Chaînes de caractères

Introduction

Les chaînes de caractères sont représentées en Java par les classes **`String`** ou **`StringBuffer`.** Ces classes sont définies dans java.lang et peuvent être utilisées dans n'importe quel programme sans les importer.

Les objets String représentent des chaînes constantes et les objets StringBuffer sont des chaînes modifiables. Java effectue cette distinction entre chaînes constantes ou modifiables pour des raisons de pure optimalisation; en particulier, Java s'autorise certaines optimalisations portant sur des objets String, telles que le partage d'un même objet String parmi plusieurs références, parce qu'il sait que ces objets ne changeront pas.

Lorsque vous avez le choix d'utilisation entre un objet String et un objet StringBuffer pour représenter une chaîne de caractères, préférez toujours l'objet String si vous êtes certain que cette chaîne de caractères ne changera pas. Ce choix améliore considérablement les performances.

Création d'un objet String

Une chaîne est un objet qu'on peut initialiser soit par initialisation, soit à l'aide de l'opérateur **`new`.**

En fait, la classe String fournit 9 constructeurs pour créer des objets String (voir exemple de Deitel page 539).

Exemples :

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

Principales méthodes de la classe String

Les méthodes du tableau suivant (sauf la dernière) sont des méthodes d'instance. Il faut donc se créer un objet String, par exemple *chaine*, pour pouvoir appeler ces méthodes d'instance. L'appel d'une de ces méthodes, appliquée à l'objet *chaine***,** se fera à l'aide de :

`chaine.nomMéthode(arguments)`

Nom de la méthode	But de la méthode
char **charAt**(int ind)	retourne le caractère de la chaîne à la position *ind*
int **length**()	retourne la longueur de la chaîne
int **compareTo**(String autre)	compare *chaine* avec *autre* et renvoie une valeur négative, nulle ou positive suivant que *chaine* est respectivement plus petite, égale ou plus grande que *autre*
boolean **equals**(Objet unObjet)	compare *chaine* à *unObjet* et renvoie *true* si *unObjet* est de la classe String et que les deux chaînes sont les mêmes
boolean **equalsIgnoreCase**(String autre)	compare les chaînes en ignorant la différence entre

	minuscules et majuscules
boolean **startsWith**(String autre)	teste si *chaine* débute avec la chaîne *autre*
boolean **startsWith**(String autre, int ind)	teste si *chaine* débute avec la chaîne *autre* à partir de la position *ind*
boolean **endsWith**(String autre)	teste si *chaine* se termine par la chaîne *autre*
int **indexOf**(char ch)	renvoie la position de la première occurrence de *ch* dans *chaine* ou -1 si pas trouvé
int **indexOf**(char ch, int ind)	renvoie la position de la première occurrence de *ch* dans *chaine* à partir de *ind* ou -1 si pas trouvé
int **indexOf**(String autre)	renvoie la position de la première occurrence de *autre* dans *chaine* ou -1 si pas trouvé
int **indexOf**(String autre, int ind)	renvoie la position de la première occurrence de *autre* dans *chaine* à partir de la position *ind* ou -1 si pas trouvé
int **lastIndexOf**(char ch)	renvoie la position de la dernière occurrence de *ch* dans *chaine* ou -1 si pas trouvé
int **lastIndexOf**(char ch, int ind)	renvoie la position de la dernière occurrence de *ch* dans *chaine* à partir de la position *ind* en reculant ou -1 si pas trouvé
int **lastIndexOf**(String autre)	renvoie la position de la dernière occurrence de *autre* dans *chaine* ou -1 si pas trouvé

int **lastIndexOf**(String autre, int ind)	renvoie la position de la dernière occurrence de *autre* dans *chaine* à partir de la position *ind* en reculant ou -1 si pas trouvé
String **substring**(int ind)	renvoie une nouvelle chaîne contenant une copie des caractères de *chaine* à partir de la position *ind* jusqu'à la fin de *chaine*
String **substring**(int ind1, int ind2)	renvoie une nouvelle chaîne contenant une copie des caractères de *chaine* à partir de la position *ind1* jusqu'à la position *(ind2 - 1)* incluse. La nouvelle chaîne contiendra *(ind2 - ind1)* caractères
String **replace**(char ch, char car)	renvoie une nouvelle chaîne dont toutes les occurrences de *ch* dans *chaine* sont remplacées par *car*. La chaîne originale demeure inchangée
String **toLowerCase**()	renvoie une nouvelle chaîne comprenant les caractères de *chaine* convertis en minuscules. La chaîne originale demeure inchangée
String **toUpperCase**()	renvoie une nouvelle chaîne comprenant les caractères de *chaine* convertis en majuscules. La chaîne originale demeure inchangée
String **trim**()	renvoie une nouvelle chaîne privée des espaces présents au début et à la fin de *chaine*. La chaîne

	originale demeure inchangée
String **toString**()	retourne la chaîne originale
static String **valueOf**(argument)	renvoie une chaîne représentant la valeur de l'argument, par exemple : `str = String.valueOf(12)` retourne la chaîne "*12*" et la place dans *str* `str = String.valueOf('a')` retourne la chaîne "*a*" et la place dans *str*

Exemple Questionnaire

```
public class Questionnaire
{
   static BufferedReader clavier = new BufferedReader(new InputStreamReader

                                        (System.in));

   public static void main(String args[]) throws IOException
   {
         String questions[] = {"", "la relativité", "L'Amérique", "l'ordinateur",
                               "la pesanteur", "Java"};
         String reponses[] =  {"", "Einstein", "Colomb", "Pascal",
                               "Archimède", "Sun"};
         int noQuest;
         int nbCorrect = 0;
         String choix;
         boolean correct;

         System.out.println("Numéro\t\tQuestions");
         for (int k = 1; k < questions.length; k++)
               System.out.println("  " + k + "\t\tQui a découvert " + questions[k]);
         do
         {
               System.out.print("\n Choisir un numéro de question : ");
               noQuest = Integer.parseInt(clavier.readLine());
               System.out.print(" Entrer votre réponse : ");
               choix = clavier.readLine();
               System.out.println("\nVotre question est: \t " + questions[noQuest]);
               System.out.print("\nVotre réponse est: \t " + choix);

               correct = choix.equalsIgnoreCase(reponses[noQuest]);
               if (correct)
               {
                     System.out.println("\t  *** correct ***");
                     nbCorrect++;
               }
               else
                     System.out.println("\t  *** incorrect ***");

         } while (encoreJouer());
         System.out.println("\nnombre de réponses correctes : " + nbCorrect);
   }
```

```
   // Demande à l'usager s'il veut encore jouer jusqu'à obtention de O ou N
   // retourne true si l'usager a entré O (minuscule ou majuscule)
   static boolean encoreJouer() throws IOException
   {
         final String LETTRES_VALIDES = "OoNn";
         char car;
         do
         {
               System.out.print("\nVoulez-vous jouer encore (O/N) ? ");
               car = clavier.readLine().charAt(0);
         } while (LETTRES_VALIDES.indexOf(car) == -1);
         return (car == 'o' || car == 'O');
   }
}
```