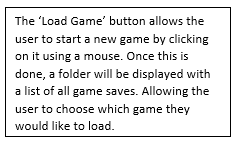
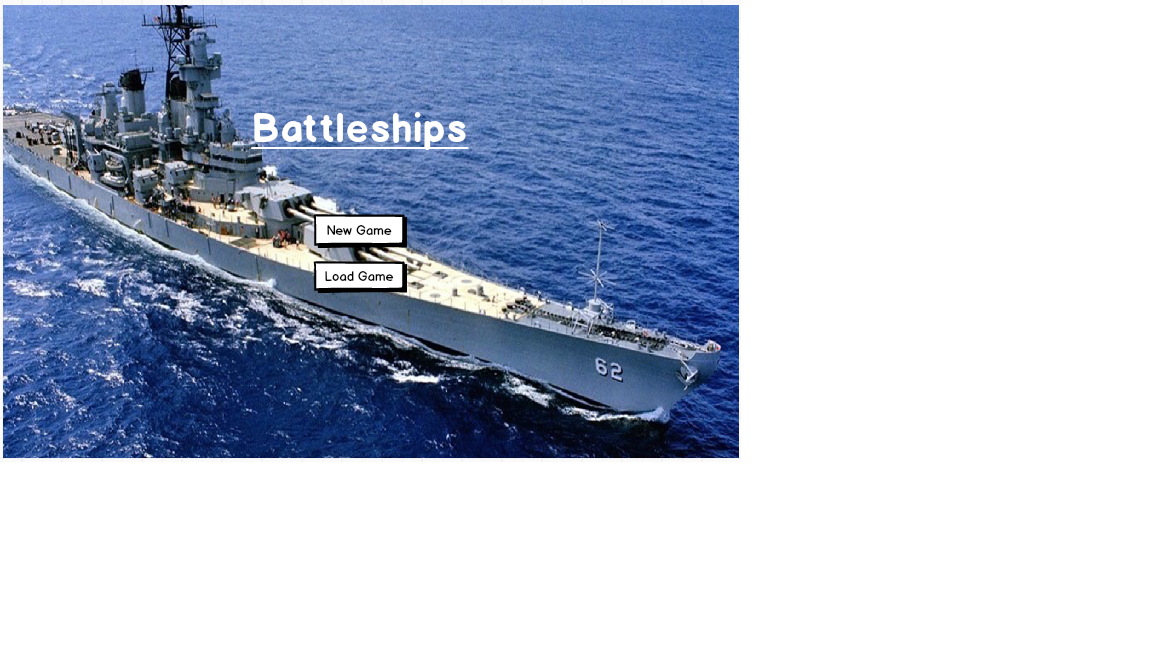
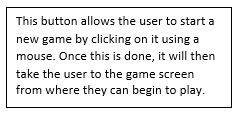
**Battleship Project**Date:27/09/17  
Student Names: Dale Brooksby, Peter Leslie, Tom Smith, Yousaf Nasir  
  
**Statement of requirements**   
  
The aim of this project is to create a functional game of Battleships in Java. The game will involve the computer randomly placing ships on a grid. The user will then select which grids to shoot at, in hopes of hitting the computer's ships. The game is finished when all the computers ships are destroyed. It will be assumed the users know the rules of battleships prior to playing. The program will be played by fellow students. These users will be able to make inputs with a mouse or keyboard. The game will feature the ability to save their current state and reload that save. Furthermore the program will feature a menu from which users can adjust the programs options.

**Requirements  
Functional Requirements**  
  
R1. The program should be able to randomly place ships at the start of a game  
R2. The program should display which grid spaces have been shot or missed.  
R3. The program must be able to Save the current game state  
R4. The program must be able to reload that save state; or any other saved file.  
R5. The Program should include a menu which users can adjust game options.  
R6. The Program should have a graphic display window that shows the current game state.  
R7. The program should display a GUI which allows them to target grids.

**Non-functional Requirements**  
  
R1. The UI needs to be aseptically appealing with a clean design

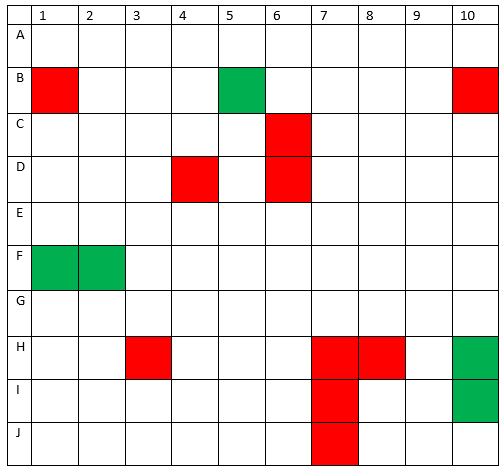
R2. The computer may make intelligent decisions of where to shoot.  
R3. The controls must be easy to use.  
R4. Grids must change their status quickly after having been selected

**User Interface**

The background is an image of a battleship.

This is to make the game more user friendly and eye catching.

As it is also more relevant to the game.

This is a visual representation of the grid system. The ‘Red’ boxes display where the user has taken a shot and missed. Whereas the ‘Green’ grids show where the user has taken a shot and hit a ship.

### **Use Cases**

|  |  |  |
| --- | --- | --- |
| Main menu | | |
| 1 | System: | Draw menu |
| 2 | User: | Select option: |
| 2.1 |  | New game |
| 2.2 |  | Load game |
| 2.3 |  | Options |

|  |  |  |
| --- | --- | --- |
| Battleship Game: Start new game (2.1) | | |
| 1 | User: | Start a new game |
| 2 | System: | Draw UI |
| 3 | System: | generate random locations for ships |

|  |  |  |
| --- | --- | --- |
| Battleship Game: Load saved game (2.2) | | |
| 1 | User: | Load from saved file |
| 2 | System | Read save from file |
| 3 | System: | Draw UI |
| 4 | System: | Place ships on saved locations |

|  |  |  |
| --- | --- | --- |
| Battleship Game: Gameplay | | |
| 1 | User: | Selects grid space to shoot |
| 2 | System: | Determine if ship was hit |
| 3 | System: | Update UI to reflect hit or miss |
| 4 | System | Save game |
| 5 | User: | Starts next turn |
| 6 | System: | repeats steps 4 - 7 until all ships hit |
| 7 | System: | Displays a Game Over screen |

|  |  |  |
| --- | --- | --- |
| Battleships Game: Save game | | |
| 1 | User: | Select save game option |
| 2 | System: | Write data to file |
| 3 | System: | Resume gameplay |

### **Classes**

##### **Candidate Classes**

The aim of this project is to create a functional game of **Battleships** in Java. The **game** will involve the **computer** randomly placing **ships** on a **grid**. The **user** will then select which grids to shoot at, in hopes of hitting the computer's ships. The game is finished when all the computers ships are destroyed. It will be assumed the users know the rules of battleships prior to playing. The program will be played by fellow **students**. These users will be able to make inputs with a **mouse or keyboard**. The game will feature the ability to **save** their current state and **reload** that save. Furthermore the program will feature a **menu** from which users can adjust the programs **options**.

|  |  |  |
| --- | --- | --- |
| **Candidate Classes** | **Result** | |
| BattleshipsCoordinator | Accept | Top level coordinator |
| Game | Reject | Irrelevant |
| Computer | Reject | Irrelevant |
| Ship | Accept | Ship class |
| Grid | Accept | Grid class |
| User | Reject | Irrelevant |
| Computers ships | Reject | Already a class (Ship) |
| Game | Reject | Irrelevant |
| Rules | Reject | Irrelevant |
| Battleships | Reject | Already covered (Ship) |
| Program | Reject | Irrelevant |
| Students | Reject | Not necessary |
| Mouse or keyboard | Reject | Irrelevant |
| **Save** | Accept | Save class |
| **Reload** | Accept | Load class |
| Menu | reject | Part of BattleshipsCoordinator |
| **Options** | Accept | Options class |

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

**BattleshipsCoordinator**

* + **Description:** Top level class. Calls main method, and coordinates all other classes in the program.
  + **Fields:**
  + **Methods:**
    - setPoints(): Sets the number of points (grid spaces) that the user has accumulated
    - main(): Main method, called when program runs

**Ship**

* + **Description:** Stores the fields and methods about the ships that are used in the game
  + **Fields:**
    - type (String): the name of the type of ship
    - size (int): the number of grid spaces the ship occupies
  + **Methods**
    - setLocation(): sets the random location of each ship

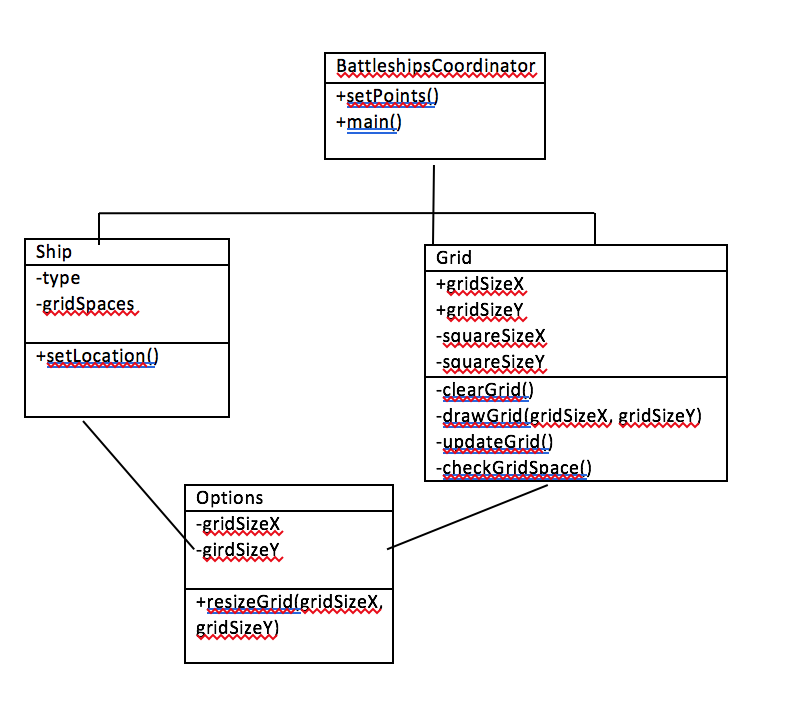
**Grid**

* + **Description:** saves information about the grid and which grid spaces are occupied by a ship. Contains methods relevant to the grid UI.
  + **Fields:**
    - gridSizeX (int): width of the grid in squares.
    - gridSizeY (int): height of the grid in squares
    - squareSizeX (int): width of the square in pixels
    - squareSizeY (int): height of the grid in pixels
  + **Methods**
    - clearGrid(): clears the grid
    - drawGrid(gridSizeX, gridSizeY): generates a grid of size (gridSizeX, gridSizeY)
    - updateGrid(): updates the UI to reflect changes on the grid
    - checkGridSpace(): Checks whether the grid space selected by the user is occupied by a battleship

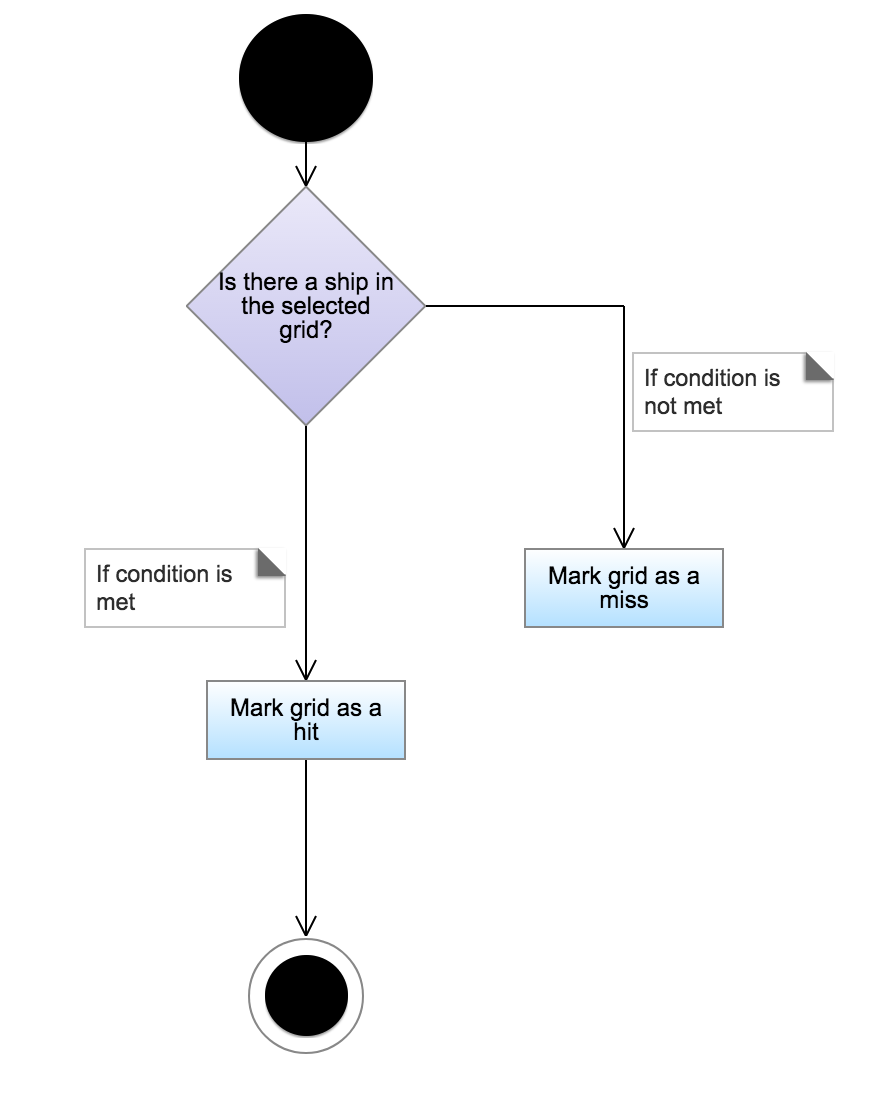
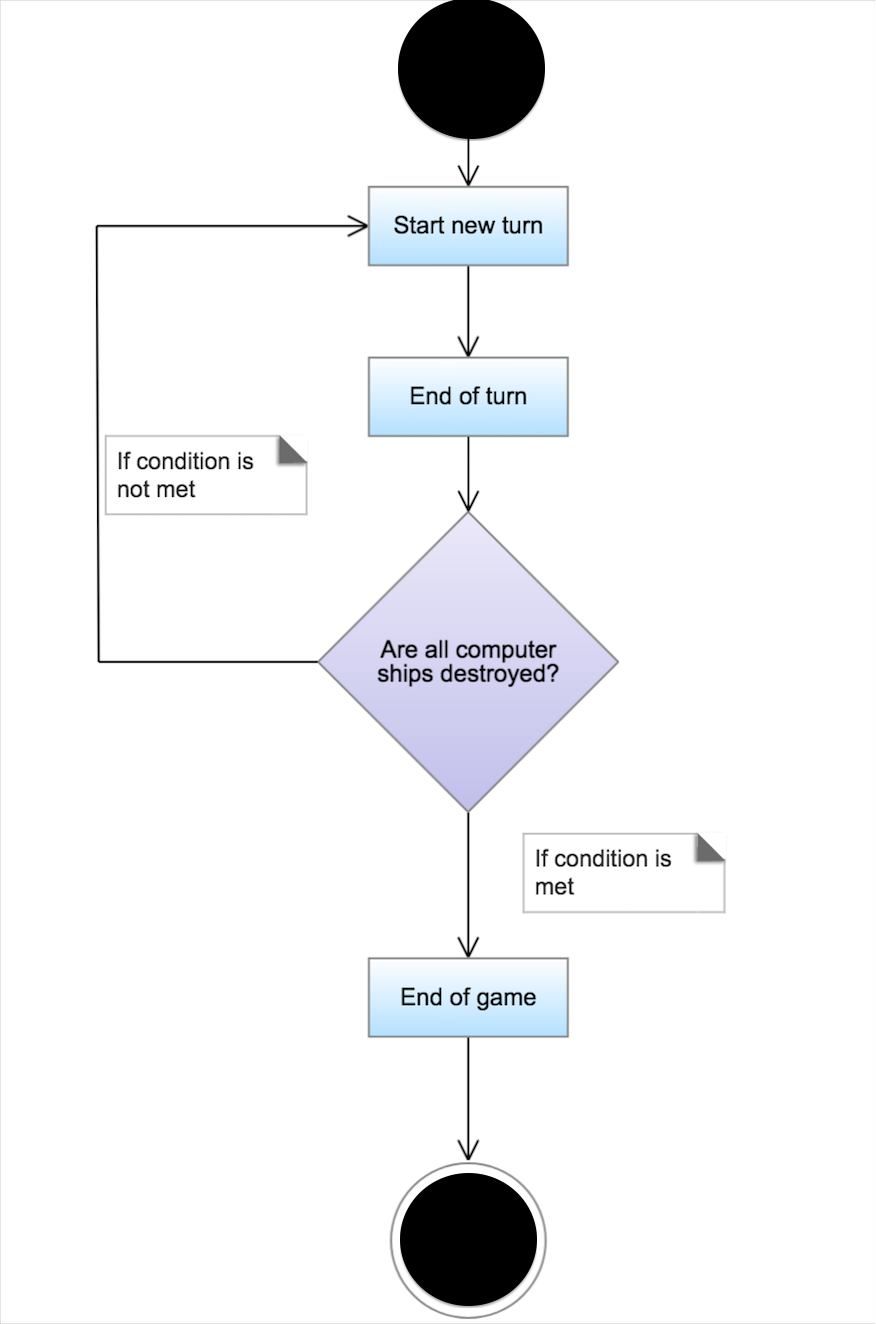
**Options**

* + **Description:** saves information for
  + **Fields:**
  + **Methods:** 
    - resizeGrid(gridSizeX, gridSizeY): changes the size of the grid which the user selects in an options menu

##### **Class Diagram**



**Activity Diagrams**



**Pseudocode**

CREATE boardArray of type ship[9][9]

GET menu input FROM mouse

**New game**

-Place random ships-

**Load game**

GET file data FROM system

SET boardArray TO file data

**Game loop**

WHILE playerWon = FALSE

IF there are enemy ships left

GET user input FROM keyboard

seperate input into x and y

IF boardArray[x][y] contains ship

ship.shipType = destroyed

END IF

ELSE

SET playerWon TO true

END IF

END WHILE

**Place random ships**

Ships should be placed from largest to smallest

WHILE shipPlaced = FALSE

Generate 2 random numbers xy, from 0-9

SET boardArray[x][y] TO ships origin square

Store coordinates of ship

Generate random direction, 0-3

FOR length of ship

Try placing next ship square

IF error

DELETE all stored coordinates for current ship

ELSE

shipPlaced = true

store coordinates of ship

END FOR

move onto next ship