Java Coding Standard

Introduction

This document contains a set of guidelines, concerning programming style, for writing programs in Java for basic programming courses at the TU/e. Students are strongly advised to follow these guidelines. This document is loosely based on [1, 2]. The following mild coding standard will ensure easily readable and modifiable Java source code. **NB. Be consistent when using the freedom that this standard leaves.**

Rules

Naming conventions

1. Always start your class names with an uppercase letter. If your class name consists of multiple words, add the following words without a separator, starting each word with an uppercase letter. Commonly referred to as **CamelCase**. For example `public class AgeCalculator` or `public class Rectangle`

2. Always start variable- and method names with a lowercase letter. Again, use CamelCase if the name consists of multiple words. For example: `int dateOfBirth` or `public void run()`.

3. A sort of exception to the previous rule: constants are written using uppercase letters and underscores. For example: `final static int MAX_WIDTH = 10`

Indentation

4. Always indent systematically with a fixed multiple of 4 spaces.

5. Never use the TAB characters in source code, because when looking at your code in a different editor, or peach, it might mess up your indentation. *(Tip: Let your editor change a* TAB *to spaces.)*

Line length

6. Never make lines longer than fit in the window on your screen, but never exceed the maximum of 120 characters per line.

One item per line

7. Always write variable declarations, including constants via **static final**, on separate lines.

8. Always write statements and assignments on separate lines.

Empty lines

9. Always write one empty line before and after the following items:

   - A block of constant definitions, a block of instance variables, and a block of local variables.
   - A function declaration, method declaration or class declaration.

Spaces

10. Never write a space before a comma, colon or semicolon, but always put a space after them *(unless at the line end)*.

11. Always write one space before and after:

    - keywords: **`if for while else`** etc.
    - operators: **`= + - * / % += -= == != < > <= >= && ||`** etc.

    There are exceptions to this rule, in particular ++ and – – should be attached to the variable it operates on (e.g. `++x` or `y--`)

Curly braces

12. Always use curly braces after the following statements:

    - **`if`** (...)
    - **`else`**, unless immediately followed by **`if`** for multiway selection (**`else if`**)
    - **`while`** (...)
    - **`for`** (...)
    - **`do`**
    - **`try`** and **`catch`** (...)

13. Always put the opening braces on the same line as the statement it belongs to.

14. Always put the closing braces on a new line and put keywords like **`else`** and **`catch`** on the same line as these closing braces.

Comments

15. Always explain each variable declaration in a comment.

16. Always put a block comment before a function or method that explains its functionality.

Notes

Well-organized source code is important for several reasons:

- The compiler may not care about this, but source code is also read by others: developers, reviewers, maintainers, teachers, graders, ...
- It is an important means to prevent errors.
- It facilitates localization of errors, both by the author, and by others.

Example

This section provides some small examples regarding the rules proposed in the previous section.

The if else-if else statement:

```
if (condition1) {
    statements;
} else if (condition2) {
    statements;
} else {
    statements;
}
```

Methods and functions:

```
/**
* This method counts from one up to and including ten
*/
public void countFromOneUntilTen() {
    int count = 0; // starting from zero
    int max = 10; // the maximum amount of iterations

    while (count < max) {
        count = count + 1;
        System.out.println(count);
    }
}
```

References

[1] Code Conventions for the Java Programming Language. Sun, 1999.

[2] Java Coding Standards. ESA, 2005.