La classe `String` / Les Tableaux

Objets immutables

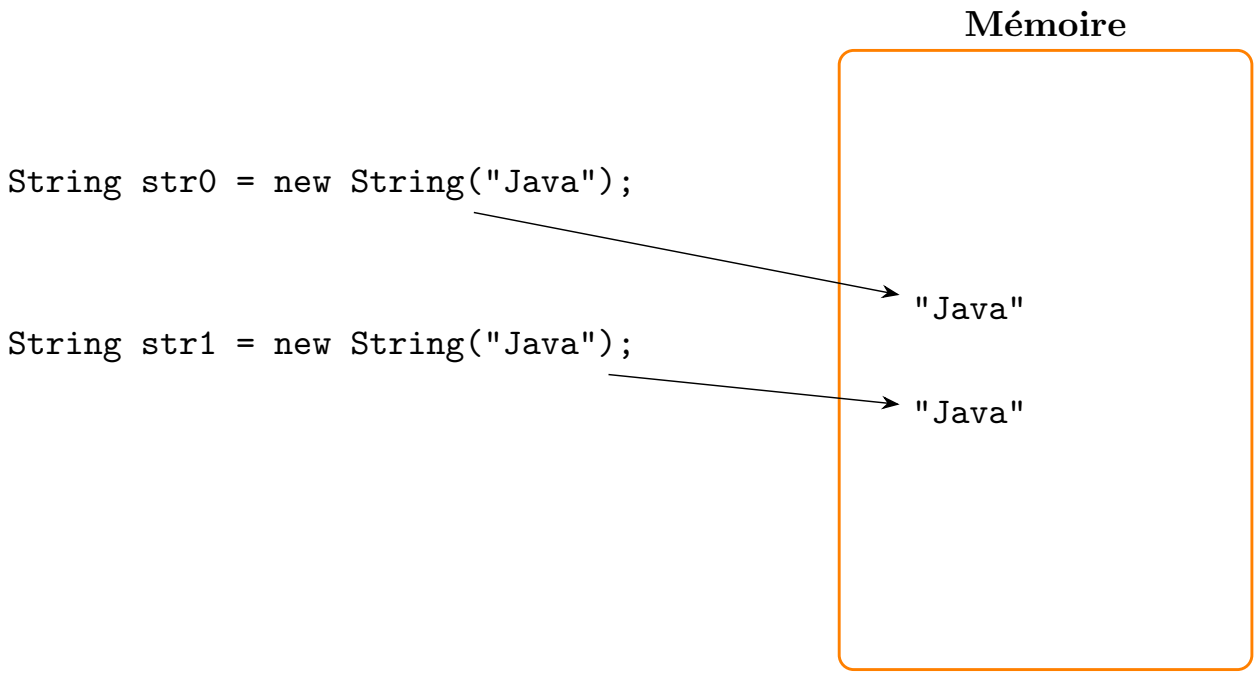
Le langage JAVA™ propose une classe pour stocker et manipuler des chaînes de caractères : la classe `String`.

En JAVA™, les objets `string` sont **immutables** (anglicisme provenant du mot anglais immutable)

Qualifie les objets dont l'état est fixée une fois pour toutes à sa création.

Gestion de la mémoire

Deux objets `String` créés avec l'opérateur **new** occupent des espaces mémoires **différents** même s'ils sont associés à la même chaîne de caractères.



Internat de chaînes

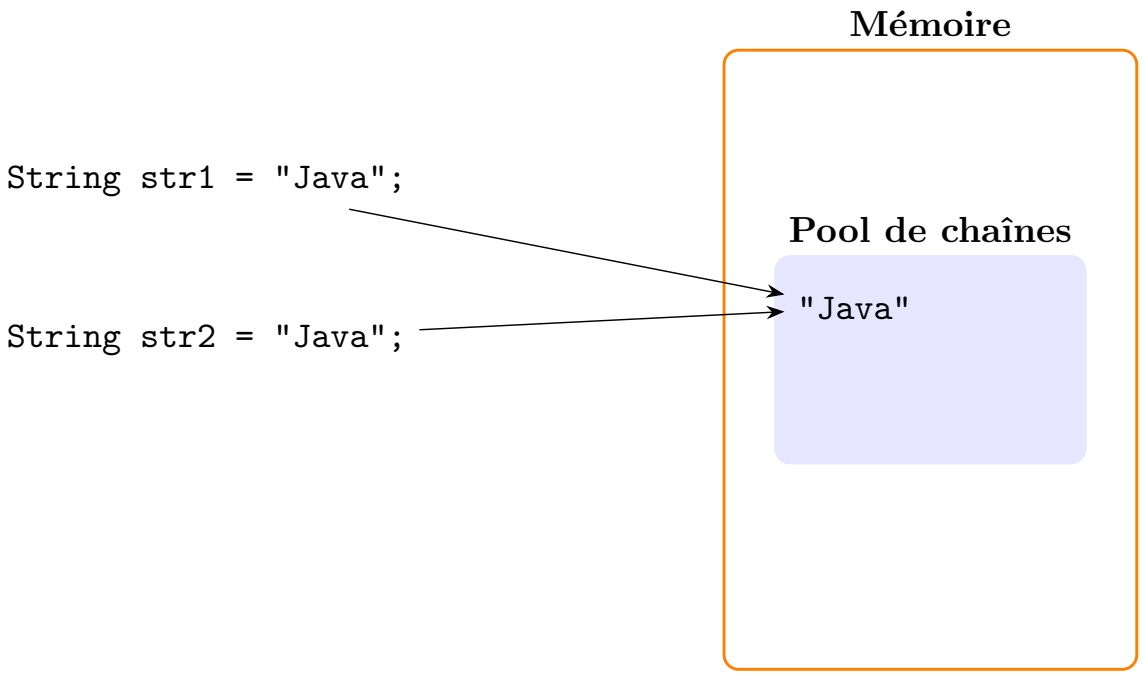
La création d'un objet étant une opération coûteuse en temps et en mémoire. Le langage JAVA™ propose une méthode de gestion des chaînes de caractères qui s'appelle l'**internat de chaînes** (en anglais string interning).

> Cette méthode de gestion des chaînes de caractères est une application d'un patron de conception (en anglais desgin pattern) : flyweight design pattern.

Internat de chaînes

Le principe est le suivant :

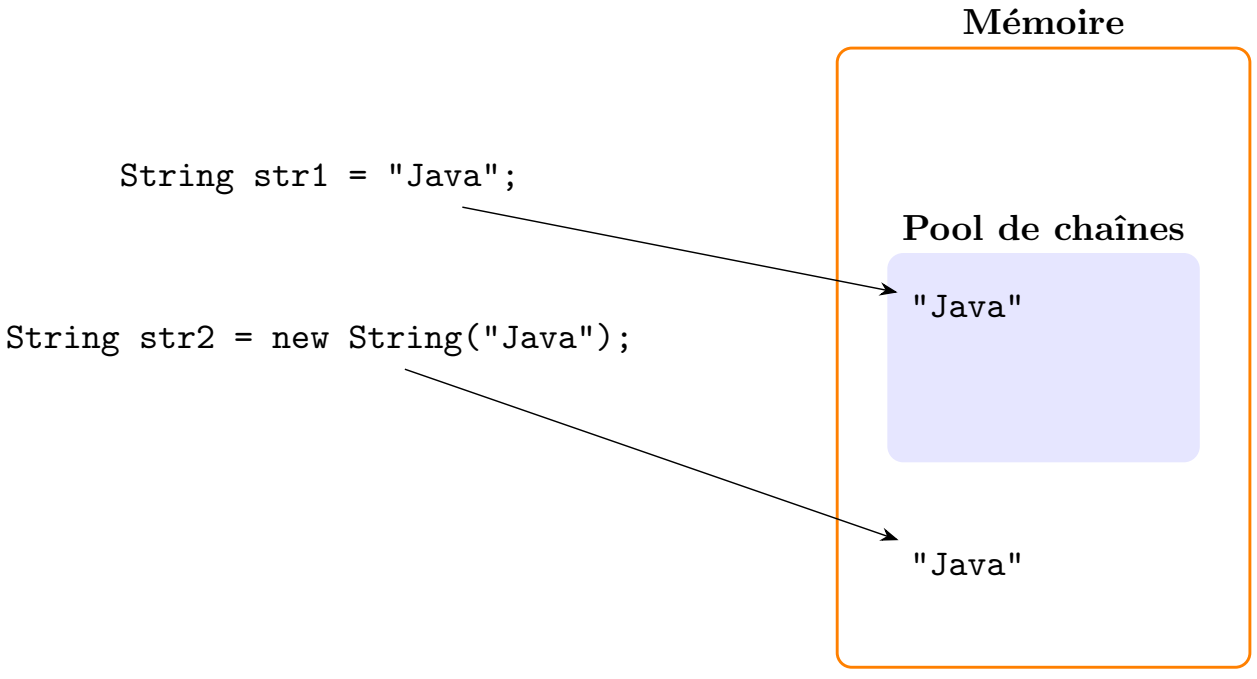
- A la première création d'une chaîne de caractères, un espace mémoire est alloué dans une partie réservée de la mémoire appelée **pool de chaînes** (en anglais String constant pool).
- Ensuite, si la chaîne de caractères existe déjà dans le pool de chaînes, le compilateur JAVA™ renverra simplement une référence à son adresse mémoire, sans allouer d'espace mémoire supplémentaire.



En résumé

Il existe deux types d'objets `String` :

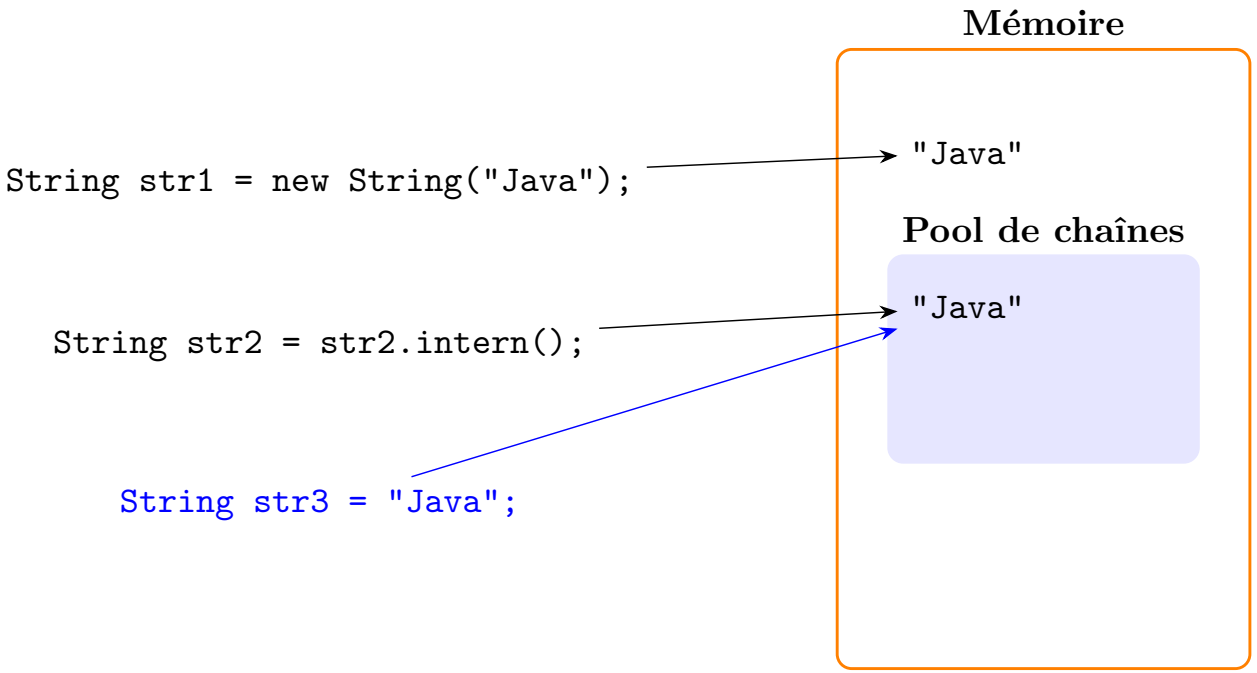
- les objets stockés dans le pool de chaînes
- les objets stockées ailleurs (ceux créés avec l'opérateur **`new`**).



Internat de chaînes

Pour enregistrer dans le pool de chaîne un objet `String` créé avec l'opérateur **new**, on doit utiliser la méthode `intern()` :

```
String str3 = str2.intern();
```



Interface de la classe `String`

Les objets `String` possèdent de nombreuses méthodes :

- `length()` qui retourne la longueur de la chaîne (nombre de caractères).
- `concat()` qui permet de mettre bout à bout deux chaînes de caractères.
- `equals()` et `compareTo()` qui permettent de comparer deux chaînes de caractères.
- `substring()` qui permet d'extraire une sous-chaîne d'une chaîne.
- `indexOf()` qui permet de rechercher une sous-chaîne dans une chaîne.
- etc.

Les tableaux

Un tableau est une structure de données contenant des éléments tous du même type.

Le type des éléments peut être un type de base (**int** , **double** , etc) ou une classe.

Les tableaux sont des objets en JAVA™.

Tableaux à une dimension

Pour créer un tableau à une dimension, on utilise l'opérateur **new** en précisant le nombre d'éléments du tableau, c'est-à-dire sa taille, entre crochets :

- **new int**[5] crée 5 variables de type **int** initialisées à 0
- **new** String[4] crée 4 références de type String initialisées à **null**

Il est possible d'enregistrer l'identité d'un tableau dans une référence :

```
int[] t1 = new int[5];
String[] t2 = new String[4];
```

La référence t1 est une référence de type **int**[] tandis que la référence t2 est une référence de type String[]

Tableaux à une dimension

Les tableaux possèdent tous un attribut `length` indiquant la taille du tableau, c'est-à-dire le nombre d'éléments d'un tableau :

```
int n1 = t1.length; // n1 = 5
int n2 = t2.length; // n2 = 4
```

Chaque case d'un tableau est une variable / référence. Les cases d'un tableau sont numérotées à partir de 0.

```
t1[0] = 3;
t1[1] = 5;
int i = t1[1]; // i = 5
t2[0] = "Licence";
t2[3] = "MIASHS";
String str1 = t2[3]; // str1 = "MIASHS"
```

Initialiser un tableau

Il est possible d'intialiser un tableau à sa création en indiquant entre accolades la liste des valeurs des cases du tableau séparées par des virgules.

```
int[] tableau1;
tableau1 = new int[] {-1, 5, 0, 4, -2};

String[] tableau2;
tableau2 = new String[] {"programmation","Orientée"
    ,"objet"};
```

Dans ce cas, on n'indique pas la taille du tableau entre crochets. La taille du tableau est calculée automatiquement lors de l'initialisation du tableau.

Parcourir les éléments d'un tableau

Pour parcourir les éléments d'un tableau, on peut écrire

```
for(int i = 0; i < tableau1.length; i++)
    System.out.println(tableau1[i]);
```

Le langage JAVA™ propose une nouvelle syntaxe de la boucle **for**

```
for(int x : tableau1) System.out.println(x);
```

Le tableau est parcouru dans le même sens que dans la première boucle.

La classe `Arrays`

Le paquetage `java.util` propose de nombreuses classes utiles quyi enrichissent les classes de base du langage.
Il contient notamment une `Arrays` non instanciable qui propose plusieurs méthodes pour faciliter la gestion des tableaux.

Un **paquetage** est une bibliothèque de classes.

Une classe **non instanciable** est une classe dont on ne peut pas créer d'instances.

La classe `Arrays`

La classe propose la méthode `fill` qui permet de remplir un tableau avec une valeur données;

```
String[] tableau;
tableau = new String[3];
Arrays.fill(tableau,"java");
```

Chaque case du tableau contient la chaîne `"java"`

> N'oubliez pas d'instancier le tableau avant de le remplir !

La classe `Arrays`

La classe propose la méthode `copyOf` qui permet de créer un **nouveau** tableau en recopiant les n premières cases d'un tableau (si n est plus grand que la taille du tableau, les derières cases sont initialisées à la valeur par défaut) :

```
    /*
     crée un tableau obtenu en recopiant les 3 premi
    ères cases de tableau
     */
String[] copie = Arrays.copyOf(tableau,3);
```