**AC11003 Java Online**

**Name: Ronan Hughes**

**Statement of requirements**

*Design the game Battleships using Java.*

*The game should be able to have players who have their own grid and their own fleet (one Battleship, two Cruisers, three Destroyers and four Submarines).*

*The user places their own fleet on their grid before the game begins, the game begins by each player taking turns to ‘shoot’ at each other by entering a position on the opposing players grid.*

*Once a shot has been fired the shooter will be informed if the shot has hit part of the opposing players fleet.*

*Once the entire fleet is destroyed the game ends with the player with some remaining fleet winning. Due to the nature of this game being played on a shared screen only one human player can play, and the other player will be automated.*

• A menu system to allow a user to control the program options. the project starts here the basic Χ problem to be solved Χ 3

• A Graphical Display Window which shows the current state of the board. (Note that we will help here by providing a display class that you can use or modify)

• A start a new game option where the computer randomly selects positions of the fleet and initialised the grid.

• A fire shot option which allows the user to select a grid position to shoot, updates the grid and keeps track of the number of shots fired.

• A view grid option where the grid showing ships hit and shots fired is displayed.

• A Save Game option to which saves the fleet and grid positions to a text file. This can be used to save a game in progress which can then be finished at a later date.

• A Load Game option which loads a previously saved game and allows the user to continue playing.

* *state clearly* ***what has to be done***
* *state who the* ***users*** *are and describe their needs*
* *state clearly what* ***data*** *is available*
* *state clearly what data is required*
* *state clearly what* ***output*** *will be*
* *state clearly usability requirements*
* *state clearly standards*
* *state clearly assumptions made*
* *state measurable* ***usability & performance*** *targets*
* ***functional*** *requirements*
  + *the tasks the system must perform*
  + *how the system should react in situations*
  + *what the system should NOT do*
* ***non-functional*** *requirements*
  + *constraints upon the system and/or us*
  + *product - usability, portability, reliability, efficiency*
  + *process - delivery, implementation & documented standards (such as the IEEE standards)*
  + *external - other programs, ethics, legal, safety*

**Requirements**

*The battleship grid will have 100 squares (10 on each side). With letters on the rows and numbers on the columns so players will find it easy to call coordinates. See below.*

*C here*

*10 here*

*9 here*

*8*

*7 here*

*6 here*

*5 here*

*4*

*1 here*

*2 here*

*3 here*

*D here*

*H here*

*G here*

*F here*

*E here*

*I here*

*J here*

*B hereB*

*A here*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  | *X* | *X* | *X* |  |  |  | *X* |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  | *O* |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | *X* |  |
| *O* |  |  | *O* |  |  |  |  | *X* |  |
|  |  |  |  |  | *O* |  |  |  |  |
|  |  |  | *O* |  | *O* |  |  |  |  |
|  |  |  |  |  |  |  | *O* |  |  |
|  |  |  | *O* |  |  |  |  |  |  |

*A players’ fleets will have one Battleship which takes up 4 grid spaces each, two cruisers which take up 3 grid spaces each, three Destroyers which take up 2 grid spaces each and three Submarines which take up a single grid space each.*

*The game will show 2 grids, the players grid which shows the position of their fleet and the opposing players grid which shows the position of both successful and unsuccessful shots.*

**What will your program be like for the user?**

*The program should be easy for the user to use, the action buttons should be obvious. The rules should be explained or at least a link to somewhere the user can read the rules.*

*The user should be able to place their fleet by clicking on the marine craft and then clicking the area on the grid where they want to place it.*

*After the positions of their fleet have been chosen the player should be able to click “Start Game” where the computer automatically selects positions for their fleet on the opposing grid. See below.*

*A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA picture containing ship, photo, boat

Description automatically generated*

LOAD GAME

Place your fleet in the grid

START GAME

*Once the play has placed their fleet and pressed the ‘START GAME’ button they will be taken to the game play screen which looks like below.*

*A screenshot of a cell phone

Description automatically generated A picture containing rain, clock

Description automatically generated*

More Information

X = successful shot

O = unsuccessful shot

QUIT

SAVE

SHOOT!!

*The user can click on ‘QUIT’ if they wish to quit the game. The user could also click on ‘More Information’ which brings them to* [*https://www.hasbro.com/common/instruct/Battleship.PDF*](https://www.hasbro.com/common/instruct/Battleship.PDF) *where they can view the rules of the game. There is a legend on the main game screen which shows that x means a successful shot was fired on the opposing player grid spaces and an O means an unsuccessful shot was fired on the opposing player grid spaces.*

**User Assumptions about the user**

The User should be able to use a simple computer and know the rules of Battleship before playing. To help the user the user can use the ‘More Information’ link to learn.

**System Requirements**

User will need a basic computer with display and a browser software.

***Usability***

*Clean UX design so users know their options and what to do next. To do this the game will be kept simple and the interface will be easy to navigate.*

**Classes, Fields and Methods**

*Candidate Classes*

|  |  |  |
| --- | --- | --- |
| **Candidate Classes** | **Accept / Reject** | **Reason for rejection** |
| User Grid (Board) | Reject | No need to split the Grid class, keep code DRY |
| Opposition Grid (Board) | Reject | No need to split the Grid class, keep code DRY |
| Grid (Board) | Accept |  |
| Player (user) | Accept |  |
| Player (bot) | Accept |  |
| Ship | Accept |  |
| Position (Grid Space) | Reject | Grid class will have this information |
| Battleship | Reject | Ship classes decided to be more relevant rather than individual ship |
| Cruiser | Reject | Ship classes decided to be more relevant rather than individual ship |
| Submarine | Reject | Ship classes decided to be more relevant rather than individual ship |
| Destroyer | Reject | Ship classes decided to be more relevant rather than individual ship |
| Start Screen | Accept |  |
| Game Screen | Accept |  |

*Class Fields*

Grid

* Position of fleet
* Position of opposing player shots
* Number of squares

Player

* User Name

Ship

* Ship Type
* Ship Alignment
* Ship Size
* Ship Damage

Start Screen

* Start Button
* Load Button

Game Screen

* Grids
* Shoot Button
* Save Button
* Quit Button
* Player Info

*Class Methods*

Grid

* addHit () - Add a opposing player successful shoot
* addMiss () – Add a opposing player unsuccessful shoot
* removeShots () – clear the board of shots
* makeGrid () – to start game

Player

* addUser () – Adds user name to player info

Ship

* selectShip () – selects ship for placement into grid
* rotateShip () – changes ship alignment prior to be being placed in grid
* shipHealth () – gives a percentage of ship remaining unhit

Start Screen

* start () – loads game screen with empty grid
* load () – loads games screen with previous stored data

Game Screen

* shoot () – decreases ship health if grid position matches ship position
* save () – saves all position and player data for use later
* quit () – exits game back to ‘Start Screen’

**Class Diagram**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

You report should be no longer than 10 pages and should comprise the following sections. (Page sizes are guidelines. Try to stick to them. Quality not quantity is what counts.)

• Executive Summary (One Page)

o Describes what your program does.

o Describe successes and failures of the project.

o Provide an overall conclusion as to the success and usefulness of the project.

• Introduction (Half Page)

o Introduces the report to the reader

• Requirements (One/Two Pages)

o Describes requirements for the project. Note that your requirements should not change much from the ones you have produced earlier. You may not meet all the requirements but that is fine as long as you document that in your report and say why.

• Design (One/Two Pages)

o Describes the design of the project. Your design should show how your program has been designed from an object-oriented perspective, and also show how you went about designing the user experience of your program. You should also include some pseudocode for the more complex methods in your program.

• Source Code o Describes any features of the source code you want to point out. The source code listing should include JavaDoc comments and the JavaDoc html should be generated. Use sensible names for classes, methods, fields and variables to make your program as self documenting as possible.

• Instructions (Half Page)

o A set of instructions for running the program in a file called readme.txt.

• Evaluation (One page)

o Describes how the program was evaluated.

o How did you prove it worked and how did you check for bugs.

o Try testing your program with other users apart from yourself.

o Are there any bugs you know of? Describe them.

o Any of the requirements missed? Any new ones added in?

• Summary and Conclusion (One Page)

o Summarises the report and provides a conclusion as to the success and usefulness of the project.

• Critical Self Evaluation (One Page)

o Provide a self evaluation of how well you have done.

o What have you learned?

o How would you do things differently if you could do it again?

o How could you provide a better product?

o Award yourself a mark out of 100 and justify it