battleship

December 7, 2012

Contents

1 Board **1**

2 CLI **2**

3 GameEngine **4**

4 House **6**

5 Player **7**

6 Ship **8**

1 Board

```
class Board
 types
  public Houses = set of House;
 values
  public static BOARD_SIZE: int = 10;

  public static MISS : int = 0;
  public static HIT : int = 1;
  public static SHIP_SUNK : int = 2;

 instance variables

  public playable: bool;
  public houses: Houses := {};

  inv card(houses) >= 1;

 operations

  public Board: bool ==> Board
   Board(t) ==
   (
    playable := t;

    for all x in set {1, ..., BOARD_SIZE}  do (
     for all y in set {1, ..., BOARD_SIZE} do (
      houses := houses union {new House(x,y)};
     )
    );
```

```
    for all h1 in set houses do h1.setBoard(self);
   )
  post card(houses) = BOARD_SIZE * BOARD_SIZE;

  -- returns:
  --0 => miss or already hit
  --1 => hit
  --2 => ship sunk
  public hit : seq of int ==> int
   hit(coords) ==
   (
    let h in set houses be st (h.x = coords(House`X) and h.y = coords(House`Y)) in (
     if not h.hasShip or h.hit then return MISS
     elseif h.hasShip then
      (
       h.ship.inc();
       h.hit := true;
       if not h.ship.isDown() then return HIT
       else return SHIP_SUNK;
      )
     else return MISS;
    );
   )
   pre House`checkCoords(coords)
   post RESULT in set {MISS, HIT, SHIP_SUNK};

  public mark : seq of int * int ==> ()
   mark(coords, res) ==
   (
    let h in set houses be st h.x = coords(House`X) and h.y = coords(House`Y) in (
     h.hit := true;
     if res = HIT or res = SHIP_SUNK then
      h.hasShip := true;
    );
   )
   pre House`checkCoords(coords);
end Board
```

Function or operation	Coverage	Calls
Board	100.0%	4
hit	95.7%	199
mark	100.0%	199
Board.vdmpp	98.1%	402

2 CLI

```
class CLI
 types
 values
 instance variables

 public static WATER  : int := 0;
 public static SHIP   : int := 1;
 public static HIT_SHIP  : int := 2;
 public static HIT_WATER : int := 3;
```

```
public static markers : map int to char := {
 WATER  |-> ' ',
 SHIP   |-> 'o',
 HIT_SHIP |-> 'X',
 HIT_WATER |-> '+'
};

operations

public static printInit : Player * Player ==> ()
 printInit (p1, p2) == (

  IO`println("\n*****************************************");
  IO`println(  "*            Battleship War Game           *");
  IO`println(  "*****************************************");

  IO`print(  "\n************* ");
  IO`print(p1.name);
  IO`print(" Vs ");
  IO`print(p2.name);
  IO`println(" **************\n");

 );

public static printEnd : Player ==> ()
 printEnd (winner) == (
  IO`print("Player ");
  IO`print(winner.name);
  IO`println(" is Victorious! You fought with bravery young lad! ");
  IO`println("I shall award you with cookies and milk. Enjoy this feast to your content!\n");
  IO`println(IO`freadval[VDMUtils`String]("src/res/cookies_and_milk.txt").#2);
 );

public static printHeader : Player ==> ()
 printHeader(player) == (
  IO`print("*** ");
  IO`print(player.name);
  IO`println("'s board ***\n");
 );

public static printBoard : Board ==> ()
 printBoard(board) == (
  dcl marker : int := WATER;

  IO`print("   ");
  for x = 1 to Board`BOARD_SIZE do (
   IO`print(x);
   IO`print(" ");
  );
  IO`println(" ");

  for y = 1 to Board`BOARD_SIZE do (
   if y <> 10 then IO`print(" ");
   IO`print(y);
   IO`print("|");
   for x = 1 to Board`BOARD_SIZE do (
    let house in set board.houses be st house.x = x and house.y = y in (

     if house.hasShip then (
      if house.hit then
       marker := HIT_SHIP
      else
       marker := SHIP;
     )
```

```
      elseif house.hit then
       marker := HIT_WATER;

      IO`print(markers(marker));
      IO`print(" ");
      marker := WATER;
     )
    );
    IO`print("|");
    IO`println(y);
   );

   IO`print("   ");
   for x = 1 to Board`BOARD_SIZE do (
    IO`print(x);
    IO`print(" ");
   );
   IO`println(" ");
   IO`println(" ");
  );

 public static printBoard : Player * Board ==> ()
  printBoard(player, board) == (
   printHeader(player);

   printBoard(board);
  );

end CLI
```

Function or operation	Coverage	Calls
printBoard	100.0%	2
printEnd	100.0%	1
printHeader	100.0%	2
printInit	100.0%	1
CLI.vdmpp	98.5%	6

3 GameEngine

```
class GameEngine
 types
 values
  public static PLAYER1 : int = 1;
  public static PLAYER2 : int = 2;

 instance variables
  public p1: Player;
  public p2: Player;
  public board1own : Board;
  public board1play : Board;
  public board2own : Board;
  public board2play : Board;
  public activePlayer: int := PLAYER1;
  public win : bool := false;

  inv activePlayer in set {PLAYER1, PLAYER2};
```

```
operations
 public GameEngine : () ==> GameEngine
  GameEngine () ==
  (
   p1 := new Player("pedro", 1);
   p2 := new Player("ze", 2);

   board1own := p1.boardown;
   board2own := p2.boardown;
   board1play := p1.boardplay;
   board2play := p2.boardplay;
  );

 public let_the_slaughter_begin : () ==> ()
  let_the_slaughter_begin () ==
  (
   dcl victorious : int := 0, loser : int := 0;

   CLI`printInit(p1, p2);

   while not win do
   (
    victorious := turn();

    if victorious = PLAYER1 then loser := PLAYER2
    else loser := PLAYER1;
   );

   CLI`printBoard(getPlayer(victorious), getPlayer(victorious).boardown);
   CLI`printBoard(getPlayer(loser), getPlayer(loser).boardown);

   CLI`printEnd(getPlayer(victorious));
  );

 public turn : () ==> int
  turn() == (

   if activePlayer = PLAYER1 then (
    let coords = p1.play() in
    (
     board1play.mark(coords, board2own.hit(coords));
     win := checkVictory(p2.ships);

     if win then return activePlayer;
     activePlayer := PLAYER2;
    )
   )
   elseif activePlayer = PLAYER2 then (
    let coords2 = p2.play() in
    (
     board2play.mark(coords2, board1own.hit(coords2));
     win := checkVictory(p1.ships);
     if win then return activePlayer;
     activePlayer := PLAYER1;
    )
   );

   return 0;
  );

 public checkVictory: seq of Ship ==> bool
  checkVictory (ships) ==
  (
   return forall s in set elems ships & s.isDown();
```

```
  );

 public getPlayer : int ==> Player
  getPlayer(nr) ==
   let player in set {p1, p2} be st player.playerNumber = nr in (
     return player;
   );


end GameEngine
```

Function or operation	Coverage	Calls
GameEngine	100.0%	1
checkVictory	100.0%	199
getPlayer	100.0%	5
let_the_slaughter_begin	100.0%	1
turn	95.6%	199
GameEngine.vdmpp	98.4%	405

4 House

```
class House
 types
  public coord = seq of int;
 values
  public static X : int = 1;
  public static Y : int = 2;
 instance variables

 public x: int;
 public y: int;
 public hit: bool := false;
 public hasShip: bool := false;
 public ship: Ship;
 public board: Board;

 -- Coordinates restriction
 inv checkCoords([x] ^ [y]);

 operations
  -- Constructor
  public House : int * int ==> House
   House (x1, y1) ==
   (
    x := x1;
    y := y1;
   )
  pre checkCoords([x1] ^ [y1]);

  public setBoard : Board ==> ()
   setBoard(b) ==
   (
    board := b;
   )
  pre is_Board(b);
```

```
 functions
  public static checkCoords : coord -> bool
   checkCoords(coords) ==
    coords(X) >= 1 and coords(X) <= Board`BOARD_SIZE and
    coords(Y) >= 1 and coords(Y) <= Board`BOARD_SIZE;

end House
```

Function or operation	Coverage	Calls
House	100.0%	400
checkCoords	100.0%	2294
setBoard	100.0%	400
House.vdmpp	100.0%	3094

5 Player

```
class Player
 types
  public String = VDMUtils`String;

 values

 instance variables
  public static sizes : seq of int := [2, 2, 2, 2, 3, 3, 3, 4, 4, 5];

  public it : int := 0;
  public name : String;
  public ships : seq of Ship;
  public boardown : Board;
  public boardplay : Board;
  public playerNumber : int;

  inv (len ships) <= len sizes;

 operations

  public Player : String * int ==> Player
  Player(n, number) ==
  (
   dcl c : seq of House`coord, orientation: seq of int,
    ship : Ship;

   name := n;
   playerNumber := number;
   boardown := new Board(false);
   boardplay := new Board(true);
   c := get_coord();
   orientation := get_orientation();
   ships := [];

   for i = 1 to (len sizes) do (
    ship := new Ship(c(i),orientation(i),sizes(i),boardown);
    ships := ships ^ [ship];
   );
  );
```

```
  public get_coord: ()  ==> seq of House`coord
   get_coord() ==
   (
    return IO`freadval[seq of House`coord]("src/res/" ^ name ^ ".coords").#2;
   )
   post forall coord in set elems RESULT &
     House`checkCoords(coord);

  public get_orientation: () ==> seq of int
   get_orientation() ==
   (
    return IO`freadval[seq of int]("src/res/" ^ name ^ ".or").#2;
   )
   post forall orientation in set elems RESULT &
    Ship`checkOrientation(orientation);

  public play : () ==> House`coord
   play () ==
   (
    dcl x : int := it - (it div 10) * 10 +1,
     y : int := (it div 10) +1;

    it := it + 1;

    return [x] ^ [y];
   )
   pre it <= 100
   post House`checkCoords(RESULT);

end Player
```

Function or operation	Coverage	Calls
Player	100.0%	2
get_coord	100.0%	2
get_orientation	100.0%	2
play	100.0%	199
Player.vdmpp	100.0%	205

6 Ship

```
class Ship
 types

 values
  public static ORIENTATION_UP : int = 0;
  public static ORIENTATION_RIGHT : int = 1;
  public static ORIENTATION_DOWN : int = 2;
  public static ORIENTATION_LEFT : int = 3;

  static orientations : map int to (seq of int) = {
   ORIENTATION_UP  |-> [ 0,-1],
   ORIENTATION_RIGHT |-> [+1, 0],
   ORIENTATION_DOWN |-> [ 0,+1],
   ORIENTATION_LEFT |-> [-1, 0]
  };
```

```
instance variables

 public static id : int := 0;
 public coord_init : House`coord;
 public coords : set of House`coord := {};
 public orientation : int := 1;
 public hits : int := 0;
 public size: int := 1;
 public board: Board;
 public my_id: int;

 inv checkOrientation(orientation);
 inv len coord_init = 2;
 inv id >= 0;
 inv card(coords) >= 0 and card(coords) <= size;

operations

public Ship: House`coord * int * int * Board ==> Ship
Ship(c,o,s,b) ==
(
 coord_init := c;
 size := s;
 coords := {c};
 orientation := o;
 board := b;
 my_id := id +1;
 id := my_id;

 for i = 1 to size-1 do (
  coords := coords union {[c(1)+orientations(o)(1)*i,c(2)+orientations(o)(2)*i]};
 );

)
pre forall x in set {c(1),c(1)+orientations(o)(1)*(s-1)}, y in set {c(2),c(2)+orientations(o)(2)*(s-1)} &
  House`checkCoords([x] ^ [y])
post fill_houses();

public fill_houses: ()==> bool
 fill_houses() ==
 (
  for all c in set coords do
  (
   let h in set board.houses be st h.x = c(1) and h.y = c(2) in
   (
    h.hasShip := true;
    h.ship := self;
   )
  );
  return true;
 )
pre forall c in set coords &
 let h in set board.houses be st h.x = c(1) and h.y = c(2) in
 (
  not h.hasShip
 );

public inc: () ==> ()
 inc() == hits := hits +1
pre hits < size
post hits <= size;

public isDown : () ==> bool
 isDown() == return size = hits;
```

```
 functions
  public static checkOrientation : int -> bool
   checkOrientation(orientation) ==
    orientation in set {ORIENTATION_UP, ORIENTATION_RIGHT, ORIENTATION_DOWN, ORIENTATION_LEFT};
end Ship
```

Function or operation	Coverage	Calls
Ship	100.0%	20
checkOrientation	100.0%	83
fill_houses	100.0%	20
inc	100.0%	59
isDown	100.0%	625
Ship.vdmpp	100.0%	807