# Résumé chaînes de caractères

## 1. La classe String

### 1.1. Les principaux constructeurs de la classe String

| | |
|---|---|
| *String()* | Crée une chaîne de caractères vide. |
| *String(char[] value)* | Crée un objet String avec la séquence de caractères passée en argument. |
| *String(String original)* | la nouvelle chaîne créée est une copie de la chaîne passée en argument. |

### 1.2. Longueur d'une chaîne de caractères

| | |
|---|---|
| *int length()* | renvoie la taille d'une chaîne de caractères |

### 1.3. Concaténation de chaînes

L'opérateur + permet de concaténer deux chaines de caractères. (+= aussi)

La concaténation peut également se faire avec la méthode : String concat(String str)

### 1.4. Comparaison des chaînes de caractères

| | |
|---|---|
| *boolean equals(Object anObject)* | Compare la chaîne de caractères avec l'objet passé en paramètre |
| *boolean equalsIgnoreCase(String anotherString)* | Compare la chaîne de caractères avec celle passée en paramètre. mais sans distinguer les majuscules des minuscules |
| *int compareTo(String anotherString)* | effectue des comparaisons lexicographiques de chaînes pour savoir laquelle de deux chaînes apparaît avant une autre, en se fondant sur l'ordre des caractères. |
| *int compareToIgnoreCase(String str)* | Même chose que la méthode précédente mais en ignorant la case. |

### 1.5. Accèder aux caractères d'une chaîne avec la méthode charAt

| | |
|---|---|
| *char charAt(int index)* | permet d'accéder à un caractère de rang *index* dans la chaîne de caractères. |

### 1.6. Recherche dans une chaîne de caractères

| | |
|---|---|
| *int indexOf(int ch)* | renvoie la position de la première occurrence d'un caractère donné. En commençant la recherche à partir du début de la chaîne jusqu'à sa fin.<br>Si le caractère n'existe pas dans la chaîne, la valeur retournée est -1. |
| *int indexOf(int ch, int fromIndex)* | Comme la méthode précédente mais ici, la recherche commence à partir de *fromIndex* |
| *int indexOf(String str)* | Renvoie la position de la première occurrence de *str* si l'objet cible contient la chaine *str*, sinon renvoie -1. |
| *int indexOf(String str, int romIndex)* | Comme la méthode précédente mais ici la recherche commence à partir de *fromIndex* |

***Auteur :*** ***Tarik BOUDAA, tarikboudaa@yahoo.fr***

## 1.7. Remplacement de caractères

| | |
|---|---|
| *String replace(char ancien, char nouveau)* | Remplace toutes les occurrences d'un caractère (*ancien)* dans une chaîne par un autre caractère (*nouveau).* |
| *String replaceAll(String regex, String remplacement)* | permet de remplacer toutes les occurrences d'une expression régulière *regex* par une expression de *remplacement* |
| *String replaceFirst(String regex, String remplacement)* | Retourne la chaîne dans laquelle on a remplacé la première occurrence d'une expression régulière *regex* par *remplacement*. |

## 1.8. Extraction de sous-chaîne

| | |
|---|---|
| *String subString(int d)* | Sous-chaîne depuis l'indice *d* jusqu'à la fin |
| *String subString(int d, int f)* | Sous-chaîne depuis l'indice *d* jusqu'au caractère d'indice *f* non inclus *(càd on s'arrête à f-1)* |

## 1.9. Conversion entre type primitif et une chaîne

Dans la classe String, la méthode statique *valueOf* est surdéfinie avec un argument des différents types primitifs, cette méthode peut être utilisée pour convertir une valeur numérique en une chaîne.

| | |
|---|---|
| *static String valueOf(boolean b)* | *Retourne la représentation en chaîne du booléen* |
| *static String valueOf(char c)* | *Retourne la représentation en chaîne du caractère* |
| *static String valueOf(char[] data)* | *Retourne la représentation en chaîne du tableau de caractères* |
| *static String valueOf(char[] data, int offset, int count)* | *Retourne la représentation en chaîne du sous-tableau de caractères définit à partir de l'indice offset et de nombre d'élément égale à count* |
| *static String valueOf(double d)* | *Retourne la représentation en chaîne du double* |
| *static String valueOf(float f)* | *Retourne la représentation en chaîne du float* |
| *static String valueOf(int i)* | *Retourne la représentation en chaîne du int* |
| *static String valueOf(long l)* | *Retourne la représentation en chaîne du long* |
| *static String valueOf(Object obj)* | *Retourne la représentation en chaîne de l'objet Si obj= null, alors elle retourne la chaîne "null"; sinon elle retourne obj.toString()* |

Pour réaliser les conversions inverses on peut utiliser les méthodes statiques des classes enveloppes :

- ***Byte.parseByte(String str),***
- ***Short.parseShort(String str),***
- ***Integer.parseInt(String str),***
- ***Long.parseLong(String str),***
- ***Float.parseFloat(String str),***
- ***Double.parseDouble(String str),***

## 1.10. Autres méthodes utiles

| Méthodes | Explication |
|---|---|
| *String toLowerCase()* | Convertit la chaîne en minuscule. |
| *String toUpperCase()* | Convertit la chaîne en majuscule. |
| *String trim()* | Supprime les espaces blancs de début et de fin de chaîne |

| | |
|---|---|
| *String[] split(String motif)* | Décompose la chaîne en un tableau de chaînes en utilisant comme séparateur un élément d'expression régulière. |
| *char[] toCharArray()* | Permet de convertir une chaîne en tableau de caractères |
| *boolean startsWith(String prefix)* | Renvoie true si *prefix* est un préfixe de la chaîne |
| *boolean startsWith(String prefix, int i)* | Renvoie true si *prefix* est un préfixe de la chaîne à partir de i. |
| *boolean endsWith(String suffix)* | Retourne true si *suffix* est un suffixe de la chaîne |

# 2. Classe StringBuffer

## 2.1. Construction

| | |
|---|---|
| *StringBuffer()* | Construit une chaîne vide. |
| *StringBuffer (int l)* | Construit une chaîne vide de capacité initiale de *l* caractères. |
| *StringBuffer (String s)* | Construit une chaîne de caractères à partir de la chaîne s |

## 2.2. Longueur d'une chaîne

| | |
|---|---|
| *int length()* | la longueur de la chaîne de caractères |

## 2.3. Concaténation

| | |
|---|---|
| *StringBuffer append(boolean b)* | Concatène la représentation en chaîne du booléen. |
| *StringBuffer append(char c)* | Concatène la représentation en chaîne du caractère. |
| *StringBuffer append(char[] str)* | Concatène la représentation en chaîne du du tableau de caractères. |
| *StringBuffer append(char[] str, int offset, int len)* | Concatène la représentation en chaîne du tableau de caractères. |
| *StringBuffer append(double d)* | Concatène la représentation en chaîne du double. |
| *StringBuffer append(float f)* | Concatène la représentation en chaîne du float. |
| *StringBuffer append(int i)* | Concatène la représentation en chaîne du int. |
| *StringBuffer append(long l)* | Concatène la représentation en chaîne du long. |
| *StringBuffer append(Object obj)* | Concatène la représentation en chaîne de l'objet |
| *StringBuffer append(String str)* | Concatène la représentation en chaîne de la chaîne. |

## 2.4. Caractère et sous-chaîne.

| | |
|---|---|
| *char charAt(int i)* | Retourne le caractère à l'indice spécifié en paramètre. |
| *String substring(int i)* | Sous-chaîne depuis i jusqu'à la fin |
| *String substring(int i, int l)* | Sous-chaîne depuis i et de longueur l. |

## 2.5. Conversions

| | |
|---|---|
| *String toString()* | Retourne une chaîne équivalente de type String |

## 2.6. Modifications

| | |
|---|---|
| *StringBuffer delete(int start, int end)* | Enlève tous les caractères compris entre start et end. |
| *StringBuffer deleteCharAt(int index)* | Enlève le caractère à l'indice index |
| *StringBuffer reverse()* | Permet de renverser la chaîne |

***Auteur : Tarik BOUDAA, tarikboudaa@yahoo.fr***

## 3. StringBuilder

Toutes les méthodes de la classe *StringBuffer* sont synchronisées, vous pouvez donc utiliser des objets de cette classe sans risque dans des programmes *multi-thread*. Si, par contre, vous vous trouvez dans un programme avec un seul thread, le fait que les méthodes sont synchronisées peut ralentir le programme. C'est pourquoi la librairie standard Java propose la classe *StringBuilder* qui possède exactement les mêmes fonctionnalités que la classe *StringBuffer* mais avec des méthodes non thread-safe.