Leçon N°1

Les chaines de caractères

Introduction

Nous aborderons dans ce chapitre les objets de la classe String, StringBuffer et StringTokenizer.

I. Les objets de la classe String

I.1. Instanciation d'un objet String

Les chaînes de caractères sont des instances de la classe **String** contenue dans le package **java.lang** et non pas des tableaux de caractères.

Les objets de type String peuvent être définis de plusieurs façons :

- `String maChaine = new String("toto");  // appel explicite du constructeur`
- `String maChaine = "toto";  // méthode simplifiée`
-
```
char[] listeCaracteres = {'t','o','t','o'};  // tableau de caractères
String maChaine = new String(listeCaracteres);
```

Remarques

1. Les objets de type String sont immutables. Il n'existe pas de méthode permettant de modifier directement ces objets. A titre d'exemple, l'instruction suivante génère une erreur de compilation :

```
maChaine[3] = 'a' ;    // Erreur
```

En revanche, il est possible d'utiliser les méthodes qui renvoient une chaîne pour modifier le contenu d'une chaîne donnée.

Exemple

```
import javax.swing.JOptionPane;
public class Chaines {
     public static void main(String[] args) {
          String maChaine = "toto";
          maChaine = maChaine.replace('o','i');
          JOptionPane.showMessageDialog(null,"maChaine = " + maChaine);
     }
}
```

Output du programme

2. Java ne fonctionne pas avec le jeu de caractères ASCII ou ANSI, mais avec Unicode (Universal Code). Ceci concerne les types char et les chaînes de caractères. Le jeu de caractères Unicode code un caractère sur plusieurs octets (8 ou 16) ce qui permet de supporter les caractères accentués, les caractères arabes, chinois, etc. Les caractères 0 à 255 correspondent exactement au jeu de caractères ASCII étendu.

I.2. Principales méthodes de la classe String

Méthode	Description
char charAt(int index)	Renvoie le caractère à la position index. L'index du premier caractère d'une chaine est 0.
boolean equals(String autreChaine)	Renvoie vrai si la chaine sur laquelle est appliquée la méthode est égale à autreChaine
boolean equalsIgnoreCase(String autreChaine)	Compare les deux chaines sans tenir compte de la casse (minuscule et majuscule).
int indexOf(String chn)	Renvoie l'index de la première occurrence de la chaîne chn dans l'objet en cours (ou -1 si la chaine est absente).
String substring(int Depart, int Fin)	Renvoie un String contenant la chaîne de départ de l'index Debut à l'index Fin-1. Autrement dit, Fin-Debut vaut la longueur de la sous-chaine extraite de la chaine actuelle.
char[] toCharArray()	Renvoie un tableau de caractères
String toUpperCase()	Renvoie la version tout en majuscules de la chaîne actuelle
String toLowerCase()	Renvoie la version tout en minuscules de la chaine actuelle
String trim()	Renvoie une chaine valant la chaine de départ sans les blancs en début et fin de chaîne.
static String valueOf(int / boolean / double / float / long)	Renvoie une chaine correspondant à la valeur du primitif donné en argument. Par exemple, valueOf(2) renverra "2". L'opération inverse peut être réalisée par int i=Integer.parseInt("2");

I.2.1. Longueur d'une chaîne

Pour obtenir la longueur d'une chaîne de caractères, il est possible d'utiliser la méthode **length( )** qui renvoie un entier représentant le nombre de caractères de la chaîne. La chaîne vide "" est de longueur zéro.

⚠ Ne pas confondre la méthode **String.length()** avec la propriété **length** d'un tableau.

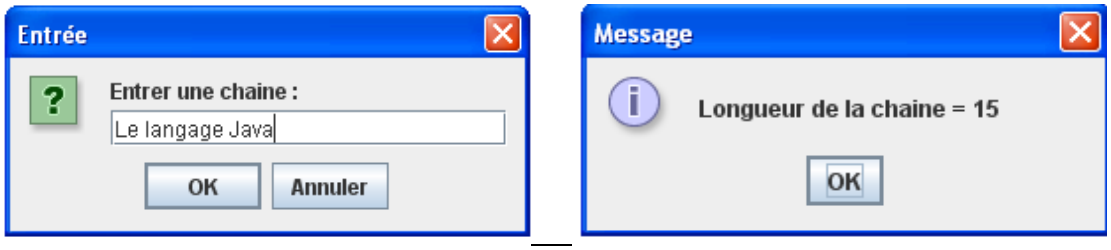
Exercice

Ecrire un programme en Java qui lit une chaine de caractères (en mode graphique) puis affiche sa longueur.

Solution

```
import javax.swing.JOptionPane;
public class Chaines {
   public static void main(String[] args) {
      String chn = JOptionPane.showInputDialog("Entrer une chaine : ");
      JOptionPane.showMessageDialog(null,"Longueur de chn= "+chn.length());
   }
}
```

Trace d'exécution



Remarque : Les caractères d'une chaine s sont situés dans l'intervalle [0..s.length( )-1].

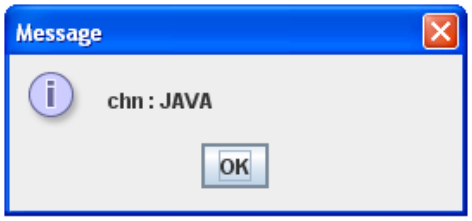
I.2.2 Modification de la casse d'une chaine

Les méthodes Java **toUpperCase( )** et **toLowerCase( )** permettent respectivement d'obtenir une chaîne tout en majuscules ou tout en minuscules.

Exemple

```
import javax.swing.JOptionPane;
public class Chaines {
      public static void main(String[] args) {
            String chn = "Java";
            JOptionPane.showMessageDialog(null,"chn : "+ chn.toUpperCase());
      }
}
```

Output du programme



I.2.3. Accès à un caractère dans une chaine

La méthode **charAt(int)** permet d'accéder individuellement à chaque caractère d'une chaine à partir de son index. Le premier caractère d'une chaîne non vide `s` est `s.charAt(0)`.

Exemple

```
String s = "une chaîne";
char c = s.charAt(1);                    // c = 'n'
```

⚠ La tentative d'accès à un caractère en dehors de l'intervalle [0 .. s.length( )-1] provoque la levée d'une exception *StringIndexOutOfBoundsException.*

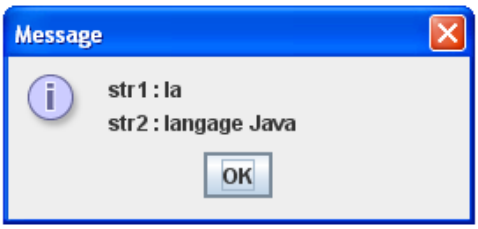
I.2.4. Extraction d'une sous-chaîne

La méthode **substring(IndexDébut, IndexFin)** permet d'extraire la chaine comprise dans l'intervalle[Indexdébut .. IndexFin-1]. Si l'index de fin n'est pas précisé l'extraction se fait jusqu'à la fin de la chaine.

Exemple

```
import javax.swing.JOptionPane;
public class Chaines {
      public static void main(String[] args) {
            String chn = "le langage Java";
            String str1 = chn.substring(3,5);
            String str2 = chn.substring(3);
            JOptionPane.showMessageDialog(null,"str1 : "+ str1 + "\nstr2 : "+
            str2);
      }
}
```

Output du programme



I.2.5. Recherche d'une sous-chaîne

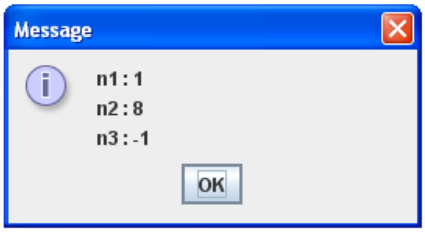
Les méthodes **indexOf( )** et **lastIndexOf( )** permettent de rechercher une sous-chaine de caractères dans une chaine. La 1ère méthode retourne l'indice de la 1ère occurrence de la sous-chaine alors que la 2ème méthode retourne l'indice de la dernière occurrence.

Si la sous-chaine n'existe pas dans la chaine, la méthode renvoie (-1).

Exemple

```
import javax.swing.JOptionPane;
public class Chaines {
   public static void main(String[] args) {
      String chn = "une chaine";
      int n1 = chn.indexOf("ne");
      int n2 = chn.lastIndexOf("ne");
      int n3 = chn.indexOf("A");
      JOptionPane.showMessageDialog(null,"n1 : "+ n1 + "\nn2 : "+ n2+"\nn3 : "+n3);
   }
}
```

Output du programme



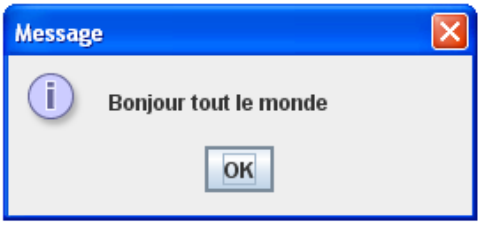
I.3. Concaténation de chaînes de caractères

En Java, l'opérateur '+' permet de concaténer plusieurs chaînes de caractères. Il est également possible d'utiliser l'opérateur +=.

Exemple

```
import javax.swing.JOptionPane;
public class Chaines {
       public static void main(String[] args) {
            String texte = "Bonjour"+" "+"tout";
            texte += " le monde";
            JOptionPane.showMessageDialog(null, texte);
      }
}
```

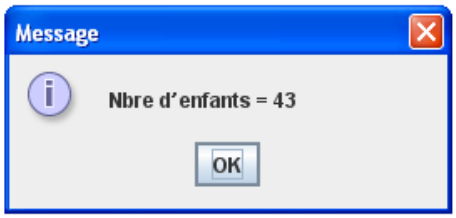
Output du programme



L'opérateur '+' sert aussi à concaténer des chaînes avec tous les types de base. La variable ou constante est alors convertie en chaîne puis ajoutée à la précédente. La condition préalable est d'avoir au moins une chaîne dans l'expression sinon le signe '+' est évalué comme opérateur mathématique. On dit que l'opérateur '+' connaît un *polymorphisme paramétrique* puisqu'il ne se comporte pas de la même façon en fonction du type de ses deux arguments.
Exemple

```
import javax.swing.JOptionPane;
public class Chaines {
     public static void main(String[] args) {
          JOptionPane.showMessageDialog(null," Nbre d'enfants = " + 4 + 3);
     }
}
```

Output du programme



Remarque: la concaténation de deux chaines peut se faire en appelant la méthode « concat » comme dans l'exemple suivant :

```
String s = "long".concat("temps");
```

I.4. Égalité de deux chaînes

Pour comparer deux chaines de caractères, il ne faut pas utiliser l'opérateur « = = » car dans ce cas, ce sont les deux références (c.-à-d. les adresses mémoires des 2 objets) qui sont

comparées, mais il faut recourir à la méthode **equals()** (qui fait la comparaison caractère par caractère) comme dans deux exemples suivants :

Exemple 1

```
import javax.swing.JOptionPane;
public class Chaines {
   public static void main(String[] args) {
      String chn1 = new String("Java");
      String chn2 = new String("Java");
      if (chn1 == chn2)
         JOptionPane.showMessageDialog(null,"les 2 chaines sont identiques");
      else
         JOptionPane.showMessageDialog(null,"les 2 chaines sont différentes");
   }
}
```

Output du programme

Exemple 2

```
import javax.swing.JOptionPane;
public class Chaines {
   public static void main(String[] args) {
      String chn1 = new String("Java");
      String chn2 = new String("Java");
      if (chn1.equals(chn2))
        JOptionPane.showMessageDialog(null,"les 2 chaines sont identiques");
      else
        JOptionPane.showMessageDialog(null,"les 2 chaines sont différentes");
   }
}
```

Output du programme

I.5. le méthode toString()

Toutes les classes depuis « Object » possèdent une méthode toString() qui renvoie un String :

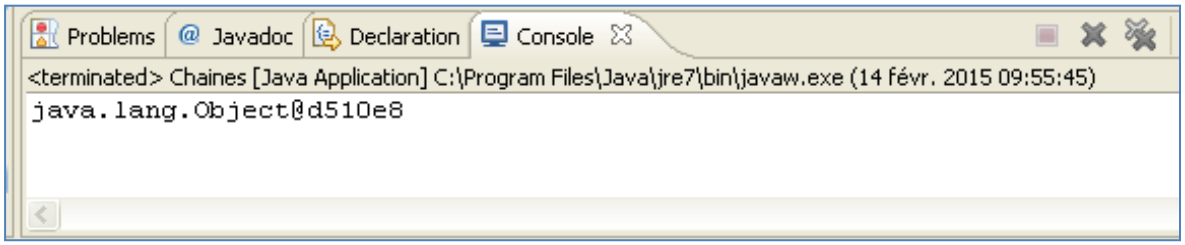
String toString()

Cette méthode sert à renvoyer une chaîne de caractère donnant des informations sur l'objet courant sous la forme d'une chaîne de caractères. Par exemple, la méthode toString( ) de la classe Object renvoie simplement le *hashcode* de l'objet.

Exemple

```
Object o = new Object();
System.out.println(o.toString());
```

Output



Il est ensuite possible de surcharger cette méthode pour donner davantage d'informations.

La méthode println() de de la classe « OutputStream » qui est très utilisée exécute toujours la méthode toString() et affiche la chaîne renvoyée.

donc faire:

```
Integer i = new Integer(2); //Integer : classe enveloppe (wrapper)
System.out.println(i);
```

est équivalent à :

```
System.out.println(i.toString());
```

Exercice : Créer une classe «**TestChaines »** qui contient 5 méthodes (toutes de type static) :

- **String Inverser(String chn)** : retourne l'inverse de la chaine chn
- **boolean Palind(String chn)** : vérifie si la chaine chn est palindrome (se lit dans les 2 sens) ou non
- **int Compter(char c, String chn)** : compte combien de fois le caractère c apparait dans la chaine chn
- **int CompterMots(String chn)** : compte le nombre de mots contenus dans la chaine chn
- **public static void main (String[] args)** : créé une chaine puis teste les différentes méthodes de la classe.

Solution

```
import javax.swing.JOptionPane;

public class TestChaines {

      public static String inverser(String s){
            String r = "";
            for (int i = 0; i < s.length();i++)
                  r=s.charAt(i)+r;
            return r;
      }
```

```
    public static boolean palind(String s){
        String s_inv = inverser(s);
        if (s.equals(s_inv))
            return true;
        else
            return false;
    }

    public static int compter(char c, String s){
        int nb = 0;
        for (int i = 0; i < s.length(); i++)
            if (s.charAt(i) == c)
                nb++;
        return nb;
    }

    public static int compterMots(String s){
        if (s.length() > 0)
            return compter(' ',s)+1;
        else
            return 0;
    }
    public static void main(String[] args) {
        String chn = JOptionPane.showInputDialog("Entrer une chaine");
        if (palind(chn))
            JOptionPane.showMessageDialog(null,"Palindrome");
        else
            JOptionPane.showMessageDialog(null,"Non Palindrome");
        JOptionPane.showMessageDialog(null,"Nbre de a : "+compter ('a',chn));
        JOptionPane.showMessageDialog(null,"Nbre de mots :"+compterMots(chn));
    }
}
```

II. Les objets de la classe StringBuffer

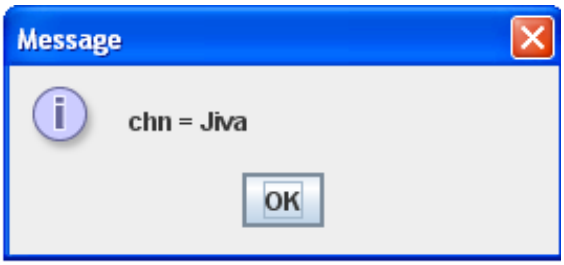
II.1. Particularités

Un objet de type StringBuffer est une chaîne de caractères modifiable qui présente certains avantages de manipulation et de performance. La classe StringBuffer possède en effet des méthodes permettant d'effectuer des modifications sur l'objet lui-même. Par exemple, ce qui était impossible pour le String (modifier un caractère de la chaîne) est possible pour le StringBuffer grâce à sa méthode setCharAt( ).

Exemple

```
import javax.swing.JOptionPane;
public class Chaines {
   public static void main(String[] args) {
      StringBuffer chn = new StringBuffer("Java");
      chn.setCharAt(1,'i');
      JOptionPane.showMessageDialog(null,"chn = " + chn);
   }
}
```

Output



II.2. Principales méthodes de la classe StringBuffer

Méthode	Description
char charAt(int index)	Renvoie le caractère à la position index. L'index du premier caractère d'une chaine est 0.
void setCharAt(int index, char car)	Mettre le caractère à l'index spécifié de la chaine actuelle au caractère donné en argument.
StringBuffer deleteCharAt(int index)	Supprime le caractère qui se trouve à la position index
void delete(int Début, int Fin)	Supprime les caractères de la chaîne de Début à Fin-1
StringBuffer insert(int Index, String str)	Insère la chaîne str à l'index précisé.
public StringBuffer append(Object obj)	Concatène la représentation textuelle de l'objet obj à la fin de la chaine actuelle
StringBuffer replace(int Debut, int Fin, String str)	Remplace la partie de la chaine comprise entre Debut et Fin-1 par la chaine str.
String substring(int Depart, int Fin)	Ne conserve de la chaîne que les caractères de l'index Debut à l'index Fin-1. Autrement dit, Fin-Debut vaut la longueur de la nouvelle chaine.
StringBuffer reverse( )	Inverse la valeur de l'objet StringBuffer qui a appelé la méthode.
String toString()	Renvoie l'objet de type String correspondant à cette chaîne.

Exemple 1 : Que va afficher le programme suivant :

```
public class StringBuf {
      public static void main (String[] args){
            StringBuffer chn = new StringBuffer("algo");
            chn.replace(0, 3, "Styl");
            System.out.println(chn.reverse());
      }
}
```

Output

```
olytS
```

Exemple 2 : Que fait la méthode « invert( ) » définie ci-dessous ?

```
class TestStringBuffer{
  public static String invert(String source) {
     int len = source.length();
     StringBuffer dest = new StringBuffer(len);
     for (int i = (len-1);i >= 0; i--) {
```

```
            dest.append(source.charAt(i));
        }
        return dest.toString();
    }
}
```

III. La classe StringTokenizer

Cette classe du package java.util sert à couper une chaîne en sous-chaînes de façon rapide. Le constructeur le plus courant est:

StringTokenizer(String input, String délimiteur)

- N'importe quelle chaîne peut servir de séparateur (des espaces, des slashs, tirets...). L'espace est le séparateur par défaut.
- La méthode **countTokens( )** retourne le nombre d'éléments restant à lire dans la chaine source.
- La méthode **nextToken( )** renvoie un String contenant la sous-chaîne suivante.
- La méthode **hasMoreTokens( )** renvoie « true » s'il reste des sous-chaînes à lire.

Exemple

```
import java.util.StringTokenizer;

public class StringTok {
      public static void main(String[] args) {
            String monTexte="Le langage Java";
            StringTokenizer st=new StringTokenizer(monTexte," ");
            System.out.println("Nbre de tokens : "+st.countTokens());
            StringBuffer sortie = new StringBuffer();
            while(st.hasMoreTokens()){
                  sortie.append(st.nextToken()+"\n");
            }
            System.out.println(sortie);
      }
}
```

Output

```
Nbre de tokens : 3
Le
langage
Java
```