Assignment 4 Specification

SFWR ENG 2AA4, COMP SCI 2ME3

Zefeng Wang, 400085065, wangz217

April 12, 2021

This Module Interface Specification (MIS) document contains modules, types and methods used to the game 2048. This a a popular game that aims to reach the target 2048 (or even more!).

Direction Module

Module

Direction

Uses

None

Syntax

Exported Constants

None

Exported Types

IndicatorT = {
UP,
DOWN,
LEFT,
RIGHT
}

Exported Access Programs

None

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Considerations

When implementing in Java, use enums.

Direction Module

Module

Command

Uses

None

Syntax

Exported Constants

None

Exported Types

Command = {
W, # *move up*
A, # *move left*
S, # *move down*
D, # *move right*
R, # *restart the game*
P, **# display the game board**
HELP, # *display the help page*
EXIT # *exit the game*
}

Exported Access Programs

None

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Considerations

When implementing in Java, use enums.

View Interface Module

Interface Module

View

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
render		String	

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

None

Cell Module

Module

Cell

Uses

None

Syntax

Exported Constants

None

Exported Types

Cell = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Cell		Cell of	
isChecked		$\mathbb{B}$	
check			
uncheck			
clear			
setValue	$\mathbb{N}$		
getValue		$\mathbb{N}$	

Semantics

State Variables

checked: boolean
value: $\mathbb{N}$

State Invariant

None

Assumptions

None

Access Routine Semantics

new Cell():

- transition: $checked$:= false, $value$: = 0
- output: out := $self$
- exception: none

isChecked():

- output: out := $checked$
- exception: none

check():

- transition: $checked$:= true
- exception: none

uncheck():

- transition: $checked$:= false
- exception: none

setValue(newValue):

- transition: $value := newValue$
- exception: none

clear():

- transition: $value$:= 0
- exception: none

getValue():

- out: $value$
- exception: none

Point Module

Module

Point

Uses

None

Syntax

Exported Constants

None

Exported Types

Point = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new Point	$\mathbb{N}$, $\mathbb{N}$	Cell of	
getX		$\mathbb{N}$	
getY		$\mathbb{N}$	

Semantics

State Variables

x: $\mathbb{N}$
y: $\mathbb{N}$

State Invariant

None

Assumptions

None

Access Routine Semantics

new Point(x, y):

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

getX():

- output: out := x
- exception: none

getY():

- output: out := y
- exception: none

ErrorMessage Module

Module

Cell

Uses

None

Syntax

Exported Constants

None

Exported Types

Cell = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new ErrorMessage		Errormessage of	
getMessage		String	
setmessage	String		

Semantics

State Variables

message:String

State Invariant

None

Assumptions

None

Access Routine Semantics

new ErrorMessage():

- transition: $message$:= "Error!"
- output: out := $self$
- exception: none

getMessage():

- output: out := $message$
- exception: none

setMessage(message):

- transition: $message$:= message
- exception: none

GameBoard Module

Module

GameBoard

Uses

None

Syntax

Exported Constants

None

Exported Types

GameBoard = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new GameBoard		GameBoard of	
getState		sequence of sequence of Cell	
setState	sequence of sequence of Cell		

Semantics

State Variables

state: sequence of sequence of Cell

State Invariant

None

Assumptions

None

Access Routine Semantics

new Cell():

- transition: $state$:= Cell[4][4]
- output: out := $self$
- exception: none

getState():

- output: out := $state$
- exception: none

setState(newState):

- transition: $checked$:= newState
- exception: none

ViewRegistry Interface Module

Interface Module

ViewRegistry

Uses

All modules that implements View interface

Syntax

Exported Constants

None

Exported Types

ViewRegistry = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new ViewRegistry		ViewRegistry	
get	String	View	
register	String, View		

Semantics

State Variables

viewMap: Map< String, View>

State Invariant

None

Assumptions

None

Access Routine Semantics

new ViewRegistry():

- transition: $viewMap$:= new Map()
- output: out := $self$
- exception: none

get(key):

- output: out := $viewMap.get(key)$
- exception: none

register(name, view):

- transition: $viewMap.put(name, view)$
- exception: $self$

WelcomeView Module

Module

Template WelcomeView Module inherits View

Uses

View

Syntax

Exported Constants

None

Exported Types

WelcomeView = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new WelcomeView		WelcomeView	
render		String	

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

new WelcomeView():

- output: $self$
- exception: none

render():

- output: # customized String for printing
- exception: none

ErrorView Module

Module

Template ErrorView Module inherits View

Uses

View

Syntax

Exported Constants

None

Exported Types

ErrorView = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new ErrorView		ErrorView	
render		String	

Semantics

State Variables

$errorMessage$: ErrorMessage

State Invariant

None

Assumptions

None

Access Routine Semantics

new WelcomeView():

- output: $self$
- exception: none

render():

- output: # customized String for printing
- exception: none

ErrorView Module

Module

Template ErrorView Module inherits View

Uses

View

Syntax

Exported Constants

None

Exported Types

ErrorView = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new ErrorView	ErrorMessage	ErrorView	
render		String	

Semantics

State Variables

$errorMessage$: String

State Invariant

None

Assumptions

None

Access Routine Semantics

new ErrorView(errorMessage):

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

setErrorMessage(message):

- transition: $errorMessage$:= message
- exception: none

render():

- output: # customized String for printing
- exception: none

HelpView Module

Module

Template HelpView Module inherits View

Uses

View

Syntax

Exported Constants

None

Exported Types

HelpVIew = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new HelpView		HelpView	
render		String	

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

new HelpView():

- output: $self$
- exception: none

render():

- output: # customized String for printing
- exception: none

GameBoard Module

Module

Template HelpView Module inherits View

Uses

View

Syntax

Exported Constants

None

Exported Types

GameBoardView = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new GameBoardView	GameBoard	GameBoardView	
render		String	
setTableCharacter	char		

Semantics

State Variables

gameBoard: GameBoard
tableCharacter: char

State Invariant

$DEFAULT\_TABLE\_CHAR$: char

Assumptions

None

Access Routine Semantics

new GameBoardView(board):

- transition: $gameBoard := board$
- output: $self$
- exception: none

render():

- output: # render the game board object using the table char.
- exception: none

setTableCharacter(c):

- transition: $tableCharacter :=$ c
- exception: none

RandomService Module

Module

RandomService

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

random		$\mathbb{R}$	
random	$\mathbb{N}$, $\mathbb{N}$	$\mathbb{N}$	
random2or4		$\mathbb{N}$	
pick	sequence of $\mathbb{N}$	$\mathbb{N}$	

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

random():

- output: out:= # random number in range [0, 1]; This is random number generation is assumed to be provided
- exception: none

random(lo, hi):

- output: $out := floor(random() \times floor(hi - lo + 1)) + lo$ # floor() is assumed to be provided by the language math library.
- exception: none

random2or4( ):

- output: $out := random() > 0.9 \implies 4 | True \implies 2$
- exception: none

pick(choices):

- output: $out := choices[random(0, choices.length - 1)]$
- exception: none

GameController Module

Module

GameController

Uses

GameBoard, Direction, RandomService, ViewRegistry

Syntax

Exported Constants

None

Exported Types

GameController = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new GameController	GameBoard, ErrorMessage, ViewRegistry	GameController	
reset		String	
checkGameOver		$\mathbb{B}$	
collapse	Direction	String	
displayWelcome		String	
displayBoard		String	
displayHelp		String	
displayError	String	String	

Semantics

State Variables

$board : GameBoard$
$errorMessage : ErrorMessage$
$viewRegistry : viewRegistry$

State Invariant

None

Assumptions

None

Access Routine Semantics

PASS

GameRunner Module

Module

GameRunner

Uses

GameController, Command

Syntax

Exported Constants

None

Exported Types

GameRunner = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new GameRunner	GameController	GameRunner	
process	String	String	
run			
exit			

Semantics

State Variables

gameController : GameController

State Invariant

None

Assumptions

The run method is going to be called to run the game.

Access Routine Semantics

PASS