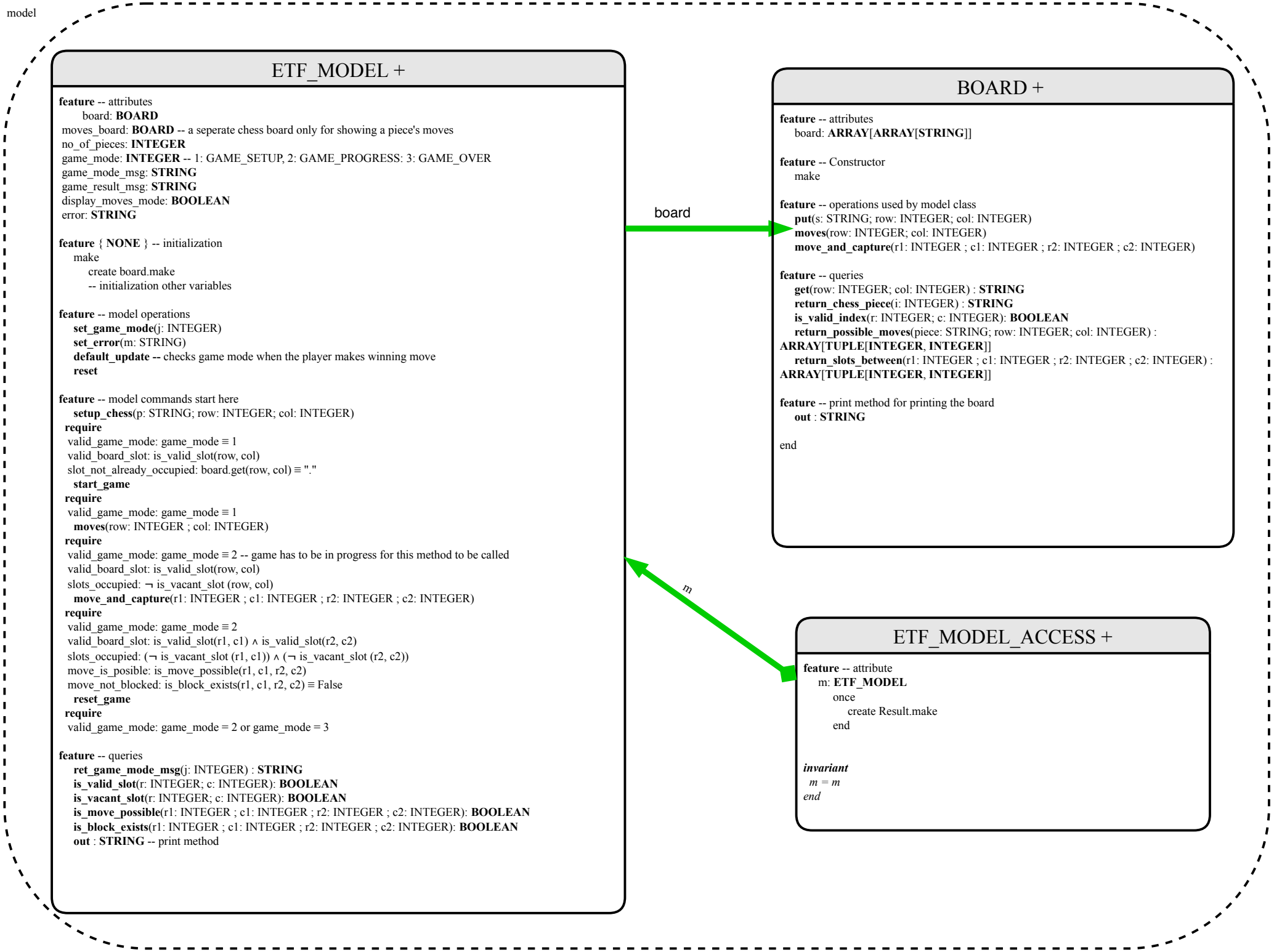
model
ETF_MODEL +
feature -- attributes
board: BOARD
moves_board: BOARD -- a seperate chess board only for showing a piece's moves
no_of_pieces: INTEGER
game_mode: INTEGER -- 1: GAME_SETUP, 2: GAME_PROGRESS: 3: GAME_OVER
game_mode_msg: STRING
game_result_msg: STRING
display_moves_mode: BOOLEAN
error: STRING
feature { NONE } -- initialization
make
create board.make
-- initialization other variables
feature -- model operations
set_game_mode(j: INTEGER)
set_error(m: STRING)
default_update -- checks game mode when the player makes winning move
reset
feature -- model commands start here
setup_chess(p: STRING; row: INTEGER; col: INTEGER)
require
valid_game_mode: game_mode ≡ 1
valid_board_slot: is_valid_slot(row, col)
slot_not_already_occupied: board.get(row, col) ≡ "."
start_game
require
valid_game_mode: game_mode ≡ 1
moves(row: INTEGER ; col: INTEGER)
require
valid_game_mode: game_mode ≡ 2 -- game has to be in progress for this method to be called
valid_board_slot: is_valid_slot(row, col)
slots_occupied: ¬ is_vacant_slot (row, col)
move_and_capture(r1: INTEGER ; c1: INTEGER ; r2: INTEGER ; c2: INTEGER)
require
valid_game_mode: game_mode ≡ 2
valid_board_slot: is_valid_slot(r1, c1) ∧ is_valid_slot(r2, c2)
slots_occupied: (¬ is_vacant_slot (r1, c1)) ∧ (¬ is_vacant_slot (r2, c2))
move_is_posible: is_move_possible(r1, c1, r2, c2)
move_not_blocked: is_block_exists(r1, c1, r2, c2) ≡ False
reset_game
require
valid_game_mode: game_mode = 2 or game_mode = 3
feature -- queries
ret_game_mode_msg(j: INTEGER) : STRING
is_valid_slot(r: INTEGER; c: INTEGER): BOOLEAN
is_vacant_slot(r: INTEGER; c: INTEGER): BOOLEAN
is_move_possible(r1: INTEGER ; c1: INTEGER ; r2: INTEGER ; c2: INTEGER): BOOLEAN
is_block_exists(r1: INTEGER ; c1: INTEGER ; r2: INTEGER ; c2: INTEGER): BOOLEAN
out : STRING -- print method
board
BOARD +
feature -- attributes
board: ARRAY[ARRAY[STRING]]
feature -- Constructor
make
feature -- operations used by model class
put(s: STRING; row: INTEGER; col: INTEGER)
moves(row: INTEGER; col: INTEGER)
move_and_capture(r1: INTEGER ; c1: INTEGER ; r2: INTEGER ; c2: INTEGER)
feature -- queries
get(row: INTEGER; col: INTEGER) : STRING
return_chess_piece(i: INTEGER) : STRING
is_valid_index(r: INTEGER; c: INTEGER): BOOLEAN
return_possible_moves(piece: STRING; row: INTEGER; col: INTEGER) : ARRAY[TUPLE[INTEGER, INTEGER]]
return_slots_between(r1: INTEGER ; c1: INTEGER ; r2: INTEGER ; c2: INTEGER) : ARRAY[TUPLE[INTEGER, INTEGER]]
feature -- print method for printing the board
out : STRING
end
m
ETF_MODEL_ACCESS +
feature -- attribute
m: ETF_MODEL
once
create Result.make
end
invariant
m = m
end