Chapitre 8

Courte introduction au type String

Il existe en java un type des chaînes de caractères. Ce type s'appelle `String` et les valeurs de ce type sont des objets. Nous verrons plus tard dans le cours ce que sont les objets, comment on les crée, etc. Pour l'instant, nous allons juste présenter très rapidement comment on peut utiliser des chaînes de caractères dans les exercices et exemples que nous traitons.

On peut utiliser le type `String` pour déclarer des variables ou des paramètres de méthodes ou encore des tableaux, exactement comme on utilise les autres types comme `int` ou `boolean`.

Les valeurs constantes du type s'écrivent avec des doubles quotes. Par exemple `"Bonjour"` est une chaîne de caractères. Si l'on veut écrire une chaîne comprennant des doubles quotes, il faut les faire précéder d'un caractère *barre oblique* : `"on m'appelle \"toto\"!"`.

La concaténation de chaînes qui permet de coller une chaîne à la suite d'une autre s'écrit avec l'opérateur `+`. Par exemple `"Bra"+"vo"` est une expression dont la valeur est la chaîne `"Bravo"`.

8.1 Utilisation de méthodes

On peut utiliser des méthodes spécifiques pour les chaînes de caractères au moyen d'une syntaxe nouvelle pour nous. Cette syntaxe c'est : une valeur de type string, un point, le nom de la méthode et les paramètres entre parenthèses. Par exemple, on peut utiliser la méthode `length` (qui n'a aucun paramètre) pour connaître la longueur de la chaîne. En voici un exemple.

Listing 8.1 – (lien vers le code brut)

```
1  public class MethodeString{
2      public static void main(String[] args){
3          String s = "la chaine 1";
4          int x;
5          x = s.length();
6          Terminal.ecrireStringln("La longueur de la chaine \""+s
7                                  + "\" est " + x);
8          x = "une autre chaine".length();
9          Terminal.ecrireStringln("La longueur de l'autre chaine est " + x);
10         x = ("Bra" + "vo").length();
11         Terminal.ecrireStringln("La longueur de Bravo est " + x);
12     }
13 }
```

Voici quelques autres méthodes intéressantes :

- `charAt(int n)` : cette méthode renvoie le nième caractère de la chaîne, la numérotation commence à 0. Par exemple, si `s` est une `String`, `s.charAt(0)` renvoie le premier caractère (type `char`) de `s`.
- `toCharArray()` permet de transformer une chaîne en un tableau de `char`. Par exemple, si `s` est la `String "bonjour"`, `s.toCharArray()` renvoie un tableau de 7 `char` :

0	1	2	3	4	5	6
'b'	'o'	'n'	'j'	'o'	'u'	'r'

- `compareTo(String s2)` : compare deux chaînes selon l'ordre lexicographique (l'ordre du dictionnaire). Si `s1` et `s2` sont deux `String`, `s1.compareTo(s2)` renvoie un entier. Cet entier est négatif si `s1` est plus petit que `s2`, positif si `s1` est plus grand que `s2`, et 0 si `s1` et `s2` sont égales.
- `s1.toLowerCase()` et `s1.toUpperCase()` renvoient une nouvelle chaîne égale à `s1` mais avec toutes les lettres en minuscule et en majuscule respectivement.

Listing 8.2 – (lien vers le code brut)

```
1  public class ExChaine2{
2      public static void main (String [] arguments){
3          String s1 = "bonjour";
4          String s2;
5          Terminal.ecrireString("Entrez une chaine: ");
6          s2 = Terminal.lireString();
7          Terminal.ecrireCharln(s1.charAt(0));
8          Terminal.ecrireStringln(s1.toUpperCase());
9          Terminal.ecrireIntln(s1.compareTo(s2));
10         Terminal.ecrireStringln("" + s1.equals(s2));
11     }
12 }
```

8.2 Egalité de chaînes

La seule chose un peu différente par rapport aux types que nous avons vus jusqu'ici est que les tests de comparaison `==` et `!=` donnent parfois des résultats surprenants quand on les utilise avec des chaînes. Pour comparer des chaînes de caractères, il est préférable d'utiliser la méthode `equals` ou la négation de cette méthode, comme illustré dans l'exemple ci-dessous.

Listing 8.3 – (lien vers le code brut)

```
1  public class ExChaine{
2      public static void main(String[] args){
3          String s1 = "Bonjour";
4          String s2 = "C'est \"bien\" ";
5          String s3;
6          String[] ts = {"Paul", "Andre", "Jacques", "Odette"};
7          Terminal.ecrireStringln(s2);
8          Terminal.ecrireString("Entrez une chaine: ");
9          s3=Terminal.lireString();
10         Terminal.ecrireStringln("s3: " + s3);
11         s2 = "Bon";
12         s3 = s2 + "jour";
13         if (s1 != s3){
```

```
14              Terminal.ecrireStringln("Bizarre: s1 n'est pas egal a s3!");
15              Terminal.ecrireStringln("s1: " + s1 + ":");
16              Terminal.ecrireStringln("s3: " + s3+ ":");
17          }
18          if (s1.equals(s3)){
19              Terminal.ecrireStringln("s1 est quand meme egal a s3!");
20          }
21          if (!s1.equals(s3)){
22              Terminal.ecrireStringln("s1 n'est toujours pas egal a s3!");
23          }
24      }
25  }
```

Le phénomène que nous observons ici est le même que pour les tableaux : il y a deux notions d'égalité différentes. Une égalité d'identité testée par `==` et `!=`. Cela permet de savoir si deux variables sont des noms différents pour la même chose (le même tableau, la même chaîne). Une égalité de contenu pour savoir si les deux variables contiennent les mêmes valeurs (les mêmes contenus de cases pour les tableaux, les mêmes caractères dans le même ordre pour les chaînes). Cette égalité de contenu est testée par la méthode `equals`.

8.3 Paramètre de la méthode main

Depuis le début de l'année, nous utilisons systématiquement la méthode `main` avec un paramètre de type `String[]`, c'est à dire un tableau de chaînes de caractères. Ce paramètre permet de transférer des informations entre la ligne de commande et le programme java. Prenons un exemple où le programme se contente d'afficher les valeurs passées sur la ligne de commande.

Listing 8.4 – (lien vers le code brut)

```
1  public class LigneCommande{
2      public static void main(String[] args){
3          for (int i=0; i < args.length; i++){
4              Terminal.ecrireStringln(args[i]);
5          }
6      }
7  }
```

Voici un exemple d'exécution :

```
> java LigneCommande un deux trois
un
deux
trois
```

La tableau `args` dans cette exécution a trois cases. Sa valeur est

0	1	2
¨un¨	¨deux¨	¨trois¨

Notons que même si l'on passe un nombre en paramètre, celui-ci est contenu dans le tableau sous forme d'une chaîne.

```
> java LigneCommande un 12 56 deux
un
```

```
12
56
deux
```

La tableaux args vaut

0	1	2	3
¨un¨	¨12¨	¨56¨	¨trois¨

Si l'on veut transformer cette chaîne en un entier, il faut utiliser une fonction de conversion.

8.4 Conversion entre chaînes et autres types

Pour les chaînes de caractères, il n'existe pas de conversion explicite avec d'autres types de données.

Par exemple, si l'on essaie d'affecter une valeur de type `String` à une variable de type `int` comme dans l'exemple suivant, cela produit une erreur.

Listing 8.5 – (lien vers le code brut)

```
1 public class StringInt{
2     public static void main(String[] args){
3         int x;
4         String s = "12";
5         x = s;
6     }
7 }
```

A la compilation, on obtient l'erreur suivante :

```
> javac StringInt.java
StringInt.java:5: incompatible types
found   : java.lang.String
required: int
        x = s;
            ^
1 error
```

Pour réaliser la conversion, il faut utiliser la méthode `Integer.parseInt` et lui donner en paramètre la chaîne à convertir.

Listing 8.6 – (lien vers le code brut)

```
1 public class StringInt2{
2     public static void main(String[] args){
3         int x;
4         String s = "12";
5         x = Integer.parseInt(s);
6         Terminal.ecrireIntln(x);
7     }
8 }
```

Pour convertir une valeur de type double, il faut utiliser la méthode `Double.parseDouble` et pour le type boolean, la méthode `Boolean.parseBoolean`.

Pour convertir dans l'autre sens, un int en chaîne, le plus simple est d'utiliser l'opérateur de concaténation : `""+12` (pour les doubles `""+12.3`, pour les booléens `""+true`). On concatène la chaîne vide avec la valeur à convertir.