EECS 3311 – ASSIGNMENT REPORT

SUMMER 2015

Completed by -

Methusaan Sivapalan (cse23172)

Sadman Sakib Hasan (cse23152)

Submitting Account: cse23152 (Sadman Sakib Hasan)

BOARD: CONTRACT VIEW

```
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

	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) ~ status
			slots_not_in_range_unchanged: 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: r1 <= r2
                        valid_column_range: c1 <= c2
                ensure
                        slots_in_range_set: across
                                        r1 |..| r2 as cursor1
                                all
                                        across
                                                c1 |..| c2 as cursor2
                                        all
                                                imp.item (cursor1.item, cursor2.item) ~ status
                                        end
                                end
                        slots_not_in_range_unchanged: 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: r1 <= r2
                        valid_column_range: c1 <= c2
                ensure
                        correct_result: Result ~
                         across
                                1 |..| number_of_rows as cursor1
                                all
                                        across
                                                1 |..| number_of_columns as cursor2
                                        all
```

```
                                        (cursor1.item < r1 and cursor1.item > r2) or (cursor2.item < c1
                                    and cursor2.item > c2) implies
                                    other.status_of (cursor1.item, cursor2.item).is_equal
                                                                    (status_of (cursor1.item, cursor2.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_output: Result = (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 = (c >= 1 and c <= number_of_columns)



	is_valid_row (r: INTEGER_32): BOOLEAN
			-- Is 'r' a valid row number?
		ensure
			correct_result: Result = (r >= 1 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
```

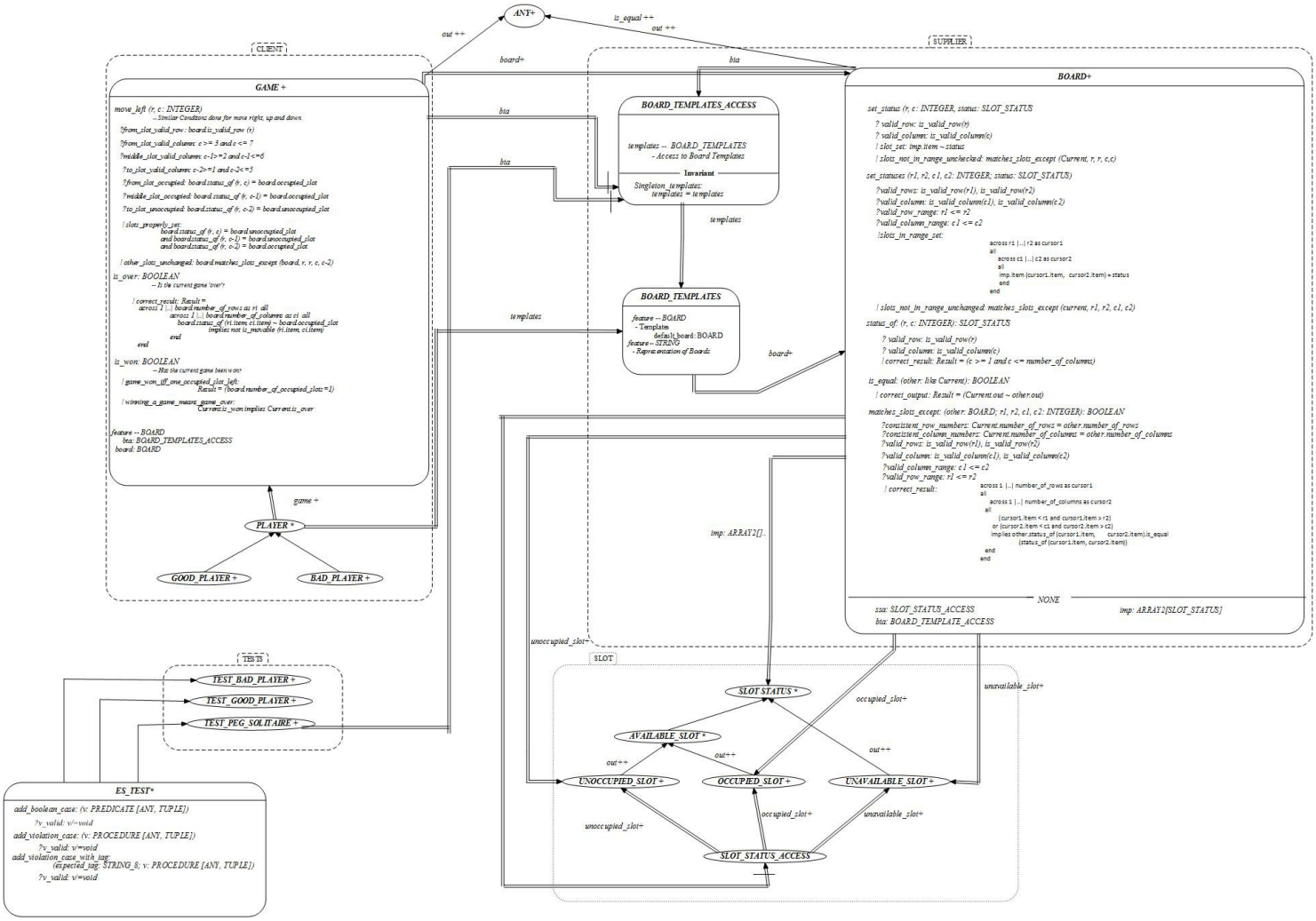
GAME: CONTRACT VIEW

```
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'.

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: r >= 1 and r <= 5
			middle_slot_valid_row: r + 1 >= 2 and r + 1 <= 6
			to_slot_valid_row: r + 2 >= 3 and r + 2 <= 7
			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: c >= 3 and c <= 7
            middle_slot_valid_column: c - 1 >= 2 and c - 1 <= 6
            to_slot_valid_column: c - 2 >= 1 and c - 2 <= 5
            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: c >= 1 and c <= 5
            middle_slot_valid_column: c + 1 >= 2 and c + 1 <= 6
            to_slot_valid_column: c + 2 >= 3 and c + 2 <= 7
            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: r >= 3 and r <= 7
            middle_slot_valid_row: r - 1 >= 2 and r - 1 <= 6
            to_slot_valid_row: r - 2 >= 1 and r - 2 <= 5
            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.out ~ new_board.out

    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 -- My own auxilary features

        peg_moveable (r, c: INTEGER_32): BOOLEAN

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 = not across
                                    1 |..| board.number_of_rows as r
                                some
                                    across
                                        1 |..| board.number_of_columns as c
                                    some
                                        peg_moveable (r.item, c.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 = (board.number_of_occupied_slots = 1)
                winning_a_game_means_game_over: Current.is_won implies Current.is_over

end -- class GAME
```


ANY+
is_equal ++
out ++
out ++
CLIENT
SUPPLIER
board+
bta
GAME +
move_left (r, c: INTEGER)
-- Similar Conditions done for move right, up and down
?from_slot_valid_row: board.is_valid_row (r)
?from_slot_valid_column: c >= 5 and c <= 7
?middle_slot_valid_column: c-1>=2 and c-1 <=6
?to_slot_valid_column: c-2>=1 and c-2<=5
?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
! 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)
is_over: BOOLEAN
-- Is the current game over?
! 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
! game_won_iff_one_occupied_slot_left:
Result = (board.number_of_occupied_slots=1)
! winning_a_game_means_game_over:
Current.is_won implies Current.is_over
feature -- BOARD
bta: BOARD_TEMPLATES_ACCESS
board: BOARD
BOARD_TEMPLATES_ACCESS
templates -- BOARD_TEMPLATES
- Access to Board Templates
Invariant
Singleton_templates:
templates = templates
templates
BOARD_TEMPLATES
feature -- BOARD
- Templates
default_board: BOARD
feature -- STRING
- Representation of Boards
board+
BOARD+
set_status (r, c: INTEGER; status: SLOT_STATUS
? valid_row: is_valid_row(r)
? valid_column: is_valid_column(c)
! slot_set: imp.item ~ status
! slots_not_in_range_unchecked: matches_slots_except (Current, r, r, c,c)
set_statuses (r1, r2, c1, c2: INTEGER; status: SLOT_STATUS)
?valid_rows: is_valid_row(r1), is_valid_row(r2)
?valid_column: is_valid_column(c1), is_valid_column(c2)
?valid_row_range: r1 <= r2
?valid_column_range: c1 <= c2
!slots_in_range_set:
across r1 |..| r2 as cursor1
all
across c1 |..| c2 as cursor2
all
imp.item (cursor1.item, cursor2.item) = status
end
end
! slots_not_in_range_unchanged: matches_slots_except (current, r1, r2, c1, c2)
status_of: (r, c: INTEGER): SLOT_STATUS
? valid_row: is_valid_row(r)
? valid_column: is_valid_column(c)
! correct_result: Result = (c >= 1 and c <= number_of_columns)
is_equal: (other: like Current): BOOLEAN
! correct_output: Result = (Current.out ~ other.out)
matches_slots_except: (other: BOARD; r1, r2, c1, c2: INTEGER): BOOLEAN
?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), is_valid_row(r2)
?valid_column: is_valid_column(c1), is_valid_column(c2)
?valid_column_range: c1 <= c2
?valid_row_range: r1 <= r2
! correct_result:
across 1 |..| number_of_rows as cursor1
all
across 1 |..| number_of_columns as cursor2
all
(cursor1.item < r1 and cursor1.item > r2)
or (cursor2.item < c1 and cursor2.item > c2)
implies other.status_of (cursor1.item, cursor2.item).is_equal
(status_of (cursor1.item, cursor2.item))
end
end
NONE
sza: SLOT_STATUS_ACCESS
bta: BOARD_TEMPLATE_ACCESS
imp: ARRAY2[SLOT_STATUS]
imp: ARRAY2[]..
unoccupied_slot+
occupied_slot+
unavailable_slot+
SLOT
SLOT_STATUS *
AVAILABLE_SLOT *
UNOCCUPIED_SLOT +
OCCUPIED_SLOT +
UNAVAILABLE_SLOT +
SLOT_STATUS_ACCESS
out++
out++
out++
unoccupied_slot+
occupied_slot+
unavailable_slot+
game +
PLAYER *
GOOD_PLAYER +
BAD_PLAYER +
TESTS
TEST_BAD_PLAYER +
TEST_GOOD_PLAYER +
TEST_PEG_SOLITAIRE +
ES_TEST*
add_boolean_case: (v: PREDICATE [ANY, TUPLE])
?v_valid: v/=void
add_violation_case: (v: PROCEDURE [ANY, TUPLE])
?v_valid: v/=void
add_violation_case_with_tag:
(tag:expected_tag: STRING_8; v: PROCEDURE [ANY, TUPLE])
?v_valid: v/=void