Developer Candidate Home Exercise

Battleship State Tracker

Practicalities

Your take-home coding exercise is designed to take between two and five hours to complete. You are under no obligation to take any specific amount of time to complete the task.

The rules

The code you write should be your own and should be written without direct assistance. However, feel free to use as many reference resources (Stack Overflow, MSDN, Google, textbooks) as you like.

The task should be completed in C#.

Please submit your solution as a link to GitHub / BitBucket project. Ideally, it will contain project definitions, e.g. if using Visual Studio, provide a project that can easily be loaded and run. If any specific instructions are required, please include a readme.

Background

This exercise is based on the classic game "Battleship".

- Two players
- Each player has a 10x10 board
- During setup, players can place an arbitrary number of "battleships" on their board. The ships are 1-by-n sized, must fit entirely on the board, and must be aligned either vertically or horizontally.
- During play, players take a turn "attacking" a single position on the opponent's board, and the opponent must respond by either reporting a "hit" on one of their battleships (if one occupies that position) or a "miss"
- A battleship is sunk if it has been hit on all the squares it occupies
- A player wins if all of their opponent's battleships have been sunk.

The task

The task is to implement a Battleship state-tracker for a single player that must support the following logic:

- Create a board
- Add a battleship to the board
- Take an “attack” at a given position, and report back whether the attack resulted in a hit or a miss
- Return whether the player has lost the game yet (i.e. all battleships are sunk)

The application should not implement the entire game, just the state tracker. No UI or persistence layer is required.

What we are looking for

There is no one “correct” solution, and there is no trick. The main focus is not a fancy algorithm. Instead, we are looking for thoughtfully written, high quality software. As you write your solution, consider things such as:

- Readability
- Maintainability
- Testability
- Extensibility

Be prepared to justify your choices.

There are no bonus points for showing off. Good software development skills are far more important to us than extreme language prowess.

“Code review” interview

If your solution is accepted, we will invite you to come onsite for a follow-up interview. Part of that will include an interactive session with some developers from Flare. Over the course of the “code review” interview, be prepared to talk the team through your solution, explain your decisions and thought processes, and find and fix bugs. There may also be extension questions where you will be asked to modify your code in a certain way to extend its capabilities.

The panel will be looking for signs that you are suited to be part of a team software development environment. That means that effective communication is important; the aim of this interview is not to answer a series of questions but rather to engage in a dialogue.