Assignment 4
Specification

SFWR ENG 2AA4

Jay Mody - 400195508 - modyj

April 2, 2020

Dot Type Module

Module

DotT

Uses

n/a

Syntax

Exported Constants

None

Exported Types

Dottypes = {R, G, B, Y} *// R for red, G for green, B for blue, Y for yellow*

Exported Access Programs

Routine name	In	Out	Exceptions
new DotT	Dottypes	DotT	

Semantics

State Variables

dot: Dottypes

State Invariant

None

Access Routine Semantics

new DotT(t):

- transition: $dot := t$
- output: $out := self$
- exception: $exc := none$

Point ADT Module

Template Module

PointT

Uses

n/a

Syntax

Exported Constants

None

Exported Types

PointT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new PointT	$\mathbb{Z}$, $\mathbb{Z}$	PointT	
row		$\mathbb{Z}$	
col		$\mathbb{Z}$	

Semantics

State Variables

r: $\mathbb{Z}$
c: $\mathbb{Z}$

State Invariant

None

Assumptions

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

Access Routine Semantics

new PointT(row, col):

- transition: $r, c := row, col$
- output: $out := self$
- exception: None

row():

- output: $out := r$
- exception: None

col():

- output: $out := c$
- exception: None

Connection ADT Module

Template Module

ConnectionT

Uses

PointT

Syntax

Exported Constants

None

Exported Types

ConnectionT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new ConnectionT	seq of PointT	ConnectionT	IllegalArgumentException
getPoints		seq of PointT	

Semantics

State Variables

points: seq of PointT

State Invariant

None

Assumptions

- The constructor new ConnectionT is called for each object instance before any other access routine is called for that object. The constructor cannot be called on an existing object.

- We assume that diagonal connections are not allowed, that is a dot must be directly above, below, to the right, or to the left to be considered a valid connection.
- We assume that overlapping connections and intersecting connections are allowed.

Access Routine Semantics

new ConnectionT(*points*):

- transition: *todo*
- output: *todo*
- exception: *todo*

getPoints():

- output: *todo*
- exception: *todo*

Local Functions

validPair: PointT, PointT $\rightarrow \mathbb{B}$
validPair(p1, p2) $\equiv$ [*insertsematicshere*]

Board ADT Module

Template Module

BoardT

Uses

DotT

Syntax

Exported Constants

None

Exported Types

BoardT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new BoardT	$\mathbb{N}$, $\mathbb{N}$, $\mathbb{Z}$	BoardT	IllegalArgumentException
getBoard		seq of (seq of DotT)	
getHeight		$\mathbb{N}$	
getWidth		$\mathbb{N}$	
getSeed		$\mathbb{Z}$	
validPoint	PointT	$\mathbb{B}$	
shufflePoint	PointT		IllegalArgumentException
toString		String	

Semantics

State Variables

board: seq of (seq of DotT) nRow: $\mathbb{N}$ nCol: $\mathbb{N}$ seed: $\mathbb{Z}$

State Invariant

None

Assumptions

- The constructor new BoardT is called for each object instance before any other access routine is called for that object. The constructor cannot be called on an existing object.
- We assume that the board is initially populated at random (uniformly), where each position has an equal probability of being a specific DotT type.

Access Routine Semantics

new BoardT($height, width, seed$):

- transition: [$insertsemanticshere$]
- output: [$insertsemanticshere$]
- exception: [$insertsemanticshere$]

getBoard():

- transition: [$insertsemanticshere$]
- output: [$insertsemanticshere$]
- exception: [$insertsemanticshere$]

getHeight():

- transition: [$insertsemanticshere$]
- output: [$insertsemanticshere$]
- exception: [$insertsemanticshere$]

getWidth():

- transition: [$insertsemanticshere$]
- output: [$insertsemanticshere$]
- exception: [$insertsemanticshere$]

getSeed():

- transition: [$insertsemanticshere$]

- output: [*insertsemanticshere*]
- exception: [*insertsemanticshere*]

validPoint(p):

- transition: [*insertsemanticshere*]
- output: [*insertsemanticshere*]
- exception: [*insertsemanticshere*]

shufflePoint(p):

- transition: [*insertsemanticshere*]
- output: [*insertsemanticshere*]
- exception: [*insertsemanticshere*]

toString():

- transition: [*insertsemanticshere*]
- output: [*insertsemanticshere*]
- exception: [*insertsemanticshere*]

Local Functions

validRow: $\mathbb{N} \to \mathbb{B}$
validRow(n) $\equiv 0 \leq n \leq (\text{nRow} - 1)$

validCol: $\mathbb{N} \to \mathbb{B}$
validCol(n) $\equiv 0 \leq n \leq (\text{nCol} - 1)$

Dots Game Module

Game Module

Dots

Uses

DotT
ConnectionT
BoardT

Syntax

Exported Constants

None

Exported Types

Game = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Game	$\mathbb{Z}$	Game	
move	ConnectionT		

Semantics

State Variables

gameboard: BoardT
moves: $\mathbb{N}$
objectiveColor: DotT
objectiveNum: $\mathbb{N}$
seed: $\mathbb{Z}$

State Invariant

n/a

Assumptions

- The constructor new BoardT is called for each object instance before any other access routine is called for that object. The constructor cannot be called on an existing object.
- We assume that after a move, the newly "empty" points in the board are populated at random uniformly, where each empty position has an equal probability of being a specific DotT type.
- We assume that the objective is to connect 10 of a certain color in 10 moves. If the target objective is reached (or surpassed) before the 10 moves is considered a win, while not completing the objective before the 10 moves is a failure.

Access Routine Semantics

new Game(*seed*):

- transition: [*insertsemanticshere*]
- output: [*insertsemanticshere*]
- exception: [*insertsemanticshere*]

move(c):

- transition: [*insertsemanticshere*]
- output: [*insertsemanticshere*]
- exception: [*insertsemanticshere*]

Local Functions

win
lose
reset

Questions

1. n/a
2. n/a