This battleship strategy works by first generating searching shots by traversing the entire grid and checking whether the column number and row number are divisible by the length of the smallest boat or not. The results for the two are then xored to create a parity boolean. This is to ensure that while searching, the shots would always be a $n-1$ space blank on the board while searching when the smallest boat is n squares long. Once ships have been hit, another criteria comes into play, which demands that the free spaces extending vertically or horizontally around the interested space must be at least n squares long, or else it will be skipped, since the boat could not possibly be there.

Once a boat is found, the algorithm generates unoccupied spaces in the four cardinal directions adjacent to the found square and stores them in an arraylist. Once another square is found, the process is repeated. Should the boat on the uncovered square is different from the first boat discovered, the new boat will also get its arraylist of such probing squares. Firing consists of removing the last element from each boat's arraylist if the arraylist exists. Once a boat is sunk, the arraylist is cleared, and will not be used again, and the algorithm returns to its seach mode.