Othello Project

Project Team:

Henry Dao

Devin Prejean

Dakota Jackson

Jairo

Sadra



Contents

[1.0 Introduction 3](#_Toc51874315)

[1.1 Glossary 3](#_Toc51874316)

[1.2 Competing Software 3](#_Toc51874317)

[1.3 Users 3](#_Toc51874318)

[1.4 Environment 3](#_Toc51874319)

[1.5 Similar Domains 3](#_Toc51874320)

[1.6 General Knowledge 3](#_Toc51874321)

[Setup 4](#_Toc51874322)

[How to play 4](#_Toc51874323)

[Ending the game 6](#_Toc51874324)

[2.0 Functional Requirements 6](#_Toc51874325)

[2.1 User Requirements 6](#_Toc51874326)

[2.2 Inputs 6](#_Toc51874327)

[2.3 Outputs 6](#_Toc51874328)

[2.4 Data Storage 6](#_Toc51874329)

[2.5 Computations 7](#_Toc51874330)

[3.0 Non-Functional Requirements 7](#_Toc51874331)

[3.1 Response Time 7](#_Toc51874332)

[3.2 Resource Usage 7](#_Toc51874333)

[3.3 Reliability 7](#_Toc51874334)

[3.4 Recovery from Failure 7](#_Toc51874335)

[3.5 Maintainability and Enhancements 7](#_Toc51874336)

[3.6 Computing Platform and Technology Used 7](#_Toc51874337)

[4.0 Use Case Diagrams 8](#_Toc51874338)

[4.1 Registration and Login 8](#_Toc51874339)

[4.1.1 Use Case 1: 8](#_Toc51874340)

[4.2.2 Use Case 2: 9](#_Toc51874341)

[4.2 Game Play 10](#_Toc51874342)

[4.2.1 Use Case 1 10](#_Toc51874343)

# 1.0 Introduction

The project team has been tasked with recreating the game Othello, using Java.

The game Othello is a strategy game that takes *a minute to learn and a lifetime to master.*™ Using two-sided game discs and your wits alone you battle it out with one other player. The goal is simple, have more discs on the board with your color face up than the other player, however getting that higher number may be easier said than done.

## 1.1 Glossary

The following terms will be used throughout this document:

|  |  |
| --- | --- |
| Term | Definition |
| Disc | The round object that is either black or white that is used by each player |
| Capture | Flipping over the disc to reveal the opposite color |
| Reversi | Precursor of Othello |
| Othello | The board game being recreated via this system |
| Java | The development language used to create this systems interface and logic |
| CSV | Comma Separated Value – A file type commonly used to store text values |

## 1.2 Competing Software

There are numerous online web-based versions of Othello available online and free of charge. These sites offer many features such as online gameplay against another human as well as local computer opponent. These sites will also keep track of the score for you during the game as well as track your wins and losses over time.

## 1.3 Users

The expected users of this system are people that want to learn to play or play against another person as well as those that want to refine their skills and strategy by playing against the computer.

## 1.4 Environment

This system will have two parts, the front-end will run on Java and be capable of launching on a Windows, Mac, or Linux computer. The database is a csv that records wins and losses of players

## 1.5 Similar Domains

Other domains you will find this game is physically via a board and physical discs as this is a recreation of the board game. Similarities include the game board, the pieces, and the rules for play. Differences include a timer as well as current and past scores during play.

## 1.6 General Knowledge

The game of Reversi was created in 1833. However, in 1971, a variant of Reversi, with a couple of changes such as the board’s initial setup, was patented. This patented version of Reversi came to be known as Othello. The game is incredibly popular in Japan, the home country where the game Reversi was initially created. The game is known for the saying “one minute to learn, but a lifetime to master”

### Setup

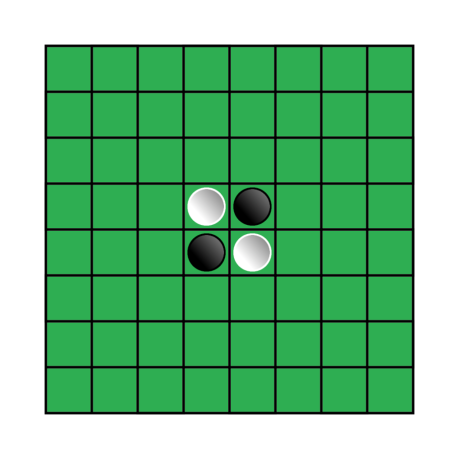
The setup includes one board with 64 squares and 64 game discs that have two sides, one black and one white. The game starts with two white pieces and two black discs in the center of the board diagonal to their own color like below.  


Figure 1 Othello Setup

### How to play

Starting with the player that is using the black sides you take turns capturing (turning over) each other’s discs. If you are unable to do that you skip your turn

#### Making a play

Each turn you must find an available spot adjacent to your opponent’s color to play your disc. This spot must “touch” the square your opponent’s disc is in, this can be horizontal, vertical, or diagonal. After placing your disc, you turn over your opponent’s discs that are in between the existing disc that was on the board and the new disc you have placed, thus “capturing” it. Sometimes this is one disc, sometimes it’s many discs that you capture in one move. If the field you place the disc in does not result in capturing the opponent’s disc(s) that is not a legal move. If you are unable to find a field where an opponent’ disc would be captured your turn is forfeited and play is passed to your opponent.

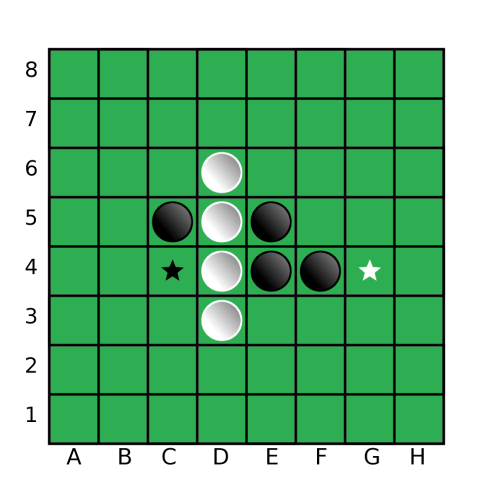


Figure 2 Make a move example for both sides

For example, in the graphic above if it is white’s turn, white places a disc in G4 capturing E4 and F4 in the process. If it was black’s turn black places a disc in C4 that would capture D4.

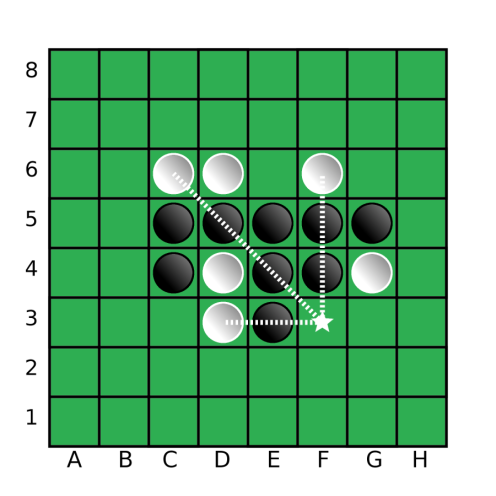


Figure 3Advance strategy for capturing multiple discs

For example, during white’s turn a disc is placed in F3. This move allows three straight lines to be created with many black discs in between. From this move white will capture D5, E4, E3, F4, and F5.

### Ending the game

This way of play continues until all the spaces on the board are filled or there are no moves for either player to make. Each player will then count the number of discs with their color, the highest wins.

# 2.0 Functional Requirements

Functional requirements involve the inputs, outputs, as well as the type of data storage.

## 2.1 User Requirements

Upon loading the game, the user will type in their username which will be 6 characters of letters and digits as well as password. They will then choose a color, white or black. The game will begin with Black going first.

Each player will take turns choosing a location to place their discs one at a time, each turn capturing an opponent’s disc changing its color.

Using the rules of Othello to decide on what discs are captured in any given move this will continue until the game is complete.

While the game is running a current score will be shown for each player at the top of the screen.

## 2.2 Inputs

The system shall receive input from the user regarding their 6 character username, password and choice of white or black discs

The system shall receive input from the user regarding where to place the next disc

The system shall receive input from the user clicking the quit button

The system shall receive input from the user clicking the pass button

## 2.3 Outputs

The system shall output the status of the game via a board with 64 squares with a green background and discs either white or black in color.

The system shall output the current and past scores of each player

The system shall show where the user placed their disc immediately after they clicked the square

The system shall output the time for each player next to their usernames

## 2.4 Data Storage

The system shall record the current and past scores of each player

The system shall make updates after every move by each player

The system shall record the date of game

The system shall record player usernames

## 2.5 Computations

The system will not allow users to select a square in which the placement of a disc does not capture an opponent’s disc.

The system will only allow users to select a legal move that is adjacent to the opponent’s discs

The system will capture all discs in between an existing disc and the disc just placed by the user

The system will not enable the corners to be captured once a disc is placed

The system will compute the score based on the sum of each color disc is currently showing

# 3.0 Non-Functional Requirements

Non-Functional requirements include system related restrictions such as response time, throughput, resource usage, reliability, and recovery from failure. Allowances for maintainability are also covered.

## 3.1 Response Time

The system shall respond to a user input within 100ms

## 3.2 Resource Usage

The system shall be no larger than 5MB after being compiled

The system shall use no more than 100MB of memory during execution

## 3.3 Reliability

The system shall have a minimum of 90% reliability, judged by running the system 10 times without more than 1 failure

## 3.4 Recovery from Failure

The system shall resume on the player’s turn that which it failed

The system shall resume the timers for each player

The system shall not lose any data if the data store is not corrupt or deleted

## 3.5 Maintainability and Enhancements

The system source code will be kept in source control for future maintenance

## 3.6 Computing Platform and Technology Used

The system shall run on any desktop with an operating system of Windows, Linux, or Mac

The system shall be developed with Java

The system shall be developed using the Waterfall methodology

# 4.0 Use Case Diagrams

Use Case Diagrams provided are to assist in visualizing the process and the expected interaction between users and the system.

## 4.1 Registration and Login

Each player can log in or Register as a new player. This System receives required Inputs as described in [section 2.2](#_2.2_Inputs).

A close up of text on a white background

Description automatically generated

Figure 4 Use Case for login/registration

### 4.1.1 Use Case 1:

**Name:** Register   
**Actor:** Player  
**Related to:**   
**Precondition:**

**Steps:**

|  |  |
| --- | --- |
| User: | System: |
| 1. Selects Register button | 1. Prompts username and password and nickname |
| 1. User fills both fields and clicks log in | 1. validates the cases and registers the user to the system. |

**Post-Condition:** If the use case was successful, the user would get registered to the system. Otherwise, state stays unchanged.

**Invalid:**

If the username or password is not valid, the system shows an error message.   
If both password fields are not the same, the system shows error.  
If username is already taken, the system shows error message.

### 4.2.2 Use Case 2:

**Name:** Login   
**Actor:** Player  
**Related to:**   
**Precondition:** user has already registered.

**Steps:**

|  |  |
| --- | --- |
| User: | System: |
| 1. Selects login | 1. Prompts username and password |
| 1. User enters both fields and clicks login | 1. System validates the cases and logs the user into the system and assigns a color |

**Post-Condition:**   
If the use case was successful, the player logs in. if not state stays unchanged.

**Invalid:**   
If the username or password is not valid, the system shows an error message.

## 4.2 Game Play

A close up of text on a white background

Description automatically generated

Figure 5 Use Case for Game Play

### 4.2.1 Use Case 1

**Name:** Play game   
**Actor:** player  
**Related to** extends quit, pass, place a disc.

**Precondition:** Two players are logged in, timer for player is not expired, game has begun

**Steps:**

|  |  |
| --- | --- |
| User: | System: |
|  | 1. System highlights the name of the player; system resumes the countdown timer |
| 1. Clicks on a section to place a disc, pushes the pass button or the quit button. | 3. a. if the selected section is invalid system stays in the same state and waits for another user action.  b. if the selected section is valid system places the disc and reverses other discs as outlined in the rules and switches turns.  c. if the pass button is pressed, the system switches turns.  d. if the quit button is pressed, system announces the other player as the winner. |

**Post-Condition:**If the user has pushed the quit button, the system announces the other user as the winner. Otherwise, system change turns to the other user.

If there is no valid move play is passed, and if the other player has no valid move the game ends as outlined in the rules.

**Invalid:**   
If the chosen section is already filled or placing a disc, there is against the rules of the game.