CSC172 LAB

JAVA REVIEW

1 Introduction

The labs in CSC172 will follow a pair programming paradigm. Every student is encouraged (but not strictly required) to have a lab partner. Labs will typically have an even number of components. The two partners in a pair programming environment take turns at the keyboard. This paradigm facilitates code improvement through collaborative efforts, and exercises the programmers cognitive ability to understand and discuss concepts fundamental to computer programming. The use of pair programming is optional in CSC172. It is not a requirement. You can learn more about the pair programming paradigm, its history, methods, practical benefits, philosophical underpinnings, and scientific validation at http://en.wikipedia.org/wiki/Pair_programming .

Every student must hand in his own work, but every student must list the name of the lab partner (if any) on all labs.

This lab has six parts. You and your partner(s) should switch off typing each part, as explained by your lab TA. As one person types the lab, the other should be watching over the code and offering suggestions. Each part should be in addition to the previous parts, so do not erase any previous work when you switch.

The textbook should present examples of the code necessary to complete this lab. However, collaboration is allowed. You and your lab partner may discuss the lab with other pairs in the lab. It is acceptable to write code on the white board for the benefit of other lab pairs, but you are not allowed to electronically copy and/or transfer files between groups.

2 Simple Programming Review

The real purpose of this lab is organizational. Practical programming knowledge is a prerequisite for this course. It is also important to get to know your lab TA, get used to the format of the labs, and make sure you know how to hand things in on the blackboard system. So, this first lab is intended mostly to be review.

1. Write a JAVA “helloworld” program with a single main method that prints out your name, your lab session, and the name of the TA that is responsible for your grades.

2. Write a recursive method to calculate the factorial function. Add it to the program above. Include some code that tests your method on the integers 1 -10.
3. Write a method that takes an integer value ***N*** as a parameter and returns an array. The length of the array should be equal to the value of the the parameter. The method should return an integer array of length N filled with random integers in the range. 0 – N. Write a method that takes a reference to an integer array as a parameter and prints the contents of the array to standard output. Add code to your program to demonstrate the code.
4. Write a method that takes a reference to an integer array as a parameter and sorts the array in ascending order. (least to greatest). Do not use the built in Java library swap routines. Rather, write your own method. Demonstrate that your method works by generating a random array, printing it, sorting it, then printing it again.
5. In a separate file, define a class "Student" students have names and graduation years which should be declared as private instance variables variables. Include a constructor and appropriate methods (accessor, mutator, toString). Add code to your program to show your class working.
6. In a separate file define a class "TA" that is a subclass of the "Student" class. TAs are Students with a an hourly pay rate. Add code to your program to show the use of this new class.

3 Hand In

Hand in the source code from this lab 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 README file that includes your contact information, your partner's name, a brief explanation of the lab (A one paragraph synopsis. Include information identifying what class and lab number your files represent.).
2. JAVA source code files representing the work accomplished in this lab. All source code files should contain author and partner 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

See 172/grading.html.

Each section (1-6) accounts for 15% of the lab grade (total 90%)

(README file counts for 10%)