Chemistry Learning In Progress

Coding Specifications and Conventions

Nathan Mikeska
Neil Alfredson
Richard Carney
Brian Navarro

Table of Contents

1. Introduction

In order to make the implementation of the CLIP system go as smoothly as possible, some conventions and standards will be applied to the code. This document lays out these conventions and standards for use by the team as they produce code for the CLIP system. The standards set by Sun will be used for the project and can be found it full at the following site: http://java.sun.com/docs/codeconv/. This document highlights some of the more important standards that apply to the CLIP system.

2. Coding Standards

2.1 Naming Conventions

Care will be taken when naming each class, method, and variable to insure that their names make sense and reflect their use.

Classes – Class names should reflect their use or the class that they are extending or implementing. Class names should have the first letter of each word capitalized.

Methods – Method names should reflect the use or purpose of the method. Names should be as descriptive as possible using as few words as possible. The first letter of each word should be capitalized, excluding the first word.

Variables – Variable names should be descriptive and reflect their type. If a variable's use is not obvious by its name, its name should be extended to include its type (Ex. saveMoveCount implies an int type; addBlankTile is vague and should be changed to addBlankTileMenuItem to ensure the programmer knows that it's a MenuItem). Variable names should be as descriptive as possible using as few words as possible. Names should have the first letter of every word capitalized, excluding the first word.

2.2 Variable Declarations

Variables should be declared at the beginning of the methods or blocks in which they are used. Declaring multiple variables on a single line should be avoided. Effort should be made to initialize all variables when declared. In the case of class variables, initialization should occur in the constructor or in methods called by the constructor.

2.3 Statements and Formatting

Multiple statements per line should be avoided. The exception to this would be short lined conditional tests and statements. An example would be: if(success) { return(true); }.

Parenthesis should be used to clear up any possible misunderstandings, particularly in the case of multiple conditions in a conditional statement. Ex. Use if((count > max) | | (count < 0)) instead of if(count > max | | count <= 0)

Braces should be properly aligned as with usual coding standards. Either form below is accepatable:

```
If(done) {
        …
}

If(done)
{
        …
}
```

Braces should also be used even in single line blocks.

```
avoid  if(done)
               return(true);
use    if(done) {
               return(true);
       }
```

2.4 Indentation and Spacing

Logical indentation and spacing shall be used such that the code is organized and easy to read and modify. No effort should be made to 'fight' the indentation practices of the team's chosen compiler (Eclipse). When declaring class variables, they shall be grouped by use or purpose and the groups should be separated blank lines. Methods should be separated by at least three or four blank lines. Code within methods should be grouped by task or purpose with blank lines separating these groups as needed to keep the code clean and readable.

3. Commenting Standards

3.1 JavaDoc Comments

Every class variable and method should be documented using JavaDoc comments. Information on JavaDoc commenting can be found at the following website: http://java.sun.com/j2se/javadoc/index.jsp.

Classes – The documentation of classes should be as complete and thorough as needed to fully explain the purpose and use of the class. JavaDoc links should be given to related classes.

Class Variables - When documenting class variables, effort should be made to give only one sentence while being as descriptive as possible. This keeps the code readable while still providing good documentation. While it should be avoided, it is acceptable to use more than one line if needed, particularly in the event that it is a very important variable that requires more explanation.

Methods – Methods should be fully documented in JavaDoc format. The first sentence of the method comments should be as descriptive as possible about what the method does (a programmer should be able to get a good idea of what the method does by reading the first sentence). The following sentences should fully explain the method in as much detail as needed. Parameters and return values should also be fully documented in JavaDoc style. In addition, any related classes, methods, or class variables should be linked with JavaDoc links.

3.2 Inline Comments

Inline comments should be used when necessary to keep the code organized and understandable. Groups of variable declarations should be headed with an inline comment that describes the group or the relationship that groups them together. The same applies to groups of statements within methods. Also, inline comments should be used to clear up any possible confusion by explaining the code or reasons for the code in any place that the purpose is not obvious.

4. References

Sun's Code Conventions for Java
http://java.sun.com/docs/codeconv/

Sun's JavaDoc Documentation
http://java.sun.com/j2se/javadoc/index.jsp

Version #	Date	Author	Description
1.0	**4/28/06**	**Nathan Mikeska**	**Initial Creation**