CPSC 233 – Team 8

Project: Battleship Game

Team members: Peisong Yang, Zhengyi Zhou, Britney Phan, Jose Enrico Francisco.

Category: Board Game

The game that we are planning on making is based on a famous guessing game that was published by various companies as a pad-and-pencil game in the 1930s.  The computer will randomly place 5 of the players' ships on a 10 by 10 grid. The locations of the ships are unknown for the other player. Players make their guess by turn and try to "shots" at the other player's ships, the objective of the game is to destroy all the opposing player's ships.

User loads into the game, which displays 10\*10 buttons on the game board. User randomly press the buttons on the game board. If user presses the right buttons which the ships locate on them, the buttons will show an “X”, this means user has already hit the ship for once. If user presses the wrong buttons which there is not ship that locate on them, then the buttons will show an “O”, this means user has not hit the ship. When all ships have been already damaged, the game will be closed. The AI part will be created in final submission.

This is the proposal for Demo1, Demo2 and Demo3.

Demo 1:

For Demo 1, we created four classes, they are: “Board”, “PlaceShip”, “Ship” and “Start”.

The Board class is used to create a game board. It includes two instance variables, they are “mapWidth” and “mapHeight”. “mapWidth” represents the board’s width, and “mapHeight” represents the board’s height. We will play game on this board.

The PlaceShip class is used to place the battleships on the board. It includes the ship’s start coordinates and the direction. Depending on the start coordinates, it can decide which direction the ship can be located. We set the directions left, up, right and down to the integers 0, 1, 2 and 3, and put these four integers into a Boolean type array list. Then, we use “if-else”, Boolean and array list methods to judge the ship’s direction. Also, the start coordinates of the ship is random.

The Ship class is used to determine ship’s size and health (We didn’t use ship’s name and type in Demo1). If we hit the ship for once in the game, then the ship will lose one health.

The Start class is used for the main class in Demo1 (We will change a lot in the future Demos, this class is only for testing in Demo1). We set the game board to the size 10 \* 10. There is only one ship on the board, its size is 4, and its health is 2. The location of the ship is random. For testing, the ship’s locating coordinates will be printed on the console. When we play this game, if we input the right coordinates, the console will tell us “HIT!!!”, it means we have already hit the ship. If we input the wrong coordinates, the console will tell us “Miss!!!”, it means we did not hit the ship. After the ship is hit for twice, the ship will sink, and the console will tell us “The ship XXX is damaged”.

Demo 2:

For Demo 2, we separated the PlaceShip class into two classes. The first one is ConfirmDirection class, the second one is Locations class. Also, we put the setter and getter methods of start coordinates into Ship class.

The Board class is the same as Demo 1, it is used to create a game board. It includes the variables mapWidth and mapHeight, they represent the board’s width and height. We set the default width and height to both 10.

The Ship class is used to set the ship’s size, health and name. It also sets the ship’s start coordinates of location on the game board. The start coordinates of ship’s location are randomly created by the computer. The ship’s size and start coordinates of location will be used in ConfirmDirection class and Locations class. There is an array type variable “location” in the Ship class, it will be used in Locations class to set ship’s location. The int type variable “direction” will be used in ConfirmDirection class to decide the ship’s direction on the game board.

The ConfirmDirection class includes four static int type values. We set the integers 1, 2, 3 and 4 to represent the directions left, down, right and up on the game board. These four directions represent the ship’s direction on the game board. The int type variable “finalDirection” will be used in the Main class. The judgement of directions is depending on the start coordinates of ship’ location, and ship’s size. It is used to confirm that the ship is located on the game board.

The Locations class is used to get all coordinates of ship’s location, which is based on the “finalDirection” variable from ConfirmDirection class. The location of ship is an array type variable. It will be used in Main class.

The ClassFactory class is used to manage all classes and all methods in this program. In this way, we can get and use the methods in all classes from this class directly. It is easier for us to create the Main class for GUI.

The Main class is used to create a GUI version for our game. We set a ship with 4 size and 4 health on the game board. The board is made up of 10 \* 10 buttons on an interface. The ship’s location will be set into four buttons randomly, and those four buttons’ coordinates will be printed out on the console for testing. Once we press one of those four buttons, the console will print out “HIT!!!”. If we press other buttons, the console will print out “MISS!!!”. Once we have already press any of those four buttons for four times, the console will print out “The ship XXX has been damaged!”, then the game interface will be closed. It is the same as we play this game with computer, because the ship is located randomly by the computer.

Demo 3/Final verision:

In Demo 3, we have already rebuilt the whole program, this is because we need to add more functions into the program. If we did not rebuild the program, it would be harder for us to add more functions from Demo 2, because we did not have a good class diagram in Demo 1 and Demo 2. Now, in Demo 3, we can already place two or more ships on the game board. We do not have to check if we have already hit the ship from the console printing, because if we press the right buttons which the ships locate on them, the buttons will show an “X”, otherwise the buttons will show an “O” which means we did not hit the ship on the game board. But for testing, the ships coordinates will still be printed on the console. Once we have already damaged all ships, the game will be closed automatically. Also, the ships’ locations are randomly on the game board. We added the AI player to our game, now, the game process would be like this: 1) The computer randomly places two ships on the board. 2) The player guesses the location of the ships by clicking on one of the 100 buttons. 3) The AI player guess the location by randomly choosing one of the buttons. 4) The player and the AI player take their turn guessing the location of the two ships. 5) The game will end automatically when all the ships sank. Additionally, added the numbered borders and a "Ship updates" to the GUI class where it just shows the ship and the health it has left. The “Ship updates” will occur on the board once the player or the AI player hit one ship.