CONVENTIONS

DISCLAIMER: I'm going to be FUCKING ANNOYING with that. Enjoy :)

First of all: This document is in English. Why? Because we're programmers, we're going to code in English. Then, if something isn't clear, don't assume. Ask. If you do something wrong, you're the one who's going to lose time!

Now, concerning global rules:

- English comments and variable/method/class names
- camelCase and classic Java conventions

COMMENTS

Préférez les blocs de commentaires plutôt que les commentaires simples. Mettez un **espace** avant chaque commentaire !

Ne mettez des commentaires que si c'est utile !!! Et si vous avez besoin de mettre un commentaire, demandez vous d'abord s'il n'y a pas moyen de simplifier l'implémentation !!!

```
/* A NE PAS FAIRE : */

   // Variable pour la longueur du tableau
   int x;

/* AU LIEU (PAS BESOIN DE COMMENTAIRES !!!) : */

   int longueurTableau;
```

Éviter de "casser" les blocs conditionnels pour commenter chaque partie. Commentez en bloc au début de la condition. Exception : Si le bloc conditionnel est trop grand.

Exemple :

```
/*
 * Commentaire du bloc conditionnel
 */
if (condition) {
    // Do something
} else if (condition2) {
    // Do something else
} else {
    // Last option...
}

MAIS PAS :
// comment
if (condition) {
    // Do something
}

// comment
else if (condition2) {
    // Do something else
}

// comment
else {
    // Last option
}
```

Documentation comments :

```
/**
 * Before a class or a function. Say what is the class/method for. This
documentation is for
 * users of the class, not for the people maintaning it!!!
 *
 * Use documentation keywords as "@authors", "@param", "@return", ...
 */
```

Implementation comments :

```
/*
 * Comments on the behaviour of the method and the logic. It is for the devs, to help for further
 * modifications and maintaning the code.
 */
```

Special comments :

- `// TODO`
- `// XXX`
- `// FIXME`

UseXXX in a comment to flag something that is bogus but works. Use FIXME to flag something that is bogus and broken.

CODING

The first non-comment line of most Java source files is a **package** statement. After that, **import** statements can follow. For example:

```
package java.awt;

import java.awt.peer.CanvasPeer;
```

10.5.2 Returning Values

Try to make the structure of your program match the intent. Example:

```
if (booleanExpression) {
    return TRUE;
} else {
    return FALSE;
}
```

should instead be written as

```
return booleanExpression;
```

Similarly,

```
if (condition) {
    return x;
}
return y;
```

should be written as

```
return (condition ? x : y);
```

- Chaque déclaration de variable prend une ligne. Pas de déclarations multiples sur la même ligne !!!
- A part pour les méthodes, tous les blocs de code (boucles, conditions, etc) ont un espace **entre le mot clé et les parenthèses** et **avant les accolades**. Exemple :
    - `public void method(int param) {}`
    - `if (condition) {}`
- Pas de constantes en dur ! Excepté pour -1, 0 et 1.

EXEMPLE COMPLET

```java
package ch.heigvd.res.lab01.impl;

import java.util.logging.Logger;

/**
 * Commentaires sur la classe : Ce qu'elle fournit, à quoi elle
 * sert, etc.
 *
 * @author Paul Brocantard
 */
public class Utils {
    /*
     * Commentaires sur l'implémentation de la classe. Utiles
     * pour les dev qui vont devoir faire des modifications.
     */

    /** Commentaire formel sur la variable */
    public static int var1 = 10;

    /**
     * Commentaire formel sur variable
     * Un peu plus long
     */
    public Object obj = new Object();

    /**
     * Documentation of the method.
     *
     * @param param1 Comments about param1
     * @param param2 Comments about param2
     */
    public static void doSomething(int param1, String param2) {
        /*
         * Comments on the logic and implementation of the method
         */

        if (param1 > 0) {
            param2 = "It's positive!";
        } else if (param1 < 0) {
            param2 = "It's negative :(";
        } else {
            param2 = "OMG it's zero D:";
        }
```

```
        System.out.println("param2");
    }

    // If we didn't find any separator, meaning "separator" is empty, we set "lines"
in the second part of the result.
    // Otherwise, we construct the result array using the index of the separator
found.
    if (separator.isEmpty()) {
      result(1) = lines;
    } else {
      // First part containing the first line and the separator
      result(0) = lines.substring(0, indexFirstSeparator + separator.length());

      // Second part containing the remaining text.
      result(1) = lines.substring(indexFirstSeparator + separator.length());
    }

    return result;
  }
}
```