Exercise 2: I require a design

Hand-out: 12 April 2005
Due: 19 April 2005

Master Solution

2. Modularity

In the given example, the query `is_valid_destination' had knowledge about all kinds of figures. Using an integer to differentiate between different figures does not make a good abstraction. Adding new features to the class FIGURE will likely require if-statements similar to the one present in query `is_valid_destination'. The below solution removes the if and replaces it with a type hierarchy. Class FIGURE is now deferred, the different figure types are each represented by a descendant of class FIGURE as indicated by class KING.

defered class 
FIGURE

feature -- Access
x: INTEGER is
  -- Position of figure on X axis of chess board
  deferred
end

y: INTEGER is
  -- Position of figure on Y axis of chess board
  deferred
end

feature -- Status
is_valid_position (an_x: INTEGER; a_y: INTEGER): BOOLEAN is
  -- Is (`an_x', `a_y') a valid position on an chess board
  -- whose range is from (`1', `1') to (`8', `8')?
...

is_valid_destination (an_x: INTEGER; a_y: INTEGER): BOOLEAN is
  -- Is (`x', `y') to (`an_x', `a_y') a valid move for the current
  -- figure?
  require
    is_valid_position: is_valid_position (an_x, a_y)
defered
end

feature -- Movement
move_to (an_x: INTEGER; a_y: INTEGER) is
  -- Move current figure from (`x', `y') to (`an_x', `a_y').
  require
    is_valid_position: is_valid_position (an_x, a_y)
    is_valid_destination: is_valid_destination (an_x, a_y)
...

invariant
    valid_current_position: is_valid_position (x, y)
end

defered class
    KING
inherit
    FIGURE

feature -- Access
    is_valid_destination (an_x: INTEGER; a_y: INTEGER): BOOLEAN is
        -- Is (`x', `y') to (`an_x', `a_y') a valid move
        -- for the current figure?
    do
        Result := ...
    end
end
...
end