JAVA

Chaînes de caractères

EL GHAZI

:: Chaînes de caractères ::

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 les classes de chaîne principales **: String, StringBuffer.**

STRING

Le type de chaîne Java le plus simple est un objet de la classe String. Ces objets sont immuables, **ce qui signifie qu'ils ne peuvent pas être modifiés.**

Exemple : Un objet string simple

Ce programme imprime certaines propriétés d'un objet String nommé alphabet.

```
1  package com.elghazi.basics;
2  public class StringExemples {
3  public static void main (String[] args) {
4  String str = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
5  System.out.println("Cette chaîne est "+ str);
6  System.out.println("Sa longueur est"     + str.length());
7  System.out.println("Le caractère de l'index 4 est "+str.charAt(4));
8  System.out.println("L'index du caractère Z est "+ str.indexOf('Z'));
9  System.out.println("Sa version en minuscules est "+
   str.toLowerCase());
10 System.out.println("La chaîne est toujours"    +     str);
            }
11 }
```

Sortie :

```
Cette chaîne est ABCDEFGHIJKLMNOPQRSTUVWXYZ

Sa longueur est 26

Le caractère de l'index 4 est E

L'index du caractère Z est 25

Sa version en minuscules est abcdefghijklmnopqrstuvwxyz

La chaîne est toujours ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

- L'instruction de sortie de la ligne 5 se contente d'imprimer l'objet string entant que chaîne.
- L'instruction de sortie de la ligne 6 appelle **la méthode length ()** afin d'imprimer le nombre de caractères qui composent la chaîne, à savoir 26.
- L'instruction de sortie de la ligne 7 appelle la méthode charAt () pour imprimer le caractère se trouvant à l'index 4 de la chaîne, c'est-à-dire la lettre E. Vous remarquerez qu'il s'agit en fait du cinquième caractère de la chaîne. En effet, le numéro d'index d'un caractère dans une chaîne correspond toujours au nombre de caractères qui le

précèdent. Dans le cas présent, la lettre E a le numéro d'index 4 dans cette chaîne parce qu'elle est précédée de 4 caractères (A, B, C et D).

- L'instruction de sortie de la ligne 8 appelle la méthode indexof () pour imprimer le numéro d'index de la lettre z dans la chaîne str. Il s'agit du numéro 25 puisque le Z est précédé de 25 caractères dans la chaîne.
- L'instruction de sortie de la ligne 9 appelle la méthode toLowerCase (), ce qui génère une copie d'une nouvelle chaîne temporaire dans laquelle toutes les lettres majuscules ont été transformées en minuscules.
- L'instruction de sortie de la ligne 10 vérifie si la chaîne a été changée. La version en minuscules est une chaîne temporaire, distincte et indépendante de l'originale, qui est générée par la méthode toLowerCase().

MÉTHODES DE LA CLASSE STRING :

Plus d'informations sur ce lien :

https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/lang/String.html

- char charAt(int index)
  - Retourne le caractère qui se trouve à index. (Le premier caractère se trouve à l'index 0.)
  - Cette méthode renvoie une exception de type IndexOutOfBoundsException.
- int compareTo(String chaine_a_comparer)
  - Compare deux String alphabétiquement.
- int compareToIgnoreCase(String chaine)
  - Compare deux String alphabétiquement, en ignorant la casse.
- boolean contains(char s)
  - Retourne true seulement si le String contient la même séquence de caractères.
  - Cette méthode renvoie une exception de type **NullPointerException.**
- boolean startsWith(String prefix)
  - Renvoie true si la String commence par prefix.
- boolean endsWith(String suffixe)
  - Retourne true si la String se termine par suffixe.
- boolean equals(Object autre_obj)
  - Compare la String avec autre_obj.
- int indexOf(int ch [, int index])
  - Retourne l'index de la première occurrence du caractère ch.
  - Si index est indiqué, la recherche commence à partir de cet index.
- int indexOf(String s [, int index])
  - Retourne l'index de la première occurrence de la String s.
  - Si index est indiqué, la recherche commence à partir de cet index.
- int lastIndexOf(int ch [, int fromIndex])
  - Retourne l'index de la dernière occurrence du caractère ch.
  - Si index est indiqué, la recherche commence à partir de cet index.
- int lastIndexOf(String str [, int fromIndex])
  - Retourne l'index de la dernière occurrence de la String s.
  - Si fromIndex est indiqué, la recherche commence à partir de cet index.
- int length()
  - Retourne la taille de la String.
- boolean isEmpty()
  - Retourne true si la taille de la String est égale à 0.
- replace(char oldChar, char newChar)

- Renvoie un nouvel objet String avec les remplacements effectués.

- trim()
  - Renvoie un nouvel objet String avec les espaces enlevés à chaque extrémité. Utilise l'ancienne String si aucun changement n'a été effectué.

- String[] split(paramètres) :
  - découpe une chaîne de caractères à l'aide d'une expression régulière passée en paramètre. Le résultat du découpage est retourné dans un tableau de String.
- char[] toCharArray() :
  - retourne un tableau de char correspondant à cette chaîne de caractères.
- ...

LES SOUS-CHAÎNES :

public String substring(int BeginIndex)

public String substring(int BeginIndex, int endIndex)

Une sous-chaîne est une chaîne dont les caractères forment une partie contiguë d'une autre chaîne. Par exemple, la chaîne EFG est une sous-chaîne de la chaîne ABCDEFGHIJ.

La classe string comprend une méthode substring () qui permet d'obtenir les sous-chaînes données d'une chaîne donnée :

```
1. package com.elghazi.basics;

2. public class StringExemple2 {
3. public static void main(String[] args)
4. {
5. String alpha = "ABCDEFGHIJKLMNOP";
6. System.out.println("alpha: " + alpha);
7. System.out.println("alpha.length(): " + alpha.length());
8. String sub = alpha.substring(2, 7);
9. System.out.println("sub=alpha.substring(2, 7): " + sub);
10.      System.out.println("sub.length: " + sub.length());
11.      System.out.println("sub.charAt(3): " + sub.charAt(3));
12.      System.out.println("alpha.charAt(3): " + alpha.charAt(3));
13.      sub= alpha.substring(4);
14.      System.out.println("sub=alpha.substring(4): " + sub);
15.      System.out.println("sub.length: " + sub.length());
16.      System.out.println("sub.charAt(4): " + sub.charAt(4));
17.      }
18.      }
```

Sortie

```
alpha: ABCDEFGHIJKLMNOP
alpha.length (): 16
sub=alpha.substring(2, 7): CDEFG
sub.length: 5
sub.charAt(3): F
alpha,charAt(3): D
sub=alpha.substring(4): EFGHIJKLMNOP
sub.length: 12
sub.charAt(4): I
```

Ce programme définit une chaîne nommée alpha à la ligne 5. Il l'imprime ainsi que sa longueur aux lignes 6 et 7. La chaîne est composée de 16 caractères,

A la ligne 8, une chaîne nommée sub est définie comme sous-chaîne d'alpha. Cette sous-chaîne est comprise dans un intervalle allant de 2 à 7; il s'agit donc de "CDEFG ". Elle est en fait composée des caractères situés aux index 2, 3, 4, 5 et 6 de la chaîne alpha.

Vous remarquerez que l'intervalle (2, 7) fait référence aux index de 2 à 6 inclus. Tous les intervalles d'index en Java commencent au premier numéro spécifié et se terminent un numéro avant le dernier. Le nombre d'éléments qui composent l'intervalle est égal à la différence entre les deux numéros d'index spécifiés. Par exemple, la sous-chaîne contient 5 =7-2 .

À la ligne 12, nous testons la méthode charAt() sur la chaîne sub. Le caractère situé à l'index 3 dans cette chaîne est 'F' parce qu'il est précédé de 3 caractères. Cela prouve que sub est totalement indépendante de la chaîne originale et que vous avez affaire à deux objets distincts, chacun ayant sa propre indexation.

La sous-chaîne est redéfinie à la ligne 13 à l'aide de la deuxième version de la méthode substring() qui prend un seul paramètres beginIndex et qui retourne la sous chaîne extraite a partir du l'index beginIndex jusqu'à la fin .

L'OPERATEUR D'EGALITE

En Java, l'opérateur d'égalité ==, permet de tester si deux références pointent vers le même objet. Par exemple, si s1 et s2 sont deux références string, l'expression s1 == s2 est vraie uniquement si s1 et s2 sont synonymes, c'est-à-dire s'il s'agit de noms différents d'un même objet, comme illustré à l'exemple :

```
1. package com.elghazi.basics;

2. public class StringExemple3 {
3. public static void main(String [] args) {
4. String s1="ABC";
5. String s2="ABC";
6. String s3=new String("ABC");
7. String s4=new String("ABC");
8. System.out.println("s1 == s2 " + (s1==s2));
9. System.out.println("s2 == s3 " + (s2==s3));
10.      System.out.println("s3 == s4 " + (s3==s4));

11.      }
12.      }
```

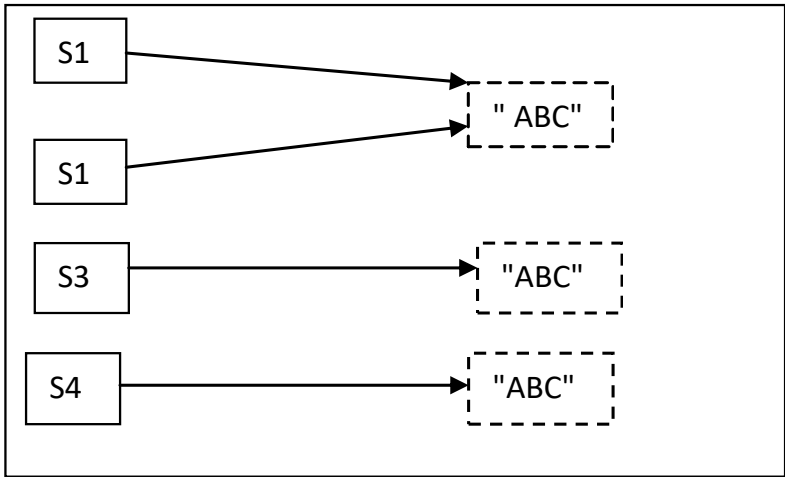
Sortie

```
s1 == s2 true

s2 == s3 false

s3 == s4 false
```

Lignes 4 et 5, nous définissons les références s1 et s2 au littéral string nommé "ABC". La sortie de la ligne 8 indique que ces deux références sont synonymes et pointent toutes les deux vers le même objet :

Vous obtenez ce résultat parce que les littéraux string comme "ABC" sont considérés comme des constantes uniques en Java. Étant donné que les chaînes sont immuables, il n'est pas nécessaire de les copier. Il est nettement plus efficace de créer un seul littéral String nommé "ABC".

La ligne 6 :L'opérateur new vous permet de forcer la création d'un nouvel objet. Par conséquent, même si elle a une valeur identique à, celle qui est référencée par s1 et s2, cette chaîne est un objet distinct. Ce résultat est testé à la ligne 8, dont la sortie vérifie si la chaîne référencée par s 3 est bien différente de celle référencée par s2. Les deux chaînes ont les mêmes valeurs, mais elles occupent des emplacements mémoire différents. Il s'agit effectivement d'objets distincts.



Remarque :

Pour comparer le contenu de deux objets (deux String) on utilise la méthode equals

s3.equals(s1) return True.

RECHERCHE D'UNE CHAINE

Les méthodes indexof () et lastIndexof () de la classe string renvoie le numéro d'index d'un caractère inclus dans une chaîne.

```
12 package com.elghazi.basics;

13 public class StringExemple4 {

14 public static void main(String[] args) {
15 String str ="This is the Mississippi River.";
16 System.out.println(str);
17 int i = str.indexOf('s');
18 System.out.println("Le premier index de 's' est "+ i);
19 i = str.indexOf('s', i+1);
20 System.out.println("L'index suivant de 's' est "+ i);
21 i = str.indexOf('s', i+1);
22 System.out.println("L'index suivant de 's' est "+ i);
23 i = str.lastIndexOf('s');
24 System.out.println("Le dernier index de 's' est "+ i);
25 System.out.println(str.substring(i));
26 }
```

```
27 }
```

Sortie :

```
This is the Mississippi River.
Le premier index de 's' est 3
L'index suivant de 's' est 6
L'index suivant de 's' est 14
Le dernier index de 's' est 18
sippi River.
```

REPRESENTATION D'UNE VALEUR PRIMITIVE DANS UNE CHAINE :

La méthode statique de "String", valueOf(), renvoie la valeur en String de l'argument.

- static String valueOf(boolean b)
  - Returns the string representation of the boolean argument.
- static String valueOf(char c)
  - Returns the string representation of the char argument.
- static String valueOf(char[] data)
  - Returns the string representation of the char array argument.
- static String valueOf(char[] data, int offset, int count)
  - Returns the string representation of a specific subarray of the char array argument.
- static String valueOf(double d)
  - Returns the string representation of the double argument.
- static String valueOf(float f)
  - Returns the string representation of the float argument.
- static String valueOf(int i)
  - Returns the string representation of the int argument.
- static String valueOf(long l)
  - Returns the string representation of the long argument.
- static String valueOf(Object obj)
  - Returns the string representation of the Object argument.

STRINGBUFFER

Etant donné que les objets String sont immuables, vous ne pouvez pas manipuler les caractères d'un objet String de manière directe. Si vous avez besoin d'objets modifiables utilisez plutôt java .lang.StringBuffer :

CONSTRUCTEURS:

- StringBuffer()
  - Construit un objet vide avec une capacité initiale de 16
- StringBuffer(int n)
  - Construit un objet vide avec une capacité initiale de n
- StringBuffer(String)
  - construit un objet à partir d'une String

Exemples :

```
1. package com.elghazi.basics;

2. public class StringBufferExemples {
3. public static void main (String[]args)
4. {
5. StringBuffer s=new StringBuffer();
6. StringBuffer st=new StringBuffer(100);
7. StringBuffer str=new StringBuffer("Bonjour");
8. System.out.println("s:= " + s);
9. System.out.println("st:= " + st);
10. System.out.println("str:= " + str);
11. System.out.println("s.length():= " + s.length());
12. System.out.println("st.length():= " + st.length());
13. System.out.println("str.length():= " + str.length());
14. System.out.println("s.capacity():= " + s.capacity());
15. System.out.println("st.capacity():= " + st.capacity());
16. System.out.println("str.capacity():= " + str.capacity());
17. } }
```

Sortie

```
s:=
st:=
str:= Bonjour
s.length():= 0
st.length():= 0
str.length():= 7
s.capacity():= 16
st.capacity():= 100
str.capacity():= 23
```

- Ligne 5: création d'une chaîne StringBuffer vide
- Ligne 6: création d'une chîne StringBuffer vide de capacité 100
- Ligne 7: création d'une chaîne StringBuffer initialisée avec la chaîne "Bonjour".
- Lignes 8, 9,10: affichage des StrignBuffer.
- Lignes 11, 12,13 : affichage des tailles.

- Lignes 14, 15,16 : affichage des capacités, par défaut une chaîne StringBuffer a une capacité 16.

Méthodes:

- public int length ()
  - Renvoie le nombre de caractères de la chaîne.
- public int capacity ()
  - Renvoie la capacité courante de la chaîne, représentant le nombre de caractères qu'il est possible d'insérer, avant qu'il ne soit alloué de l'espace supplémentaire.
- public char charAt (int index)
  - Renvoie le caractère mémorisé à l'indice index
- public void setLength (int newLength)
  - Modifie la taille de la chaîne. Si la chaîne mémorisée est plus longue, elle sera tronquée. Les caractères éventuellement ajoutés sont nuls ('\u0000'), pour que length () renvoie la valeur newLength .
- public StringBuffer append (Object obj)
- public StringBuffer append (String str)
- public StringBuffer append (char str [ ])
  - Ajoute en fin de chaîne un objet, une chaîne ou un tableau de caractères.
- public StringBuffer append (int i)
  - Ajoute en fin de chaîne la valeur d'un type primitif convertie en chaîne.
- public StringBuffer reverse ()
  - Inverse les caractères d'une chaîne.
- public StringBuffer insert(int offset, int i)
  - Insère à l'indice offset d'une chaîne la valeur d'un type primitif convertie en chaîne.

Exemples :

// Crée un tampon StringBuffer à partir d'une chaîne de caractères

```
StringBuffer b = new StringBuffer("N'est");
```

// Extrait et définit des caractères individuels du tampon StringBuffer

```
char c = b.charAt(0); // Retourne 'N'      équivalent à String.charAt()
```

```
b.setCharAt(0, 'C');  //  b est modifié et contient la valeur

// "C'est" on ne peut pas faire cela avec un objet String !
```

// Ajoute des données à un objet StringBuffer

```
b.append(' ');    // Ajoute un caractère

b.append("le moment.");          // Ajoute une chaîne de caractères

b.append(23);  // Ajoute un entier ou une valeur quelconque
```

// Insère des chaînes de caractères ou d'autres valeurs dans un tampon StringBuffer

```
b.insert(6, "pas");//b contient maintenant       "C'est pas le !! moment.23"
//Remplace un sous-ensemble de caractères avec une chaîne de caractères donnée
b.replace(2, 9, "est");   // Retour à "C'est le moment.23"
```

//Supprime des caractères

```
b.delete(15,18);          //Supprime un sous-ensemble :"C'est le moment"
b.deleteCharAt(2);        //Supprime le deuxième caractère :"C'st le moment
b.setLength(4);           // Tronque selon la taille donnée :"C'st "
```

Autres opérations utiles

```
b.reverse();     // Inverser les caractères :"ts'C"
String s=b.toString();          //reconversion vers une chaîne immuable
b.setLength(0);          //Ecrase le tampon; il est désormais prêt a être réutilisé.
```