Trek Wars

Purpose: classes, inheritance, class diagrams, virtual, testing

Description

Universal Merriment Development (UMD) is creating an online space battle game called (Trek Wars). You are tasked with creating some classes for this game. Again you have to just write the class and test it. A description of the classes is given below:

Figure 1: Graphics from the game.

enum Alignment{us, them, chaotic};

Ship (the base class) attributes

name : string
align : Alignment
xLoc : integer
yLoc : integer
range: int
currentHealth : integer
attackPower: integer
maxHealth: integer

methods

public Ship(name : string, x : integer , y : integer , align : Alignment , maxHealth : integer,
range :integer, attackPower : integer); // currHealth is set to maximum
public virtual attack(target: * Ship) : void //
private virtual getType() : string //''Battleship'', ''Cruiser'', ''Corvette'', ''Repairship''
public getX():int // returns the x coordinate
public getY():int // returns the y coordinate
public getAlign():Alignment // returns the alignment
public virtual status () : string // see format below
public virtual move() : void // changes position by the amount of that type of ship,
// increases health by 1 (until max reached)
public changeAlign() :void // changes the alignment.
public assessDamage(amt : int) : void // changes the health by amt,
// (keeping it within bounds [0,maxHealth])

Battle(derived from the Ship class) range is 10, maxHealth =100 ,attack = 10
attributes
torpedoes // int initially 10
methods
// always moves along the vector (-1, -1)
move() attack(target: Ship *) : void// attacks and fires torpedo >0 and
// does additional 10 damage, 1 less torpedo
status () // also indicates number of torpedoes

Cruiser (derived from Ship class) range is 50, maxHealth =50, attack = 5
method
// always moves along the vector (1, 2)
move()
attack(target: Ship *) : void //attacks 5

Corvette (derived from Ship class) range is 25, maxHealth =20
method
// always moves along the vector (5, 5)
move()
attack(target: Ship *) : void
// Everyone loves corvettes so their attack
// flips ships in range to its state
// (if self is us, turns them to us,
// if self is them, turns us to them)

Repair (derived from Cruiser class) range is 25, maxHealth = 20

method attack(target: Ship *) : void //its attack repairs a ship of own kind to max health

Input

No input

Output

For status() print each
name : <name>
type : <Class name>
health:<healthStr>
location: (xLoc, yLoc)
torpedoes: // only if this is a battleship

Except for repair ships, attacks only work if the target of the attack is of the opposite alignment and in range. . By opposite alignment, we mean (us attacks them and them attacks us). Chaotic ships attack everyone. Repair ships only attack (in this case repair), ships of the same alignment.