****

**Ravisher Singh**

**CSCI 2312**

**101844874**

**November 6,2020**

**CSCI 2312 - Design Document**

Problem Description  
Battleship game is a guessing game of two players. Objective of the game is to destroy opposite player’s entire fleet. There are four 10 X 10 grids and two grids are used to mark each players’ fleets of ships(including battleships).Location of the fleet is concealed from the other player so that they don’t know the locations of opponents ship. Players alternate turns by ‘firing torpedoes’ at the other player's ships. In our game, ‘**firing a torpedo**’ will be allowing the player **to take a guess** at where on the grid their opponent may have placed a ship. Human player faces computer opponent . Players take turns to try to guess the location of the other's ships by naming a square. The opponent declares the square to be a hit or a miss, depending on whether there is a ship occupying that square. When all the squares occupied by a particular ship have been guessed, the player must announce that that particular ship is sunk.

1. Input Requirements  
   A detailed **list of all external inputs** (from files or keyboard) including a description of the **data type** and **range of valid values** for each input. For input file format and interactive user input, you need to write what data type is used for every field and valid value and length.

Each Ship player places following requirements to follow:

* + We will need a 10 x10 grid. We will have to consider the ship rows and columns.
  + Ship row must be an **int** type and must be a **positive number** and it must be less than the **grid height**
  + Ship column must be an **in**t type and must be **positive** and it must be less than **grid width**
  + Each ship occupies a certain number of consecutive squares on the grid (sizes of ships are in the following table), arranged either horizontally or vertically, which means I can use a **bool to verify the orientation** of the ship
  + Will have to make sure that the ships cannot overlap so only one ship can occupy any given square in the grid.
  + Check if ship is sunk or not , a bool value
  + Count the total number of hits of ship, int
  + If there is space available on grid to place the ship, bool value

**Each shot fired by player must follow following requirements:**

* + **Ask user to fire the shot at (r,c) in enemy’s grid**
  + Shot row must be an **int** type and must be a **positive number**
  + Shot column must be an **in**t type and must be **positive number**
  + Must check array bounds
  + Have to make sure it does not overlap with existing shot

Ask user whether they will like to continue playing game or not: bool(1/ 0)(y/n)

Along with that we have File Inputs

|  |  |  |
| --- | --- | --- |
| Description | Data Type | Range/valid inputs |
| Type of ship | string | 1.Carrier 2.Battleship 3.Cruiser 4.Submarine 5.Destroyer |
| col | int | 1 to 10 |
| row | Int(ASCII conversion) | 1 to 10, A to J |
| Orientation | string | H or V |
|  |  |  |

CSV file

1. Output Requirements  
   A detailed **list or description of all outputs (**to files or to the screen).

1.A menu

2. Game display

3. Placing ships from file for human player, and computer place his ships randomly

4.Whenver it is user’s turn you will see enemy’s last shot, a message from computer whether human player hit last shot at computer grid or not. So will print a ship placement grid, attack grid and also results of user’s turn

Player’s side:

User grid which shows where ships are placed as per orientation

Once enemy hit us, a display of what is destroyed and what still exists

User’s attack grid will also be displayed at each turn of players, where shots were fired and if fired shots were successful.

Enemy’s side(Computer):

We assuming enemy is firing shots. Where player fired the shot and label them as hit or miss.

1. Problem Solution Discussion  
   It is mentioned in the project to use the modular approach. Modular solution is inefficient and takes more work but because of ease of use, reusability and ease of maintenance this program we will use modular approach.

There are 2 players in the game. Players take turn to fire shots at each other’s grids until all the ships of one player has been sunk which means other player wins. Each player’s turn will be similar but they will use different functions and checks since computer(enemy) makes use of random function while our player manually inputs shot positions. Each input from player is checked to make sure its legal. Ship inputs must be checked for ship overlap , we can do this by comparing it to every other ship and check if rectangles overlap. Same for shots fired, we can check hit/miss but comparing a shot to every ship and location of the ship. To check if ship is sunk we can compare every spot to every shot fired into the grid.

1. (UML diagrams) A UML Class Diagram and Sequence Diagram

Chart, box and whisker chart

Description automatically generated

Sequence Diagram Reference

**Diagram

Description automatically generated**

\*\*\* update to above of diagram instead of using score=17 I used empty() function that will check whether all the spots have been hit, I did it because it was more efficient way.

Sequence Diagram Reference

1. Overall Software Architecture
   * A player class, we will have computer player class and human player class which will inherit from Player class.
   * A game class which will have functionality of playing the game, playing taking turns until one player wins.
   * As we can see from my diagram player class while both computer and player class has “is a “ relationship.
   * Computer class will use random function to fire the toepedo shots and ship placements.
   * Human Player class will not need any private variables but will use public functions to check Input, to display grid and attack grid while computer will not need any of that.
   * Also grid must be two dimensional array.
   * I choose array because we know the size. I could also use vectors but will have to check which is more memory efficient.
2. Test scenarios and cases:

Test cases were verified within.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario Number** | **Test Scenario** | | **Test Cases** | **Pass/Fail** | |
| **1** | **Check whether the computer ship has been placed in its grid** | | **Since we are using random method, computer will try 1000 different tries to place the ship until successful** | **Pass** | |
|  |  | | **If computer even after 1000 tired fails to place the ship we will print out an error msg.** | **Fail** | |
| **2** | **Check whether the computer can place can place a shot in opposite player’s grid** | | **Check if shot is placed is in empty grid** | **pass** | |
|  |  | | **If shot is has been fired in same place** | **Fail** | |
| **3** | **Human Player**  **Firing shot at computer’s grid** | | **If shot bounds entered by human player are valid** | **pass** | |
|  |  | | **If shot bounds entered by human player are out of bound** | **Fail** | |
| **4** | **Check if ship overlaps with another ship** | | **If one rectangle is above another** | **Fail** | |
|  |  | | **If one rectangle is on left side of other** | **Fail** | |
| **5** | **check starting position (col, row) and add ship's size,** | | **If the starting position plus the ship’s size is greater than last element** | **fail** | |
|  |  | | **Check whether location is already occupied(set a occupied value) ,if any position value is equal to occupied value it will -** | **Fail** | |
| **6** | **Check for ship's total size to the starting point to determine if ship would be out of bounds.** | | **8+4(out of bounds)** | **Fail** | |
|  |  | | **8+4 Horizontal placement left or vertical placement up** | **Pass** | |
| **7** | **Check whether someone has won the game** | | **If someone won the game** | **pass** | |
|  |  | **If not** | | | **fail** | |
| **8** |  |  | | |  | |
|  |  |  | | |  | |
| **9** |  |  | | |  | |
|  |  |  | | |  | |
| **10** |  |  | | |  | |
|  |  |  | | |  | |
|  |  |  | | |  | |
|  |  |  | | |  | |