Coding Standard

Table of Contents

Introduction

Our coding standards are designed to simplify, clarify and make it easier to read, write, understand and maintain code.

All classes and methods coded from the previous semester is to stay. If and only if we are modifying a part of last semesters code can modifications be made to the code to comply with our standards.

Comments that might be required to make the developers understanding of what particular methods can only be implemented if a code request ha been submitted and authorized.

Indentation and Layout

Opening and closing brackets are to be on a line to themselves and are to be aligned vertically.

There is to be 2 lines separating each function.

Spacing for expressions, brackets, square brackets and commas.
Please see the table 1.1 below.

Spacing: Token	Space before	Space after
Binary operators (+, -, <, =, etc)	One	One
Unary operators (~)	One	None
((for boolean expr)	One	None
((for parameter lists)	None	None
[	None	None
])	None	One (unless followed by, bracket or ;)
,	None	One

Table 1.1

Names of Identifiers

The names of variables are to be meaning full and describe their purpose. No single character are to be used unless in the use of for loops, while and do while loops.

All Vectors, list(s), arrays are to be declared at the start of the class unless they are only need for the duration of a function. (That is only when they are able to declared at the start of a function).

Constants

Constants are to be declared in uppercase and if a space is required a underscore will be substituted for it.

Variables/ Functions

The first word of the function name is to be in lowercase with internal words capitalized. There are to be no spacing the name declaration.

Variables are to have an char “m” in front of the first letter of the variable declaration name in upper case.

Class naming

The first word of the class name is to be capitalized with internal words also capitalized. There is to be no spacing in the name declaration.

Comments

Comments are to be as clear as possible. One (if not all three) of the following comments are to used for each function depending on circumstances.

Heading comments

Used at the beginning of a class or function or constructor. These comments should be in Javadoc format.

Heading comments are required to have a purpose, date, modified date(s) (if needed), author and short description of the purpose for the class or function.

Example:

```
/*************************************************************
* @purpose: This program converts an infix expression from an input file
* to a postfix expression and evaluates the postfix expression. It is the solution
* for assign 2 for programming unit.
* @author: Johnny Tam
* @date written: July 2004
* @modified: August 2004
*************************************************************/
```

Block comments

Used to describe more than 8-10 lines of code. The code is to be broken into "paragraphs" and the comments are to be on the line above with // starting the comments.

Example:

```
if (Boolean)
{
    //Comments here if needed
    <statements>
}
```

End-of-line comments

Used when defining a parameter at the start of a class or function. They explain the purpose of the parameter. They can also be used at the declaration of variable as well.

Example

```
    void printStuff (
        double buffer[];     //print buffer (in)
        int bufferSize )     //number of items (in)
{
}//end printStuff
```

Functions

A Function is to complete a task or query. It is not to do both.

If the function is returning nothing then the return type must be void.
If the function return type is a int, char, float, boolean etc it is to return a value of the same type.

Constants are to be declared at the beginning of the function. Parameters should be expected to have end-of-line comments and a "//end-of-function" comment at the end of the function as well.

Classes

A class documents class and functions for the user. There is to be no public data members, comments for each function within the class.