****ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Факультет Информатики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
 к курсовой работе по дисциплине «Объектная распределенная обработка»

по теме «Распределенное клиент-серверное приложение "Реверси" с использованием технологии servlets»

Студент А.А. Епифанцев

Руководитель О.А. Гордеева

Самара 2017

****ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА» (САМАРСКИЙ УНИВЕРСИТЕТ)»

Факультет Информатики   
Кафедра Программных систем

**ЗАДАНИЕ**

на курсовую работу по дисциплине

«Программная инженерия»

студентам группы № 6412 Б 300  
А.А. Епифанцеву

1. **Тема проекта:** «Автоматизированная система составления и разгадывания линейного кроссворда по выбранной теме»
2. **Исходные данные к проекту:** см. приложение к заданию
3. **Перечень вопросов, подлежащих разработке в курсовой работе:**
   1. Произвести анализ предметной области: изучить правила игры «Реверси» и различные тактики игры
   2. Разработать информационно-логический проект по методологии UML
   3. Разработать и реализовать программное и информационное обеспечение, провести его тестирование и отладку .
   4. Оформить документацию курсовой работы
   5. Подготовить презентацию по разработанной системе
4. **Перечень графических разработок**
   1. Структурная схема системы
   2. Диаграммы UML
   3. Схемы основных алгоритмов
5. **Календарный план выполнения работ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Содержание работы по этапам | Объем этапа в % к общему объему проекта | Срок  окончания | Фактическое выполнение |
| 1 | Оформление технического задания и его утверждение | 5 |  |  |
| 2 | Анализ и описание предметной области (1 раздел) | 10 |  |  |
| 3 | Проектирование системы (2 раздел) | 30 |  |  |
| 3.1 | Разработка структурной схемы системы | 5 |  |  |
| 3.2 | Разработка функциональной спецификации системы | 10 |  |  |
| 3.3 | Разработка информационно-логического проекта системы и его предъявление руководителю | 15 |  |  |
| 4 | Реализация проекта, разработка контрольных примеров. Предъявление реализации руководителю (3 раздел). | 45 |  |  |
| 5 | Корректировка проекта и оформление документации проекта. Защита проекта с представлением презентации. | 10 |  |  |

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_ /А.А. Епифанцев / \_\_\_\_\_\_\_\_\_\_\_

**ПРИЛОЖЕНИЕ  
к заданию на курсовую работу**студентам группы № 6412 Б 300   
А.А. Епифанцев

Тема проекта: **«Автоматизированная система составления и разгадывания линейного кроссворда по выбранной теме»**

**Исходные данные к проекту:**

1. **Характеристика объекта автоматизации:**

## объект автоматизации: игра «Реверси»;

## виды автоматизируемой деятельности:

* + процесс отрисовки игрового поля;
  + процесс игры (искусственный интеллект);

## размер поля по вертикали – 8 клеток;

## размер поля по горизонтали – 8 клеток;

1. **Требования к информационному обеспечению:**
2. обмен сообщениями между клиентом и сервером осуществляется по протоколу HTTP;
3. клиентской частью является браузер.
4. **Требования к техническому обеспечению:**
5. тип ЭВМ – IBM PC совместимый;
6. монитор с разрешающей способностью не ниже 800 х 600;
7. манипулятор – мышь;
8. веб браузер;
9. технические характеристики определяются в процессе выполнения проекта.
10. **Требования к программному обеспечению:**
11. тип операционной системы ‑ Windows 7/8, Windows Vista;
12. язык программирования – Java;
13. среда программирования –IntelliJ IDEA 2017;
14. среда проектирования – StarUML 5.0.

Руководитель   
курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / О.А. Гордеева /

Задание принял  
 к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ А.А. Епифанцев /

**СОДЕРЖАНИЕ**

[Введение 8](#_Toc480317159)

[1 Описание и анализ предметной области 9](#_Toc480317160)

[1.1 Описание предметной области 9](#_Toc480317161)

[1.2 Постановка задачи 10](#_Toc480317162)

[2 Проектирование системы 11](#_Toc480317163)

[2.1 Структурная схема системы 11](#_Toc480317164)

[2.2 Разработка прототипа интерфейса пользователя системы 12](#_Toc480317165)

[2.3 Информационно-логический проект системы 14](#_Toc480317166)

[2.3.1 Язык UML 14](#_Toc480317167)

[2.3.2 Диаграмма вариантов использования 14](#_Toc480317168)

[Сценарий «Вход в систему» 15](#_Toc480317169)

[2.3.3 Диаграмма классов 16](#_Toc480317170)

[2.3.4 Диаграмма состояний 16](#_Toc480317171)

[2.3.5 Диаграмма деятельности 17](#_Toc480317172)

[2.3.6 Диаграмма последовательности 18](#_Toc480317173)

[2.3.7 Логическая модель данных (при необходимости) 18](#_Toc480317174)

[2.4 Выбор и обоснование алгоритмов обработки данных /Разработка и описание алгоритмов обработки данных 21](#_Toc480317175)

[2.5 Выбор и обоснование комплекса программных средств 21](#_Toc480317176)

[2.5.1 Выбор языка программирования и среды разработки 21](#_Toc480317177)

[2.5.2 Выбор операционной системы 21](#_Toc480317178)

[2.5.3 Выбор среды программирования 21](#_Toc480317179)

[2.5.4 Выбор системы управления базами данных (при необходимости) 22](#_Toc480317180)

[3 Реализация системы 23](#_Toc480317181)

[3.1 Разработка и описание интерфейса пользователя 23](#_Toc480317182)

[3.1.1 Разработка и описание пользовательского меню 23](#_Toc480317183)

[3.1.2 Описание тестового примера 23](#_Toc480317184)

[3.2 Физическая модель данных (при необходимости) 23](#_Toc480317185)

[3.3 Реализация классов и структур данных 24](#_Toc480317186)

[3.4 Диаграммы реализации 24](#_Toc480317187)

[3.4.1 Диаграмма компонентов 24](#_Toc480317188)

[3.4.2 Диаграмма развертывания 24](#_Toc480317189)

[3.5 Выбор и обоснование комплекса технических средств 25](#_Toc480317190)

[3.5.1 Расчет объема занимаемой памяти 25](#_Toc480317191)

[Расчет объема внешней памяти 25](#_Toc480317192)

[Расчет объема ОЗУ 27](#_Toc480317193)

[3.5.2 Минимальные требования, предъявляемые к системе 27](#_Toc480317194)

[Заключение 29](#_Toc480317195)

[Список использованных источников 30](#_Toc480317196)

[Приложение А Руководство пользователя 31](#_Toc480317197)

[А.1 Назначение системы 31](#_Toc480317198)

[А.2 Условия работы системы 31](#_Toc480317199)

[А.3 Установка системы 31](#_Toc480317200)

[А.4 Работа с системой 31](#_Toc480317201)

[А.4.1 Работа с системой в режиме администратора (если необходимо) 32](#_Toc480317202)

[Вход в систему (авторизация) 32](#_Toc480317203)

[А.4.2 Работа с системой в режиме пользователя 32](#_Toc480317204)

[Вход в систему (авторизация) 32](#_Toc480317205)

[Вход в систему (регистрация) 32](#_Toc480317206)

[Настройка параметров кроссворда 32](#_Toc480317207)

[Приложение Б Листинг модулей программы 33](#_Toc480317208)

[приложение в Стили для оформления отчета и записки 34](#_Toc480317209)

[раздел - 1 уровень 34](#_Toc480317210)

[3.6 Подраздел (2 уровень) 35](#_Toc480317211)

[3.6.1 Параграф (3 уровень) 35](#_Toc480317212)

[Подпараграф (4 уровень) 36](#_Toc480317213)

Введение

Реверси – настольная игра, рассчитанная на двух игроков. Игровое поле представляет из себя доску размером 8 на 8 клеток. Один из игроков играет за черные фишки, а другой за белые. Цель игрока максимизировать количество его фишек на поле.

Игра была изобретена в Великобритании в 1880 году и пользовалась большой популярностью, но впоследствии была забыта. Возродили её в Японии, где она в 1971 году под названием отелло вновь стала популярна. С 1977 года регулярно проводятся чемпионаты мира по игре в реверси[1].

Реверси является стратегической игрой, схожей с шашками и шахматами. Так же как и в шахматах, принято разделять партию на три части: дебют (начало), миттельшпиль (середина игры) и эндшпиль (концовка). Однако, в отличие от шахмат, количество возможных дебютов здесь намного меньше, и все они легко запоминаются. Все сколько-либо серьёзные игроки знают дебюты на 5-6 ходов вперёд, чтобы избежать заведомо проигрышных ходов на данной стадии. Миттельшпиль, пожалуй, является наиболее «свободной» и одновременно сложной частью игры, когда положение можно либо упрочить, либо изменить в свою пользу. Несмотря на это, многие, казалось бы, проигранные в миттельшпиле партии обретают новые качества при вступлении в конечную стадию игры — эндшпиль. Золотое правило концовки — не спешить и считать. Считать принято фишки, которые результируют конечный исход игры для конкретной тактики. Естественно, количество исходов зависит от того, с какого хода начинать считать, и именно поэтому компьютеры играют намного лучше людей — они могут позволить себе просчитать все возможные варианты (их, по компьютерным меркам, немного) и всегда выбирают тот, при котором минимизируется результат человека и максимизируются очки компьютера[1].

1. Описание и анализ предметной области
   1. Описание предметной области

В игре используется квадратная доска размером 8х8 (все клетки могут быть одного цвета) и 64 специальные фишки, окрашенные с разных сторон в контрастные цвета, например, в белый и чёрный. Клетки доски нумеруются от верхнего левого угла: вертикали - латинскими буквами, горизонтали - цифрами. Один из игроков играет белыми, другой - чёрными. Внешний вид игрового поля приведен на рисунке 1.

  
Рисунок 1 –поле для игры в реверси

В начале игры в центр доски выставляются 4 фишки: чёрные на d5 и e4, белые на d4 и e5. Первый ход делают чёрные. Далее игроки ходят по очереди.

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.

* 1. Постановка задачи

В результате курсового проектирование будет разработано веб приложение реализующее игру «Реверси». Будет разработан пользовательский интерфейс, который будет отображаться в браузере. Пользовательский интерфейс будет реализован с помощью языка разметки html.

Серверная часть, отвечающая за игровую логику и искусственный интеллект, будет реализована на языке Java. Серверная часть будет реализована с помощью технологии servlet.

Обмен данными между клиентом и сервером будет производится по протоколу HTTP.

Таким образом, система должна решать следующие задачи:

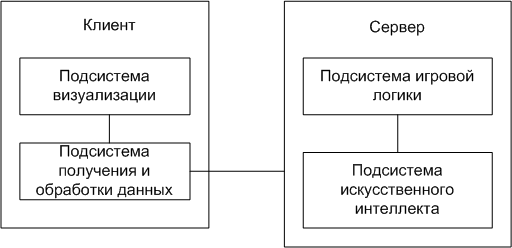
1. отрисовка пользовательского интерфейса;
2. обмен данными между клиентом и сервером;
3. игра с компьютером;
4. обеспечение соблюдения правил игры;
5. Проектирование системы
   1. Структурная схема системы

На основании проведенного выше анализа предметной области формируются общие требования к функциям, которые должна выполнять система. Проектируемую систему разобьем на подсистемы и определим взаимосвязи между ними.

Система – это совокупность объектов, компонентов или элементов произвольной природы, образующих некоторую целостность. Структура системы – это устойчивая во времени совокупность взаимосвязей между ее элементами или компонентами. Структура системы предполагает вложенность элементов одной системы в другую.

Структурная схема системы строится на основе функционально-модульного подхода, основанного на принципе алгоритмической декомпозиции с выделением функциональных элементов и установлением строгого порядка выполняемых действий, т.е. в основе лежит иерархический подход с выделением вначале функциональных действий, затем независимых компонентов с дальнейшей их детализацией

Структурная схема системы игры реверси представлена на рисунке 2.

  
Рисунок 2 – структурная схема системы игры в реверси

В проектируемую систему входит шесть подсистем:

1. *Клиент*, в которого входят подсистемы:

*Подсистема визуализации*, которая отвечает за отображение пользовательского интерфейса;

*Подсистема получения и обработки данных*, которая отвечает за обмен данными между клиентом и сервером. А также обрабатывает полученные данные и передает их в подсистему визуализации.

1. *Сервер*, в которого входят:

*Подсистема игровой логики*, которая отвечает за соблюдение правил игры;

*Подсистема искусственного интеллекта*, позволяющая пользователю играть с компьютером

* 1. Информационно-логический проект системы
     1. Язык UML

Унифицированный язык моделирования UML (Unified Modeling Language) – это преемник того поколения методов объектно-ориентированного анализа и проектирования, которые появились в конце   
80-х и начале 90-х годов. Отсутствие единого стандарта и метода построения единой и цельной объектной модели системы мешало широкому распространению объектно-ориентированных методов, поэтому создатели трех наиболее распространенных методологий Г. Буч, Д. Рамбо и А. Якобсон объединили свои усилия под эгидой Rational Software Corporation для создания UML, который и стал стандартом языка объектного моделирования .

* + 1. Диаграмма вариантов использования

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки. Она описывает функциональное назначение системы.

Разработка диаграммы вариантов использования преследует цели:

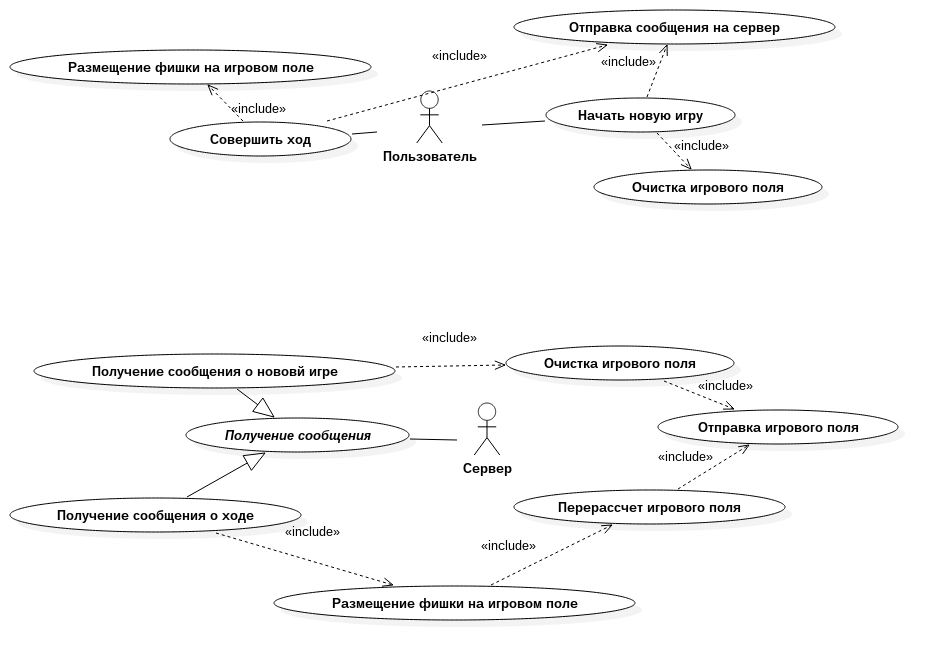
* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

В языке UML диаграммы вариантов использования позволяют визуализировать поведение системы, подсистемы или класса, чтобы пользователи могли понять как их использовать, а разработчики – реализовать соответствующий элемент.

Таким образом, суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне.

Вариант использования (use case) – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами.

Диаграмма вариантов использования разрабатываемой системы представлена на рисунке 3.

  
Рисунок 3 – диаграмма вариантов использования

Данная диаграмма иллюстрирует, как пользователь может взаимодействовать с системой, и как сервер будет реагировать на поведение пользователя.

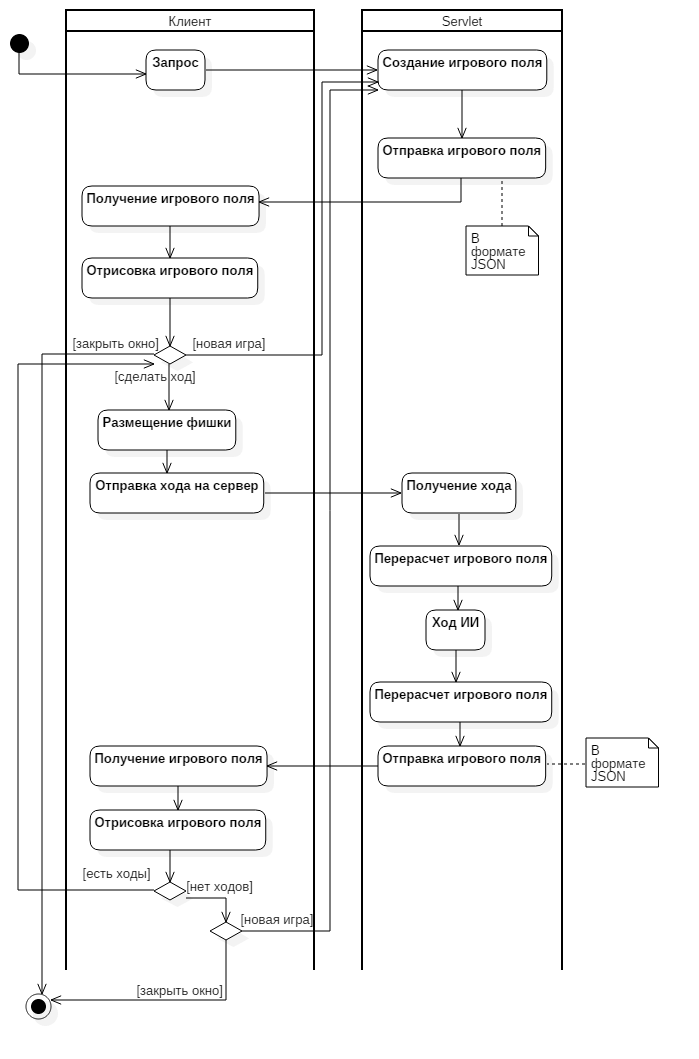
Пользователь может либо совершить ход, либо начать новую игру. В ответ на действия пользователя сервер реагирует разными способами. При совершении хода сервер перерасчитывает игровое поле и вычисляет следующий ход компьютера. При выборе новой игры игровое поле очищается.

* + 1. Диаграмма деятельности

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций.

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на этих диаграммах также присутствуют обозначения состояний и переходов. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние выполняется только при завершении этой операции.

Таким образом, диаграммы деятельности можно считать частным случаем диаграмм состояний. Они позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних деятельностей и действий. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности системы представлена на рисунке 4.

  
Рисунок 4 – диаграмма деятельности

* 1. Разработка и описание алгоритмов обработки данных

Для реализации искусственного интеллекта был выбран алгоритм Альфа-бета-отсечение, поскольку он прост в реализации и полностью удовлетворяет требованиям.

Альфа-бета-отсечение (англ. *alpha-beta pruning*) – алгоритм поиска, стремящийся сократить количество узлов, оцениваемых в дереве поиска алгоритмом минимакса. Предназначен для антагонистических игр и используется для машинной игры (в компьютерных шахматах, компьютерном го и других). В основе алгоритма лежит идея, что оценивание ветви дерева поиска может быть досрочно прекращено (без вычисления всех значений оценивающей функции), если было найдено, что для этой ветви значение оценивающей функции в любом случае хуже, чем вычисленное для предыдущей ветви. Альфа-бета-отсечение является оптимизацией, так как результаты работы оптимизируемого алгоритма не изменяются. Выбор и обоснование комплекса программных средств

* 1. Выбор и обоснование комплекса программных средств

Для реализации приложения была выбрана технология servlet.

Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.

Также в данном приложении используется технология AJAX. Он удобен тем, что для получения ответа от сервера не нужно перезагружать страницу, что позволяет сделать пользовательский интерфейс более удобным и ускорить работу приложения.

Заключение

Во время курсового проектирования было разработано веб приложение «Игра реверси», позволяющее пользователю играть в реверси с компьютером с помощью веб браузера.

Приложение А  
Листинг программы

***RevesiBoard***

package edu.ssau.revercy;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.IOException;

import java.io.OutputStream;

/\*\*

\* Created by EAA on 15.04.2017.

\*/

public class ReversiBoard {

public int getBoardSize() {

return boardSize;

}

public void setBoardSize(int boardSize) {

this.boardSize = boardSize;

}

public char[][] getBoard() {

return board;

}

public void setBoard(char[][] board) {

this.board = board;

}

private int boardSize;

private char board[][];

public ReversiBoard(int size){

boardSize = size;

board = new char[size][size];

for(int i = 0; i < size; i++){

for(int j = 0; j < size; j++){

if(((i == size/2) || (i == size/2 - 1)) &&

((j == size/2) || (j == size/2 - 1))){

if(i == j){

board[i][j] = 'L';

}else{

board[i][j] = 'D';

}

}else{

board[i][j] = ' ';

}

}

}//end loop

}

public ReversiBoard(ReversiBoard oldBoard){

boardSize = oldBoard.boardSize;

board = new char[oldBoard.boardSize][oldBoard.boardSize];

for(int i = 0; i < oldBoard.boardSize; i++){

for(int j = 0; j < oldBoard.boardSize; j++){

board[i][j] = oldBoard.board[i][j];

}

}

}

double getFillFactor(){

int filledSlots = 0;

for(int i = 0; i < boardSize; i ++){

for(int j = 0; j < boardSize; j++){

if(board[i][j] != ' ')

filledSlots ++;

}

}

return (double)filledSlots/(double)(boardSize \* boardSize);

}

int getSpotsLeft(){

int emptySpots = 0;

for(int i = 0; i < boardSize; i ++){

for(int j = 0; j < boardSize; j++){

if(board[i][j] == ' ')

emptySpots++;

}

}

return emptySpots;

}

void printScore(){

System.out.println("Score: Light " + getScore('L') + " - Dark " + getScore('D'));

}

public boolean playMove(char playerColor, int i, int j){

if(isValid(playerColor, i, j) == false){

// System.out.println("Invalid Move");

return false;

}

board[i][j] = playerColor;

for(int x = -1; x <= 1; x++){//in each direction

for(int y = -1; y <= 1; y++){

if(y!= 0 || x!= 0 ){

if(checkDirection(playerColor, i, j, x, y)){

int tempI = i + x;

int tempJ = j + y;

while(board[tempI][tempJ] == (char)(144 - (int)playerColor)){

board[tempI][tempJ] = playerColor;

tempI += x;

tempJ += y;

}

}

}

}

}

return true;

}

public boolean isValid(char playerColor, int i, int j){

//check every direction for a valid reason to accept move

if(board[i][j] != ' ')

return false;

for(int x = -1; x <= 1; x++){

for(int y = -1; y <= 1; y++){

if(y!= 0 || x!= 0 ){

if(checkDirection(playerColor, i, j, x, y)){

return true;

}

}

}

}

return false;

}

public boolean checkDirection(char playerColor, int i, int j, int iMod, int jMod){

int currentI = i + iMod;

int currentJ = j + jMod;

//returns false if checking off edge of board

if(currentI < 0 || currentI > boardSize - 1 || currentJ < 0 || currentJ > boardSize - 1)

return false;

//returns false if first piece in that direction is not the opposite color

if(board[currentI][currentJ] != (char)(144 - (int)(playerColor))) //ascii hacking to check if opposite color

return false;

//adjust currentI and currentJ to check first square after opposite color

currentI += iMod;

currentJ += jMod;

while(currentI > -1 && currentI < boardSize && currentJ > -1 && currentJ < boardSize){

if(board[currentI][currentJ] == playerColor)

return true;

if(board[currentI][currentJ] == ' ')

return false;

currentI += iMod;

currentJ += jMod;

}

return false;

}

public void printBoard(OutputStream out){

String ANSI\_RESET = "";

String ANSI\_BLUE = "";

String ANSI\_RED = "";

String breakLine = "&nbsp;&nbsp;&nbsp;+";

for(int i = 0; i < boardSize; i++){

breakLine += "---+";

}

String line = "";

line += "&nbsp;&nbsp;&nbsp;&nbsp;";

for(int i = 0; i < boardSize; i++){

char letter = (char) (i + 97);

line+= "&nbsp;&nbsp;" + letter + " ";

}

try {

out.write((line + "<br>").getBytes());

out.write((breakLine + "<br>").getBytes());

} catch (IOException e) {

e.printStackTrace();

}

for(int j = 0; j < boardSize; j++){

line = "&nbsp;" + (j+1);

if(j + 1 < 10)

line += "&nbsp;|";

else

line += "|";

for(int i = 0; i < boardSize; i++){

char printChar = board[i][j];

if(printChar == 'L'){

line += "&nbsp;&nbsp;" + ANSI\_RED + "L" + ANSI\_RESET + "&nbsp;|";

}else if(printChar == 'D'){

line += "&nbsp;" + ANSI\_BLUE + "D" + ANSI\_RESET + "&nbsp;|";

}else{

line += "&nbsp;&nbsp;&nbsp;|";

}

}

try {

out.write((line + "<br>").getBytes());

out.write((breakLine + "<br>").getBytes());

} catch (IOException e) {

e.printStackTrace();

}

}

}

public JSONObject boardToJSON() throws JSONException {

//0 - свободное поле, но в него нельзя поставить фишку

//1 - поле, в которое игрок может поставить свою фишку

//2 - фишка игрока

//3 - фишка компьютера

JSONObject json = new JSONObject();

Integer[][] boardJSON = new Integer[boardSize][boardSize];

for (int i = 0; i < boardSize; i++){

for (int j = 0; j < boardSize; j++){

if(board[i][j] == ' ' && isValid('L', i, j)){

boardJSON[i][j] = new Integer(1);

}

else if(board[i][j] == ' '){

boardJSON[i][j] = new Integer(0);

}

else if(board[i][j] == 'L'){

boardJSON[i][j] = new Integer(2);

}

else if(board[i][j] == 'D'){

boardJSON[i][j] = new Integer(3);

}

}

}

json.put("board", boardJSON);

json.put("playerScore",getScore('L'));

json.put("computerScore",getScore('D'));

return json;

}

boolean isFull(){

for(int i = 0; i < boardSize; i++){

for(int j = 0; j < boardSize; j++){

if(board[i][j] == ' '){

return false;

}

}

}

return true;

}

boolean isOneColor(){

char color = ' ';

for(int i = 0; i < boardSize; i++){

for(int j = 0; j < boardSize; j++){

if(board[i][j] != ' ' && color != ' ' && board[i][j] != color){

return false;

}

if(color == ' ')

color = board[i][j];

}

}

return true;

}

char getWinner(){

if(getScore('L') > getScore('D'))

return 'L';

if(getScore('D') > getScore('L'))

return 'D';

else

return 'T';

}

boolean hasMove(char playerColor){

for(int i = 0; i < boardSize; i++){

for(int j = 0; j < boardSize; j++){

if(isValid(playerColor, i, j))

return true;

}

}

return false;

}

int getScore(char playerColor){

int result = 0;

for(int i = 0; i < boardSize; i++){

for(int j = 0; j < boardSize; j++){

if(board[i][j] == playerColor)

result++;

}

}

return result;

}

int calculateScore(char playerColor){

double result = getScore(playerColor) - getScore((char)(144 - (int) playerColor));

if(board[0][0] == playerColor){

result += 3;

for(int i = 1; i < boardSize - 1; i++){

if(board[0][i] == playerColor)

result += 1;

else

i = boardSize + 1;

}

for(int i = 1; i < boardSize - 1; i++){

if(board[i][0] == playerColor)

result += 1;

else

i = boardSize + 1;

}

}

if(board[0][boardSize - 1] == playerColor){

result += 3;

for(int i = boardSize - 2; i > 0; i--){

if(board[0][i] == playerColor)

result += 1;

else

i = -1;

}

for(int i = 1; i < boardSize - 1; i++){

if(board[i][boardSize - 1] == playerColor)

result += 1;

else

i = boardSize + 1;

}

}

if(board[boardSize - 1][0] == playerColor){

result += 3;

for(int i = 1; i < boardSize - 1; i++){

if(board[boardSize - 1][i] == playerColor)

result += 1;

else

i = boardSize + 1;

}

for(int i = boardSize - 2; i > 0; i--){

if(board[i][0] == playerColor)

result += 1;

else

i = -1;

}

}

if(board[boardSize - 1][boardSize - 1] == playerColor){

result += 3;

for(int i = boardSize - 2; i > 0; i--){

if(board[boardSize - 1][i] == playerColor)

result += 1;

else

i = -1;

}

for(int i = boardSize - 2; i > 0; i--){

if(board[i][boardSize - 1] == playerColor)

result += 1;

else

i = -1;

}

}

return (int)result;

}

}

***RevesiPlayer***

package edu.ssau.revercy;

/\*\*

\* Created by EAA on 15.04.2017.

\*/

public class ReversiPlayer {

public static String playMove(ReversiBoard board, char playerColor){

if(board.hasMove(playerColor) == false)

return "no move to play";

int[] result = new int[3];

/\*if(board.getSpotsLeft() > 8 && board.getFillFactor() < .6){

result = minimaxWithAB(board, playerColor, 5, true, playerColor, -10000, 10000);

}else{ //run a slightly different algorithm for endgame

\*/

result = minimax(board, playerColor, 1, true, playerColor);

// }

int iVal = result[1];

int jVal = result[2];

board.playMove(playerColor, iVal, jVal);

String playedMove = "";

playedMove += (char)(iVal + 97);

playedMove += (jVal + 1);

return playedMove;

}

public static int[] minimaxWithAB(ReversiBoard board, char playerColor, int depth, boolean maximizingPlayer, char myColor, int alpha, int beta){

if(depth == 0 || (board.hasMove(playerColor) == false && board.hasMove((char)(144 - (int)playerColor)) == false) || board.isFull() == true){

int[] result = new int[3];

if((board.hasMove(playerColor) == false && board.hasMove((char)(144 - (int)playerColor)) == false) || board.isFull() == true){

if(board.getScore(myColor) == board.getScore((char)(144 - (int)myColor))){

result[0] = 0;

}else{

result[0] = (board.getScore(myColor) > board.getScore((char)(144 - (int)myColor)) ? 10000 : -10000);

}

result[1] = -1;

result[2] = -1;

return result;

}

result[0] = board.calculateScore(myColor);

result[1] = -1;

result[2] = -1;

return result;

}

if(maximizingPlayer == true){

int bestVal = -999999999;

int iVal = -1;

int jVal = -1;

boolean hasMoves = false;

for(int i = 0; i < board.getBoardSize(); i++){

for(int j = 0; j < board.getBoardSize(); j++){

if(board.isValid(playerColor, i, j)){

hasMoves = true;

System.out.println("maximizing with move: " + i + " " + j);

ReversiBoard tempBoard = new ReversiBoard(board);

tempBoard.playMove(playerColor, i, j);

int result[] = minimaxWithAB(tempBoard, (char)(144-(int)playerColor), depth - 1, false, myColor, alpha, beta);

if(result[0] > alpha){

alpha = result[0];

if(result[0] > bestVal){

iVal = i;

jVal = j;

}

}

if(alpha >= beta){

result[0] = alpha;

return result;

}

}

}

}

if(hasMoves == false){

return minimaxWithAB(board, (char)(144 - (int)playerColor), depth - 1, false, myColor, alpha, beta);

}

int returnArr[] = new int[3];

returnArr[0] = alpha;

returnArr[1] = iVal;

returnArr[2] = jVal;

return returnArr;

}else{ /\*minimizing player\*/

int bestVal = 9999999;

int iVal = -1;

int jVal = -1;

boolean hasMoves = false;

for(int i = 0; i < board.getBoardSize(); i++){

for(int j = 0; j < board.getBoardSize(); j++){

if(board.isValid(playerColor, i, j)){

System.out.println("minimizing with move: " + i + " " + j);

hasMoves = true;

ReversiBoard tempBoard = new ReversiBoard(board);

tempBoard.playMove(playerColor, i, j);

int result[] = minimaxWithAB(tempBoard, (char)(144-(int)playerColor), depth - 1, true, myColor, alpha, beta);

if(result[0] < beta){

beta = result[0];

if(result[0] < bestVal){

iVal = i;

jVal = j;

}

}

if (beta <= alpha){

result[0] = beta;

System.out.println("" + i + " " + j);

return result;

}

}

}

}

if(hasMoves == false){

System.out.println("heeerrrreeE");

return minimaxWithAB(board, (char)(144 - (int)playerColor), depth - 1, true, myColor, alpha, beta);

}

int returnArr[] = new int[3];

returnArr[0] = beta;

returnArr[1] = iVal;

returnArr[2] = jVal;

System.out.println("LOOK: " + returnArr[1] + " " + returnArr[2] );

return returnArr;

}

}

public static int[] minimax(ReversiBoard board, char playerColor, int depth, boolean maximizingPlayer, char myColor){

if(depth == 0 || (board.hasMove(playerColor) == false && board.hasMove((char)(144 - (int)playerColor)) == false) || board.isFull() == true){

int[] result = new int[3];

if((board.hasMove(playerColor) == false && board.hasMove((char)(144 - (int)playerColor)) == false) || board.isFull() == true){

if(board.getScore(myColor) == board.getScore((char)(144 - (int)myColor))){

result[0] = 0;

}else{

result[0] = (board.getScore(myColor) > board.getScore((char)(144 - (int)myColor)) ? 10000 : -10000);

}

result[1] = -1;

result[2] = -1;

return result;

}

result[0] = board.calculateScore(myColor);

result[1] = -1;

result[2] = -1;

return result;

}

if(maximizingPlayer == true){

int bestVal = -999999999;

int iVal = -1;

int jVal = -1;

boolean hasMoves = false;

for(int i = 0; i < board.getBoardSize(); i++){

for(int j = 0; j < board.getBoardSize(); j++){

if(board.isValid(playerColor, i, j)){

hasMoves = true;

ReversiBoard tempBoard = new ReversiBoard(board);

tempBoard.playMove(playerColor, i, j);

int result[] = minimax(tempBoard, (char)(144-(int)playerColor), depth - 1, false, myColor);

if(result[0] > bestVal){

bestVal = result[0];

iVal = i;

jVal = j;

}

}

}

}

if(hasMoves == false){

return minimax(board, (char)(144 - (int)playerColor), depth - 1, false, myColor);

}

int returnArr[] = new int[3];

returnArr[0] = bestVal;

returnArr[1] = iVal;

returnArr[2] = jVal;

return returnArr;

}else{ //minimizing player

int bestVal = 9999999;

int iVal = -1;

int jVal = -1;

boolean hasMoves = false;

for(int i = 0; i < board.getBoardSize(); i++){

for(int j = 0; j < board.getBoardSize(); j++){

if(board.isValid(playerColor, i, j)){

hasMoves = true;

ReversiBoard tempBoard = new ReversiBoard(board);

tempBoard.playMove(playerColor, i, j);

int result[] = minimax(tempBoard, (char)(144-(int)playerColor), depth - 1, true, myColor);

if(result[0] < bestVal){

bestVal = result[0];

iVal = i;

jVal = j;

}

}

}

}

if(hasMoves == false){

return minimax(board, (char)(144 - (int)playerColor), depth - 1, true, myColor);

}

int returnArr[] = new int[3];

returnArr[0] = bestVal;

returnArr[1] = iVal;

returnArr[2] = jVal;

return returnArr;

}

}

public static String playRandomMove(ReversiBoard board, char playerColor){

for(int i = 0; i < board.getBoardSize(); i++){

for(int j = 0; j < board.getBoardSize(); j++){

if(board.isValid(playerColor, i, j)){

board.playMove(playerColor, i, j);

String playedMove = "";

playedMove += (char)(i + 97);

playedMove += (j+1);

return playedMove;

}

}

}

return "cannot play a move";

}

}

***ReversiServlet***

package edu.ssau.servlet;

import edu.ssau.revercy.ReversiBoard;

import edu.ssau.revercy.ReversiPlayer;

import org.json.JSONException;

import org.json.JSONObject;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.io.PrintWriter;

/\*\*

\* Created by EAA on 15.04.2017.

\*/

public class ReversiServlet extends HttpServlet{

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

resp.setContentType("application/json;charset=utf-8");

HttpSession session = req.getSession();

JSONObject object = new JSONObject();

char human = 'L';

char computer = 'D';

ReversiBoard board = (ReversiBoard)session.getAttribute("reversiboard");

if(board == null){

board = new ReversiBoard(8);

session.setAttribute("reversiboard", board);

}

else {

String newGame = req.getParameter("newGame");

if (newGame == null) {

String x = req.getParameter("x");

String y = req.getParameter("y");

if (x != null && y != null) {

int i = Integer.valueOf(x);

int j = Integer.valueOf(y);

if (board.isValid(human, i, j)) {

board.playMove(human, i, j);

}

try {

object.put("playerMove", board.boardToJSON());

} catch (JSONException e) {

e.printStackTrace();

}

ReversiPlayer.playMove(board, computer);

}

}

else if(newGame.equals("true")){

board = new ReversiBoard(8);

session.removeAttribute("reversiboard");

session.setAttribute("reversiboard", board);

}

}

PrintWriter pw = resp.getWriter();

try {

object.put("computerMove", board.boardToJSON());

pw.write(object.toString());

} catch (JSONException e) {

e.printStackTrace();

}

}

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req, resp);

}

}

***Index.jsp***

<%--

Created by IntelliJ IDEA.

User: EAA

Date: 15.04.2017

Time: 1:00

To change this template use File | Settings | File Templates.

--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>$Title$</title>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>

<script src="res/js/ajax.js"></script>

<link rel="stylesheet" href="res/style.css">

</head>

<body>

<div class="header">

<div class="left">

Игрок:<span id="player"></span>

</div>

<div class="right">

<span id="computer"></span>:Компьютер

</div>

<center style="margin: 0px 120px 0px 120px;">

<input type="button" id="newGame" value="Новая игра"/>

</center>

</div>

<div class="wraper">

<table class="table">

<tr>

<td id="cell\_11" class="cell" onclick="sendMove(0, 0)" onclick="sendMove(0, 0)"></td>

<td id="cell\_12" class="cell" onclick="sendMove(0, 1)"></td>

<td id="cell\_13" class="cell" onclick="sendMove(0, 2)"></td>

<td id="cell\_14" class="cell" onclick="sendMove(0, 3)"></td>

<td id="cell\_15" class="cell" onclick="sendMove(0, 4)"></td>

<td id="cell\_16" class="cell" onclick="sendMove(0, 5)"></td>

<td id="cell\_17" class="cell" onclick="sendMove(0, 6)"></td>

<td id="cell\_18" class="cell" onclick="sendMove(0, 7)"></td>

</tr>

<tr>

<td id="cell\_21" class="cell" onclick="sendMove(1, 0)"></td>

<td id="cell\_22" class="cell" onclick="sendMove(1, 1)"></td>

<td id="cell\_23" class="cell" onclick="sendMove(1, 2)"></td>

<td id="cell\_24" class="cell" onclick="sendMove(1, 3)"></td>

<td id="cell\_25" class="cell" onclick="sendMove(1, 4)"></td>

<td id="cell\_26" class="cell" onclick="sendMove(1, 5)"></td>

<td id="cell\_27" class="cell" onclick="sendMove(1, 6)"></td>

<td id="cell\_28" class="cell" onclick="sendMove(1, 7)"></td>

</tr>

<tr>

<td id="cell\_31" class="cell" onclick="sendMove(2, 0)"></td>

<td id="cell\_32" class="cell" onclick="sendMove(2, 1)"></td>

<td id="cell\_33" class="cell" onclick="sendMove(2, 2)"></td>

<td id="cell\_34" class="cell" onclick="sendMove(2, 3)"></td>

<td id="cell\_35" class="cell" onclick="sendMove(2, 4)"></td>

<td id="cell\_36" class="cell" onclick="sendMove(2, 5)"></td>

<td id="cell\_37" class="cell" onclick="sendMove(2, 6)"></td>

<td id="cell\_38" class="cell" onclick="sendMove(2, 7)"></td>

</tr>

<tr>

<td id="cell\_41" class="cell" onclick="sendMove(3, 0)"></td>

<td id="cell\_42" class="cell" onclick="sendMove(3, 1)"></td>

<td id="cell\_43" class="cell" onclick="sendMove(3, 2)"></td>

<td id="cell\_44" class="cell" onclick="sendMove(3, 3)"></td>

<td id="cell\_45" class="cell" onclick="sendMove(3, 4)"></td>

<td id="cell\_46" class="cell" onclick="sendMove(3, 5)"></td>

<td id="cell\_47" class="cell" onclick="sendMove(3, 6)"></td>

<td id="cell\_48" class="cell" onclick="sendMove(3, 7)"></td>

</tr>

<tr>

<td id="cell\_51" class="cell" onclick="sendMove(4, 0)"></td>

<td id="cell\_52" class="cell" onclick="sendMove(4, 1)"></td>

<td id="cell\_53" class="cell" onclick="sendMove(4, 2)"></td>

<td id="cell\_54" class="cell" onclick="sendMove(4, 3)"></td>

<td id="cell\_55" class="cell" onclick="sendMove(4, 4)"></td>

<td id="cell\_56" class="cell" onclick="sendMove(4, 5)"></td>

<td id="cell\_57" class="cell" onclick="sendMove(4, 6)"></td>

<td id="cell\_58" class="cell" onclick="sendMove(4, 7)"></td>

</tr>

<tr>

<td id="cell\_61" class="cell" onclick="sendMove(5, 0)"></td>

<td id="cell\_62" class="cell" onclick="sendMove(5, 1)"></td>

<td id="cell\_63" class="cell" onclick="sendMove(5, 2)"></td>

<td id="cell\_64" class="cell" onclick="sendMove(5, 3)"></td>

<td id="cell\_65" class="cell" onclick="sendMove(5, 4)"></td>

<td id="cell\_66" class="cell" onclick="sendMove(5, 5)"></td>

<td id="cell\_67" class="cell" onclick="sendMove(5, 6)"></td>

<td id="cell\_68" class="cell" onclick="sendMove(5, 7)"></td>

</tr>

<tr>

<td id="cell\_71" class="cell" onclick="sendMove(6, 0)"></td>

<td id="cell\_72" class="cell" onclick="sendMove(6, 1)"></td>

<td id="cell\_73" class="cell" onclick="sendMove(6, 2)"></td>

<td id="cell\_74" class="cell" onclick="sendMove(6, 3)"></td>

<td id="cell\_75" class="cell" onclick="sendMove(6, 4)"></td>

<td id="cell\_76" class="cell" onclick="sendMove(6, 5)"></td>

<td id="cell\_77" class="cell" onclick="sendMove(6, 6)"></td>

<td id="cell\_78" class="cell" onclick="sendMove(6, 7)"></td>

</tr>

<tr>

<td id="cell\_81" class="cell" onclick="sendMove(7, 0)"></td>

<td id="cell\_82" class="cell" onclick="sendMove(7, 1)"></td>

<td id="cell\_83" class="cell" onclick="sendMove(7, 2)"></td>

<td id="cell\_84" class="cell" onclick="sendMove(7, 3)"></td>

<td id="cell\_85" class="cell" onclick="sendMove(7, 4)"></td>

<td id="cell\_86" class="cell" onclick="sendMove(7, 5)"></td>

<td id="cell\_87" class="cell" onclick="sendMove(7, 6)"></td>

<td id="cell\_88" class="cell" onclick="sendMove(7, 7)"></td>

</tr>

</table>

</div>

</body>

</html>

***Reverse.js***

/\*\*

\* Created by EAA on 15.04.2017.

\*/

function sendMove(i, j) {

if($('#cell\_' + (i + 1) + (j + 1)).attr('class').indexOf('yellowCell') !== -1) {

var data = {"x": i, "y": j};

$('#cell\_' + (i + 1) + (j + 1)).removeClass('yellowCell');

$('#cell\_' + (i + 1) + (j + 1)).html("<div class='playerCell'>");

$.ajax({

url: "reversi",

type: 'get',

data: data,

dataType: 'json',

data: data,

success: function (result) {

var computerMove = result.computerMove;

var playerMove = result.playerMove;

renderBoard(playerMove, 'player');

var millisecondsToWait = 500;

setTimeout(function () {

renderBoard(computerMove, 'computer');

}, millisecondsToWait);

}

});

}

}

$(document).ready(function () {

$.ajax({

url: "reversi",

type: 'get',

dataType: 'json',

success: function (result) {

renderBoard(result.computerMove, 'computer');

}

});

$('#newGame').click(function () {

$('div.playerCell').remove();

$('div.computerCell').remove();

var data = {"newGame": true};

$.ajax({

url: "reversi",

data: data,

type: 'get',

dataType: 'json',

success: function (result) {

alert("Новая игра начинается!");

renderBoard(result.computerMove, 'computer');

}

});

})

});

function renderBoard(json, who) {

var board = json.board;

var validMove = 0;

for(var i = 0; i < board.length; i++){

for(var j = 0; j < board[0].length; j++){

if(board[i][j] === 0){

$('#cell\_'+(i+1)+(j+1)).removeClass('yellowCell');

}

if(board[i][j] === 1){

$('#cell\_'+(i+1)+(j+1)).addClass('yellowCell');

validMove++;

}

if(board[i][j] === 2){

$('#cell\_'+(i+1)+(j+1)).removeClass('yellowCell');

$('#cell\_' + (i + 1) + (j + 1)).html("<div class='playerCell'>");

}

if(board[i][j] === 3){

$('#cell\_'+(i+1)+(j+1)).removeClass('yellowCell');

$('#cell\_'+(i+1)+(j+1)).html("<div class='computerCell'>");

}

}

}

var playerScore = json.playerScore;

var computerScore = json.computerScore;

$('#player').html(playerScore);

$('#computer').html(computerScore);

if(validMove == 0 && who == 'computer'){

var message;

if(playerScore > computerScore){

message = "Вы выйграли";

}

else if(computerScore == playerScore){

message = "Ничья";

}

else{

message = "Вы проиграли";

}

alert(message);

}

}