Les chaînes de caractères en Java

Table des matières :

1 Introduction

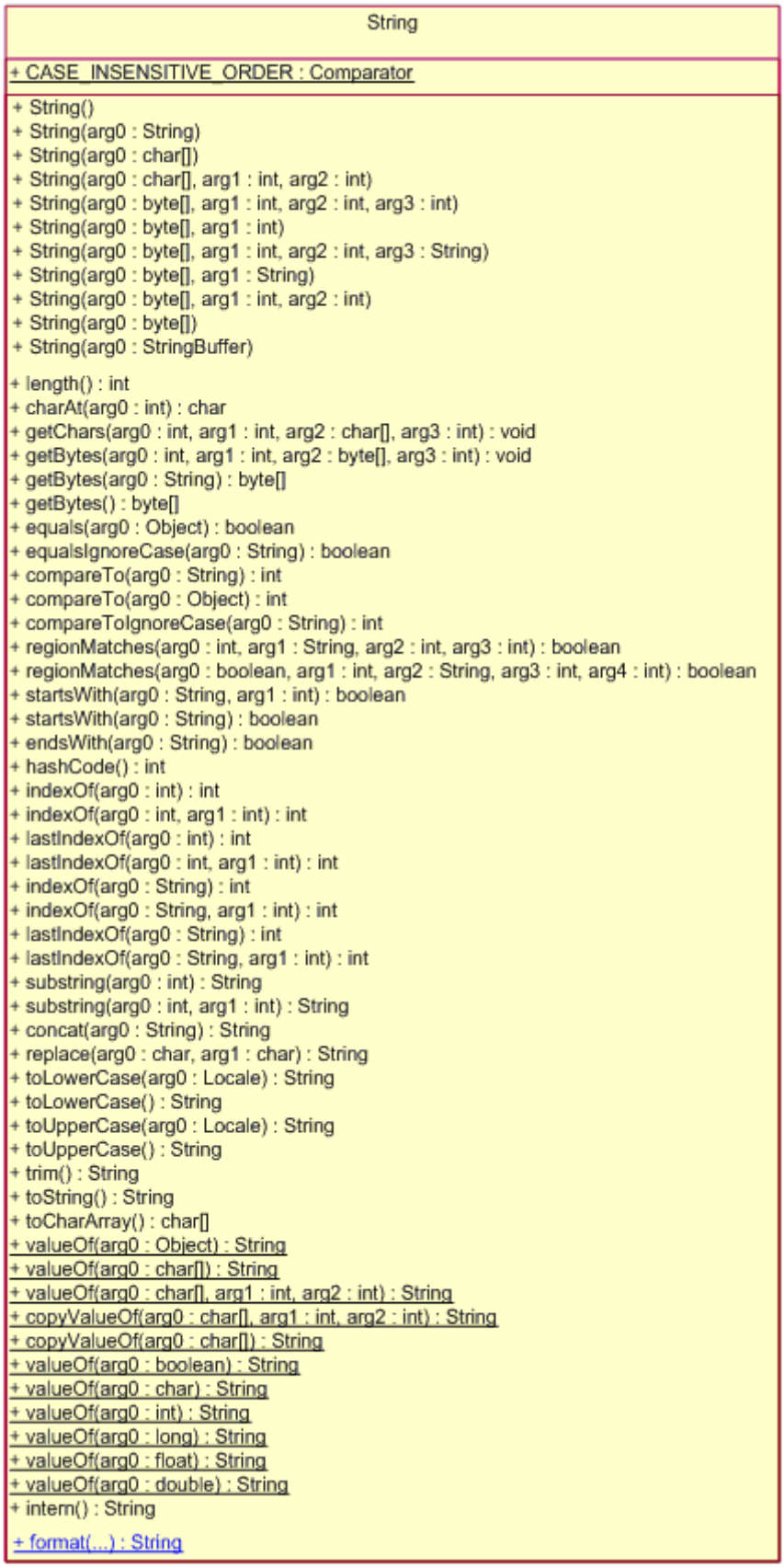
Les chaînes de caractères (que nous appellerons "strings" par la suite) sont souvent utilisées en programmation. Un string est une suite ou une séquence de caractères.
Dans beaucoup de langages un string n'est autre qu'un tableau de caractères, alors qu'en Java un string est un objet.
En fait Java propose 2 classes apparentées aux chaînes de caractères:

- La classe String (chaîne de caractères non modifiables)
- La classe StringBuffer (chaîne de caractères modifiables à volonté)

Dans la plupart de cas il convient d'utiliser String pour créer, stocker et gérer des chaînes de caractères.
La classe StringBuffer permet une plus grande souplesse et s'utilise de la même manière que la classe String.

2 Classe String

Le diagramme de classes suivant présente la classe String (J2SE 1.3):



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 plus loin. On peut trouver les autres dans l'aide de Java. La méthode valueOf est statique et c'est un outil qui nous servira pour les conversions.

2.1 Tableau des méthodes les plus utiles de la classe String

Non-exhaustif …

signature de la méthode	Description de la catégorie
int length() char charAt(int index)	Interrogation du nombre de caractères de la chaîne et du caractère à un indice donné.
boolean equals(Object o) boolean equalsIgnoreCase(String anotherString)	Comparaison avec d'autres chaînes
int indexOf(String sMaChaine) int lastIndexOf(String sMaChaine)	Recherche de caractères ou de sous-chaînes(la valeur renvoyée est l'indice du texte recherché dans cette chaîne ou -1 si la recherche à échoué)
String substring(int iDepart, int iFin) String toUpperCase() String toLowerCase() String trim()	Construction d'autres chaînes (sans modification de cette chaîne)
static String valueOf(int / boolean / double / float / long)	Conversion de données d'un type primitif ou d'objets en chaînes de caractères (ce sont ces méthodes de classe qui sont en fait appelées par l'opérateur + de concaténation pour convertir un opérande en chaîne de caractères)

2.2 Construction d'une chaîne de caractères

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";

String firstName = prenom ;
```

Ici la copie de référence fonctionne mais c'est une exception !!!

2.3 Comparaison de deux chaînes

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 >, >=, <, <=

2.4 Concaténation de chaînes

On peut utiliser la méthode concat pour concaténer deux chaînes:

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

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

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

C'est la seul surcharge d'opérateur que l'on trouve en Java.

2.5 String non-mutable (immuable)

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".

2.6 Extraction de sous-chaînes

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

2.6.1 Version 1

```
public String substring (int début, int fin)
```

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

Par exemple:

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

Retourne la chaîne "nouveau"

2.6.2 Version 2

```
public String substring (int début)
```

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

Par exemple :

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

Retourne la chaîne "premier cas"

2.7 Longueur d'une chaîne

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

```
txt.length()
```

retourne 11

2.8 Accès aux caractères

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.lenght()-1.

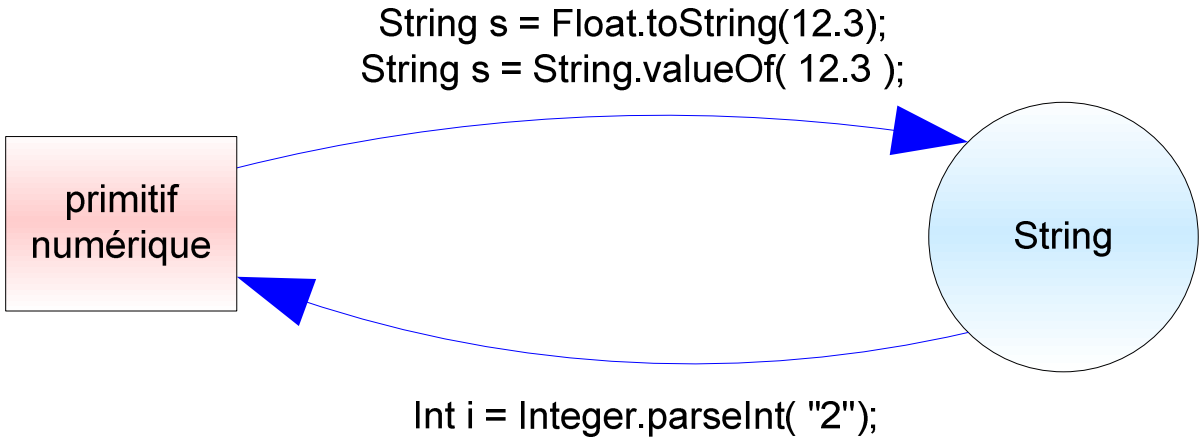
2.9 Recherche d'un caractère ou d'une sous-chaîne

Voir indexOf et lastIndexOf

2.10 Transformations 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 str.replace (oldchar, newChar) permet de remplacer tous les caractère oldChar d'une chaîne par des caractères newChar.

2.11 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)..
Exemple pour le passage primitif/String et String/primitif :

2.12 La méthode split depuis Java 1.4

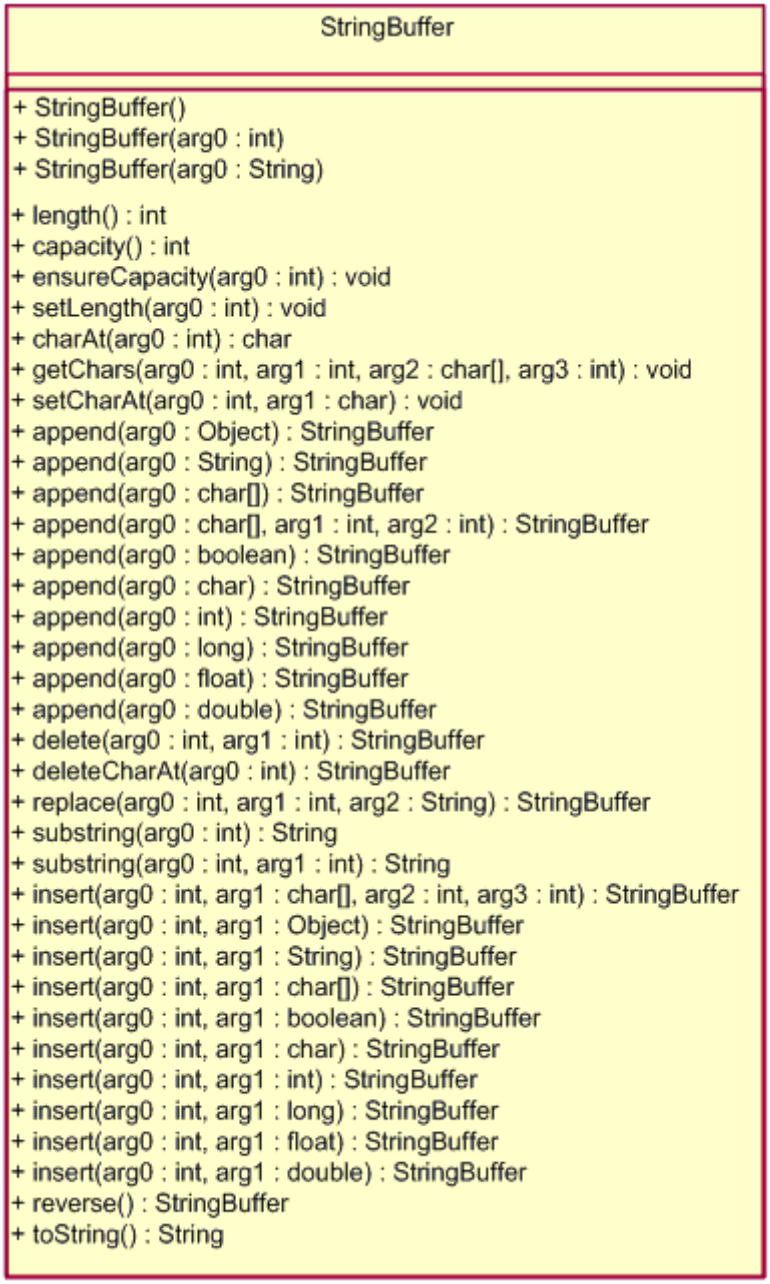
Depuis Java 1.4, cette méthode existe. Elle utilise en argument les expressions régulières …

2.13 La méthode format depuis Java 1.5

Pour l'affichage avec un format (printf)

3 Classe StringBuffer

Le diagramme de classes (UML) suivant représente la classe StringBuffer (J2SE 1.3):



De manière générale un StringBuffer peut être utilisé partout où un String est utilisé. Il est simplement plus flexible: on peut modifier son contenu ! C'est une chaîne mutable. 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 la Javadoc) :

append, capacity, charAt, delete, insert, length, replace, reverse, setCharAt, setLength, substring

3.1 Construction d'un StringBuffer

Les 3 manières 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.

3.2 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.") ;
```

II 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.

3.3 Capacité et longueur

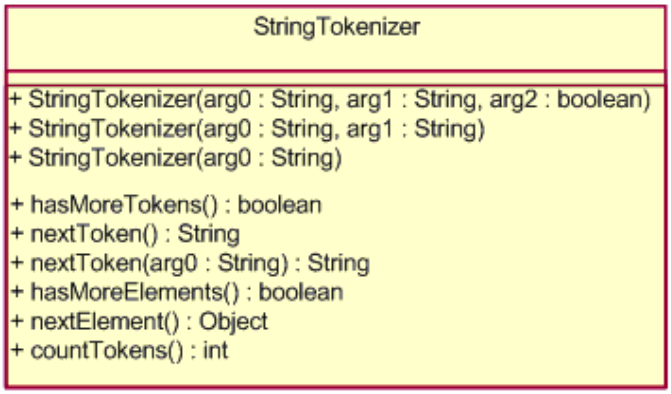
La méthode capacity retourne la capacité, c'est-à-dire le nombre de caractères qu'une StringBuffer peut recevoir.

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').

3.4 La classe StringTokenizer

Le diagramme de classes suivant présente la classe StringTokenizer (J2SE 1.3):



A découvrir en autonomie

3.5 La classe StringBuilder depuis Java 1.5

Comme StringBuffer, c'est une chaîne mutable. A découvrir en autonomie