|  |  |
| --- | --- |
| **Course** | BSCH |
| **Stage / Year:** | 3 |
| **Module:** | HGP |
| **Assignment No:** | Repeat Project: Building a Working Game of Abalone |
| **Deadline:** | TBC |
| **Submission:** | Upload to Moodle |
| **Weighting:** | 100% of project (60% of module grade) |
| **Group size:** | 2 |

# 1. Introduction.

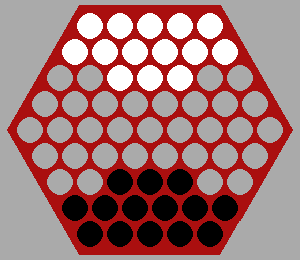
**N.B.:**

* **Please read the project description in full before you start coding. Failure to do so will result in you missing out on important information and reducing your ability to avail of marks. Please attempt to work in line with the file structure provided. Very large departures from this template may result in loss of marks.**
* In this project you will be tasked with building a fully working game of Abalone.



**stage**

**Cell**



**White Piece**

**Black Piece**

**Figure 1. Draughts User Interface – basic without clock or status bar (these must be included)**

# 2. Resources

* **Moodle**
  + This Description
* **From your lab work** 
  + Example20. The solution is heavily based on this and it is essential to have a detailed working knowledge of Example20 before proceeding with this project. You may like to begin this project using a working version of Example 20.
  + UltimateXOs. The skills you learned in the design of your UltimateXOs application will stand to you.

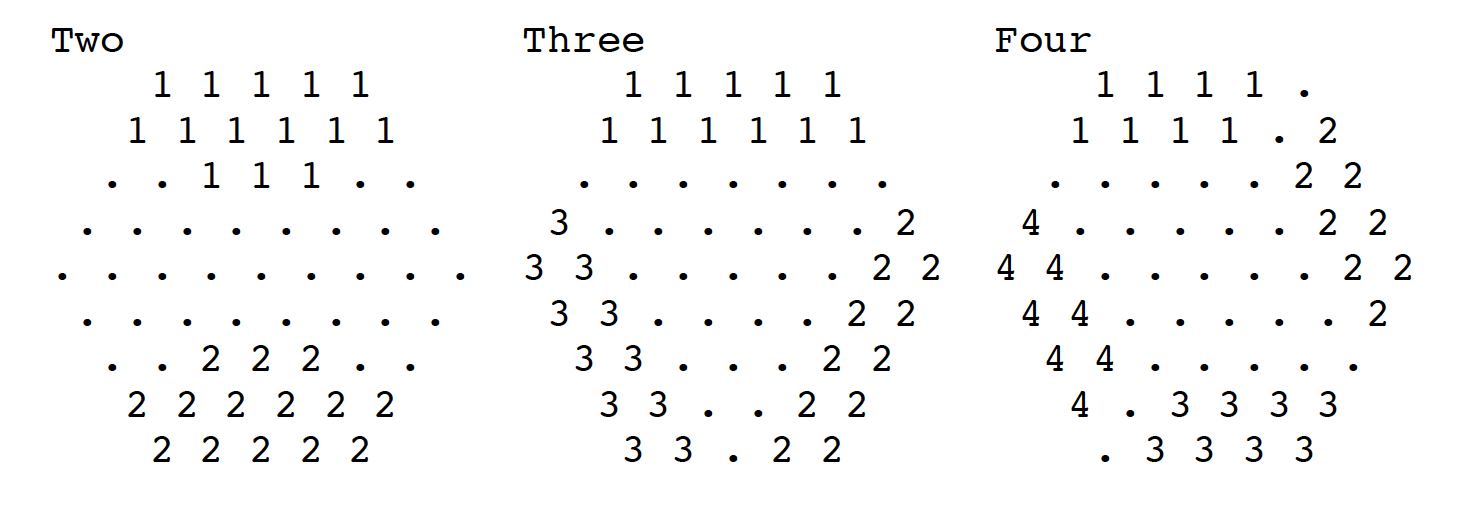
# 3. Explanation of the Game

## Object of the game:

On a hexagonal board (of side size 5) two to six players have armies of marbles. Players take turns "pushing" 1, 2 or 3 linearly connected marbles, attempting to push their opponents' marbles off the board. First player to push 6 of their opponent's marbles off the board wins. In the 3+ player version, the 6 marbles may be any combination of 6 opponents' marbles.

## Initial board layout:

Below initial board layouts for 2, 3 and 4 players.



Movement:

Black goes first.

You may move a group of 1, 2, or 3 adjacent marbles of your colour (a group of 3 marbles must be in a straight line) one space in any one of the six possible directions

A group of two or three marbles may push an opponent's smaller group when moving in a direction that the line is pointing (i.e., not when moving "sideways").

## Additional Resources

* Detailed rules of the game and some points regarding stagey are available at <http://www.gamerz.net/pbmserv/abalone.html>
* A short video showing the initial 2 player set up and the rules of the game are available here <https://www.youtube.com/watch?v=EqMlpFJVB7Q>
* Detailed discussion on the game is available here <https://en.wikipedia.org/wiki/Abalone_(board_game>)

# 4. Comparison between Abalone and Xs & Os

* Below is a table showing some of the files that you MUST have in your solution and their counterparts in Example020 with comparisons

|  |  |  |  |
| --- | --- | --- | --- |
| **File name in Project** | **Draughts** | **File name in  Example20** | **Xs & Os** |
| **AbaloneApplication.java** | * Has a custom control with a Abalone board. | **Example020.java** | * Has a custom control with an Xx & Os board. |
| **AbaloneBoard.java** | * Hexagonal grid of side 5. * Pieces can move to “valid” locations determined by a complex set of rules. Valid moves are defined in the Piece/GameLogic class. * Maps mouse click coordinates to hexagonal grid. | **XOBoard.java** | * 3 wide by 3 high rectangular grid * Grid division is highlighted with horizontal and vertical lines * Background is black * Maps mouse click coordinates to 3 x 3 grid. * Pieces can be placed if the cell is empty |
| **Cell.java** | * This is an element of the Hexagonal board which will have neighbours |  | * Not present |
| **Piece.java** | * N (2 to 6) distinct pieces each of which must be a different colour. | **XOPiece.java** | * 2 distinct pieces (X and O) |
| **CustomControl.java** | * Detects mouse clicks, drags etc. and keyboard actions | **CustomControl.java** | * Detects mouse clicks and keyboard actions |
| **CustomControlSkin.java** | * Identical | **CustomControlSkin.java** | * Identical |
| **GameLogic.java** | * Determines when a winner has occurred * Additional methods can feature here too. * You may decide to include some of your movement logic in this class as opposed to Piece.java |  | * Not present |
| **Additional Classes** | * To be added with justification |  | * Not present |

# 5. Working in Pairs

* Agree on a common IDE for code development. Converting from one project format to the other is messy at best.
* If you can use a SCM (source code manager) to develop and share code please do. I would recommend using Git if you can (particularly if you have access to a git server somewhere where all changes can be uploaded).
* Decide on a development strategy. Try to work on independent parts if you can. Working in parallel will cut the time in half.
* “Weeks of programming can save you hours of planning” a quote from Scott Meyers. Basically it means design the project first before implementing it. This means deciding on your data structures, UI design, view design etc. before you start. Pay particular attention to your data structure for the game board and how you implement the rules.

# 6. Tasks, Deliverables & Marking

* **Deliverables**
  + The full list of deliverables is

1. This document with the following sections completed
   1. Summary of Division Of Work Word
   2. Screen Shots of Working Application
2. Folder containing all the java source files, images etc. relevant to your application
   * You are **both** required to required to submit a single **identical** archive (zip) file to Moodle that has a filename of **<lastname1>\_<firstname1>\_<studentnumber1>\_<lastname2>\_<firstname2>\_<studentnumber2>\_abalone.zip** If the files are not identical (MD5 hash etc.) then you will be called to interview.
   * In that zip file I should find a single folder containing all your deliverables and it should be named **<lastname1>\_<firstname1>\_<studentnumber1>\_<lastname2>\_<firstname2>\_<studentnumber2>\_abalone**

* All work is expected to BE YOUR OWN OR YOUR PARTNERS. **PLAGIARISM** will not be tolerated under any circumstances.

|  |  |  |
| --- | --- | --- |
| Task Number | Description | % of Grade |
| 1 | Generate an application that is forced to be square in size. It should display a full Abalone board of side size 5 with the initial state of the game. | 20 |
| 2 | Add highlighting to an application to show what piece(s) was selected and where it can move. | 10 |
| 3 | Implement fully working movement of pieces including pushing and taking. CustomControl should accept user input for making moves (e.g. using multiple clicks and drag). | 30 |
| 4 | Implement winner detection, the game should then end immediately with an appropriate notification | 5 |
| 5 | Add menus/buttons/labels to your application to   * Allow the game to be reset * Show the current state (number of pieces taken) * Show whose turn it is * Show how many pieces each player has left on the board * Show if there is a winner | 15 |
| 6 | Implement multiplayer mode with 3 to 6 players. Majority of credit is awarded for allowing the selection of 2 or 3 players. | 10 |
| 7 | Additional features select one of the following:   * N timers one for each player to implement speed Abalone. Each player should have 2 minutes to make moves. The 1st players timer should start to count down when the game is started, 2nd players timer counts down when 1st player has completed his move and so on. If a player runs out of time then they will lose the game and forfeit future turns. * Animation of moves * Other additional feature of your choice with similar complexity | 10 |
| Please Note that Marks will be Deducted for the following:   * Code that fails to compile -30% * Submission in wrong format. Your project should be submitted in zip format only -10% * Submission with the wrong filename -10% | | |

# 7. Summary of Division of Work

Student Name1: ***Julian Rengstorf***  Student Number1: ***2926988***

Student Name2: ***Markus Jähn***  Student Number2: ***2927494***

Please complete the sections below with regard to the estimate of the division of work between the two partners

If the work was split in the range of 45% to 55% per partner then that is fine and simply say “Work was evenly divided”. If this was not the case then state with a summary sentence. This is the important statement of this file.

Division of work: **work was evenly divided** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Code repository log (if applicable)

<https://github.com/Jules182/Abalone>

## Percentage of work completed by each partner on each class / task

Some area require more work than others so this is only for reference. An average of these values will not be calculated.

|  |  |  |
| --- | --- | --- |
| **Filename / Task** | **Student Name 1** | **Student Name 2** |
| 1 | 50% | 50% |
| 2 | 50% | 50% |
| 3 | 50% | 50% |
| 4 | 50% | 50% |
| 5 | 50% | 50% |
| 6 | 50% | 50% |
| 7 | 50% | 50% |

# 8. Screen Shots of Working Application

**N.B. Be sure to comment what is working and not working for each of the tasks. The boxes should be expanded to contain the content.**

|  |
| --- |
| **Task 1 (1 image with description + what is working/not working)** |
| This screenshot shows the initial state of the game. Each player has 14 pieces on the board. In the sidebar, the current player is displayed. Below, the game state for each player is displayed: How many pieces the player has taken and how many he has left on the board. |

|  |
| --- |
| **Task 2 (4 images of different types of selection including description + what is working/not working)** |
| In this case, one piece was selected and has three possible directions in which it can move.    Here, two pieces were selected and can perform a parallel or linear movement. The destinations are always related to the last most recently selected piece. |

|  |
| --- |
| **Task 3 (6 image (3 different types of move with before and after states) + what is working/not working)** |
| Movement of cells      Pushing of a cell    Taking of a cell |

|  |
| --- |
| **Task 4 (3 images with description + what is working/not working)** |
|  |

|  |
| --- |
| **Task 5 (3 image with description+ what is working/not working)** |
| In the menu is a button to reset the game    There is a side menu which shows:  - Show the current state (number of pieces taken)  - Show whose turn it is  - Show how many pieces each player has left on the board    There is a Alert which shows if there is a winner |

|  |
| --- |
| **Task 6 (2 to 5 images with description + what is working/not working)** |
| limited to 2 player mode |

|  |
| --- |
| **Task 7 (3 images with description + what is working/not working)** |
| There aren’t additional features. |
|  |