Les caractères et les chaînes de caractères

Les types

- char
- String

Le type char

1. Les littéraux caractères
2. Déclaration d'une variable de type caractère
3. Affectation
4. Saisie à partir du clavier
5. Affichage à l'écran
6. Code Unicode d'un caractère
7. Transformer un code Unicode en caractère
8. Comparaison
9. Transformations majuscules ou minuscules

Les littéraux caractères

Les valeurs des caractères sont représentés par des littéraux

Les littéraux caractères sont encadrés par des aspostrophes et non par des guillemets

Des caractères visibles ou imprimables

La lettre A majuscule	'A'
La lettre a minuscule	'a'
Le chiffre 1	'1'
L'espace	' '
La virgule	','

Des caractères de contrôle

La fin de ligne (new line)	'\n'
Le retour chariot (carriage return)	'\r'
La tabulation	'\t'
Le caractère apostrophe	'\''

Des caractères en Unicode (généralisation du code Ascii sur 16 bits)

La lettre A majuscule	'\u0041'
La lettre a miniscule	'\u0061'
Le chiffre 1	'\u0031'
La fin de ligne (new line)	'\u000a'
Le retour chariot (carriage return)	'\u000d'
La tabulation	'\u0009'

Déclaration et affectation

Une variable **c** de type caractère doit être déclarée

```
char c ;
```

On peut alors lui affecter un littéral caractère

```
c = 'A' ;
```

ou la valeur d'une autre variable de type caractère

```
char c1 ;
  ...
c = c1 ;
```

Saisie et affichage

Saisie à partir du clavier

```
c = EntreeClavier.readChar ( "? " ) ;
```

Affichage à l'écran en restant sur la ligne

```
System.out.print ( c ) ;
```

Affichage à l'écran en passant à la ligne

```
System.out.println ( c ) ;
```

Code Unicode d'un caractère

Pour obtenir le code unicode d'une variable c de type caractère, il suffit de ranger sa valeur dans une variable entière.

Les instructions suivantes permttent d'afficher le code Ascii de la lettre **A** majuscule.

```
char c ;
c = 'A' ;
int code ;
code = c ;
System.out.println ( code ) ;
```

Obtenir un caractère à partir de son code Unicode

```
int code ;
code = 65 ;
char c ;
c = ( char ) code ;
System.out.println ( c ) ;
```

Le cast (char) permet de transformer la valeur entière de code de 32 bits(4 octets) en une valeur caractère de 16 bits (2 octets).

Comparaison

Les caractères sont considérés comme de valeurs numériques, il est donc possible d'utiliser les opérateurs de comparaison.

L'expression :

```
c == 'Y' || c == 'y'
```

prend la valeur vrai si c est égal à lettre Y majuscule ou minuscule.

L'expression :

```
c >= '0' && c <= '9'
```

prend la valeur vrai si c est un chiffre décimal.

Transformations en majuscules ou en minuscules

La fonction **Character.toUpperCase** retourne la caractère passé en paramètre en majuscule.

Par exemple

```
char c ;
c = 'a' ;
c = Character . toUpperCase ( c ) ;
System.out.println ( c ) ;
```

La fonction **Character.toLowerCase** retourne la caractère passé en paramètre en minuscule.

Exercices

- La fonction estChiffreHexa
- Table Ascii en décimal
- Table Ascii en hexadécimal

Le type String

1. Les littéraux chaînes de caractères
2. Déclaration d'une variable de type chaîne de caractères
3. Affectation
4. Saisie à partir du clavier et affichage à l'écran
5. Longueur d'une chaîne de caractères
6. Accès au caractère de rang i
7. Concaténation
8. Egalité de deux chaînes de caractères
9. Ordre lexicographique
10. Comparaison de deux chaînes de caractères
11. Recherche d'un caractère
12. Conversion d'une chaîne en nombres
13. Conversion d'un nombre en chaîne
14. Extraction d'une sous-chaîne

Les littéraux chaînes de caractères

Les valeurs des chaînes caractères sont représentés par des littéraux

Les littéraux chaînes de caractères sont encadrés par des guillemets et non par des aspostrophes

"RT-1"	4 caractères
""	chaîne vide aucun caractère
"A"	un seul caractère
" "	un espace
"abc\n12"	6 caractères
"\""	un guillemet

Il est très important de ne pas confondre le caractère `'A'` et la chaîne de caractères `"A"`

Déclaration et affectation

Une variable **ch** de type chaîne de caractères doit être déclarée

```
String ch ;
```

On peut alors lui affecter un littéral chaîne de caractères

```
ch = "RT-1" ;
```

ou la valeur d'une autre variable de type chaîne de caractères

```
String ch1 ;
   ...
ch = ch1 ;
```

Saisie et affichage

Saisie à partir du clavier

```
ch = EntreeClavier.readString ( "? " ) ;
```

Affichage à l'écran en restant sur la ligne

```
System.out.print ( ch ) ;
```

Affichage à l'écran en passant à la ligne

```
System.out.println ( ch ) ;
```

Longueur d'une chaîne de caractères

L'expresssion

```
ch . length ( )
```

donne le nombre de caractères (ou longueur) de la chaîne de caractères ch

```
String ch1 ;
ch1 = "RT-1" ;
System.out.println ( ch1 . length ( ) ) ;
String ch2 ;
ch2 = "" ;
System.out.println ( ch2 . length ( ) ) ;
```

Longueur d'une chaîne de caractères (suite)

Ne pas confondre

- la longueur d'une chaîne de caractères
- et le nombre d'éléments d'un tableau

La longueur de la chaîne de caractères **ch** est

```
ch . length ( )
```

Le nombre d'éléments du tableau **a** est

```
a . length
```

Accès au caractère de rang i

L'expresssion

```
ch . charAt ( i )
```

donne le caractère de rang **i** de la chaîne **ch**

Le rang **i** doit être compris entre `0` et `ch . length ( ) -1`.

Dans le cas contraire une exception

```
StringIndexOutOfBoundsException
```

est déclenchée et l'exécution du programme est stoppée.

Remarque : `setCharAt` n'existe pas pour les chaînes de caractères.

Exercice

Ecrire la fonction qui permet d'afficher la chaîne de caractères passée en paramètre à l'envers.

```
public static void afficheEnvers ( String ch ) {

            ...

}
```

Exercice

Ecrire la fonction qui retourne la chaîne de caractères passée en paramètre à l'envers.

```
public static String envers ( String ch ) {

            ...

}
```

Exercice

Ecrire la fonction qui retourne le nombre d'espaces de la chaîne de caractères passée en paramètre.

```
public static int nbEspaces ( String ch ) {

            ...

}
```

Concaténation

L'opérateur + permet de concaténer deux chaînes de caractères.

Par exemple

```
      ...
ch1 = "RT" ;
ch2 = "Béthune" ;
ch1 = ch1 + "␣" + ch2 ;
  // la valeur de ch1 est "RT Béthune"
```

Egalité de deux chaînes de caractères

Il n'est pas possible d'utiliser les opérateurs d'égalité ou d'inégalité avec les chaînes de caractères
L'expression

```
s1 . equals ( s2 )
```

retourne vrai si les deux chaînes sont égales et faux dans le cas contraire.

Egalité de deux chaînes de caractères (suite)

```
    ...
s1 = "ABC" ;
s2 = "abc" ;
System.out.println( s1.equals ( s2 ) ) ;
  // -> false
System.out.println( s1.equalsIgnoreCase ( s2 ) ) ;
  // -> true
```

Ordre lexicographique

La chaîne **s1** précède la chaîne **s2** (**s1** $\prec$ **s2**) **ssi**

- **s1** est un préfixe de **s2**

  bal $\prec$ ballon

- Le premier caractère de **s1** qui ne coïncide pas avec le caractère de même range de **s2** est inférieur à ce dernier

  ballon $\prec$ bas

Comparaison de deux chaînes de caractères

L'expression

```
s1 . compareTo ( s2 )
```

$$\text{retourne une valeur} \begin{cases} < 0 & \text{si} \quad s1 \prec s2 \\ 0 & \text{si} \quad s1 \text{ est égal à } s2 \\ > 0 & \text{si} \quad s1 \succ s2 \end{cases}$$

L'expression

```
s1 . compareToIgnoreCase ( s2 )
```

effectue la même comparaison sans tenir compte des majuscules et minuscules.

Recherche d'un caractère

L'expression

```
s . indexOf ( c )
```

retourne

- la première occurrence du caractère **c** dans la chaîne **s**
- -1 si le caractère **c** n'a pas été trouvé

Recherche d'un caractère

L'expression

```
s . indexOf ( c , i0 )
```

retourne

- la première occurrence du caractère **c** dans la chaîne **s** à partir du rang **i0**
- -1 si le caractère **c** n'a pas été trouvé

Exercice

Réécrire la fonction **nbEspaces** en utilisant la méthode **indexOf**.

```
public static int nbEspaces ( String ch ) {

            ...

}
```

Conversion d'une chaîne de caractères en nombre

L'expression

```
Integer . parseInt ( s )
```

donne la valeur numérique de la chaîne **s**

```
...
s = "125" ;
int v ;
v = Integer . parseInt ( s )
v = 2 * v ;
  // la valeur de v est 250
```

Conversion d'une chaîne de caractères en nombre (suite)

Si la chaîne **s** ne représente pas un nombre entier une exception

`NumberFormatException`

est déclenchée et l'exécution du programme est stoppée.

Conversion d'un nombre en chaîne de caractères

L'expression

```
String . valueOf ( n )
```

donne la chaîne de caractères représentant le nombre n

```
String s ;
s = 125 ;
int n ;
n = Integer . parseInt ( s ) ;
n = 2 * n ;
s = String . valueOf ( n ) ;
```

Extraction d'une sous-chaîne

L'expression

```
s . substring ( iDeb , iFin )
```

retourne une nouvelle chaîne constituée des caractères de **s** dont les rangs sont compris entre **iDeb** et **iFin - 1**

L'expression

```
s . substring ( iDeb )
```

retourne une nouvelle chaîne constituée des caractères de **s** dont les rangs sont compris entre **iDeb** et **s . length ( ) - 1**

Exercice

Ecrire la fonction **heureEnSecondes**

qui admet un paramètre de type chaîne de caractères

de la forme `"HH:MM:SS"`

et retourne le nombre de secondes correspondant.

Par exemple

```
heureEnSecondes ( "01:02:20" )
```

retourne `3740 ( 3600 + 120 + 20 )`

Exercice

Ecrire la fonction **secondesEnHeure**

qui admet un paramètre de type entier le nombre de secondes

et retourne une chaîne de caractères de la forme

de la forme `"HH:MM:SS"`

Par exemple

```
secondesEnHeure ( 3740 )
```

retourne `"01:02:20"`