Support de cours

Cours:

## Initiation à la programmation (en Java)

Vidéo:

## Init-JAVA-05-3-stringtrait-pt3

Concepts (extraits des sous-titres générés automatiquement) :

**Variable test. Chaîne vide. Caractère d'indice. Instant test. Variable essai. Premier tour de boucle. Indice de la valeur de cette expression. Fonction nextchar. Instance du type scanner. Test test. Classe scanner. Pool des littéraux. Instruction d'affichage. Boucle for. Fonction nextline.**

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

vers la vidéo

EPFL

# String : traitements
## (Partie 3)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

# Les chars d'un String (2)

EPFL

Exercice : qu'affichera le programme suivant :

```
String essai = "essai";
String test = "";
for (int i = 1; i <= 3; ++i) {
 test = test + essai.charAt(6-2*i);
 test = essai.charAt(i) + test;
}
System.out.println(test);
```

Alors, la variable essai est initialisée à la chaîne "essai"

notes

résumé

0m 1s

# Les chars d'un String (2)


Exercice : qu'affichera le programme suivant :

```
String essai = "essai";
String test = "";
for (int i = 1; i <= 3; ++i) {
test = test + essai.charAt(6-2*i);
 test = essai.charAt(i) + test;
}
System.out.println(test);
```

i : 1 2 3

La variable test est initialisée à la chaîne vide. On entre dans cette boucle for, où i va varier de 1 à 3. Au premier tour de boucle, on va exécuter cette affectation.

notes

résumé

0m 6s

# Les chars d'un String (2)


Exercice : qu'affichera le programme suivant :

```
String essai = "essai";
String test = "";
for (int i = 1; i <= 3; ++i) {
test = test + essai.charAt(6-2*i);
 test = essai.charAt(i) + test;
}
System.out.println(test);
```

Pour l'instant test est la chaîne vide et on va lui concaténer le caractère qui se trouve dans essai à l'indice de la valeur de cette expression. i vaut 1, donc 6 moins 2 fois i vaut 4. On va donc considérer le caractère d'indice 4 de essai. C'est à dire, 0,1, 2, 3, 4, donc tout ceci, vaut 'i'. On va concaténer ce caractère 'i' avec la chaîne vide et donc obtenir i

notes

résumé

0m 35s

# Les chars d'un String (2)


Exercice : qu'affichera le programme suivant :

```
String essai = "essai";
String test = "";
for (int i = 1; i <= 3; ++i) {
test = test + essai.charAt(6-2*i);
test = essai.charAt(i) + test;
}
System.out.println(test);
```

i : 1 2 3

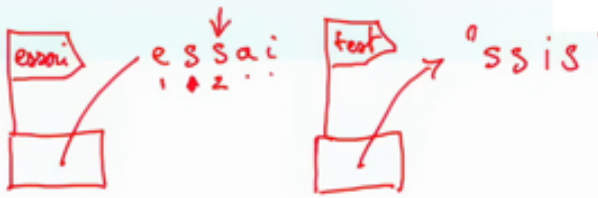
et mettre le résultat dans test test va donc pointer maintenant sur la chaîne "i"

notes

résumé

1m 9s

# Les chars d'un String (2)




Exercice : qu'affichera le programme suivant :

```
String essai = "essai";
String test = "";
for (int i = 1; i <= 3; ++i) {
 test = test + essai.charAt(6-2*i);
 test = essai.charAt(i) + test;
}
System.out.println(test);
```

i : 1 2 3

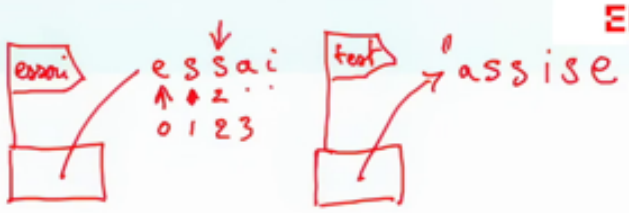
Passons ensuite à cette instruction, qui va concaténer le caractère d'indice i de la chaîne essai, i vaut 1, le caractère est le caractère 's' et le concaténer avec la chaîne contenue par test c'est à dire tout simplement "i". Le résultat de la concaténation est la chaîne "si", on va mettre "si" ici. On revient au début de la boucle for. i vaut maintenant 2. On exécute cette affectation. test vaut maintenant "si". Ce caractère est le caractère d'indice 2 dans essai. Le caractère d'indice 2, c'est (zéro, un, deux) c'est un 's', qu'on va concaténer avec la chaîne "si" donc on va obtenir "sis" qu'on va affecter à la variable test Donc obtenir ici. Passons à la seconde affectation, essai.charAt(i), comme i vaut 2, c'est ce 's' ici qu'on va concaténer avec la chaîne qui correspond à test c'est à dire "sis" et on va obtenir "ssis" qu'on va mettre dans test.

notes

résumé

1m 19s

# Les chars d'un String (2)




Exercice : qu'affichera le programme suivant :

```
String essai = "essai";
String test = "";
for (int i = 1; i <= 3; ++i) {
test = test + essai.charAt(6-2*i);
test = essai.charAt(i) + test;
}
System.out.println(test);
```

On va donc revenir maintenant une dernière fois ici, avec i qui vaut 3. test vaut "ssis" et on va lui concaténer ce caractère-ci donc le caractère d'indice 0 dans la chaîne essai c'est à dire ce 'e', ici. On va mettre le résultat dans la chaîne test c'est à dire qu'on va mettre "ssise" dans test On passe à cette affectation. essai.charAt(i): i vaut 3, c'est le caractère d'indice 3 de essai c'est à dire, 0, 1, 2 , 3 c'est à dire 'a' qu'on va concaténer avec ce que contient test c'est à dire "ssise", et obtenir la chaîne "assise". test va donc pointer maintenant chaîne "assise". On sort de la boucle puisque i valait 3.

notes

résumé

3m 1s

## Pas de `nextChar()` dans la classe `Scanner` !?

Pour récupérer un caractère (`char`) avec la classe `Scanner`, il faut faire :

```
// Lire la ligne qui contient un caractère
Scanner keyb = new Scanner(System.in);
String s = keyb.nextLine();

// Prendre comme caractère le premier élément de la String
char c = s.charAt(0);
```

On arrive à cette instruction d'affichage qui va afficher tout simplement assise.

notes

résumé

4m 13s

# Littéraux introduits par l'utilisateur

Un littéral introduit par l'utilisateur suite à une instruction de lecture n'est pas dans le pool des littéraux

Pour qu'il y soit, il faut l'y mettre explicitement au moyen de `intern`

Exemple

```
Scanner s = new Scanner(System.in);
String response;
do {
   response = s.nextLine();
   //on met le littéral lu dans le pool
   response = response.intern();
   System.out.println("Read: " + response);
   // sans le intern, la boucle ne s'arrête pas!
} while (response != "oui");
```



Un mot sur les entrées / sorties avec les caractères. Il se trouve qu'il n'y a pas de fonction nextChar() dans la classe Scanner et pour récupérer un caractère en utilisant cette classe, il faut faire d'abord comme d'habitude la déclaration d'un Scanner. Ensuite, lire une ligne en utilisant la fonction nextLine() appliquée à l'instance du type Scanner et mettre le résultat dans une variable de type chaîne que j'ai appelée s et prendre le premier caractère de cette chaine s en utilisant la fonction charAt() en lui passant comme argument l'indice 0. Donc dans c ici , on va obtenir le premier caractère qu'aura tapé l'utilisateur à ce moment-ci. Un littéral, introduit par l'utilisateur suite à une instruction de lecture avec un Scanner par exemple, n'est pas dans le pool des littéraux. Pour qu'il y soit, il faut l'y mettre explicitement au moyen de la fonction intern. Considérons cet exemple où je commence par déclarer un Scanner s qui va me permettre de lire ce que l'utilisateur tape au clavier. Je déclare ici une variable response de type chaine de caractère et j'ai une boucle dowhile qui devrait s'arrêter quand l'utilisateur rentre oui comme réponse. Alors ce oui est une chaîne de caractères qui est dans le pool de littéraux. Continuons maintenant le code. Supposons que je lise au clavier ce qu'a tapé l'utilisateur et que je mette le résultat dans ma variable response. Je vais obtenir quelque chose comme ça. Supposons qu'il ait déjà tapé oui. parce qu'il a envie de sortir de la boucle. Mais, si vous avez suivi ce que je viens de dire, ce "oui" n'est pas dans le pool des littéraux, c'est à dire que ce "oui" est différent de ce "oui" ci. Cet affichage va bien m'afficher oui pour response. Par contre ce test

notes

résumé

4m 18s

# Littéraux introduits par l'utilisateur

EPFL

Un littéral introduit par l'utilisateur suite à une instruction de lecture n'est pas dans le pool des littéraux

Pour qu'il y soit, il faut l'y mettre explicitement au moyen de `intern`

Exemple

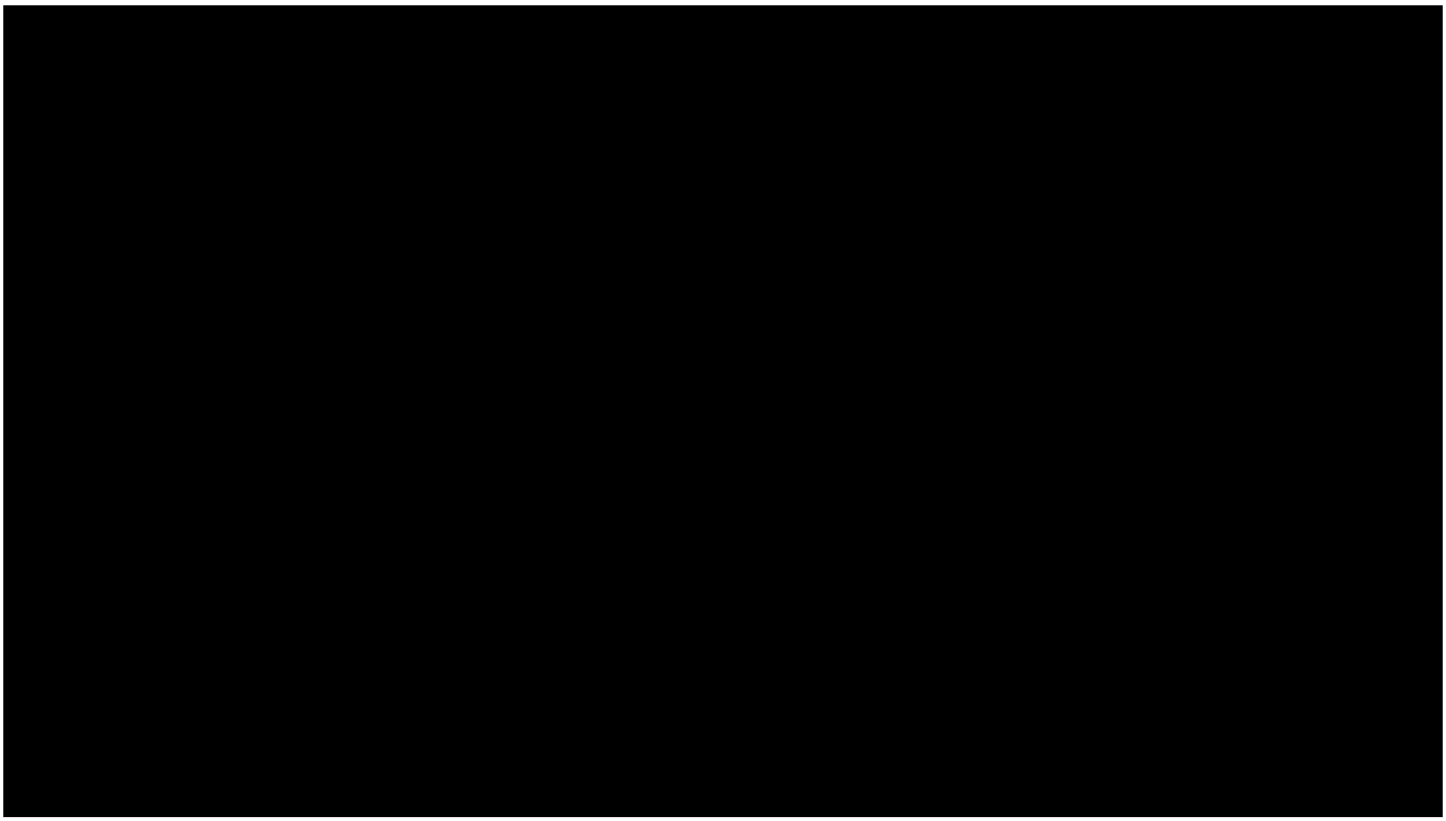
```
Scanner s = new Scanner(System.in);
String response;
do {
   response = s.nextLine();
   //on met le littéral lu dans le pool
   response = response.intern();
   System.out.println("Read: " + response);
   // sans le intern, la boucle ne s'arrête pas!
} while (response != "oui");
```



ne va pas faire ce que je veux, puisque je teste ici les références. response contient une référence vers ce "oui" et ce "oui" ici n'est pas stocké à la même référence puisqu'il est dans le pool des littéraux. Donc dans ce test, cette condition va être fausse, même si response contient bien "oui". Je vais rajouter l'appel à la fonction intern, que j'avais caché ici. Que se passe-t-il dans ce cas là ?

notes

résumé



Quand on exécute cette nouvelle instruction, le "oui" qui est contenu par response va être mis dans le pool des littéraux. Il se trouve qu'il y a déjà un "oui" dans le pool des littéraux Ce qu'il va se passer, c'est que maintenant response va contenir une référence vers ce "oui" là. Et ce lien est perdu. Cette instruction d'affichage va afficher comme avant un oui. C'est ce oui ci qui va être affiché. Et maintenant, quand on fait ce test là. On va tester toujours les références. C'est à dire la référence contenu dans response avec la référence de ce "oui" qui est dans le pool des littéraux. Mais comme cette référence va vers le "oui" du pool des littéraux, c'est la même référence et ce test sera donc vrai cette fois-ci. Tout ceci grâce à la fonction intern. à la fonction intern.

notes

résumé

7m 13s