Project 2

Title

Battleship

Board Game

Course

CIS-17A

Section

48290

Due Date

December 11, 2022

Author

Ting Yin Sha

TABLE OF CONTENTS

Introduction

“You sank my battleship!”

Battleship is the classic 2-Player naval combat board game that brings together competition, strategy, and excitement. Good for ages above 6 to play. In a head-to-head battle, you search for the enemy’s fleet of ships and destroy them one by one. No ship is safe in this game of stealth and suspense. Try to protect your own fleet while you annihilate your opponent. It is a battle that you must win! Feel the authentic thrill of the battle when you wage war on the high seas in the game of Battleship. Take charge and command your own fleet to defeat the enemy. With convenient portable battle cases and realistic naval crafts, Battleship puts you right in the middle of the action. It is a full-out assault. Position your ships strategically to survive the relentless strikes. Then target your opponent’s ships and wipe them out. You know you can do it!

How the Board Game Works

Object of the Games

Includes 2 portable battle cases, 10 plastic ships, 84 red hit pegs, 168 white miss pegs, label sheet.

Rules of the Game

1. On the first round of the game, you call out five shots (guesses) and mark each shot with a white peg in your target grid.
2. After you have called out all five shots (a salvo), your opponent announces which ones were hits and which ships they hit.
3. Change the white pegs on your target grid to red pegs for hits. Meanwhile, your opponent will place red pegs in the holes of any enemy ships that you hit.
4. Alternate back and forth in this manner until one of your ships is sunk. At that point, you lose one shot from your salvo. ...

5. Continue gameplay until one player sinks all the opposing ships and wins the game.

Development Summary

Translating Game Play Rules to Programming Language

The game needs to each player place 5 ships in different places, can't be overlapped, and each ship has a different length 5,4,3,3,2, how do correctly set up the ships? After guessing, how to check whether hit or not? If hit, how to record and minus from the ship length? How to calculate whether the ship is sunk? Also, the games need to take turns and save data for each player.

Development

Project 2 included pointers, memory allocation, and class which is the main concept. Class makes the project a very logical and almost complete implementation of the battleship game features. Another upgrade is setting every step automatically except the need to type the player's name. I developed 8 different versions and updated them little by little, the last version meets all the requirements. Most of the project is about class, constructor, destructor, operator overloading, inheritance, and advanced classes.

Version 1 has the basic game frame, board class implemented the game board set. When I run this version it proved to redefine two players' game boards with the same board function was not a good idea, because the second board will copy the first board's data. So I rewrote this part in other versions and set two game boards with two same functions.

Version 2 added a player.h file, this .h file is for storing the player's name and score.

Version 3 added a game.h and cpp file, this file is for implementing gaming.

Version 4 added an array of objects, which works not good.

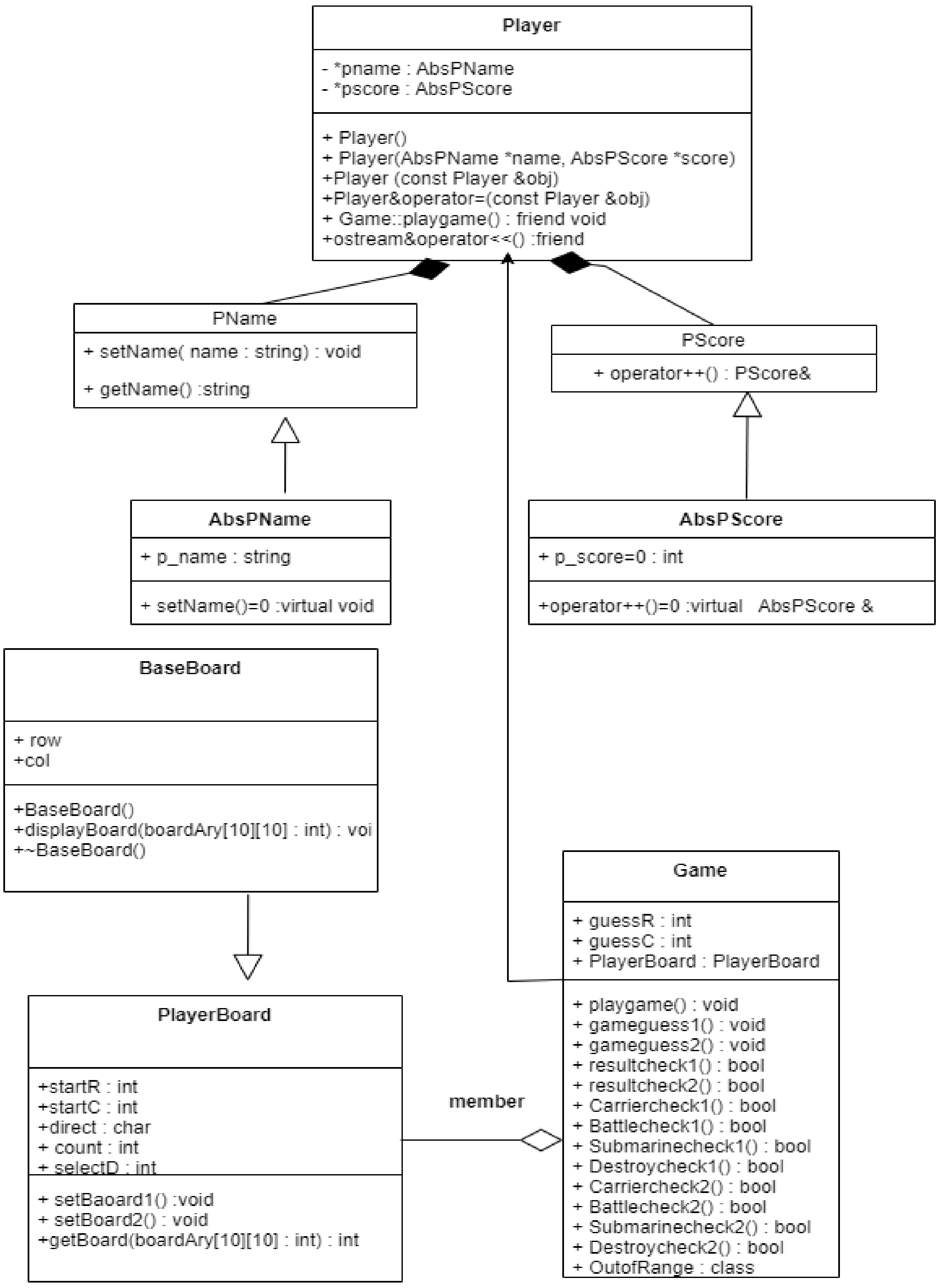
Version 5 added exceptions templates /copy constructor

Versions 6 & 7 fixed all the bugs, and updated and expand the project.

Here are the details about the final version, including four header files: board, game, player, and template. Also includes four cpp files: board, game, player, and main.

The player file has four classes, two virtual classes as the base classes, and two derived classes to set up the player's name and score. The player class is a polymorphism type, which allows calling the name and score class to gather all information from one player. Because there have pointers so I wrote a deep copy constructor to copy the data and a destructor to release the allocated memory. **The board file** includes one base class and one derived class. The base class initializes the base board array with 0; The derived class uses the base class's board array and adds two more functions: board set and get function. Project 1 only uses 4 ships of different sizes, project 2 updated to 5 ships and two ships of the same size. There are two directions to place the ships: R=Right, D=Down. The game board will be marked as 5,4,3,2 with the ship's size. A throw exception was used in this part to check whether the game board is correctly set. **The game file** does most of the battle game's job. Only one class with an aggregation board class. Includes guess, result check, and single ship check functions, and defined an exception class inside it. The check functions all use bool in case it repeats printing while running in a loop. There also include exceptions with different catch types: class, string, int. After an exception is caught will request re-input instead of exit. **The template h file** includes the class template and function template, and also works with vector to show the result. **The main file** mostly does the name set and score update job. The final game information will display in different ways in the main file.

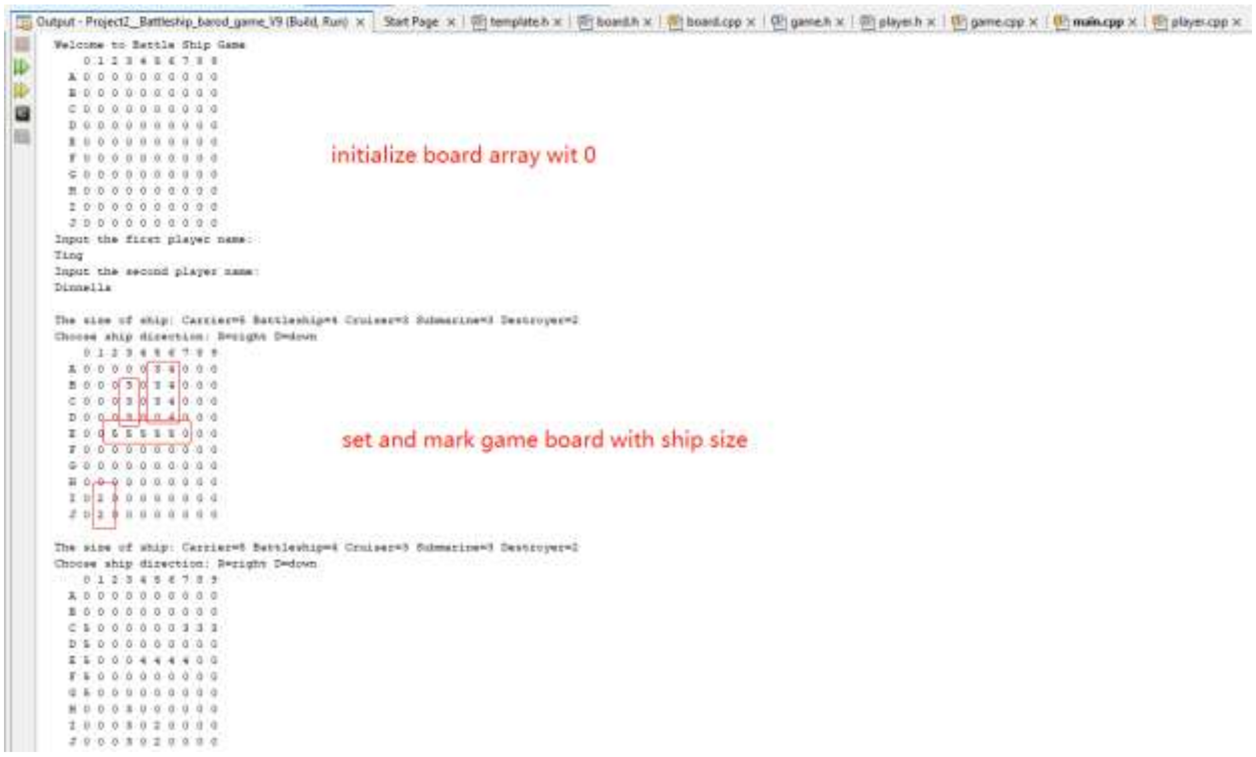
The logic of it All **UML**



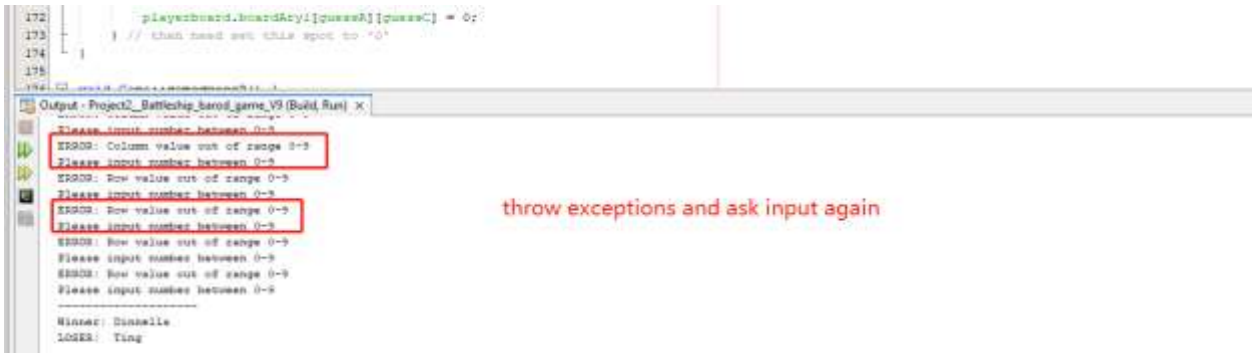
List of class

- c**AbsPName**
- c**AbsPScore**
- c**BaseBoard**
- ▼c**Game**
- c**OutofRange**
- c**Person**
- c**Player**
- c**PlayerBoard**
- c**PName**
- c**PScore**

Example of a Working Product


Output - Project2_Battleship_barod_game_V9 (Build, Run)
Start Page
template.h
board.h
board.cpp
game.h
player.h
game.cpp
main.cpp
player.cpp
initialize board array wit 0
set and mark game board with ship size

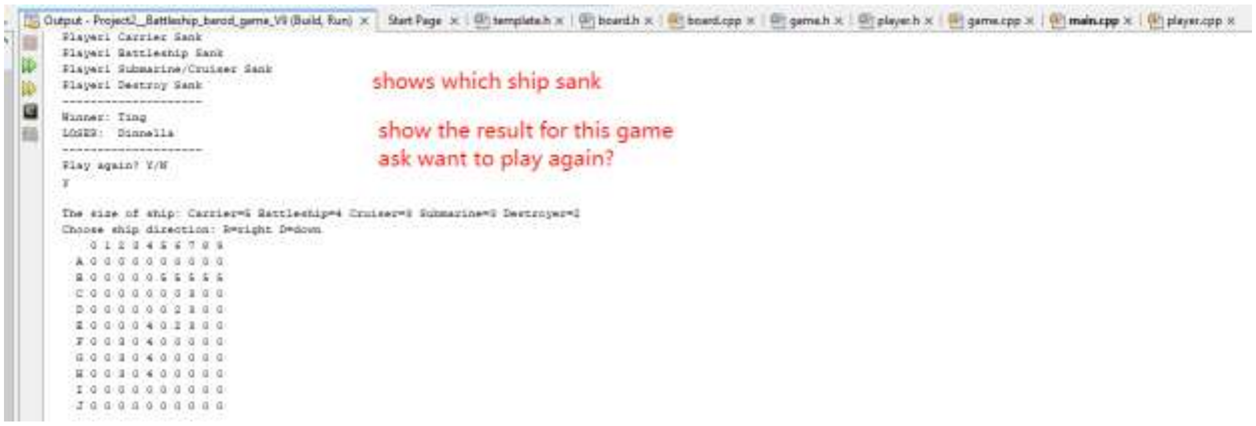
```
Welcome to Battle Ship Game
    0 1 2 3 4 5 6 7 8 9
  A 0 0 0 0 0 0 0 0 0 0
  B 0 0 0 0 0 0 0 0 0 0
  C 0 0 0 0 0 0 0 0 0 0
  D 0 0 0 0 0 0 0 0 0 0
  E 0 0 0 0 0 0 0 0 0 0
  F 0 0 0 0 0 0 0 0 0 0
  G 0 0 0 0 0 0 0 0 0 0
  H 0 0 0 0 0 0 0 0 0 0
  I 0 0 0 0 0 0 0 0 0 0
  J 0 0 0 0 0 0 0 0 0 0
Input the first player name:
Ting
Input the second player name:
Dinnella

The size of ship: Carrier=5 Battleship=4 Cruiser=3 Submarine=3 Destroyer=2
Choose ship direction: R=right D=down
    0 1 2 3 4 5 6 7 8 9
  A 0 0 0 0 0 3 4 0 0 0
  B 0 0 0 3 0 3 4 0 0 0
  C 0 0 0 3 0 3 4 0 0 0
  D 0 0 0 3 0 0 4 0 0 0
  E 0 0 5 5 5 5 5 0 0 0
  F 0 0 0 0 0 0 0 0 0 0
  G 0 0 0 0 0 0 0 0 0 0
  H 0 0 0 0 0 0 0 0 0 0
  I 0 2 0 0 0 0 0 0 0 0
  J 0 2 0 0 0 0 0 0 0 0

The size of ship: Carrier=5 Battleship=4 Cruiser=3 Submarine=3 Destroyer=2
Choose ship direction: R=right D=down
    0 1 2 3 4 5 6 7 8 9
  A 0 0 0 0 0 0 0 0 0 0
  B 0 0 0 0 0 0 0 0 0 0
  C 5 0 0 0 0 0 0 3 3 3
  D 5 0 0 0 0 0 0 0 0 0
  E 5 0 0 0 4 4 4 4 0 0
  F 5 0 0 0 0 0 0 0 0 0
  G 5 0 0 0 0 0 0 0 0 0
  H 0 0 0 3 0 0 0 0 0 0
  I 0 0 0 3 0 2 0 0 0 0
  J 0 0 0 3 0 2 0 0 0 0
```



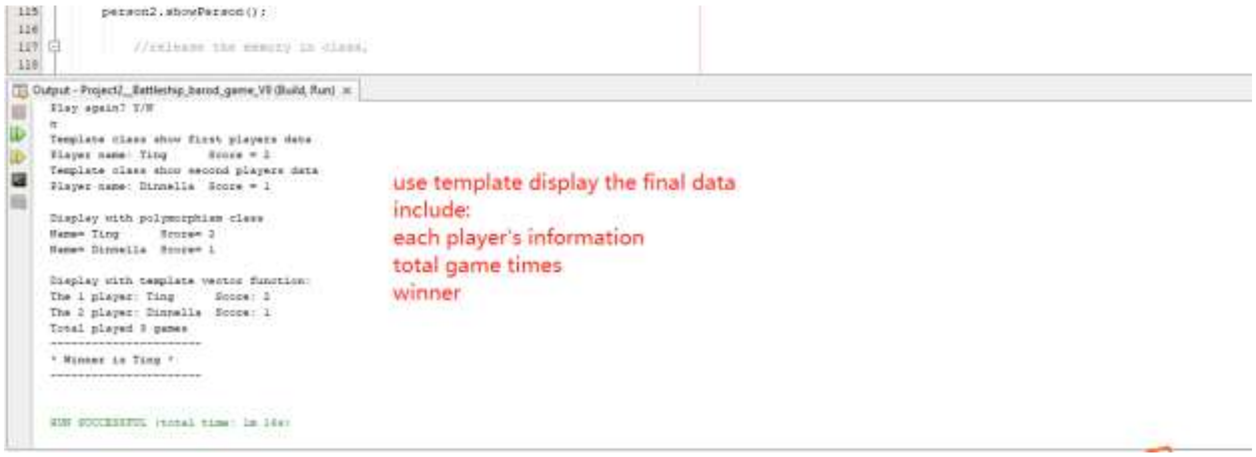
```
            playerboard.boardAry1[guessR][guessC] = 0;
        } // than need set this spot to "0"
    }
```

```
Please input number between 0-9
ERROR: Column value out of range 0-9
Please input number between 0-9
ERROR: Row value out of range 0-9
Please input number between 0-9
ERROR: Row value out of range 0-9
Please input number between 0-9
ERROR: Row value out of range 0-9
Please input number between 0-9
ERROR: Row value out of range 0-9
Please input number between 0-9
--------------------
Winner: Dinnelle
LOSER:  Ting
```

Output - Project2_Battleship_barod_game_V9 (Build, Run)
throw exceptions and ask input again



```
Player1 Carrier Sank
Player1 Battleship Sank
Player1 Submarine/Cruiser Sank
Player1 Destroy Sank
--------------------
Winner: Ting
LOSER:  Dinnella
--------------------
Play again? Y/N
Y

The size of ship: Carrier=5 Battleship=4 Cruiser=3 Submarine=3 Destroyer=2
Choose ship direction: R=right D=down
    0 1 2 3 4 5 6 7 8 9
  A 0 0 0 0 0 0 0 0 0 0
  B 0 0 0 0 0 5 5 5 5 5
  C 0 0 0 0 0 0 0 3 0 0
  D 0 0 0 0 0 0 2 3 0 0
  E 0 0 0 0 4 0 2 3 0 0
  F 0 0 3 0 4 0 0 0 0 0
  G 0 0 3 0 4 0 0 0 0 0
  H 0 0 3 0 4 0 0 0 0 0
  I 0 0 0 0 0 0 0 0 0 0
  J 0 0 0 0 0 0 0 0 0 0
```



```
        person2.showPerson();

            //release the memory in class,
```

```
Play again? Y/N
n
Template class show first players data
Player name: Ting       Score = 2
Template class show second players data
Player name: Dinnella  Score = 1

Display with polymorphism class
Name= Ting       Score= 2
Name= Dinnella  Score= 1

Display with template vector function:
The 1 player: Ting       Score: 2
The 2 player: Dinnella  Score: 1
Total played 3 games
----------------------
* Winner is Ting *
----------------------

RUN SUCCESSFUL (total time: 1m 16s)
```

Output - Project2_Battleship_barod_game_V9 (Build, Run)
use template display the final data
include:
each player's information
total game times
winner

Program

/*

 * File: main.cpp

```
 * Author: Ting Yin Sha
 * Created on November 27, 2022, 1:30 PM
 * Project2: Battle ship board game Version 9(use 5 ships and display which ship sank)
 */

//  Class with private member, .h .cpp files split
//constructor, destructors
//Arrays of Objects
//Static, friends, copy constructors
//overload 3operators, aggregation
//inheritance: protected members, base class to derived, polymorphic
//Abstract class, exception and templates, STL
#include <iostream>  //I/O Library
#include <string>    //String Library
#include <ctime>   //Random Number Generator, Setting Seed
#include <cstring>
#include <iomanip>  //Formatting Library
#include <cstdlib>   //Type Conversion, memory allocation, srand/Rand
#include <cctype>
#include <vector>   //vector STL
#include "board.h" //this class for setting up the ships
#include "player.h" //this class for setting players
#include "game.h"  //this class for playing game
#include "template.h"

using namespace std;
int main(int argc, char** argv) {
```

```
srand(static_cast<unsigned short>(time(0)));
char *name1; //ask for the players name
char *name2;
char again;  //ask player play again or not
vector<char*>playmap; //to store the name in vector
vector<int>playscore;

  cout<<"Welcome to Battle Ship Game"<<endl;
 //initialize the game board and display how it looks
 BaseBoard baseboard;
 baseboard.displayBoard(baseboard.boardAry1);



 //Add the first player, allocate memory for the PNmame class
 AbsPName *pname1 = new PName;
 //pass the name as an argument
 cout << "Input the first player name:" << endl;
 cin.getline(name1,20); //the name size is 20
 //push back the player name into a vector
 pushback(playmap,name1);
 //set name to class
 pname1->setName(name1);
 //score need to be set after the game started, and update after each game
 AbsPScore *pscore1 = new PScore;

 //Add the second player
```

```
AbsPName *pname2 = new PName;
//pass the name as an argument
cout << "Input the second player name:" << endl;
cin.getline(name2, 20); //the name size is 20
//push back the second player name
pushback(playmap, name2);
pname2->setName(name2);
//score need to be set after the game started, and update after each game
AbsPScore *pscore2 = new PScore;

//use a do while to continue playing
do{
   //define a Game class
   Game game;
   //call the playgame function from Game class to start play

   game.playgame();
   //result check and score++
   if (game.resultCheck1() && !game.resultCheck2()) {
      cout << "--------------------" << endl;
      cout << "Winner: " << pname2->p_name << endl;
      //increment score operator for the winner second player
      pscore2->operator++();
      cout << "LOSER:  " << pname1->p_name << endl;
      cout << "--------------------" << endl;
   }
   else {
```

```
            cout << "--------------------" << endl;
            cout << "Winner: " << pname1->p_name << endl;
            //increment score operator for the winner first player
            pscore1->operator++();
            cout << "LOSER:  " << pname2->p_name << endl;
            cout << "--------------------" << endl;
        }

        cout << "Play again? Y/N" << endl;
        cin>>again;

    } while (toupper(again) == 'Y');

    //push back the score
      pushback(playscore,pscore1->p_socre);
      pushback(playscore,pscore2->p_socre);
    //use template class print
    cout << "Template class show first players data " << endl;
    Person<string>person1(name1, pscore1->p_socre);
    person1.showPerson();
    cout << "Template class show second players data " << endl;
    Person<string>person2(name2, pscore2->p_socre);
    person2.showPerson();

        //release the memory in class,

    //polymorphism player, and use <<overload operator print out player's data
    cout<<"\nDisplay with polymorphism class"<<endl;
    Player* player1=new Player(pname1,pscore1);
    cout<<*player1<<endl;


    Player* player2=new Player(pname2,pscore2);
    cout<<*player2<<endl;


    cout<<"\nDisplay with template vector function:"<<endl;
    showVector(playmap, playscore);
    cout<<endl;
        return 0;
}



/*
 * File:   board.h
 * Author: Ting Yin Sha
 * Created on November 29, 2022, 10:56 AM
 * Project2: Battle ship board game board headerfile
 */
#include <iostream>  //I/O Library
#include <string>


using namespace std;
#ifndef BOARD_H
#define BOARD_H


class BaseBoard
{
public:
    int row=10,col=10;
    //use static member initialize board array to 0
     static int boardAry1[10][10];
     static int boardAry2[10][10];
   //default constructor
```

```
    BaseBoard( ){};
    void displayBoard(int boardAry[10][10]);//display initial board
    ~BaseBoard(); //destructor delete allocate memory
};

//player board inheritance from baseboard
class PlayerBoard : public BaseBoard
{

protected:
   // int guessR, guessC;
    int startR; //first spot row
    int startC; //first spot column
    char direct; //arrange direction
    int count;
    int selectD;
    //default constructor
public:
    PlayerBoard();
    //set all the ships spot
    //use two different set function to set each two player's board
    void setBoard1();
    void setBoard2();

    //get the player board array data after set or hit
    int getBoard(int boardAry[10][10]);
   // ~PlayerBoard();
};



#endif /* BOARD_H */


/*
 * File:   board.cpp
 * Author: Ting Yin Sha
 * Created on November 29, 2022, 10:56 AM
 * Project2: Battle ship board game board cpp
 */
#include <iostream>  //I/O Library
#include <string>    //String Library
#include <ctime>
#include "board.h"
```

```
using namespace std;

//initialize static boardAry in base class
int BaseBoard::boardAry1[10][10]={0};
int BaseBoard::boardAry2[10][10]={0};

 PlayerBoard::PlayerBoard() {
    startR = 0; //first spot row
    startC = 0; //first spot column
    direct; //arrange direction
    count = 0;
    selectD;
    //initialize array with for loop
   for(int i=0;i<10;i++){
           for(int j=0;j<10;j++){
              boardAry1[i][j]=0;
               boardAry2[i][j]=0;
           }
       }

    }
//first player set the ships
void PlayerBoard::setBoard1()
{

    cout<<"The size of ship: Carrier=5 Battleship=4 Cruiser=3 Submarine=3
Destroyer=2"<<endl;
    cout<<"Choose ship direction: R=right D=down"<<endl;
    //initialize an integer array of 5 ships size
    int sizeAry[5]={5,4,3,3,2};
   //use do while to arrange ships

    do{
        //initialize array to 0 every time when need reset
       for(int i=0;i<10;i++){
           for(int j=0;j<10;j++){
              boardAry1[i][j]=0;
           }
       }
         int size; // ship size
         count=0; // for count how many spot,if count=14 means all set

         //use a for loop control the ship size an set up board
         for(int i=0; i<5; i++) {
```

```
        // pass the ship size
        size = sizeAry[i];
        selectD = rand() % 2 + 1; //random get 1 or 2,represents Right or Down
        //R =right, D=down
        if (selectD == 1)
           direct = 'R';
        else
           direct = 'D';

        startR = rand() % (11 - size); //random get row start point
        startC = rand() % (11 - size); //random get column start point

  //Use switch case to check ship direction
  switch(toupper(direct)) {
           case 'R':for (int i = startC; i < (startC + size); i++) {
                 boardAry1[startR][i] = size;
              }
              break;

           case 'D': for (int i = startR; i < (startR + size); i++) {
                 boardAry1[i][startC] = size;
              }
              break;
           default:
              break;
        }

      }

     //check ship arrange successes, total should 14 one
     for (int i = 0; i < row; i++) {
        for (int j = 0; j < row; j++) {
           { //!=0 means set up, if count doesn't meet 17=5+4+3+3+2 need to be reset
              if (boardAry1[i][j] != 0)count++;

           }
        }
     }

  } while (count < 17);

  try {
     int c1 = 0;
     for (int i = 0; i < row; i++) {
        for (int j = 0; j < row; j++) {
                if (boardAry1[i][j] != 0)
                    c1++;
            }
        }

        if (c1 > 17) throw "Error, set ship failed";

    }   catch (string expstr) {
        exit(0);
    }

}

void PlayerBoard::setBoard2() {

    cout << "The size of ship: Carrier=5 Battleship=4 Cruiser=3 Submarine=3 Destroyer=2" <<
endl;
    cout << "Choose ship direction: R=right D=down" << endl;

    //initialize an integer array of 5 ships size
    int sizeAry[5] = {5, 4, 3, 3, 2};
    //use do while to arrange ships

    do {
        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                boardAry2[i][j] = 0;
            }
        }
        int size;
        count = 0; // for count how many spot,if count=14 means all set
        for (int i = 0; i < 5; i++) {
            size = sizeAry[i];
            selectD = rand() % 2 + 1; //random get 1 or 2,represents Right or Down
            if (selectD == 1)
                direct = 'R';
            else
                direct = 'D';

            startR = rand() % (11 - size); //random get row start point
            startC = rand() % (11 - size); //random get column start point

            //Use switch case to check ship direction
            switch (toupper(direct)) {
                case 'R':for (int i = startC; i < (startC + size); i++) {
                        boardAry2[startR][i] = size;
                    }
                    break;

                case 'D': for (int i = startR; i < (startR + size); i++) {
                        boardAry2[i][startC] = size;
                    }
                    break;
                default:
                    break;
            }
        }

        //check ship arrange successes, total should 14 one
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < row; j++) {
                { //!=0 means set up, if count doesn't meet 17 need to be reset
                    if (boardAry2[i][j] != 0)
                        count++;

                }
            }
        }

    } while (count < 17);

    try {
        int c2 = 0;
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < row; j++) {
                if (boardAry2[i][j] != 0)
                    c2++;
            }
        }

        if (c2 > 17) throw "Error, set ship failed";

    }
    catch (string expstr) {
        exit(0);
    }

}

int PlayerBoard::getBoard(int boardAry[10][10]) {
    return boardAry[10][10];
}

void BaseBoard::displayBoard(int boardAry[10][10]) {

    cout << "   ";
    for (int i = 0; i < row; i++) {
        cout << " " << i;
    }
    cout << endl;
    for (int i = 0; i < row; i++) {
        cout << "  " << char('A' + i);
        for (int j = 0; j < col; j++) {
            cout << " " << boardAry[i][j];
        }

        cout << endl;
    }
}

BaseBoard::~BaseBoard() {


}


/*
 * File:   player.h
 * Author: Ting Sha
 * Created on November 29, 2022, 1:46 PM
 *  Project2: Battle ship board game player headerfile
 */

#include <iostream>  //I/O Library
#include <string>    //String Library
#include <cstring>
using namespace std;
#ifndef PLAYER_H
#define PLAYER_H
#include "game.h"
#include "board.h"
//player file to get player's data
//Abstract class and polymorphic association

class AbsPName
```

```
{
public:

    string p_name; //player name
    //pure virtual member function
    virtual void setName(string) = 0;
};

class AbsPScore
{
public:
    int p_socre = 0; //player score
    virtual AbsPScore& operator++() = 0; //pure virtual function
};
//  polymorphism pass PName and PScore two virtual class to Player class
//so we can get the player information

class Player {
    //define <<operator as a friend to read private member
    friend ostream&operator<<(ostream &cout, Player player);
    friend void Game::playgame();

private:
    //polymorphism requires pointers or reference
    AbsPName *pname;
    AbsPScore *pscore;

public:
    //default constructor to initialize members
    Player();
    //constructor with argument to pass the value to the member
    Player(AbsPName *name, AbsPScore *score);
    //copy constructor, class has pointer need deep copy constructor
    Player(const Player &obj);
    //copy operator==
    Player&operator=(const Player &obj);///??????how

    //destructor to release memory
    ~Player();
};


//encapsulation into a real PName and PScore class

class PName : public AbsPName {
```

```
public:
    //implementation of pure vitual AbsPName function

    void setName(string name) {
        p_name = name;
    }

    string getName() {
        return p_name;
    }
};

//encapsulation into a real PScore class
class PScore:public AbsPScore
{
public:
    //overload ++operator, to increment the player win score
    PScore& operator++(){
        p_socre++;
        return (*this);
    }

};
#endif /* PLAYER_H */

/*
 * File:   player.cpp
 * Author: Ting Yin Sha
 * Created on November 29, 2022, 1:46 PM
 *  Project2: Battle ship board game player cpp
 */

#include <iostream>  //I/O Library
#include <string>    //String Library
#include <iomanip>
using namespace std;
#include "player.h"

Player::Player()
{
        pname->p_name="";
        pscore->p_socre=0;
}

Player::Player(AbsPName* name, AbsPScore* score)
```

```
{
    pname = name;
    pscore = score;
}

//copy constructor ??????????
Player::Player(const Player &obj){
     // Allocate memory for the name.
    pname=new PName;
     // Allocate memory for the score.
    pscore=new PScore;
   // Copy the elements of obj's class.
    pname=obj.pname;
    pscore=obj.pscore;
}

//copy operator==
Player& Player::operator=(const Player &obj){

    if(this==&obj) return *this;
    delete pname;
    delete pscore;
    //allocate memory to deep copy
    pname=new PName;
    pscore=new PScore;
    pname->p_name=obj.pname->p_name;
    pscore->p_socre=obj.pscore->p_socre;
}

//overload operator <<to cout player information
ostream&operator<<(ostream &cout, Player player)
{
    cout << "Name= " << setw(10) << setfill(' ')
            << left << player.pname->p_name
            << "Score= " << player.pscore->p_socre;
    return cout;
}

// destructor to release memory
Player::~Player()
{
    // cout<<"player constructor"<<endl;
        if(pname !=NULL){
            delete pname;
            pname=NULL;
        }
        if(pscore != NULL){
            delete pscore;
            pscore=NULL;
        }
}


/*
 * File:   game.h
  * Author: Ting Yin Sha
 * Created on November 29, 2022, 1:46 PM
 *  Project2: Battle ship board game player headerfile
 */
#include <iostream>  //I/O Library
#include <string>    //String Library
#include <ctime>
using namespace std;

#ifndef GAME_H
#define GAME_H
#include "board.h"

//this file shows game part
class Game
{
public:
    int guessR, guessC;
    //Aggregation, Game class contains PlayerBoard class
    PlayerBoard playerboard;
    //default constructor to initialize elements
    Game(){
         guessR=0; guessC=0;
    }
    //start game guess function
    void playgame( );
    //two players take turns to guess, call different guess functions
    void gameguess1();
    void gameguess2();
    bool resultCheck1( );
    bool resultCheck2( );
    //bool function to check which ship sank
     bool Carriercheck1();
    bool Battlecheck1();
    bool Submarinecheck1() ;
    bool Destroycheck1();

    bool Carriercheck2();
    bool Battlecheck2();
    bool Submarinecheck2() ;
    bool Destroycheck2();
    //Exception class
    class OutofRange{};// Empty class declaration
   // ~Game()
};


#endif /* GAME_H */

/*
 * File:   game.cpp
 * Author: Ting Yin Sha
 * Created on November 29, 2022, 10:56 AM
 * Project2: Battle ship board game cpp
 */

#include <iostream>  //I/O Library
#include <string>    //String Library
#include <ctime>
#include <vector>
#include "game.h"
#include "board.h"

using namespace std;

void Game::playgame(){

        cout << endl;
       // cout << pname1->p_name << " set ships" << endl;
        playerboard.setBoard1();
        playerboard.getBoard(playerboard.boardAry1);
        playerboard.displayBoard(playerboard.boardAry1);
        //   set the game board for the second player
        cout << endl;
       // cout << pname2->p_name << " set ships" << endl;
        playerboard.setBoard2();
        playerboard.getBoard(playerboard.boardAry2);
        playerboard.displayBoard(playerboard.boardAry2);
        //set each ship check bool to false
        bool isCarrier1=false;
        bool isBattle1=false;
        bool isSubmarine1=false;
        bool isDestroy1=false;

        bool isCarrier2=false;
        bool isBattle2=false;
        bool isSubmarine2=false;
        bool isDestroy2=false;

        //use while loop to check the result and keep guessing
   while(!resultCheck1() && !resultCheck2() ){

        //player1 guess second board and return the new game board after hit
      try{
         gameguess2( );
       playerboard.getBoard(playerboard.boardAry2);

       //if the guess number out of range throw exception class
       if(guessR<0 ||guessR>9 ||guessC<0 ||guessC>9)
       {
          throw OutofRange();
       }
      }
      //Exception class catch
      catch(Game::OutofRange)
      {
         cout<<"Error: A out of range value was entered"<<endl;
         //reinput value again after exception call
         cout<<"Please input number between 0-9"<<endl;
          gameguess2( );
       playerboard.getBoard(playerboard.boardAry2);
      }
       // catch string exception
      catch(const char* exceptionStr)
      {
         cout<<"ERROR: Row value out of range 0-9"<<endl;
         cout<<"Please input number between 0-9"<<endl;
         gameguess2();
       playerboard.getBoard(playerboard.boardAry2);
      }
      // catch int exception
      catch (int exceptionInt)
      {
         cout<<"ERROR: Column value out of range 0-9"<<endl;
         cout<<"Please input number between 0-9"<<endl;
         gameguess2();
       playerboard.getBoard(playerboard.boardAry2);
      }

       //player2 guess the first board and return the new game board after hit
      try{

       gameguess1();
       playerboard.getBoard(playerboard.boardAry1);


       if(guessR<0 ||guessR>9 )
       {
          throw "ERROR: Row value out of range 0-9";
       }
       if(guessC<0 ||guessC>9)
       {
          throw 0;
       }
      }
      // catch string exception
      catch(const char* exceptionStr)
      {
         cout<<"ERROR: Row value out of range 0-9"<<endl;
         cout<<"Please input number between 0-9"<<endl;
         gameguess1();
       playerboard.getBoard(playerboard.boardAry1);
      }
      // catch int exception
      catch (int exceptionInt)
      {
         cout<<"ERROR: Column value out of range 0-9"<<endl;
         cout<<"Please input number between 0-9"<<endl;
         gameguess1();
       playerboard.getBoard(playerboard.boardAry1);
      }
       //Exception class catch
      catch(Game::OutofRange)
      {
         cout<<"Error: A out of range value was entered"<<endl;
         cout<<"Please input number between 0-9"<<endl;
         //re-input value again after exception call
          gameguess1( );
       playerboard.getBoard(playerboard.boardAry1);
      }
```

```
//check player2 Carrier sank or not
          if(!isCarrier2 && Carriercheck2()){
                 cout<<"Player2 Carrier Sank"<<endl;
              isCarrier2=true;
          }
      if(!isBattle2 && Battlecheck2()){
          cout<<"Player2 Battleship Sank"<<endl;
          isBattle2=true;
      }

      if(!isSubmarine2 && Submarinecheck2()){
          cout<<"Player2 Submarine/Cruiser Sank"<<endl;
          isSubmarine2=true;
      }
      if(!isDestroy2 && Destroycheck2()){
          cout<<"Player2 Destroy Sank"<<endl;
          isDestroy2=true;
      }
      //check player1 Carrier sank or not
          if(!isCarrier1 && Carriercheck1()){
                 cout<<"Player1 Carrier Sank"<<endl;
              isCarrier1=true;
          }
      if(!isBattle1 && Battlecheck1()){
          cout<<"Player1 Battleship Sank"<<endl;
          isBattle1=true;
      }

      if(!isSubmarine1 && Submarinecheck1()){
          cout<<"Player1 Submarine/Cruiser Sank"<<endl;
          isSubmarine1=true;
      }
      if(!isDestroy1 && Destroycheck1()){
          cout<<"Player1 Destroy Sank"<<endl;
          isDestroy1=true;
      }

    }

}

void Game::gameguess1() {
    guessR = rand() % +10; //random row range 0-9
    guessC = rand() % +11; //random column range 0-9
    //if guess spot is not 0 means hit
    if (playerboard.boardAry1[guessR][guessC] !=0)
    {

        playerboard.boardAry1[guessR][guessC] = 0;
    } // then need set this spot to '0'
}

void Game::gameguess2() {
    //random row range 0-9, in order to call the exception, so extend the rang 0-10
    guessR = rand() % +10;

    guessC = rand() % +10; //random column range 0-9
    //if guess spot is not 0 means hit
    if (playerboard.boardAry2[guessR][guessC] !=0)
    {

        playerboard.boardAry2[guessR][guessC] = 0;
    } // then need set this spot to '0'
}


 bool Game::resultCheck1( ){
    //check the first player's ship
    bool result=true; //true means all sank

    for(int i=0; i<playerboard.row; i++){
        for (int j=0; j<playerboard.col; j++){

            if(playerboard.boardAry1[i][j]!=0)
                result=false;
            // return false if the array not all 0,
             //otherwise return true, means all the ships sank
        }

    }

    return result;
}
  bool Game::resultCheck2( ){
    //check the second player's ship
    bool result=true;
    for(int i=0; i<playerboard.row; i++){
        for (int j=0; j<playerboard.col; j++)
        {
            if(playerboard.boardAry2[i][j]!=0)
                result=false;
           // return false if the array not all 0,
            //otherwise return true, means all the ships sank

        }
    }

    return result;
}
  bool Game::Carriercheck2(){
      bool sank=false;

          for(int i=0; i<playerboard.row; i++){
        for (int j=0; j<playerboard.col; j++){
            if(playerboard.boardAry2[i][j]!=5 )

            sank=true;
        }
    }
     return sank;
  }

bool Game::Battlecheck2() {
    bool sank = false;

    for (int i = 0; i < playerboard.row; i++) {
        for (int j = 0; j < playerboard.col; j++) {
            if (playerboard.boardAry2[i][j] != 4)

                sank = true;
        }
    }
    return sank;
}

bool Game::Submarinecheck2() {
    bool sank = false;

    for (int i = 0; i < playerboard.row; i++) {
        for (int j = 0; j < playerboard.col; j++) {
            if (playerboard.boardAry2[i][j] != 3)

                sank = true;
        }
    }
    return sank;
}

bool Game::Destroycheck2() {
    bool sank = false;

    for (int i = 0; i < playerboard.row; i++) {
        for (int j = 0; j < playerboard.col; j++) {
            if (playerboard.boardAry2[i][j] != 2)

                sank = true;
        }
    }
    return sank;
}

  bool Game::Carriercheck1(){
      bool sank=false;

         for(int i=0; i<playerboard.row; i++){
        for (int j=0; j<playerboard.col; j++){
            if(playerboard.boardAry1[i][j]!=5 )

            sank=true;
        }
    }
     return sank;
  }

bool Game::Battlecheck1() {
    bool sank = false;

    for (int i = 0; i < playerboard.row; i++) {
        for (int j = 0; j < playerboard.col; j++) {
            if (playerboard.boardAry1[i][j] != 4)

                sank = true;
        }
    }
    return sank;
}

bool Game::Submarinecheck1() {
    bool sank = false;
    for (int i = 0; i < playerboard.row; i++) {
        for (int j = 0; j < playerboard.col; j++) {
            if (playerboard.boardAry1[i][j] != 3)

                sank = true;
        }
    }
    return sank;
}

bool Game::Destroycheck1() {
    bool sank = false;

    for (int i = 0; i < playerboard.row; i++) {
        for (int j = 0; j < playerboard.col; j++) {
            if (playerboard.boardAry1[i][j] != 2)

                sank = true;
        }
    }
    return sank;
}


/*
 * File:   template.h
 * Author: Ting Yin Sha
 * Created on December 6, 2022, 9:16 PM
 */

#include <iostream>  //I/O Library
#include <string>    //String Library
#include <vector>    //vector STL
#include <iomanip>

#ifndef TEMPLATE_H
#define TEMPLATE_H

//template class
template<class TN, class TS=int>
class Person{
    public:
        TN Name;
        TS Score;
        Person(TN name, TS score){
            this->Name=name;
            this->Score=score;

        }
        void showPerson(){
            cout<<"Player name: "<<setw(10) << setfill(' ') << left<<this->Name
                    << "Score = "<<this->Score<<endl;
        }
};

//template function for print
template <class T, class TT>
void showVector(vector<T>v1, vector<TT>v2){
    for(int i=0; i<v1.size(); i++){
        cout<<"The "<<i+1<<" player: "<<setw(10) << setfill(' ') << left<<v1[i]
                <<"Score: "<<v2[i]<<endl;
    }
    cout<<"Total played "<<v2[0]+v2[1]<<" games"<<endl;
    if (v2[0] > v2[1]) {
        cout<<"----------------------"<<endl;
        cout << "* Winner is " << v1[0] <<" *"<< endl;
        cout<<"----------------------"<<endl;
    }
    else if(v2[0] < v2[1] )
    { cout<<"----------------------"<<endl;
        cout << "* Winner is " << v1[1] <<" *"<< endl;
     cout<<"----------------------"<<endl;}
    else
    {cout<<"----------------------"<<endl;
        cout<<"* Tie *"<<endl;
        cout<<"----------------------"<<endl;
    }

}
//template function for push back an element
template<class T1>
void pushback(vector<T1>&v, T1 name){
    //  srand(static_cast<unsigned int> (time(0)));
    //pointer copyV dynamically allocated memory to a new vector, 1 size greater than old vector
    //  vector<int>*copyV= new vector<int>(v.size()+1); this will copy the old size and add new
size=5+5+1
    vector<T1>*copyV = new vector<T1>;
    //copy the old vector to new
    //method 1
    for (int i = 0; i < v.size(); i++) {
        copyV->push_back(v[i]);
    }

    //push back a name or score

    copyV->push_back(name);

    //copy new to oldj
    v=*copyV;

     // free memory
    delete copyV;
}

//template function for pop back an element
template<class T2>
void popback(vector<T2>v, T2 score){
      vector<T2>*popV = new vector<T2>;
      *popV=v;
      popV->pop_back();

      //clear all the element
      v.clear();

      //copy new to old
      v=*popV;
  // free memory
      delete popV;
}

#endif /* TEMPLATE_H */
```