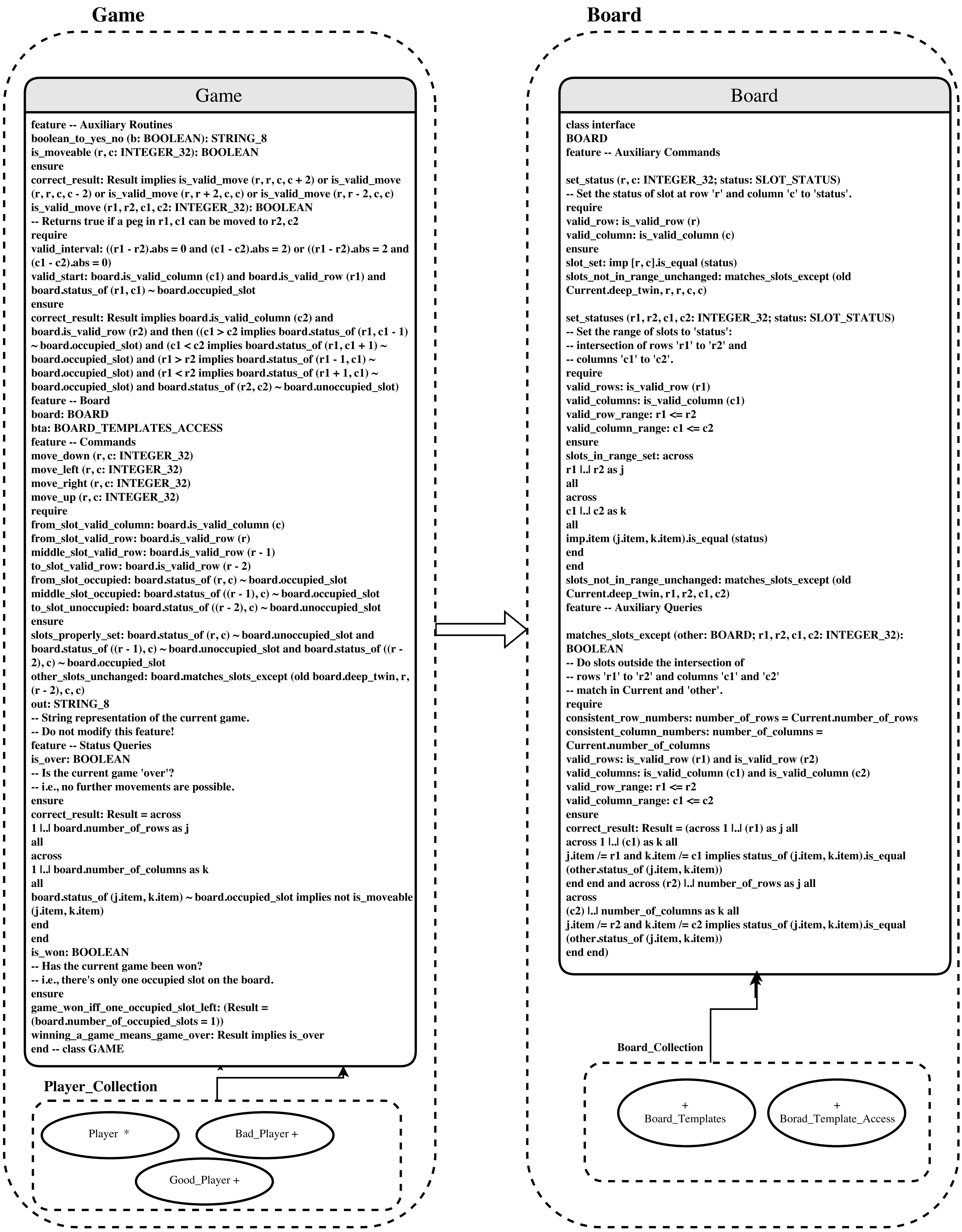

Game

```
feature -- Auxiliary Routines
boolean_to_yes_no (b: BOOLEAN): STRING_8
is_moveable (r, c: INTEGER_32): BOOLEAN
ensure
correct_result: Result implies is_valid_move (r, r, c, c + 2) or is_valid_move
(r, r, c, c - 2) or is_valid_move (r, r + 2, c, c) or is_valid_move (r, r - 2, c, c)
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)
move_left (r, c: INTEGER_32)
move_right (r, c: INTEGER_32)
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 (old board.deep_twin, r,
(r - 2), c, c)
out: STRING_8
-- String representation of the 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 j
all
across
1 |..| board.number_of_columns as k
all
board.status_of (j.item, k.item) ~ board.occupied_slot implies not is_moveable
(j.item, k.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: Result implies is_over
end -- class GAME
```

Board

```
class interface
BOARD
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 [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)
valid_columns: is_valid_column (c1)
valid_row_range: r1 <= r2
valid_column_range: c1 <= c2
ensure
slots_in_range_set: across
r1 |..| r2 as j
all
across
c1 |..| c2 as k
all
imp.item (j.item, k.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_rows = Current.number_of_rows
consistent_column_numbers: 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
ensure
correct_result: Result = (across 1 |..| (r1) as j all
across 1 |..| (c1) as k all
j.item /= r1 and k.item /= c1 implies status_of (j.item, k.item).is_equal
(other.status_of (j.item, k.item))
end end and across (r2) |..| number_of_rows as j all
across
(c2) |..| number_of_columns as k all
j.item /= r2 and k.item /= c2 implies status_of (j.item, k.item).is_equal
(other.status_of (j.item, k.item))
end end)
```