Chapitre3 :

Les chaînes de caractères

Table des matières

3.1 La classe *String*....2
3.1.1 Création d'objet String....2
Chaîne littérale....2
Constructeurs de la classe String....2
3.1.2 Longueur String et accès aux caractères d'une chaîne....3
3.1.3 Sous chaîne....3
3.1.4 Recherches dans une chaîne....3
3.1.5 Concaténation de chaîne....4
l'opérateur +....4
La méthode *concat*....4
3.1.6 Comparaison des chaînes de caractère....5
L'opérateur d'égalité ==....5
La méthode *equals*....5
La méthode *compareTo*....5
3.1.7 Autres fonctionnalités....6
Remplacement de caractères ou de sous-chaîne....6
Interchanger majuscule est minuscule....6
3.2 La classe *BufferString*....6

Une chaîne est une séquence de caractères. Les mots, les phrases et les noms sont donc des chaînes. Par exemple, le message "Hello, World!" est une chaîne. Ce chapitre décrit deux classes de chaîne principales : String et StringBuffer (ou encore StringBuilder).

3.1 La classe String

La classe String est la classe la plus utilisée pour traiter les chaînes de caractères en Java. un objet String est immuable, cela veut dire qu'une fois crée, un objet String, ne peut être modifier.

3.1.1 Création d'objet String

Deux manières pour créer un objet String

- utiliser une chaîne littérale
- utiliser le mot clé new constructeurs

Chaîne littérale

Un littérale String est une séquence de caractères entre des doubles cote ("). Exemple :

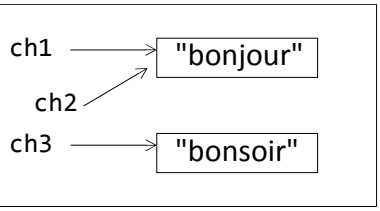
```
String ch="abc" ;
```

Java gère les littéraux String d'une manière particulière : il ne crée qu'une seule instance de l'objet String pour un littérale donné.
Chaque fois qu'une référence pour une chaîne littéral est déclarée, java vérifie si la chaîne existe en mémoire (une mémoire réservée pour les chaînes littérales), dans le cas échéant, la nouvelle référence pointe sur la chaîne existante sinon un nouvel objet String est créé et qui sera pointé par la nouvelle référence.

Exemple

```
String ch1="bonjour" ;
String ch2="bonjour" ;//ne créera pas une nouvelle instance

String ch3="bonsoir" ;//créera  une nouvelle instance
```



Constructeurs de la classe String

Contrairement aux chaînes littérales, une création d'un objet String avec le mot clé ***new*** crée systématiquement une nouvelle instance de la classe String
la classe String dispose de plusieurs constructeurs, dont les plus importants sont :
String(), String (String s), String(char[] tabChar)

Exemple

```
String s1="java";//création d'une chaîne littérale
String s2=new String() ; //création d'une chaîne vide

char ch[]={'s','t','r','i','n','g','s'};
String s2=new String(ch);//constructeur utilisant un tableau de caractères
String s3=new String("example");//constructeur utilisant un objet de type String
```

3.1.2 Longueur String et accès aux caractères d'une chaîne

La longueur d'un objet String est le nombre de caractères qui le composent, on peut la connaître en utilisant la méthode *length()*.
Dans une chaîne de caractère, pour accéder à un caractère d'une position donnée *i*, on utilise la méthode *charAt(i).* Il est à noter que les caractères d'un chaîne sont indexés de 0 jusqu'à la (length()-1)

Exemple

```
String s="bravo" ;
System.out.print(s.charAt(0));// affiche : b
System.out.print(s.charAt(3));// affiche : v
System.out.print(s.charAt(s.length()));//erreur de compilation
System.out.print(s.charAt(s.length()-1));// affiche : o
```

3.1.3 Sous chaîne

La méthode *substring* permet d'extraire plus d'un seul caractère dans une chaîne. Elle possède deux variantes :
String substring(int debut, int fin) : renvoie une nouvelle chaîne qui commence à la position *debut* et se termine à la position *fin*, le caractère de position *fin* n'est pas pris dans la nouvelle sous-chaîne.

String substring(int debut) : renvoie une nouvelle chaîne qui commence à la position *debut* et se termine à la *fin* de la chaîne

Exemple

```
String s1="Cours Java";
System.out.println(s1.substring(2,4));//affiche : ur
System.out.println(s1.substring(2));//affiche : urs Java
```

3.1.4 Recherches dans une chaîne

La classe String fournie des méthodes qui permettent de retrouver un caractère ou une sous-chaîne dans une chaîne de caractère. Il s'agit de *indexOf* et *lastIndexOf : indexOf* effectue une recherche vers l'avant, par contre *lastIndexOf* effectue une recherche vers l'arrière. Si le caractère ou de la sous-chaîne recherché est trouvé, les méthodes *indexOf et lastIndexOf* retourne sa position positif, sinon elles retourne -1.

int indexOf (int car) : retourne la position du premier *car* dans la chaîne
int indexOf (int car, int pos) : retourne la position du premier *car* recherché à partir de *pos*
int indexOf (String ch) : retourne la position de la première occurrence de *ch* dans la chaîne
int indexOf (String ch, int pos) : retourne la position de la première occurrence de *ch* dans la chaîne recherché à partir de *pos*

Exemple

```
String s1="Voici un exemple de indexOf";

//Passer une sous chaine
int index1=s1.indexOf("un");//renvoie la position de la première occurrence de la
sous-chaîne "ex":6
int index2=s1.indexOf("ex");//renvoie la position de la première occurrence de
"ex":9

//Passer une sous chaine et une position départ de recherche
int index3=s1.indexOf("ex",10);//renvoie la position de la première occurrence de "ex"
à partir de la position 10 : 23

//Passer une valeur de type caractère
int index4=s1.indexOf('i');//renvoie la position de la première occurrence 'i' :2
int index5=s1.indexOf('s');//renvoie la position de 's':-1 ('s' non trouvé)
```

3.1.5 Concaténation de chaîne

La classe String fournie deux façons de concaténer des chaînes de caractères:

- l'opérateur +
- la méthode *concat*

l'opérateur +

Une des spécificités de classe String est l'utilisation de l'opérateur + pour concaténer des chaînes de caractères.

Exemple

```
String ch1="bon", ch2="jour" ;
String ch3=ch1+ch2 ;//ch3 référence "bonjour"
```

Si l'opérande est de type primitif, l'opérateur de concaténation + entraînera aussi la conversion de cette opérande en une chaine de caractères avant la réalisation de concaténation

Exemple

```
String ch1="jour"+1; // donne "jour1"
String ch2="valeur :"+false; //donne "valeur :false"
String ch3=3+5+"valeur"+4+4; //donne "8valeur44"
```

La méthode concat

Cette méthode permet de concaténer l'objet String passer en paramètre l'objet String courent

```
public String concat(String autreChaine)
```

exemple

```
String ch1="bon", ch2="jour" ;
String ch3=ch1.concat(ch2) ;//ch3 référence "bonjour"
```

3.1.6 Comparaison des chaînes de caractère

L'opérateur d'égalité ==

Comme pour toute référence Java, pour deux références String, l'opérateur d'égalité (==) permet de tester ces derniers pointent vers le même objet.
Exemple :

```
String s1 = "ABC";
String s2 = "ABC";
System.out.println("s2 == s1: " + (s2 == s1));
String s3 = new String("ABC");
System.out.println("s3 == s2: " + (s3 == s2));
String s4 = new String("ABC");
System.out.println("s4 == s3: " + (s4 == s3));
```

```
L'exécution donne l'affichage suivant :

s2 == s1: true
s3 == s2: false
s3 == s4: false
```

On constate clairement que l'opérateur == ne permet pas de comparer le contenu de deux objets String

La méthode equals

La classe String propose la méthode *equals* qui permet l'égalité les contenus des chaînes de caractères. Dans l'exemple précédent, on remplace l'opérateur d'égalité == par la méthode *equals :*

```
String s1 = "ABC";
String s2 = "ABC";
System.out.println("s2.equals(s1): " + s2.equals(s1);
String s3 = new String("ABC");
System.out.println("s3.equals(s2): " + s3.equals(s2);
String s4 = new String("ABC");
System.out.println("s4.equals(s3): " + s4.equals(s3);
```

l'exécution donnera l'affichage suivant :

```
s2.equals(s1): true
s3.equals(s2): true
s4.equals(s3): true
```

La méthode compareTo

La Classe String permet de comparer des chaînes de caractère **lexicographiquement.** La comparaison est basée sur la valeur Unicode de chaque caractère dans les chaînes.

- Le résultat est un entier négatif si cet objet String précède lexicographiquement la chaîne d'argument.
- Le résultat est un entier positif si cet objet String suit lexicographiquement la chaîne d'arguments.
- Le résultat est nul si les chaînes sont égales; *CompareTo* retourne 0 exactement quand le contenu des deux chaînes est identique.

Exemple :

```
System.out.print("bon".compareTo("ton")) ;// affiche -18, parce que 'b'<'t'
System.out.print("arme".compareTo("arbre")) ;// affiche 11 ,parce que 'm'>'b'
System.out.print("abc".compareTo("Abc")) ;// affiche 32 ,parce que 'a'>'b'
```

3.1.7 Autres fonctionnalités

Remplacement de caractères ou de sous-chaîne

`String replace(char ancienCar, char nouveauCar)` ou

`String replace(String ancienneCh, String nouvelleCh)` : La méthode *replace* permet de remplacer, dans une chaîne, les occurrences d'un caractère (respectivement d'une sous-chaine) par un autre caractère (respectivement par une autre sous-chaine)

exemple :

```
System.out.println("l'Algérie-est-un-beau-paye!".replace('-',' ');
//Affiche :l'Algérie est un beau paye!
System.out.println("abcde".replace("cd","CD");
//Affiche :abCDe
```

Interchanger majuscule est minuscule

String toUpperCase() : créer une nouvelle chaîne en remplaçant les caractères minuscule par des caractères majuscules
Exemple : `String s="coUrs jAva".toUpperCase() ;// s référence "COURS JAVA"`

String toLowerCase() : créer une nouvelle chaîne en remplaçant les caractères majuscule par des caractères minuscules

Exemple : `String s="coUrs jAva".toLowerCase() ;// s référence "cours java"`

élimination des espaces qui entoure une chaine

conversion d'une chaîne vers un tableau

3.2 La classe BufferString

Malgré que les objets de type *String* permettent d'effectuer la plupart des manipulations de chaînes, la plus importante caractéristique de la classe String est que ses objets ne sont pas modifiables, et la moindre modification d'une chaîne ne peut se faire qu'en créant une nouvelle chaîne (c'est le cas d'une simple concaténation).

Dans les programmes manipulant intensivement des chaînes, la perte de temps qui en résulte peut devenir gênante. C'est pourquoi Java dispose d'une classe *StringBuffer* destinée elle aussi à la manipulation de chaînes, mais dans laquelle les objets sont modifiables.

La classe *StringBuffer* dispose de fonctionnalités classiques. On peut créer un objet de type *StringBuffer* à partir d'un objet de type *String*. Il existe des méthodes :

- de modification d'un caractère de rang donné : ***setCharAt***,
- d'accès à un caractère de rang donné : ***charAt***,
-
- d'ajout d'une chaîne en fin : la méthode ***append*** accepte des arguments de tout type

primitif et de type String,

- d'insertion d'une chaîne en un emplacement donné : ***insert***,
- de remplacement d'une partie par une chaîne donnée : ***replace***,
- de conversion de StringBuffer en String : ***toString*.**