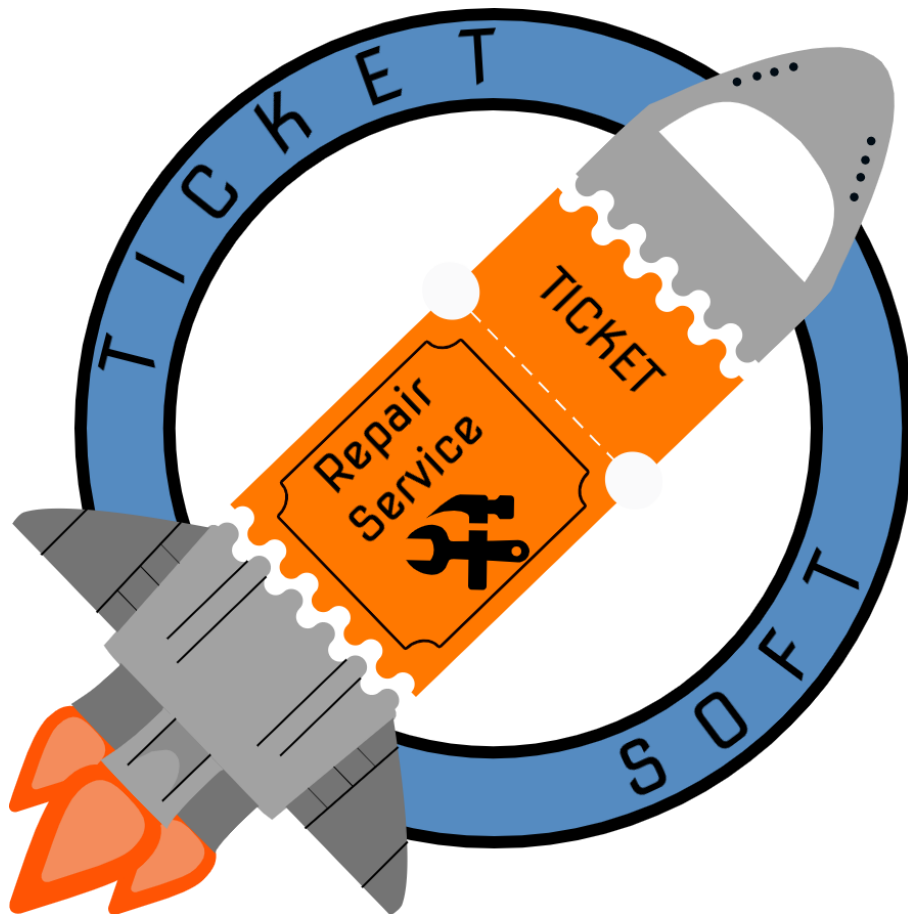


Convention de Codage Projet / version 1.0

TicketSoft

Convention de Codage Projet



Projet Génie Logiciel

2019-2020 (durée 6 mois)

Commanditaires

Valerie Guimard (valerie.guimard@u-psud.fr)

Frédéric Voisin (fv@lri.fr)

Rédacteur(s)

Thomas von Ascheberg

Approuvé par Thomas von Ascheberg

Convention de Codage Projet / version 1.0

Sommaire

Introduction.....	3
Général	4
Nommage	4
JavaScript (JS)	5
Java	6



Convention de Codage Projet / version 1.0

Introduction

Ce document s'adresse à tous les développeurs du projet. On va ici détailler le style et le format de code à respecter pour que le projet ait une certaine cohérence. On demandera à tous les programmeurs de faire l'effort de respecter ces codes au maximum pour simplifier la relecture du code et faciliter la compréhension par les autres membres du groupe du code.

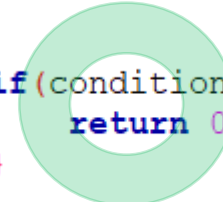
Aussi, Il est fortement recommandé de coder sous Windows 10 avec Eclipse pour la partie Java pour éviter tout problème de compatibilité lié à chaque plate-forme.



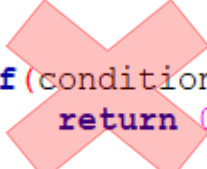
Convention de Codage Projet / version 1.0

Général

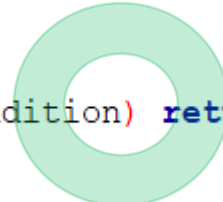
- Tout le code sera écrit en anglais (code et commentaires)
- Essayez de limiter la longueur des lignes de code à MAXIMUM 150 caractères
- Le style de la tabulation est de 4 espaces
- La documentation du code (JS et Java) se fait avec la syntaxe Javadoc. Dans un cas c'est compilé par JSDoc et l'autre par Javadoc standard.
- N'utilisez que des commentaires sur une ligne `//` et pas de commentaires multilignes `/* ... */` (La seule exception est `/** ... */` pour faire la Javadoc de vos fonctions)
- Commentez votre code !
- Toutes les accolades doivent apparaître. Il est interdit de faire des conditionnelles ou des boucles sans les accolades.
(Est tolérée la syntaxe raccourcie du `if` pour les expressions courtes)



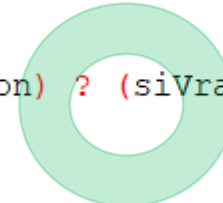
```
if(condition){  
    return 0;  
}
```



```
if(condition)  
    return 0;
```



```
if(condition) return 0;
```



```
(Condition) ? (siVrai) : (siFaux)
```

Nommage

- Tous les noms seront donnés en anglais (pas de nom de variable/fonction en français)
- On utilise la convention de nommage lower CamelCase pour les variables
(https://fr.wikipedia.org/wiki/Camel_case)
- On utilise la convention de nommage upper CamelCase pour les Classes et les Enum
- Les constantes doivent être écrites en MAJUSCULES. Si le nom est composé de plusieurs mots, on utilisera `_` pour séparer ces mots

```
double PI = 3.14;  
int MAX_SIZE = 10;
```



Convention de Codage Projet / version 1.0

JavaScript (JS)

- On utilisera la directive "**use strict**" au début du code pour assurer une certaine qualité du code.
- Utilisez au maximum **let** plutôt que **var**. Le but est de rendre votre code le plus local possible pour éviter qu'il interfère avec le code des autres (par exemple en redéfinissant une variable qui existe ailleurs).
- Utilisez JQuery avec parcimonie ! La syntaxe à base de \$ de JQuery rend le code assez difficile à lire et à debug. Il est souvent plus clair (et aussi rapide) d'utiliser les outils vanilla de manipulation du DOM tels que `document.getElementById(...)` ou `document.getElementsByClassName(...)`.



Convention de Codage Projet / version 1.0

Java

- Les tokens de visibilité doivent être dans l'ordre private, protected, public. Un token non utilisé n'est pas obligé d'apparaître.
- Pour chacun des champs (privé, protégé et public), les champs doivent apparaître dans l'ordre suivant :
 - Les attributs
 - Le(s) constructeur(s)
 - Getter puis Setter associé (s'ils sont là)
 - Les méthodes de la classe (pour les méthodes héritées à redéfinir, on garde l'ordre de la classe mère)
 - Les surcharges d'opérateurs
 - Un main (s'il en existe un pour debug)
- On met toujours les attributs et méthodes statiques avant les non-statiques.

```
/**
 * Sub class of Parent class
 * Doing stuff...
 */
public class SubClass extends ParentClass {

    private int price;
    private String someString;

    public SubClass() {
        this.price = 0;
        this.someString = "";
    }

    public int myMethod1(int arg0) {
        //doing stuff...
        return 0;
    }

    public static void main(String[] args) {
        //...
    }
}
```

- Si l'on souhaite qu'une méthode ne puisse pas être redéfinie, on ajoute le mot-clef **final** juste avant le point-virgule.
- Lorsqu'une classe est finie et testée, il est impératif de faire sa documentation (JavaDoc). Cela inclut la JavaDoc des méthodes

