Course: EECS 3311

Semester: FALL 2017

Name: Mustafa Alsoufi

CSE login: mzuhair

**Contract View**

**Board Class**

note

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

author: "Mustafa Alsoufi and Jackie Wang"

date: "$Date$"

revision: "$Revision$"

class

BOARD

inherit

ANY

redefine

out,

is\_equal

end

create

make\_default,

make\_easy,

make\_cross,

make\_plus,

make\_pyramid,

make\_arrow,

make\_diamond,

make\_skull

feature -- Constructor

make\_default

-- Initialize a default board with all slots unavailable.

do

create imp.make\_filled (unavailable\_slot, 7, 7)

ensure

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

end

make\_easy

-- Initialize an easy board.

do

make\_default

set\_status (1, 4, unoccupied\_slot)

set\_status (4, 4, unoccupied\_slot)

set\_status (6, 4, unoccupied\_slot)

set\_statuses (2, 3, 4, 4, occupied\_slot)

set\_status (5, 4, occupied\_slot)

ensure

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

end

make\_cross

-- Initialize a Cross board.

do

make\_easy

create imp.make\_filled (unavailable\_slot, 7, 7)

set\_statuses (1, 2, 3, 3, unoccupied\_slot)

set\_statuses (1, 2, 5, 5, unoccupied\_slot)

set\_statuses (3, 4, 1, 1, unoccupied\_slot)

set\_statuses (3, 4, 2, 2, unoccupied\_slot)

set\_statuses (4, 5, 3, 3, unoccupied\_slot)

set\_statuses (4, 5, 5, 5, unoccupied\_slot)

set\_statuses (3, 4, 6, 6, unoccupied\_slot)

set\_statuses (3, 4, 7, 7, unoccupied\_slot)

set\_statuses (6, 7, 3, 3, unoccupied\_slot)

set\_statuses (6, 7, 4, 4, unoccupied\_slot)

set\_statuses (6, 7, 5, 5, unoccupied\_slot)

set\_status (1, 4, unoccupied\_slot)

set\_status (5, 1, unoccupied\_slot)

set\_status (5, 2, unoccupied\_slot)

set\_status (5, 6, unoccupied\_slot)

set\_status (5, 7, unoccupied\_slot)

set\_statuses (2, 3, 4, 4, occupied\_slot)

set\_statuses (4, 5, 4, 4, occupied\_slot)

set\_status (3, 3, occupied\_slot)

set\_status (3, 5, occupied\_slot)

ensure

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

end

make\_plus

-- Initialize a Plus board.

do

create imp.make\_filled (unavailable\_slot, 7, 7)

set\_statuses (1, 2, 3, 3, unoccupied\_slot)

set\_statuses (1, 2, 5, 5, unoccupied\_slot)

set\_statuses (3, 4, 1, 1, unoccupied\_slot)

set\_statuses (3, 4, 7, 7, unoccupied\_slot)

set\_statuses (5, 5, 1, 2, unoccupied\_slot)

set\_statuses (5, 6, 3, 3, unoccupied\_slot)

set\_statuses (5, 6, 5, 5, unoccupied\_slot)

set\_statuses (5, 5, 6, 7, unoccupied\_slot)

set\_statuses (7, 7, 3, 4, unoccupied\_slot)

set\_status (1, 4, unoccupied\_slot)

set\_status (3, 2, unoccupied\_slot)

set\_status (3, 3, unoccupied\_slot)

set\_status (3, 5, unoccupied\_slot)

set\_status (3, 6, unoccupied\_slot)

set\_status (7, 5, unoccupied\_slot)

set\_statuses (2, 3, 4, 4, occupied\_slot)

set\_statuses (4, 4, 2, 3, occupied\_slot)

set\_statuses (4, 4, 4, 5, occupied\_slot)

set\_statuses (5, 6, 4, 4, occupied\_slot)

set\_status (4, 6, occupied\_slot)

ensure

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

end

make\_pyramid

-- Initialize a Pyramid board.

do

create imp.make\_filled (unavailable\_slot, 7, 7)

set\_statuses (1, 2, 3, 3, unoccupied\_slot)

set\_statuses (1, 2, 5, 5, unoccupied\_slot)

set\_statuses (3, 3, 1, 2, unoccupied\_slot)

set\_statuses (6, 7, 3, 3, unoccupied\_slot)

set\_statuses (6, 7, 4, 4, unoccupied\_slot)

set\_statuses (6, 7, 5, 5, unoccupied\_slot)

set\_status (1, 4, unoccupied\_slot)

set\_status (4, 7, unoccupied\_slot)

set\_status (4, 1, unoccupied\_slot)

set\_status (3, 7, unoccupied\_slot)

set\_status (3, 5, unoccupied\_slot)

set\_status (3, 6, unoccupied\_slot)

set\_statuses (4, 4, 2, 3, occupied\_slot)

set\_statuses (4, 4, 4, 5, occupied\_slot)

set\_statuses (4, 5, 6, 6, occupied\_slot)

set\_statuses (5, 5, 1, 2, occupied\_slot)

set\_statuses (5, 5, 3, 4, occupied\_slot)

set\_statuses (2, 3, 4, 4, occupied\_slot)

set\_status (3, 3, occupied\_slot)

set\_status (3, 5, occupied\_slot)

set\_status (5, 5, occupied\_slot)

set\_status (5, 7, occupied\_slot)

ensure

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

end

make\_arrow

-- Initialize a Arrow board.

do

create imp.make\_filled (unavailable\_slot, 7, 7)

set\_statuses (4, 5, 1, 1, unoccupied\_slot)

set\_statuses (4, 5, 2, 2, unoccupied\_slot)

set\_statuses (4, 5, 3, 3, unoccupied\_slot)

set\_statuses (4, 5, 5, 5, unoccupied\_slot)

set\_statuses (4, 5, 6, 6, unoccupied\_slot)

set\_statuses (4, 5, 7, 7, unoccupied\_slot)

set\_status (1, 3, unoccupied\_slot)

set\_status (1, 5, unoccupied\_slot)

set\_status (3, 1, unoccupied\_slot)

set\_status (3, 7, unoccupied\_slot)

set\_statuses (1, 2, 4, 4, occupied\_slot)

set\_statuses (3, 3, 2, 3, occupied\_slot)

set\_statuses (3, 3, 5, 6, occupied\_slot)

set\_statuses (6, 7, 3, 3, occupied\_slot)

set\_statuses (6, 7, 4, 4, occupied\_slot)

set\_statuses (6, 7, 5, 5, occupied\_slot)

set\_statuses (3, 4, 4, 4, occupied\_slot)

set\_status (2, 3, occupied\_slot)

set\_status (2, 5, occupied\_slot)

set\_status (5, 4, occupied\_slot)

ensure

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

end

make\_diamond

-- Initialize a Diamond board.

do

create imp.make\_filled (unavailable\_slot, 7, 7)

set\_status (1, 3, unoccupied\_slot)

set\_status (1, 5, unoccupied\_slot)

set\_status (3, 1, unoccupied\_slot)

set\_status (3, 7, unoccupied\_slot)

set\_status (4, 4, unoccupied\_slot)

set\_status (5, 1, unoccupied\_slot)

set\_status (5, 7, unoccupied\_slot)

set\_status (7, 3, unoccupied\_slot)

set\_status (7, 5, unoccupied\_slot)

set\_statuses (1, 2, 4, 4, occupied\_slot)

set\_statuses (3, 3, 2, 3, occupied\_slot)

set\_statuses (3, 3, 4, 5, occupied\_slot)

set\_statuses (3, 4, 6, 6, occupied\_slot)

set\_statuses (4, 5, 2, 2, occupied\_slot)

set\_statuses (4, 5, 3, 3, occupied\_slot)

set\_statuses (6, 7, 4, 4, occupied\_slot)

set\_statuses (4, 5, 5, 5, occupied\_slot)

set\_status (2, 3, occupied\_slot)

set\_status (2, 5, occupied\_slot)

set\_status (4, 1, occupied\_slot)

set\_status (4, 7, occupied\_slot)

set\_status (5, 4, occupied\_slot)

set\_status (5, 6, occupied\_slot)

set\_status (6, 3, occupied\_slot)

set\_status (6, 5, occupied\_slot)

ensure

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

end

make\_skull

-- Initialize a Skull board.

do

create imp.make\_filled (unavailable\_slot, 7, 7)

set\_statuses (4, 5, 1, 1, unoccupied\_slot)

set\_statuses (4, 5, 7, 7, unoccupied\_slot)

set\_status (4, 3, unoccupied\_slot)

set\_status (4, 5, unoccupied\_slot)

set\_status (3, 1, unoccupied\_slot)

set\_status (3, 7, unoccupied\_slot)

set\_statuses (1, 2, 3, 3, occupied\_slot)

set\_statuses (1, 2, 4, 4, occupied\_slot)

set\_statuses (1, 2, 5, 5, occupied\_slot)

set\_statuses (3, 4, 2, 2, occupied\_slot)

set\_statuses (3, 4, 4, 4, occupied\_slot)

set\_statuses (3, 4, 6, 6, occupied\_slot)

set\_statuses (5, 6, 3, 3, occupied\_slot)

set\_statuses (5, 6, 4, 4, occupied\_slot)

set\_statuses (5, 6, 5, 5, occupied\_slot)

set\_status (3, 3, occupied\_slot)

set\_status (3, 5, occupied\_slot)

set\_status (5, 2, occupied\_slot)

set\_status (5, 6, occupied\_slot)

set\_status (7, 3, occupied\_slot)

set\_status (7, 4, occupied\_slot)

set\_status (7, 5, occupied\_slot)

ensure

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

end

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)

do

imp.item (r, c) := status

ensure

slot\_set: imp.item (r, c) ~ status

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

end

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

do

across

r1 |..| r2 as row

loop

across

c1 |..| c2 as column

loop

if (row.item >= r1 and row.item <= r2) and (column.item >= c1 and column.item <= c2) then

imp.put (status, row.item, column.item)

end

end

end

ensure

slots\_in\_range\_set: across

r1 |..| r1 as row

all

across

c1 |..| c1 as column

all

(row.item >= r1 and row.item <= r2 and column.item >= c2 and column.item <= c2 implies status\_of (row.item, column.item) ~ status)

end

end

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

end

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: other.number\_of\_rows = Current.number\_of\_rows

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

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

local

i: INTEGER\_32

j: INTEGER\_32

do

from

i := 1

until

i > 7

loop

from

j := 1

until

j > 7

loop

if i >= r1 and i <= r2 then

if j >= c1 and j <= c2 then

j := c2 + 1

end

end

Result := Current.status\_of (i, j) ~ (other.status\_of (i, j))

j := j + 1

end

i := i + 1

end

ensure

correct\_result: Result ~ across

1 |..| 7 as m

all

across

1 |..| 7 as n

all

(m.item < r1 and m.item > r2) or (n.item < c1 and n.item > c2) implies other.status\_of (m.item, n.item).is\_equal (Current.status\_of (m.item, n.item))

end

end

end

unavailable\_slot: UNAVAILABLE\_SLOT

-- A slot not available for movement.

do

Result := ssa.Unavailable\_slot

ensure

Result = ssa.Unavailable\_slot

end

occupied\_slot: OCCUPIED\_SLOT

-- A slot available for moment but currently occupied.

do

Result := ssa.Occupied\_slot

ensure

Result = ssa.Occupied\_slot

end

unoccupied\_slot: UNOCCUPIED\_SLOT

-- A slot available for moment and currently unoccupied.

do

Result := ssa.Unoccupied\_slot

ensure

Result = ssa.Unoccupied\_slot

end

feature -- Queries

number\_of\_rows: INTEGER\_32

-- Number of rows in the board of game.

do

Result := imp.height

ensure

correct\_result: Result = imp.height

end

number\_of\_columns: INTEGER\_32

-- Number of columns in the board of game.

do

Result := imp.width

ensure

correct\_result: Result = (imp.width)

end

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

-- Is 'r' a valid row number?

do

Result := r > 0 and r <= number\_of\_rows

ensure

correct\_result: Result = (r > 0 and r <= number\_of\_rows)

end

is\_valid\_column (c: INTEGER\_32): BOOLEAN

-- Is 'x' a valid column number?

do

Result := c > 0 and c <= number\_of\_columns

ensure

correct\_result: Result = (c > 0 and c <= number\_of\_columns)

end

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)

do

Result := imp.item (r, c)

ensure

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

end

number\_of\_occupied\_slots: INTEGER\_32

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

do

Result := 0

across

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

loop

across

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

loop

if imp.item (i.item, j.item) ~ (ssa.Occupied\_slot) then

Result := Result + 1

end

end

end

end

feature -- Equality

is\_equal (other: like Current): BOOLEAN

-- Is current board equal to 'other'?

do

Result := (Current.out ~ other.out)

ensure then

correct\_result: Result = (Current.out ~ other.out)

end

feature -- Output

out: STRING\_8

-- String representation of current board.

local

i, j: INTEGER\_32

do

create Result.make\_empty

from

i := 1

until

i > 7

loop

from

j := 1

until

j > 7

loop

if imp.item (i, j) ~ occupied\_slot then

Result.append ("O")

elseif imp.item (i, j) ~ unavailable\_slot then

Result.append ("\*")

else

Result.append (".")

end

if i < number\_of\_rows and j = number\_of\_columns then

Result.append ("%N")

end

j := j + 1

end

i := i + 1

end

end

feature {NONE} -- Implementation

ssa: SLOT\_STATUS\_ACCESS

bta: BOARD\_TEMPLATES\_ACCESS

-- Note: ARRAY2 takes row (y) and then column (x)

imp: ARRAY2 [SLOT\_STATUS]

end -- class BOARD

**Contract View**

**Game Class**

note

description: "A game of peg solitaire."

author: "Mustafa Alsoufi and Jackie Wang"

date: "$Date$"

revision: "$Revision$"

class

GAME

inherit

ANY

redefine

out

end

create

make\_from\_board,

make\_easy,

make\_cross,

make\_plus,

make\_pyramid,

make\_arrow,

make\_diamond,

make\_skull

feature -- Constructors

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

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

do

board := new\_board

ensure

board\_set: board.out ~ new\_board.out

end

make\_easy

-- Initialize a game with easy board.

do

create board.make\_easy

ensure

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

end

make\_cross

-- Initialize a game with Cross board.

do

create board.make\_cross

ensure

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

end

make\_plus

-- Initialize a game with Plus board.

do

create board.make\_plus

ensure

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

end

make\_pyramid

-- Initialize a game with Pyramid board.

do

create board.make\_pyramid

ensure

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

end

make\_arrow

-- Initialize a game with Arrow board.

do

create board.make\_arrow

ensure

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

end

make\_diamond

-- Initialize a game with Diamond board.

do

create board.make\_diamond

ensure

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

end

make\_skull

-- Initialize a game with Skull board.

do

create board.make\_skull

ensure

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

end

feature -- Commands

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).is\_equal (board.occupied\_slot)

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

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

do

board.set\_status (r, c, board.unoccupied\_slot)

board.set\_status (r, c - 1, board.unoccupied\_slot)

board.set\_status (r, c - 2, board.occupied\_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)

end

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).is\_equal (board.occupied\_slot)

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

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

do

board.set\_status (r, c, board.unoccupied\_slot)

board.set\_status (r, c + 1, board.unoccupied\_slot)

board.set\_status (r, c + 2, board.occupied\_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)

end

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).is\_equal (board.occupied\_slot)

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

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

do

board.set\_status (r, c, board.unoccupied\_slot)

board.set\_status (r - 2, c, board.occupied\_slot)

board.set\_status (r - 1, 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 - 2, r, c, c)

end

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

do

board.set\_status (r, c, board.unoccupied\_slot)

board.set\_status (r + 1, c, board.unoccupied\_slot)

board.set\_status (r + 2, c, board.occupied\_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 + 1, c, c)

end

feature -- Status Queries

is\_over: BOOLEAN

-- Is the current game 'over'?

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

do

Result := not across

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

some

across

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

some

moving\_peg (rows.item, columns.item)

end

end

ensure

correct\_result: Result = not across

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

some

across

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

some

moving\_peg (rows.item, columns.item)

end

end

end

is\_won: BOOLEAN

-- Has the current game been won?

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

do

Result := (board.number\_of\_occupied\_slots = 1) and is\_over

ensure

game\_won\_iff\_one\_occupied\_slot\_left: Result = (board.number\_of\_occupied\_slots = 1)

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

end

feature -- Output

out: STRING\_8

-- String representation of current game.

-- Do not modify this feature!

do

create Result.make\_empty

Result.append ("Game is over: " + boolean\_to\_yes\_no (is\_over) + "%N")

Result.append ("Game is won : " + boolean\_to\_yes\_no (is\_won) + "%N")

Result.append ("Board Status:%N")

Result.append (board.out)

end

feature -- checking if the peg can move left/right/up/down

moving\_peg (r, c: INTEGER\_32): BOOLEAN

local

up, down, left, right: BOOLEAN

do

if (board.is\_valid\_row (r - 2)) then

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

end

if (board.is\_valid\_row (r + 2)) then

down := (board.status\_of (r, c) ~ board.occupied\_slot and board.status\_of (r + 2, c) ~ board.unoccupied\_slot and board.status\_of (r + 2, c) ~ board.occupied\_slot)

end

if (board.is\_valid\_column (c - 2)) then

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

end

if (board.is\_valid\_column (c + 2)) then

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

end

Result := (up or down or left or right)

end

feature -- Auxiliary Routines

boolean\_to\_yes\_no (b: BOOLEAN): STRING\_8

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

do

if b then

Result := "Yes"

else

Result := "No"

end

end

feature -- Board

bta: BOARD\_TEMPLATES\_ACCESS

board: BOARD

end -- class GAME

**A close up of text on a white background

Description generated with high confidenceArchitectural Diagram**