Comp 401 - Assignment 1: Writing a Number Scanner

Date Assigned: Wed Aug 24, 2011
Completion Date: Fri Sep 2, 2011 (midnight)
Early Submission Date: Wed Aug 31, 2011 (midnight)

In this assignment, you will revise your programming skills by doing an assignment involving the use of many of the concepts that are a pre-requisite for this course, which include arrays, loops, and methods (procedures).

Assignment Specification
Write a Java program that expects exactly one string argument to the main method consisting of digit and space characters, converts each digit sequence into a number, and prints these numbers and their sum and product. It gives appropriate error messages if the program arguments are not as expected. You can assume that there:

1. is exactly one space character after each number (including the last one).
2. is at least one number in the argument.
3. there are no errors.

The following interaction illustrates these requirements and results expected in boundary conditions. Thus, if the argument to the main method is:

"2 3 0100"

The output should be of the form:

Numbers: 2 3 100 Sum: 105 Product: 600

You should not pass multiple arguments to the program, as in:

2 3 0100

If you do so, Java will do the scanning for you, and put each string in a separate argument. The goal here is to make you write code that does the scanning.

The scanned string should not be read from the console – it should be a string argument to the main method.

As your string consists of only space and digits, you do not have to worry about negative numbers. Thus, ‘-‘ is an illegal character.

Constraints
1. Java has libraries that make the writing of this program trivial. The only library functions you should use are the Character.isDigit(), substring(), charAt(), length() and the Integer.parseInt() functions. Character.isDigit() is like Character.isUppercase() except that it tells us whether a character is a digit rather than whether it is an uppercase letter. substring(), charAt(), and length(), applicable to any
string, is explained in the class material. `Integer.parseInt()` takes a string consisting of digits and converts into an integer. Thus, `Integer.parseInt("100")` returns the integer 100. It assumes that the number represented by the string is small enough to be stored as `int`. Your assignment should make the same assumption. In this and all other assignments, check with us before you use some Java feature not presented in class.

2. You should store the numbers you scan into an array for later use. This demonstrates that you understand arrays.

3. You should decompose your program into at least two methods, that is, you should not write a monolithic main method. This will demonstrate you ability to write and call methods. The more meaningful methods you write, the more points you will get.

4. Subsequent assignments will teach you to create a multi class program, but for this assignment it is sufficient to create a single class. However, if you feel comfortable doing so, do try and create multiple classes.

5. If we have not specified some aspects of the extra credit or regular work, you are free to choose your own interpretation.

**Debugger User**

It is easy to make mistakes writing a program of this complexity; therefore you will probably need a debugger for this and later assignments. To encourage you to do learn the debugger quickly, we require that you demonstrate the ability to set break points, step into and over a statement, and examine the stack, which are explained in the class material. You must submit screen shots showing you know how to use these features on this programming assignment. These can be similar to the ones shown in the class material. To print a Windows window, put the mouse in the window and press the Alt and PrintScreen buttons simultaneously. This puts the window in the cut/copy buffer. Now you can paste the window into a Word or PPT document.

**Extra Credit**

Allow (a) a number to be succeeded or preceded by a variable number of blanks as in “ 2 456 25 3000 “, (b) an arbitrary number of numbers in a line. Do not terminate the program after encountering the first illegal (unexpected) character. Print the illegal character and continue scanning assuming the character had not been input. How you recover from the errors is up to you – for example, you can break up 123.23 into two legal numbers, 123 and 23, or a single token, 12323.

**Submission Instructions**

The submission instructions for this and future assignments:

1. Submit source code of your class(es), the debugging screen shots, and the screen shots showing executions of the program on test data. In general, you should submit screenshots
that show that you have implemented all the regular and extra credit features in the assignment. These should be sufficient to tell the grader that the program works - they should not have to run the program to verify this. If you do not show evidence that a feature works, we will assume that it does not.

2. Upload the assignment directory, as explained in the instructions linked from the course page.

3. To make it easy for the TAs to find your main class, put it in a package called main, and call it Assignment1. In general, for assignment N, name the main class as AssignmentN, and put it in the main package.

4. 

*Good luck with your first 401 program!*