SWEN222

ASSIGNMENT 1 - SWORDS AND SHIELDS

Daniel Walker - 12 August 2017

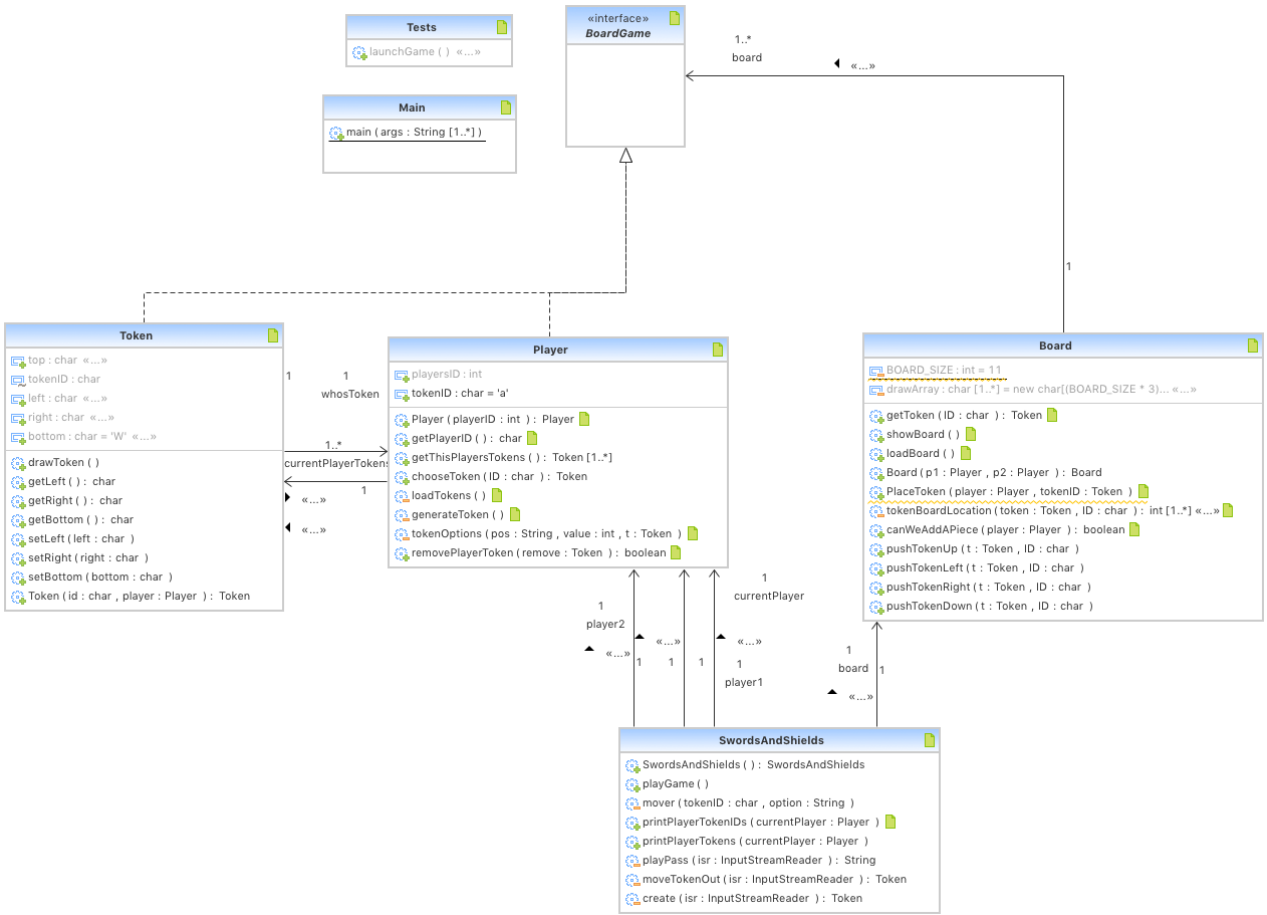
Introduction

Itinerary:

1. **Class Diagram (pg 2)**
2. **Design report (pg 3)**
3. **Good code highlight (pg 4)**
4. **Testing discussion (pg 5)**

1 - Class Diagram

Created using UML Lab (Also hand drawn copy at the end if required)



2 - Design Report

My program meets stage 1 requirements and I have implemented token rotations for stage 2. However I have not successfully implemented "Undo".

Basic steps on how to play:

Game: "pass or play?"
User: "play"
Game: "which token would you like to play"
User: "a" (plays token with id "a")
Game: "would you like to move/rotate the token?"
User: "yes"
Game: "rotation…"90,180,270"
User: "90"
Game: "which direction.. "left, right, up, down"
User: "right"

Game then switches to player 2 and you can repeat the process

I have implemented a 'Board.java' class that loads and redraws (which checks for any new tokens) a 10 x 10 board after each turn of the game.

I have successfully implemented the following:

- Have a board with two players. Green = player1, Yellow = player2
- Each player has a creation square
- Each player receives 24 randomly generated tokens
- Tokens can be chosen using a unique ID given to each token (represented by lower case letters i.e this token has an id: 'a', a shield '#' on the left, a sword '-' to the right and below, and no weapon on the top.

- Players have to option to move a token when it is their turn.
    - Option 1: player is asked if they would like to move the token once they placed a token.
    - Option 2: if a player wanted to "play" their turn, and there is a token in the creation square, they are prompted to move it.
- The program can push other tokens when performing a move. The logic is only basic and it does not full work. (See Class Board (line 81-153).
- Undo is not implemented
- Rotation has been implemented but is a bit buggy, (most likely because of the way I draw the token)

Not part of design report
I have spent about 15-20 hours per week on this assignment (a few more than the recommended 5/week I know haha *crying on inside*) . I struggled a lot in the first few weeks as I was could not understand how to implement the logic into a textual interface, but by mid way through the last week, I developed a better understanding and started to get better at implementing basic features for the game. If I had another week, I think I could've gotten much more of the brief's requirements done.

3 - Good Code Highlight

```
/**
 * Load 24 tokens for each player which consists of a random permutation of a combo of swords/shields/nothing
 */
private void generateToken(){
    Token newToken = new Token(tokenID++,  player: this);
    Random random = new Random();

    //generates a random number for each side for assigning weapon
    int top = random.nextInt( bound: 3) + 1;
    int left = random.nextInt( bound: 3) + 1;
    int bottom = random.nextInt( bound: 3) + 1;
    int right = random.nextInt( bound: 3) + 1;

    // add an option to each side of the new token
    tokenOptions( pos: "top", top, newToken);
    tokenOptions( pos: "left", left, newToken);
    tokenOptions( pos: "bottom", bottom, newToken);
    tokenOptions( pos: "right", right, newToken);

    currentPlayerTokens.add(newToken);
}
```

I think my way of generating the pieces for each player demonstrates "brilliant code". I had spent so long on trying to figure out how to do this, exploring several options. Discussing solutions among friends, I came up with this solution to be the best. Another good way I was considering was to represent a token by splitting it into 3 rows. getTOP(), getMID() etc. But for me, I found my solution easier to allow me to draw tokens to the board. Using loadBoard, I was able to check whether the next [row][col] was instanceOf Token. I discovered a similar implementation to this for a simple card game, which was used for dealing cards randomly. This game me the idea to use Random for generating the weapons.

```
/**
 * Considers the three possible options for each token (Sword
 * Checks which side to assign the symbol
 * 1 for Sword, 2 for Shield, 3 for nothing
 */
private void tokenOptions(String pos, int value, Token t){

    if(pos.equals("top")){
        char weapon = 'W';
        if(value == 1){
            weapon = '|';
        } else if (value == 2){
            weapon = '#';
        } else{
            weapon = ' ';
        }
        t.setTop(weapon);
    }

    if (pos.equals("left")){
        char weapon = 'W';
        if(value == 1){
            weapon = '-';
        } else if (value == 2){
            weapon = '#';
        } else{
            weapon = ' ';
        }
        t.setLeft(weapon);

    }

    if (pos.equals("bottom")){
        char weapon = 'W';
        if(value == 1){
            weapon = '|';
        } else if (value == 2){
            weapon = '#';
        } else{
            weapon = ' ';
        }
        t.setBottom(weapon);
    }
```

The generateToken method allows each player to receive their 24 tokens. Using Random, each side of the token is assigned a random integer between 1-3. Then, in the tokenOptions method, I read in the position (top,left,bottom,right) and check which integer it was allocated. 1 gives a sword, 2 gives a shield, and 3 gives nothing. generateToken is called in loadToken while (currentPlayersTokens() < 24).

This may not the most efficient implementation, but I thought a simple approach would be robust and a clear implementation.

Testing Discussion

I have split my testing into 3 sections.
- BoardTests
- TokenTests
- PlayerTests

Some tests are working correctly. Some are not yet finished. I struggled to figure out how to compare the location of particular tokens to a location on the board. I had the intension of testing more thoroughly but I didn't know how to do them correctly

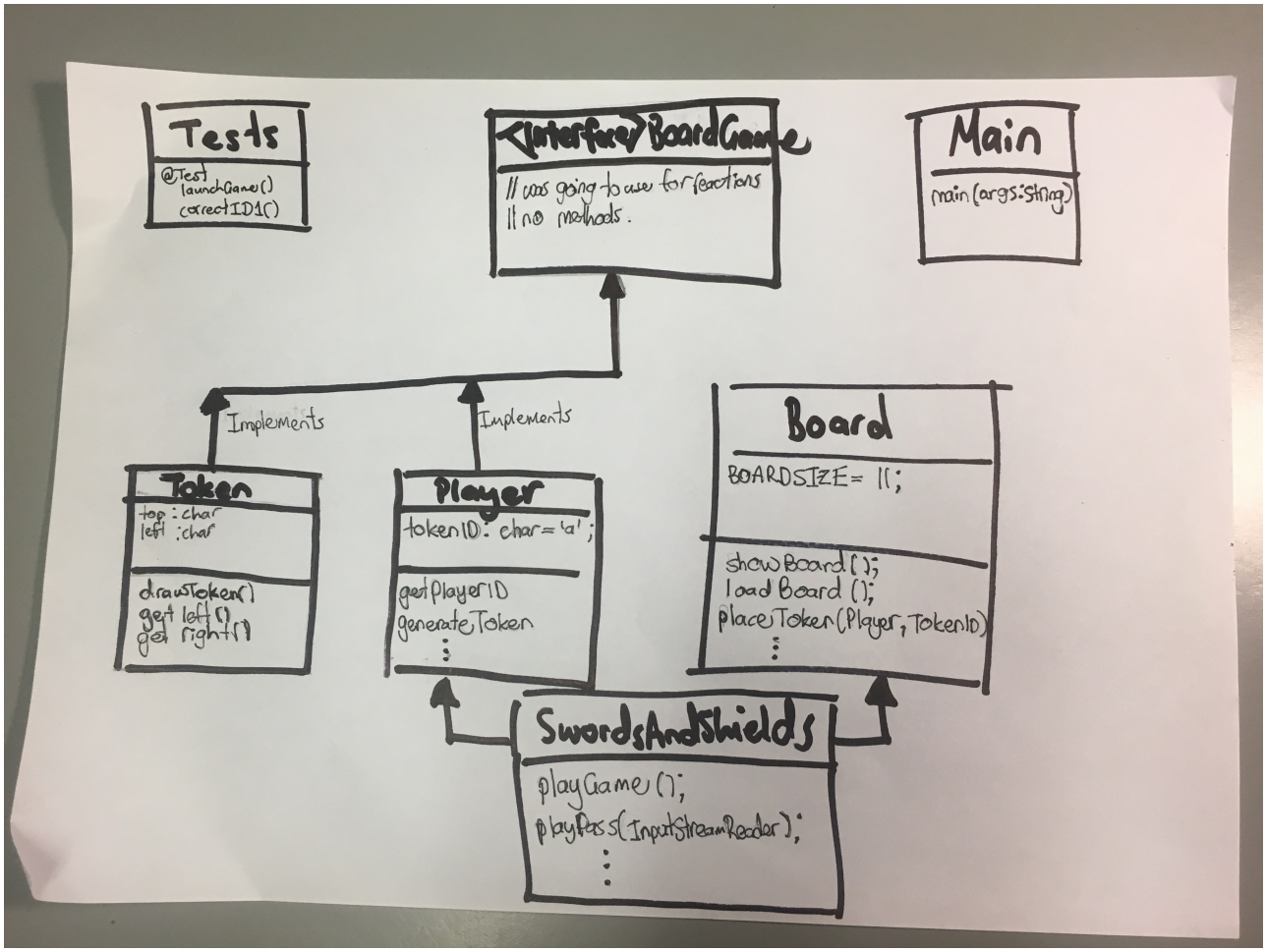
Player tests show the game allocates the correct IDs to the player and that their creation square is is the correct location. It also shows the program can correctly return the token requested.

Token Tests show the game allocates the correct number of tokens to each player. It shows the game will check if we the creation square is free, and tell us correctly if we can/can't place a token.

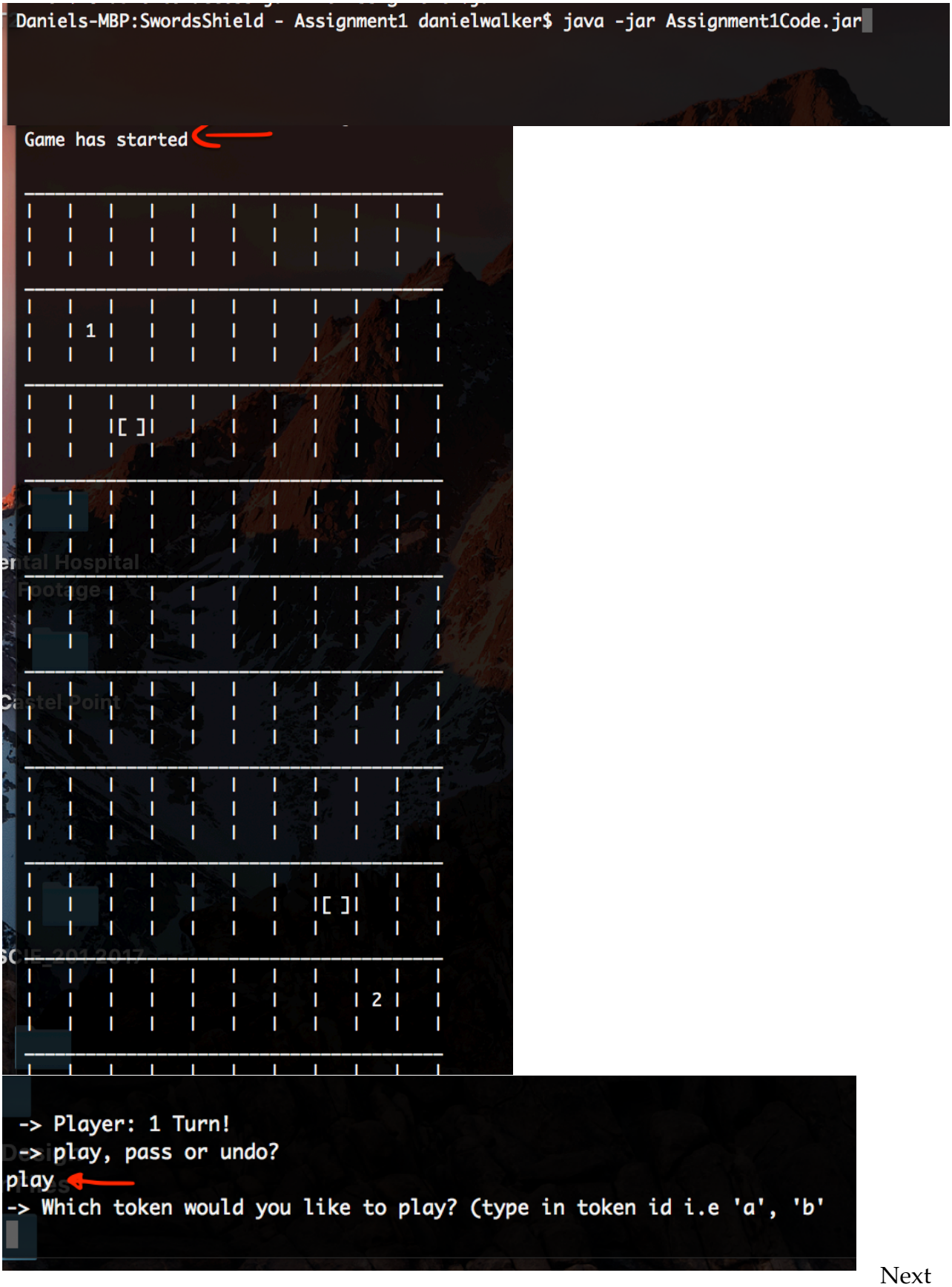
Board tests show so we can create a new game. I wanted to write more tests on checking instances of the board during mid game. But I couldn't figure out how to compare two instances of a board. (I feel like some insight on this sort of thing could be useful in lectures).

I only have 13 tests, but I will be working on adding more tests to improve the correctness of my code.

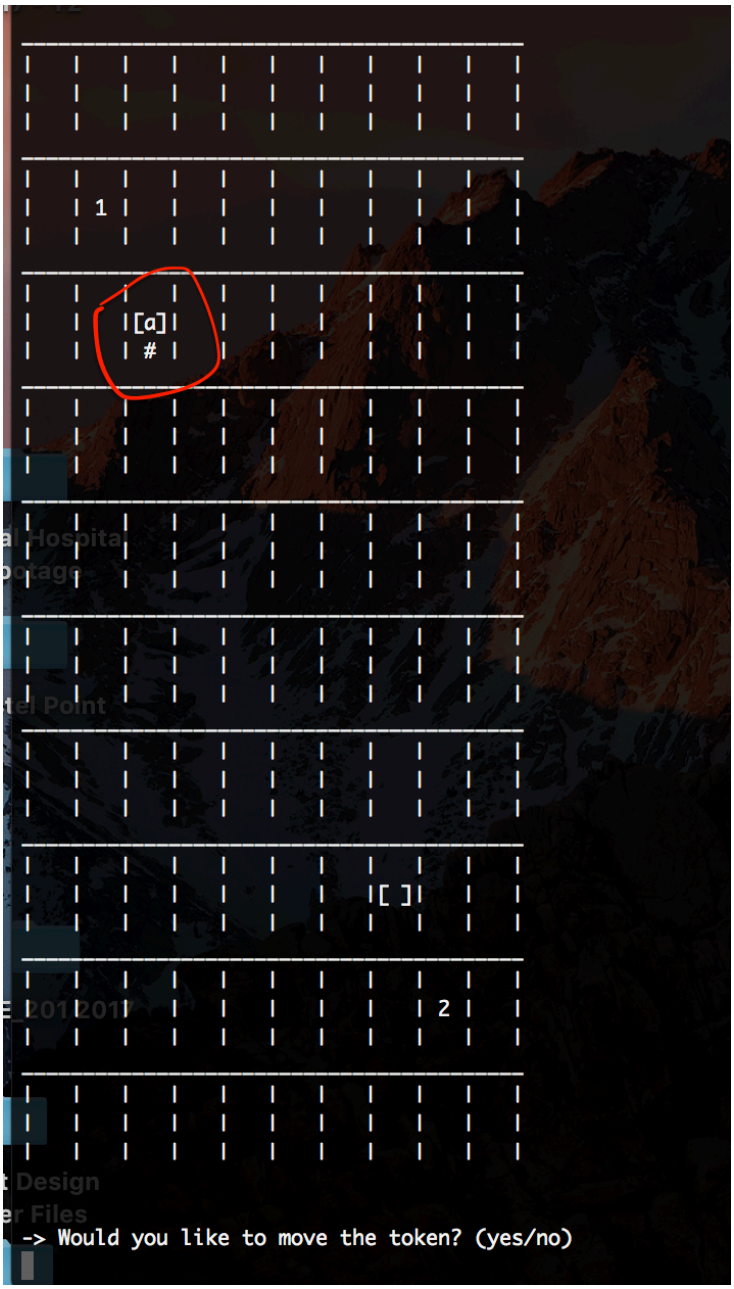
Hand Drawn Class diagram if required:



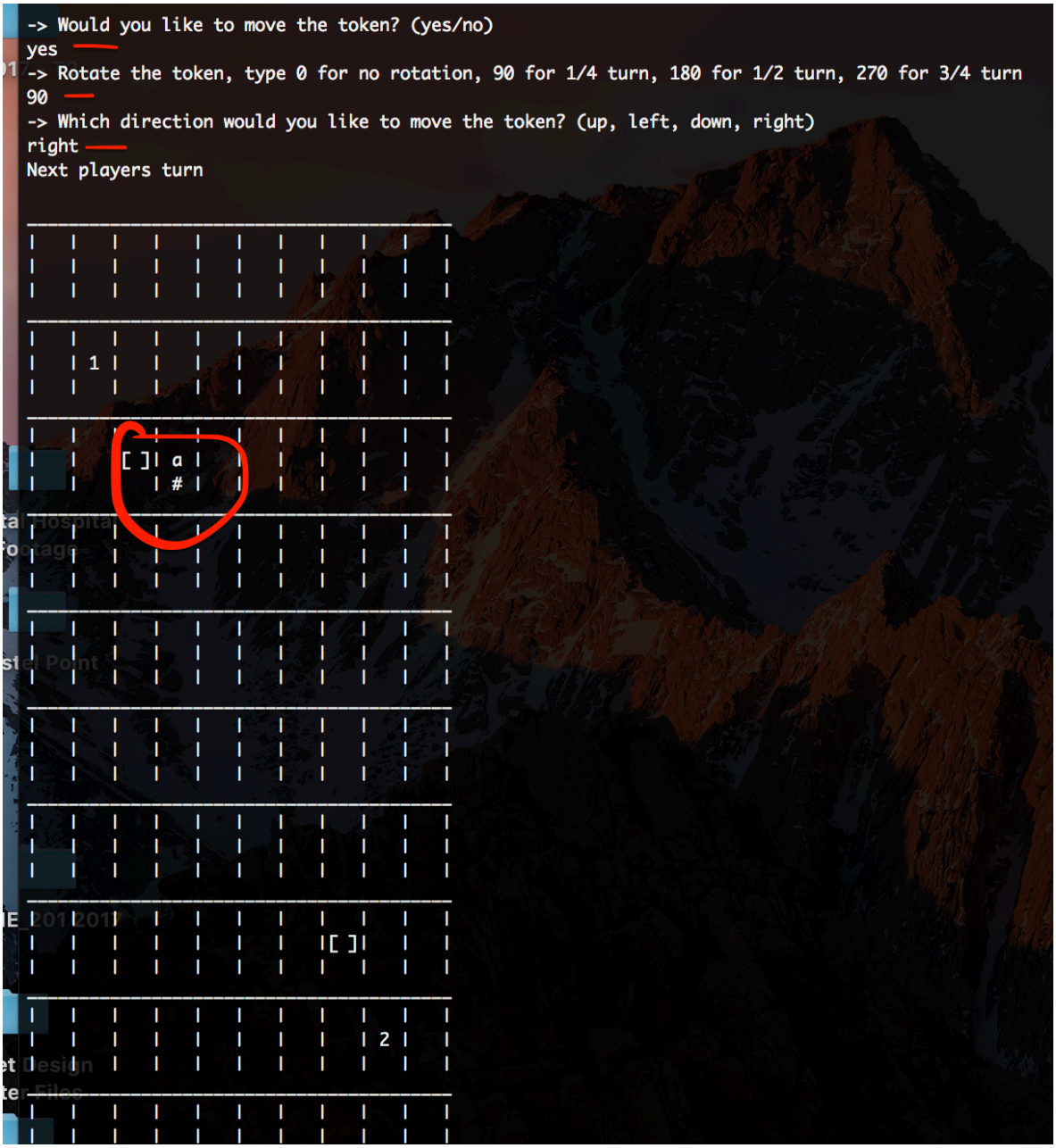
How to play if you need help:



Next Page…



Next page….


-> Would you like to move the token? (yes/no)
yes
-> Rotate the token, type 0 for no rotation, 90 for 1/4 turn, 180 for 1/2 turn, 270 for 3/4 turn
90
-> Which direction would you like to move the token? (up, left, down, right)
right
Next players turn
1
[ ] a
#
[ ]
2