Java Program Documentation Requirements (Version 1.0)

BE-104, Spring '05, Dr. C. S. Tritt

Part 1 – External (Report)

Full size, professional programs require at least three documents. These are the *Program Design Document*, the *User's Manual* and the *Technical Reference Manual*. In this course, you will combine these documents into a single report, since your programs are relatively short and simple. Double space your report (except for the cover page, tables and the source code section) to give me room to write comments. Your report should contain the following sections:

Cover Page

On this page, list the title of the program, course and assignment numbers, your name(s), my name, due date and submission date.

Introduction

In this section, describes what the program does. The introduction should essentially restate the assigned problem and layout specifications for the work to be done.

Operating Instructions

In this section, explain how to use (not compile!) the program. This section serves as the User's Manual.

Input/Output Description

In this section, describe the input required by the program (if any) and the output it produces. This section should contain a clear statement regarding whether the program operates in batch or interactive mode. If the program is interactive, this section should indicate whether the user interface is graphical or text/console based. It should include descriptions of all values that must be entered during program execution, the format of any input and output data files (not your source files) and the units of all values having units.

Program Design/Algorithm

In this section, list the classes used and how they interact. For each user defined class used, the purpose for class should be described (with emphasis on what it represents), the type and meaning of each member variable should be listed and the function of each method should be described. For methods involving more than one selection or looping construct there should be a prose description plus pseudo-code **or** a flowchart (both are not required, but can be included) of how it works. This section serves as the Design and Technical Reference documents.

Sample Solutions and Test Results

These are solutions to the problem using some method other than the computer program. They are often done by hand, but could include experimental or other results. Sometimes they are solutions for some special cases of the problem that are particularly easy (or at least possible) to solve by hand. These results should be **explicitly compared** to those from the program to demonstrate that the program works. This material is often best presented in tabular format. The number of specific cases done should be sufficient to prove beyond a reasonable doubt that the program works.

Appendices (as appropriate)

Screen Shots

Screen shots of the program in operation with explanatory text.

Source Code Listing

The complete source code (.java) files. Remember to include plenty of comments (see below) in your source code. This section should be in 10 point Courier typeface with 1 inch margins. To avoid wrapping, lines should not exceed 78 characters. In the Eclipse source code editor, use *Window* | *Preferences* | *General* | *Editors* | *Text Editors* | *Show Print Margin* to display an appropriate margin. If your source file contains tabs, you may need to set tab stops at ¼ inch intervals to achieve proper indentation.

Data Listings (if any)

Input and output data files as appropriate. Output data may be in the form of screen output captured from your PC (use Fn PrtSc on most laptops) or pasted into your report document by selecting and copying text from the DOS box (console window) using the dashed selection box icon.

Literature Sources (if any)

Citations (references) to any literature used in the design and development of the program. This could include class handouts, websites (like the Medical Algorithms site, www.medal.org), textbooks, professional journal articles, etc.

Your documentation should be neatly prepared and submitted as a single Word (.doc) document named according to my standard naming convention (generally your MSOE user name followed by the assignment number) and attached to an e-mail message sent to tritt@msoe.edu.

Part 2 - Javadoc

The Javadoc system provides an easy way to generate extensive and detailed online (web based) program documentation. See my “Javadoc Notes” handout for details regarding how to use

Javadoc. Javadoc documentation should be a supplement to, rather than a substitute for, the external documentation described above. Javadoc documentation should be written for other programmers who will be maintaining and modifying your program or reusing your classes.

Javadoc comments should be located:

a) Before each Class statement in your source file(s). Class comments should include a brief description of what the class represents and/or its purpose. These comments should include @author and @version tags.

b) Before each property (member variable) declaration. Property comments should include a brief description of the variable including its units if any.

c) Before each method (member function) definition. Method comments generally start with verbs and describe the purpose or function of the method. These comments should include @param, @return and @throws tags as appropriate.

You generally do not have to submit your javadoc documentation since I can always regenerate it.

Part 3 – Internal (non-Javadoc)

a) File Header – In addition to the standard Javadoc comments your source files should contain the following introductory comments:

```
/*
Filename: (of this source code file)
Last Revised: mm/dd/yy
Course and Assignment Numbers: BE-104 Program z
Instructor: Dr. C. S. Tritt

(A brief description of what this program or class does and how it does
 it. Include information about data file names and formats, program
 limitations, etc. Include literature citations to any particular equations
 or algorithms you use.)
*/
```

b) Other Comments - Your program must contain enough comments to make its operation easy to understand. Nearly every variable should be commented on where it is declared. Other comments should be included as needed to make program design and operation easy to understand.

c) Other Internal Documentation - A logical program design, consistent indentation and mnemonic variable, class and function names go a long way toward creating a well documented a program and will be considered in the grading.