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 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

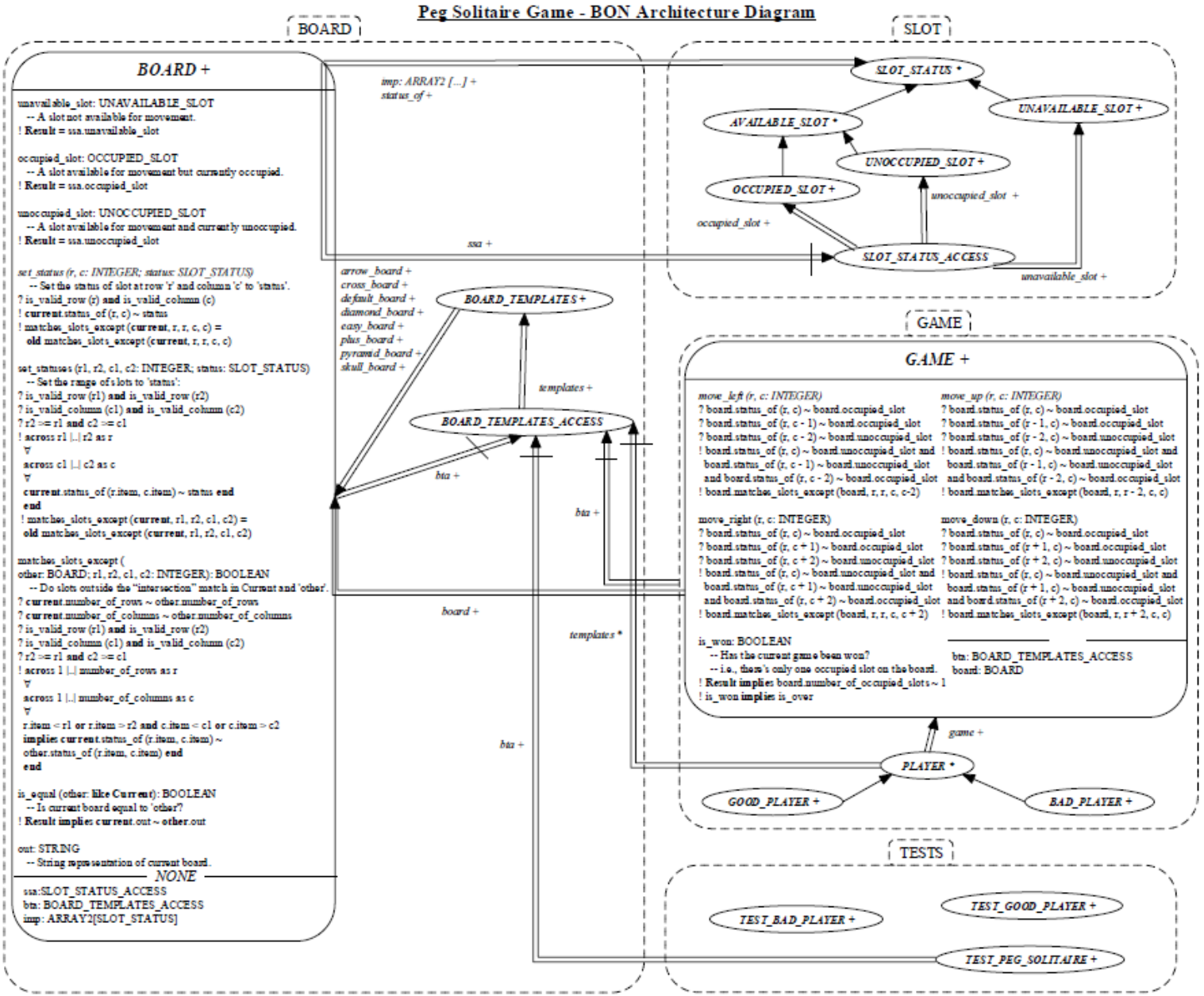
```
note
    description: "A board for the peg solitaire game."
    author: "Camillo John (CJ) D'Alimonte & Dinesh Kalia"
    date: "July 10th 2015"
    revision: "$Revision$"

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 (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: r2 >= r1
            valid_column_range: c2 >= c1
        ensure
            slots_in_range_set: across
                    r1 |..| r2 as r
                all
                    across
                        c1 |..| c2 as c
                    all
                        Current.status_of (r.item, c.item) ~ status
                    end
                end
            slots_not_in_range_unchanged: matches_slots_except (Current, r1, r2, c1, c2) =
                                          old matches_slots_except (Current, 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: Current.number_of_rows ~ other.number_of_rows
            consistent_column_numbers: Current.number_of_columns ~ other.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: r2 >= r1
            valid_column_range: c2 >= c1
        ensure
            correct_result: across
                    1 |..| number_of_rows as r
                all
                    across
                        1 |..| number_of_columns as c
                    all
                        r.item < r1 or r.item > r2 and c.item < c1 or c.item > c2 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
```


Peg Solitaire Game - BON Architecture Diagram
BOARD
SLOT
BOARD +
unavailable_slot: UNAVAILABLE_SLOT
-- A slot not available for movement.
! Result = ssa.unavailable_slot
occupied_slot: OCCUPIED_SLOT
-- A slot available for movement but currently occupied.
! Result = ssa.occupied_slot
unoccupied_slot: UNOCCUPIED_SLOT
-- A slot available for movement and currently unoccupied.
! Result = ssa.unoccupied_slot
set_status (r, c: INTEGER; status: SLOT_STATUS)
-- Set the status of slot at row 'r' and column 'c' to 'status'.
? is_valid_row (r) and is_valid_column (c)
! current.status_of (r, c) ~ status
! matches_slots_except (current, r, r, c, c) =
old matches_slots_except (current, r, r, c, c)
set_statuses (r1, r2, c1, c2: INTEGER; status: SLOT_STATUS)
-- Set the range of slots to 'status':
? is_valid_row (r1) and is_valid_row (r2)
? is_valid_column (c1) and is_valid_column (c2)
? r2 >= r1 and c2 >= c1
! across r1 |..| r2 as r
∀
across c1 |..| c2 as c
∀
current.status_of (r.item, c.item) ~ status end
end
! matches_slots_except (current, r1, r2, c1, c2) =
old matches_slots_except (current, r1, r2, c1, c2)
matches_slots_except (
other: BOARD; r1, r2, c1, c2: INTEGER): BOOLEAN
-- Do slots outside the "intersection" match in Current and 'other'.
? current.number_of_rows ~ other.number_of_rows
? current.number_of_columns ~ other.number_of_columns
? is_valid_row (r1) and is_valid_row (r2)
? is_valid_column (c1) and is_valid_column (c2)
? r2 >= r1 and c2 >= c1
! across 1 |..| number_of_rows as r
∀
across 1 |..| number_of_columns as c
∀
r.item < r1 or r.item > r2 and c.item < c1 or c.item > c2
implies current.status_of (r.item, c.item) ~
other.status_of (r.item, c.item) end
end
is_equal (other: like Current): BOOLEAN
-- Is current board equal to 'other'?
! Result implies current.out ~ other.out
out: STRING
-- String representation of current board.
NONE
ssa:SLOT_STATUS_ACCESS
bta: BOARD_TEMPLATES_ACCESS
imp: ARRAY2[SLOT_STATUS]
imp: ARRAY2 [...] +
status_of +
SLOT_STATUS *
AVAILABLE_SLOT *
UNAVAILABLE_SLOT +
UNOCCUPIED_SLOT +
OCCUPIED_SLOT +
unoccupied_slot +
occupied_slot +
ssa +
SLOT_STATUS_ACCESS
unavailable_slot +
arrow_board +
cross_board +
default_board +
diamond_board +
easy_board +
plus_board +
pyramid_board +
skull_board +
BOARD_TEMPLATES +
templates +
BOARD_TEMPLATES_ACCESS
bta +
bta +
board +
templates *
bta +
GAME
GAME +
move_left (r, c: INTEGER)
? board.status_of (r, c) ~ board.occupied_slot
? board.status_of (r, c - 1) ~ board.occupied_slot
? board.status_of (r, c - 2) ~ board.unoccupied_slot
! 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
! board.matches_slots_except (board, r, r, c, c-2)
move_up (r, c: INTEGER)
? board.status_of (r, c) ~ board.occupied_slot
? board.status_of (r - 1, c) ~ board.occupied_slot
? board.status_of (r - 2, c) ~ board.unoccupied_slot
! 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
! board.matches_slots_except (board, r, r - 2, c, c)
move_right (r, c: INTEGER)
? board.status_of (r, c) ~ board.occupied_slot
? board.status_of (r, c + 1) ~ board.occupied_slot
? board.status_of (r, c + 2) ~ board.unoccupied_slot
! 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
! board.matches_slots_except (board, r, r, c, c + 2)
move_down (r, c: INTEGER)
? board.status_of (r, c) ~ board.occupied_slot
? board.status_of (r + 1, c) ~ board.occupied_slot
? board.status_of (r + 2, c) ~ board.unoccupied_slot
! 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
! board.matches_slots_except (board, r, r + 2, c, c)
is_won: BOOLEAN
-- Has the current game been won?
-- i.e., there's only one occupied slot on the board.
! Result implies board.number_of_occupied_slots ~ 1
! is_won implies is_over
bta: BOARD_TEMPLATES_ACCESS
board: BOARD
game +
PLAYER *
GOOD_PLAYER +
BAD_PLAYER +
TESTS
TEST_BAD_PLAYER +
TEST_GOOD_PLAYER +
TEST_PEG_SOLITAIRE +