**Project 2**

**<Battleship: The Board Game>**

**CIS-17A 48290**

**Name: Baca, Daniel**

**Date: 12/16/2022**

**Introduction**

Title: Battleship - The Board Game

The board game, Battleship, has been around since 1967, inspired by the events of the time with World War I and II still lingering in the minds of the people, the game started off as a pencil and paper game that children would play at their leisure. Over the course of the past 50 years, the game has seen countless adaptations and new ways to play. My first introduction with the game was with the modern Hasbro set of the classic board game, given to me and my brother as a gift for Christmas at a young age. Since then, I have had fond memories of the game and when given the opportunity to recreate the game from scratch using only the power of modern-day C++ programming I stood up to the challenge.

**Summary**

Project size: 1200 + Lines

The number of variables: 22

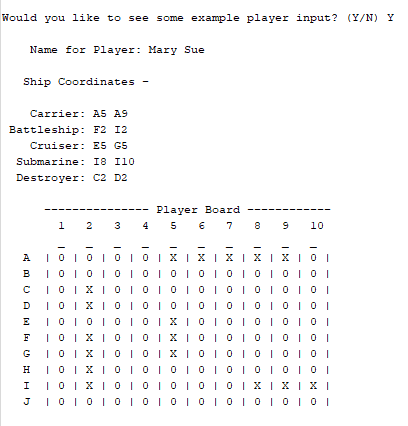
Overall, I am satisfied with how the second project has turned out. It certainly was an entirely new challenge this time, reconstructing the already existing code and its pieces so that it would still at end of the day play and function as a game of Battleship. I was not able to hit every topic on the checklist, even now with the inclusion of classes, but I did hit a majority of them and managed to implement each of them in a way I felt like made sense. The entire project took me more or less two full weeks again to repurpose, as I now had to integrate classes and rework all of the structures and already existing functions to become entirely new classes. Working on different versions and iterating on what would eventually become the complete thing was all too familiar from the last project. It took a good amount of time simply re-establishing the foundation for the way the game would work, but with time I managed to get there piece by piece and eventually created another fully functional game of Battleship, only now introducing classes into the mix.

**Description**

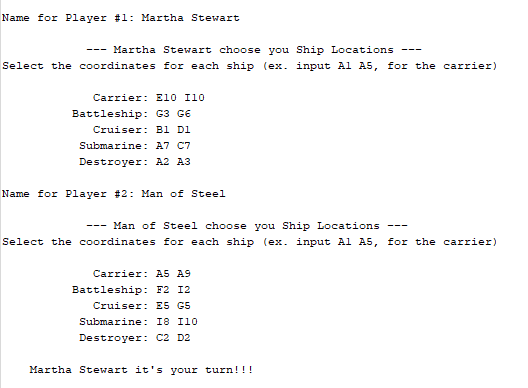
The layout of the program is more or less the same as before, the program begins by using a function called, game\_start(), in this function both players can input their names, enter their ship’s coordinates and also see example inputs for each portion. The function then returns to main where the function ,coor\_to\_num() from the Board Class, converts the input of ship coordinates from each player into usable number coordinates to use for the program and the rest of the game. The next function, numToboard() from the Board Class, takes these recently converted number coordinates and then uses them to accurately place each of the ships onto the player’s boards. From here the program will loop between four functions, disp\_board1(), shooting1(), disp\_board2(), and shooting2() , all also in the Board class, essentially having each player keep shooting each other’s boards, displaying to the players their updated boards and then indicating where and what shots have been taken. Unlike in the first project the players move’s will be recorded but will not be recorded into any files that the player can review afterwards. The players will still be asked if they would like a recap of information based on the game that was just played, if they would like to see it then these outputs will be shown.

**Sample Inputs / Outputs**

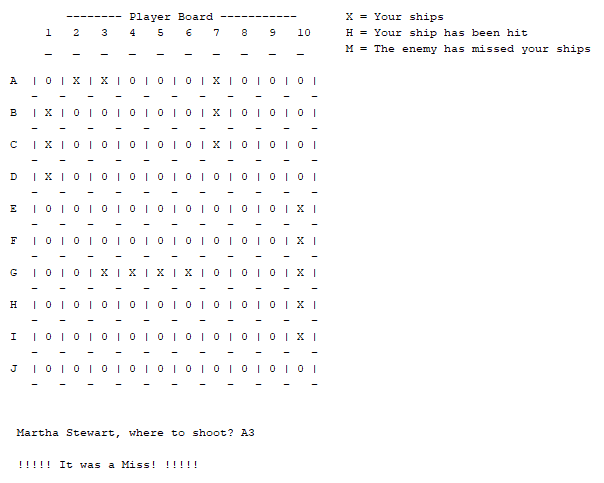
When you first start the game it will prompt the player, asking if they would like to see some example input for how to input their name and ship coordinates, the image below shows exactly this.



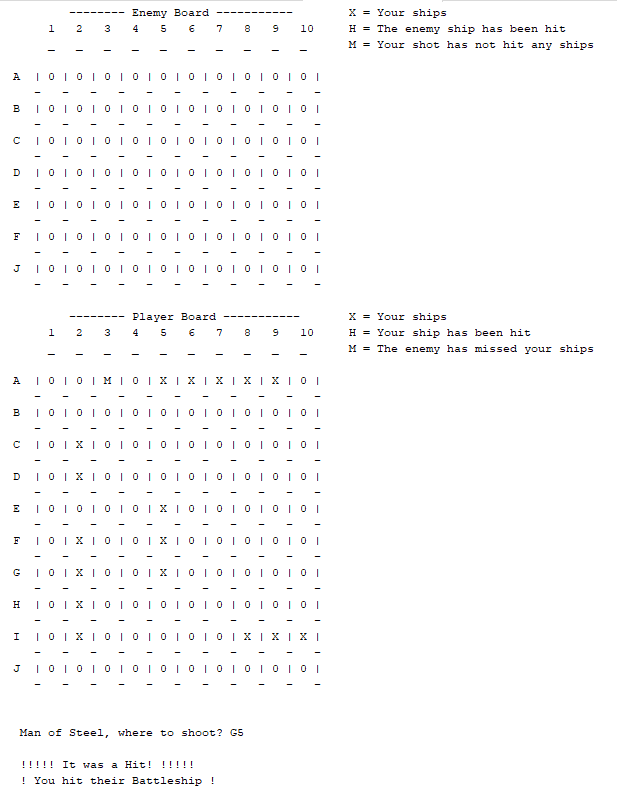
After the user is prompted about the example input, the two players can then input their names and coordinates. Names are written normally using the keyboard, but ship coordinates must be in the format, capital letter and number, corresponding to the way Battleship is normally played on a grid with the rows and columns being assigned letters and numbers. Example inputs are show in the following image.



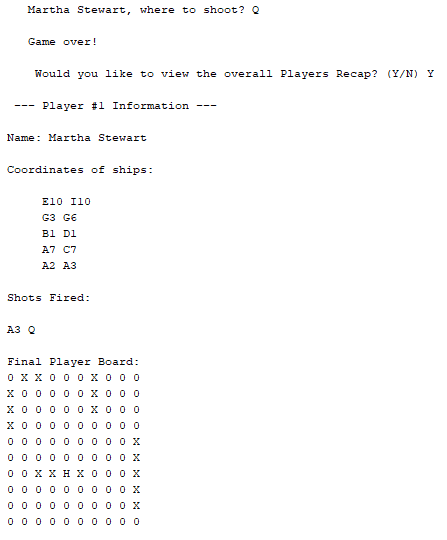
This image shows the act of the individual user shooting a shot at the enemy and then missing.



The following image shows the entire board when it is a players turn, they will be able to see both their player board and the enemy board where shots the player has made on the other player will be displayed and updated with every turn. Also when shooting, if the player has landed a shot on one of the enemy’s ships then it will indicated which ship was hit, just like in the real life game.



Lastly, once the game has ended either from one of the players sinking all of their opponents ships or voluntairly by quitting the game as done in the following image, the player will be prompted if they would like to view the endgame player recap information, if yes, then the players can view their names, ship coordinates, shots fired and even their final player boards.



**UML**

A picture containing graphical user interface

Description automatically generated

**Versions**

**Version 6:** This is the first version after starting this second project and this version is mostly me trying to plan out the reconstruction of the already existing code to also accommodate classes and all of those concepts. I would begin by making a Player Class and a Board class inherited from the Player Class, much of the old code and functions from the first project were stored in the Board Class for later use and repurposing. In main I changed the game\_start() function, it would now use exceptions as input validation for bad player ship coordinates input. Aside from that I also moved many of the variables used originally in the structures from the first project and divided them up between the Player and Board Classes. The program when ran doesn’t do much aside from ask to see if the player wants to see example output, collect player’s names and take ship coordinates from each player.

**Version 7:** This version of the code is a bit more fleshed out, it still has the same classes as before only now I try to implement an abstract class that the Player and Board classes can be based off of. Also in this version I realize that for the arrays that I need to use to run the game I will need to make them static so that they can remain consistent between objects, hence the large list of static arrays that are present in the Board .h file. I’ve also now gotten the actual member functions working with the static arrays so now the game should be functional, yet there are still some imperfections. The player class remains largely untouched in the meanwhile as does everything in main.

**Version 8:** In this version I now begin to separate everything related to the classes into their own .cpp files, whereas before they were only found in the same classes .h files. I have also now overloaded three operators, two of which are for the Board class and one for the Choice class. These three overloaded operators were ==, != and << respectively. Speaking of the Choice class, it is an aggregated class utilized in the Board Class to get the choice made at the beginning of the game for whether or not example output should be shown. The program still runs as expected being a mostly functional game of battleship between two players.

**Version 9:** This version of the code is the most refined version yet, nearly everything I could’ve possibly wanted to implement now exists and has been tested to be fully functional, I’ve now commented a lot of the code, organized and cleaned up essentially everything that is already here. Noticeable, the program doesn’t run all of the way in this version but that will be an issue that I resolve in the last and final version of the code.

**Version 10:** The finalized version of the code, the only noticeable changes are that everything is now fully commented, everything is now organized, all of the elements I wanted to have are here and also the Choice class has been reworked a bit to now become a template class. Aside from that the program is fully operational and is able to play entire normal games of Battleship with ease.

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |
| Integer | ship\_coors1[][] | 2D Integer array holding numeric ship coordinates | Lines: 28 (Board.h) |
| Integer | ship\_coors2[][] | 2D Integer array holding numeric ship coordinates | Lines: 30 (Board.h) |
| Integer | count1 | Counter for the number of times Player 1 has shot at Player 2 | Lines: 43 (Board.h) |
| Integer | count2 | Counter for the number of times Player 2 has shot at Player 1 | Lines: 45 (Board.h) |
| Integer | num[] | Array holding the converted user number coordinates | Lines: 169 and 310 (shooting1() and shooting2() in Board.cpp) |
| Integer | hits | Number of ships coordinates not yet shot | Lines: 170 and 311 (shooting1() and shooting2() in Board.cpp) |
| Integer | count | Counter used to determine what kind of ship was hit | Lines: 171 and 312 (shooting1() and shooting2() in Board.cpp) |
| Integer | numshps | Integer number of ships | Lines: 143 (main.cpp) |
| Char | player\_name[] | C-string array holding the names of each player | Lines: 27 (Player.h) |
| Char | player\_board1[][] | C-string 2D array holding the players board moves | Lines: 35 (Board.h) |
| Char | enemy\_board1[][] | C-string 2D array holding the enemy board moves | Lines: 41 (Board.h) |
| Char | player\_board2[][] | C-string 2D array holding the players board moves | Lines: 39 (Board.h) |
| Char | enemy\_board2[][] | C-string 2D array holding the enemy board moves | Lines: 37 (Board.h) |
| Char | choice | The character input for whether for not to have endgame review | Lines: 72 (main.cpp) |
| <char> | response | The character input for whether or not to display example input | Lines: 27 (Choice.h) |
| Char | letters[] | Char array for letters on the game board | Lines: 450 and 503 (disp\_board1() and disp\_board2() in Board.cpp) |
| String | ships[] | String array holding the ship coordinates input by the player | Lines: 28 (Player.h) |
| String | shts\_frd[] | Dynamic String Array holding the shots fired by the player | Lines: 32 (Board.h) |
| String | ship\_name[5] | String Array holding ship names | Lines: 148 (main.cpp) |
| Bool | gme\_ovr | Bool variable gotten from the shooting functions telling whether or not the game has ended | Lines: 70 (main.cpp) |
| Bool | tryagain | Bool used for input validation loop | Lines: 150 (main.cpp) |
| Board | play[2] | Object array used to play game for two players | Lines: 73 (main.cpp) |

**C++ Concepts**

|  |  |  |
| --- | --- | --- |
| Chapter | New syntax and Keywords | Location |
| 13 | Instances of a Class –  Board play[2];  Choice<char> output; | Main.cpp – Line.73  Board.h – Line.34  (Classes in general found in AbsPlayer.h, Board.h, Choice.h, Player.h files) |
| Private Data Members –  static int ship\_coors1[20][20];  static int ship\_coors2[20][20];  string \* shts\_frd = new string[100];  Choice<char> output;  static char player1\_board[10][10];  static char enemy2\_board[10][10];  static char player2\_board[10][10];  static char enemy1\_board[10][10];  static int count1;  static int count2;  T response; | Board.h – Lines. (27 – 46)  Choice.h – Lines. (26 – 27) |
| Specification vs. Implementation –  Player.h and Player.cpp files  Board.h and Board.cpp files  Choice.h and Choice.pp files | View AbsPlayer.h,Player.h,Player.cpp, Board.h, Board.cpp, Choice.h, Choice.cpp files |
| Inline –  (found in Player.h file)  void setplyrnm(string pn)  {  player\_name = pn;  }    void setships(string s[]);    //Get player name function  string getplyrnm()  {  return player\_name;  }  (found in Board.h file)  void setplyrnm(string pn)  {  player\_name = pn;  }  string getplyrnm()  {  return player\_name;  }    void setchoice(char c)  {  cout << output;  output.respond(c);  } | Player.h – Lines. (62 – 76)  Board.h – Lines. (151 – 168)  Choice.h – Lines. (33 – 95) |
| Constructors –  (found in Player.h file)  Player()  {  player\_name = " ";    for(int i=0; i<5;i++)  { ships[i] = " "; }  }    Player(string pn)  {  player\_name = pn;    for(int i=0; i<5;i++)  { ships[i] = " "; }  }    Player(string pn, string s[])  {  player\_name = pn;    for(int i=0; i<5;i++)  { ships[i] = s[i]; }  } | Player.h – Lines. (31 – 56)  Board.h – Lines. (48 – 76)  Choice.h – Lines. (29 – 31) |
| Destructors –    (found in Board.h file)  ~Board()  {  delete[] shts\_frd;  } | Board.h – Lines. (124 - 128) |
| Arrays of objects –  (found in main.cpp)  Board play[2];  if(play[0] == play[1])  else if(play[0] != play[1]) | Main.cpp – Lines. 73, 79, 84, 90, 91 as some examples of an array being used |
| UML | See the image in the previous section for the UML Diagrams |
| 14 | Static –  (found in Board.h file)  private:  static int ship\_coors1[20][20];  static int ship\_coors2[20][20];  string \* shts\_frd = new string[100];  Choice<char> output;  static char player1\_board[10][10];  static char enemy2\_board[10][10];  static char player2\_board[10][10];  static char enemy1\_board[10][10];  static int count1;  static int count2; | Board.h – Lines. (28 - 31), (35 – 46)  Board.cpp – Lines. (18 - 26) |
| Friends –  (found in Choice.h file)  friend ostream& operator<<(ostream& os, Choice<char> &same); | Choice.h – Lines. 98  Choice.cpp – Lines. (9 - 26) |
| Copy Constructors –  (found in Board.h file)  Board(Board &obj)  {  shts\_frd = new string[100];    for(int i=0;i<100;i++)  shts\_frd[i] = obj.shts\_frd[i];  } | Board.h – Lines. (130 - 137) |
| Operator Overloading –  (found in Board.h file)  bool operator==(Board &same)  {  int num = 0;  if(player\_name==same.player\_name)  {  for(int i=0;i<5;i++)  {  if(ships[i]==same.ships[i])  num++;  }  if(num == 5)  return true;  }  return false;  } | Board.h – Lines. (78 - 122), (163 – 168)  Main.cpp – Lines. (79 – 84)  Choice.h – Lines. 98  Choice.cpp – Lines. (9 - 26) |
| Aggregation –  (found in Board.h file)  Choice<char> output;  void setchoice(char c)  {  cout << output;  output.respond(c);  } | Board.h – Lines. 34, (163 – 168)  Aggregation of the Choice class in the Board Class |
| 15 | Protected Members –  (found in Player.h file)  protected:  string player\_name; //String Array holding the name of the player  string ships[5]; | Player.h – Lines. (26 – 29) |
| Base Class to Derived –  (found in Player.h and Board.h files)  class Player : public AbsPlayer  class Board : public Player | Player.h – Lines. 24  Board.h – Lines. 25 |
| Polymorphic Associations – | N/A |
| Abstract Classes –  (found in AbsPlayer.h file)  class AbsPlayer  {  public:    //Virtual function for setplyrnm function  virtual void setplyrnm(string) = 0;    //Virtual function for getplyrnm function  virtual string getplyrnm() = 0;  }; | View the AbsPlayer.h file  and Player.h – Lines. 24 |
| 16 | Execptions –  try  {  play[i].setships(ships);  tryagain = false;  }  catch(Player::BadShips)  {  cout << endl << " Please follow the input format for the ship "  << "coordinates. (ex. B3 B7)" << endl << endl;  for(int j=0;j<numshps;j++)  {  cout << setw(20) << ship\_name[j] << ": ";  getline(cin,ships[j]);  }  } | Main.cpp – Lines. (191 – 214)  Player.h – Lines. (58 – 60)  Player.cpp – Lines. (18 – 30) |
| Templates –  (found in Choice.h file)  template <class T>  class Choice  (found in Choice.cpp file)  ostream& operator<<(ostream& os, Choice<char> & same)  (found in Board.h file)  Choice<char> output; | Choice.h – Lines. (22 – 99)  Choice.cpp – Line. 10  Board.h – Lines. 34 |
| STL | N/A |

**References**

1. Gaddis, Tony “Starting Out with C++ from Control Structures to Objects – 9th Edition”
2. Various C++ Programming Websites (<https://www.geeksforgeeks.org/>, <https://stackoverflow.com/>, <https://cplusplus.com/> )
3. Source Used as Reference Material for Rules of Battleship Board Game: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi1qd2g5IT7AhX0MUQIHVIADQUQFnoECBQQAQ&url=https%3A%2F%2Fwww.hasbro.com%2Fcommon%2Finstruct%2Fbattleship.pdf&usg=AOvVaw2XhO4v8ZigPERaaI8L0_K4>

**Program**

/\*

\* File: main.cpp

\* Author: Daniel Baca

\* Created on December 4th, 2022, 7:49 PM

\* Purpose: Version 10 (Final Version) - Battleship

\*

\* Create a functional game of BattleShip, implementing many of the programming

\* concepts from Chapter 13 to 16, while following the rules of the original

\* board game which are:

\*

\* 1. Players place their 5 ships on their player board, the rules here are:

\*

\* - Place each ship in any horizontal or vertical position, but not

\* diagonally.

\* - Do not place a ship so that it overlaps the edge of the board or another

\* ship.

\* - Do not change the position of any ship once the game has begun.

\*

\* 2. Decide who will go first, each player will alternate turns, calling out

\* one shot per turn to try and hit each other's ships.

\*

\* 3. On your turn, pick a target hole on your enemy board and call out its

\* location by the letter and number. Each target hole has a letter-number

\* coordinate that corresponds with the same coordinate on your

\* enemy's board. To determine each coordinate, find its

\* corresponding letter on the left side of the target board and its number

\* on the top of the board. For example, when you call a shot, your opponent

\* must tell you whether your shot is a hit or a miss.

\*

\* 4. It's a Hit - If you call out a shot location that is occupied by a ship on

\* your enemy's board, your shot is a hit! Your opponent tells you which ship

\* you have hit (cruiser, submarine, etc.). Your shots will be recorded for

\* you on each of the boards. Your opponent's board will be updated in the

\* corresponding hole of the ship you have hit on his or her board.

\*

\* 5. It's a Miss - If you call out a shot location not occupied by a ship on

\* your opponent's board, it's a miss. The board will record this move, so

\* you cannot call this shot again.

\*

\* Play continues in this manner, with you and your opponent calling one shot

\* per turn.

\*

\* 6. Once all the holes in any one ship are hit then the ship is considered

\* sunk. The owner of the ship must announce which ship was sunk.

\*

\* 7. If you're the first player to sink your opponents entire fleet of 5 ships,

\* you win the game!

\*

\*/

//System Libraries

#include <iostream>

#include <iomanip>

#include <string>

#include <ctime>

#include <cstdlib>

#include <cmath>

#include <fstream>

#include <cstring>

#include "Board.h"

using namespace std;

//Function Prototype

void game\_start(Board []); //Get both player's names and ship coordinates

//Execution of Code Begins Here

int main(int argc, char\*\* argv)

{

//Declare all variables

bool gme\_ovr; //The bool variable gotten from the shooting functions

//telling whether or not the game has ended or not

char choice; //Char variable for the choice

Board play[2]; //Array of object variables used for the game

//Receive both player's starting information

game\_start(play);

//Check to see if each player has unique names and coordinates

if(play[0] == play[1])

{

cout << endl << " You cannot have the same names and ship "

<< "coordinates. Please Try Again." << endl;

}

else if(play[0] != play[1])

{

cout << endl << " Name and Coordinates look good to go..." << endl;

//Set the player coordinates

play[0].coorsTonum1();

play[1].coorsTonum2();

//Set the players board up

play[0].numToboard1();

play[1].numToboard2();

//Loop for each player's turn, shooting their shots until someone wins

while(!gme\_ovr)

{

//Start Player 1's turn

cout << endl << " " << play[0].getplyrnm() << " it's your turn!!!"

<< endl << endl;

play[0].disp\_board1();

gme\_ovr = play[0].shooting1();

//End game if necessary

if(gme\_ovr)

break;

//Start Player 2's turn

cout << endl << " " << play[1].getplyrnm() << " it's your turn!!!"

<< endl << endl;

play[1].disp\_board2();

gme\_ovr = play[1].shooting2();

}

//Prompt players to input choice for Players Info

cout << " Would you like to view the overall Players Recap? (Y/N) ";

cin >> choice;

//Use choice input to determine what to input for end of program

if(choice == 'Y' || choice == 'y')

{

play[0].results1();

play[1].results2();

cout << endl << " Thank you for Playing!!! " << endl;

}

else if(choice == 'N' || choice == 'n')

cout << endl << " Thank you for Playing!!! " << endl;

}

//Exit Stage Right

return 0;

}

//Function for receiving all player information including name and ship

//coordinates

void game\_start(Board play[])

{

//Declare Variables

char choice; //Char variable holding player choice

int numshps = 5; //Number of ships per player

string player\_name; //String Array holding the name of the player

string ships[5]; //String Array holding the ship coordinates input

//by the player

//String array holding the ship names

string ship\_name[5] = {"Carrier", "Battleship", "Cruiser",

"Submarine", "Destroyer"};

bool tryagain = true; //Bool variable for exception loop

//Ask player if they would like to see example input

cout << endl << " Would you like to see some example player input? (Y/N) ";

cin >> choice;

cout << endl;

//Output player choice response

play[0].setchoice(choice);

//Formatting

cin.ignore(1);

//Loop to get input for each player

for(int i=0; i<2;i++)

{

// Get Name of Player

cout << endl << "Name for Player #" << (i+1) << ": ";

getline(cin, player\_name);

cout << endl;

//Set player name to corresponding player object

play[i].setplyrnm(player\_name);

//Prompt user to input ship coordinates

cout << " --- " << play[i].getplyrnm()

<< " choose you Ship Locations ---" << endl;

cout << "Select the coordinates for each ship "

<< "(ex. input A1 A5, for the carrier)" << endl << endl;

//Loop for ship coordinates input

for(int j=0;j<numshps;j++)

{

//Input the ship coordinates

cout << setw(20) << ship\_name[j] << ": ";

getline(cin,ships[j]);

}

//Reset the bool variable for loop

tryagain = true;

//Catch exceptions and mistakes in ship coordinate input

while(tryagain)

{

try

{

//Set the ships coordinates into corresponding player object

play[i].setships(ships);

tryagain = false;

}

catch(Player::BadShips)

{

//Prompt the user to enter valid input

cout << endl << " Please follow the input format for the ship "

<< "coordinates. (ex. B3 B7)" << endl << endl;

//Loop for ship coordinates input

for(int j=0;j<numshps;j++)

{

//Input the ship coordinates

cout << setw(20) << ship\_name[j] << ": ";

getline(cin,ships[j]);

}

}

}

}

}

/\*

\* File: AbsPlayer.h

\* Author: Daniel Baca

\*

\* Created on November 26, 2022, 9:28 PM

\*/

#ifndef ABSPLAYER\_H

#define ABSPLAYER\_H

#include <string>

#include <cstring>

using namespace std;

//Abstract class for all player objects created and then inherited

class AbsPlayer

{

public:

//Virtual function for setplyrnm function

virtual void setplyrnm(string) = 0;

//Virtual function for getplyrnm function

virtual string getplyrnm() = 0;

};

#endif /\* ABSPLAYER\_H \*/

/\*

\* File: Player.h

\* Author: Daniel Baca

\*

\* Created on November 23, 2022, 5:00 PM

\*/

#ifndef PLAYER\_H

#define PLAYER\_H

//System Libraries

#include "AbsPlayer.h"

#include <iostream>

#include <iomanip>

#include <string>

#include <ctime>

#include <cstdlib>

#include <cmath>

#include <fstream>

#include <cstring>

using namespace std;

//Player class holding basic info for each player as the game goes on

class Player : public AbsPlayer

{

protected:

string player\_name; //String Array holding the name of the player

string ships[5]; //String Array holding the ship coordinates input

//by the player

public:

//Default Constructor

Player()

{

player\_name = " ";

for(int i=0; i<5;i++)

{ ships[i] = " "; }

}

//Initial Constructor

Player(string pn)

{

player\_name = pn;

for(int i=0; i<5;i++)

{ ships[i] = " "; }

}

//Initial Constructor

Player(string pn, string s[])

{

player\_name = pn;

for(int i=0; i<5;i++)

{ ships[i] = s[i]; }

}

//Exception class

class BadShips

{ };

//Set player name function

void setplyrnm(string pn)

{

player\_name = pn;

}

//Function to set the ship coordinates

void setships(string s[]);

//Get player name function

string getplyrnm()

{

return player\_name;

}

};

#endif /\* PLAYER\_H \*/

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

#include "Player.h"

#include <iostream>

#include <iomanip>

#include <string>

#include <ctime>

#include <cstdlib>

#include <cmath>

#include <fstream>

#include <cstring>

using namespace std;

//Set player ship coordinates function

void Player::setships(string s[])

{

//Verify to make sure input is valid

for(int i=0;i<5;i++)

{

if(s[i].length() > 7 || s[i].length() <= 4 ||

!isupper(s[i][0]) || !isdigit(s[i][1]))

throw BadShips();

else

ships[i] = s[i];

}

}

/\*

\* File: Board.h

\* Author: Daniel Baca

\*

\* Created on November 23, 2022, 8:39 PM

\*/

#ifndef BOARD\_H

#define BOARD\_H

//System Libraries

#include "Player.h"

#include "Choice.h"

#include <iostream>

#include <iomanip>

#include <string>

#include <ctime>

#include <cstdlib>

#include <cmath>

#include <fstream>

#include <cstring>

using namespace std;

//Board class holding board related info for each player as the game goes on

class Board : public Player

{

private:

static int ship\_coors1[20][20]; //Player 1's number Array with the number

//coordinates converted from user input

static int ship\_coors2[20][20]; //Player 2's number Array with the number

//coordinates converted from user input

string \* shts\_frd = new string[100]; //Dynamic String Array holding the

//shots fired by the player

Choice<char> output; //Aggregated class for initial output response

static char player1\_board[10][10]; //C-string array holding Player 1's player

//board moves

static char enemy2\_board[10][10]; //C-string array holding Player 2's enemy

//board moves

static char player2\_board[10][10]; //C-string array holding Player 2's player

//board moves

static char enemy1\_board[10][10]; //C-string array holding Player 1's enemy

//board moves

static int count1; //Counter used to keep track of how many shots Player 1

//has made

static int count2; //Counter used to keep track of how many shots Player 2

//has made

public:

//Initial Constructor

Board()

{

//Initialize the ship coordinates

for(int i=0;i<20;i++)

for(int j=0;j<20;j++)

ship\_coors1[i][j]=0;

for(int i=0;i<20;i++)

for(int j=0;j<20;j++)

ship\_coors2[i][j]=0;

//Initialize the player boards

for(int i=0;i<10;i++)

for(int j=0;j<10;j++)

player1\_board[i][j]='0';

for(int i=0;i<10;i++)

for(int j=0;j<10;j++)

enemy1\_board[i][j]='0';

for(int i=0;i<10;i++)

for(int j=0;j<10;j++)

player2\_board[i][j]='0';

for(int i=0;i<10;i++)

for(int j=0;j<10;j++)

enemy2\_board[i][j]='0';

}

//Overloaded == operator

bool operator==(Board &same)

{

int num = 0; //Temporary counter variable

//Determine if the player names are the same

if(player\_name==same.player\_name)

{

//Check to see if the ships coordinates are the same

for(int i=0;i<5;i++)

{

if(ships[i]==same.ships[i])

num++;

}

//Return true if the two objects have the same values

if(num == 5)

return true;

}

//Return false if the two objects don't have the same values

return false;

}

//Overloaded != operator

bool operator!=(Board &same)

{

int num = 0; //Temporary counter variable

if(player\_name!=same.player\_name)

return true;

for(int i=0;i<5;i++)

{

if(ships[i]==same.ships[i])

num++;

}

//Return true if the two objects don't have the same values

if(num < 5)

return true;

//Return false if the two objects do have the same values

return false;

}

//Destructor

~Board()

{

delete[] shts\_frd;

}

//Copy Constructor

Board(Board &obj)

{

shts\_frd = new string[100];

for(int i=0;i<100;i++)

shts\_frd[i] = obj.shts\_frd[i];

}

//Function prototypes

void coorsTonum1();

void coorsTonum2();

void numToboard1();

void numToboard2();

bool shooting1();

bool shooting2();

void disp\_board1();

void disp\_board2();

void results1();

void results2();

//Set player name function

void setplyrnm(string pn)

{

player\_name = pn;

}

//Get player name function

string getplyrnm()

{

return player\_name;

}

//Setchoice function to get player initial response

void setchoice(char c)

{

cout << output;

output.respond(c);

}

};

#endif /\* BOARD\_H \*/

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

#include "Board.h"

#include <iostream>

#include <iomanip>

#include <string>

#include <ctime>

#include <cstdlib>

#include <cmath>

#include <fstream>

#include <cstring>

using namespace std;

//Static Variable initializations

int Board::ship\_coors1[20][20] = {};

int Board::ship\_coors2[20][20] = {};

char Board::player1\_board[10][10] = {};

char Board::enemy2\_board[10][10] = {};

char Board::player2\_board[10][10] = {};

char Board::enemy1\_board[10][10] = {};

int Board::count1 = 0;

int Board::count2 = 0;

//Function used to convert Player 1's string input to usable int number

//coordinates for the rest of the program

void Board::coorsTonum1()

{

for(int i=0;i<5;i++)

{

//See if the string is a certain length and choose option based on it

if(ships[i].length() == 5 || ships[i].length() == 6)

{

for(int j=0;j<5;j++)

{

// Determine if the portion of the string is a letter and convert to

// an int

if(isalpha(ships[i][j]))

ship\_coors1[i][j] = ships[i][j] - 64;

// Determine if the portion of the string is a number and convert to

// an int

if(isdigit(ships[i][j]) && ships[i][j+1] == '0')

{

ship\_coors1[i][j] = 10;

break;

}

else if(isdigit(ships[i][j]))

{

ship\_coors1[i][j] = ships[i][j] - 48;

}

// Determine if the portion of the string is a space

if(isspace(ships[i][j]))

ship\_coors1[i][j] = 0;

}

}

else

{

//Convert string of chars to usable number coordinates

ship\_coors1[i][0] = ships[i][0] - 64;

ship\_coors1[i][3] = ships[i][4] - 64;

ship\_coors1[i][2] = 0;

ship\_coors1[i][1] = 10;

ship\_coors1[i][4] = 10;

}

}

}

//Function used to convert Player 2's string input to usable int number

//coordinates for the rest of the program

void Board::coorsTonum2()

{

for(int i=0;i<5;i++)

{

//See if the string is a certain length and choose option based on it

if(ships[i].length() == 5 || ships[i].length() == 6)

{

for(int j=0;j<5;j++)

{

// Determine if the portion of the string is a letter and convert to

// an int

if(isalpha(ships[i][j]))

ship\_coors2[i][j] = ships[i][j] - 64;

// Determine if the portion of the string is a number and convert to

// an int

if(isdigit(ships[i][j]) && ships[i][j+1] == '0')

{

ship\_coors2[i][j] = 10;

break;

}

else if(isdigit(ships[i][j]))

{

ship\_coors2[i][j] = ships[i][j] - 48;

}

// Determine if the portion of the string is a space

if(isspace(ships[i][j]))

ship\_coors2[i][j] = 0;

}

}

else

{

//Convert string of chars to usable number coordinates

ship\_coors2[i][0] = ships[i][0] - 64;

ship\_coors2[i][3] = ships[i][4] - 64;

ship\_coors2[i][2] = 0;

ship\_coors2[i][1] = 10;

ship\_coors2[i][4] = 10;

}

}

}

//Function used to have the converted coordinates format Player 1's boards

//with each of their ships

void Board::numToboard1()

{

// Initialize empty board layout

for(int i=0;i<10;i++)

for(int j=0;j<10;j++)

{

player1\_board[i][j] = '0';

enemy2\_board[i][j] = '0';

}

// Place X's that represent ships on the player's boards

for(int i=0;i<5;i++)

{

if(ship\_coors1[i][0] == ship\_coors1[i][3])

for(int k=(ship\_coors1[i][1]-1);k<ship\_coors1[i][4];k++)

player1\_board[(ship\_coors1[i][0]-1)][k] = 'X';

else

for(int a=(ship\_coors1[i][0]-1);a<ship\_coors1[i][3];a++)

player1\_board[a][(ship\_coors1[i][1]-1)] = 'X';

}

}

//Function used to have the converted coordinates format Player 2's boards

//with each of their ships

void Board::numToboard2()

{

// Initialize empty board layout

for(int i=0;i<10;i++)

for(int j=0;j<10;j++)

{

player2\_board[i][j] = '0';

enemy1\_board[i][j] = '0';

}

// Place X's that represent ships on the player's boards

for(int i=0;i<5;i++)

{

if(ship\_coors2[i][0] == ship\_coors2[i][3])

for(int k=(ship\_coors2[i][1]-1);k<ship\_coors2[i][4];k++)

player2\_board[(ship\_coors2[i][0]-1)][k] = 'X';

else

for(int a=(ship\_coors2[i][0]-1);a<ship\_coors2[i][3];a++)

player2\_board[a][(ship\_coors2[i][1]-1)] = 'X';

}

}

//Function used to have Player 1 shoot at Player 2, output hits and ships hit or

//missed, and decides if the game is over yet

bool Board::shooting1()

{

//Declare Variables

int num[2]; //Array holding converted user number coordinates

int hits = 0; //Number of ship coordinates not yet shot

int count = 0; //Counter used to determine what kind of ship was hit

// Initial shot user input

cout << endl << " " << player\_name << ", where to shoot? ";

getline(cin,shts\_frd[count1]);

cout << endl;

//Check to see if player wants to quit the game

if(shts\_frd[count1][0] == 'Q' || shts\_frd[count1][0] == 'q')

{

cout << " Game over!" << endl << endl;

return true;

}

// Input Validation for user shot input

while(!isalpha(shts\_frd[count1][0])

|| !isupper(shts\_frd[count1][0])

|| !isdigit(shts\_frd[count1][1]))

{

cout << "Please input a correct coordinate. (ex. A10, B7, I3)" << endl;

cout << endl << " " << player\_name << ", where to shoot? ";

getline(cin,shts\_frd[count1]);

cout << endl;

}

// Convert the input coordinates into usable number coordinates

for(int j=0;j<2;j++)

{

if(isalpha(shts\_frd[count1][j]))

num[j] = shts\_frd[count1][j] - 64;

if(isdigit(shts\_frd[count1][j]) && shts\_frd[count1][j+1] == '0')

{

num[j] = 10;

break;

}

else if(isdigit(shts\_frd[count1][j]))

{

num[j] = shts\_frd[count1][j] - 48;

}

}

// Change the game board to show missed and hit shots and output it

for(int m=0;m<11;m++)

{

for(int l=0;l<11;l++)

{

if(l == num[1] && m == num[0])

{

if(player2\_board[m-1][l-1]== 'X')

{

//Output that a ship has been hit

cout << " !!!!! It was a Hit! !!!!!" << endl;

player2\_board[m-1][l-1] = 'H';

enemy1\_board[m-1][l-1] = 'H';

//Determine what ship has been shot at

for(int x=0;x<5;x++)

{

if(m >= ship\_coors2[x][0] && m <= ship\_coors2[x][3]

&& l == ship\_coors2[x][1] && l == ship\_coors2[x][4])

{

//Length of the ship shot used for output

count = (ship\_coors2[x][3]+1) - ship\_coors2[x][0];

//Output which ship was hit

if(count == 5)

cout << " ! You hit their Carrier !"

<< endl;

else if(count == 4)

cout << " ! You hit their Battleship !"

<< endl;

else if(count == 3)

cout << " ! You hit their Submarine or"

<< " Cruiser !" << endl;

else if(count == 2)

cout << " ! You hit their Destroyer !"

<< endl;

}

else if(l >= ship\_coors2[x][1] && l <= ship\_coors2[x][4]

&& m == ship\_coors2[x][0] && m == ship\_coors2[x][3])

{

//Length of the ship shot used for output

count = (ship\_coors2[x][4]+1) - ship\_coors2[x][1];

//Output which ship was hit

if(count == 5)

cout << " ! You hit their Carrier !"

<< endl;

else if(count == 4)

cout << " ! You hit their Battleship !"

<< endl;

else if(count == 3)

cout << " ! You hit their Submarine or"

<< " Cruiser !" << endl;

else if(count == 2)

cout << " ! You hit their Destroyer !"

<< endl;

}

}

}

else

{

//Output that the shot has missed hitting any ships and

//record it

cout << " !!!!! It was a Miss! !!!!!" << endl;

enemy1\_board[m-1][l-1] = 'M';

player2\_board[m-1][l-1] = 'M';

}

}

}

}

// Detect to see if there are any ships coordinates not shot yet

for(int z=0;z<11;z++)

for(int y=0;y<11;y++)

if(player2\_board[z][y] == 'X')

hits++;

// Condition to see if the game has ended yet

if(hits == 0)

{

cout << endl << " " << player\_name

<< " has won the game!" << endl << endl;

return true;

}

// Increase count for recording shots

count1++;

// Return false if the game hasn't ended

return false;

}

//Function used to have Player 2 shoot at Player 1, output hits and ships hit or

//missed, and decide if the game is over yet

bool Board::shooting2()

{

//Declare Variables

int num[2]; //Array holding converted user number coordinates

int hits = 0; //Number of ship coordinates not yet shot

int count = 0; //Counter used to determine what kind of ship was hit

// Initial shot user input

cout << endl << " " << player\_name << ", where to shoot? ";

getline(cin,shts\_frd[count2]);

cout << endl;

//Check to see if player wants to quit the game

if(shts\_frd[count2][0] == 'Q' || shts\_frd[count2][0] == 'q')

{

cout << " Game over!" << endl << endl;

return true;

}

// Input Validation for user shot input

while(!isalpha(shts\_frd[count2][0])

|| !isupper(shts\_frd[count2][0])

|| !isdigit(shts\_frd[count2][1]))

{

cout << "Please input a correct coordinate. (ex. A10, B7, I3)" << endl;

cout << endl << " " << player\_name << ", where to shoot? ";

getline(cin,shts\_frd[count2]);

cout << endl;

}

// Convert the input coordinates into usable number coordinates

for(int j=0;j<2;j++)

{

if(isalpha(shts\_frd[count2][j]))

num[j] = shts\_frd[count2][j] - 64;

if(isdigit(shts\_frd[count2][j]) && shts\_frd[count2][j+1] == '0')

{

num[j] = 10;

break;

}

else if(isdigit(shts\_frd[count2][j]))

{

num[j] = shts\_frd[count2][j] - 48;

}

}

// Change the game board to show missed and hit shots and output it

for(int m=0;m<11;m++)

{

for(int l=0;l<11;l++)

{

if(l == num[1] && m == num[0])

{

if(player1\_board[m-1][l-1]== 'X')

{

//Output that a ship has been hit

cout << " !!!!! It was a Hit! !!!!!" << endl;

enemy2\_board[m-1][l-1] = 'H';

player1\_board[m-1][l-1] = 'H';

//Determine what ship has been shot at

for(int x=0;x<5;x++)

{

if(m >= ship\_coors1[x][0] && m <= ship\_coors1[x][3]

&& l == ship\_coors1[x][1] && l == ship\_coors1[x][4])

{

//Length of the ship shot used for output

count = (ship\_coors1[x][3]+1) - ship\_coors1[x][0];

//Output which ship was hit

if(count == 5)

cout << " ! You hit their Carrier !"

<< endl;

else if(count == 4)

cout << " ! You hit their Battleship !"

<< endl;

else if(count == 3)

cout << " ! You hit their Submarine or"

<< " Cruiser !" << endl;

else if(count == 2)

cout << " ! You hit their Destroyer !"

<< endl;

}

else if(l >= ship\_coors1[x][1] && l <= ship\_coors1[x][4]

&& m == ship\_coors1[x][0] && m == ship\_coors1[x][3])

{

//Length of the ship shot used for output

count = (ship\_coors1[x][4]+1) - ship\_coors1[x][1];

//Output which ship was hit

if(count == 5)

cout << " ! You hit their Carrier !"

<< endl;

else if(count == 4)

cout << " ! You hit their Battleship !"

<< endl;

else if(count == 3)

cout << " ! You hit their Submarine or"

<< " Cruiser !" << endl;

else if(count == 2)

cout << " ! You hit their Destroyer !"

<< endl;

}

}

}

else

{

//Output that the shot has missed hitting any ships and

//record it

cout << " !!!!! It was a Miss! !!!!!" << endl;

enemy2\_board[m-1][l-1] = 'M';

player1\_board[m-1][l-1] = 'M';

}

}

}

}

// Detect to see if there are any ships coordinates not shot yet

for(int z=0;z<11;z++)

for(int y=0;y<11;y++)

if(player1\_board[z][y] == 'X')

hits++;

// Condition to see if the game has ended yet

if(hits == 0)

{

cout << endl << " " << player\_name

<< " has won the game!" << endl << endl;

return true;

}

// Increase count for recording shots

count2++;

// Return false if the game hasn't ended

return false;

}

// Function used to display the boards for Player 1 on their turn

void Board::disp\_board1()

{

// The char array for letters in the game board

char letters[10] = {'A','B','C','D','E','F','G','H','I','J'};

// Format the output and general layout of the game boards

cout << " -------- Enemy Board ----------- "

<< "X = Your ships" << endl

<< " 1 2 3 4 5 6 7 8 9 10"

<< " H = The enemy ship has been hit" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

<< " M = Your shot has not hit any ships" << endl << endl;

// Loop for more accurate board displays

for(int x=0;x<10;x++)

{

cout << " " << letters[x] << " ";

for(int z=0;z<10;z++)

cout << "| " << enemy1\_board[x][z]<< " ";

cout << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

//Formatting

cout << endl;

// Format the output and general layout of the game boards

cout << " -------- Player Board ----------- "

<< "X = Your ships" << endl

<< " 1 2 3 4 5 6 7 8 9 10"

<< " H = Your ship has been hit" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

<< " M = The enemy has missed your ships" << endl << endl;

// Loop for more accurate board displays

for(int x=0;x<10;x++)

{

cout << " " << letters[x] << " ";

for(int z=0;z<10;z++)

cout << "| " << player1\_board[x][z]<< " ";

cout << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

//Formatting

cout << endl;

}

// Function used to display the boards for Player 2 on their turn

void Board::disp\_board2()

{

// The char array for letters in the game board

char letters[10] = {'A','B','C','D','E','F','G','H','I','J'};

// Format the output and general layout of the game boards

cout << " -------- Enemy Board ----------- "

<< "X = Your ships" << endl

<< " 1 2 3 4 5 6 7 8 9 10"

<< " H = The enemy ship has been hit" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

<< " M = Your shot has not hit any ships" << endl << endl;

// Loop for more accurate board displays

for(int x=0;x<10;x++)

{

cout << " " << letters[x] << " ";

for(int z=0;z<10;z++)

cout << "| " << enemy2\_board[x][z]<< " ";

cout << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

//Formatting

cout << endl;

// Format the output and general layout of the game boards

cout << " -------- Player Board ----------- "

<< "X = Your ships" << endl

<< " 1 2 3 4 5 6 7 8 9 10"

<< " H = Your ship has been hit" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

<< " M = The enemy has missed your ships" << endl << endl;

// Loop for more accurate board displays

for(int x=0;x<10;x++)

{

cout << " " << letters[x] << " ";

for(int z=0;z<10;z++)

cout << "| " << player2\_board[x][z]<< " ";

cout << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

//Formatting

cout << endl;

}

//Function used to display Player 1's final information

void Board::results1()

{

//Output the player's name

cout << endl <<" --- Player #1" << " Information ---"

<< endl << endl << " Name: " << player\_name << endl;

//Output the coordinates of the player's ships

cout << endl << " Coordinates of ships:" << endl << endl;

for(int z=0;z<5;z++)

cout << " " << ships[z] << endl;

//Output the shots fired by each player

cout << endl << " Shots Fired: " << endl << endl;

for(int z=0;z<100;z++)

cout << " " << shts\_frd[z];

//Output the player's final game board

cout << endl << " Final Player Board:" << endl;

for(int z=0;z<10;z++)

{

for(int f=0;f<10;f++)

cout << " " << player1\_board[z][f];

cout << endl;

}

}

//Function used to display Player 2's final information

void Board::results2()

{

//Output the player's name

cout << endl <<" --- Player #2" << " Information ---"

<< endl << endl << " Name: " << player\_name << endl;

//Output the coordinates of the player's ships

cout << endl << " Coordinates of ships:" << endl << endl;

for(int z=0;z<5;z++)

cout << " " << ships[z] << endl;

//Output the shots fired by the player

cout << endl << " Shots Fired: " << endl << endl;

for(int z=0;z<100;z++)

cout << " " << shts\_frd[z];

//Output the player's final game board

cout << endl << " Final Player Board:" << endl;

for(int z=0;z<10;z++)

{

for(int f=0;f<10;f++)

cout << " " << player2\_board[z][f];

cout << endl;

}

}

/\*

\* File: Choice.h

\* Author: Daniel Baca

\*

\* Created on November 28, 2022, 8:19 PM

\*/

#ifndef CHOICE\_H

#define CHOICE\_H

//System Libraries

#include <iostream>

#include <iomanip>

#include <string>

#include <ctime>

#include <cstdlib>

#include <cmath>

#include <fstream>

#include <cstring>

using namespace std;

//Choice class, allowing players to ask for example input using a response

template <class T>

class Choice

{

private:

T response; //Variable used to make decision

public:

//Initialize Constructor

Choice()

{ response = ' '; }

//Choice function used to output requested output

void respond(char c)

{

response = c;

//Use choice input to determine what to output

if(response == 'Y' || response == 'y')

{

cout << endl << " Name for Player: Mary Sue" << endl << endl

<< " Ship Coordinates -" << endl << endl

<< " Carrier: A5 A9" << endl

<< " Battleship: F2 I2" << endl

<< " Cruiser: E5 G5" << endl

<< " Submarine: I8 I10" << endl

<< " Destroyer: C2 D2" << endl << endl

<< " --------------- Player Board ------------" << endl

<< " 1 2 3 4 5 6 7 8 9 10" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl

<< " A | 0 | 0 | 0 | 0 | X | X | X | X | X | 0 | " << endl

<< " B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " C | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " D | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " E | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " F | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " G | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " H | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " I | 0 | X | 0 | 0 | 0 | 0 | 0 | X | X | X | " << endl

<< " J | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< endl;

}

}

//Choice function used to output requested output

void respond(string c)

{

response = c;

//Use choice input to determine what to output

if(response == "Y" || response == "Y")

{

cout << endl << " Name for Player: Mary Sue" << endl << endl

<< " Ship Coordinates -" << endl << endl

<< " Carrier: A5 A9" << endl

<< " Battleship: F2 I2" << endl

<< " Cruiser: E5 G5" << endl

<< " Submarine: I8 I10" << endl

<< " Destroyer: C2 D2" << endl << endl

<< " --------------- Player Board ------------" << endl

<< " 1 2 3 4 5 6 7 8 9 10" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl

<< " A | 0 | 0 | 0 | 0 | X | X | X | X | X | 0 | " << endl

<< " B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " C | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " D | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " E | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " F | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " G | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " H | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " I | 0 | X | 0 | 0 | 0 | 0 | 0 | X | X | X | " << endl

<< " J | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< endl;

}

}

//Friendly overloaded << operator function

friend ostream& operator<<(ostream& os, Choice<char> &same);

};

#endif /\* CHOICE\_H \*/

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

#include "Choice.h"

//Overloaded << operator function

ostream& operator<<(ostream& os, Choice<char> & same)

{

os << " ----------- Battleship -----------" << endl

<< " The classic strategy board game!!!" << endl;

os << " ----- Somethings to know -----" << endl

<< " Carrier Size (5 spaces)" << endl

<< " Battleship Size (4 spaces)" << endl

<< " Cruiser Size (3 spaces)" << endl

<< " Submarine Size (3 spaces)" << endl

<< " Destroyer Size (2 spaces)" << endl << endl

<< " --- Keep this in mind for when you place your ships!!! ---"

<< endl << " --- Once the game starts and each player can shoot ---"

<< endl << " --- you can input Q to Quit the Game whenever. ---"

<< endl << endl;

return os;

}