

COMP 123 — INTERNET SERVICES & PROTOCOLS

Kevin Jeffay

Spring 2004

Guidelines for Documenting & Structuring Programs

January 7

Here are some general guidelines for documenting and structuring your programs. With regards to documentation, *note the pervasive use of the word **briefly***. Documentation will be assessed by *quality*; not by *volume*. The primary goal is to make your programs easier for you to write and easier for an outsider to read, use, and modify. Note that these guidelines are very generic (*i.e.*, they are not specific to Java) and are based on guidelines that have been developed for programs in COMP 14, 114, and 121.

For each program:

- Briefly describe the problem and your approach. Outline the overall structure of your program.

For each class:

- Briefly describe the function of, and operations allowed on class types. Give examples of how to use each class type.
- For the implementation of a class, if the class maintains (non-trivial) state across method invocations, give the strongest class invariant you can for the persistent state.

For each non-trivial procedure and function:

- Briefly document the purpose of each procedure and function as well as the function of all parameters. If applicable (non-trivial), describe the algorithm or data structure(s) used in the procedure.
- If the procedure or function uses or modifies global state it should have as meaningful a pre- and postcondition as is possible to state. If you cannot express the desired condition in executable code then express it as best you can in English and insert in your program as a comment.

For each non-trivial loop:

- Each loop should have as meaningful an invariant as is possible to state as well as a brief English description of what the loop is doing.

Some general guidelines of what *not* to do:

- Don't imbed "magic numbers" in your program. Declare them as constants. (For our purposes, a magic number is defined as a number whose role in the program is not immediately obvious in the statement in which it appears.)

- Don't be long winded. In particular, don't assert or comment on properties that are guaranteed by the semantics of the language (unless doing so is desirable for emphasis).
- Don't try to impress the TA or the Professor with overly clever, cryptic, portable, or optimized code.