Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-04-4-tableaux-affectation-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Types de base. Cas d'un type de base. Moyen d'une boucle. Cas de types évolués. Types évolués. Cas des types de base. Taille du tableau. Troisième entrée du tableau. Variables de type tableau. Condition d'arrêt. Tableaux. Affectation telle. Cas de figure. Genre d'affectation. Type de base.

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

vers la vidéo

EPFL

Tableaux : affectation et comparaison
(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

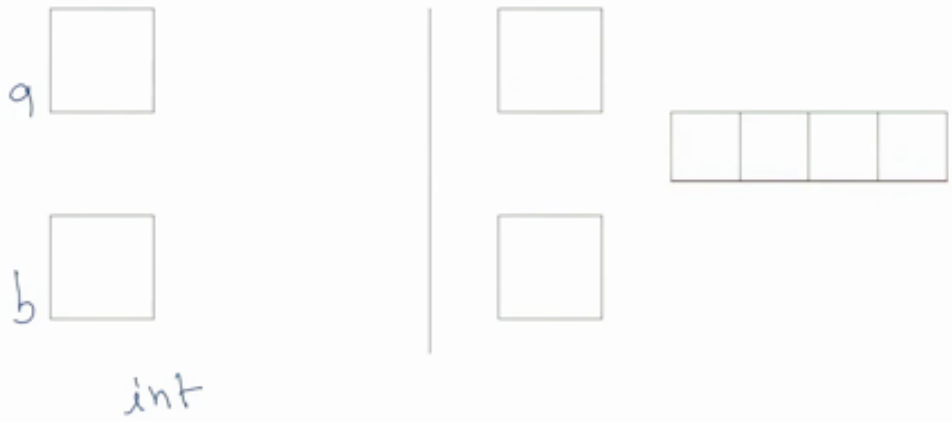
...

notes

résumé

0m 0s

Types de base / Types évolué (rappel)



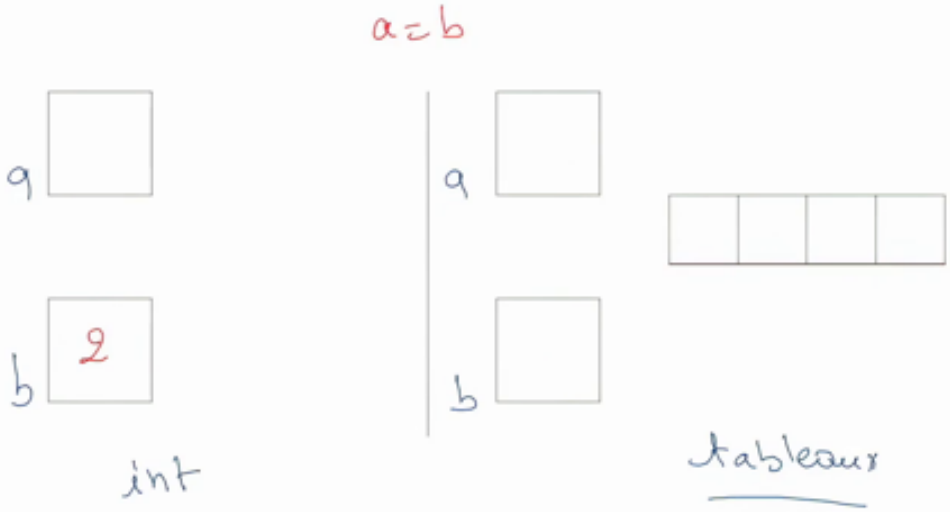
Nous allons maintenant nous intéresser à ce que signifie l'affectation lorsqu'on l'emploie avec des tableaux. Pour ceci, il va être nécessaire de revenir un peu sur notre discussion en comparant les types de base et les types évolués en Java. Supposons que nous ayons affaire à deux variables « a » et « b », tantôt dans le cas d'un type de base, par exemple ici « a » et « b » sont des entiers,

notes

résumé

0m 1s

Types de base / Types évolué (rappel)



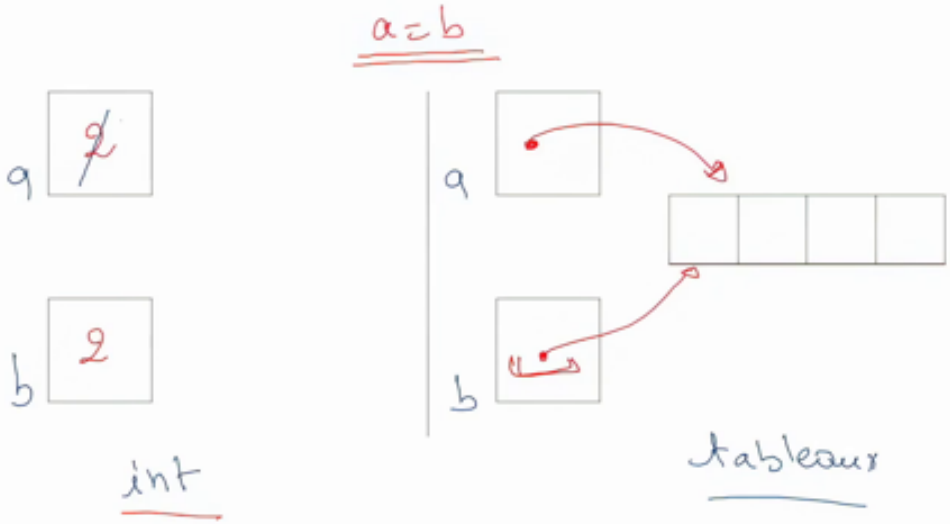
tantôt dans le cas de types évolués, ici nous supposerons que « a » et « b » sont des tableaux. Que va signifier d'écrire a = b dans l'un ou l'autre des cas ? Nous imaginons ici, par exemple, qu'à l'origine « b » contient 2,

notes

résumé

0m 25s

Types de base / Types évolué (rappel)



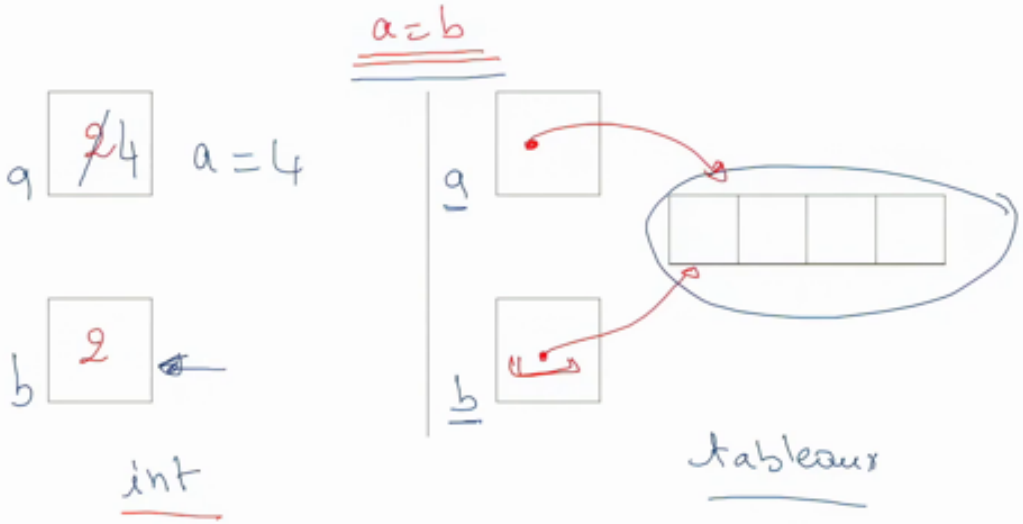
et que ici, à l'origine, « b » contient une référence vers un tableau. Donc si j'exécute une affectation dans le cas des types de base, je me trouve dans la situation où « a » contient au final la même valeur que « b ». Pareil dans ce cas de figure : si j'exécute cette instruction, je vais copier dans « a » la même valeur que « b », c'est-à-dire une référence vers le même tableau, ici en l'occurrence. Pour un type de base, le fait d'avoir réalisé cette affectation n'introduit aucun lien particulier entre « a » et « b ».

notes

résumé

0m 41s

Types de base / Types évolué (rappel)



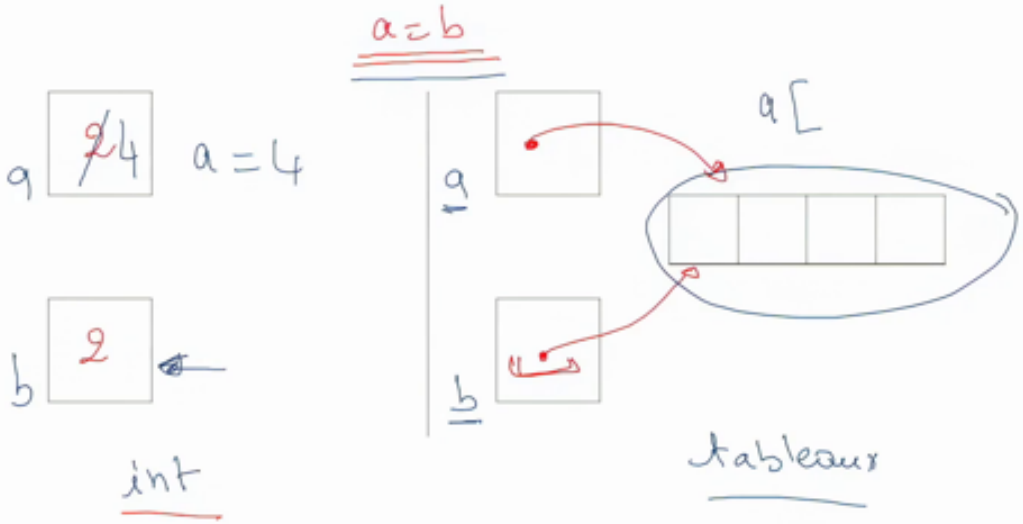
Si par exemple ici je décide après coup de modifier « a » par une affectation telle que celle-ci, cela n'a évidemment aucune incidence sur « b ». Un discours différent a lieu pour les tableaux concernant l'objet référencé. Lorsque je réalise ce genre d'affectation, il devient possible de modifier le tableau référencé aussi bien par « a » que par « b »,

notes

résumé

1m 17s

Types de base / Types évolué (rappel)



ce qui veut dire que, par exemple ici, si j'écris a [0] = 300,

notes

résumé

1m 37s

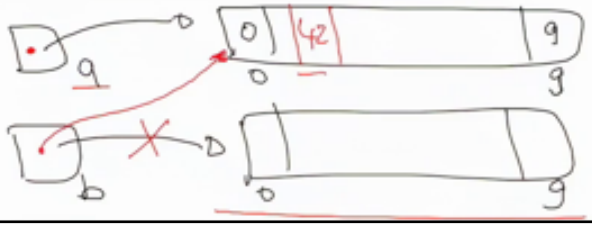
Tableaux : sémantique de l'opérateur =

```
// Les tableaux a et b pointent vers deux emplacements
// différents en memoire
int[] a = new int[10]; // tableau de 10 entiers
int[] b = new int[10]; // tableau de 10 entiers

for (int i = 0; i < a.length; ++i) {
    a[i] = i; // remplissage du tableau pointé par a
}
b = a; // opérateur =  (affectation)
System.out.println("a[2] vaut " + a[2] + " et b[2] vaut " + b[2]);
a[2] = 42;
System.out.println("a[2] vaut " + a[2] + " et b[2] vaut " + b[2]);
```

ce qui affiche :

a[2] vaut 2 et b[2] vaut 2
a[2] vaut 42 et b[2] vaut 42



b [0] vaudra aussi 300. Donc, dans le cas des types de base, il y a une réelle autonomie entre les deux objets, la modification d'un des objets n'a absolument aucune incidence sur l'autre, par contre, lorsque je réalise une affectation sur des objets comme des tableaux, il existera une dépendance par rapport à l'objet référencé. Le message ici est que, contrairement au type de base, réaliser une affectation entre deux variables de type tableau va créer une dépendance entre ces deux objets. Ici, nous partons d'une situation où nous avons à l'origine deux tableaux complètement distincts, autonomes en mémoire, donc une première variable « a » qui contient une référence vers un tableau à 10 cases, et une seconde variable « b » qui contient une référence vers un second tableau à 10 cases, à l'origine complètement autonome. Donc nous commençons par remplir le tableau a simplement au moyen d'une boucle « for », en mettant dans chaque case les mêmes valeurs que les indices, donc nous aboutissons à cette situation mémoire, et nous exécutons notre fameuse affectation de « a » dans « b ». Donc par l'affectation, nous allons mettre dans « b » la même adresse que celle contenue dans « a », ce qui voudra dire que désormais « b » va pointer vers ce tableau. Ce lien est cassé, ces zones en mémoire ne sont plus accessibles. Et ici, si je modifie la troisième entrée du tableau via « a »,

notes

résumé

1m 41s

Tableaux : utilisation (rare) de l'opérateur =

EPFL

a = b

A moins de vouloir deux noms de variables pour le même tableau, il n'y a pas d'intérêt à vouloir assigner un tableau un à autre

☞ l'utilisation de l'opérateur = pour les tableaux est donc rare !

Pour avoir deux tableaux distincts `a` et `b` qui ont les mêmes valeurs il aurait fallu utiliser :

```
for(int i = 0; i < a.length; ++i) {
 b[i] = a[i];
}
```

Attention : il faut que `b.length` ≥ `a.length` !

je vois que la même entrée du tableau accédée via « b » est également modifiée. On voit donc que l'affectation de deux tableaux au moyen de = n'a en réalité de sens que si l'on souhaite avoir deux noms différents pour un même tableau, ce qui est relativement rare. Dans le cas le plus général, par l'affectation on souhaiterait plutôt faire en sorte que chaque case du tableau « b » soit mise dans des cases correspondantes du tableau a, tout en garantissant que chacun des deux tableaux garde son autonomie. A ce moment-là, il faut procéder plutôt au moyen d'une boucle, qui va permettre de copier chacune des cases des valeurs des cases

notes

résumé

3m 13s

du premier tableau dans la case correspondante du second tableau, en étant attentif, bien sûr, à rester dans les bornes licites ! Donc si, par exemple, nous choisissons comme condition d'arrêt la taille du tableau a, eh bien, pour que cette boucle ne provoque pas d'erreur, il va être nécessaire que la taille de « b » soit au moins égale à celle de a, autrement il faudrait formuler la condition d'arrêt de façon un peu différente. de façon un peu différente.

notes

résumé

3m 49s