Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-06-3-Arguments-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Simples valeurs. Cas du passage. Séquence précédente. Programmation de façon générale. Moment de l'appel. Dernière séquence. Copie locale de l'argument. Zone locale. Programme principal. Altération de cette zone locale. Différentes étapes. Sortir de cette méthode. Variable v1 du même type. Méthode. Situation concrète suivante.

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

vers la vidéo

EPFL

Fonctions : passage des arguments
(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Dans une séquence précédente, nous avons examiné les différentes étapes qui se déroulent lorsqu'on appelle une méthode. Dans les exemple vus jusqu'alors, les arguments passés à la méthode étaient

notes

résumé

0m 1s

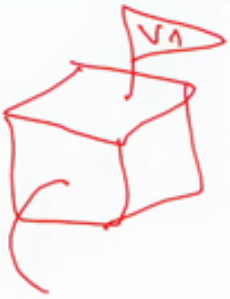
Le passage des arguments (1)

EPFL

Considérons la situation suivante (pseudo-code) :

```
static void methode(Type v) {
  // traitement modifiant v
}

// ailleurs, dans le programme principal,
// par exemple:
Type v1 = .. ; // initialisation de v1
methode(v1);
// v1 EST-ELLE MODIFIEE ICI OU NON???
```



En programmation de façon générale, on dira que :

- L'argument `v` est **passé par valeur** si `methode` ne peut pas modifier `v1` : `v` est une **copie locale** de `v1`.
- L'argument `v` est **passé par référence** si `methode` peut modifier `v1`

non

oui

soit de simples valeurs, soit des expressions à évaluer. Nous allons maintenant affiner un petit peu le discours et voir ce qui se passe lorsque l'argument passé à la méthode est une variable, et en particulier dans la situation où la méthode modifie le paramètre qui lui est passé. Donc nous allons nous placer dans une situation concrète suivante. Supposons par exemple que nous soyons dans un programme principal qui déclare une variable v1 du même type que celui attendu par la méthode. Nous invoquons ensuite la méthode en lui passant un argument, la variable v1. Nous avons vu lors de la dernière séquence que v1 était copiée dans v et cette méthode a pour vocation de modifier v. Donc la question que nous posons concrètement maintenant c'est au sortir de cette méthode, est-ce que la variable v1 déclarée dans ce programme principal sort modifiée ou pas de ce traitement ? En programmation de façon générale, on dira que v est passé par valeur si la réponse à cette question est non, et on dira a contrario que v est passé par référence si la réponse à la question est oui. Dans le cas du passage par valeur, ce qui se produit est que la méthode va travailler sur une copie locale de l'argument qui lui est passé, v1. Donc, au moment de l'appel, lorsqu'on réalise cet appel, la variable v1

notes

résumé

0m 13s

est tout simplement copiée dans une zone locale v qui est locale à la fonction, et toute altération de cette zone locale n'a aucune incidence sur la variable d'origine. la variable d'origine.

notes

résumé

1m 49s