Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

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

# КУРСОВОЙ ПРОЕКТ

**Reversi with AI**

по дисциплине «Алгоритмы и структуры данных (Java)»

|  |  |  |
| --- | --- | --- |
| Выполнил |  |  |
| студент гр.23531/2  Руководитель | \_\_\_\_\_\_\_\_\_ | Д.В. Павлов |
|  | \_\_\_\_\_\_\_\_\_  Санкт-Петербург  2018 | А.О. Алексюк  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г. |

Санкт-Петербургский политехнический университет Петра Великого

**ЗАДАНИЕ**

**НА ВЫПолнение курсового проекта**

студенту группы 23531/2 Павлову Даниилу Вячеславовичу

*(номер группы) (фамилия, имя, отчество)*

1. ***Тема проекта:*** Reversi with AI

***2. Срок сдачи законченного проекта:*** 28.05.18

***3. Исходные данные к проекту***: IDE: IntelliJ IDEA 2018.1.2 (Ultimate Edition), JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o, Version Java: 10

***4. Содержание пояснительной записки*** (перечень подлежащих разработке вопросов): введение, основная часть (текст программы, описание программы, испытания программы), заключение, список использованных источников.

***Дата получения задания***: « 1 » сентября 2018 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.О. Алексюк

*(подпись) (инициалы, фамилия)*

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В Павлов

*(подпись студента) (инициалы, фамилия)*

\_\_\_\_\_\_\_\_\_\_\_\_

*(дата)*

# Введение

Правила игры:

В игре используется квадратная доска размером 8 × 8 клеток (все клетки могут быть одного цвета) и 64 специальные фишки, окрашенные в белые и чёрные цвета соответственно. Один из игроков играет белыми, другой — чёрными.

Делая ход, игрок должен поставить свою фишку на одну из клеток доски таким образом, чтобы между этой поставленной фишкой и одной из имеющихся уже на доске фишек его цвета находился непрерывный ряд фишек соперника, горизонтальный, вертикальный или диагональный (другими словами, чтобы непрерывный ряд фишек соперника оказался «закрыт» фишками игрока с двух сторон). Все фишки соперника, входящие в «закрытый» на этом ходу ряд, переворачиваются на другую сторону (меняют цвет) и переходят к ходившему игроку.

Если в результате одного хода «закрывается» одновременно более одного ряда фишек противника, то переворачиваются все фишки, оказавшиеся на всех «закрытых» рядах.

Игрок вправе выбирать любой из возможных для него ходов. Если игрок имеет возможные ходы, он не может отказаться от хода. Если игрок не имеет допустимых ходов, то ход передаётся сопернику.

Игра прекращается, когда на доску выставлены все фишки или когда ни один из игроков не может сделать хода. По окончании игры проводится подсчёт фишек каждого цвета, и игрок, чьих фишек на доске выставлено больше, объявляется победителем. В случае равенства количества фишек засчитывается ничья.

# Текст программы

[**https://github.com/DaniilPavlov/ProgrammingLabAutumn2018**](https://github.com/DaniilPavlov/ProgrammingLabAutumn2018)

# Описание программы

* **Main.java**

Данный класс отвечает за создание окна, его размеры, расположение и привязывает к нему игру.

# logic package

* **ComputerDecision.java**

Данный класс отвечает за выбор наилучшего хода AI из всех доступных, исходя из позиции фишки на доске (её ряда\столбца и соседства с фишками другого цвета). За нахождение в угле начисляется **30 очков**, просто у края

доски – **5 очков**, но у опасного края доски будет вычитаться **6 очков**. Также вычитается **12 очков** в пред. угловой зоне и начисляется **15 очков** при нахождении фишки между двумя фишками соперника.

* **Direction.java**

Данный класс отвечает за возможные направления фишек на доске. К каждой стороне света привязывается движение в координатах: **EAST** = (1, 0),

**NORTH** = (0, 1), **NORTH\_EAST** = (1, 1), **WEST** = (-1, 0), **SOUTH** = (0, -1),

**SOUTH\_WEST** = (-1, -1), **SOUTH\_EAST** = (1, -1), **NORTH\_WEST** = (-1, 1).

* **GameBoardStructure.java**

Данный класс отвечает за создание логики на игровой доске и привязку к каждой клетке значения. Клетки (4, 4) и (5, 5) по правилам игры занимают белые фишки, а (5, 4) и (4, 5) – черные. Остальные задаются вакантными.

* **Move.java**

Данный класс является реализацией движения фишек.

* **Reversi.java**

Основной класс. Отвечает за сам ход, который каждый раз вызывается новым потоком. 1 поток – 1 ход. Реализует в себе все правила игры. Занимается поиском возможных ходов, определяет победителя.

* **ScoreAddition.java**

Данный класс предоставляет пищу для размышления для **“ComputerDecision.java”.** Здесь осуществляется определение позиции фишки на доске и начисление очков. Если фишка не подходит ни под одну из специальных позиций, то ей будет начислено количество перевернутых фишек соперника.

# GUI package

* **Animation.java**

Данный класс отвечает за визуализацию изменения цвета фишек путем осветления или затемнения, в зависимости от цвета.

* **ScoreBoard.java**

Данный класс отвечает за показ текущего хода (зеленым цветом окрашивается имя “Player” или “Computer”, в зависимости от того, чей сейчас ход) и за показ текущих очков у игроков.

* **Update.java**

Данный класс отвечает за обновление доски. Если клетка является возможным ходом для текущего игрока, она окрашивается в зеленый цвет и в ее центре находится число, показывающее количество будущих отнятых фишек у противника. Также данный класс отвечает за хранение в порядке фишек и не разрешает ходить в место, где уже есть какая-либо из фишек.

# Испытание программы

Были реализованы тесты для проверки точности анимации (AnimationTest.java). Тесты для верности подсчета очков игроков приведены в классе “ScoreBoardTest.java”. Также был создан класс для проверки начисления данных очков. Очень много тестов для разных развитий событий, чтобы полностью проверить работы логики. Это реализовано в классе “ScoreAdditionTest.java”. Ну и конечно тесты для основной логики игры находятся в классе “ReversiTest.java”. Проверки направлений движения фишек, их возможности или невозможности. Проверка на то, выбирает ли AI наилучший вариант или нет.

Все тесты подтвердили корректность программы. Также было проведено ручное тестирование и не однократно. Краевые случаи исследованы, при рестарте игры проблем тоже не возникало. Ни в процессе игры, ни после окончания. Количество возможных захватов фишек показывается верно и ни единого бага не было найдено.

# Заключение

В заключении хочется подвести итог: недопустимо много времени было проведено за устранением проблем, решение которых, по законам жанра, занимало одну строчку.

Изначально работа шла очень тяжело, но постепенно, с приращением функционала, появлялось все больше идей, реализация которых занимала не так много времени, как раньше, так-как я уже более-менее понимал, как действовать в той или иной ситуации. Однако больше всего времени заняла не сама разработка, а ручное тестирование программы. Очень часть приходилось играть с компьютером по 10, а то и 20 минут для создания определенных краевых позиций и проверки корректности работы AI. И если на первых парах это немного нервировало, так как приходилось еще многое сделать, то в конце это вызывало лишь ностальгию, так как всегда тяжело начинать и ты вступаешь в неизвестность. Перед тобой пустые классы и ты еще только представляешь, как будет выглядеть программа в дальнейшем.

# Список используемых источников

* StackOverFlow
* JavaDoc
* К. Арлольд, Д.Гослинг: Язык программирования JAVA
* Эккель Б. Философия Java
* https://github.com/Kotlin-Polytech/FromKotlinToJava