Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-05-2-stringcomp-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Comparaison de chaînes de caractères. Variables de type string. Chaîne de caractères. Utilisation du pool des littéraux. Résultat de cette condition. Non-égalité. Initialisation d'une variable. Nombre de subtilités. Variable de type string. Zone mémoire. Pool des littéraux. Chaîne vide. Concaténation de la variable. Valeur de cette condition. Référence.

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

vers la vidéo

EPFL

String : comparaisons
(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

(Non-)Egalité de `Strings`

EPFL

Les opérateurs suivants :

`==`	égalité
`!=`	non-égalité

testent si deux variables `String` font référence (ou non) à la même zone mémoire (occupée par une chaîne de caractères).

Ceci est le cas lorsque les variables de types `String` ont été initialisées au moyen de littéraux

Exemple : utilisation de l'opérateur `!=`

```
while (reponse != "oui") ....;
```

Dans cette vidéo, nous allons nous concentrer sur la comparaison de chaînes de caractères. Attention, les opérateurs « == » pour l'égalité et « ! = » pour la non-égalité, quand ils sont appliqués sur deux variables de type string, ils testent si ces variables font référence (ou non) à la même zone mémoire,

notes

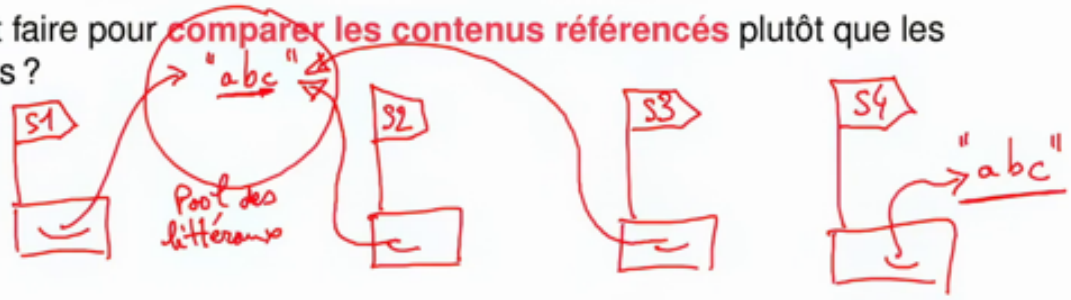
résumé

0m 1s

(Non-)Egalité de Strings (2)

```
String s1 = "abc";     // s1 pointe vers le littéral "abc"
String s2 = "abc";     // idem (donc même zone mémoire que s1)
String s3 = s2;        // s3 stocke la même adresse que s2
String s4 = s1 + "";   // s4 contient l'adresse d'une nouvelle chaîne
                       // (construite par concaténation)
System.out.println((s1==s2) && (s2==s3)); // affiche true
System.out.println(s4);                   // affiche abc
System.out.println((s1==s4));             // affiche false
```

Comment faire pour comparer les contenus référencés plutôt que les références ?



qui contiendrait une chaîne de caractères. Etant donné, en plus, l'utilisation du pool des littéraux, ceci entraîne un certain nombre de subtilités, que nous allons voir sur cet exemple. J'ai ici la déclaration et l'initialisation d'une variable « s1 » qui va faire référence à une chaîne "abc". Ici, « s2 » est initialisé à la chaîne "abc" également. Cette chaîne "abc" appartient au pool des littéraux. C'est pour cela que « s1 » et « s2 » ont la même référence sur la même chaîne. Ici, je déclare une variable « s3 » dans lequel je copie le contenu de « s2 ». « s2 » contient une référence vers cette chaîne "abc" et donc, « s3 » va maintenant aussi contenir une référence sur cette même chaîne "abc". Je fais maintenant cette dernière affectation. A « s4 », je vais lui affecter la concaténation de la variable « s1 » qui est, "abc". Et la chaîne vide, que je vais représenter ainsi. Et j'obtiens donc, tout simplement, la chaîne "abc". Et « s4 » va recevoir une référence vers cette chaîne "abc". Mais attention, cet "abc"-ci n'est pas le même que celui-ci. Plus exactement, elles ne sont pas au même endroit dans la mémoire et cette référence et ces références sont différentes. Alors, pourquoi exactement? C'est parce que « s1 », « s2 » et « s3 » ont été initialisés directement à partir de littéraux, alors que ce n'est pas le cas pour « s4 » ; et ce qu'on peut vérifier en exécutant ces instructions-ci. Ce « println » affiche le résultat de cette condition.

notes

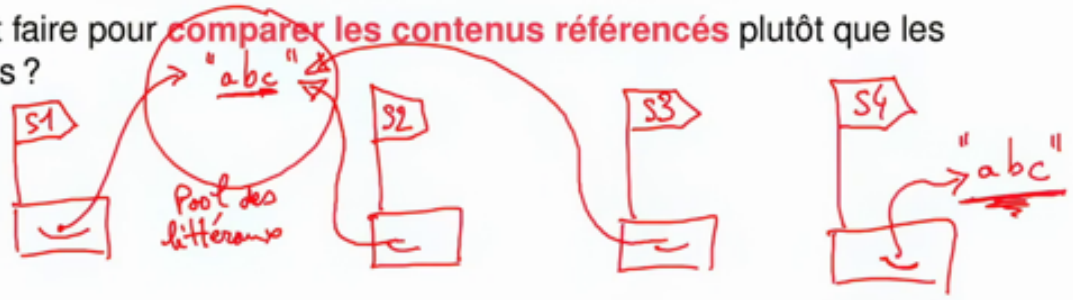
résumé

0m 25s

(Non-)Egalité de Strings (2)

```
String s1 = "abc";     // s1 pointe vers le littéral "abc"
String s2 = "abc";     // idem (donc même zone mémoire que s1)
String s3 = s2;        // s3 stocke la même adresse que s2
String s4 = s1 + "";   // s4 contient l'adresse d'une nouvelle chaîne
                       // (construite par concaténation)
System.out.println((s1==s2) && (s2==s3)); // affiche true
System.out.println(s4);                   // affiche abc
System.out.println((s1==s4));             // affiche false
```

Comment faire pour comparer les contenus référencés plutôt que les références ?



Cette condition est vraie quand « s1 » est égal à « s2 », c'est-à-dire quand la référence contenue par « s1 » et la référence contenue par « s2 » sont les mêmes. Et sur le schéma on constate que c'est bien le cas. Donc cette condition est vraie. « s2 » est égal, effectivement, à « s3 » puisque « s2 » et « s3 » contiennent toutes les deux une référence vers cette même chaîne "abc". Cette condition est vraie. Et comme ici c'est un « ET », toute cette condition est vraie. On va donc afficher « true », puisqu'on peut le vérifier en pratique. Quand j'exécute cette instruction-ci, qui affiche la chaîne « s4 » ; alors je vous rappelle, quand on exécute la fonction « print » ou « println » sur une variable de type string, on n'affiche pas la référence, mais bien la chaîne qui est référencée. Ce qui est pratique. Et on va obtenir, ici, l'affichage de "abc" qui est, cette chaîne-ci. Maintenant, si j'affiche la valeur de cette condition,

notes

résumé

2m 37s

qui est « s1 == s4 » « s1 » contient cette référence, « s4 » contient cette référence, qui sont des références différentes, et donc la condition est fausse. Et ce « println » va m'afficher « false », même si les deux chaînes qui correspondent à « s1 » et « s4 » sont les mêmes, sont toutes les deux "abc". Donc c'est bien les références qui sont comparées ici. qui sont comparées ici.

notes

résumé

3m 49s