# CSCI 2312-001: Object Oriented Programming

# Assignment 3

# Assigned:

# Due: 11:59 PM

**Instructor: Marlu Oswald**

**Email: marlu.oswald@ucdenver.edu**

**Assignment 3**

(Remember ROCkET programming methodology!)

Create a class called WaterVehicle that has the following member variables to represent the ship as described in Assignment 1:

* length of ship (in number of grid spaces)
* starting grid location
* horizontal or vertical orientation on grid
* sunk (Boolean indicating whether the ship has been sunk or not)

You will also need to create a reasonable complement of member functions (constructors, accessor, and mutator functions) to create the ships, to set the location of the ship, get the location of the ship, determine if the ship was hit, and determine if the ship is sunk.

Then create a Carrier, a Battleship, a Cruiser, a Submarine, and a Destroyer which are all of type WaterVehicle and have the following lengths:

|  |  |
| --- | --- |
| Ship Type | Number of Grid Squares |
| Carrier | 5 |
| Battleship | 4 |
| Cruiser | 3 |
| Submarine | 3 |
| Destroyer | 2 |

Then write a program which will allow input of the ship length, grid location, and orientation. Assume the grid where the submarine is to be placed is a 10 by 10 grid and is indexed as described in Assignment 1. Randomly generate 15 torpedo shots in the main of the program, output the shot grid locations to the screen, and then output whether the torpedo hit any ship and if it did, whether that ship was sunk. You will need private member variables to keep track of where a ship has been hit to understand if it has been sunk.

Finally overload the << operator to print a WaterVehicle with tabs between fields:

* length of ship (in number of grid spaces)
* starting grid location
* horizontal or vertical orientation on grid
* sunk (Boolean indicating whether the ship has been sunk or not)

Demonstrate the use of this operator for each WaterVehicle after the torpedos have been fired.

**Grading Rubric**

Follow best practices in programming with declarations in .h files and implementations in .cpp files. Ensure your name is on every file in comments. Update your makefile (not makefile.txt!) AFTER you have tested successfully on the csegrid, update your readme.txt file then move over to your system. Create a folder named lastnameOB1 and place (just) your .h, .cpp, readme.txt, and makefile in that folder. Zip the folder and turn it into Canvas by the due date.

You may (and are encouraged to) discuss general strategy with classmates (or other help) but you MUST document all extra help other than the textbook (including web sites) with comments. Failure to document extra help may result in a zero for the assignment. While you should help your classmates, stop short of giving exact lines of code or pseudocode. If we can’t determine who shared and who copied, we may give you both a zero for the assignment.

|  |  |
| --- | --- |
| Correct and Complete Functionality | 14 |
| Compile and Fault Tolerance (on csegrid) | 2 |
| makefile and readme.txt | 2 |
| Documentation / Comments | 2 |