Battleship

Natalie Chin

December 19, 2017

Constant Module

Module

Constant

Uses

N/A

Syntax

Exported Constants

width = 8
height = 8

Exported Access Programs

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

None

Coordinate Module

Template Module

Coordinate

Uses

None

Syntax

None

Exported Types

Coordinate = (x,y)

Exported Access Programs

Routine name	In	Out	Exceptions
new Coordinate	real, real	Coordinate	
getX		real	
getY		real	

Semantics

State Variables

x: real
y: real

State Invariant

None

Assumptions

None

Access Routine Semantics

new Coordinate (xc, yc):

- transition: $x, y := xc, yc$
- output: $out := self$
- exception: none

getX():

- transition: none
- output: $out := x$
- exception: none

getY():

- transition: none
- output: $out := y$
- exception: none

Local Functions

None

Fleet Module

Template Module

Fleet

Uses

Coordinate, Ships

Syntax

Exported Types

allShips : sequence of Ships

Exported Access Programs

Routine name	In	Out	Exceptions
new Fleet		Fleet	
getFleet		sequence of Ships	
addToFleet	integer, Coordinate, direction		DuplicateIDException, FullException

Semantics

State Variables

allShips: sequence of Ships

State Invariant

MAX_SIZE = 5

Assumptions

None

Access Routine Semantics

new Fleet ():

- transition: $allShips = []$
- output: $out := self$
- exception: none

getFleet():

- transition: none
- output: $out := self$
- exception: none

addToFleet(id, length, coord, direction):

- output: out :=!!!
- exception: $exc := (iden(uid)) \Rightarrow$ DuplicateIDException $\vee\ allShips.size() == 5 \Rightarrow FullException$

Local Functions

iden: real $\rightarrow$ boolean
iden(id) $\equiv \exists(s : Ships|(s.getID() == id) : (self.getID() == id))$

overlap: Coordinate $\rightarrow$ boolean
overlap(id) $\equiv \exists(s : Ships|(s.getLoc() == Coordinate) : (s.getLoc() == Coordinate))$

vertical: real $\times$ real $\times$ real $\rightarrow$ boolean
vertical(id) $\equiv \exists(s : Ships|(s.getLoc() == Coordinate) : (s.getLoc() == Coordinate))$

horizontal: real $\times$ real $\times$ real $\rightarrow$ boolean
horizontal (id) $\equiv \exists(s : Ships|(s.getLoc() == Coordinate) : (s.getLoc() == Coordinate))$

withinBound: real $\times$ Coordinate $\rightarrow$ boolean
withinBound(c) $\equiv \exists c.getY()+length > Constants.height \vee c.getX()+length > Constants.width$

zeroCheck: Coordinate $\rightarrow$ boolean
zeroCheck(c) $\equiv c.getX() < 0 || c.getY() < 0$

Game Module

Template Module

Game

Uses

Gamefield

Syntax

Exported Constants

None

Exported Access Programs

Routine name	In	Out	Exceptions
new Game	real, real	Gamefield	

Semantics

State Variables

*player*1: Gamefield
*player*2: Gamefield

State Invariant

None

Assumptions

None

Gamefield Module

Module

Gamefield

Uses

Square, Ships, Coordinate, Fleet

Syntax

Exported Constants

None

Exported Access Programs

Routine name	In	Out	Exceptions
new GameField	real, real	Gamefield	
hitOrMiss	Coordinate	boolean	
printFleet			
addFleet	Fleet		

Semantics

State Variables

board: sequence of sequence of Square
Fleet: Fleet

State Invariant

None

Assumptions

None

Access Routine Semantics

GameField(width, height):

- transition: $board$:=<<>> , $fleet$:=<>
- exception: none

hitOrMiss(c):

- transition $\exists(c$: Coordinate|board[c.getX()][c.getY()].hasShip() : board[c.getX()][c.getY().hasShip()])
- exception: none

addFleet(f):

- transition: $s := \forall(s : Ships | s \in f.getFleet() : addShip(s.getID(), s.getLoc())$
- exception: none

Local Functions

addShip: real $\times$ sequence of Coordinate $\rightarrow$
addShip(uid, c) $\equiv \forall$(p: Coordinate|($p \in$ c) : $board[p.getX()][p.getY()].addShip(uid)[Ships(uid, loc)]$)

Ships Module

Module

Ships

Uses

Coordinate

Syntax

Exported Constants

None

Exported Access Programs

Routine name	In	Out	Exceptions
new Ship	real, sequence of Coordinate	Ships	
getID		real	
getSunk		boolean	
getLoc		sequence of Coordinates	
doesOccupy	sequence of Coordinates	boolean	
setSunk	Coordinate		

Semantics

State Variables

uID: real
$sunk$: boolean
loc: sequence of Coordinates
$sunk$: sequence of boolean

State Invariant

None

Assumptions

None

Access Routine Semantics

Ships(uID, location):

- transition: $self.uID$ = uID, $self.sunk$ = False, $self.loc$ = location, $self.sunk$ = <>
- exception: none

getID(c):

- transition: none
- output: $self.uID$
- exception: none

getSunk(c):

- transition: none
- output: $self.sunk$
- exception: none

getLoc(c):

- transition: none
- output: $self.loc$
- exception: none

doesOccupy(loc):

- transition: none
- output: helper(location)¿=0
- exception: none

setSunk($coord$):

- transition: $sunk[index] = True$
- output: none
- exception: none

Local Functions

helper: sequence of Coordinates $\rightarrow$ real
helper($location$) $\equiv \exists(c : Coordinate | c \in location : True)$

initSunk: self $\rightarrow$ self
initSunk(c) $\equiv \exists(i : \mathbb{N} | 0 \leq i \leq size(loc) : [False])$

isSunk: self $\rightarrow$ boolean
isSunk() $\equiv (i : \mathbb{N} | 0 \leq i \leq size(loc) sunk[i] = False : False)$

Square Module

Module

Square

Uses

Coordinate

Syntax

Exported Constants

Exported Access Programs

Routine name	In	Out	Exceptions
new Square		Square	
getGuessed		boolean	
setGuessed	boolean		
hasShip		boolean	
shipHit			
setID	real		
getID		real	

Semantics

State Variables

guessed: boolean
hit: boolean
shipId: real
location: sequence of Coordinates

State Invariant

None

Assumptions

None

Access Routine Semantics

Square (uID, location):

- transition: $self.guessed$, $self.hit$ = False, $self.shipID$ = None, $self.location$ = None
- exception: none

getGuessed():

- transition: none
- output: $self.guessed$
- exception: none

setGuessed(c):

- transition: $self.guessed$ = c
- output: none
- exception: none

hasShip():

- transition: none
- output: ! $self.shipID \leq 0$
- exception: none

addShip(id):

- transition: $self.shipID$ = id
- output: none
- exception: none

shipHit():

- transition: $self.hit$ = True
- output: none

- exception: none

getID():

- transition: none
- output: $self.shipID$
- exception: none

Local Functions

None