Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-05-1-stringintro-pt2

Concepts (extraits des sous-titres générés automatiquement) :

Chaîne de caractères. Variable de type string. Chaîne vide. Littéraux de type string. Variable chaine. Valeur de chaine2. Affichage de la chaîne de caractères. Variable chaîne. Affichage d'une variable de type. Stade de votre apprentissage. Chaînes de caractères. Seul espace. Simples quotes. Types évolués. Types de base.

vers la recherche de séquences vidéo
(dans Initiation à la programmation (en Java).)

vers la vidéo

EPFL

String : introduction
(Partie 2)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

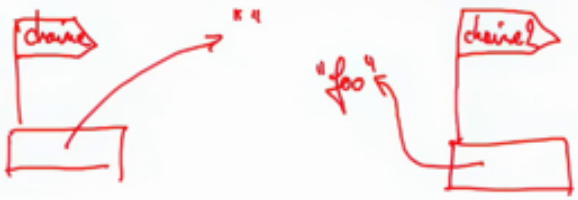
résumé

0m 0s

String : Sémantique des opérateurs = et ==

Comme pour les tableaux, une variable de type `String` contient une référence vers une chaîne de caractères. La sémantique des opérateurs = et == est donc la même que pour les tableaux :

```
String chaine = "";      // chaine pointe vers ""
String chaine2 = "foo"; // chaine2 pointe vers "foo"
chaine = chaine2 ;       // chaine et chaine2 pointent vers "foo"
if(chaine == chaine2){   // [redacted] true
```



Comme pour les tableaux, une variable de type String contient une référence vers une chaîne de caractères. La sémantique des opérateurs = et == est donc la même que pour les tableaux. Par exemple, je déclare ici une variable "chaîne" qui contient une référence vers la chaîne vide; ici je déclare une variable "chaine2" qui contient une référence vers la chaîne "foo". Quand je fais cette opération, cette instruction, j'affecte la valeur

notes

résumé

0m 1s

String : Sémantique des opérateurs = et ==

Les littéraux de type `String` occupent une zone mémoire unique

☞ « Pool » des littéraux

```
String chaine1 = "foo";  // chaine1 pointe vers le litteral "foo"
String chaine2 = "foo" ; // chaine2 pointe vers le litteral "foo"
if (chaine1 == chaine2 ) // true : chaine1 et chaine2 contiennent la même
                         //adresse
```

de chaine2 à la variable chaine, chaine2 contient une référence vers "foo", donc je vais copier cette référence dans ma variable chaîne, la référence qui faisait référence à la chaîne vide est perdue, puisqu'elle est écrasée par cette nouvelle référence, et "chaîne" pointe donc également sur "foo". Si maintenant je fais "if (chaine == chaine2)", eh bien cette condition est vraie, puisque "chaine" et "chaine2" contiennent donc la même référence, toutes les deux vers "foo". Les littéraux de type String occupent une zone mémoire unique qui s'appelle le "Pool" des littéraux. Par exemple, quand je fais cette déclaration et initialisation,

notes

résumé

0m 49s

String : Affichage

Qu'affiche le code suivant ?

```
String chaine = "Welcome";
System.out.print(chaine);
```

Puisque la variable `chaine` contient une référence à la zone mémoire contenant la chaîne `"Welcome"`, il est raisonnable de penser que ce code affiche une adresse (comme pour les tableaux de manière générale) : **ce n'est pas le cas !**

- Le code précédent affiche `Welcome`
- Pour les `String` l'affichage est défini de sorte à prendre en compte la référence pointée plutôt que la référence elle-même. C'est une exception.

je déclare chaine1 qui va contenir une référence vers "foo". En fait, cette chaîne "foo" va être mise dans le pool des littéraux et quand je fais cette seconde déclaration et initialisation, chaine2 va être également initialisé à la chaîne "foo" et c'est cette même référence, c'est cette même chaîne "foo" qui va être réutilisée. Et donc chaine1 et chaine2 vont contenir la même référence, ça veut dire que quand j'utilise cette condition chaine1 == chaine2, cette condition est vraie. Passons maintenant à l'affichage d'une variable de type String. Je déclare et j'initialise ici une variable appelée "chaine" avec la chaîne de caractères "Welcome"; "chaine" contient donc une référence vers "Welcome". Quand maintenant j'appelle la fonction "print" en lui passant "chaine", je serais en droit de m'attendre à ce que "print" affiche ce que contient "chaine", c'est-à-dire la référence à la mémoire où est stockée effectivement la chaîne "Welcome". Eh bien, ce n'est pas le cas et "print" affiche effectivement la chaîne "Welcome". A ce stade de votre apprentissage, on peut considérer ceci comme une exception pour les Strings par rapport aux types évolués, mais c'est une exception bien pratique, puisque quand on fait

notes

résumé

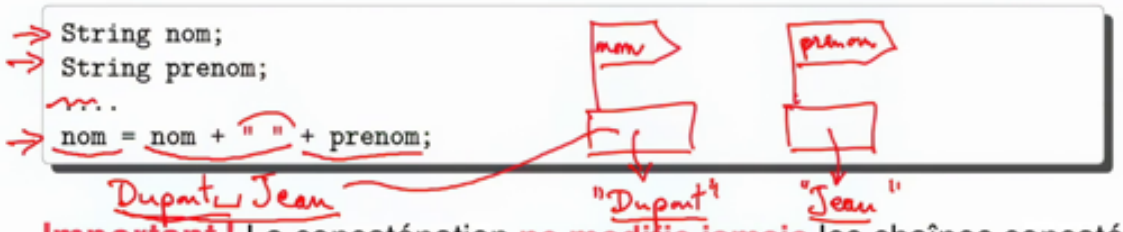
1m 37s

Concaténation

`chaine1 + chaine2` produit une **nouvelle chaîne** associée à la valeur littérale constituée de la **concaténation** des valeurs littérales de `chaine1` et de `chaine2`.

Exemple : constitution du nom complet à partir du nom de famille et du prénom :

```
String nom;
String prenom;
...
nom = nom + " " + prenom;
```



Important ! La concaténation **ne modifie jamais** les chaînes concaténées. Elle effectue une **copie** de ces chaînes dans une autre zone en mémoire.

un print d'une chaîne, on a plutôt envie d'avoir l'affichage de la chaîne de caractères plutôt qu'une référence sur cette chaîne. On peut concaténer des chaînes de caractères, c'est-à-dire mettre bout à bout ces chaînes en utilisant le caractère "+"; dans cet exemple la chaîne1 et chaine2 sont des variables de type "chaînes de caractères", et chaine1 + chaine2 est une nouvelle chaîne de caractères constituée au début de chaine1 et à la fin des caractères de chaine2. Par exemple, je déclare ici une variable qui s'appelle "nom" de type "chaîne de caractères", ici une variable "prenom". Supposons qu'ici j'ai mis une valeur pour "nom", par exemple "Dupont", et pour "prenom", une autre valeur, "Jean". Dans cette affectation ici, je commence par faire cette première concaténation, c'est-à-dire que je mets ensemble "Dupont", et cette chaîne de caractères, qui est constituée d'un seul espace, c'est-à-dire que j'obtiens pour l'instant une chaîne de caractères qui est "Dupont" suivi du caractère espace, et à cette chaîne, je vais concaténer encore une fois, en lui ajoutant la valeur de "prenom", c'est-à-dire "Jean". Et tout ceci me constitue une nouvelle chaîne de caractères, et comme je vais affecter le résultat à ma variable "nom", "nom" va contenir

notes

résumé

3m 13s

Concaténation

Les combinaisons suivantes sont possibles pour la concaténation de deux chaînes :

`String + String`

`String + typeDeBase` `typeDeBase + String`

Exemple revisité (avec `char`) :

```
String nom;
String prenom;
....
nom = nom + ' ' + prenom;
```

une référence vers cette chaîne de caractères, et cette référence sera perdue. Alors, attention, la concaténation ne modifie jamais les chaînes concaténées. Elle effectue une copie de ces chaînes dans une autre zone en mémoire. C'est ce qu'on voit sur le schéma : "Dupont" n'a pas été modifié.

notes

résumé

4m 49s

Concaténation

Les combinaisons suivantes sont possibles pour la concaténation de deux chaînes :

`String + String`

`String + typeDeBase` (int, double, char, ...)

`typeDeBase + String`

Exemple revisité (avec `char`) :

```
String nom;
String prenom;
....
nom = nom + ' ' + prenom;
```

Les combinaisons suivantes sont possibles pour la concaténation de deux chaînes. Je peux concaténer tout simplement deux variables de type String, mais je peux aussi commencer par une variable de type String et la faire suivre par une variable d'un des types de base, c'est-à-dire "int", "double", "char", etc, ou faire l'inverse, c'est-à-dire commencer par une variable d'un des types de base "int", "double", "char", etc,

notes

résumé

5m 10s

Concaténation

Les combinaisons suivantes sont possibles pour la concaténation de deux chaînes :

`String + String`

`String + typeDeBase` `typeDeBase + String`

Exemple revisité (avec `char`) :

```
String nom;
String prenom;
....
nom = nom + ' ' + prenom;
```

et la faire suivre par une variable de type String.

notes

résumé

5m 37s

Si je reprends l'exemple précédent, j'avais utilisé une chaîne de caractères pour insérer un espace entre le nom et le prénom, on pouvait reconnaître que c'était une chaîne de caractères parce que j'avais utilisé des guillemets qui sont appelés aussi des "doubles quotes", mais je peux utiliser aussi un simple caractère, c'est ce que nous voyons sur cette combinaison. Alors je vais utiliser, dans cet exemple, cette fois-ci un caractère qui est reconnaissable au fait que j'utilise des apostrophes ou des simples quotes, et cette affectation va faire exactement la même chose que l'affectation précédente. que l'affectation précédente.

notes

résumé

5m 41s