DigiPen
INSTITUTE OF TECHNOLOGY

Programming Evaluation

Module 2 – High-level Programming II

Purpose

This programming evaluation will give you some practice with object-oriented programming (classes, objects, constructors, destructors, etc.), namespaces, references, dynamic memory allocation, and 2-dimensional arrays. It will also give you a chance to learn about interfaces and how to read source code.

Information

The task is to simply convert the previous WarBoats assignment (implemented as a procedural program) into an object oriented version. Because you can re-use almost all of your code from the previous assignment, this shouldn't take too long. The difficult parts have already been written.

0	0	0	0	0	0	0	0		−1	−1	−1	0	−1	−1	0	−1
0	0	0	0	0	**2**	0	0		−1	−1	−1	−1	−1	**102**	−1	0
0	0	0	0	0	**2**	0	0		−1	0	−1	−1	−1	**102**	−1	−1
0	**1**	**1**	**1**	**1**	**2**	0	0		−1	**101**	**101**	**101**	**101**	**102**	−1	−1
0	0	0	0	0	**2**	0	0		−1	−1	0	−1	−1	**102**	−1	−1
3	**3**	**3**	**3**	0	0	0	0		**103**	**103**	**103**	**103**	−1	−1	−1	−1
0	0	0	0	0	0	0	0		−1	−1	−1	−1	−1	−1	−1	−1
0	0	0	0	0	0	0	0		−1	−1	−1	−1	−1	−1	−1	−1

An 8x8 board with 3 boats placed. **The board after sinking all 3 boats.**

Some Details (from the previous assignment)

Instead of simply hard-coding the size of the board at 10x10, we are going to allow the player to specify the dimensions of the board. (The board is the ocean.) You should be able to handle boards of any rectangular size (square and non-square). You are also going to allow the player to specify the number of boats to place in the ocean. The only limit to the number of boats is the size of the ocean, since boats will not be allowed to overlap or leave the ocean. (Contrary to what Ferdinand Magellan's crew claimed, the world is flat, and if you go too far, you will fall off the end of it.)

All of your implementation will be placed in ***Ocean.cpp***, and this will be the only source file that you will submit. You are not allowed to include any other files in ***Ocean.cpp***. There are a couple of files included already, but you will not need any others.

New Stuff

The interface to the game is included in a header file named ***Ocean.h***. This file contains all of the information that the client (the player) needs. A partial ***Ocean.cpp*** file is provided which includes the implementation of a couple of new methods.

Since 95% of the internal code is going to be the same as the previous assignment, it shouldn't take a lot of time to implement. It is just a matter of going through the existing functions and modifying them to be part of the Ocean class, instead of simply being global functions.

Remember to use ***const*** wherever appropriate. This includes both function parameters as well as member functions.

Provided Files

File	Description
Warboat.h	This is the interface file. It contains all enums, structs and function prototype that you will use or implement. This file will be included in all files that are using the enums, structs or prototyped functions.
Ocean.h	This file contains the Ocean class definition. This file is included anywhere an Ocean instance is created and/or used.
Ocean.cpp	This is where you will be implementing all the required methods for the game to be played. This is the only file that you will be changing and submitting. This file is partially implemented, you just need to implement all the remaining member functions that are prototyped in ***Ocean.h***.
PRNG.h	The PRNG files are used to generate random numbers. These files will be used by the drivers. You don't need to change anything in them. They are already included and used in the driver files. You just need to add them as part of the project.
PRNG.cpp	
driver_sample.cpp	The ***driver_sample*** file tests all of your implemented functions (the ones you are implementing in ***Ocean.cpp***). You need to pass all tests found in here first. Once these tests are passed, you will use the ***driver_big*** file to run some more tests.
driver_big.cpp	This file contains more rigorous tests that you need to run/pass after you are done with the initial tests found in ***driver_smaple.cpp***.

Code Details

There are only 5 functions that you need to write for this assignment, and three of them are fairly simple.

Functions listed in the interface
`Ocean(int num_boats_, int x_quadrants_, int y_quadrants_);`
Constructor - The client calls this to create an ocean (game board). Client specifies the dimensions of the ocean and the number of boats that will be placed in it. All private fields are initialized. No return value.
`~Ocean(void);`
Destructor - This method is responsible for deallocating anything that was allocated in the constructor. No return value.
`BoatPlacement PlaceBoat(const Boat& boat_);`
The client calls this to place boats in the ocean. The client will create a boat and pass it to the method. The method must ensure that the boat will "fit" in the ocean. Do this by checking the location where it is to be placed, the orientation (horizontal/vertical), and whether or not it will overlap with another boat or stick outside of the ocean. The return value indicates whether or not the boat could be placed in the ocean.
`ShotResult TakeShot(const Point &coordinate_);`
Client calls this in an attempt to hit one of the boats. The coordinate parameter indicates where the client is attempting to strike. There are several possible results: Hit, Miss, Sunk, Duplicate, or Illegal. Hit, Miss, and Duplicate are obvious. Sunk is returned when a shot hits the last undamaged part of a boat. Sunk implies Hit. Illegal is any coordinate that is outside of the ocean (e.g. x/y less than 0 or outside the range).
`ShotStats& GetShotStats(void);`
Client calls this to get the current status of the game.

Testing your code

A "Grading Scripts" folder is provided in order for you to check if your implementation is correct. You will find two folders inside:

- **DriverSampleTestsScripts:** This is what you should start with. The driver_sample.cpp file is being used here. You have to pass all 3 simple tests found inside.
- **DriverBigTestsScripts:** After passing the above tests, you should run your code and make sure you pass the more rigorous tests found here. The driver_big.cpp file is being used here. You have to pass all 9 tests.

In order to run the tests follow these steps:

- grab your **Ocean.h** and **Ocean.cpp** files (which has all of your implementation) and place it in the "DriverSampleTestsScripts" or the "DriverBigTestsScripts" folder.
- Double click on the **"RunAll.cmd"** file.
    - All tests will run automatically. If a test fails, the script will stop at that test and you will get a message on the console that explains which test failed and why.

      NOTE: ***If you are working on a DigiPen lab/classroom machine you need to run the "DigiPen-RunAll.cmd" file instead of "RunAll.cmd"***

What to submit

You must submit two files ***(Ocean.h, Ocean.cpp)*** in a single .zip file (go to the class moodle page and you will find the submition link).

Do not submit any other files than the ones listed.