

Chaînes de caractères en Java

Introduction

Les chaînes de caractères (que nous appellerons "strings" par la suite) sont souvent utilisées en programmation. Un string est une suite ou une séquence de caractères.

Dans beaucoup de langages un string n'est autre qu'un tableau de caractères, alors qu'en Java un **string** est un **objet**.

En fait Java propose 3 classes apparentées aux chaînes de caractères:

- La classe String (chaîne de caractères non modifiables)
- La classe StringBuffer (chaîne de caractères modifiables à volonté)
- La classe StringTokenizer (séparation d'une chaîne en plusieurs entités)

Dans la plupart de cas il convient d'utiliser String pour créer, stocker et gérer des chaînes de caractères.

La classe permet une plus grande souplesse et s'utiliser de la même manière que la classe String.

Classe String

Cette classe dispose de 11 constructeurs et plus de 40 méthodes pour examiner les caractères d'une chaîne, comparer, chercher, extraire etc. Parmi ces méthodes, certaines sont décrites ci-dessous. On peut trouver les autres dans l'aide de Java.

Construction d'un string:

Une chaîne de caractère se déclare normalement de la manière suivante:

```
String prenom = new String("Pierre");
```

Comme les strings sont un type de données très utilisé, Java permet une déclaration plus concise:

```
String prenom = "Pierre";
```

Comparaison de strings

On peut être tenté de comparer deux string à l'aide de l'opérateur == comme dans:

```
If (str1 == str2) ...
```

Or, cette comparaison, bien que correcte, ne compare pas si les deux chaînes sont égales, mais si str1 et str2 pointent vers le même objet.

Une comparaison de chaînes s'effectue de la manière suivante:

```
If (str1.equals (str2))...
```

Il existe aussi la méthode `compareTo`:

```
str1.compareTo (str2);
```

Cette méthode retourne 0 si les deux chaînes sont égales, une valeur négative si `str1` est plus petit que `str2`, ou une valeur positive si `str2` est plus grand que `str1`.

Il ne faut donc pas utiliser les opérateurs `>`, `>=`, `<`, `<=`

Concaténation

On peut utiliser la méthode `concat` pour concaténer deux chaînes:

```
String s3 = s1.concat (s2);
```

Il est également possible de faire appel au signe d'addition `+` pour effectuer une concaténation:

```
String mot = message + " et " + "puis s'en vont";
```

Extraction de sous-chaînes

Un objet de la classe `String` est immuable. Une fois qu'il a été défini, sa valeur ne peut pas être modifiée. On peut, en revanche, lui attribuer une nouvelle chaîne de caractères. Si `msg` est défini comme suit:

```
String msg = "Attention!";
```

On peut parfaitement changer son contenu:

```
msg = "nouveau texte";
```

En fait, la chaîne "Attention!" est oubliée et l'objet `msg` pointe vers la nouvelle chaîne "nouveau".

On peut extraire une sous-chaîne à l'aide de la méthode `substring` de la classe `String`. Deux versions existent:

- `public String substring (int debut, int fin)`

Retourne un nouveau string qui débute au caractère `debut` et va jusqu'au caractère à la position `fin - 1`.

Par exemple: `txt = msg.substring(0, 6);`

Retourne la chaîne "nouveau"

- `public String substring (int debut)`

Retourne un nouveau string qui commence au caractère `debut` et qui se termine à la fin du string initial.

Par exemple: `txt = "premier " + "nouveau cas".substring(8);`

Retourne la chaîne "premier cas"

Longueur d'une chaîne et accès aux caractères

La longueur d'une chaîne est obtenue à l'aide de la méthode `length()`. Par exemple:

```
txt.length() retourne 11
```

Si on a une chaîne `str`, la méthode `str.charAt(index)` retourne le caractère spécifié de la chaîne `str`. Le paramètre `index` peut avoir une valeur comprise entre 0 et `str.length()-1`.

Conversions diverses

Bien que le contenu d'une chaîne ne puisse pas être modifié, il est possible d'effectuer des conversions en créant une nouvelle chaîne.

Les méthodes `toLowerCase` et `toUpperCase` permettent d'obtenir une chaîne respectivement en minuscules et en majuscules

La méthode `trim` permet d'obtenir une nouvelle chaîne sans espaces au début ou à la fin.

La méthode `replace(oldChar, newChar)` permet de remplacer tous les caractères `oldChar` d'une chaîne par des caractères `newChar`.

Conversion de caractères et de valeurs numériques en chaîne de caractères

La classe `String` dispose de plusieurs méthodes `valueOf` permettant de convertir un caractère, un tableau de caractères et des valeurs numériques en chaînes de caractères. Ces méthodes ont le même nom, mais se différencient par le type de paramètre qui leur est fourni (`char`, `char[]`, `double`, `long`, `int` et `float`).

Classe `StringBuffer`

De manière générale un `StringBuffer` peut être utilisé partout où un `String` est utilisé. Il est simplement plus flexible: on peut modifier son contenu. Cette classe dispose de 3 constructeurs et de plus de 30 méthodes dont les plus utilisées sont les suivantes (les paramètres ne sont pas indiqués ici, on trouvera plus de détail dans l'aide de Java):

- `append`
- `capacity`
- `charAt`
- `delete`
- `insert`
- `length`
- `replace`

- reverse
- setCharAt
- setLength
- substring

Construction d'un StringBuffer

Les 3 manière de construire un StringBuffer sont:

- `public StringBuffer()`
qui construit une chaîne vide d'une capacité initiale de 16 caractères
- `public StringBuffer (int longueur)`
qui construit une chaîne vide de la capacité indiquée par `longueur`.
- `public StringBuffer (String str)`
qui construit une chaîne recevant le paramètre `str`. La capacité de la chaîne est 16 plus la longueur de `str`.

Exemples d'utilisation

La méthode `append` cache plusieurs méthodes du même nom permettant d'ajouter des expressions de type `char`, `char[]`, `double`, `float`, `int`, `long`, et `String`. Exemple de construction utilisant la méthode `append`:

```
StringBuffer stbuf = new StringBuffer();
stbuf.append ("Cours");
stbuf.append (" ");
stbuf.append ("de ");
stbuf.append ("Java.");
```

Il en va de même pour la (les) méthode(s) `insert`. Voici un exemple:

```
stbuf.insert (9, "HTML et ");
```

Un `StringBuffer` évolue dynamiquement. Si on augmente la longueur de son contenu, l'espace mémoire alloué est automatiquement augmenté.

Capacité et longueur

La méthode `capacity` retourne la capacité, c'est-à-dire le nombre de caractères qu'une `StringBuffer` peut recevoir.

La méthode `length` retourne le nombre de caractères effectivement placés dans le `StringBuffer`.

La méthode `setLength(int newlong)` permet de spécifier la longueur d'un `StringBuffer`. Si la valeur du paramètre `newlong` est inférieure au nombre de caractères dans le `StringBuffer`, celui-ci est tronqué. Si la valeur de `newlong` est supérieure au nombre de caractères du `StringBuffer` sont contenu est complété par des caractères nuls ('\u0000').

Classe StringTokenizer

Cette classe est utilisée pour séparer une chaîne de caractères en plusieurs morceaux. Voyons comment la classe StringTokenizer peut reconnaître des mots. Tout d'abord il est possible de spécifier des séparateurs lors de la construction d'un StringTokenizer. Voici les constructeurs possibles:

- `public StringTokenizer(String s, String delim, boolean tokens)`
qui construit un StringTokenizer pour le string s avec les séparateurs indiqués par delim. Si tokens est vrai le séparateur est retourné en tant que morceau du string,
- `public StringTokenizer(String s, String delim)`
qui construit un StringTokenizer pour le string s avec les séparateurs indiqués par delim. Le séparateur n'est pas considéré comme un morceau du string,
- `public StringTokenizer(String s)`
qui construit un StringTokenizer pour le string s avec les séparateurs par défaut (espace, newline, tab et return) et le séparateur est retourné en tant que morceau du string,

Voici un exemple montrant comment extraire les mots d'une phrase:

```
String s = "Java est un langage moderne";
StringTokenizer st = new StringTokenizer (s);

System.out.println ("Nombre de mots:" + st.countTokens())
while (st.hasMoreTokens())
    System.out.println(st.nextToken());
```

Exercice 1

Ecrire une applet Java comportant une zone de texte et un bouton. L'utilisateur entre un mot dans la zone de texte. En cliquant sur le bouton le programme indique si le mot est un palindrome (mot qui se lit dans les deux sens). Utilisation obligatoire de la classe String.

Exercice 2

Comme l'exercice 1, mais en utilisant un StringBuffer.

Exercice 3

Ecrire un programme Java (application ou applet) qui affiche le calendrier pour un mois et une année donnés sous la forme:

```
Lu ma me je ve sa di
    1  2  3  4  5  6
  7  8  9 10 11 12 13
etc
```

Exercice 4

Ecrire une application utilisant StringTokenizer qui permet de calculer une expression entrée par l'utilisateur. Par exemple, si l'expression est:

12+4+3

L'application doit afficher:

Le résultat est 19