Objective
Create a single-player version of the game “Battleship” as in Assignment #1. This version of the program will be object-based. There will be objects for Ships and for the Board. The graphical display will interrogate these objects to do its work.

In addition to the File menu as in the last application, your application should have a Help menu with an option that brings up an “About” box. If you use the Application Wizard to create the application, this will be built for you automatically.

Activity
Create class modules Board and Ship. The signature of these classes will be as follows:

**Ship**
- **Properties**
  - Name (get, let)
  - LocationX (get, let) Location of the top left edge of the ship
  - LocationY (get, let)
  - Orientation (get, let)
  - State (get)
  - Size (get, let)
- **Functions**
  - Hit () Returns True if the hit was legal (ship was on the board and not already sunk.)

**Board**
- **Properties**
  - Size (get, let)
- **Functions**
  - Hit (X, Y) Mark location and any ship as hit. Return True if a ship is in the given location.
  - IsHit (X, Y) Return True if the player has hit this location.
  - PlaceShip (Ship, X, Y) Attempt to place the given ship on the board. Return True if the ship was successfully placed in the location.
  - RandomPlaceShip (Ship) Return True if the ship was successfully placed in a random location.
  - ShipAt (X, Y) Return a Ship if there is one at the given location, otherwise return Nothing.

The main form should instantiate a group of ships and place them on the board when the user selects “New Game”.

The main module for the program should contain the following Enums

```csharp
Public Enum btlOrientation
    btlLeftRight = 0
    btlUpDown = 1
End Enum
```
Public Enum btlShipState
    btlUnplaced = 0
    btlFloating = 1
    btlHit = 2
    btlSunk = 3
End Enum

These enumerated types will be used at the values of the “Orientation” and “State” properties of Ship. The board size should also be defined as a constant in the main module, and should be changeable.

Submission

Students should

- Turn in **at the start of class** a hardcopy of the code of the application with a cover page clearly indicating the number and name of the assignment and the student’s name and ID #.

- **Before class time**, submit a folder containing the complete VB project for the application to the on-line course Drop Box. (H:\Faculty\Burke\A1). The folder should be named with your last name, last four digits of your Student Id # and the assignment number. For example, “Smith 2233 A1”

Assessment

This assignment will be assessed on the completeness of the solution to the problem. Partial solutions will be given partial credit but only for those features of the application that operate. No credit will be given for non-functional code. Programs will also be graded according to adherence to object-oriented principles, namely modularity and separation of concerns. The Board and Ship classes should contain no graphics-related code.

Hints and Notes

1. Implement the board as two arrays, one of which stores the locations that the user has “hit.” This can be simply a Boolean type. The other array should store references to the Ship objects. A ship will be stored in multiple locations in the array, so that the ShipAt function can find it

2. Note the similarity between the process of placing a ship on the board and the process of checking to see whether a location is legal. You may want to consolidate these operations into a single function with a boolean argument switching between modes.

3. The X and Y arguments to the Board functions should be passed by value (ByVal) but Ship arguments must be passed by reference. (ByRef) Don’t forget that assignments involving object references must be done with Set. For exam

   ```
   Set A (5, 6) = AShip
   ```

   Not

   ```
   A (5, 6) = AShip  \(\ne\) error
   ```

4. Note that the State property is read-only. This should be implemented as a computation based on data associated with the ship, rather than a value stored in a member variable.