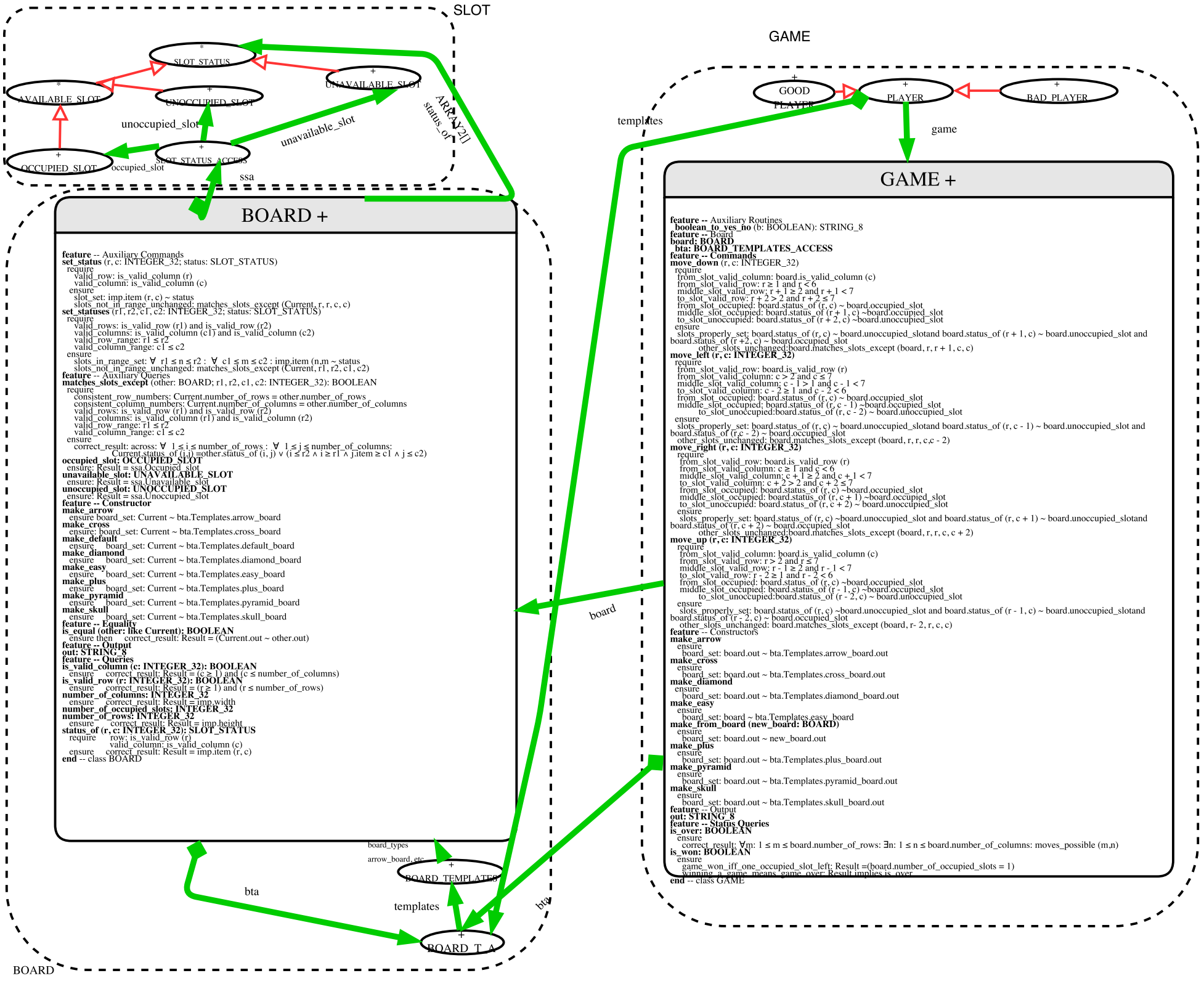
SLOT
SLOT_STATUS
AVAILABLE_SLOT
UNOCCUPIED_SLOT
UNAVAILABLE_SLOT
unoccupied_slot
unavailable_slot
ARRAY2[1]
status_of
OCCUPIED_SLOT
occupied_slot
SLOT_STATUS_ACCESS
ssa
BOARD +
feature -- Auxiliary Commands
set_status (r, c: INTEGER_32; status: SLOT_STATUS)
require
valid_row: is_valid_column (r)
valid_column: 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)
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: ∀ r1 ≤ n ≤ r2 : ∀ c1 ≤ m ≤ c2 : imp.item (n,m ~ status
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
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 (r1) and is_valid_column (r2)
valid_row_range: r1 ≤ r2
valid_column_range: c1 ≤ c2
ensure
correct_result: across ∀ 1 ≤ i ≤ number_of_rows : ∀ 1 ≤ j ≤ number_of_columns:
Current.status_of (i,j) =other.status_of (i, j) ∨ (i ≤ r2 ∧ i ≥ r1 ∧ j.item ≥ c1 ∧ j ≤ c2)
occupied_slot: OCCUPIED_SLOT
ensure: Result = ssa.Occupied_slot
unavailable_slot: UNAVAILABLE_SLOT
ensure: Result = ssa.Unavailable_slot
unoccupied_slot: UNOCCUPIED_SLOT
ensure: Result = ssa.Unoccupied_slot
feature -- Constructor
make_arrow
ensure board_set: Current ~ bta.Templates.arrow_board
make_cross
ensure: board_set: Current ~ bta.Templates.cross_board
make_default
ensure board_set: Current ~ bta.Templates.default_board
make_diamond
ensure board_set: Current ~ bta.Templates.diamond_board
make_easy
ensure board_set: Current ~ bta.Templates.easy_board
make_plus
ensure board_set: Current ~ bta.Templates.plus_board
make_pyramid
ensure board_set: Current ~ bta.Templates.pyramid_board
make_skull
ensure board_set: Current ~ bta.Templates.skull_board
feature -- Equality
is_equal (other: like Current): BOOLEAN
ensure then correct_result: Result = (Current.out ~ other.out)
feature -- Output
out: STRING_8
feature -- Queries
is_valid_column (c: INTEGER_32): BOOLEAN
ensure correct_result: Result = (c ≥ 1) and (c ≤ number_of_columns)
is_valid_row (r: INTEGER_32): BOOLEAN
ensure correct_result: Result = (r ≥ 1) and (r ≤ number_of_rows)
number_of_columns: INTEGER_32
ensure correct_result: Result = imp.width
number_of_occupied_slots: INTEGER_32
number_of_rows: INTEGER_32
ensure correct_result: Result = imp.height
status_of (r, c: INTEGER_32): SLOT_STATUS
require row: is_valid_row (r)
valid_column: is_valid_column (c)
ensure correct_result: Result = imp.item (r, c)
end -- class BOARD
BOARD
board_types
arrow_board, etc.
BOARD_TEMPLATES
bta
templates
BOARD_T_A
bta
board
GAME
GOOD PLAYER
PLAYER
BAD_PLAYER
templates
game
GAME +
feature -- Auxiliary Routines
boolean_to_yes_no (b: BOOLEAN): STRING_8
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 < 6
middle_slot_valid_row: r + 1 ≥ 2 and r + 1 < 7
to_slot_valid_row: r + 2 > 2 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_slotand 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)
move_left (r, c: INTEGER_32)
require
from_slot_valid_row: board.is_valid_row (r)
from_slot_valid_column: c > 2 and c ≤ 7
middle_slot_valid_column: c - 1 > 1 and c - 1 < 7
to_slot_valid_column: c - 2 ≥ 1 and c - 2 < 6
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_slotand 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 < 6
middle_slot_valid_column: c + 1 ≥ 2 and c + 1 < 7
to_slot_valid_column: c + 2 > 2 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_slotand 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 > 2 and r ≤ 7
middle_slot_valid_row: r - 1 ≥ 2 and r - 1 < 7
to_slot_valid_row: r - 2 ≥ 1 and r - 2 < 6
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_slotand board.status_of (r - 2, c) ~ board.occupied_slot
other_slots_unchanged: board.matches_slots_except (board, r- 2, r, c, c)
feature -- Constructors
make_arrow
ensure
board_set: board.out ~ bta.Templates.arrow_board.out
make_cross
ensure
board_set: board.out ~ bta.Templates.cross_board.out
make_diamond
ensure
board_set: board.out ~ bta.Templates.diamond_board.out
make_easy
ensure
board_set: board ~ bta.Templates.easy_board
make_from_board (new_board: BOARD)
ensure
board_set: board.out ~ new_board.out
make_plus
ensure
board_set: board.out ~ bta.Templates.plus_board.out
make_pyramid
ensure
board_set: board.out ~ bta.Templates.pyramid_board.out
make_skull
ensure
board_set: board.out ~ bta.Templates.skull_board.out
feature -- Output
out: STRING_8
feature -- Status Queries
is_over: BOOLEAN
ensure
correct_result: ∀m: 1 ≤ m ≤ board.number_of_rows: ∃n: 1 ≤ n ≤ board.number_of_columns: moves_possible (m,n)
is_won: BOOLEAN
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 -- class GAME