Les chaînes de caractères

Les variables de type String permettent la manipulation de chaînes de caractères par votre application.

Nous allons regarder comment réaliser les opérations les plus courantes sur les chaînes de caractères.

Création d'une chaîne de caractères

La méthode la plus simple pour créer une chaîne de caractères consiste à considérer le type String comme un type de base du langage et non comme un type objet. C'est dans ce cas l'affectation d'une valeur à la variable qui va provoquer la création d'une instance de la classe String. La création d'une chaîne de caractères comme un objet est bien sûr également possible en utilisant l'opérateur new et un des nombreux constructeurs disponibles dans la classe String. L'exemple de code suivant présente les deux solutions.

```
String chaine1="eni";
String chaine2=new String("eni");
```

Après sa création une chaîne de caractères ne peut plus être modifiée. L'affectation d'une autre valeur à la variable provoque la création d'une nouvelle instance de la classe String. La classe String contient de nombreuses méthodes permettant la modification de chaînes de caractères. À l'utilisation, nous avons l'impression que la fonction modifie le contenu de la chaîne initiale mais en fait c'est une nouvelle instance contenant le résultat qui est renvoyée par la fonction.

Affectation d'une valeur à une chaîne

Nous avons vu que pour affecter une valeur à une chaîne il faut la spécifier entre les caractères " et ", un problème se pose si nous voulons que le caractère " fasse partie de la chaîne. Pour qu'il ne soit pas interprété comme caractère de début ou de fin de chaîne il faut le protéger par une séquence d'échappement comme dans l'exemple ci-dessous :

```
String Chaine;
Chaine=" il a dit : \" ça suffit ! \"";
System.out.println(Chaine);
```

Nous obtenons à l'affichage : il a dit : "ça suffit ! "

Pour les exemples suivants, nous allons travailler avec deux chaînes :

```
chaine1 = "l'hiver sera pluvieux";
chaine2 = "l'hiver sera froid";
```

Extraction d'un caractère particulier

Pour obtenir le caractère présent à une position donnée d'une chaîne de caractères, il faut utiliser la fonction charAt en fournissant comme argument l'index du caractère que l'on souhaite obtenir. Le premier caractère a l'index zéro comme pour un tableau. Cette fonction retourne un caractère (char).

```
System.out.println("le troisieme caractere de la chaine1 est " +
chaine1.charAt(2));
```

Obtenir la longueur d'une chaîne

Pour déterminer la longueur d'une chaîne, la fonction length de la classe String est disponible.

```
System.out.println("la chaine1 contient " + chaine1.length() + "
caracteres");
```

Découpage de chaîne

La fonction substring de la classe String retourne une portion de chaîne en fonction de la position de départ et de la position de fin qui lui sont passées comme paramètres. La chaîne obtenue commence par le caractère situé à la position de départ et se termine au caractère précédant la position de fin.

```
System.out.println("un morceau de la chaine1 : " + chaine1.substring(2,8));
```

Nous obtenons à l'affichage :

```
un morceau de la chaine1 : hiver
```

Comparaison de chaînes

Lorsqu'on fait une comparaison de deux chaînes, on est tenté d'utiliser le double égal (==), comme précédemment. Cet opérateur fonctionne correctement sur les types de base mais il ne faut pas perdre de vue que les chaînes de caractères sont des types objet. Il faut donc utiliser les méthodes de la classe String pour effectuer des comparaisons de chaînes de caractères. La méthode equals effectue une comparaison de la chaîne avec celle qui est passée comme paramètre. Elle retourne un boolean égal à true si les deux chaînes sont identiques et bien sûr un boolean égal à false dans le cas contraire. Cette fonction fait une distinction entre minuscules et majuscules lors de la comparaison. La fonction equalsIgnoreCase effectue un traitement identique mais sans tenir compte de cette distinction.

```
if (chaine1.equals(chaine2))
{
    System.out.println("les deux chaines sont identiques");
}
else
{
    System.out.println("les deux chaines sont différentes");
}
```

Ne vous laissez pas tromper par les apparences. Dans certains cas, l'opérateur == est bien capable de réaliser une comparaison correcte de chaînes de caractères. Le code ci-dessous fonctionne correctement et fournit bien le résultat attendu et considère que les deux chaînes sont identiques.

```
String s1="toto";
String s2="toto";
if (s1==s2)
{
   System.out.println("chaines identiques");
}
else
{
   System.out.println("chaines différentes");
}
```

En fait, pour économiser de l'espace en mémoire, Java n'utilise dans ce cas qu'une seule instance de la classe String pour les variables s1 et s2 car le contenu des deux chaînes est identique.

Les deux variables s1 et s2 référencent donc la même zone mémoire et l'opérateur == constate donc l'égalité.

Si par contre nous utilisons le code suivant qui demande explicitement la création d'une instance de la classe String pour chacune des variables s1 et s2, l'opérateur == ne constate bien sûr plus l'égalité des chaînes.

```
String s1=new String("toto");
String s2=new String("toto");
if (s1==s2)
{
   System.out.println("chaines identiques");
}
else
{
   System.out.println("chaines différentes");
}
```

Pour réaliser un classement, vous devez par contre utiliser la méthode compareTo de la classe String ou la fonction compareToIgnoreCase. Avec ces deux solutions il faut passer comme paramètres la chaîne à comparer. Le résultat de la comparaison est retourné sous forme d'un entier inférieur à zéro si la chaîne est inférieure à celle reçue comme paramètre, égal à zéro si les deux chaînes sont identiques, et supérieur à zéro si la chaîne est supérieure à celle reçue comme paramètre.

```
if (chaine1.compareTo(chaine2)>0)
{
   System.out.println("chaine1 est superieure a chaine2");
}
else
if (chaine1.compareTo(chaine2)<0)
{
   System.out.println("chaine1 est inferieure a chaine2");
}
else
{
   System.out.println("les deux chaines sont identiques");
}
```

Les fonctions startsWith et endsWith permettent de tester si la chaîne débute par la chaîne reçue en paramètre ou si la chaîne se termine par la chaîne reçue en paramètre. La fonction endsWith peut par exemple être utilisée pour tester l'extension d'un nom de fichier.

```
String nom="Code.java";

if (nom.endsWith(".java"))
{
    System.out.println("c\'est un fichier source java");
}
```

Suppression des espaces

La fonction trim permet de supprimer les espaces situés avant le premier caractère significatif et après le dernier caractère significatif d'une chaîne.

```
String chaine="          eni          ";
System.out.println("longueur de la chaine : " + chaine.length());
System.out.println("longueur de la chaine nettoyee : " + chaine.trim()
.length());
```

Changer la casse

Tout en majuscules :

```
System.out.println(chaine1.toUpperCase());
```

Tout en minuscules :

```
System.out.println(chaine1.toLowerCase());
```

Recherche dans une chaîne

La méthode indexOf de la classe String permet la recherche d'une chaîne à l'intérieur d'une autre. Le paramètre correspond à la chaîne recherchée. La fonction retourne un entier indiquant la position à laquelle la chaîne a été trouvée ou -1 si la chaîne n'a pas été trouvée. Par défaut la recherche commence au début de la chaîne, sauf si vous utilisez une autre version de la fonction indexOf qui, elle, attend deux paramètres, le premier paramètre étant pour cette version, la chaîne recherchée et le deuxième la position de départ de la recherche.

```
String recherche;
      int position;
      recherche = "e";
      position = chaine1.indexOf(recherche);
      while (position > 0)
      {
         System.out.println("chaîne trouvée à la position " + position);
         position = chaine1.indexOf(recherche,position+1);
      }
      System.out.println("fin de la recherche");
```

Nous obtenons à l'affichage :

```
chaîne trouvée à la position 5
chaîne trouvée à la position 9
chaîne trouvée à la position 18
fin de la recherche
```

Remplacement dans une chaîne

Il est parfois souhaitable de pouvoir rechercher la présence d'une chaîne à l'intérieur d'une autre, comme dans l'exemple précédent, mais également remplacer les portions de chaînes trouvées. La fonction replace permet de spécifier une chaîne de substitution pour la chaîne recherchée. Elle attend deux paramètres :

- la chaîne recherchée
- la chaîne de remplacement

```
String chaine3;
chaine3= chaine1.replace("hiver", "ete");
System.out.println(chaine3);
```

Nous obtenons à l'affichage :

```
l'ete sera pluvieux
```

Formatage d'une chaîne

La méthode format de la classe String permet d'éviter de longues et fastidieuses opérations de conversion et de concaténation. Le premier paramètre attendu par cette fonction est une chaîne de caractères spécifiant sous quelle forme on souhaite obtenir le résultat. Cette chaîne contient un ou plusieurs motifs de formatage représentés par le caractère % suivi d'un caractère spécifique indiquant sous quelle forme doit être présentée l'information. Il doit ensuite y avoir autant de paramètres qu'il y a de motifs de formatage. La chaîne renvoyée est construite par le remplacement de chacun des motifs de formatage par la valeur du paramètre correspondant, le remplacement se faisant dans l'ordre d'apparition des motifs. Le tableau suivant présente les principaux motifs de formatage disponibles.

Motif	Description
%b	Insertion d’un booléen
%s	Insertion d’une chaîne de caractères
%d	Insertion d’un nombre entier
%o	Insertion d’un entier affiché en octal
%x	Insertion d’un entier affiché en hexadécimal
%f	Insertion d’un nombre décimal
%e	Insertion d’un nombre décimal affiché au format scientifique
%n	Insertion d’un saut de ligne

L’exemple de code suivant :

```
boolean b=true;
   int i=56;
   double d=19.6;
   String s="chaine";
   System.out.println(String.format("boolean : %b %n" +
                  "chaine de caractères : %s %n" +
                  "entier : %d %n" +
                  "entier en hexadécimal : %x %n" +
                  "entier en octal : %o %n" +
                  "décimal : %f %n" +
                  "décimal au format scientifique : %e%n",
                  b,s,i,i,i,d,d));
```

affiche ce résultat sur la console :

```
boolean : true
chaine de caractères : chaine
entier : 56
entier en hexadécimal : 38
entier en octal : 70
décimal : 19,600000
décimal au format scientifique : 1,960000e+01
```