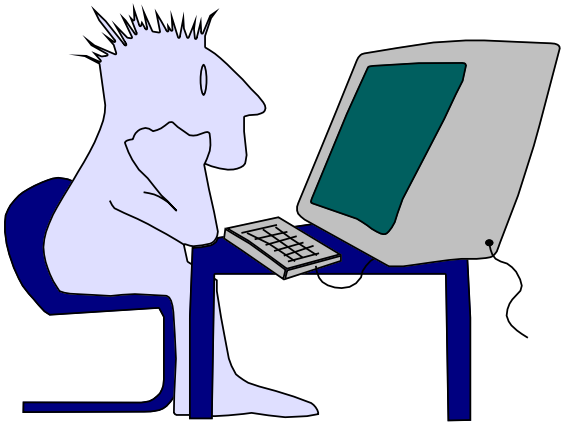
La présentation dans les programmes Java



Introduction

La nécessité de respecter des règles de présentation dans l'écriture des programmes est apparue très tôt dans l'histoire de la programmation.. Elle a été particulièrement bien formalisée par Sun dans son document CodeConventions.pdf. Le présent document est, pour l'essentiel, une traduction/résumé du document de Sun.

Présentation des instructions

Utilisation des blancs pour augmenter la clarté:

Les opérateurs binaires sont précédés et suivis d'un espace.
Les arguments d'une fonction sont séparés par au moins un espace après la virgule.

Une seule instruction par ligne

sauf dans le cas d'instructions simples, courtes et corrélées.

Alignement des groupes d'assignations liées

Exemple 1: Alignez les assignations liées:

```
nomEmploye    = nomSaisi;
prenomEmploye = prenomSaisi;
```

Exemple 2: mais désalignez les assignations non liées:

```
nomEmploye    = nomSaisi;
prenomEmploye = prenomSaisi;
ageCapitaine = racineCarree(marqueVelo);
```

Longueur des lignes:

Il est en général préférable de limiter les lignes à 80 caractères. En effet, les lignes de plus de 80 caractères ne tiennent généralement pas dans la largeur de l'écran ni dans celle d'une feuille 21 X 29,7

Les lignes de suite:

Lorsqu'une instruction nécessite plusieurs lignes,

- Gardez groupés les éléments liés.

Ainsi :

```
while ( ( nomduChemin [ debutChemin + positionCourante ] <> ';' )
    &&    ( debutChemin + positionCourante <> longueur [ nomduChemin ] ) )
 ...
```

est meilleur que:

```
while ( ( nomduChemin [ debutChemin +
        positionCourante ] <> ';' ) && ( debutChemin
        + positionCourante <> longueur [ NomduChemin ] ) )
 ...
```

- Indentez la deuxième ligne par rapport à la première et alignez les suivantes sur la deuxième.

- Utilisez une indentation claire:

Exemple 1: Indentation standard (+4):

```
while ( ( nomduChemin [ debutChemin + positionCourante ] <> ';' )
    &&    ( debutChemin + positionCourante <> longueur [ nomduChemin ] ) )
 ...
```

Exemple 2: Indentation sur l'opérateur d'assignation:

```
totalVentes =  ventesDirectes
             + ventesparDistributeur
             + ventesparCorrespondance;
```

Exemple 3: Arguments d'appel d'une fonction:

```
chiffreAffaires = sommePonderee (   nombreSavonnettes, prixSavonnette,
                                    nombreTubesDentif, prixTubesDentif,
                                    nombreBainsMouss,  prixBainsMouss);
```

Présentation des déclarations de données

- Alignez les déclarations de données
Comme pour les assignations
- Ecrivez une seule déclaration par ligne
- Ordonnez les déclarations de façon logique

L'indentation

Principes généraux:

```
entete d'instruction
    corps d'instruction
```

ou

```
entete d'instruction
{
    instruction
    ...
    instruction
}
```

L'indentation est placée entre l'accolade et les instructions du bloc.

L'indentation est faite sur la base de tabulations 1, 5, 9, 13, 17, etc...

L'instruction if

Le else est toujours aligné sur le if

L'instruction do while

Le while est toujours aligné sur le do

L'instruction switch

```
switch (...)
{
    case ... :
        instruction;
        instruction;
    case ... :
        instruction;
        instruction;
    ...
    default ... :
        instruction;
        instruction;
}
```

Tolérances

Lorsque les blocs sont très courts, on peut simplifier les règles.
Exemples:

```
if ( a < 0 ) a = -a;
```

```
for ( i = 0; i <= 9; i++) { ta[i] = 0; tb[i] = 1; }
```

```
switch (a)
{
    case 1 : x = 'A'; break;
    case 2 : x = 'B'; break;
    case 3 : x = 'C'; break;
    default: x = 'Z';
}
```

Variante

Un usage assez répandu consiste à mettre l'accolade ouvrante sur la ligne de l'entête d'instruction.
Exemples:

```
entete d'instruction {
    instruction
    ...
    instruction
}
```

L'organisation des fichiers sources

- Séparez visiblement les classes d'un fichier source
par au moins deux lignes vierges
- Séparez visiblement les fonctions membres d'une classe
par au moins une ligne vierge

Les commentaires

Ils sont destinés :

- au développeur lui-même et à son équipe pour faciliter la relecture et la maintenance du code.
- Aux développeurs d'application qui auront à utiliser les classes concernées.

Dans le monde Java, l'existence de l'outil ***javadoc*** et les conventions proposées par Sun gouvernent la forme des commentaires de classes, de champs et de méthodes (voir le chapitre javadoc)

Le programmeur pourra ajouter des commentaires relatifs au code.

Les commentaires relatifs au code:

- Indentez les commentaires avec le code qu'ils accompagnent

Commentaires de fin de ligne:

En langage évolué, les commentaires relatifs à une seule ligne sont rarement utiles. A ceci, deux exceptions:
- les commentaires en fin de ligne pour annoter les déclarations de données
- les notes de maintenance

Les noms des identificateurs

Les identificateurs doivent avoir des noms suffisamment descriptifs pour permettre une bonne lisibilité du programme.
Un nom du genre `tauxTVA` devra toujours être préféré à `x1`, pas assez explicite

- Le nom choisi devra être constitué d'une suite de mots choisis dans la langue courante ou dans les abréviations claires.
- Le premier mot d'un identificateur de donnée ou de classe sera un substantif
Exemples:
```
revenuMensuel,temperatureVapeur, niveauMax
```
- Le premier mot d'une méthode sera un nom ou plus généralement un verbe.
Exemples:
```
getTemperature()
setSeuilNiveau(int niveau)
afficheBilan()
calculIRPP()
```

- Dans le cas d'un identificateur de 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'underscore '_', 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 règles 1 et 2 doivent être suivies sans concessions. De toute manière, la plupart de ces règles sont obligatoires.

Exemple :

```
com.sun.eng
com.apple.quicktime.v2
edu.cmu.cs.bovik.cheese
```

- Dans le cas d'un identificateur de classe tous les mots (y compris le premier) commenceront par une majuscule, les lettres suivantes étant des minuscules

- Dans le cas des Interfaces, les mêmes conventions que les noms de classes s'appliquent.

- Dans le cas cas d'un identificateur de variable ou de méthode, le premier mot commencera par une minuscule et les autres mots par une majuscule. Ne jamais commencer les noms avec '$' ou '_' bien que ce soit possible.

Exemples:
```
revenuMensuel,temperatureVapeur, niveauMax
```

- Dans le cas cas d'un identificateur de constante,toutes les lettres sont en majuscule et les mots sont séparés par un trait de soulignement.
Exemples:
```
MAX_HEIGHT,TAILLE_MINI
```

- On pourra ajouter un préfixe de 1 à 4 lettres descriptif de type. Ce préfixe sera constitué de caractères minuscules.
Exemples:
```
fTauxTVA, bAlarmeDeclenchee, nAgeCapitaine, strNomPays
avec
f     pour  flottant
b     pour  booleen
n     pour  entier
str   pour  chaine de caractères
...
```

- Toutefois, les variables de portée très locale pourront avoir des noms simplifiés.
Exemple: i pour un indice de boucle

Pour plus de détails, voir le site suivant qui snntetise bien l'ensemble des conventions d'ecriture java à garder tout au long de votre programme : http://www.loribel.com/java/normes/nommage.html

javadoc

Sun propose un outil de génération automatique de documentation : javadoc
Cet outil est un parser (analyseur) qui lit les sources java, détecte automatiquement les commentaires de documentation et génère la documentation sous forme html

Les commentaires de documentations commencent toujours par `/**` et se terminent par `*/`
Ils précèdent toujours l'entité qu'ils décrivent.
Les entités décrites par les commentaires de documentation sont les classes, les variables et les méthodes.

Les commentaires de documentation peuvent contenir du texte, des balises html et des balises javadoc.Parmi les balises javadoc : @author @version @param @return

Exemple :

```
/**
 * La classe <code>Ex003</code> permet le traitement de chaines de caractères.
 * @version 1.0 06/05/2004
 * @author Luc Vannier
 */
public class Ex003
{
/**
 * Compte le nombre d'occurences d'un sous-chaine dans une chaine de
caractères.
 * <br>La recherche de sous chaine se fait en "fenêtre glissante".
 * <br>Par exemple dans la chaine "azaza", la sous-chaine "aza"
 * sera détectée 2 fois
 * @param   sch   La sous-chaine
 * @param   ch    La chaine
 * @return le nombre d'occurences
 */
      public static int getNbOccurences(String sch, String ch)
      {
            int nbo = 0;      // Nombre d'occurences
            for (int i=0; i < ch.length(); i++)
            {
                  if ( i + sch.length() > ch.length() ) break;
                  String s = ch.substring(i, i + sch.length());
                  if (s.compareTo(sch) == 0) nbo++;
            }
            return nbo;
      }

/**
 * Acquiert une chaine de caractères et une sous-chaine et affiche
 * le nombre d'occurences de la sous-chaine dans la chaine
 */
      public static void main(String[] args)
      {
            String souschaine;
            String chaine;
            System.out.print("Entrez la chaine:");
            chaine = Console.lireStr();
            System.out.print("Entrez la sous-chaine:");
            souschaine = Console.lireStr();
            int nbOccurences = getNbOccurences(souschaine, chaine);
            System.out.println("Le nombre d'occurences de " + souschaine
                  + " dans " + chaine + " = " + nbOccurences);
      }
}
```

Utilisation de javadoc depuis une fenêtre d'invite de commande

```
> javadoc *.java –d doc
```

Cette commande parse tous les fichiers d'extension .java dans le répertoire courant et génère les pages de documentation qui seront rangés dans le répertoire doc

Remarque : L'utilisation de la comande javadoc suppose que la variable d'environnement path contienne le chemin du répertoire bin du jdk.

Utilisation de javadoc depuis Eclipse

Pour ajouter un outil « javadoc » dans votre IDE Eclipse, je vous invite à voir le lien suivant qqui explique en bien la démarche à suivre : http://www.objis.com/formation-java/tutoriel-java-creation-javadoc-eclipse.html