Juan Cruz-Tapia

Project 2

Battleship Game

The battleship game is an object oriented program. It uses two class objects one derived from the other and a template. In the main file it has five functions to perform different tasks. First the program ask the user for three inputs: the number of rows in the board, the number of columns and the number of ships. Then the program initializes a class object from the Players class to store those values. The Player class is a derived class form the base class Board. After storing the user inputs in the Board class the program runs an exception to check for errors in the inputs. Then the program prints a reference board using the user inputs and calls the functions getPlayer from the Players class to get the positions in the board to place the ships, it places a “O” in those places. It does the same with the enemy board calling the getEnemy function but the positions are randomly selected.

The program then runs a while loop. Inside the loop a new player and enemy board is printed. The player board displays the ships and the enemy board is blind. Coordinates are requested to launch attack into the board, the player enters inputs for the rows and the columns. Then the program uses those numbers in the enemy array to check if there are any ships in that position. If there is a ship that spot is marked with the “#” symbol and a one is added to a counter called playerPoints, the program keeps track of the number of ships destroyed. If the attacked space is blank an “X” is placed in that position. The same task is performed with the player array but the numbers are randomly generated.

When the number of points for the player or the enemy is the same as the number of ships the while loop breaks and the program ends.

Board Class

Board
Member Variables
int rows;
int columns;
int ships;
Member Functions
Board();
Board(int,int,int);
class InvalidInput{}
void setRows(int);
void setColumns(int);
void setShips(int);
int getRows() const;
int getColumns() const;
int getShips() const;

The member variables are the inputs from the user requested in main. The InvalidInput is an exception to check for invalid input, it is thrown if the number of ships surpasses the number of positions in the board. The setRows, setColumns, setShips are mutators to store the member variables values individually. The getRows, getColumns and getShips are accessors to get the member variables values.

Players Class

Players

Member Variables
char **playerBoard;
char **enemyBoard;
Member Functions
Players(){}
Players(int r, int c, int s) : Board(r,c,s){}
~Players()
{
delete [] playerBoard;
delete [] enemyBoard;
}
char **getPlayer();
char **getEnemy();

The member variables hold a pointer array that has the player and the enemy ship positions. The constructor takes the input from the object declaration and transfers it to the Board class using its own constructor. It has a terminator to delete the variables after execution. The **getPlayer uses the member variables of the Board object and creates a pointer array, the ship positions are stored in that array. The function then returns that array. The **getEnemy function does the same as the **getPlayer function but the positions of the ships are randomly generated without user input.

CheckWinner Template

CheckWinner

Member Variables

```
 int plPoints;
 int enPoints;
```

Member Functions

```
CheckWinner()
{
  plPoints = 0;
  enPoints = 0;
}
CheckWinner(int pl, int en){ plPoints = pl; enPoints = en;}
void setPlPoints(int pl){plPoints = pl;}
void setEnPoints(int en){enPoints = en;}
void getResult();
```

The variables store the points achieved by the user and the enemy during the game. The constructor get those values the the object is created. SetPlPoints and setEnPoints are mutators that allow the user to input the variable values individually. The function getResul compares the number of points of the player and the enemy and outputs whoever the winner is.