COS 161 – Algorithms in Programming

Project 04 – Battleship

Objectives

The objective of this assignment is to use recursion and the skills you have learned on previous projects to design a working, playable version of the game Battleship.

General Instructions for all Assignments

For each assignment you will write a Java program and test it. Start your programs with a comment block such as:

```
/*
   NAME: <your name>
   COS 161, <Semester> <Year>, Prof. <instructor name>
   Project ##
   File Name: CLASS_NAME.java
*/
```

Your programs should be neatly formatted, and follow the indenting, formatting, spacing, and commenting practices taught in class.

Read each assignment carefully to see what is required. If you do not understand something in the assignment, ask a tutor or the instructor. You will get partial credit if you finish only part of an assignment or it is not working correctly. Turn it in and explain what you completed and what issues it has. It is usually better to turn in an imperfect assignment on the due date, rather than falling behind in the course.

Part 1 (80 points) Battleship

The idea for this project is to allow you to design an entire program from the ground up. Make sure you spend some time thinking through the project before you get started. The first task is to understand the game Battleship. As with other projects, if you are not familiar with Battleship, you can find information at the following link: https://en.wikipedia.org/wiki/Battleship_(game)

The game must conform to the following:

1. Allow for at 2 players, and alternate turns guessing.

2. Must allow for players to place their own ships at the beginning
    - Needs to include the option to place vertically or horizontally
    - Must inform a player if they try to make an invalid placement

3. Must display the 10x10 game board using the DrawingPanel.
    - You must conform to the A-J and 1-10 labeling structure of the original game
    - Must track hits and misses for each player on one grid while displaying current ship status on another grid (meaning two 10 x 10 grids – like in battleship with the top and bottom displays).
    - Ships can be indicated just with a different color square (say gray when the water/background is blue) or you can try for the extra credit from the beginning if you so choose.
    - Hint: look at the Chess project for an example of how to setup the board!

4. Must use the traditional ships sizes
    - Carrier 5, Battleship 4, Destroyer 3, Submarine 3, Patrol Boat 2

5. Must use recursion to run the game.
    - The base case should be all a player's ships being destroyed
    - The recursive case should be a player taking their turn

It is up to you to decide how to structure the data and methods of this project. However, I do expect you to use good practices and to avoid writing overly inefficient code. As always, comments are required and should explain what your code does.

Extra Credit (10 Points) Improved Ships

Make the DrawingPanel display show actual ship shapes. Make sure you have them fit with the sizes and general shapes of the actual ship pieces.

Extra Credit (70 points) Technical Documentation

Create a design document (separate from the solution document). This

should explain every design decision you made in this project, including what classes you made, what data types you used, and why you made each decision. It should be readable by someone who is non-technical. This is a great opportunity to learn something about the process of producing documentation, especially when it is being presented to a client or decision maker who is not a programmer.

For full (extra) credit, your design document must include a description of every major algorithm you produced, specific instructions for using the program, and some form of UML diagram (there are plenty of free online tools to help with this). This document should be no shorter than 3 pages, using single spaced 12-point font and narrow margins (I like Courier New, but use any readable font you like). Feel free to include any graphics or diagrams, but these will be in addition to the 3-page text length requirement (about 1800 words if that is easier to keep track of).

This is not an essay writing class, but proper sentence structure and grammar is expected. If you do this properly, you should have no problem hitting and exceeding the length requirement on this. Really try to think about what information would be useful to both future developers working on the project and the information a client or supervisor would need about a project. You can also find plenty of examples of documentation or software manuals on the internet if you need inspiration on where to get started.

What to Turn In

Turn in a solution document as a single pdf file. This solution document should contain the following (make sure formatting is clear):

- Two (2) methods of your choice that you believe are most critical to your project.
- At least three screenshots of console input of game being played.
- At least four screenshots of the game board DrawingPanel
- Any additional extra credit materials (be sure to clearly mark it as extra credit!).

Once you have the solution document ready, submit it as an attachment on Brightspace. Be sure to also attach all the .java files you created or edited in this project. Do not zip or compress the files, submissions with archives of any type will be ignored.