**Class Side (superclass)**

*VARIABLES*

*board* is an int[8][8] which contains your own pieces.

0 represents nothing

1 represents part of a 2-long ship

2 represents part of a 3-long ship

3 represents part of a 4-long ship

4 represents part of a 5-long ship

*oBoard* is an int[8][8] which contains your knowledge of the opponent’s board.

0 represents unknown

1 represents a miss

2 represents a hit but not a sink

11-14 represent sinks

*boats* is an int[4] which contains the status of your boats

Index 0 holds 2 which corresponds to the two “lives” of your 2-long boat

Index 1 holds 3 which corresponds to the three “lives” of your 3-long boat

Index 2 holds 4 which corresponds to the four “lives” of your 4-long boat

Index 3 holds 5 which corresponds to the four “lives” of your 5-long boat

When a ship is hit, its number decrements– you know a ship has sunk when the number at its index reaches 0

*oBoats* is a boolean[4] which keeps track of which boats you’ve sunk (default for all is false)

Index 0 corresponds to whether you sunk the 2-long boat

Index 1 corresponds to whether you sunk the 3-long boat

Index 2 corresponds to whether you sunk the 4-long boat

Index 3 corresponds to whether you sunk the 5-long boat

*METHODS*

*hitOrMiss(x:int,y:int):int*

Determines if one of your ships is at x,y

Returns 0 if it is a miss

Returns 1 if it is a hit, but not a sink

Returns the value of the number on your board at x,y plus 10 (arbitrary number) if it is a

sink

How to use the value you get back:

If you get a 0, then you know it’s a miss

If you get a 1, then you know it’s a hit

If you get a number larger than 1, then subtract 10 and that corresponds to the ship

number that you sank (if you get 11, subtract 10 and you know you sunk the ship that has

boat number 1, which is the two-long ship)

**Class Player (extends Side)**

*Default constructor (int[][] brd)*

Sets *board* to *brd*

Sets *oBoard* to a new int[8][8]

**Class AI (extends Side)**

*VARIABLES*

*difficulty* is an int

0 = easy (fires randomly and tracks down ships)

1 = medium (fires diagonally and tracks down ships)

2 = hard (fires diagonally, tracks down ships, and calculates where ships can fit)

*pX* and *pY* are ints that correspond to where the AI previously fired

Used for tracking down ships

*fire* is an int[]

Used to know which orientation the ship it’s attacking might be facing

For example, {1,0} means 1 right, {0,-1} means 1 down, etc.

*found* is a boolean

If true, then the AI knows for sure which way the ship its attacking is facing

*target* is a boolean

If true, a ship has been found and it enters target mode

If false, then fire at random

*p* is the Player that the AI is playing against (instantiated with constructor)

*misses* is an ArrayList<int[]> that keeps track of unintentional hits it should come back to

*METHODS*

*Default constructor(int[][] brd, Player pl)*

Set *board* to *brd*

Set *oBoard* to a new int[8][8]

Set *difficulty* to 0

Set *p* to *pl*

*setDifficulty(d:int):void*

Set difficulty

*fireAtRandom(strategy:boolean):void*

Fires at random, if strategy is true, then it only fires at diagonals

*getOBoard():int[][]* //Just for debugging

Returns *oBoard*

*fire():void*

Fires based on difficulty

*followFire():void*

Fires, trying to finish off a ship

*update(x:int,y:int):int*

Determines the outcome of a shot

Returns 0 if miss

Returns 1 if hit

Returns 2 if sink

*printOBoats():void* //Just for debugging

Prints which boats the AI thinks its sunk