TACT factory
mobile agency

GUIDE

CONVENTION DE NOMMAGE

Le but de ce document est de présenter les conventions de nommage qui sont utilisées pour le projet QCM

AVANT-PROPOS

Source

Ce document a été rédigé en majorité à partir des conventions officielles de nommage du langage de programmation Java.

Voici la source de référence :

- “Code Conventions for the Java Programming Language”, Sun
- http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

Table des matières

TABLE DES MATIERES

Formatage du code

TAILLE DES SOURCES

Il est recommandé de ne pas excéder 2000 lignes dans un fichier .java et de ne pas dépasser les 110 colonnes.

INDENTATION

Tabulation

Il ne faut pas utiliser le caractère de tabulation car son interprétation varie selon les éditeurs. Configurez votre éditeur pour que la tabulation écrive 8 caractères espace.

Blocs

L'entrée dans un nouveau bloc fils impose le rajout d'un niveau d'indentation. Deux blocs de même niveau doivent débuter sur la même colonne (même niveau d'indentation).

Taille des lignes

Une ligne ne doit pas excéder 110 colonnes.

Retour de ligne

Le retour à la ligne se fait à la suite d'une virgule séparant différents paramètres d'une méthode, où à la précédence d'un opérateur par exemple, et à l'extérieur de toute parenthèse, de préférence.

On revient à la ligne dans les conditions suivantes :

- Après une virgule
- Avant un opérateur
- De préférence au sein d'une parenthèse de haut niveau, plutôt que de petit niveau
- Le niveau d'indentation doit être le même que celui de la ligne précédente, sauf si cela peut prêter à confusion, à ce moment-là, on augmente le niveau d'indentation.

```
53    setMyVarOfMyClass(myVeryLongVar1, MyVeryLongExpression1, myVeryLongVar2,
54    myLonguestExpression2, myLonguestVariable4);
55
56    if (!(condition1 || condition2)
57            && (condition3 || condition4)
58            && (condition5 || !condition6)) {
59        doSomethingAboutIt();
60    }
```

FORMATAGE DU CODE

Lignes blanches

- Les lignes blanches doivent être utilisées pour séparer les méthodes, et des portions de code distinctes.

Nommage

METHODE « CamelCase »

Chaque première lettre d'un mot prend une majuscule, que tous les mots sont collés les uns aux autres et petite subtilité, le premier mot ne prend pas de majuscule.

PACKAGE

Le nom d'un package doit respecter les conventions suivantes :

- **Tout en minuscule.**
- **Utiliser seulement [a-z], [0-9] et le point '.'**
- Ne pas utiliser de tiret '-', d'Under score '_', d'espace, ou d'autres caractères ($, *, accents, ...).
- La convention de Sun indique que tout package doit avoir comme root un des packages suivant : com, edu, gov, mil, net, org ou les deux lettres identifiants un pays (ISO Standard 3166, 1981).

Les noms doivent de préférence être le nom de l'entreprise, du département, du projet.

Exemple :

```
1   package fr.tactfactory.qcm;
2
```

CLASSE ET INTERFACE

- Le nom des classes et interface doit être en minuscule, hormis les initiales des mots le composant.
- La première lettre de chaque mot doit être en majuscule.
- Il faut aussi éviter les abréviations et essayer d'utiliser des mots qui décrivent bien la classe tout en étant pas trop long.

Exemple :

```
3   class MaFavoritClass;
```

METHODE

- Les noms des méthodes doivent être en minuscule hormis les initiales des mots le composant (sauf le premier).

Exemple :

```
5   public void myFavoritMethod() {}
```

- Les accesseurs directs (getters et setters) des attributs d'une classe doivent être préfixés d'un get pour la lecture et d'un set pour l'écriture. Le suffixe doit être le nom de l'attribut.

Exemple :

```
7   public int getLevel() {
8   public void setLevel(int level) {
```

- Le préfixe **is** doit être utilisé par les méthodes retournant un booléen.
- Certains autres préfixes particuliers doivent être utilisés :
    - **compute**, pour le calcul
    - **find** pour la recherche
    - **init** pour l'initialisation
    - **delete** pour la suppression
    - **add** pour l'ajout
    - **close** pour la fermeture d'objets

Exemple :

```
9   public boolean isVisible() {
```

ATTRIBUT, VARIABLE ET PARAMETRE

- Les attributs de classe, les variables locales comme globales ainsi que les paramètres des méthodes doivent être en minuscule hormis les initiales des mots le composant (sauf le premier).
- Le nom des variables doit être le plus explicite possible sur son contenu et être très court, à l'exception des variables temporaires
- Le dollars ($) et le soulignement (_) sont proscrits.
- Les collections d'objets doivent être nommées au pluriel.

Exemple :

```
12  for (i = 0; i < 10; i++) {
13  Question nextQuestion = questions.get(this.id + 1);
14  float myWidth = 145.5;
15  Collection Questions;
```

NOMMAGE

CONSTANTE

- Les noms des constantes doivent être entièrement en majuscule. Le séparateur de mot est le caractère de soulignement (underscore : « _ »).

Exemple :

```
18    static final int PROMPT_LOG = 1;
```

Les commentaires

BLOC DE COMMENTAIRE

- Commenter une méthode, une classe, un attribut à l'aide d'un bloc de commentaire

Exemple :

```
19
20  /*
21   * La classe MyClass fournis telles fonctionnalités…
22   */
23  public class MyClass() {
```

(Avec paramètre entrant et valeur de retour)

```
35  /**
36   * Méthode pour vérifier qu'un temps donné est compris entre le temps de début
37   * et le temps de fin.
38   * @param time le temps avec lequel comparé les temps
39   * @return true si le temps est compris entre les temps, false sinon
40   */
41  public boolean isCoveredTime(double time) {
42      boolean isCoveredTime = false;
43      if (time >= startTime && time <= endTime) {
44          isCoveredTime = true;
45      }
46      return isCoveredTime;
47  }
```

COMMENTAIRE MONOLIGNE

- Insertion d'un commentaire afin d'expliquer le comportement du code

Exemple :

```
49    //Crée le sous-titre
50    sub = new FXSubtitle(text, sTime, eTime);
```

Declarations

VARIABLES

Indentation

- Les variables doivent de préférence être déclarées lignes par lignes.
- Sauf lorsqu'il s'agit de variables temporaires itératives pour lesquelles une déclaration globale sur une seule et même ligne est recommandée.

Exemple :

```
62   int level;
63   String libelle;
64
65   int i, j, k;
```

Emplacement

- Les variables doivent être déclarées au plus tôt, sitôt après l'accolade ouvrante du bloc. Et non pas juste avant leur utilisation dans le code.

METHODES

- Les noms des méthodes sont accolés à la parenthèse ouvrante listant les paramètres. Aucun espace ne doit y être inséré.

Exemple :

```
67   void myMethod() {
```

BLOCS

- Tout bloc est délimité par des accolades.
- L'accolade ouvrante doit être placée en fin de ligne, à la suite d'un espace, après l'instruction/méthode/classe créant le bloc.
- L'accolade fermante doit être placée en début d'une ligne vierge à la suite de la dernière instruction du bloc

Exemple :

```
69   void myMethod() {
70     int level;
71     ...
72   }
```

DECLARATIONS

ORDRE

L'ordre de déclaration des entités du code source doit être le suivant :

- Les attributs de la classe :
  - En premier les statiques (**static**)
  - En deuxième les publiques (**public**)
  - Ensuite les protégés (**protected**)
  - Enfin les privés (**private**)
- Les méthodes
  - En premier les statiques
  - En deuxième les publiques
  - Ensuite les protégées
  - Enfin les privées

Structure de contrôle

BOUCLE FOR

Syntaxe générale :

```
74  for (initialisation; condition; modification) {
75      instructions;
76  }
```

BOUCLE WHILE

Syntaxes générales

```
78  while (condition) {
79      instructions;
80  }
```

BOUCLE DO

Syntaxe générale

```
82  do {
83      instructions;
84  } while (condition);
```

CONDITION IF

Syntaxes générales

```
86  if (condition) {
87      instructions;
88  }
89  if (condition) {
90      instructions;
91  } else {
92      instructions;
93  }
94  if (condition) {
95      instructions;
96  } else if (condition) {
97      instructions;
98  } else {
99      instructions;
100 }
```

CONDITION SWITCH

Syntaxe générale

STRUCTURE DE CONTROLE

```
102  switch (condition) {
103      case valeur1:
104          instructions;
105          // passe à travers
106
107      case valeur2:
108          instructions;
109          break;
110
111      case valeur3:
112          instructions;
113          break;
114
115      default:
116          instructions;
117      break;
118  }
```

Coordonnées

CONTACT

YOAN PINTAS
CHEF DE PROJETS

Tél 01.83.64.25.33
contact@tactfactory.com

ENTREPRISE

TACTfactory
13 rue de l'union, 72000, Le Mans, France
Tél 01.83.64.25.33
www.tactfactory.com