Point ADT Module

Template Module

PointT

Uses

none

Syntax

Exported Types

PointT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
PointT	real, real	PointT	PtOutBonds

Semantics

State Variables

x: real
y: real

State Invariant

none

Assumptions

The constructor PointT is called for each abstract object before any other access routine is called for that object. The constructor cannot be called on an existing object.

Access Routine Semantics

PointT(x, y):

- transition: $x, y := x, y$
- output: $out := self$
- exception $exc := ((\neg(0 < x \leq 10) \vee \neg(0 < y \leq 10)) \Rightarrow \text{PtOutBonds})$

Ship ADT Module

Template Module

ShipT

Uses

PointT

Syntax

Exported Types

ShipT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
ShipT	PointT, real, real, string	ShipT	OutOfBond

Semantics

State Variables

pts: PointT[ ]

State Invariant

None

Assumptions

The constructor ShipT is called for each abstract object before any other access routine is called for that object. The constructor cannot be called on an existing object.

Access Routine Semantics

RegionT$(p, l, turn)$:

- transition: $turn == "left" \Rightarrow (\forall(i : \mathbb{N} | 0 \leq i < l : pts[i].x = p.x, pts[i].y = p.y - i)$
  $|turn == "right" \Rightarrow (\forall(i : \mathbb{N} | 0 \leq i < l : pts[i].x = p.x, pts[i].y = p.y + i)$
  $|turn == "up" \Rightarrow (\forall(i : \mathbb{N} | 0 \leq i < l : pts[i].x = p.x - i, pts[i].y = p.y)$
  $|turn == "down" \Rightarrow (\forall(i : \mathbb{N} | 0 \leq i < l : pts[i].x = p.x + i, pts[i].y = p.y)$
- output: $out := self$
- exception: $exc := \exists(i : \mathbb{N} | 0 \leq i < 5 : \neg 0 < pts[i].x \leq 10 \vee \neg 0 < pts[i].y \leq 10) \Rightarrow$ OutOfBond

Board ADT Module

Template Module

BoardT

Uses

PointT, ShipT

Syntax

Exported Types

BoardT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
BoardT		BoardT	
state	PointT	real	
occupied	ShipT		
conflict		boolean	
sunk	ShipT	boolean	

Semantics

State Variables

board: real[ ][ ]

State Invariant

None

Assumptions

The constructor ShipT is called for each abstract object before any other access routine is called for that object. The constructor cannot be called on an existing object.

Access Routine Semantics

BoardT( ):

- transition: $\forall(i, j : \mathbb{N} | 0 \leq i < 10 \wedge 0 \leq j < 10 : board[i][j] = 0)$
- output: $out := self$

state(pt):

- output: $out := board[pt.x - 1][pt.y - 1]$

occupied($ship$):

- transition: $\forall(i : \mathbb{N} | 0 \leq i < ship.pts.length() : board[ship.pts[i].x - 1][ship.pts[i].y - 1] = board[ship.pts[i].x - 1][ship.pts[i].y - 1] + 6)$

conflict( ):

- output: $out := \exists(i, j : \mathbb{N} | 0 \leq i < 10 \wedge 0 \leq i < 10 : board[i][j] > 6)$

sunk($ship$):

- output: $out := \forall(i : \mathbb{N} | 0 \leq i < ship.pts.length() : board[ship.pts[i].x - 1][ship.pts[i].y - 1] == 2)$

Ship Placement ADT Module

Template Module

shipplaceT

Uses

PointT, ShipT, BoardT

Syntax

Exported Types

ShipPlaceT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
ShipPlaceT	ShipT, ShipT, ShipT, ShipT, ShipT	ShipPlaceT	ShipConflict
valid	PointT	boolean	

Semantics

State Variables

ships: ShipT[ ]
temp: BoardT

State Invariant

None

Assumptions

The constructor ShipT is called for each abstract object before any other access routine is called for that object. The constructor cannot be called on an existing object.

Access Routine Semantics

ShipPlaceT($s1, s2, s3, s4, s5$):

- transition: $ships = [s1, s2, s3, s4, s5]$;
  $\forall(i : \mathbb{N} | 0 \leq i < 5 : temp.occupied(ships[i]))$
- output: $out := self$
- exception: $exc := temp.conflict() \Rightarrow ShipConflict$

valid(pt):

- output: $out := \exists(i : \mathbb{N} | 0 \leq i < 5 : \exists(j : \mathbb{N} | 0 \leq j < ships[i].pts.length() : ships[i].pts[j] == pt))$

Player ADT Module

Template Module

PlayerT

Uses

PointT, ShipT, BoardT, ShipPlaceT

Syntax

Exported Types

PlayerT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
PlayerT	string, ShipPlaceT	PlayerT	
isover		boolean	
gotshot	PointT		
isSunk		boolean	

Semantics

State Variables

name: String
ships: ShipPlaceT
board: BoardT
ship_sunk: real

State Invariant

None

Assumptions

The constructor ShipT is called for each abstract object before any other access routine is called for that object. The constructor cannot be called on an existing object.

Access Routine Semantics

PlayerT($name, ships$):

- transition: $name, ships, ship\_sunk = name, ships, 0;$
  $\forall(i : \mathbb{N} | 0 \leq i < 5 : board.occupied(ships.ships[i]))$
- output: $out := self$

isover( ):

- output: $out := ship\_sunk == 5$

gotshot(pt):

- transition: $(ships.valid(pt) \Rightarrow (isSunk() \Rightarrow sunk\_ship = sunk\_ship + 1) \wedge board.board[pt.x - 1][pt.y - 1] = 2)$
  $|(\neg ships.valid(pt) \Rightarrow board.board[pt.x - 1][pt.y - 1] = 1)$

isSunk( ):

- output: $out := +(i : \mathbb{N} | 0 \leq i < 5 \wedge board.sunk(ships.ships[i]) : 1)$
  $\geq sunk\_ship)$

Client Module

Template Module

Client

Uses

PointT, ShipT, BoardT, ShipPlaceT, PlayerT

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
init	PlayerT, PlayerT		
play	PointT		

Semantics

State Variables

p1: PlayerT
p2: PlayerT
round: real

State Invariant

None

Assumptions

init() is called before any other access program.

Access Routine Semantics

init(p1,p2):

- transition: $p1, p2, round := p1, p2, 0$

- output: $out := self$

play(pt):

- transition: $notover(p1, p2) \Rightarrow (turn \mod 2 == 0 \Rightarrow p2.gotshot(pt) \wedge round = round + 1$
  $|turn \mod 2 == 1 \Rightarrow p1.gotshot(pt) \wedge round = round + 1)$

Local Functions

notover: PlayerT$\times$ PlayerT $\rightarrow$ boolean
isSunk $(p1, p2) \equiv \neg p1.isover() \vee \neg p2.isover()$