Battleship Group Project

Section 4: Classes and Objects

Credits and Copyright

Copyright notices

Copyright © 2016 by The Learning House.
All rights reserved. No part of these materials may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of The Learning House. For permission requests, write to The Learning House, addressed “Attention: Permissions Coordinator,” at the address below.

The Learning House
427 S. 4th Street #300
Louisville KY 40202

Battleship Group Project

Introduction

You have recently joined a games company that has been working on an implementation of the classic Battleship game. In Battleship, each player has their own hidden board which is a 10 x 10 grid where the rows are denoted 1-10 and the columns by the letters A-I.

A previous developer had started writing the logic for this game, and has created some classes for the business logic. Unfortunately, they won the lottery and promptly quit their job to pursue their dream of living on a beach in Costa Rica... leaving us without any documentation except some unit tests and a rough list of remaining tasks. Before walking out the door, the developer did assure us that the logic of placing ships on the board and determining responses to shots fired is working properly... however, this will need to be verified.

Task List

Done	Task
Y	Create class to represent coordinates on the board.
Y	Create class for ships of different lengths with a ShipType enum.
Y	Create class that given a ship type will return a properly sized ship.
Y	Create class to represent the board.
Y	Keep track of the shot history for the board.
Y	Each shot fired will either be invalid, duplicate, hit, miss, hit and sunk, or victory if all ships are sunk. Create an enum to represent this.
Y	Create logic to place a ship on the board. Players will choose a coordinate, ship type, and direction (up/down/left/right). Ships may not overlap or run off the bounds of the board grid.
N	Create a start menu for the game, prompt for each player's name.
N	Create an object that translates a number to its corresponding letter for the X coordinate (A=1, B=2, etc.).
N	Create a game workflow object that will contain two boards, keep track of which player's turn it is, and process each player's turn.
N	The game should set up the boards.
N	Each player should be prompted to place their ships on their board by giving a starting coordinate and a direction. Clear the screen when a player is finished so the other player can't cheat!

N A player's turn is as follows:

1. Show a grid with marks from the opponent board's shot history. Place a yellow M in a coordinate if a shot has been fired and missed at that location or a red H if a shot has been fired that has hit.

2. Prompt the user for a coordinate entry (ex: B10).

3. Validate the entry; if valid, create a coordinate object, convert the letter to a number, and call the opponent board's FireShot() method.

4. Retrieve the response from the FireShot method and display an appropriate message to the user. Possible results are:

   a. Invalid coordinate - repeat turn

   b. Duplicate shot - repeat turn

   c. Hit - "You hit something!" - next player's turn

   d. Hit and sunk - "You sank your opponent's {ship name}" - next player's turn

   e. Miss - "Your projectile splashes into the ocean, you missed!" - next player's turn

   f. Victory - "You have sunk all your opponent's ships, you win!" - end game

5. Remember to prompt to continue and clear the screen to keep things clean.

N When the game ends, prompt the players if they would like to play again. If so, go back to setting up the boards. If not, quit the program.