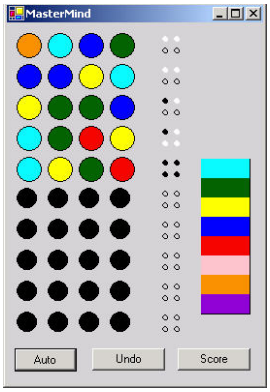
CSC172 PROJECT 1

MASTERMIND



1 Introduction

The game MASTERMIND is a code-breaking guessing game that presents an interesting problem in combinatorics. In the traditional game one player selects (in order, with replacement) four pegs from six possible colors. The combination is kept secret. The opposing player guesses a combination. The first player tells the second player how many pegs are of the correct color and in the correct location (traditionally signified by a black token) and also how many pegs are of the correct color but in the wrong location (white token). You can learn a lot about the history of the game, along with a few hints about how to play it at http://en.wikipedia.org/wiki/Mastermind_(board_game).

The internet has a lot of resources for this game. Intellectual collaboration is allowed. You may discuss the project with other students in the lab. It is acceptable to write code on a white board for the benefit of discussion, but you are not allowed to transfer code either electronically or on paper copy. You must hand in a program that you and no one else wrote. You must properly reference any on line source you use.

2 What to build

Starting with the interface, below, implement a mastermind player. Your program is to be the *codebreaker*, the human user is to be the *codemaker*. There are 4 key methods required: a constructor that instantiates the codebreaker, a nextMove method that returns an array of strings describing the next guess, a response method that processes the feedback, and a reset method which starts the game over. The user should be able to specify how many colors and how many positions at the start of the game. (Maximum values are allowed, and you don't need to make the user enter the token colors, but you MUST support a variable number of tokens and positions for full credit.)Your write up should include a discussion of the algorithm you used.

```
public class mm {
    public mm(String[] tokencolors, int positions) {

     // The constructor must take an array of strings as input

     //   For instance, if:
     //     tokencolors ==
     //    {"ORANGE", "PINK","RED", "BLUE", "GREEN", "YELLOW"};

     //  then the engine returns guesses from this set,

     // However, it would be just as valid to pass in
     //     tokencolors ==
     //    {"SNEZZY", "SLEEPY", "DIRTY", "FILTHY", "GRUMPY", "HAPPY", "DOC"};



     //So, the number of elements that your system guesses

     //over is determined by the array

    }

    public void response(int colorsRightPositionWrong,

                         int positionsAndColorRight) {}

    public void newGame() {}   // Reset the game

    public String [] nextMove() {}  // return the next guess


}
```

3 Hand In

Hand in the source code at the appropriate location on the blackboard system at my.rochester.edu. You should hand in a single compressed/archived (i.e. “zipped” file that contains the following.)

1. A plain text file named README that includes your contact information, a brief explanation of the lab (A one paragraph synopsis. Include information identifying what class and lab number your files represent.). And a one sentence explaining the contents of all the other files you hand in.
2. Source code files (you may decide how many you need) representing the work accomplished in this project. All source code files should contain author identification in the comments at the top of the file.
3. A plain text file named OUTPUT that includes author information at the beginning and shows the compile and run steps of your code. The best way to generate this file is to cut and paste from the command line.

4 Grading

Each method in the class is worth 20%. The “driver” program that performs user I/O is worth 10%. Even though this is a command line (text) game, you are expected to have a reasonable and pleasant human-computer interface (greeting the user prompting the user, providing feedback, etc). Proper coding style is worth 10%.