# **Battleships**

### **Self-evaluation:**

One of the first challenges was getting started with the project and visualizing how it would run. It was quite difficult to decide on classes, their interactions and algorithms. Possibly the most difficult algorithms to visualize were regarding placing enemy and user’s ships as a lot of checking needed to be made in order to make sure the ship is not being placed on top of another ship or out of boundaries. I discovered that I get lost in my own code quite quickly and realized the importance of adding comments as you are programming. That has not been an issue before since the algorithms were less complicated. The same can be told about pseudocode which was very useful to write before beginning to code the more difficult algorithms. Other difficulties included deciding whether the methods/fields should be static or non-static to get the most optimal outcome. I have also discovered that interaction of classes is a lot more intricate that I had imagined previously. However, completing this project aided me in understanding how classes work/interact. I believe that completing the code helped me to strengthen my knowledge gained from previous assignments. I have learned how to throw exceptions in order to deal with incorrect user input which deemed to be incredibly useful to ensure the robustness of the program. All in all, I am quite satisfied with the outcome and I am happy that I was able to implement two game modes (against responsive and non-responsive AI). I am also satisfied with the ability to pause the game at any point in order to save it or return to menu since it took more time to implement.

### **Requirements**

**Functional Requirements**

R1. The system shall run on JVM.

R2. The system shall have a main menu screen.

R3. The system shall be an option to begin a new game by generating a new text file.

R4. The system shall have an option to save the game progress into a text file.

R5. The system shall have an option to load a previously saved game file.

R6. The system shall have an option to exit the game.

R7. The system shall have a mode against non-responsive AI.

R8. The system may have a mode against a responsive AI.

R9. In the playing screen the system shall be displayed a 10x10 grid.

R10. The columns on grid shall be numbered vertically from 1 to 10 and horizontally from A to J.

R11. The system shall have a grid that saves position of user ships.

R12. The system shall have a grid that saves position of AI ships.

R13. The system shall have a grid that records user shots.

R14. In the playing screen the system shall allow user to fire by entering coordinates.

R15. In the playing screen there should be an additional menu:

R15.1 In the additional menu there should be an option to save the game.

R15.2 In the additional menu there should be an option to return to menu.

R15.3 In the additional menu there should be an option to exit the game.

R15.4 In the additional menu there should be an option to continue the game.

R16. The system shall record data of overall shots user made.

R17. The system shall record data of the overall shots AI made.

R18. The system shall record data of the successful shots user made.

R19. The system shall record data of the successful shots AI made.

R20. The system shall display statistics at the end of the game.

**Non-functional Requirements**

NFR1. The device shall have Java Runtime Environment installed.

NFR2. The device shall have a minimum of 2GB of RAM.

NFR3. The device shall have a minimum of 20MB of HDD/SSD space.

NFR5. The time for the AI to make the decision on its turn should not take longer than 2 seconds.

NFR8. The program shall be compatible with Windows 7/10 32/64 bit.

### **Use Cases**

**User Case Title: Play a new game against non- responsive AI**

**User Case Description: User starts a new game against a non-responsive AI.**

User: Selects the option ‘Start a new game against non-responsive AI’ in the main menu.

System: Clears the grid.

System: Randomly places AI ships.

System: Starts the game.

User: Selects a point on the system’s grid and shoots.

System: Checks whether the shot was successful.

System: Grid is updated with every shot and displayed to the user.

System: Once all AI ships are destroyed, the statistics are displayed.

**User Case Title: Play a new game against functional AI**

**Use Case Description: User starts a new game against a responsive AI.**

User: Selects the option ‘Start a new game against responsive AI’ in the main menu.

System: Clears the grid.

System: Randomly places AI ships.

System: Prompts user to place ships.

User: Places ships.

System: Starts the game

User: Chooses a location to shoot at.

System: Updates the grid.

System: Shoots at the user’s grid.

System: Updates the grid.

….

System: The winner is announced at the end of the game.

System: Statistics are displayed.

**User Case Title: Choose to ‘Load Game’**

**User Case Description: User loads a previously paused game on the Menu interface.**

User: Selects the ‘Load Game’ option in the main menu.

System: Loads data from previous game into the system.

System: Continues the game.

User: Plays the game from the point it was last saved.

**User Case Title: Choose to ‘Exit Game’**

**User Case Description: User chooses the Exit Game option on the Menu interface.**

User: Selects the ‘Exit Game’ option in the main menu.

System: Closes game window.

**Use Case Title: Choose to ‘Save Game’**

**User Case Description: User chooses to save the progress of the current game.**

User: Selects ‘Save Game’ option on the main menu,

System: Updates the save file with current game data.

System: Informs user that the game has been saved.

**Use Case Title: Choose to ‘Go back to main menu’**

**User Case Description: User is sent back to main menu.**

User: Selects ‘Go back to menu’ option

System: Returns user to the main menu

**Use Case Title: Choose to ‘Continue’**

**Use Case Description: User chooses to continue game once paused**

User: Selects ‘Continue’ option on the menu

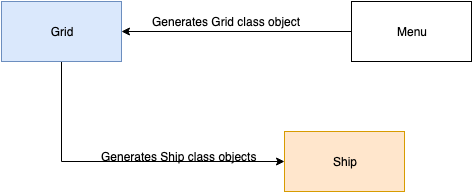
System: Continues the game from where it was paused.

### **Classes**

##### **Class Descriptions including Responsibilities, Fields and Methods**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Responsibilities** | **Fields** | **Methods** |
| Grid | Stores information about the game grids and shots made by both AI and the user. | grid[]  enemyGrid[]  displayedGrid[]  shotsFired  successfulShots  AIShotsFired  AISuccessfulShots  GRID\_SIZE  (based on Ship class)  battleship  cruiser  destroyer  submarine  allShips[] | setEmptyValues()  saveGameToFile()  loadGameFromFile()  placeAShipByUser()  placeAllUserShips()  placeAComputerShip()  placeAllComputerShips()  fireShot()  AIFireShot()  displayBoard()  displayEnemyBoard()  displayedBoard()  convertLetterIntoNumber()  getAIShotsFired()  getAISuccessfulShots()  getShotsFired()  getSuccessfulShots() |
| Ship | Stores information about each ship type. | shipSize  shipQuantity | getShipQuantity()  getShipSize()  setShipQuantity()  setShipSize()  setShipValues() |
| Menu | Handles user choices in main and additional option menus. | game | processUserChoices()  openAdditionalOptions()  displayMenu() |

##### **Class Diagram**

****

***Activity Diagrams / Pseudocode***

**Placing enemy ships:**

Data recorded to enemy grid

Method is called for each type of ship.

Loop for each ship of the determined type - input parameters are ship size and quantity.

* Randomly generates two numbers between 1 and 10 included

*[Make sure the numbers are not out of boundaries]*

* Start placing the ship in the randomly generated coordinate field

*[Randomly generated coordinate x;y]*

* Choose vertical or horizontal location randomly

If vertical: add to coordinates y, scan the vertical row for the number of fields that ship takes to make sure it doesn’t overlap.

If it overlaps, regenerate the coordinates

If horizontal: add to coordinates x, scan the horizontal row for the number of fields that ship takes to make sure it doesn’t overlap.

If it overlaps, regenerate the coordinates

* Check if ship is not out of grid the entire time

If ship is about to be placed out of the grid, regenerate the coordinates.

**Start a new game:**

* Use the method to set blank fields
* Use the method to place enemy ships
* User specifies a grid position

Compare the user’s board with enemy board to see if there is a ship that was hit

If the enemy ship is hit mark with X, if shot is missed mark with o

Increase number of shots by 1;

If shot is successful increase the number of successful shots by 1;

Board is refreshed.

Game ends when successful shots reach 19 as it is the number to destroy the entire enemy fleet.