Program Title: Battleship Game
Author: Michael Hoover
Student ID: 109475011
Date: December 3, 2019

Problem Description:

The program allows the user to play a game of battleship against the computer. It allows the user to place the ships at the beginning of the game and to guess shots throughout the game. The program generates ship placement and shots for the computer player. It terminates when either side wins the game or when the user wants to quit.

Extra Credit:

The program completes two requirements for extra credit. It requires that ships be hit at every location before they are sunk. The program also implements a strategy for the AI. The computer takes shots in a checkered pattern so as to uses it’s guesses most efficiently. When the computer hits a ship, it will follow a decision tree to find the rest of the ship until the ship has been sunk.

Input Requirements:

The program will accept a file in csv format for user ship placement. The program will read 5 lines from the file corresponding to the 5 different types of ship. It will read 3 values for every line:

- The first value will be a string type value and will denote the type of ship. Valid inputs are “carrier”, “battleship”, “submarine”, “cruiser”, and “destroyer”. Capitalization will not matter. Any other input for this value will be invalid. Also, each line must have a unique string so that the program can place 5 unique ships. Repeated values will be invalid.
- The second value of each line will be a string type. The string must consist of a letter, A through J, followed by a number, 1 through 10. There will be no white space between the letter and the number so the program will need to separate these values during processing. This string denotes the placement of the front of the ship on the grid.
- The third value in each line will be a char type, either “H” or “V” to denote whether the ship is placed horizontally or vertically. All other characters will be invalid inputs.

The program must also check that all the inputs are valid in conjunction with one another. Any inputs which place any part of the ship outside the 10x10 grid are invalid. Any inputs which have more than one ship occupying a square on the grid are invalid. So, for instance, although the two lines “Carrier, A7, V” and “Destroyer, A8, H” both contain values that are within valid range when considered individually, when taken together they will be considered invalid inputs because 1) the input file attempts to place the carrier so that part of the ship is outside the bounds of the 10x10 grid and 2) the input file attempts to have both the carrier and destroyer occupy the square A8.

While the game is being played the program will accept inputs from the keyboard for the user's shots. The inputs will be string type and must contain a letter, A through J, followed by a number, 1 through 10, with no whitespace between the letter and the number. The letter expresses the x coordinate on the grid and the number expresses the y coordinate. So, during processing the program will separate this string into two distinct values. The program will consider repeated guesses to be invalid. The user must guess a point on the grid he/she has not previously guessed.

Output Requirements:

For every shot the user takes the program will output a message to the screen announcing whether the shot was a hit or a miss. For example, if the user guesses D5 the program might output "D5 is a hit!" to the screen.

The program will output the guesses the computer player makes to the screen. It will also announce whether the computer player scored a hit or a miss. For instance, when it is the computer's turn the program might output "Player 2 guesses J9. It's a miss." to the screen.

The program will output an announcement to the screen every time a ship is sunk.

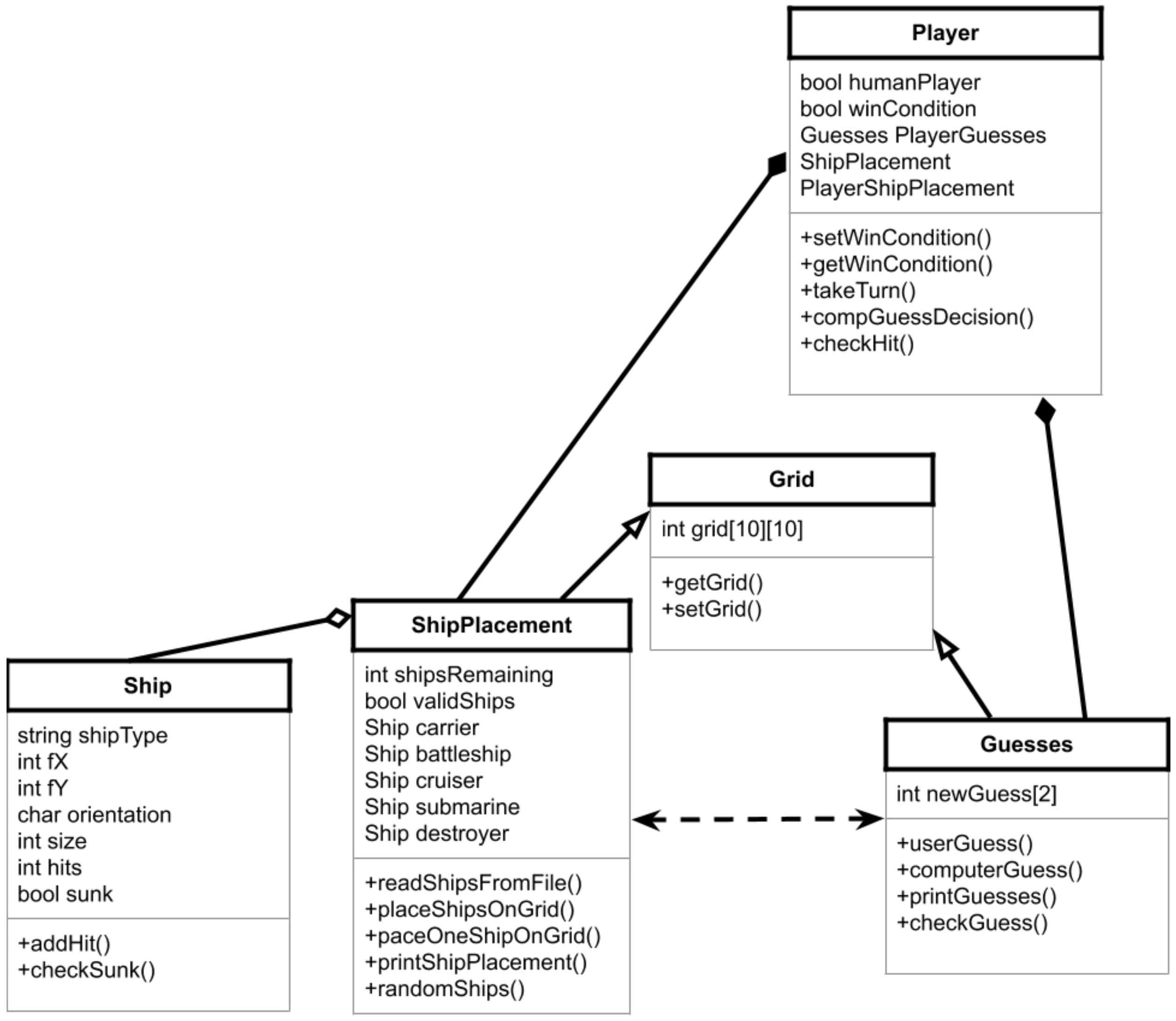
The program will output the winner of the game to the screen when either the user or the computer sinks all 5 ships.

After the conclusion of the game the program will output the placement of the computer player's ships.

Problem Solution Discussion

The program will randomly generate the position of the computer's ships. Then the computer will store the location of the user's and the computer's ships in in ShipPlacement objects that use 2 dimensional arrays. When the players fire torpedoes at each other, the program will check that the guesses are in range, that it is not a repeated guess, and finally it will check the coordinate against the opponents ship placement grid to see if it scored a hit. Each player will have a Guesses class object that uses a 2-dimensional array to keep track of the previous shots fired and whether or not they scored misses or hits. After every turn the program will check to see if the player has met the winning condition, sinking all 5 opponent ships. The program will continue to accept guesses until one of the players achieves the winning condition or the user quits the game.

UML Diagram



Overall Software Architecture:

Class Player member functions

- setWinCondition(): this function calls the getShipsRemaining for the opponents ships. If it returns 0 then the function sets the member variable winCondition to true.
- getWinCondition(): this function returns the value of the member variable winCondition.

- takeTurn(): The function allows the player to take a turn in the game of battleship. The function checks the value of humanPlayer. If the variable is true, meaning it is the user's turn, then the function will call userGuess from the Guesses class. If the player is the computer player then the function will call the comGuessDecision() function.
- compGuessDecision(): The function determines how the computer player should make a guess. The function uses 5 linked lists, one for each type of ship. The lists track the any hits that the computer has made against the opponent player. When the function finds there is a ship that has been hit, but not sunk, it will go into a decision tree to attempt to sink that ship. If there is only one hit on the ship, the function will randomly guess one of the four adjacent locations to the hit. If it guesses a location that has already been guessed then it will guess again until it finds a valid guess. If there is more than one hit on a ship the function will determine the orientation of the ship based on the hits stored in the linked list. It will then guess a location that is in line with the ships orientation and also adjacent to a previous hit on that ship. If the function finds that there are no ships that have hits and are also not sunk then it will call the computerGuess() function from the Guesses class to make a random guess.
- checkHit(): The function checks the last shot the player took against the placement of the opponent's ships. If the guess was incorrect, it outputs a message to the screen letting the user know it was a miss. If the guess was correct it outputs a message to the screen informing the user it was a hit. It also marks the guess grid to indicate the location of the hit, calls the addHit() for the corresponding ship, for a computer player the function will add the hit to the list of hits to the corresponding ship to be used by comGuessDecision, and then checks to see if the ship was sunk. If the ship was sunk then the function outputs a message to the screen declaring which ship has been sunk.

Class ShipPlacement member functions

- readShipsFromFile(): The function accepts and input stream of a file as a parameter. The file should be in csv format. It will read the information from the file and will create an object of class ship for each line. There should be 5 lines in the file, one that corresponds to each ship. The function will throw exceptions if the file is not formatted correctly or if there is invalid data.
- placeShipsOnGrid(): this function calls the paceOneShipOnGrid() function five times, once for each type of ship.
- placeOnShipOnGrid(): This function places a single ship on the grid. It changes the int stored in the grid locations to correspond to the type of ship being placed. It also checks for valid placement. If will throw an exception if two ships overlap or if a ship is out of bounds.
- PrintShipPlacement(): This function outputs a grid to the screen. The grid is marked with the placement of the 5 ships.
- randomShips(): This function places 5 ships on the grid at random location. It makes sure that ships are placed in bounds and that the ships do not overlap.

Class Guesses member functions

- userGuess(): The function prompts the user to input a guess on the keyboard. It then

checks to make sure the guess is valid. If the guess is not valid the function prompts the user for another guess until it receives a valid guess. It then calls the checkGuess() function.
- computerGuess(): The function generates a random guess for the computer player.
- printGuesses(): The function prints the guess grid to the screen. The grid indicates all the coordinates that have been guesses and whether or not they were hits or misses.
- checkGuess(): The function checks if the newGuess has been guesses already. If the newGuess is a duplicate guess the function returns false. If the newGuess has not been previously guesses, the function marks it on the guess grid and returns true.

Class Ship member functions
- addHit(): The function increments the private member variable hits by one. If the member variable hits is equal to the member variable size, then the function sets the member variable sunk to true.
- checkSunk(): The function returns the value of private member variable sunk.

Program Structure

The program will start by creating two Player objects. It will get the inputs from the file and will create Ship objects for each ship. The program will then call the placeShipsOnGrid function for the user and place the ships on the grid using the information read from the file. If the placement of the user's ships is invalid it will catch an exception and terminate the program.

For player2 (the computer) it will randomly place the ships on the grid. It will check each placement to make sure it is valid. If not, it will generate another random location for the ship and repeat until it finds a valid placement.

The program will begin a loop to alternate turns between the user and the computer. During a turn the player will input a guess. For the user, this input will come from the keyboard. If the user wishes to quit, an input of “Q” or “q” will end the game. For the computer the guess will be generated by the program. The program will store the guess in memory and check that it is a valid guess. If it is an invalid guess it will keep asking for a new guess until it receives a valid guess.

It will then call the function to check if the guess hit an opponent's ship. It will output whether it was a hit or miss to the screen. Before each turn the program will check to see if getWinCondition is true for either player. When it finds a getWindCondition that is true the program will exit the loop for taking turns and will declare the winner. It will also output to the screen the location of the computer's ships.

Then the program will end.