Java Style Guide

Lutz Hamel
Dept. of Computer Science and Statistics
University of Rhode Island

January 29, 2003

1 Introduction

Program texts are not only formal instructions for the computer to execute, but they also represent documents that are created, read, and modified by developers. In the case of large software systems program texts are read by many dozens of developers over decades. Therefore, not only is the formal correctness of the program important but also its readability. The following is a set of guidelines which were originally developed by Kernighan and Ritchie in the early seventies for their C programming language and later refined by the group at Sun that invented Java. Guidelines are just that, guidelines, you are free to choose whichever style you like. However, the guidelines here were developed by experienced programmers and they found that the guidelines given here worked well in communicating the intent of a program to other developers. If you invent your own style the most important thing is: be consistent. Once you pick a style stick with it (at least for the program at hand).

2 Class Definitions

Class definitions should start on a new line, left justified. Preferably with white space between the previous line and the line that defines a new class. Something that is easily readable is as follows:

```
class foobar {
       <body of class>
};
```

The eye can easily scan a source file and pick out were a class definition begins and ends. Note the position of the braces that start the body of the class definition.

3 Function Definitions

Similar to class definitions, a function definition should start on a new line with white space surrounding it. However, the definition should be indented according to the class definition surrounding it. One indentation level usually consists of a single TAB character. The function name with its attributes and the argument list should appear on a line by itself with the body of the function defined on the next line.

Example:

```
class example {
	public void func (int arg1, float arg2) {
		<function body>
	};
};
```

4 Structured Statements

Any of the more complex statements that need block structure, such as if statements, for loops, or switch statements, follow the basic pattern of the previous two examples:

```
	...
	if (something_is_true) {
		<then clause>
	}
	else {
		<else clause
	}
	...
```

Of course the statements appear properly indented according to their position in the surrounding program text.

5 Names

In a Java program there are two types of names: type names (names defined by class definitions) and everything else. Type names start with uppercase letters. If the name consists of a concatenation of words then the first letter of each word is capitalized. E.g., RecordSet, MachineLanguage. All other names should start with lowercase letters. If the name consists of a concatenation of words then the first letter of each following word is capitalized. E.g., coolVariable, stringArray, recursiveFunction.

6 Comments and Whitespace

Be liberal with both. Sometimes inserting whitespace between two difficult lines of program text might make the program more readable. Of course you should always add comments explaining problematic areas of your code.

7 Example

Here is an example that applies these few rules and creates very readable program text.

```
import java.io.*

//////////////////////////////
class DataConvert {

       public int convert(byte ch) {
              return ch - ’ch’;
       };
};

//////////////////////////////
class DataStore extends DataConvert {

       public void initial(int a){
              ci = 0;
              size = a;
       };

       void save(int a) {
              store[ci++] = a;
       };

       int setPrint() {
              ci = 0;
              return size;
       };

       int printVal() {
              return store[ci++];
       };

       int sum() {
              int arrsum = 0;

              for (ci = 0; ci < size; ci++) {
                     arrsum = arrsum + store[ci];
              };

              return arrsum;
       };

       // object attributes

       private static int maxSize = 9;
       int size; // size of array
       int ci;   // current index into array
       int[] store = new int[maxSize];
};

//////////////////////////////
class Sample {

    public static void main(String argv[]){
       int sz, j;
       byte[] line = new byte[10];
       DataStore x = new DataStore();

       try {
           while((sz = System.in.read(line)) != 0) {
              int k = x.convert(line[0]);
              x.initial(k);

              for (j = x.setPrint(); j > 0; j--) {
                  System.out.print(x.printVal());
              }

              System.out.print("; SUM = ");
              System.out.println(x.sum());
           }
       }
       catch(Exception e) {
           System.out.println("File error.");
       }
    };
};
```