**Modular Interface Specification**

Battleship Class Reference

* void getGameMode ()
* void setGameModeOn ()
* void setGameModeOff ()
* void getSetMode ()
* void setSetModeOn ()
* void setSetModeOff ()
* void setPlayerShip ()
* void setGrid ()
* void setCompShip ()
* boolean checkInGrid ()
* void setShipInGrid ()
* void createAndPlaceAShip ()
* int randomInt ()

BattleshipGUI Class Reference

* updateGameTable()
* updateSetTable
* actionPerformed()
* mouseClicked ()
* mousePressed ()
* mouseReleased ()
* mouseEntered ()
* mouseExited ()

BattleShipStatisticsFileIO Reference

* Int getUserHighScore(String)
* Int getOverallHighScore()
* writeUserHighScore()

GameLoadSaveIO Reference

* + - openGame ()

1. saveGame ()

Modular Interface Specification

***1.1 MODULE BattleShip***

Enables user to play Battleship game against computer.

**Access Programs:**

getGameMode ()

* Checks for the game mode
* References: NONE

setGameModeOn ()

* Sets the game mode on
* References: NONE

setGameModeOff ()

* Sets the game mode off
* References: NONE

getSetMode ()

* Checks the status of the set game
* References: NONE

setSetModeOn ()

* Sets the already set game on
* References: NONE

setSetModeOff ()

* Set the mode of the already set game off
* References: NONE

setPlayerShip ()

* Sets the ships for the player
* References: x, y, shipType

setGrid ()

* Sets the 10x10 grid which users will be using to play the game
* References: NONE

setCompShips()

* Randomly places the computer’s ships on the grid
* References: NONE

checkInGrid()

* Checks the grid to see if there is more space available
* References: x, y, length, direction

setShipInGrid()

* Allows the user to set their ships at desired places
* References: x, y, shipType, length, direction

createAndPlaceAShip

* Creates users and computers ship and then calls setShipInGrid function to set the newly created ship in the grid
* References: shipType, length, quantity

randomInt()

* Creates a random integer between 0 and x
* References: in

***MODULE* BattleShipGUI**

The module uses the following classes:

* java.awt
* java.awt.event.ActionEvent
* java.awt.event.ActionListener
* java.awt.event.MouseEvent
* java.awt.event.MouseListener
* javax.swing
* javax.swing.border.Border

updateGameTable()

updateSetTable

actionPerformed()

mouseClicked ()

mousePressed ()

mouseReleased ()

mouseEntered ()

mouseExited ()

***1.3 MODULE* BattleShipStatisticsFileIO**

This module provides methods which the user is able to check and modify the high Scores.

The module uses the following classes:

* java.io
* util.Scanner
* util.ArrayList

int getUserHighScore(String)

* Input : String username
  + Desired username of high score
* Attempts to find user name in high score list
* Returns -2 for IO error
* Returns -1 if name not found
* Returns positive integer, representing corresponding high score for username otherwise

**int getOverallHighScore()**

* Input: NONE
* Attempts to get overall high score from high score list
* Returns “-1” if there is no overall high score
* Returns “-2” for IO error
* Otherwise, returns integer for overall high score

**String getOverallUserName()**

* Input: NONE
* Attempts to get overall user name from high score list
* Returns “-1” if there is no overall high score
* Returns “-2” for IO error
* Otherwise, returns String for overall user name

**Int writeUserHighScore()**

* Input: String UserName, int HighScore
* Get high score list from high score file, check if overall high score is broken and replace with user high score if it is, then add user high score to list, write back high score file
* Returns -2 for IO error
* Return 0 if all is well

**Int clearUser()**

* Input: String UserName
* Attempts to find user name in high score list
  + If found, deletes user name and score from list, and finds new overall high score if deleted user was overall high score then return 0 for correct process
  + If not found, return -1
* Returns -2 for IO error

***1.4 MODULE GameLoadSave***

This module allows users to load a previous saved game or to save a game while playing.

USES: java.io

readFile ()

* INPUT: NONE
* OUTPUT: INT
* Reads game data from file, returns 0 on expected execution and -1

getUserHealth ()

* INPUT: NONE
* OUTPUT: INT array of length 7
* Returns user health values

getCompHealth ()

* INPUT: NONE
* OUTPUT: INT array of length 7
* Returns Computer’s health values

getUserGrid()

* INPUT: NONE
* OUTPUT: INT 2D array of 10x10
* Returns user’s grid values

getCompGrid()

* INPUT: NONE
* OUTPUT: INT 2D array of 10x10
* Returns Computer’s grid values

saveGame()

* INPUT: NONE
* OUTPUT: INT
* Writes game data to file, returns 0 for expected execution, and -1 for I/O error