Homework 2

cs1007 - Object-oriented programming and design in Java
Prof. Shlomo Hershkop
Dept of Computer Science
Columbia University
Fall 2005

Programming Due: Oct 17 11:59pm.
Written Part Due: Class Oct 18

Reading: Chapters 2, 3
Objective:

1. Get some theory practice with design docs.
2. Practice with your java.
3. Answer some object oriented design questions

---

1) Consider a website which allows customers to order items from its catalog and pay with a credit card. Draw a UML diagram that shows the relationship between the classes:
    a. Customer
    b. Order
    c. RushOrder
    d. Product
    e. Address
    f. CreditCard

    use violet (as demoed in class) to create the UML diagrams.

2) Create and document a use case of a customer returning an item which they ordered.

3) Draw a sequence diagram from problem 1 of a customer ordering a specific product from the website.

4) Critique the java.io.File class.
    a. Where is it not cohesive ?
    b. Where does it lack clarity?
    c. Where is it inconsistent?

5) The job of the NumberFormat class is to format numbers so that they can be presented to a human reader in a format such as an invoice or table. For example, to format a floating point value with two digit precision and trailing zeroes, you use the following code:

    ```
    NumberFormat formatter = NumberFormat.getNumberInstance();
    formatter.setMinimumFractionDigits(2);
    formatter.setMaximumFractionDigit(2);
    String formattedNumber = formatter.format(x);
    ```

    critique this class. Is it convenient? is it clear? Is it complete? (Hint: How would you format a table of values for that columns line up?)

6) This is in preparation for the next homework assignment. You will design a program to help users built family tree information databases.
    a. Using the CRC cards, sketch out the system for a graphical based interface for a family tree program.
    b. Now draw out the UML design using the same classes sketched out in your part a.

Programming Section

It seems that 1007, is sometimes not fun enough. In order to make it a little more fun, we will be programming a game for this homework.

Mastermind™.

Mastermind, you read that right!! This is a game you will program the computer to play against yourself or any of your friends. In this game, the computer randomly picks 3 numbers, each in the range 1 through 6. Think of this as 3 slots, with 6 possible values per slot. You, in turn, keep guessing what the 3 values are until you get them correct. Each time you guess, the computer gives you hints about how close you came by telling you two things:

1. How many slots you got the correct value for.
2. How many of the values you guessed appear somewhere in the computer's choice. That is, how many correct values you got, even if in the wrong slot.

For example, if the computer picks (4,2,3) and you guess (4,1,2), then the computer would tell you that you're WRONG, however it would also respond with "1" and "2", respectively. Note that the 4 you guessed counted in both parts of the hint.

Each of your guesses can only count once in the second part of the hint. For example, if the computer picks (2,2,2) and you guess (1,2,3), the hint is "1" and "1". However, if the computer picked (1,2,3) and you guess (2,2,2), the response should be "1", "3".

There is a limit to how many guesses you can guess. Set it to 20, but make sure it can be easily changed (see step 2). You will need to generate new games using a random number generator. Use the java.util.Random class to get random numbers.

Hint:

```
Random generator = new Random();
int n = generator.nextInt(7);
```

Note: the above call gives you a random number between 0-6, how would you adopt it to give you a random number on the range of 1-7?

Here are requirements and suggestions for your implementation:

1) It ends when the user guesses correctly, and it tells the user how many guesses it took (the less guesses the user took, the more likely it is that the user is either really smart, or cheated somehow). You need to end the game if the number of guesses are used up, and the user has not correctly guessed the sequence.

2) It must be easy to change the game to any number of slots and any number of possible values per slot and any number of maximum guesses, by simply changing three (static) constants defined at the top of your source code of the appropriate class.
   Hint: To do this, you'll have to use some object to hold the computer's choice of values, and another something to hold the user's guess. Also, you'll need to use well-planned method/loops to compare the guess to the computer's choice and produce the two values that compose a "hint".

3) To help you test and debug your code, put an assert :print statement that show what the computer's choice is, so you know if the hints it produces are correct.
   Example:
   assert true : “this will always print in assert mode\n”;

   Hint: running it in assert mode, will essentially help you debug.
4) Code for all assignments should be well organized and well commented, using javadoc conventions.

   Hint: Eclipse can automatically generate comments for any method, if you program the method and the type /** and hit enter, it will create a minimum javadocs required for the particular method.

5) **EXTRA CREDIT**: After you get it working, incorporate graphics in some way, using colors instead of numbers for slot values. The graphics window can show the user's guesses, and finally the correct answer.
6) **EXTRA CREDIT 2**: learn to use a graphical debugger program and take a snapshot of the screen as you are debugging (include it in the submission and in the README file), along with a short explanation of what you found most annoying and useful about it. At minimum you should know how to step through, step over, and watch expressions in debug mode.

WARNING: we can not be held responsible if you somehow get addicted to playing MasterMind™

You will be graded on the following points:

1. The program compiles and executes successfully.
2. The program contains all of the required programming elements listed in the program specification
3. The program is well organized, as far as object oriented design. That means classes, are well defined and divided by task. It will help you if you create some documentation to sketch out your ideas before starting.
4. A README file is included with a list of all files and their purposes and instructions on how to run your program.
5. The program is well commented, and documented in javadoc.
6. The code is nicely formatted (easy to read visually). For eclipse you can format the code, for emacs there is a format command to format the code.

Submission instructions are on the class website. Please post general questions to the class web board, and learn to use any IDE Debugger.

Hint: Start Early and have fun!!!