[Formatage du Code 1](#_Toc1314267971)

[Indentation : 1](#_Toc851352566)

[Taille des Lignes : 1](#_Toc862022176)

[Lignes Blanches : 1](#_Toc1815887555)

[Espaces : 2](#_Toc87115153)

[Déclarations de Classe avec Extends et Implements : 2](#_Toc749612276)

[Résumé - Formatage du code : 3](#_Toc82605062)

[Nommage 3](#_Toc538691774)

[Général : 3](#_Toc1565505243)

[Packages : 3](#_Toc119794660)

[Classes et Interfaces : 4](#_Toc2045945408)

[Résumé - Nommage : 4](#_Toc540087158)

[Méthodes 5](#_Toc643265309)

[Nommage : 5](#_Toc1496583716)

[Déclaration : 5](#_Toc1425594688)

[Instructions Return : 6](#_Toc544827642)

[Résumé - Métohdes : 6](#_Toc216504173)

[Variables 7](#_Toc1119230072)

[Nommage : 7](#_Toc983860814)

[Déclaration : 7](#_Toc1064919368)

[Constantes : 7](#_Toc1374006445)

[Types de Variables : 7](#_Toc483597457)

[Résumé - Variable : 8](#_Toc1932290336)

[Blocs de Code 8](#_Toc154147727)

[Structures Conditionnelles : 8](#_Toc1309170496)

[Boucles : 9](#_Toc514852205)

[Résumé - Blocs de code : 9](#_Toc770819090)

[Commentaires 10](#_Toc1052516481)

[Documentation : 10](#_Toc1599320815)

[JavaDoc : 10](#_Toc471405512)

[Autres Conventions Spécifiques 10](#_Toc1414518603)

[Saut de Ligne : 10](#_Toc641942443)

[Caractères Spéciaux : 11](#_Toc1953459107)

**Conventions de Codage Java pour le Projet**

# **Formatage du Code**

## **Indentation :**

* Utilisez des espaces plutôt que des tabulations pour l'indentation. Configurez votre éditeur pour remplacer chaque tabulation par 4 espaces.

Exemple :

**if** (condition) {

// Ce bloc est indenté avec 4 espaces

executerTache();

}

* Chaque nouveau bloc de code (par exemple, à l'intérieur d'une instruction **if** ou **for**) doit augmenter le niveau d'indentation.

## **Taille des Lignes :**

* Limitez la longueur des lignes de code à 80 colonnes, y compris les espaces d'indentation, pour garantir la lisibilité sur différents écrans.

Exemple :

// Cette ligne ne dépasse pas 80 colonnes

String message = "Assurez-vous que la longueur de cette ligne est inférieure à 80 caractères";

## **Lignes Blanches :**

* Utilisez des lignes blanches pour séparer logiquement différentes sections du code, telles que les méthodes ou les blocs de code au sein d'une méthode.

## **Espaces :**

* Ajoutez des espaces autour des opérateurs (par exemple, **a + b** plutôt que **a+b**)
* Après les virgules,
* Après les mots réservés du langage.
* N'utilisez pas d'espace immédiatement à l'intérieur des parenthèses.
* N’utilisez pas d’espaces après le ‘ ;’ après les lignes de codes.

Exemple :

int sum = a + b; // Espace autour de l'opérateur '+'

for (int i = 0; i < n; i++) { // Espaces autour des opérateurs '<' et '='

## **Déclarations de Classe avec Extends et Implements :**

* Lorsqu'une classe étend une autre classe ou implémente des interfaces, placez chaque mot-clé (**extends**, **implements**) sur une nouvelle ligne pour une meilleure lisibilité.

Exemple :

public class MyArrayList

extends ArrayList

implements Serializable {

// Corps de la classe

}

## **Résumé - Formatage du code :**

|  |  |  |
| --- | --- | --- |
| **Sujet** | **À faire** | **À ne pas faire** |
| **Espaces (Opérateurs et Syntaxe)** | int sum = a + b;  for (int i = 0; i < n; i++) {  // corps de la classe  } | int sum = a+b;  for (int i=0;i<n;i++) {  // corps de la classe  } |
| **Espaces (Retour à la ligne)** | public int somme(int a, int b){  int sum = a + b;  return sum;  } | public int somme(int a, int b) {  int sum = a + b;  return sum;  } |
| **Déclarations de Classe** | public class MyArrayList  extends ArrayList  Implements Serializable {  // Corps de la classe  } | public class MyArrayList extends ArrayList Implements Serializable {  // Corps de la classe  } |

# **Nommage**

## **Général :**

* Les identifiants doivent être explicites. Évitez les noms trop longs tout en assurant qu'ils soient descriptifs.

## **Packages :**

* Utilisez des noms en minuscules.
* Évitez les noms qui existent déjà dans le JDK.
* Ne pas utiliser d'accents.
* Le nom doit être descriptif de ce que contient le package.

Exemple :

package mypackage ;

## **Classes et Interfaces :**

* Utilisez la casse CamelCase avec la première lettre de chaque mot en majuscule

Exemple :

public class MaClasse { } // Classe

public interface MonInterface { } // Interface

## **Résumé - Nommage :**

|  |  |  |
| --- | --- | --- |
| **Sujet** | **À faire** | **À ne pas faire** |
| **Packages** | package mypackage; | package MyPackage; |
| **Classes et Interfaces** | public class MaClasse { }  // Classe  public interface MonInterface { } // Interface | public class maclasse { }  // Classe  public interface moninterface { } // Interface |

# **Méthodes**

## **Nommage :**

* Les noms des méthodes doivent commencer par une minuscule et utiliser la casse CamelCase pour les mots suivants.

Exemple :

public void maMethode() {// Nom de méthode

## **Déclaration :**

* Préférez déclarer chaque paramètre sur une ligne séparée lorsque vous avez de nombreux paramètres, pour une meilleure lisibilité.
* Incluez les exceptions potentielles lancées par la méthode dans la déclaration (ex : **throws IllegalArgumentException**).

Exemple :

public void listerNomPersonnel(

// Si trop de paramètres, les mettre sur des lignes séparées

int premierParam,

int secondParam,

int troisièmeParam,

int quatrièmeParam

)

throws IllegalArgumentException {

// Corps de la méthode

}

## **Instructions Return :**

* Utilisez des parenthèses pour les opérations au sein des instructions return pour clarifier la priorité des opérations (ex : **return (a + b) \* c;**).

Exemple :

public int additionnerEtMultiplier (int a, int b, int c) {

return (a + b) \* c; // Utilisation de parenthèses

}

## **Résumé - Métohdes :**

|  |  |  |
| --- | --- | --- |
| **Sujet** | **À faire** | **À ne pas faire** |
| **Nommage** | public void maMethode() {  // Nom de méthode | public void MaMethode() {  // Nom de méthode |
| **Déclaration des Méthodes** | public void listerNomPersonnel(  // Si trop de paramètres, les mettre :  int premierParam,  int secondParam,  int troisièmeParam,  int quatrièmeParam  )  throws IllegalArgumentException{ // Corps de la méthode  } | public void listerNomPersonnel(int premierParam, int secondParam, int troisièmeParam, int quatrièmeParam)  throws IllegalArgumentException{ // Corps de la méthode  } |
| **Instructions Return** | public int calcul (int a, int b, int c) {  return (a + b) \* c; | public int calcul (int a, int b, int c) {  return a + b \* c; |

# **Variables**

## **Nommage :**

* Utilisez des noms courts et descriptifs pour les variables. Utilisez la casse camelCase, en commençant par une lettre minuscule

(ex : int nbPersonnel; // Nom court et descriptif)

## **Déclaration :**

* Déclarez les variables ligne par ligne, sauf pour les variables temporaires itératives qui peuvent être déclarées sur une même ligne (ex : **int i, j, k;**).

Exemple :

int x; // Déclaration individuelle

int y;

int i, j, k; // Variables itératives sur la même ligne

## **Constantes :**

* Nommez les constantes en majuscules avec des underscores pour séparer les mots (ex : **MAX\_SIZE**, **DEFAULT\_VALUE**).

## **Types de Variables :**

* Évitez de déclarer des variables de différents types sur la même ligne.

## **Résumé - Variable :**

|  |  |  |
| --- | --- | --- |
| **Sujet** | **À faire** | **À ne pas faire** |
| **Nommage des Variables** | int nbPersonnel;  // Noms courts, descriptifs, en camelCase. | int nb;  // Noms non descriptifs ou ne suivant pas la casse camelCase. |
| **Déclaration des Variables** | int x;  int y;  int i, j, k;  //Variables déclarées ligne par ligne, sauf pour les variables itératives. | int x, y;  // Déclaration de plusieurs variables non itératives sur la même ligne. |
| **Constantes** | final int MAX\_SIZE = 100;  final int DEFAULT\_VALUE = 10;  //Constantes en majuscules avec des underscores. | final int Maxsize = 100;  final int defaultvalue = 10;  // Constantes non écrites en majuscules ou sans underscores. |

# **Blocs de Code**

## **Structures Conditionnelles :**

* Utilisez des accolades même pour les instructions conditionnelles d'une seule ligne pour maintenir la cohérence et faciliter les modifications ultérieures.

Exemple :

if (condition) { // Utilisation des accolades même pour une seule ligne

executerTache();

}

## **Boucles :**

* Suivez les mêmes règles d'indentation et d'espace que pour les autres blocs de code.

Exemple :

for (int i = 0; i < n; i++) { // Boucle avec des règles d'indentation standard

process(i);

}

## **Résumé - Blocs de code :**

|  |  |  |
| --- | --- | --- |
| **Sujet** | **À faire** | **À ne pas faire** |
| **Structures Conditionnelles** | if (condition) {  executerTache();  } | if (condition)  {  executerTache();  } |
| **Boucles** | for (int i = 0; i < n; i++) {  process(i);  } | for (int i=0;i<n;i++) process(i);  //Manque d'indentation et d'espaces appropriés, et manque d'accolades. |

# **Commentaires**

## **Documentation :**

* Documentez chaque classe, méthode et section importante de code avec des commentaires clairs et concis.

## **JavaDoc :**

* Utilisez JavaDoc pour générer automatiquement la documentation du code. Incluez une description, les paramètres, les valeurs de retour, et les exceptions pour chaque méthode.

Exemple :

/\*\*

\* Cette méthode effectue une tâche spécifique.

\* @param param Le paramètre pour la méthode

\* @return La valeur calculée

\*/

public int maMethode(int param) {

// Corps de la méthode

}

# **Autres Conventions Spécifiques**

## **Saut de Ligne :**

* Effectuez un saut de ligne si une ligne de code dépasse la limite de 80 colonnes pour assurer la lisibilité sur un écran standard.

## **Caractères Spéciaux :**

* Évitez l'utilisation de caractères spéciaux comme le dollar ($) et le soulignement (\_) dans les noms de variables non constantes ou de méthodes.

**Accents :**

* Évitez l'utilisation des accents dans le code pour garantir la compatibilité et la lisibilité sur différents systèmes.