Assignment 4, Part 1, Specification

SFWR ENG 2AA4

April 10, 2018

Submit your design specification, written in LaTeX, of the MIS for the game state module. If your specification requires additional modules, you should include their MISes as well. It is up to you to determine your modules interface; that is, you decide on the exported constants, access programs, exceptions etc. You also determine your state variables and specify the semantics for your access program calls. Your design does not need to concern itself with performance.

Cards ADT Module

Template Module

CardsT

Uses

N/A

Syntax

Exported Types

CardsT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
CardsT	Strings, Strings	CardsT	
getRank		Strings	
getSuit		Strings	
getColour		Strings	

Semantics

State Variables

$ranks$: Strings
$suits$: Strings

State Invariant

None

Assumptions

The constructor CardsT 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

CardsT(r, s):

- transition: $ranks, suits := r, s$
- output: $out := \mathit{self}$
- exception: None

getRank():

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

getSuit():

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

getColour():

- output: $out :=$ "$Heart$" $\vee$ "$Diamond$" $\implies$ "Red"$|$"$Black$"
- exception: None

CardsGame Module

Template Module

Game

Uses

CardsT

Syntax

Exported Types

CardsDeck = ?

Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
CardsDeck		CardsDeck	
ShufflingDeck			
DealingCards			
moveCard	Z, Z	Sequence of (Sequence of CardsT)	
moveCardFreecell	Z		
moveCardFoundation	Z, Z		
checkValidMove	Z, Z	B	
checkValidMoveFreecell		B	
checkValidMoveFoundation	Z, Z	B	
winningSituation		B	

Semantics

State Variables

$deck[52]$: CardsT
$columns$: Sequence of (Sequence of CardsT)
$freecell$: Sequence of CardsT
$foundation$: Sequence of (Sequence of CardsT)

State Invariant

SIZE = 52

Assumptions

The constructor CardsDeck 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

CardsDeck():

- transition: $i \in N | i \in [0..SIZE] : deck[i] = CardsT(ranks[i\%13], suits[i/13])$
- output: $out := self$
- exception: None

ShufflingDeck():

- transition: $i \in N, j \in N | i \in [0..SIZE], j = (rand() + time(0))\%SIZE : deck[i], deck[j] = deck[j], deck[i]$
- exception: None

DealingCards():

- transition: $i \in N | i \in [0..7] : i = 0 \implies deck[0..6] | i = 1 \implies deck[7..13] | i = 2 \implies deck[14..20] | i = 3 \implies deck[21..27] | i = 4 \implies deck[28..33] | i = 5 \implies deck[34..39] | i = 6 \implies deck[40..45] | i = 7 \implies deck[46..51] |$
- exception: None

moveCard(from,to):

- transition: $(checkValidMove(from, to) = true) \implies (columns.at(to).back() = columns.at(from).back())$
- output: $out := columns$
- exception: None

moveCardFreecell(from):

- transition: $(checkValidMoveFreecell() = true) \implies (freecell.back() = columns.at(from).back())$
- exception: None

moveCardFoundation(from,to):

- transition: $(checkValidMoveFoundation(from, to) = true) \implies (foundation.at(to).back() = columns.at(from).back())$
- output: $out := columns$
- exception: None

checkValidMove(from,to):

- output: $out := (columns.at(to).back().getRank().index() - columns.at(from).back().getRank().inde$ $1) \wedge (columns.at(from).back().getColour()! = columns.at(to).back().getColour()) \implies true$
- exception: None

checkValidMoveFreecell():

- output: $out := freecell.size() < 4 \implies true$
- exception: None

checkValidMoveFoundation(from,to):

- output: $out := (foundation.at(to).back().getRank().index() - columns.at(from).back().getRank().i$ $1) \wedge (columns.at(from).back().getSuit() = foundation.at(to).back().getSuit()) \implies true$
- exception: None

winningSituation():

- output: $+(i \in N | i \in [0..4] | foundation.at(i).size() = 13 : 1) = 4 \implies true$
- exception: None