**EECS 3311: SOFTWARE DESIGN**

**SUMMER 2015**

**ASSIGNMENT: PEG SOLITAIRE GAME**

**Camillo John (CJ) D’Alimonte [cjdal34]**

**&**

**Dinesh Kalia [dinesh49]**

**Course Instructor: Jackie Wang**

**Submitting Account: cjdal34**

**Date: July 10th, 2015**

Table of Contents

**Contract Views3**

GAME class3

BOARD class8

**BON Diagram13**

**CONTRACT VIEW**

GAME class

note

description: "A game of peg solitaire."

author: "Camillo John (CJ) D'Alimonte & Dinesh Kalia"

date: "July 10th 2015"

revision: "$Revision$"

class interface

GAME

create

make\_from\_board,

make\_easy,

make\_cross,

make\_plus,

make\_pyramid,

make\_arrow,

make\_diamond,

make\_skull

feature *-- Auxiliary Routines*

boolean\_to\_yes\_no (b: BOOLEAN): STRING\_**8**

*-- 'Yes' or 'No' corresponding to 'b'.*

feature *-- Board*

board: BOARD

bta: BOARD\_TEMPLATES\_ACCESS

feature *-- Commands*

move\_down (r, c: INTEGER\_**32**)

require

from\_slot\_valid\_column: board.is\_valid\_column (c)

from\_slot\_valid\_row: board.is\_valid\_row (r)

middle\_slot\_valid\_row: board.is\_valid\_row (r + **1**)

to\_slot\_valid\_row: board.is\_valid\_row (r + **2**)

from\_slot\_occupied: board.status\_of (r, c) ~ board.occupied\_slot

middle\_slot\_occupied: board.status\_of (r + **1**, c) ~ board.occupied\_slot

to\_slot\_unoccupied: board.status\_of (r + **2**, c) ~ board.unoccupied\_slot

ensure

slots\_properly\_set:

board.status\_of (r, c) ~ board.unoccupied\_slot and board.status\_of (r + **1**, c) ~ board.unoccupied\_slot and board.status\_of (r + **2**, c) ~ board.occupied\_slot

other\_slots\_unchanged:

board.matches\_slots\_except (board, r, r + **2**, c, c)

move\_left (r, c: INTEGER\_**32**)

require

from\_slot\_valid\_row: board.is\_valid\_row (r)

from\_slot\_valid\_column: board.is\_valid\_column (c)

middle\_slot\_valid\_column: board.is\_valid\_column (c - **1**)

to\_slot\_valid\_column: board.is\_valid\_column (c - **2**)

from\_slot\_occupied: board.status\_of (r, c) ~ board.occupied\_slot

middle\_slot\_occupied: board.status\_of (r, c - **1**) ~ board.occupied\_slot

to\_slot\_unoccupied: board.status\_of (r, c - **2**) ~ board.unoccupied\_slot

ensure

slots\_properly\_set:

board.status\_of (r, c) ~ board.unoccupied\_slot and board.status\_of (r, c - **1**) ~ board.unoccupied\_slot and board.status\_of (r, c - **2**) ~ board.occupied\_slot

other\_slots\_unchanged:

board.matches\_slots\_except (board, r, r, c, c - **2**)

move\_right (r, c: INTEGER\_**32**)

require

from\_slot\_valid\_row: board.is\_valid\_row (r)

from\_slot\_valid\_column: board.is\_valid\_column (c)

middle\_slot\_valid\_column: board.is\_valid\_column (c + **1**)

to\_slot\_valid\_column: board.is\_valid\_column (c + **2**)

from\_slot\_occupied: board.status\_of (r, c) ~ board.occupied\_slot

middle\_slot\_occupied: board.status\_of (r, c + **1**) ~ board.occupied\_slot

to\_slot\_unoccupied: board.status\_of (r, c + **2**) ~ board.unoccupied\_slot

ensure

slots\_properly\_set:

board.status\_of (r, c) ~ board.unoccupied\_slot and board.status\_of (r, c + **1**) ~ board.unoccupied\_slot and board.status\_of (r, c + **2**) ~ board.occupied\_slot

other\_slots\_unchanged:

board.matches\_slots\_except (board, r, r, c, c + **2**)

move\_up (r, c: INTEGER\_**32**)

require

from\_slot\_valid\_column: board.is\_valid\_column (c)

from\_slot\_valid\_row: board.is\_valid\_row (r)

middle\_slot\_valid\_row: board.is\_valid\_row (r - **1**)

to\_slot\_valid\_row: board.is\_valid\_row (r - **2**)

from\_slot\_occupied: board.status\_of (r, c) ~ board.occupied\_slot

middle\_slot\_occupied: board.status\_of (r - **1**, c) ~ board.occupied\_slot

to\_slot\_unoccupied: board.status\_of (r - **2**, c) ~ board.unoccupied\_slot

ensure

slots\_properly\_set:

board.status\_of (r, c) ~ board.unoccupied\_slot and board.status\_of (r - **1**, c) ~ board.unoccupied\_slot and board.status\_of (r - **2**, c) ~ board.occupied\_slot

other\_slots\_unchanged:

board.matches\_slots\_except (board, r, r - **2**, c, c)

feature *-- Constructors*

make\_arrow

*-- Initialize a game with Arrow board.*

ensure

board\_set: board ~ bta.Templates.arrow\_board

make\_cross

*-- Initialize a game with Cross board.*

ensure

board\_set: board ~ bta.Templates.cross\_board

make\_diamond

*-- Initialize a game with Diamond board.*

ensure

board\_set: board ~ bta.Templates.diamond\_board

make\_easy

*-- Initialize a game with easy board.*

ensure

board\_set: board ~ bta.Templates.easy\_board

make\_from\_board (new\_board: BOARD)

*-- Initialize a game with 'new\_board'.*

ensure

board\_set: board ~ new\_board

make\_plus

*-- Initialize a game with Plus board.*

ensure

board\_set: board ~ bta.Templates.plus\_board

make\_pyramid

*-- Initialize a game with Pyramid board.*

ensure

board\_set: board ~ bta.Templates.pyramid\_board

make\_skull

*-- Initialize a game with Skull board.*

ensure

board\_set: board ~ bta.Templates.skull\_board

feature *-- Output*

out: STRING\_**8**

*-- String representation of current game.*

*-- Do not modify this feature!*

feature *-- Status Queries*

is\_over: BOOLEAN

*-- Is the current game 'over'?*

*-- i.e., no further movements are possible.*

ensure

correct\_result: Result = False or not (across

**1** |..| board.number\_of\_rows as r

some

across

**1** |..| board.number\_of\_columns as c

some

(board.status\_of (r.item, c.item) ~ board.occupied\_slot) and

(board.is\_valid\_row (r.item) and board.is\_valid\_column (c.item) and board.is\_valid\_column (c.item - **1**) and board.is\_valid\_column (c.item - **2**) and board.status\_of (r.item, c.item) ~ board.occupied\_slot and board.status\_of (r.item, c.item - **1**) ~ board.occupied\_slot and board.status\_of (r.item, c.item - **2**) ~ board.unoccupied\_slot)

or (board.is\_valid\_row (r.item) and board.is\_valid\_column (c.item) and board.is\_valid\_column (c.item + **1**) and board.is\_valid\_column (c.item + **2**) and board.status\_of (r.item, c.item) ~ board.occupied\_slot and board.status\_of (r.item, c.item + **1**) ~ board.occupied\_slot and board.status\_of (r.item, c.item + **2**) ~ board.unoccupied\_slot)

or (board.is\_valid\_column (c.item) and board.is\_valid\_row (r.item) and board.is\_valid\_row (r.item + **1**) and board.is\_valid\_row (r.item + **2**) and board.status\_of (r.item, c.item) ~ board.occupied\_slot and board.status\_of (r.item + **1**, c.item) ~ board.occupied\_slot and board.status\_of (r.item + **2**, c.item) ~ board.unoccupied\_slot)

or (board.is\_valid\_column (c.item) and board.is\_valid\_row (r.item) and board.is\_valid\_row (r.item - **1**) and board.is\_valid\_row (r.item - **2**) and board.status\_of (r.item, c.item) ~ board.occupied\_slot and board.status\_of (r.item - **1**, c.item) ~ board.occupied\_slot and board.status\_of (r.item - **2**, c.item) ~ board.unoccupied\_slot)

end

end)

is\_won: BOOLEAN

*-- Has the current game been won?*

*-- i.e., there's only one occupied slot on the board.*

ensure

game\_won\_iff\_one\_occupied\_slot\_left: Result implies board.number\_of\_occupied\_slots ~ **1**

winning\_a\_game\_means\_game\_over: is\_won implies is\_over

end *-- class GAME*

**CONTRACT VIEW**

BOARD class

note

description: "A board for the peg solitaire game."

author: "Camillo John (CJ) D'Alimonte & Dinesh Kalia"

date: "July 10th 2015"

revision: "$Revision$"

class interface

BOARD

create

make\_default,

make\_easy,

make\_cross,

make\_plus,

make\_pyramid,

make\_arrow,

make\_diamond,

make\_skull

feature *-- Auxiliary Commands*

set\_status (r, c: INTEGER\_**32**; status: SLOT\_STATUS)

*-- Set the status of slot at row 'r' and column 'c' to 'status'.*

require

valid\_row: is\_valid\_row (r)

valid\_column: is\_valid\_column (c)

ensure

slot\_set: Current.status\_of (r, c) ~ status

slots\_not\_in\_range\_unchanged: matches\_slots\_except (Current, r, r, c, c) =

old matches\_slots\_except (Current, r, r, c, c)

set\_statuses (r**1**, r**2**, c**1**, c**2**: INTEGER\_**32**; status: SLOT\_STATUS)

*-- Set the range of slots to 'status':*

*-- intersection of rows 'r1' to 'r2' and*

*-- columns 'c1' to 'c2'.*

require

valid\_rows: is\_valid\_row (r**1**) and is\_valid\_row (r**2**)

valid\_columns: is\_valid\_column (c**1**) and is\_valid\_column (c**2**)

valid\_row\_range: r**2** >= r**1**

valid\_column\_range: c**2** >= c**1**

ensure

slots\_in\_range\_set: across

r**1** |..| r**2** as r

all

across

c**1** |..| c**2** as c

all

Current.status\_of (r.item, c.item) ~ status

end

end

slots\_not\_in\_range\_unchanged: matches\_slots\_except (Current, r**1**, r**2**, c**1**, c**2**) =

old matches\_slots\_except (Current, r**1**, r**2**, c**1**, c**2**)

feature *-- Auxiliary Queries*

matches\_slots\_except (other: BOARD; r**1**, r**2**, c**1**, c**2**: INTEGER\_**32**): BOOLEAN

*-- Do slots outside the intersection of*

*-- rows 'r1' to 'r2' and columns 'c1' and 'c2'*

*-- match in Current and 'other'.*

require

consistent\_row\_numbers: Current.number\_of\_rows ~ other.number\_of\_rows

consistent\_column\_numbers: Current.number\_of\_columns ~ other.number\_of\_columns

valid\_rows: is\_valid\_row (r**1**) and is\_valid\_row (r**2**)

valid\_columns: is\_valid\_column (c**1**) and is\_valid\_column (c**2**)

valid\_row\_range: r**2** >= r**1**

valid\_column\_range: c**2** >= c**1**

ensure

correct\_result: across

**1** |..| number\_of\_rows as r

all

across

**1** |..| number\_of\_columns as c

all

r.item < r**1** or r.item > r**2** and c.item < c**1** or c.item > c**2** implies

Current.status\_of (r.item, c.item) ~ other.status\_of (r.item, c.item)

end

end

occupied\_slot: OCCUPIED\_SLOT

*-- A slot available for movement but currently occupied.*

ensure

Result = ssa.Occupied\_slot

unavailable\_slot: UNAVAILABLE\_SLOT

*-- A slot not available for movement.*

ensure

Result = ssa.Unavailable\_slot

unoccupied\_slot: UNOCCUPIED\_SLOT

*-- A slot available for movement and currently unoccupied.*

ensure

Result = ssa.Unoccupied\_slot

feature *-- Constructor*

make\_arrow

*-- Initialize a Arrow board.*

ensure

board\_set: Current ~ bta.Templates.arrow\_board

make\_cross

*-- Initialize a Cross board.*

ensure

board\_set: Current ~ bta.Templates.cross\_board

make\_default

*-- Initialize a default board with all slots unavailable.*

ensure

board\_set: Current ~ bta.Templates.default\_board

make\_diamond

*-- Initialize a Diamond board.*

ensure

board\_set: Current ~ bta.Templates.diamond\_board

make\_easy

*-- Initialize an easy board.*

ensure

board\_set: Current ~ bta.Templates.easy\_board

make\_plus

*-- Initialize a Plus board.*

ensure

board\_set: Current ~ bta.Templates.plus\_board

make\_pyramid

*-- Initialize a Pyramid board.*

ensure

board\_set: Current ~ bta.Templates.pyramid\_board

make\_skull

*-- Initialize a Skull board.*

ensure

board\_set: Current ~ bta.Templates.skull\_board

feature *-- Equality*

is\_equal (other: like Current): BOOLEAN

*-- Is current board equal to 'other'?*

ensure then

correct\_result: True

Result implies Current.out ~ other.out

feature *-- Output*

out: STRING\_**8**

*-- String representation of current board.*

feature *-- Queries*

is\_valid\_column (c: INTEGER\_**32**): BOOLEAN

*-- Is 'x' a valid column number?*

ensure

correct\_result: Result implies **1** <= c and c <= number\_of\_rows

is\_valid\_row (r: INTEGER\_**32**): BOOLEAN

*-- Is 'r' a valid row number?*

ensure

correct\_result: Result implies **1** <= r and r <= number\_of\_rows

number\_of\_columns: INTEGER\_**32**

*-- Number of columns in the board of game.*

ensure

correct\_result: Result = imp.width

number\_of\_occupied\_slots: INTEGER\_**32**

*-- Number of slots occupied by pegs on current board.*

number\_of\_rows: INTEGER\_**32**

*-- Number of rows in the board of game.*

ensure

correct\_result: Result = imp.height

status\_of (r, c: INTEGER\_**32**): SLOT\_STATUS

*-- Is the slot at row 'r' and column 'c'*

*-- unavailable, occupied, or unoccupied?*

require

valid\_row: is\_valid\_row (r)

valid\_column: is\_valid\_column (c)

ensure

correct\_result: Result = imp.item (r, c)

end *-- class BOARD*

