MoveResult`.

Метод `toBoard()` преобразует пиксельные координаты в координаты на доске.

Метод `start()` запускает приложение, создавая сцену и отображая окно приложения.

Метод `makePiece()` создает экземпляр класса `Piece` для представления фигуры шашки на доске.

В соответствии с выбранным шаблоном разработки, пользователь взаимодействует с view, все команды от пользователя обрабатывает controller.

7

**ТЕСТИРОВАНИЕ ПРОГРАММЫ**

Для тестирование бизнес-логики было написано 5 автоматических тестов,

использующих возможности библиотеки JUnit.

Первый тест проверяет, что метод `getType()` возвращает правильный тип хода. В этом тесте создается экземпляр класса `MoveResult` с типом хода `MoveType.NORMAL`. Затем мы используем метод `assertEquals()` для проверки, что возвращаемое значение метода `getType()` соответствует ожидаемому значению `MoveType.NORMAL`.

Второй тест проверяет, что метод `getPiece()` возвращает правильную фигуру. В этом тесте создается фигура `Piece` и экземпляр класса `MoveResult` с типом хода `MoveType.KILL` и этой фигурой. Затем мы используем метод `assertEquals()` для проверки, что возвращаемое значение метода `getPiece()` соответствует ожидаемой фигуре.

Третий тест проверяет, что конструктор `MoveResult` с одним аргументом типа `MoveType` правильно инициализирует переменную `piece` значением `null`. В этом тесте создается экземпляр класса `MoveResult` с типом хода `MoveType.NONE`. Затем мы используем метод `assertNull()` для проверки, что возвращаемое значение метода `getPiece()` равно `null`.

Четвёртый тест проверяет, что конструктор `MoveResult` с двумя аргументами типа `MoveType` и `Piece` правильно инициализирует переменные `type` и `piece`. В этом тесте создается фигура `Piece` и экземпляр класса `MoveResult` с типом хода `MoveType.JUMP` и этой фигурой. Затем мы используем метод `assertEquals()` для проверки, что возвращаемые значения методов `getType()` и `getPiece()` соответствуют ожидаемым значениям `MoveType.JUMP` и созданной фигуре соответственно.

Тестируется функциональность класса `MoveResult` в различных ситуациях, чтобы убедиться, что он работает должным образом и возвращает правильные результаты ходов.

8

**ЗАКЛЮЧЕНИЕ**

Было создано приложение с графическим интерфейсом, предназначенное

для игры двух человек в Reversi. Также были разработаны автоматические тесты для проверки работоспособности кода. В ходе выполнения этого задания мною были изучены библиотека JavaFX и шаблон MVC.

Исходные файлы приложения лежат в репозитории на GitHub:

https://github.com/svitas3/chekers.



Рис 1. Скриншот приложения

9

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. https://metanit.com/java/tutorial/ – описание языка Java

2. https://metanit.com/java/javafx/ – описание JavaFX

3. <https://ru.wikipedia.org/wiki/Шашки> — правила шашек

4. https://refactoring.guru/ru/design-patterns – сборник шаблонов разработки приложений

5. https://logic-games.spb.ru/checkers/?lang=ru%27 – онлайн игра Шашки