Configurer IntelliJ IDEA (v2022.3.1)

– Pour le développement Java –

Nous allons voir dans ce document les concepts basiques pour savoir configurer IntelliJ en fonction de ses préférences mais aussi en fonction du contexte dans lequel on évolue (perso / pro).

La plupart des chapitres aborde les préférences que vous pouvez afficher avec la commande `CTRL+ALT+S` ou bien via le menu "`File > Settings…`".

> ***Note*** *: Ce document n'est qu'une introduction et je vous invite à vous former progressivement à IntelliJ en suivant le parcours de leçons intégrées directement à l'IDE. Pour lancer le module de cours, tapez deux fois de suite sur le bouton "MAJ" ou "SHIFT" et saisissez la commande "learn" (appuyez sur entrée pour valider).*

Table des matières

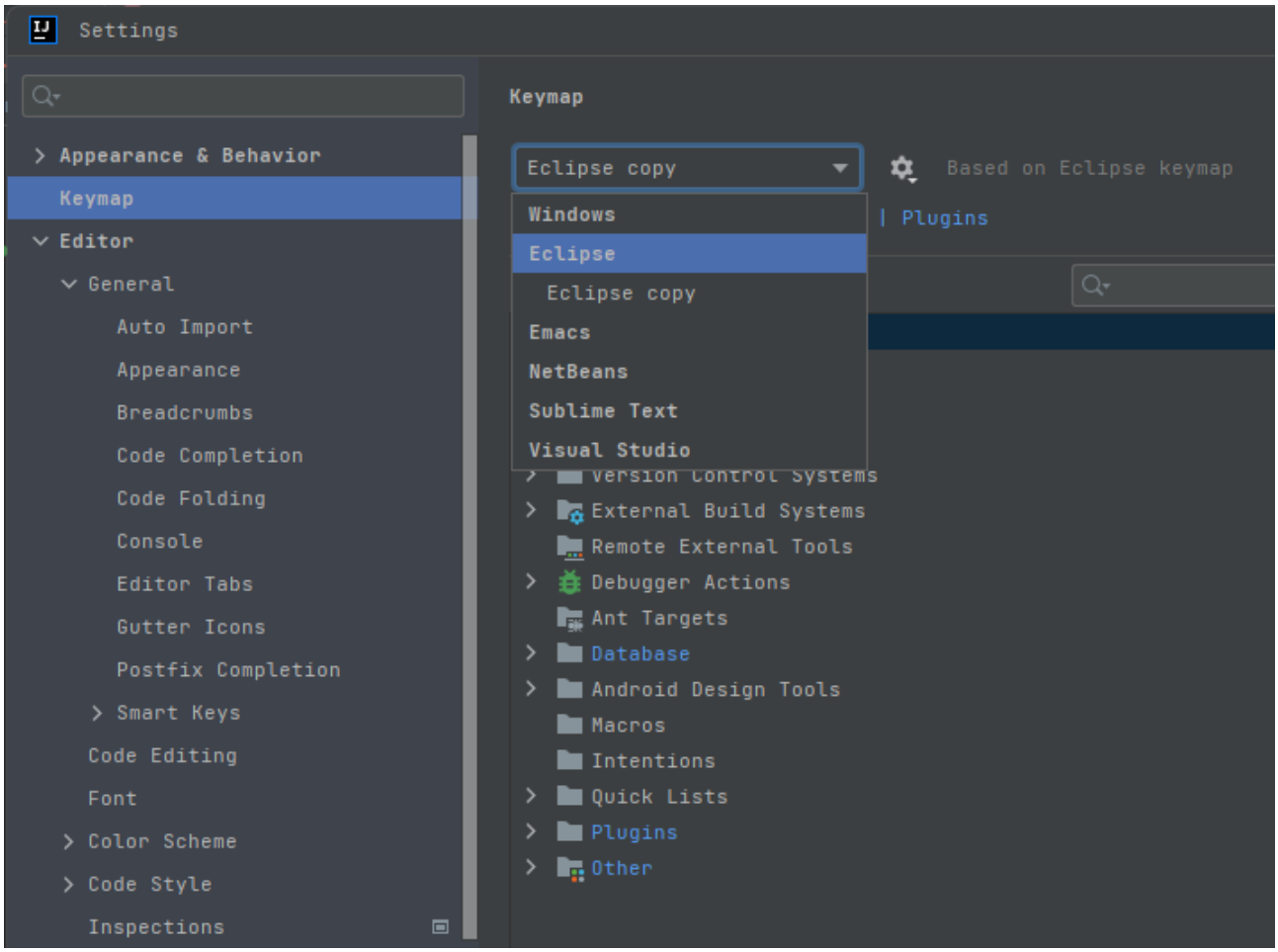
Configurer IntelliJ IDEA (v2022.3.1)....1

1. Settings > Keymap....3
2. Settings > Editor > Code Style > Java....4
3. Settings > Editor > General > Auto Import....5
4. Settings > Editor > General > Code Completion....5
5. Settings > Editor > General > Postfix Completion....6
6. Settings > Editor > General > Smart Keys....8

1. Settings > Keymap

Une map ou un "mapping" est une **table de correspondances**. À une propriété on fait correspondre une valeur. Ici la *Keymap* contient pour chacun des menus de l'application le raccourci-clavier qui est associé.

Si vous utilisez souvent certaines fonctions, c'est dans ce menu que vous pouvez adapter le *mapping* des raccourcis clavier. IntelliJ propose des *mappings* par défaut, calqués sur d'autres IDE comme Eclipse ou Visual Studio. C'est très pratique si vous avez plutôt l'habitude d'utiliser Eclipse par exemple (ce qui est mon cas comme vous le voyez ci-dessous) :



Les commandes à connaître absolument :

- ALT+ALT+L [Reformat Code]
  - Formater son code
- ALT+MAJ+M [Extract Method]
  - Extraire le code sélectionné en tant que méthode
- ALT+MAJ+R [Rename all occurrences]
  - Renommer toutes les occurrences d'une variable ou d'une méthode
- CTRL+MAJ+ENTER [Complete Current Statement]
  - Termine la ligne en cours en mettant un point-virgule et déplace le curseur à la fin

Pour connaître tous les raccourcis clavier, cliquez sur le menu **`Help > Keyboard Shortcuts PDF`**

2. Settings > Editor > Code Style > Java

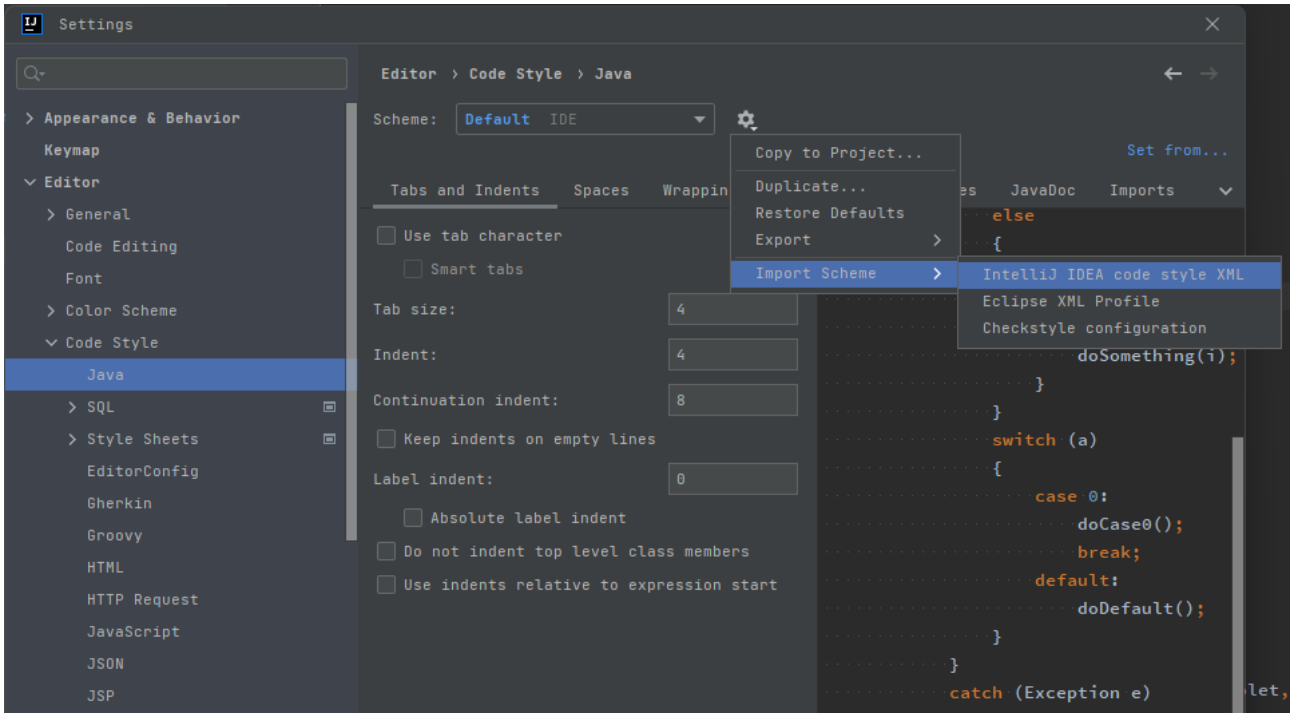
Lorsqu'on programme en milieu professionnel, on ne se soucie généralement jamais du formatage de notre code pour la simple raison que c'est l'IDE qui s'en charge pour nous.

Dans IntelliJ, pour formater votre code vous tapez **ALT+MAJ+L**. Dans Eclipse, vous taperez **CTRL+MAJ+F**. Les raccourcis pour formater son code dépendent de votre *Keymap* (voir chapitre correspondant).

Lorsqu'on travaille en équipe, on utilise Git pour synchroniser ses développements avec ceux de ses collègues. Git vous permet de visualiser rapidement les différences qu'il y a entre deux versions d'un même fichier édité par 2 personnes différentes. Mais si vous utilisez des styles de formatage de code différents, ça ne sera pas exploitable ! Donc vous devez impérativement utiliser le même style de formatage que vos collègues. Ce style sera décidé en général au début d'un projet en fonction des exigences de votre client ou bien de l'organisation de votre équipe.

Pour paramétrer correctement son formateur de code, il faut :

1. Que tous les membres de l'équipe laissent les valeurs par défaut
2. Ou bien que chaque membre mporte un fichier de configuration au format IntelliJ IDEA XML :



Pour travailler en groupe, importez le fichier que j'ai déposé à l'adresse suivante :

3. Settings > Editor > General > Auto Import

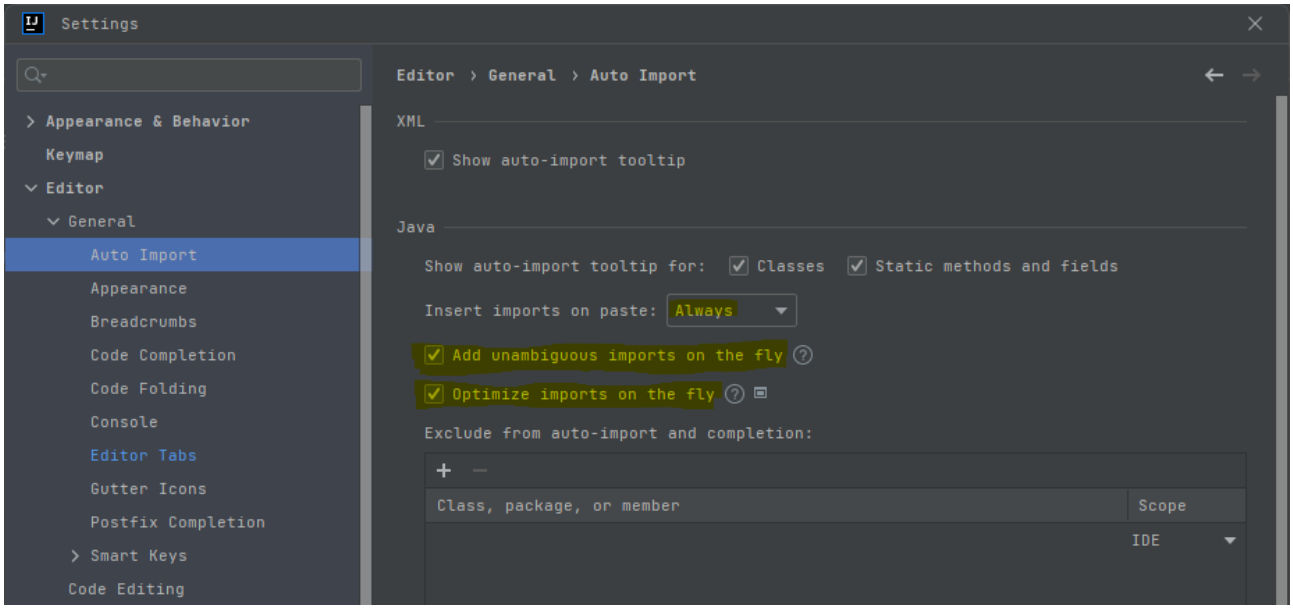


Figure 1: Settings > Editor > General > Auto Import - Configuration recommandée

Les imports automatiques sont très utiles en Java car à chaque fois que vous voulez utiliser une classe dans votre code, il faut l'importer en début de fichier en mettant son nom complet. Or c'est souvent compliqué de savoir de quel package exact est issue une classe. Je vous invite donc à utiliser le maximum d'assistance possible sur ce point, en cochant les cases surlignées en jaune.

4. Settings > Editor > General > Code Completion

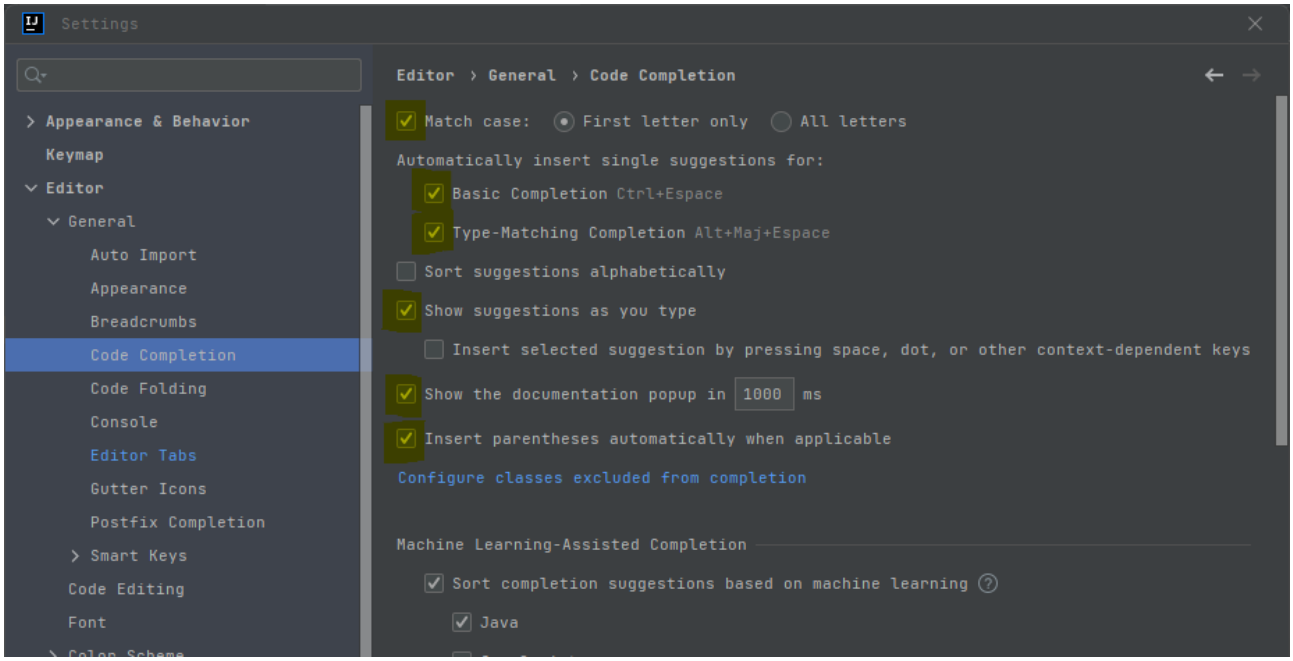


Figure 2: Settings > Editor > General > Code Completion - Configuration recommandée

Ce qu'on appelle la "*completion*" est sans doute le plus grand atout d'un IDE ! Vous devez l'utiliser le plus possible, c'est une assistance qui vous propose de compléter votre texte avec des éléments du langage dans lequel vous codez.

Par exemple si vous tapez "`int maVariable = maFo`" puis CTRL+ESPACE, IntelliJ va tenter de compléter automatiquement le code ; première possibilité : il essaye de trouver si un nom de méthode commence par "`maFo`"

dans votre classe courante. Si c'est le cas il vous proposera une liste déroulante avec les fonctions qui commencent par "maFon" ! Ce qui est beaucoup plus rapide que de tout écrire à la main, d'autant plus qu'il vous écrira également les parenthèses et les éventuelles virgules qui séparent les paramètres de votre fonction.

5. Settings > Editor > General > Postfix Completion

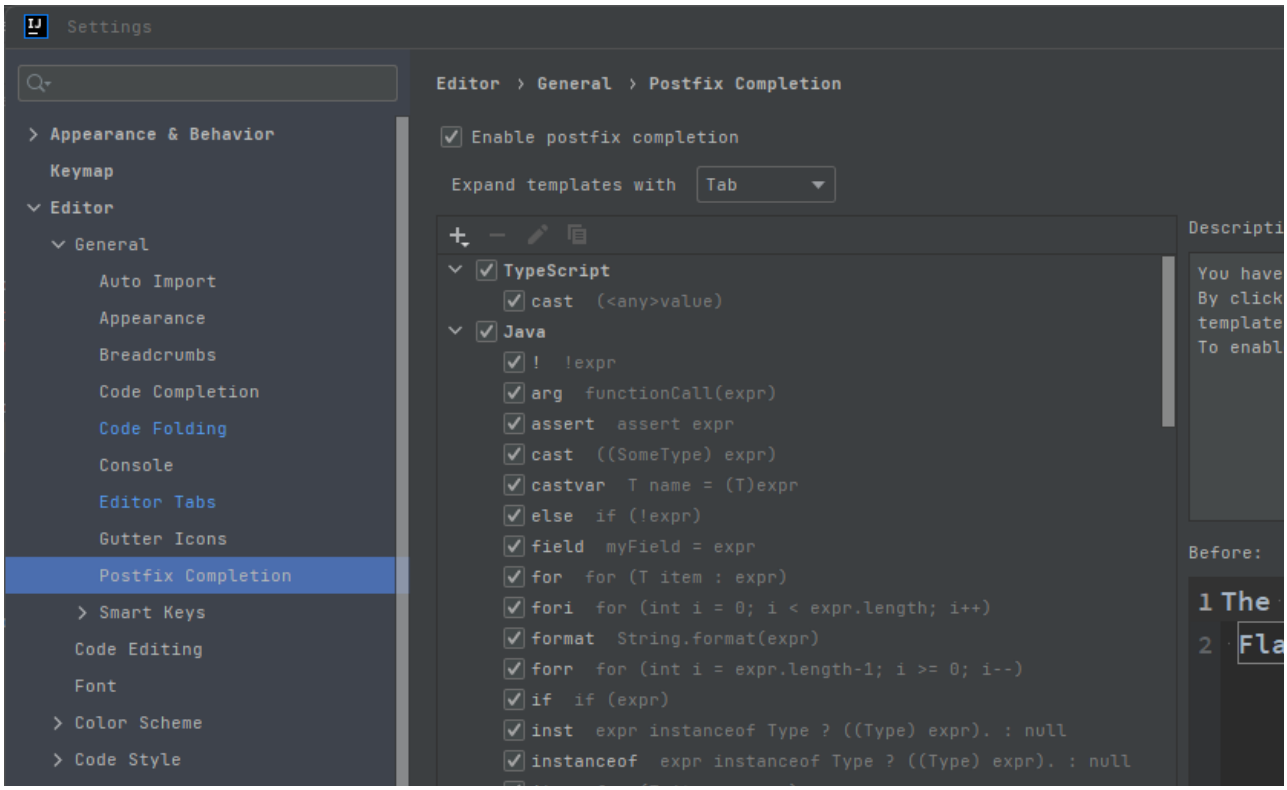
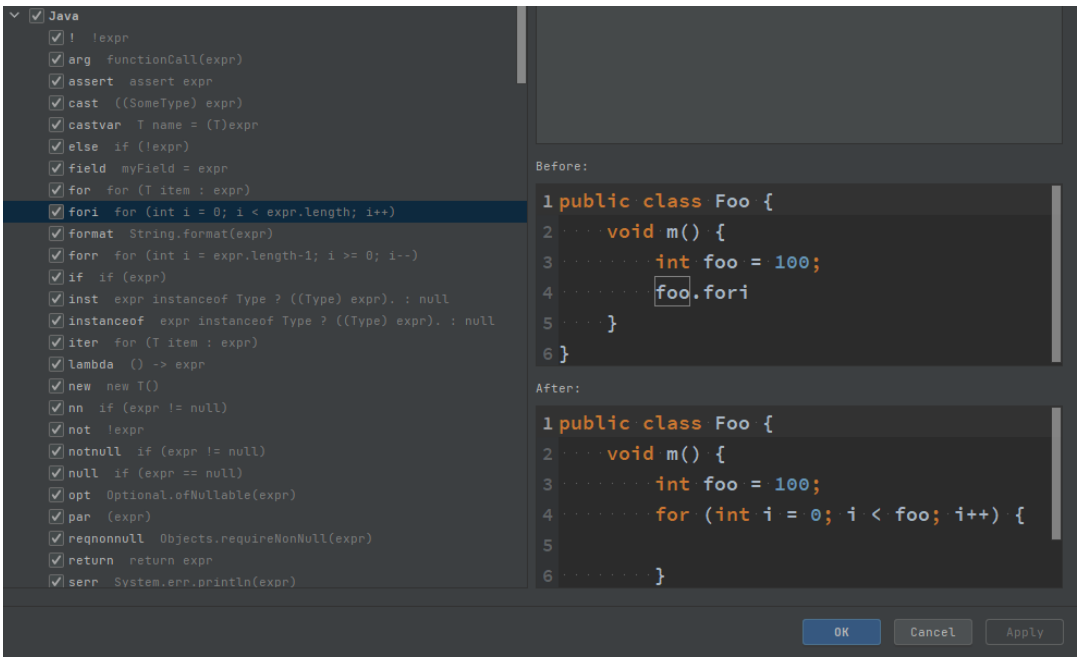


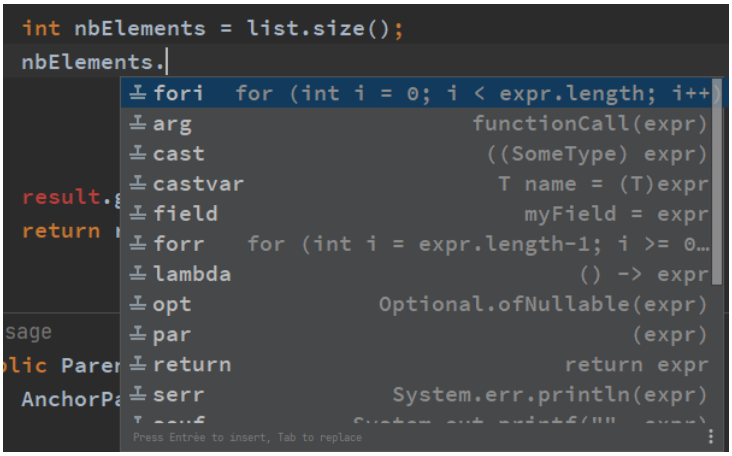
Figure 3: Settings > Editor > General > Postfix Completion - Configuration recommandée

Il s'agit des raccourcis textuels qui vous évitent de taper l'intégralité d'une ligne de code. Par exemple pour éviter de taper "`System.out.println()`" manuellement, vous pouvez utiliser le mot-clé "`sout`" : tapez simplement "`sout`" dans votre code Java et IntelliJ va automatiquement vous proposer de remplacer "`sout`" par "`System.out.println()`".

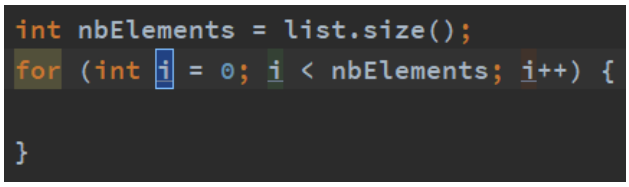
Chaque raccourci vous est expliqué dans les 2 panneaux latéraux de droite :



Par exemple pour le raccourci "`fori`" regardez ci-dessous comment il s'utilise (si besoin, tapez `CTRL+ESPACE` pour réactiver la completion après le point de "`nbElements.`") :



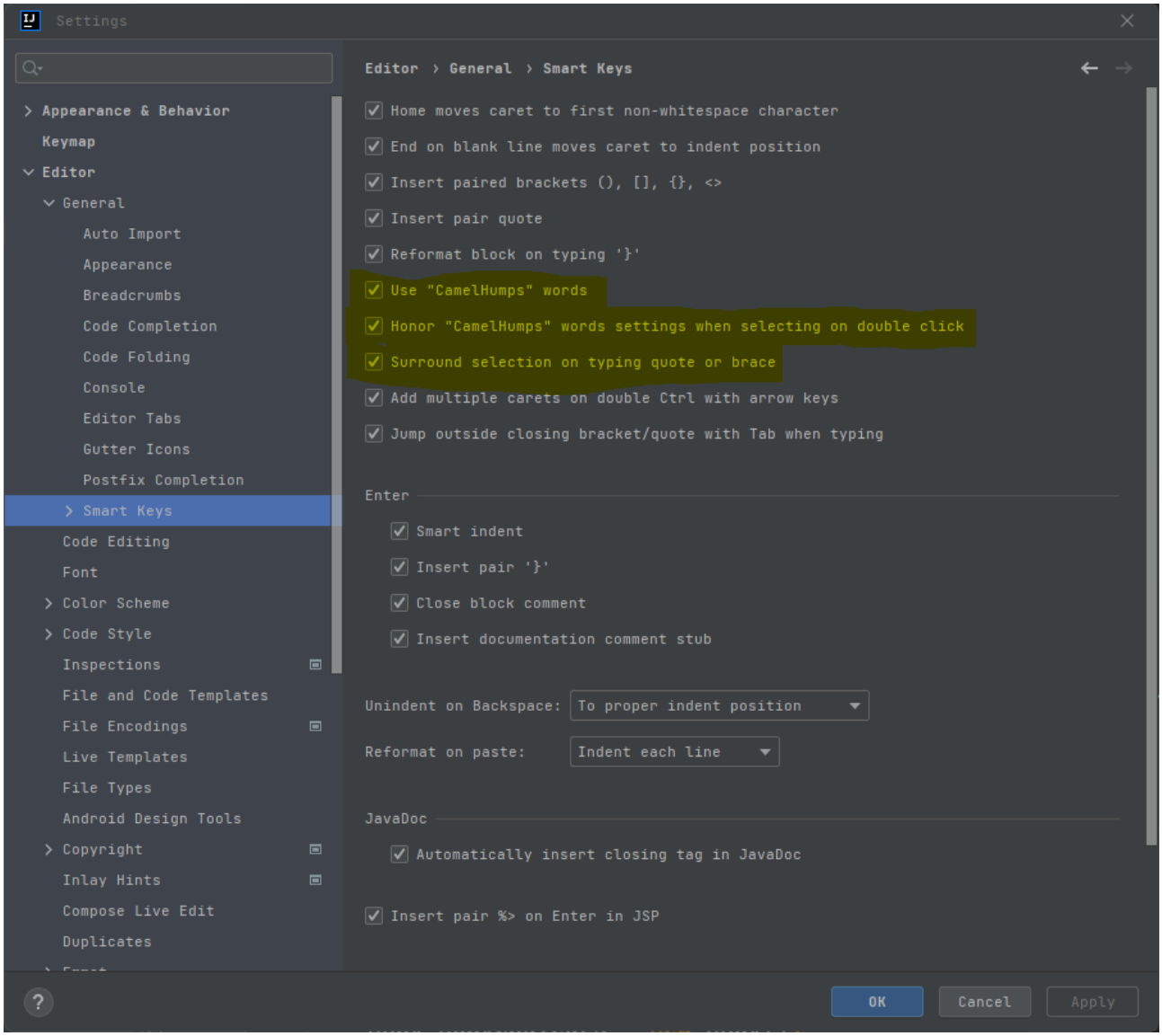
Et voici ce qui est automatiquement inséré si j'appuie sur `Entrée` :



Notez qu'à cet instant, le "`i`" est sur fond bleu : ça signifie que **IntelliJ attend une action de votre part** pour renommer la variable "`i`". Si son nom vous convient, appuyez sur **`Entrée`** une seconde fois. Sinon tapez le nom qui vous convient puis **`Entrée`**.

6. Settings > Editor > General > Smart Keys

Les Smart Keys sont très intéressantes dans IntelliJ et vous font gagner énormément de temps. Je ne vais pas revenir en détail sur chaque option de cette section, mais je voulais attirer votre attention sur les 3 options surlignées en jaune ci-dessous :



☒ `Use "CamelHumps" words` :

> Pour illustrer à quoi sert cette option il me faut d'abord introduire ce qu'est la navigation "mot à mot". Dans n'importe quel éditeur de texte ou traitement de texte, vous pouvez vous déplacer d'un mot à l'autre avec les touches `CTRL+GAUCHE` ou `CTRL+DROITE`. Essayez dans votre éditeur de texte favori !
>
> En Java, on écrit tous les noms de variables, de méthodes et de classes avec une convention de nommage qui s'appelle "Camel case" (la casse – "*case*" en anglais – c'est la distinction entre majuscules et minuscules).
>
> **Si un nom de variable/méthode/classe est composé de plusieurs mots, la première lettre de chaque mot doit être en Majuscule**.

Exemple :

```
int numberOfStudents = getNumberOfStudentsInTheSchool() ;
```

Si vous vous êtes trompés d'orthographe dans un nom de méthode par exemple, vous trouverez intéressant de pouvoir vous déplacer à l'intérieur du nom "`getNumberOfStudentsInTheSchool`" comme s'il s'agissait de plusieurs mots, pour vous déplacer de mot en mot avec les touches `CTRL+GAUCHE / CTRL+DROITE` au lieu de vous déplacer d'un seul caractère à la fois ou bien d'aller à la fin du mot avec CTRL+DROITE.

C'est exactement ce que propose de faire l'option "`Use 'CamelHumps' words`" ! Essayez-la, je vous garantis que vous ne pourrez plus vous en passer.

☒ Honor "CaleHump" words settings when selecting on double click :

Cette option est indépendante de la précédente mais elle fait appel au même concept de "CamelHump words" permettant d'interpréter des noms avec alternance majuscules/minuscules comme plusieurs mots.

Si vous activez cette option, le comportement du double-clic va changer : il ne sélectionnera pas tout le nom d'une variable/méthode/classe mais simplement le mot sur lequel vous cliquez.

Exemple si je double-clique sur le "`d`" de "`Students`" :

```
int numberOfStudents = getNumberOfStudentsInTheSchool() ;
```

L'option permet de ne sélectionner que le mot situé sous le curseur de la souris au moment où j'ai cliqué. Essayez-la pour tester mais si ça ne vous convient pas décochez cette option.

☒ Surround selection on typing quote or brace :

Cette option est intéressante mais peut se révéler piégeuse si on ne sait pas qu'elle existe et qu'elle est activée par défaut.

Elle sert à mettre des double-quotes "" ou des accolades {} ou des parenthèses ( ) ou des crochets [ ] autour du texte qu'on vient de sélectionner.

Exemple 1 :

```
String sql = SELECT name FROM Employees WHERE birthday=today() ;
```

Appuyez ensuite sur la touche double-quote (touche 3) "

```
⇒    String sql = "SELECT name FROM Employees WHERE birthday=today()" ;
```

Exemple 2 :

```
// Il manque des crochets [] à "array2" :

int[] array = {1, 2, 3, 4};

int arrayElement = array2; // Sélectionnez le 2
```

Appuyez maintenant sur la touche crochet "["

```
⇒ int arrayElement = array[2]; // IntelliJ a entouré la sélection avec []
```