

# Assignment 4 Specification

SFWR ENG 2AA4

Jay Mody - 400195508 - modyj

March 25, 2020

[insert short introduction about program and MIS here]

## Example Module

### Module

n/a

### Uses

n/a

### Syntax

#### Exported Constants

n/a

#### Exported Types

n/a

#### Exported Access Programs

Routine name	In	Out	Exceptions
routine1	$\mathbb{Z}, \mathbb{Z}$	$\mathbb{Z}$	
routine2		$\mathbb{Z}$	
routine3		$\mathbb{Z}$	
routine3	$\mathbb{Z}, \mathbb{Z}$	$\mathbb{Z}$	

### Semantics

#### State Variables

n/a

#### State Invariant

n/a

#### Assumptions

n/a

## Access Routine Semantics

routine(*parameters*):

- transition: *variable* := *something*
- output: *out* := *something*
- exception: *exc* := *something*

## Local Functions

n/a

## Considerations

n/a

# 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:  $\mathbb{N} \rightarrow \mathbb{B}$

validRow( $n$ )  $\equiv 0 \leq n \leq (\text{nRow} - 1)$

validCol:  $\mathbb{N} \rightarrow \mathbb{B}$

validCol( $n$ )  $\equiv 0 \leq n \leq (\text{nCol} - 1)$

# Dots Game Module

## Game Module

Dots

## Uses

n/a

## Syntax

### Exported Constants

n/a

### Exported Types

n/a

### Exported Access Programs

Routine name	In	Out	Exceptions
routine1	$\mathbb{Z}, \mathbb{Z}$	$\mathbb{Z}$	
routine2		$\mathbb{Z}$	
routine3		$\mathbb{Z}$	
routine3	$\mathbb{Z}, \mathbb{Z}$	$\mathbb{Z}$	

## Semantics

### State Variables

n/a

### State Invariant

n/a

### Assumptions

n/a

## Access Routine Semantics

`routine(parameters):`

- transition: *variable := something*
- output: *out := something*
- exception: *exc := something*

## Local Functions

n/a

## Considerations

n/a

## Questions

1.  $n/a$
2.  $n/a$