Assignment 4 Specification

SFWR ENG 2AA4

April 10, 2018

Freecell is a popular csolitaire card game that utilizes a set of a full set of 52 cards. Unlike other solitaire games, the chances of getting an unsolvable problem is highly unlikely. This c++ implementation of Freecell simulates the different states a player can find themselves in while playing the game. This Document will provide the descriptive specifications behind the program, using proper modularization and adhering to other software engineering principles.

The implementation uses a minimal interface which can clearly be observed based on the use of frequent methods that are aimed to do one job. The interface is general, and allows users to move cards from pile to pile just as one would during a game.

Card ADT Module

Template Module

Card

Uses

N/A

Syntax

Exported Types

Card = ?
Suit = {C, D, S, H}
Colour = {Red, Black}
Rank = {Empty, ace, two, three, four, five, six, seven , eight, nine, ten, jack, queen, king}

Exported Access Programs

Routine name	In	Out	Exceptions
Card	Rank, Suit	Card	
getRank		Rank	
getColour		Colour	
getSuit		Suit	

Semantics

State Variables

rank: Rank
suit: Suit
colour: Colour

State Invariant

None

Assumptions

The constructor Card 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

Card(r, s):

- transition: $rank, suit, colour := r, s, (s = D \vee s = H \Rightarrow Red \| True \Rightarrow Black)$
- output: $out := self$
- exception: None

getColour():

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

getRank():

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

getSuit():

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

Tableau ADT Module

Template Module

Tableau

Uses

Card

Syntax

Exported Types

Tableau = ?

Exported Access Programs

Routine name	In	Out	Exceptions
Tableau	seq of Card	Tableau	
addCard	Card		invalid_move
removeCard			not_available
topCard		Card	not_available

Semantics

State Variables

tab: seq of Card

State Invariant

None

Assumptions

The constructor Tableau 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

Tableau(s):

- transition: $tab := s$
- output: $out := self$
- exception: None

addCard(c):

- transition: $tab := tab || < c >$
- exception: ($c.rank{+}1 \neq tab_{|tab|-1}.rank \vee c.colour = tab_{|tab|-1}.colour \Rightarrow$ invalid_move)

removeCard():

- transition: $tab := [0..tab_{|tab|-2}]$
- exception: ($|tab| = 0 \Rightarrow not\_available$)

topCard():

- output: $out := tab_{|tab|-1}$
- exception: ($|tab| = 0 \Rightarrow not\_available$)

Foundation ADT Module

Template Module

Foundation

Uses

Card

Syntax

Exported Types

Foundation = ?

Exported Access Programs

Routine name	In	Out	Exceptions
Foundation	Suit	Foundation	
addCard	Card		invalid_move
removeCard			not_available
isFull		$\mathbb{B}$	
topCard		Card	not_available

Semantics

State Variables

pile: sequence of Card suit: Suit

State Invariant

None

Assumptions

- The constructor Foundation 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

Foundation(s):

- transition: $suit := s$
- output: $out := self$
- exception: None

addCard(c):

- transition: $pile := pile || < c >$
- exception: $(c.rank{+}1 \neq pile_{|pile|-1}.rank \vee c.colour \neq pile_{|pile|-1}.colour \Rightarrow$ invalid_move)

removeCard():

- transition: $pile := [0..pile_{|pile|-2}]$
- exception: $|pile| = 0 \Rightarrow not\_available$

topCard():

- output: $out := pile_{|pile|-1}$
- exception: $|pile| = 0 \Rightarrow not\_available$

isFull():

- output: $out := (|pile| = 0 \Rightarrow True|False)$
- exception: None

getSuit():

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

FreeCell ADT Module

Template Module

FreeCell

Uses

Card

Syntax

Exported Types

FreeCell = ?

Exported Access Programs

Routine name	In	Out	Exceptions
FreeCell		FreeCell	
addCard	Card		full
removeCard	Rank, Suit		not_available
isFull		$\mathbb{B}$	
searchCard	Rank, Suit	Card	not_available

Semantics

State Variables

cells: sequence of Card

State Invariant

None

Assumptions

- The constructor FreeCell 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

FreeCell():

- transition: None
- output: $out := self$
- exception: None

addCard(c):

- transition: $cells := cells || < c >$
- exception: $(isFull() \Rightarrow$ full)

removeCard(rank, suit):

- transition: $cells := (\forall i : \mathbb{Z} \mid 0 <= i < 4 : (cells_i.rank = rank \wedge cells_i.suit = suit) \Rightarrow cells - \{cells_i\})$
- exception: $\neg(\exists i : \mathbb{Z} \mid 0 <= i < 4 : (cells_i.rank = rank \wedge cells_i.suit = suit)) \Rightarrow not\_available$

searchCard(rank, suit):

- output: $cells := (\forall i : \mathbb{Z} \mid 0 <= i < 4 : (cells_i.rank = rank \wedge cells_i.suit = suit) \Rightarrow cells_i)$
- exception: $\neg(\exists i : \mathbb{Z} \mid 0 <= i < 4 : (cells_i.rank = rank \wedge cells_i.suit = suit)) \Rightarrow not\_available$

isFull():

- output: $out := (|cells| = 4 \Rightarrow True|False)$
- exception: None

Setup ADT Module

Template Module

Setup

Uses

Card, FreeCell, Tableau, Foundation

Syntax

Exported Types

Setup = ?

Exported Access Programs

Routine name	In	Out	Exceptions
Setup	seq (seq of Card))	Setup	
tabToTab	$\mathbb{Z}, \mathbb{Z}$		
tabToFound	$\mathbb{Z}$		
freeToTab	Rank, Suit, $\mathbb{Z}$		
tabToFree	$\mathbb{Z}$		full
freeToFound	Rank, Suit		
foundToTab	Suit, $\mathbb{Z}$		
winningGame		bool	

Semantics

State Variables

board: seq of Tableau
founds: seq of Foundation
free: FreeCell

State Invariant

None

Assumptions

- The constructor Setup() 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

Setup(s):

- transition: $(\forall i : Z | 0 <= i < 7 : board_i = Tableau(s_i))$
- output: $out := self$
- exception: None

tabToTab(from, to):

- transition: $board_{to}, board_{from} := board_{to} || < (board_{from}.topCard()) >, board_{from}.removeCard()$
- exception: none

tabToFound(from):

- transition: $(\forall i : \mathbb{Z} \mid 0 <= i < 4 : (founds_i.getSuit() = board_{from}.topcard().getSuit()) \Rightarrow (founds_i, board_{from} := founds_i || < (board_{from}.topCard()) >, board_{from}.removeCard()$
- exception: none

freeToTab(rank, suit, to):

- transition: $board_{to}, free :=$ $board_{to} || < (Card(rank, suit)) >, [free_0..free_{free-1}] - Card(rank, suit)$
- exception: None

tabToFree(from):

- transition: $free, board_{from} :=$ $free || < (board_{from}.topCard()) >, board_{from}.removeCard()$
- exception: $(free.isFull()) \Rightarrow full$

freeToFound(rank,suit):

- transition:
  $(\forall i : \mathbb{Z} \mid 0 <= i < 4 : (founds_i.getSuit() = free.searchCard(rank, suit)).topcard().getSuit()) \Rightarrow$

  $(founds_i, free := founds_i || < (free.searchCard(rank, suit))) >, free.removeCard(rank, suit)$
- exception: none

foundToTab(rank,to):

- transition:
  $(\forall i : \mathbb{Z} \mid 0 <= i < 4 : (founds_i.getSuit() = suit) \Rightarrow$
  $(board_{to}, founds_i := board_{to} || < (founds_i.topCard()) >, founds_i.removeCard()$
- exception: none

winningGame():

- output: $\wedge\ (i : \mathbb{N} | i \in [0..3] : founds_i.isFull())$
- exception: none

Local Functions

- tabTopCards: A method that outputs the top cards on the tableau to clearly see what is available suring a specific state.