Github repo

# Memory

Design Document

Brice Halder

## 0 - Introduction

I've implemented the classic card game Memory in Java as a console-based, Unix-compatible game mostly as a supplement to my application for the Kleiner Perkins Fellows Program, but also because it seemed like a fun thing to code on the side. This program uses emoji's encoded as their HTML Entity decimal codes in order to make it more visually appealing.

## 1 - Building & Running

This project uses Maven as a build system. Execute `mvn package` while in the project root directory to package the program into a jar, then `java -cp target/Memory-1.0-SNAPSHOT.jar memory.Memory -Dfile.encoding=UTF-8` to run the program. The flag at the end is to ensure that the emojis render correctly.

## 2 - Modularized Class Overview

The following is a high-level overview of what each class does:

**Memory.java:** This class contains the main method/acts as driver class/handles the general game loop & turn switching. Runs until there are no more unmatched cards(implemented as emojis) on the board, then prints each players score.

**Board.java:** Maintains the state of the board (i.e. all of the cards). Determines which cards are visible & in play. Handles printing of the board to stdout.

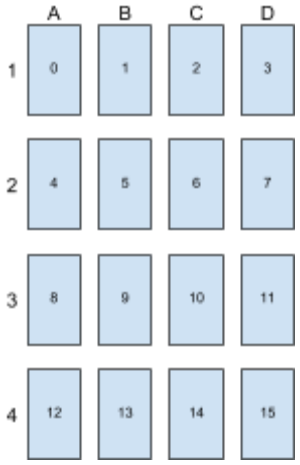
**Card.java:** The individual cards. Each card has a symbol (emoji). Printing the card simply prints its symbol to stdout.

**Player.java:** Class to hold information about each player. Keeps track of their player number, the number of matching pairs chosen, and the number of non-matching pairs chosen.

## 3 - Design Choice Analysis

+ Modularity of and within classes allows for upgrades/modifications to easily be made in the future. It also allows for the program to be highly scalable.

+ While using a HashMap from card number to coordinates would also make sense, the choice to use an ArrayList to keep track of Cards allows for an in-place O(N) Fisher-Yates shuffling to set up the initial randomized playing board (a LinkedList would take O(N) space as it would have to be copied to an array).
    - Indices in a List easily map to 2D coordinates in our playing board, which allows for boards of any dimension.
        - The user is asked for input for the rows and columns at the start of the game.
    - The ArrayList is initialized with the total # of cards, so no resizing should occur.
    - The only operations used are adding (to the end of the list) and retrieval, which are both constant time for ArrayLists.
    - We want to maintain the ordering of the cards and keep duplicates, so a list is a better choice than a set or multiset.




List indices easily map to a coordinate system

+ A HashSet is used to keep track of whether a card has been matched or not. We add one of each unique card to the set initially, then check throughout the game if the card is a member of the set to see if it has been matched, and removing them once they have been matched. HashSets are the natural choice to use here with constant time operations for everything listed.

- Board.java could probably be refactored into some other classes, as it currently handles a lot of functionality. However, I wouldn't call it a god-class.

## 4 - Interaction & User Input

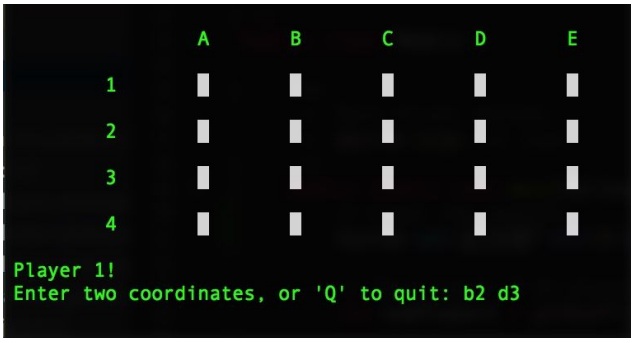
The game starts by asking the user to input the number of players and the dimensions of the board.

```
Enter number of players: 2

Enter rows and columns (e.g. "2 3" for a 2x3 board): 4 5
```

The board then generates the cards from a set of 100 different emoji and players take turns guessing coordinates until the game is complete. Either user can input ‘Q’ to quit the game.

- ➢ Coordinate entries should be in the format [letter][number]
- ➢ All commands are case insensitive





The program validates input by checking if the coordinates exist and are of two unmatched cards The “cards” are flipped for 1.5s, and then flipped back face-down if it wasn’t a match.

The game proceeds until all cards have been matched, and the players are ranked according to the number of matching pairs. The number of non-matching pairs chosen is also recorded and used in ranking in the case of a tie.