JAVA03 : LE TYPE CHAINE DE CARACTÈRE : String

VARIABLE PRIMITIVE OU OBJET

String n'est pas un type primitif, c'est un type complexe, surnommé dans la POO, une classe.
Notez que String commence par une majuscule alors que les types primitifs (double, int, ...) commencent par une minuscule. Une variable de type String n'est pas une variable primitive mais un objet de la classe String.
Une variable, qu'elle soit primitive ou objet doit être déclarée et intialisée pour être utilisée.

DÉCLARER UN OBJET STRING

Comme toute variable, elle doit être déclarée. Comme tout objet, son initialisation se fait grâce au mot clef **new**. New permet d'allouer l'espace mémoire nécessaire pour contenir la ou les données contenue(s) dans l'objet:

- création d'une référence pour l'objet (adresse mémoire)
- initialisation des valeurs contenues dans l'objet

```
String maChaine ; // déclaration
maChaine = new String("toto" ) ;//initialisation

String maChaine = new String("toto");//déclaration et initialisation
```

Par défaut, avant d'être initialisé, tout objet est égal à null (référence nulle). Etant donné que String est un type couramment utilisé, déjà existant dans les langages procéduraux, les concepteurs de JAVA ont voulu faciliter son utilisation. On peut donc écrire :

```
String maChaine ="toto" ;
```

MANIPULER UN OBJET STRING

Pour manipuler un objet String, il faut utiliser les méthodes définies pour la classe String. En POO, on dit qu'on applique une méthode à un objet.

```
Syntaxe  : nomObjet . nomMethode(paramètres éventuels....) ;
Exemple   int taille = maChaine.length( ) ; // taille = 4
```

Remarque : attention lorsqu'une méthode retourne un résultat, il faut déclarer une variable compatible avec le résultat retourné, par exemple la méthode **lenght()** de la classe String nous retourne un entier.
Attention n'utilisez pas des opérateurs classiques pour traiter des objets. Comparer le contenu de 2 objets ne sait pas avec l'opérateur classique d'égalité (==) mais avec une méthode définie dans la classe de l'objet, pour la classe String , il s'agit de la méthode **equals(..)**

- == compare la référence des objets
- **equals** compare les valeurs contenues dans les objets

```
String s1 = "toto";
String s2 = new String("toto");
if(s1.equals(s2)) // on teste si les valeurs contenues dans p1 et p2 sont égales
```

EXTRAIT DES MÉTHODES DE LA CLASSE STRING

Java possède une API (Application ProgrammingInterface) extrêmement riche. En effet, sans celle-ci Java n'est rien. C'est avec elle que le programmeur va pouvoir accéder à toutes les ressources du langage. Cette vaste librairie est divisée en packages : des ensembles de classes. Par exemple, la librairie Swing propose des composants graphiques.

Pour utiliser, une librairie vous devez utiliser l'instruction d'import : **import nom du package.** Par exemple, pour utilisez l'Objet **Scanner**, vous devez importer le package : **java.util.Scanner**, ensuite vous pouvez créer un Objet Scanner et utiliser toutes ses méthodes.

Pour la classe String aucun import n'est nécessaire car elle fait partie des classes standards. si vous voulez connaître toutes les méthodes associées à cette classe, vous devez consulter l'API JAVA.

La consultation de l'API est un élément indispensable, si vous ne voulez pas réinventer la roue, car chaque classe contient une multitude de méthode permettant de résoudre les principaux problèmes de programmation.
Attention, vous devez consulter l'API correspondant à votre version du JDK :
http://docs.oracle.com/javase/7/docs/api/java/lang/String.html

java.lang

Class String

`java.lang.Object`
  `java.lang.String`

Constructor Summary

`String``()` Initializes a newly created `String` object so that it represents an empty character sequence.	
`String``(char[] value)` Allocates a new `String` so that it represents the sequence of characters currently contained in the character array argument.	
`String``(char[] value, int offset, int count)` Allocates a new `String` that contains characters from a subarray of the character array argument.	
`String``(String original)` Initializes a newly created `String` object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.	

Dans une première partie l'API présente les différents constructeurs, qui vont nous servir à créer un objet c'est à dire une instance de la classe :
exemple d'utilisation d'un constructeur :

String nom = new String("toto") ;

Ensuite lorsqu'une instance de la classe est créée, on peut accéder aux différentes méthodes :

Method Summary	
char	**charAt**(int index) Returns the char value at the specified index.
String	**concat**(String str) Concatenates the specified string to the end of this string.
boolean	**contains**(CharSequence s) Returns true if and only if this string contains the specified sequence of char values.
boolean	**contentEquals**(CharSequence cs) Compares this string to the specified CharSequence.
boolean	**contentEquals**(StringBuffer sb) Compares this string to the specified StringBuffer.
boolean	**endsWith**(String suffix) Tests if this string ends with the specified suffix.
boolean	**equals**(Object anObject) Compares this string to the specified object.
boolean	**equalsIgnoreCase**(String anotherString) Compares this String to another String, ignoring case considerations.
int	**indexOf**(int ch) Returns the index within this string of the first occurrence of the specified character.
int	**length**() Returns the length of this string.

Certaines méthodes retournent un type de données, par exemple la méthode length() retourne un int, la méthode charAt(int index) retourne le caractère à l'index spécifié.
Exemple :

String nom = new String("toto");
int longueur = nom.length();// longueur vaut 4
char car = nom.charAt(0) ; // car vaut t