Kuncheng Feng
CSC 466 Presentation

3/23/2023

Abstract:

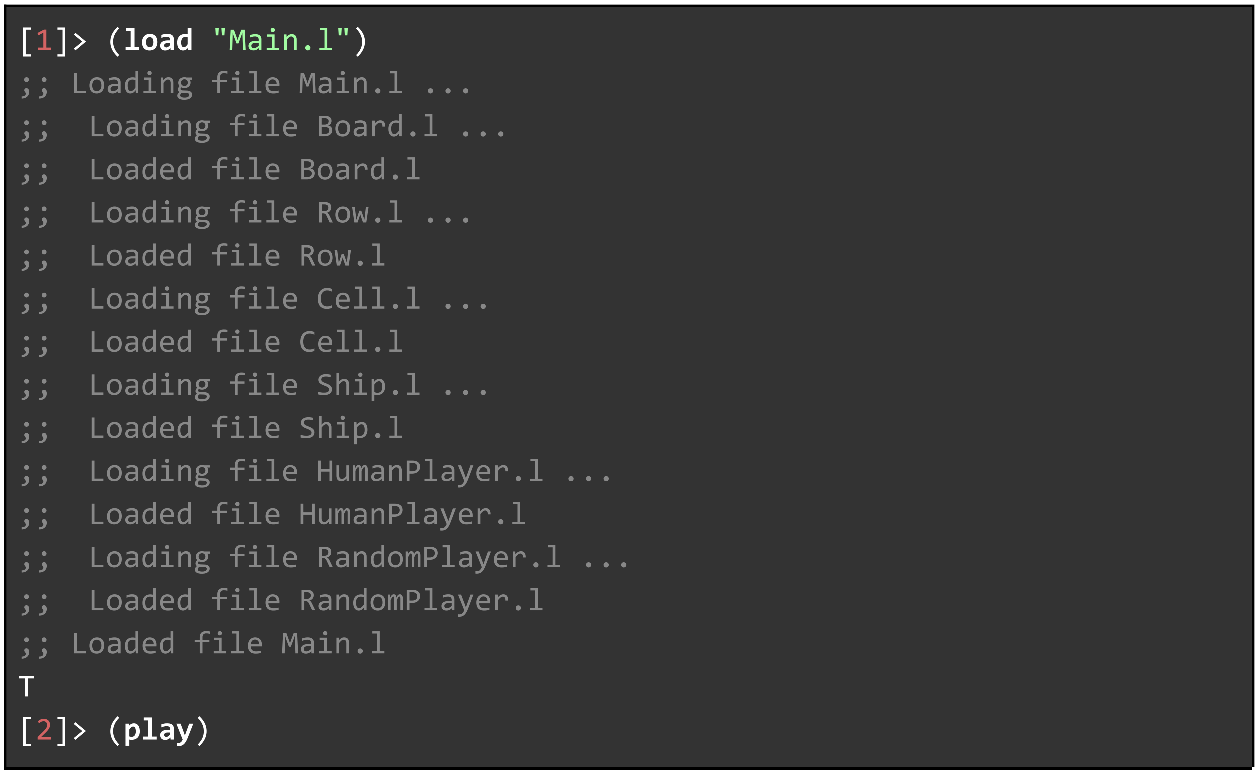
Since the last update, the ship placement system (task 3) has been reimplemented, now it has a validation function with texted feedback, a validation function without texted feedback, and the actual placement function. This way humans won’t see tons of unnecessary texts that will be generated when AI is placing ships.

A “shots” system has been implemented, in the same structure as the ship placement system. It gives feedback on if the player has struck a ship or not.

A randomly playing machine has been implemented, it does everything truly random, given that such moves are valid.

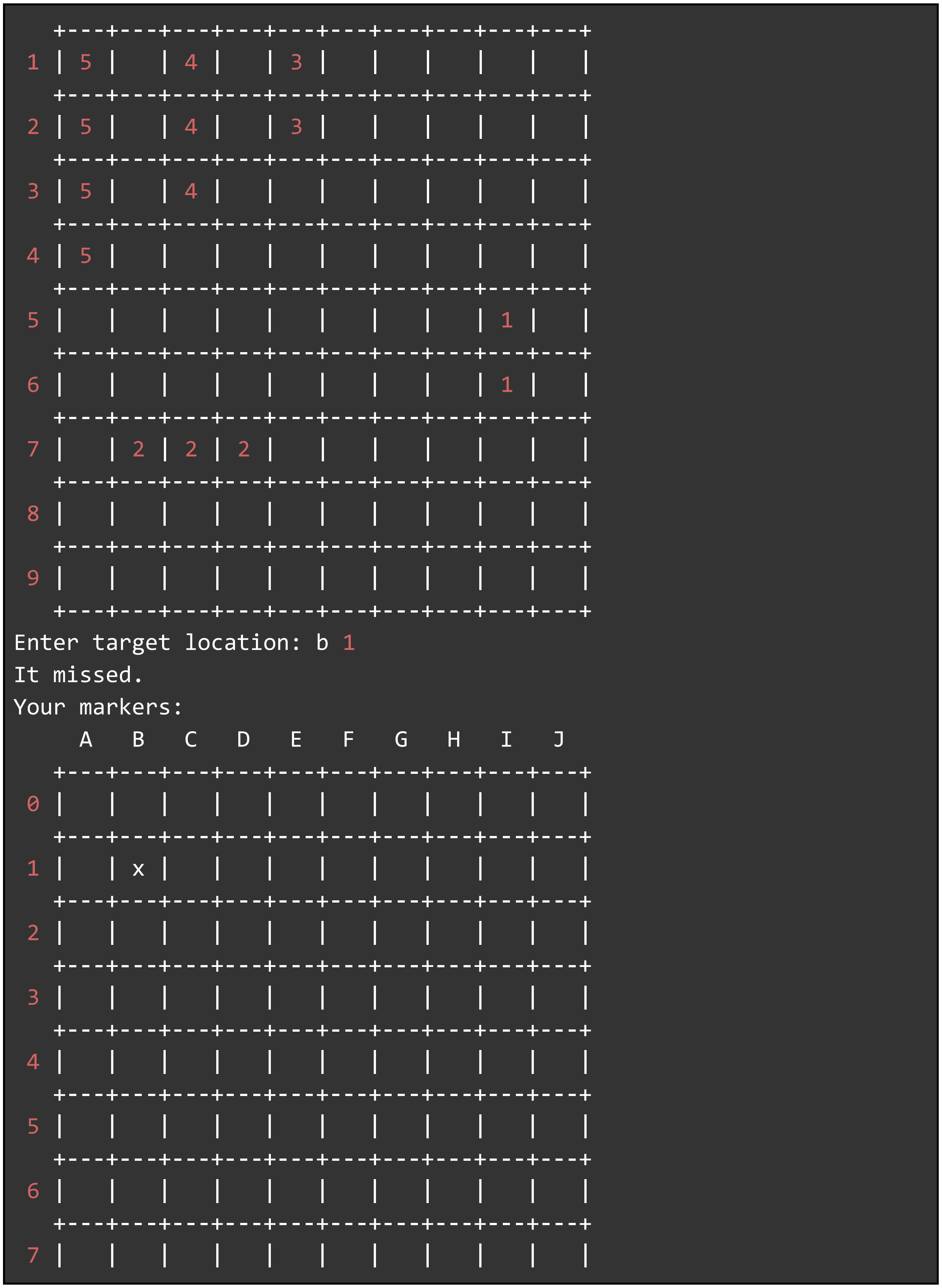
As of the current progress, I’m working on giving better feedback for the shots system, then I will develop a system that recognizes if the game is over.

Demo:

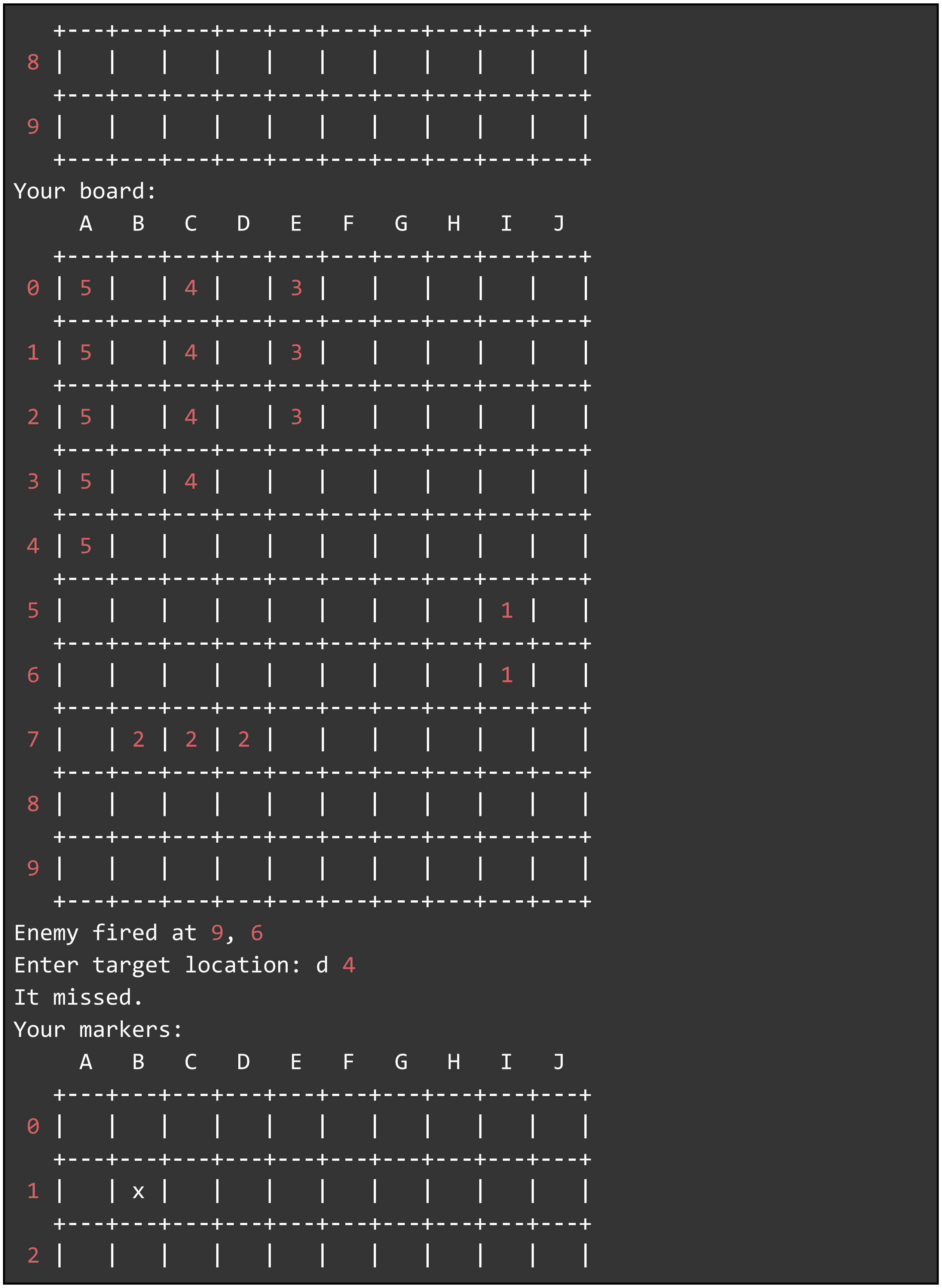


```
[1]> (load "Main.l")
;; Loading file Main.l ...
;;  Loading file Board.l ...
;;  Loaded file Board.l
;;  Loading file Row.l ...
;;  Loaded file Row.l
;;  Loading file Cell.l ...
;;  Loaded file Cell.l
;;  Loading file Ship.l ...
;;  Loaded file Ship.l
;;  Loading file HumanPlayer.l ...
;;  Loaded file HumanPlayer.l
;;  Loading file RandomPlayer.l ...
;;  Loaded file RandomPlayer.l
;; Loaded file Main.l
T
[2]> (play)
```

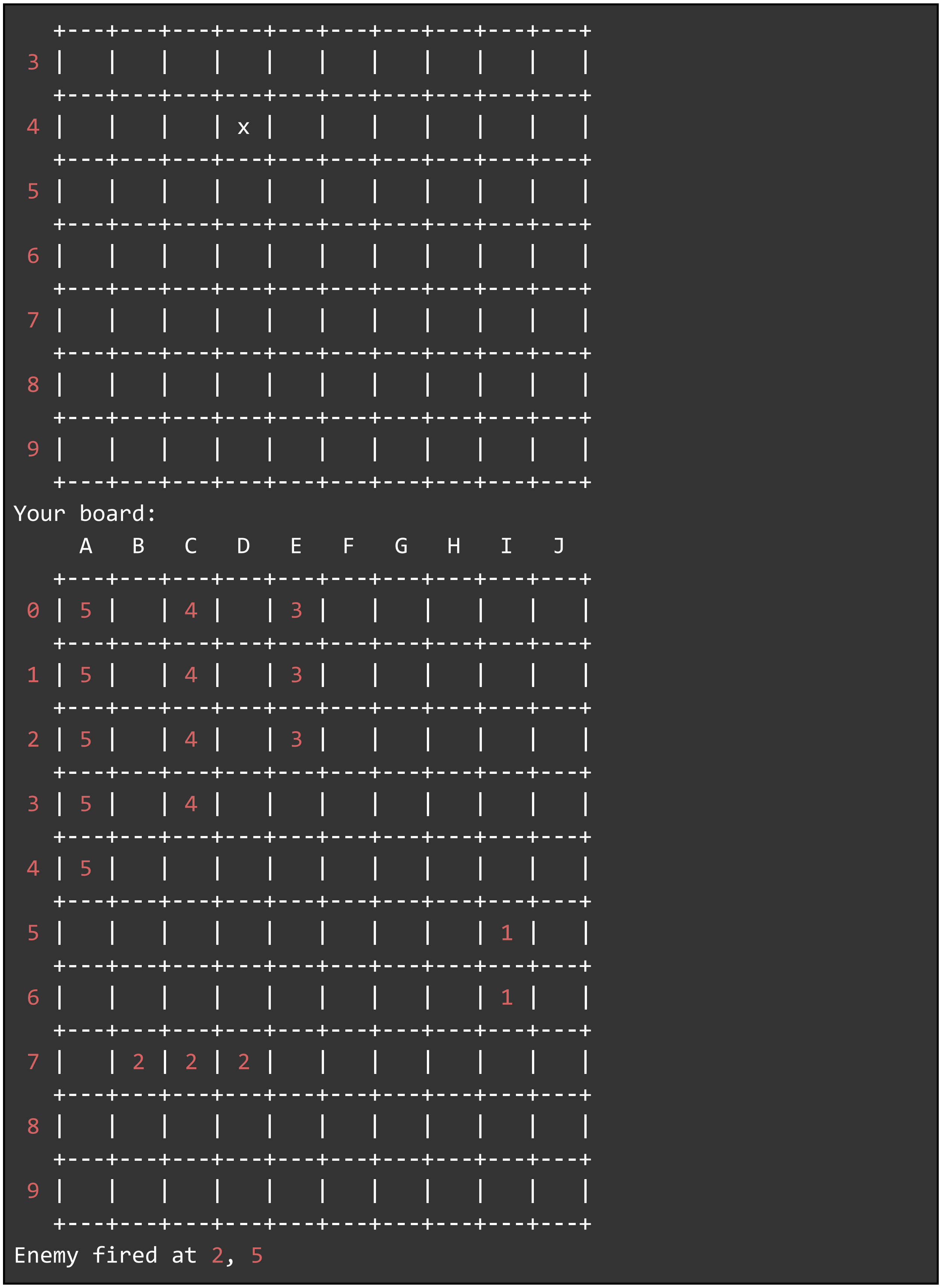
```
Welcome to the Battleship game.
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Now placing: CARRIER, size: 5
Enter position 1: a 0
Enter position 2: a 4
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Now placing: BATTLESHIP, size: 4
Enter position 1: c 0
Enter position 2: c 3
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Now placing: CRUISER, size: 3
Enter position 1: e 0
Enter position 2: e 2
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Now placing: SUBMARINE, size: 3
Enter position 1: b 7
Enter position 2: d 7
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   | 2 | 2 | 2 |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Now placing: DESTROYER, size: 2
Enter position 1: i 5
Enter position 2: i 6
All ships have been placed.


Your markers:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Your board:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   | 4 |   | 3 |   |   |   |   |   |
```



```
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   | 1 |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   | 1 |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   | 2 | 2 | 2 |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Enter target location: b 1
It missed.
Your markers:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 |   | x |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
```



```
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Your board:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   | 1 |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   | 1 |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   | 2 | 2 | 2 |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Enemy fired at 9, 6
Enter target location: d 4
It missed.
Your markers:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 |   | x |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 |   |   |   |   |   |   |   |   |   |   |
```



```
   +---+---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 |   |   |   | x |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Your board:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   | 1 |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   | 1 |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   | 2 | 2 | 2 |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Enemy fired at 2, 5
```

```
Enter target location: g 7
It's a hit!.
Your markers:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 |   | x |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 |   |   |   | x |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   |   |   |   |   |   | o |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Your board:
     A   B   C   D   E   F   G   H   I   J
   +---+---+---+---+---+---+---+---+---+---+
 0 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 1 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 2 | 5 |   | 4 |   | 3 |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 3 | 5 |   | 4 |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 4 | 5 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 5 |   |   |   |   |   |   |   |   | 1 |   |
   +---+---+---+---+---+---+---+---+---+---+
 6 |   |   |   |   |   |   |   |   | 1 |   |
   +---+---+---+---+---+---+---+---+---+---+
 7 |   | 2 | 2 | 2 |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
 9 |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+
Enemy fired at 1, 3
Enter target location: g 8
It's a hit!.
(3 4)
```

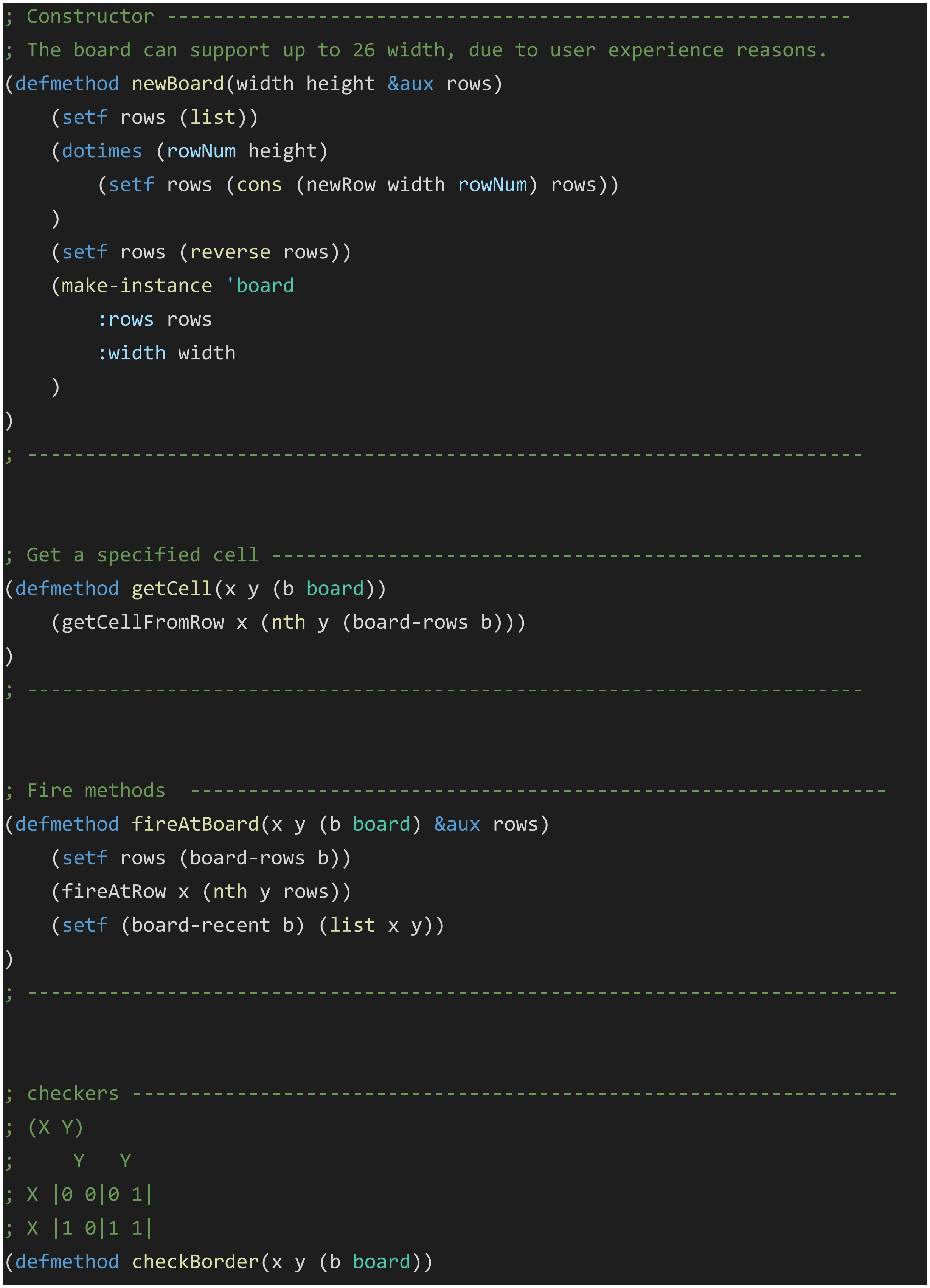
Code:

Main.l

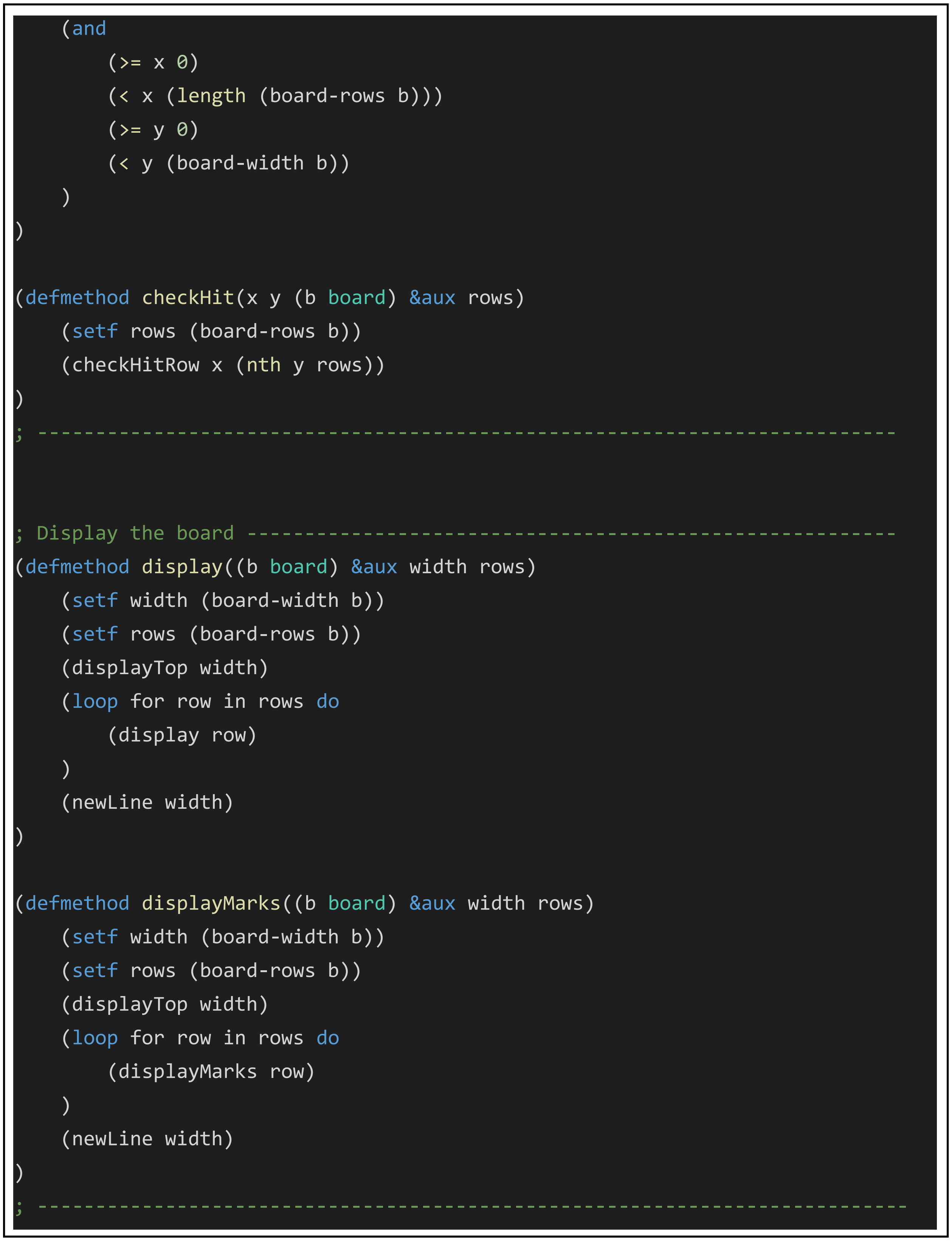
```
; File: Main.l
(load "Board.l")
(load "Row.l")
(load "Cell.l")
(load "Ship.l")
(load "HumanPlayer.l")
(load "RandomPlayer.l")


; Main method to start everything
--------------------------------------------------
(defun play(&aux human ai board1 board2 ships1 ships2)
    (format t "Welcome to the Battleship game.~%")
    (setf board1 (newBoard 10 10))
    (setf board2 (newBoard 10 10))
    (setf ships1 (reverse (generateShips)))
    (setf ships2 (reverse (generateShips)))

    (setf human (newHumanPlayer board1 board2 ships1))
    (setf ai (newRandomPlayer board2 board1 ships2))

    (playerPlaceShips human)
    (playerPlaceShips ai)

    (playerOpenFire human)
    (playerOpenFire ai)

    (playerOpenFire human)
    (playerOpenFire ai)

    (playerOpenFire human)
    (playerOpenFire ai)

    (playerOpenFire human)
    (playerOpenFire ai)
)

(defun generateShips(&aux ships)
    (setf ships (list))
    (loop for ship in *shipTypes* do
        (setf ships (cons (newShip ship) ships))
    )
    ships
)
```

Board.l

```
; File: Board.l
(defclass board()
    (
        (rows :accessor board-rows :initarg :rows)
        (width :accessor board-width :initarg :width)
        (recent :accessor board-recent :initform nil)
    )
)
```

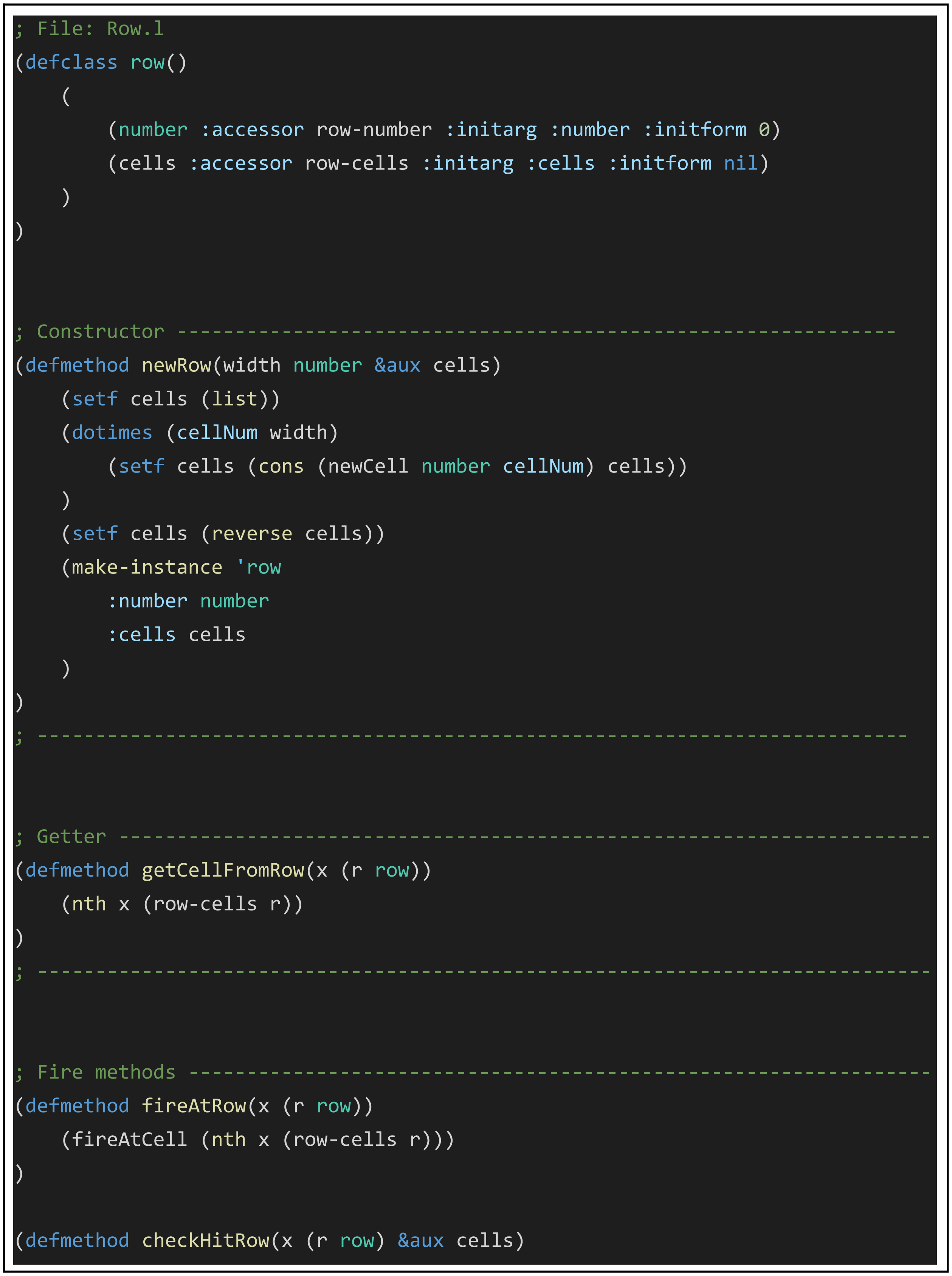


```lisp
; Constructor -----------------------------------------------------------
; The board can support up to 26 width, due to user experience reasons.
(defmethod newBoard(width height &aux rows)
    (setf rows (list))
    (dotimes (rowNum height)
        (setf rows (cons (newRow width rowNum) rows))
    )
    (setf rows (reverse rows))
    (make-instance 'board
        :rows rows
        :width width
    )
)
; ------------------------------------------------------------------------



; Get a specified cell ---------------------------------------------------
(defmethod getCell(x y (b board))
    (getCellFromRow x (nth y (board-rows b)))
)
; ------------------------------------------------------------------------



; Fire methods  ------------------------------------------------------------
(defmethod fireAtBoard(x y (b board) &aux rows)
    (setf rows (board-rows b))
    (fireAtRow x (nth y rows))
    (setf (board-recent b) (list x y))
)
; ---------------------------------------------------------------------------



; checkers ------------------------------------------------------------------
; (X Y)
;     Y   Y
; X |0 0|0 1|
; X |1 0|1 1|
(defmethod checkBorder(x y (b board))
```



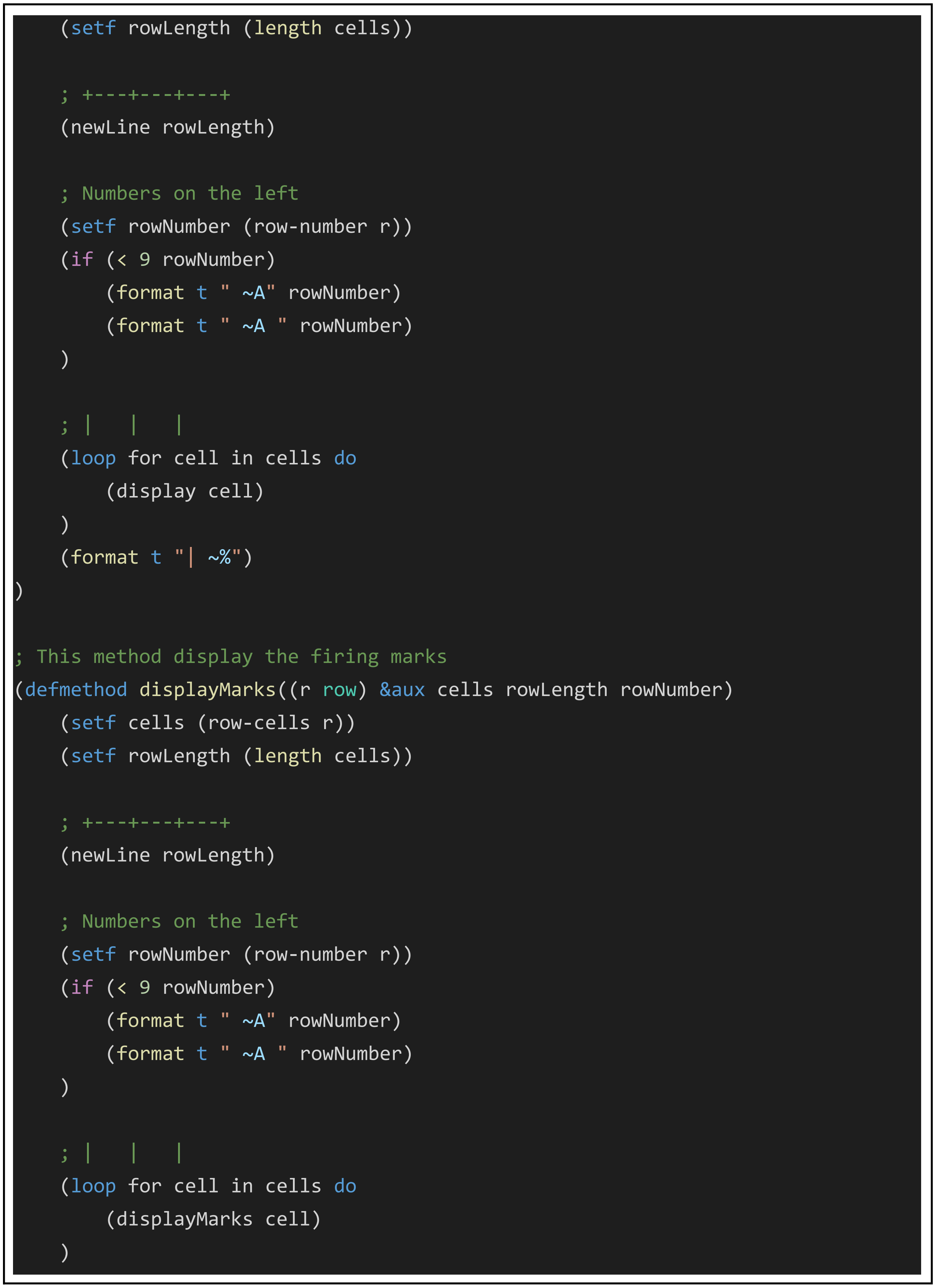
```lisp
    (and
        (>= x 0)
        (< x (length (board-rows b)))
        (>= y 0)
        (< y (board-width b))
    )
)

(defmethod checkHit(x y (b board) &aux rows)
    (setf rows (board-rows b))
    (checkHitRow x (nth y rows))
)
; --------------------------------------------------------------------------



; Display the board --------------------------------------------------------
(defmethod display((b board) &aux width rows)
    (setf width (board-width b))
    (setf rows (board-rows b))
    (displayTop width)
    (loop for row in rows do
        (display row)
    )
    (newLine width)
)

(defmethod displayMarks((b board) &aux width rows)
    (setf width (board-width b))
    (setf rows (board-rows b))
    (displayTop width)
    (loop for row in rows do
        (displayMarks row)
    )
    (newLine width)
)
; ---------------------------------------------------------------------------
```

Row.l



```
; File: Row.l
(defclass row()
    (
        (number :accessor row-number :initarg :number :initform 0)
        (cells :accessor row-cells :initarg :cells :initform nil)
    )
)


; Constructor -------------------------------------------------------------
(defmethod newRow(width number &aux cells)
    (setf cells (list))
    (dotimes (cellNum width)
        (setf cells (cons (newCell number cellNum) cells))
    )
    (setf cells (reverse cells))
    (make-instance 'row
        :number number
        :cells cells
    )
)
; ---------------------------------------------------------------------------


; Getter -------------------------------------------------------------------
(defmethod getCellFromRow(x (r row))
    (nth x (row-cells r))
)
; -----------------------------------------------------------------------------


; Fire methods -----------------------------------------------------------
(defmethod fireAtRow(x (r row))
    (fireAtCell (nth x (row-cells r)))
)

(defmethod checkHitRow(x (r row) &aux cells)
```

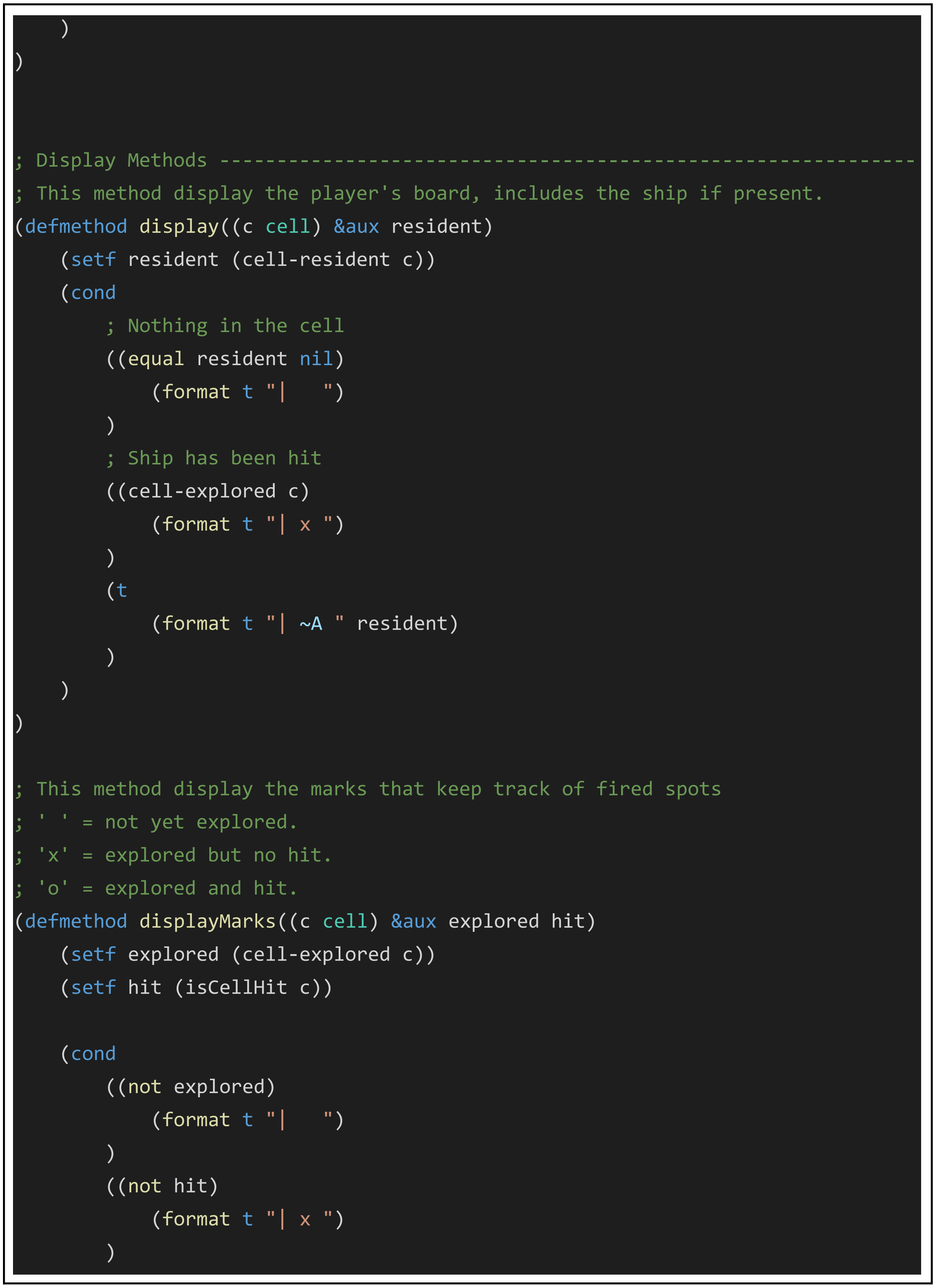
```lisp
    (setf cells (row-cells r))
    (isCellHit (nth x cells))
)
; ----------------------------------------------------------------------------



; Display methods ------------------------------------------------------
; Display the top line, called in Board.l
(defun displayTop(boardWidth)
    (format t "   ")
    (dotimes (n boardWidth)
        (format t "  ~A " (cellToLetter n))
    )
    (format t " ~%")
)

(setf *letters* '(a b c d e f g h i j k l m n o p q r s t u v w x y z))

(defun cellToLetter(cellNumber)
    (nth cellNumber *letters*)
)

(defun letterToCell(cellLetter)
    (position cellLetter *letters*)
)

; Print a new formated line
(defun newLine(cellLength)
    (format t "   ")
    (dotimes (n cellLength)
        (format t "+---")
    )
    (format t "+ ~%")
)

; The main display method
(defmethod display((r row) &aux cells rowLength rowNumber)
    (setf cells (row-cells r))
```



```lisp
    (setf rowLength (length cells))

    ; +---+---+---+
    (newLine rowLength)

    ; Numbers on the left
    (setf rowNumber (row-number r))
    (if (< 9 rowNumber)
        (format t " ~A" rowNumber)
        (format t " ~A " rowNumber)
    )

    ; |   |   |
    (loop for cell in cells do
        (display cell)
    )
    (format t "| ~%")
)

; This method display the firing marks
(defmethod displayMarks((r row) &aux cells rowLength rowNumber)
    (setf cells (row-cells r))
    (setf rowLength (length cells))

    ; +---+---+---+
    (newLine rowLength)

    ; Numbers on the left
    (setf rowNumber (row-number r))
    (if (< 9 rowNumber)
        (format t " ~A" rowNumber)
        (format t " ~A " rowNumber)
    )

    ; |   |   |
    (loop for cell in cells do
        (displayMarks cell)
    )
    (format t "| ~%")
)
; ---------------------------------------------------------------------
```

Cell.l

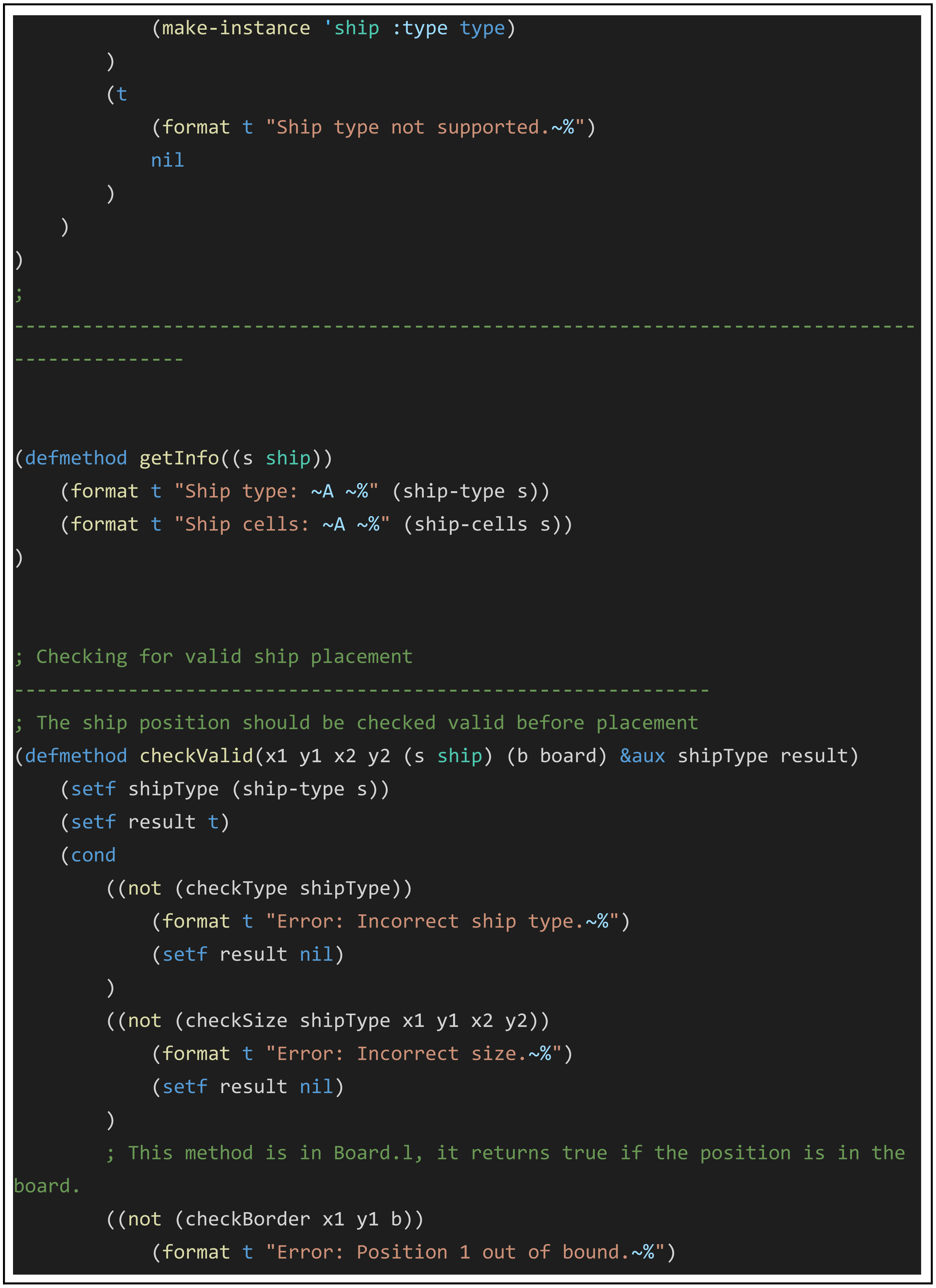
```
; File: Cell.l
; Note: cell-resident should be a positive integer less than 10.
(defclass cell()
    (
        (resident :accessor cell-resident :initform nil)
        (explored :accessor cell-explored :initform nil)
        (cellRow :accessor cell-row :initarg :cellRow :initform 0)
        (cellNum :accessor cell-num :initarg :cellNum :initform 0)
    )
)

(defmethod newCell(rowNumber cellNumber)
    (make-instance 'cell
        :cellRow rowNumber
        :cellNum cellNumber
    )
)

(defmethod setCellResident((c cell) resident)
    (setf (cell-resident c) resident)
)

(defmethod fireAtCell((c cell))
    (setf (cell-explored c) t)
)

(defmethod isCellHit((c cell) &aux explored resident)
    (setf explored (cell-explored c))
    (setf resident (cell-resident c))
    (and
        explored
        (not (equal resident nil))
```



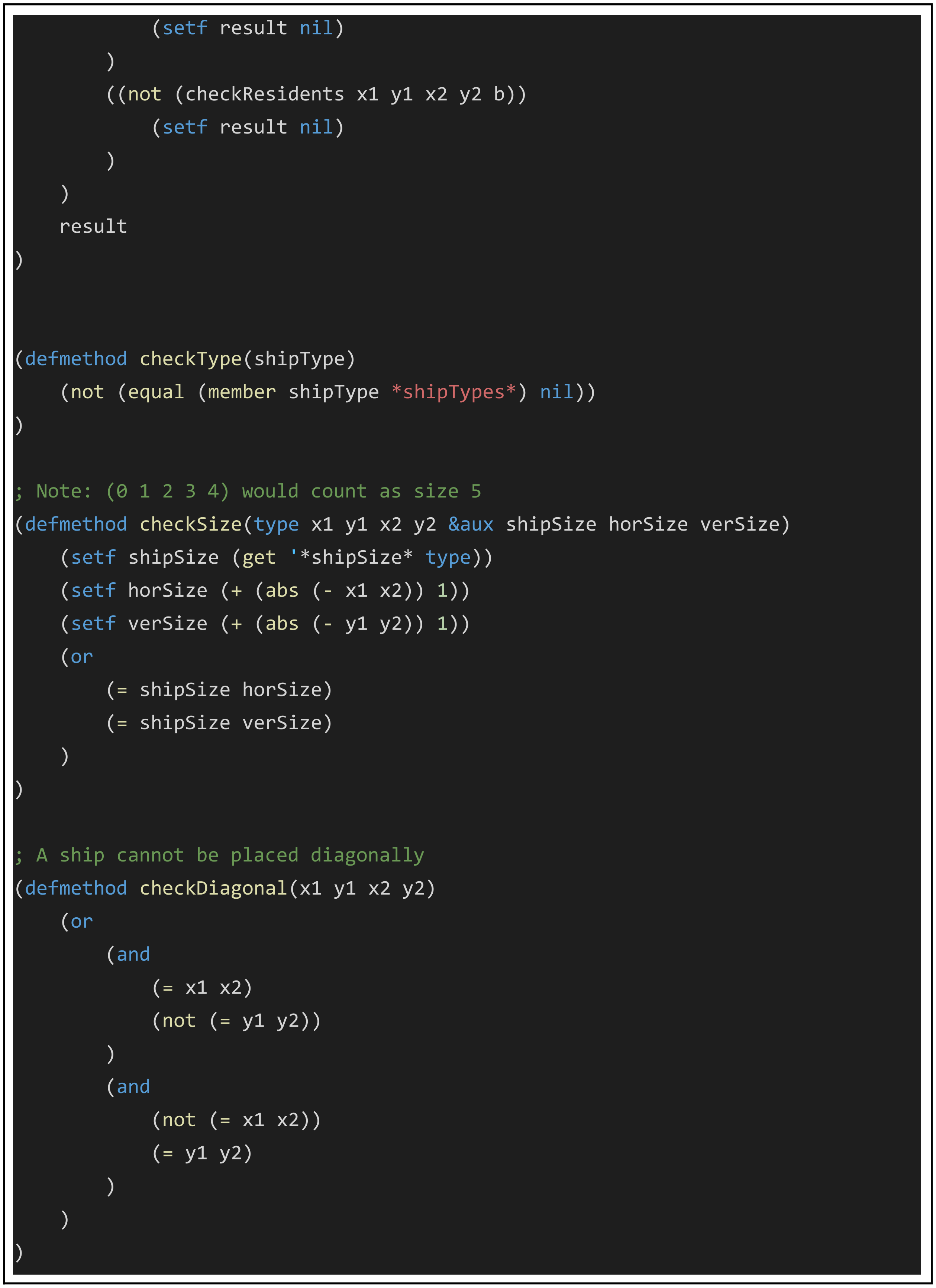
```lisp
    )
)



; Display Methods ---------------------------------------------------------------
; This method display the player's board, includes the ship if present.
(defmethod display((c cell) &aux resident)
    (setf resident (cell-resident c))
    (cond
        ; Nothing in the cell
        ((equal resident nil)
            (format t "|   ")
        )
        ; Ship has been hit
        ((cell-explored c)
            (format t "| x ")
        )
        (t
            (format t "| ~A " resident)
        )
    )
)

; This method display the marks that keep track of fired spots
; ' ' = not yet explored.
; 'x' = explored but no hit.
; 'o' = explored and hit.
(defmethod displayMarks((c cell) &aux explored hit)
    (setf explored (cell-explored c))
    (setf hit (isCellHit c))

    (cond
        ((not explored)
            (format t "|   ")
        )
        ((not hit)
            (format t "| x ")
        )
        (t
            (format t "| o ")
        )
    )
)
;
-------------------------------------------------------------------------------

; This method is for development purposes
(defmethod getInfo((c cell))
    (format t "Cell row: ~A~%" (cell-row c))
    (format t "Cell number: ~A~%" (cell-num c))
    (format t "Cell resident: ~A~%" (cell-resident c))
    (format t "Cell explored: ~A~%" (cell-explored c))
)
```

Ship.l

```
; File Ship.l
(defclass ship()
    (
        (type :accessor ship-type :initarg :type)
        (cells :accessor ship-cells :initform nil)
    )
)

(setf *shipTypes* '(carrier battleship cruiser submarine destroyer))
(setf (symbol-plist '*shipRep*) '(carrier 5 battleship 4 cruiser 3 submarine 2
destroyer 1))
(setf (symbol-plist '*shipSize*) '(carrier 5 battleship 4 cruiser 3 submarine 3
destroyer 2))


; Constructor class
----------------------------------------------------------------------------
(defmethod newShip(type)
    (cond
        ((not (equal (member type *shipTypes*) nil))
```



```
            (make-instance 'ship :type type)
        )
        (t
            (format t "Ship type not supported.~%")
            nil
        )
    )
)
;
-------------------------------------------------------------------------------
---------------



(defmethod getInfo((s ship))
    (format t "Ship type: ~A ~%" (ship-type s))
    (format t "Ship cells: ~A ~%" (ship-cells s))
)



; Checking for valid ship placement
-------------------------------------------------------------
; The ship position should be checked valid before placement
(defmethod checkValid(x1 y1 x2 y2 (s ship) (b board) &aux shipType result)
    (setf shipType (ship-type s))
    (setf result t)
    (cond
        ((not (checkType shipType))
            (format t "Error: Incorrect ship type.~%")
            (setf result nil)
        )
        ((not (checkSize shipType x1 y1 x2 y2))
            (format t "Error: Incorrect size.~%")
            (setf result nil)
        )
        ; This method is in Board.l, it returns true if the position is in the
board.
        ((not (checkBorder x1 y1 b))
            (format t "Error: Position 1 out of bound.~%")
            (setf result nil)
        )
        ((not (checkBorder x2 y2 b))
            (format t "Error: Position 2 out of bound.~%")
            (setf result nil)
        )
        ((not (checkDiagonal x1 y1 x2 y2))
            (format t "Error: Ship needs to be either vertical or
horizontal.~%")
            (setf result nil)
        )
        ((not (checkResidents x1 y1 x2 y2 b))
            (format t "Error: Cells already occupied.~%")
            (setf result nil)
        )
    )
    result
)

; This one is for non-players
(defmethod checkValidNoText(x1 y1 x2 y2 (s ship) (b board) &aux shipType
result)
(setf shipType (ship-type s))
    (setf result t)
    (cond
        ((not (checkType shipType))
            (setf result nil)
        )
        ((not (checkSize shipType x1 y1 x2 y2))
            (setf result nil)
        )
        ((not (checkBorder x1 y1 b))
            (setf result nil)
        )
        ((not (checkBorder x2 y2 b))
            (setf result nil)
        )
        ((not (checkDiagonal x1 y1 x2 y2))
```

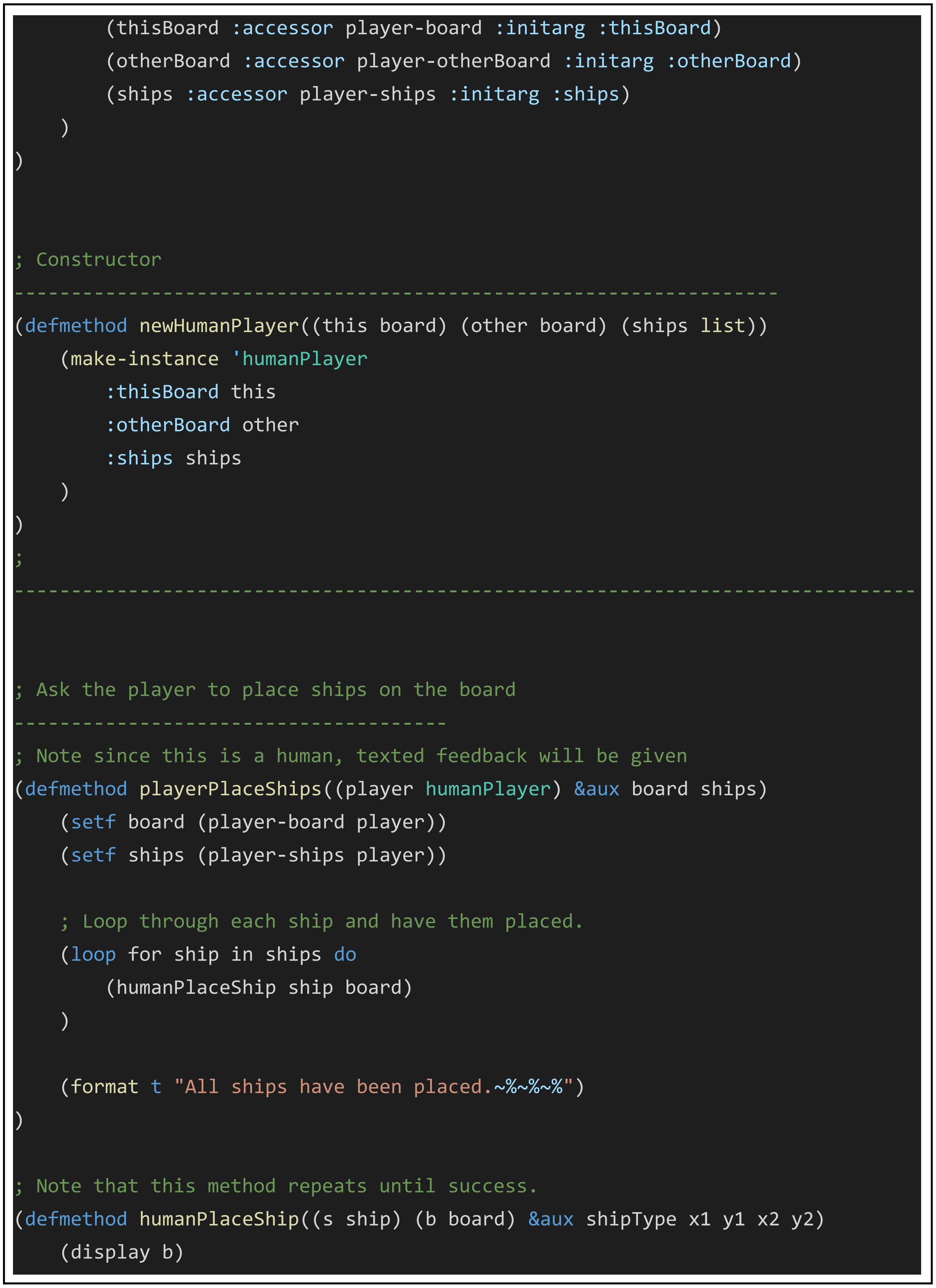


```
            (setf result nil)
        )
        ((not (checkResidents x1 y1 x2 y2 b))
            (setf result nil)
        )
    )
    result
)



(defmethod checkType(shipType)
    (not (equal (member shipType *shipTypes*) nil))
)

; Note: (0 1 2 3 4) would count as size 5
(defmethod checkSize(type x1 y1 x2 y2 &aux shipSize horSize verSize)
    (setf shipSize (get '*shipSize* type))
    (setf horSize (+ (abs (- x1 x2)) 1))
    (setf verSize (+ (abs (- y1 y2)) 1))
    (or
        (= shipSize horSize)
        (= shipSize verSize)
    )
)

; A ship cannot be placed diagonally
(defmethod checkDiagonal(x1 y1 x2 y2)
    (or
        (and
            (= x1 x2)
            (not (= y1 y2))
        )
        (and
            (not (= x1 x2))
            (= y1 y2)
        )
    )
)
```

```
; All the cells that the ship is going to occupy have to be empty
; The methods uses here are defined down below.
(defmethod checkResidents(x1 y1 x2 y2 (b board) &aux cells result)
    (if (= x1 x2)
        (setf cells (sameXCells x1 y1 y2 b))
        (setf cells (sameYCells x1 x2 y1 b))
    )
    (setf result t)
    (loop for cell in cells do
        (setf result (and result (checkCell cell)))
    )
    result
)

; Check if the cell is empty
(defmethod checkCell((c cell))
    (equal (cell-resident c) nil)
)
;
-------------------------------------------------------------------------------
---------------------



; Placing ships
-------------------------------------------------------------------------------
-------


; Note, ship is placed with this function
; Ship position should be checked before this function is called!
; Can't specify ship object because it is loaded after this
; (carrier battleship cruiser submarine destroyer)
(defmethod placeShip(x1 y1 x2 y2 (s ship) (b board) &aux cells shipType)
    (setf shipType (ship-type s))
    (if (= x1 x2)
        (setf cells (sameXCells x1 y1 y2 b))
        (setf cells (sameYCells x1 x2 y1 b))
    )
    (setResidents cells shipType)
    (setShipCells s cells)
)



; Mark the resident at the given cells
(defmethod setResidents(cells shipType &aux shipNum)
    (setf shipNum (get '*shipRep* shipType))
    (dotimes (n (length cells))
        (setCellResident (nth n cells) shipNum)
    )
)

; Associate this ship with its cells
(defmethod setShipCells((s ship) cells)
    (setf (ship-cells s) cells)
)

; Return all the cells between two Ys.
(defmethod sameXCells(x y1 y2 (b board))
    (cond
        ((= y1 y2)
            (list (getCell x y1 b))
        )
        ((< y1 y2)  ; Up to down
            (cons (getCell x y1 b) (sameXCells x (+ y1 1) y2 b))
        )
        (t          ; Down to up
            (cons (getCell x y2 b) (sameXCells x y1 (+ y2 1) b))
        )
    )
)

; Return all the cells between two Xs.
(defmethod sameYCells(x1 x2 y (b board))
    (cond
        ((= x1 x2)
            (list (getCell x1 y b))
        )
        ((< x1 x2)  ; Left to right
            (cons (getCell x1 y b) (sameYCells (+ x1 1) x2 y b))
        )
        (t          ; Right to left
            (cons (getCell x2 y b) (sameYCells x1 (+ x2 1) y b))
        )
    )
)
;
-------------------------------------------------------------------------------
--



; Check if a ship has been sunk, returns true if it is
-----------------------------
(defmethod isShipSunk((s ship) &aux cells result hit)
    (setf cells (ship-cells s))
    (setf result t)

    ; Loop through each cell and see if it had been fired at.
    (dotimes (n (length cells))
        (setf hit (cell-explored (nth n cells)))
        (setf result (and result hit))
    )

    result
)
;
-------------------------------------------------------------------------------
---
```

HumanPlayer.l

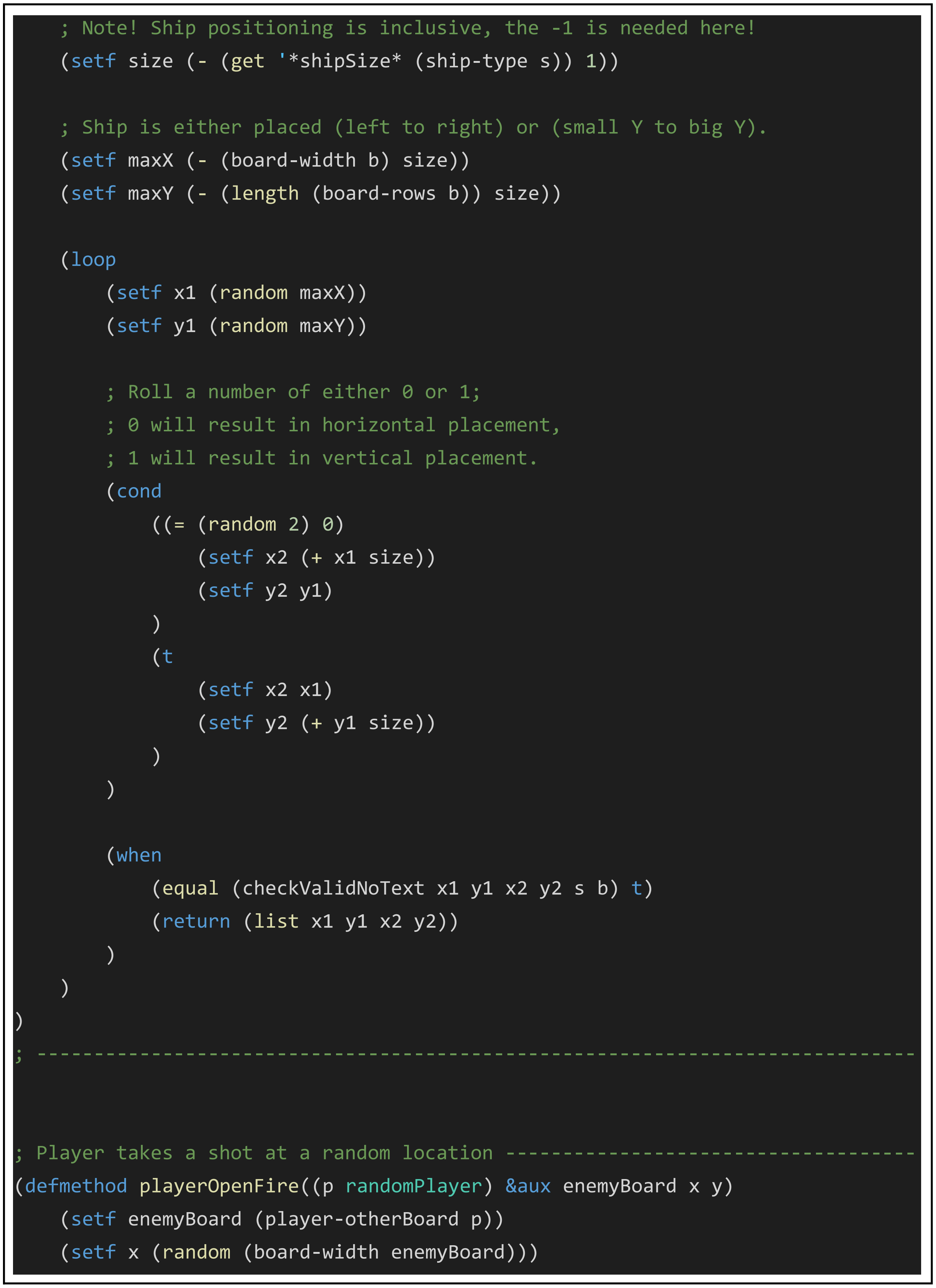
```
; File HumanPlayer.l

(defclass humanPlayer()
    (
```



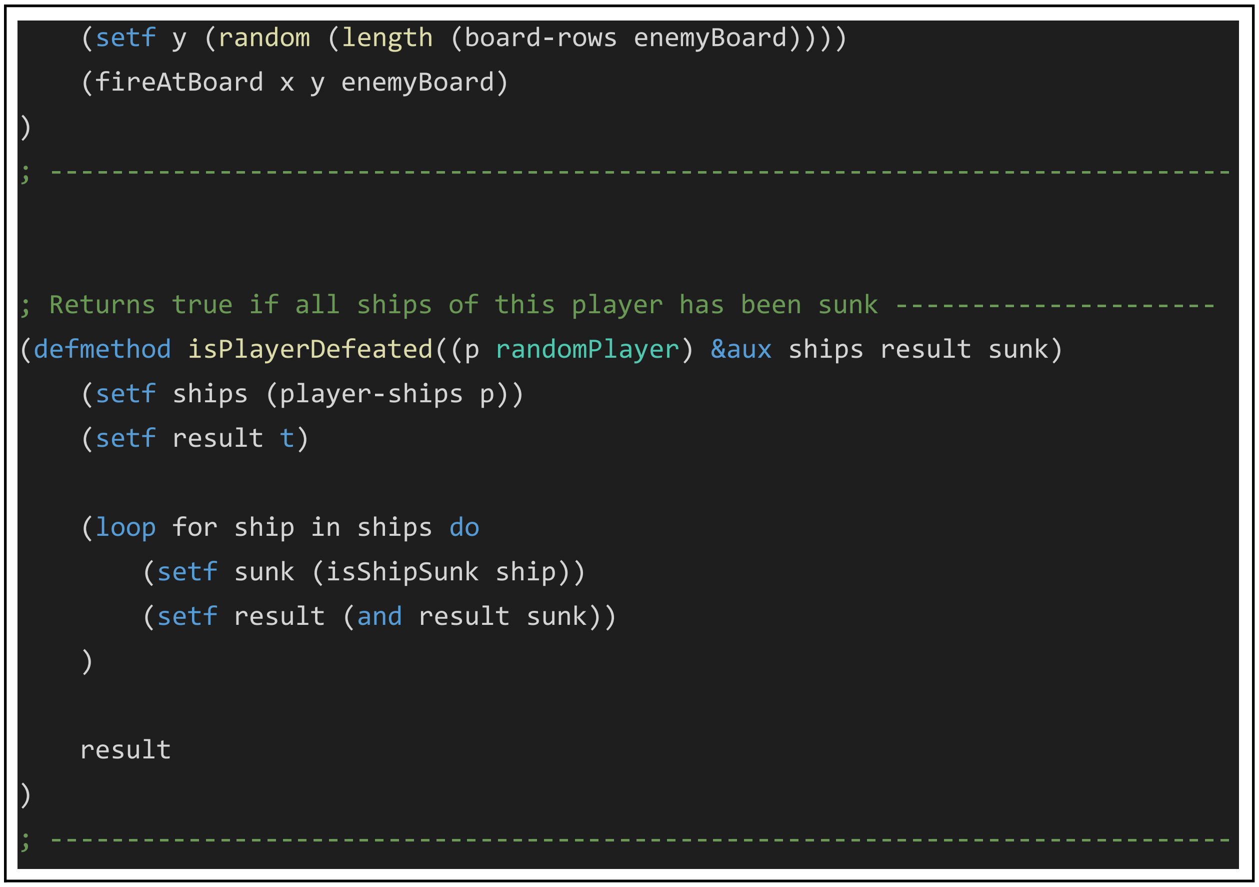
```
        (thisBoard :accessor player-board :initarg :thisBoard)
        (otherBoard :accessor player-otherBoard :initarg :otherBoard)
        (ships :accessor player-ships :initarg :ships)
    )
)



; Constructor
-------------------------------------------------------------------
(defmethod newHumanPlayer((this board) (other board) (ships list))
    (make-instance 'humanPlayer
        :thisBoard this
        :otherBoard other
        :ships ships
    )
)
;
-------------------------------------------------------------------------------



; Ask the player to place ships on the board
--------------------------------------
; Note since this is a human, texted feedback will be given
(defmethod playerPlaceShips((player humanPlayer) &aux board ships)
    (setf board (player-board player))
    (setf ships (player-ships player))

    ; Loop through each ship and have them placed.
    (loop for ship in ships do
        (humanPlaceShip ship board)
    )

    (format t "All ships have been placed.~%~%~%")
)

; Note that this method repeats until success.
(defmethod humanPlaceShip((s ship) (b board) &aux shipType x1 y1 x2 y2)
    (display b)
    (setf shipType (ship-type s))
    (format t "Now placing: ~A, size: ~A~%" shipType (get '*shipSize*
shipType))
    (format t "Enter position 1: ")
    (setf x1 (read))
    (setf x1 (letterToCell x1))
    (setf y1 (read))
    (format t "Enter position 2: ")
    (setf x2 (read))
    (setf x2 (letterToCell x2))
    (setf y2 (read))

    (if (checkValid x1 y1 x2 y2 s b)
        (placeShip x1 y1 x2 y2 s b)
        (humanPlaceShip s b)
    )
)
;
-------------------------------------------------------------------------------
-----



; Allow the player to take a shot at inputed position
--------------------------------
(defmethod playerOpenFire((p humanPlayer) &aux myBoard enemyBoard pos)
    (setf myBoard (player-board p))
    (setf enemyBoard (player-otherBoard p))

    (format t "Your markers: ~%")
    (displayMarks enemyBoard)

    (format t "Your board: ~%")
    (display myBoard)

    ; Give a little feedback on where the enemy recently fired.
    (setf pos (board-recent myBoard))
    (if (not (equal pos nil))
        (format t "Enemy fired at ~A, ~A~%" (first pos) (second pos))
    )

    (setf pos (getPlayerInput enemyBoard))
    (fireAtBoard (first pos) (second pos) enemyBoard)

    ; Give a little feedback on the result of recent firing.
    (if (isCellHit (getCell (first pos) (second pos) enemyBoard))
        (format t "It's a hit!.~%")
        (format t "It missed.~%")
    )
)

(defmethod getPlayerInput((b board) &aux x y)
    (format t "Enter target location: ")
    (setf x (read))
    (setf x (letterToCell x))
    (setf y (read))

    (cond
        ((checkBorder x y b)
            (list x y)
        )
        (t
            (format t "Position out of bound.~%")
            (getPlayerInput b)
        )
    )
)
;
--------------------------------------------------------------------------------
-----
```

RandomPlayer.l

```
; File: RandomPlayer.l

(defclass randomPlayer()
    (
        (thisBoard :accessor player-board :initarg :thisBoard)
        (otherBoard :accessor player-otherBoard :initarg :otherBoard)
        (ships :accessor player-ships :initarg :ships)
    )
)



; Constructor -----------------------------------------------------------------
(defmethod newRandomPlayer((this board) (other board) (ships list))
    (make-instance 'randomPlayer
        :thisBoard this
        :otherBoard other
        :ships ships
    )
)
; -----------------------------------------------------------------------------



; Ask the random player to place ships on the board ---------------------------
; Note, the board should be at least 5 x 5 in order for all ships to be placed.
; Loop through the ships this player have, placing them one at a time.
(defmethod playerPlaceShips((player randomPlayer) &aux board ships)
    (setf ships (player-ships player))
    (setf board (player-board player))

    ; Loop through the ships that have to be placed
    (loop for ship in ships do
        (randomlyPlaceShip ship board)
    )
)

(defmethod randomlyPlaceShip((s ship) (b board) &aux pos)
    (setf pos (getRandomPosition s b))
    (placeShip (first pos) (second pos) (third pos) (fourth pos) s b)
)

(defmethod getRandomPosition((s ship) (b board) &aux size maxX maxY x1 y1 x2
y2)
```



```lisp
    ; Note! Ship positioning is inclusive, the -1 is needed here!
    (setf size (- (get '*shipSize* (ship-type s)) 1))

    ; Ship is either placed (left to right) or (small Y to big Y).
    (setf maxX (- (board-width b) size))
    (setf maxY (- (length (board-rows b)) size))

    (loop
        (setf x1 (random maxX))
        (setf y1 (random maxY))

        ; Roll a number of either 0 or 1;
        ; 0 will result in horizontal placement,
        ; 1 will result in vertical placement.
        (cond
            ((= (random 2) 0)
                (setf x2 (+ x1 size))
                (setf y2 y1)
            )
            (t
                (setf x2 x1)
                (setf y2 (+ y1 size))
            )
        )

        (when
            (equal (checkValidNoText x1 y1 x2 y2 s b) t)
            (return (list x1 y1 x2 y2))
        )
    )
)
; ----------------------------------------------------------------------------



; Player takes a shot at a random location ------------------------------------
(defmethod playerOpenFire((p randomPlayer) &aux enemyBoard x y)
    (setf enemyBoard (player-otherBoard p))
    (setf x (random (board-width enemyBoard)))
```



```
    (setf y (random (length (board-rows enemyBoard))))
    (fireAtBoard x y enemyBoard)
)
; -----------------------------------------------------------------------------



; Returns true if all ships of this player has been sunk ---------------------
(defmethod isPlayerDefeated((p randomPlayer) &aux ships result sunk)
    (setf ships (player-ships p))
    (setf result t)

    (loop for ship in ships do
        (setf sunk (isShipSunk ship))
        (setf result (and result sunk))
    )

    result
)
; -----------------------------------------------------------------------------
```