Programming Documentation Requirements

I. **"External" Documentation (or Program Information):** In programming courses, the comprehensive set of documents that detail the design, development, and structure of a program are usually condensed into a comparatively brief 'block comment' at the top of the source code. This "external" documentation will minimally include:

a. Author(s) name, the course name/number, assignment name/number, instructor's name, and due date.
b. Detailed description of the problem the program was written to solve, including the algorithm used to solve the problem.
c. The program's operational requirements, such as the programming language, special compilation information, and the input information.
d. Required features of the assignment that author(s) were not able to complete, and/or information about the existing bugs.

II. **Documentation about the "Classes":** When writing the code for a `class` in an object–oriented programming language, it should be preceded by a block comment minimally containing the following:

a. The class name, (author(s) name in team projects,) the names of any external packages upon which the class depends, the name of the package for the classes containing this class (if any), and the inheritance information.
b. An explanation of the purpose of the class.
c. Brief descriptions of the class and instance constants and variables.
d. Brief descriptions of constructors as well as the implemented class and instance methods.

III. **"Internal" Documentation (or in-program documentation):** The details of the program are explained by comments and placed within the code. The internal documentation should minimally include the following:

a. A 'block comment' which should be placed at the head of every method (also known as the function or subprogram). This will include the method name; the purpose of the method; the method's pre– and post–conditions; the method's return value (if any); and a list of all parameters, including direction of information transfer (into this method, out from the method back to the calling method, or both), and their purposes.
b. Meaningful identifier names. Traditionally, simple loop variables may have single letter variable names, but all others should be meaningful. Never use nonstandard abbreviations. If the programming language has a naming convention for variables, methods, classes, etc., then those conventions should be used.
c. Each variable and constant must have a brief comment immediately after its declaration that explains its purpose. This applies to all variables, as well as to fields of structure declarations.
d. Complex sections of the program that need some more explanations should have comments just before or embedded in those program sections.

IV. **Miscellaneous / Optional Requirements:**

a. Write programs with appropriate modularity; that is, create classes when appropriate, write methods that accomplish limited, well-defined tasks, etc.
b. Global/public variables should be avoided in programs, unless it is required.
c. Use "white spaces" (blank lines) to set apart logically related sections of code.
d. Indent bodies of methods, loops, and "`if`" statements, and do so with a single, consistent style.

e. Unconditional branching (such as the "`goto`" statement) should be avoided in programs unless it is required for that specific language (such as the assembly language).

Notes. There is a number of standards and tools for program documentation, such as IEEE 1063-2001 "Standard for Software User Documentation" written by IEEE, ISO/IEC 18019-2004 and ISO/IEC TR 9294 written by the International Standards Organization (ISO) and the International Electrotechnical Commission (IEC).

Tools such as Doxygen, `javadoc`, ROBODoc, and TwinText can be used to auto-generate the code documents. Hence, these tools add more capabilities for document preparation. For example, they are able to extract the comments from the source code and create reference manuals in such forms as text or HTML files.

References

1. O. McCann. "Toward Developing Good Programming Style". http://www.cs.arizona.edu/people/mccann/style.html, [accessed today, November 11, 2011]
2. P. DePasquale. http://www.comtor.org/ [accessed today, November 10, 2011]
3. O. Paull, "The Importance of Software Documentation", http://www.ehow.co.uk/about_6706857_importance-software-documentation.html [accessed today, November 11, 2011]
4. Dimitri van Heesch: "Doxygen Documentation. Generate documentation from source code", 2011, http://www.stack.nl/~dimitri/doxygen/ [accessed today, November 11, 2011]