EECS 3311 (Summer 2015)

Assignment 1

Submission By:

Thalammeherage Perera

CSE11011

#211429610

Table of Contents

Contract view (BOARD) 2

Contract view (GAME) 6

# Contract view (BOARD)

note

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

author: ""

date: "$Date$"

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

parse\_slot\_map (map: STRING\_8)

-- Parses a string containing slot definitions to the current board

require

valid\_input: map.split ('%N'.to\_character\_32).count = number\_of\_rows and across

map.split ('%N'.to\_character\_32) as line\_iter

all

line\_iter.item.count = number\_of\_columns and across

line\_iter.item as character\_iter

all

character\_iter.item.out ~ unavailable\_slot.out or else character\_iter.item.out ~ occupied\_slot.out or else character\_iter.item.out ~ unoccupied\_slot.out

end

end

ensure

correct\_slots\_assigned: across

1 |..| number\_of\_rows as rr

all

across

1 |..| number\_of\_columns as cc

all

imp.item (rr.item, cc.item).out ~ map.at (((rr.item - 1) \* (number\_of\_rows + 1)) + cc.item).out

end

end

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: imp.item (r, c).is\_equal (status)

slots\_not\_in\_range\_unchanged: matches\_slots\_except (old Current.deep\_twin, r, r, c, c)

set\_statuses (r1, r2, c1, c2: 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 (r1) and is\_valid\_row (r2)

valid\_columns: is\_valid\_column (c1) and is\_valid\_column (c2)

valid\_row\_range: r1 <= r2

valid\_column\_range: c1 <= c2

ensure

slots\_in\_range\_set: across

r1 |..| r2 as rr

all

across

c1 |..| c2 as cc

all

imp.item (rr.item, cc.item).is\_equal (status)

end

end

slots\_not\_in\_range\_unchanged: matches\_slots\_except (old Current.deep\_twin, r1, r2, c1, c2)

feature -- Auxiliary Queries

matches\_slots\_except (other: BOARD; r1, r2, c1, c2: 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: number\_of\_columns = other.number\_of\_columns

consistent\_column\_numbers: number\_of\_rows = other.number\_of\_rows

valid\_rows: is\_valid\_row (r1) and is\_valid\_row (r2)

valid\_columns: is\_valid\_column (c1) and is\_valid\_column (c2)

valid\_row\_range: r1 <= r2

valid\_column\_range: c1 <= c2

ensure

correct\_result: Result ~ across

1 |..| number\_of\_columns as col\_iter

all

across

1 |..| number\_of\_rows as row\_iter

all

(col\_iter.item < c1 and col\_iter.item > c2) or (row\_iter.item < r1 and row\_iter.item > r2) implies other.status\_of (row\_iter.item, col\_iter.item).is\_equal (status\_of (row\_iter.item, col\_iter.item))

end

end

occupied\_slot: OCCUPIED\_SLOT

-- A slot available for moment 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 moment 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: Result = (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 = (c > 0 and c <= number\_of\_columns)

is\_valid\_row (r: INTEGER\_32): BOOLEAN

-- Is 'r' a valid row number?

ensure

correct\_result: True

Result = (r > 0 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: True

Result ~ imp.item (r, c)

end -- class BOARD

# Contract view (GAME)

note

description: "A game of peg solitaire."

author: ""

date: "$Date$"

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'.

is\_movable (r, c: INTEGER\_32): BOOLEAN

is\_valid\_move (r1, r2, c1, c2: INTEGER\_32): BOOLEAN

-- Returns true if a peg in r1, c1 can be moved to r2, c2

require

valid\_interval: ((r1 - r2).abs = 0 and (c1 - c2).abs = 2) or ((r1 - r2).abs = 2 and (c1 - c2).abs = 0)

valid\_start: board.is\_valid\_column (c1) and board.is\_valid\_row (r1) and board.status\_of (r1, c1) ~ board.occupied\_slot

ensure

correct\_result: Result implies board.is\_valid\_column (c2) and board.is\_valid\_row (r2) and then ((c1 > c2 implies board.status\_of (r1, c1 - 1) ~ board.occupied\_slot) and (c1 < c2 implies board.status\_of (r1, c1 + 1) ~ board.occupied\_slot) and (r1 > r2 implies board.status\_of (r1 - 1, c1) ~ board.occupied\_slot) and (r1 < r2 implies board.status\_of (r1 + 1, c1) ~ board.occupied\_slot) and board.status\_of (r2, c2) ~ board.unoccupied\_slot)

feature -- Board

board: BOARD

bta: BOARD\_TEMPLATES\_ACCESS

feature -- Commands

move\_down (r, c: INTEGER\_32)

-- Moves a peg at row r and column c to two slots below

-- given that the middle slot is occupied and

-- bottom most is slot is unoccupied

require

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

from\_slot\_valid\_row: board.is\_valid\_column (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 (old board.deep\_twin, r, r + 2, c, c)

move\_left (r, c: INTEGER\_32)

-- Moves a peg at row r and column c to two slots left

-- given that the middle slot is occupied and

-- left most is slot is unoccupied

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 (old board.deep\_twin, r, r, c - 2, c)

move\_right (r, c: INTEGER\_32)

-- Moves a peg at row r and column c to two slots right

-- given that the middle slot is occupied and

-- right most is slot is unoccupied

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 (old board.deep\_twin, r, r, c, c + 2)

move\_up (r, c: INTEGER\_32)

-- Moves a peg at row r and column c to two slots above

-- given that the middle slot is occupied and

-- top most is slot is unoccupied

require

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

from\_slot\_valid\_row: board.is\_valid\_column (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 (old board.deep\_twin, r - 2, r, 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 = across

1 |..| board.number\_of\_rows as ri

all

across

1 |..| board.number\_of\_columns as ci

all

board.status\_of (ri.item, ci.item) ~ board.occupied\_slot implies not is\_movable (ri.item, ci.item)

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: Result implies is\_over

end -- class GAME