## Résumé des Variables

Le langage Java emploie dans sa terminologie les termes de **champs** et de **variables**. Les **variables d'instance** (champs non statiques) sont individuelles à chaque instance d'une classe. Les **variables de classe** (champs statiques) sont déclarés à l'aide de l'atérateur `static` ; il n'existe qu'une copie de la variable pour la classe, quelque soit le nombre d'objets qui ait été créé de cette classe. Les **variables locales** stockent des données temporaires nécessaires que déroulement d'une méthode. Les paramètres sont des variables qui apportent de l'information à une méthode ; Les variables locales et les **paramètres** sont qualifiés tous deux de "variables" (et non "champs"). Il existe un ensemble de conventions et de règles pour nommer les champs et les variables qu'il est recommandé (expressément) de suivre.

Je lis...

Les **huit types de données primitifs** sont : `byte`, `short`, `int`, long, `float`, `double`, `boolean`, et `char`. La classe `java.lang.String` représente les chaînes de caractères. Le compilateur affecte une valeur par défaut raisonnable aux champs de ces types ; les variables locales n'ont pas de valeur préaffectée. Un **litéral** est la représentation dans le source d'une valeur fixe. Un tableau est une construction qui rassemble un nombre fixé d'élément du même type. La longueur d'un tableau est fixée définitivement au moment de sa création.

Je lis...

## Résumé des Operateurs

La référence qui suit est un rappel des opérateurs disponibles en Java.

**Opérateur d'affectation simple**

```
=       Simple assignment operator
```

**Opérateurs arithmétiques**

```
+       Additive operator (also used for String
        concatenation)
-       Subtraction operator
*       Multiplication operator
/       Division operator
%       Remainder operator
```

**Opérateurs unaires**

```
+       Unary plus operator; indicates positive
        value (numbers are positive without this, however)
-       Unary minus operator; negates an expression
++      Increment operator; increments a value by 1
--      Decrement operator; decrements a value by 1
!       Logical compliment operator; inverts the value of a boolean
```

**Opérateurs de test d'égalité et opérateurs relationnels**

```
==      Equal to
!=      Not equal to
>       Greater than
>=      Greater than or equal to
<       Less than
<=      Less than or equal to
```

**Opérateurs conditionnels**

```
&&      Conditional-AND
||      Conditional-OR
?:      Ternary (shorthand for if-then-else statement)
```

**Opérateur de comparaison de type**

```
instanceof     Compares an object to a specified type
```

**Opérateurs binaires**

```
~       Unary bitwise complement
<<      Signed left shift
>>      Signed right sift
>>>     Unsigned right shift
&       Bitwise AND
^       Bitwise exclusive OR
|       Bitwise inclusive OR
loading...
```

## Résumé des instructions de contrôle d'exécution

L'instruction `if-then` est la plus basique des instructions de contrôle d'exécution. Elle dit qu'un programme n'exécutera un bloc de code *que si* une certaine condition vaut `true`. L'instruction `if-then-else` propose une alternative d'exécution si la condition vaut `false`. Contrairement aux instructions `if-then` et `if-then-else`, l'instruction `switch` permet de choisir parmi un nombre indéterminé d'alternatives. Les boucles `while` et `do-while` exécutent un bloc de code continuellement tant qu'une certaine condition vaut `true`. `do-while` et `while` diffèrent en cela que `do-while` évalue la condition à la fin de la boucle. De ce fait, les instructions du bloc `do` sont au moins exécutées une fois. L'instruction `for` propose une syntaxe compacte pour itérer dans un ensemble connu de valeurs. Elle présente deux forme, dont l'une particulièrement adaptée au parcours de collections ou de tableaux de valeurs.

Je lis...