Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-04-5-tableaux-multi-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Tableaux de taille fixe. Tableaux unidimensionnels. 1d. Élément d'indice. Nombre de lignes. Nombre de joueurs. Exemple concret. Grandes dimensions. Nombre de parties. Différentes notes. Structure de données. Tableau de taille fixe. Tableau de tableaux. Mécanisme d'indexation. Point de vue de la syntaxe.

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

vers la vidéo

EPFL

Tableaux à plusieurs dimensions
(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

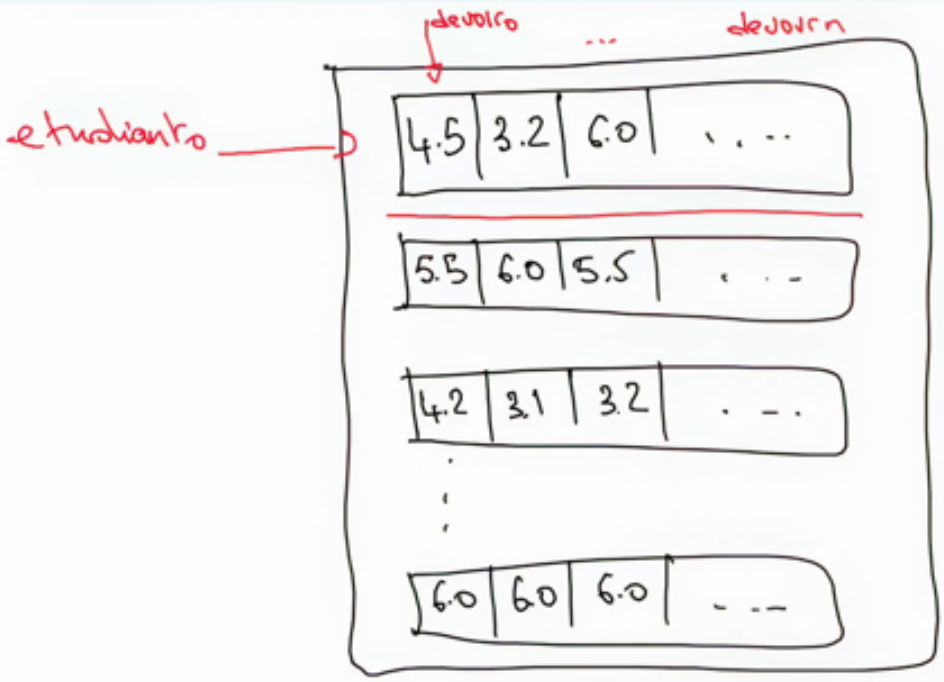
...

notes

résumé

0m 0s

Tableaux à plusieurs dimensions



Les tableaux de taille fixe auxquels nous nous sommes intéressés jusqu'ici étaient des tableaux unidimensionnels. Ils nous permettaient de stocker des séquences simples de valeurs. Dans certaines situations, il est nécessaire d'avoir recours à des tableaux de plus grandes dimensions. Illustrons ceci par un exemple concret. Supposons par exemple que je souhaite stocker les différentes notes obtenues aux différents devoirs à rendre dans le cadre de ce cours pour un étudiant, un seul. Donc ici, je peux utiliser un tableau à 1D qui s'appellerait par exemple notes et qui, pour chacune de ses entrées, contiendrait une note à un devoir. Donc ici ça serait la note du devoir 0, donc du premier devoir. Ici ça serait la note du devoir 1 et ainsi de suite. Donc ici, je réussis à comptabiliser de façon satisfaisante, les notes d'un seul étudiant aux différents devoirs qu'il a accompli dans le cadre de ce cours. Supposons maintenant que je souhaite comptabiliser non pas les notes d'un seul étudiant mais de tous les étudiants suivant le cours. A ce moment-là, j'aurais besoin plutôt d'utiliser un tableau à 2D ici, où la première dimension qui correspond ici aux lignes, correspondrait à un étudiant. Donc ici j'aurais le premier étudiant du cours. Et chacune des colonnes représenterait en fait des notes obtenues aux devoirs par un étudiant. Donc ici par exemple, j'aurais la note du premier devoir, et ici la note du dernier devoir. Donc ici je vois que j'ai besoin d'avoir recours à une structure de données plus complexe, un tableau à 2D

notes

résumé

0m 1s

Tableaux à plusieurs dimensions

EPFL

Comment déclarer un tableau à plusieurs dimensions ?

☞ On ajoute simplement un niveau de [ ] de plus :

C'est en fait un tableau de tableaux...

Exemples :

```
double[][] statistiques =
            new double[nbCantons][nbCommunes];

int[][] scores =
          new int[nbJoueurs][nbParties];
```

3 4

`scores[i]` est un tableau de `nbParties` entiers

☞ `scores` est bien un tableau de tableaux.

où j'ai des lignes correspondant aux étudiants et des colonnes correspondant aux devoirs rendus et les tableaux à plusieurs dimensions permettent de répondre exactement à ce besoin. En JAVA, un tableau de taille fixe à 2D n'est autre qu'un tableau de tableaux. En réalité, c'est un petit peu plus complexe que ça puisque nous savons qu'en JAVA, les tableaux sont manipulés au travers de références. Donc le schéma plus correct serait quelque chose qui ressemblerait à ceci : dans chaque case, nous avons une référence vers un tableau. Donc le schéma que vous avez ici sous les yeux, correspond en fait à une structure qui a cette allure en JAVA. Il s'agit bien d'un tableau de tableaux. Du point de vue de la syntaxe, ajouter une dimension supplémentaire à un tableau en JAVA revient simplement à ajouter un niveau de crochet supplémentaire. Vous en avez ici deux exemples concrets sous les yeux. Et décortiquons maintenant de façon un peu plus détaillée l'un de ces exemples. Ici, je déclare un tableau à 2D, j'ai deux niveaux de crochets, qui contient de façon homogène des entiers et qui s'appelle « score ». Donc il s'agit d'un tableau à 2D, je vais donc l'initialiser par une construction analogue à ce que je connais pour les tableaux à 1D. Ici j'ai un certain nombre de lignes, qui correspond à un certain nombre de joueurs, j'imagine par exemple que j'ai au plus 3 joueurs qui m'intéressent. Et des colonnes, un certain nombre de colonnes qui correspondent à un nombre de parties jouées par les joueurs, donc j'imagine que j'en ai par exemple, au plus, 4. Le but du tableau « score » est de comptabiliser pour chacun des joueurs, les scores obtenus à chacune des parties. Donc concrètement, à quelle structure je vais aboutir en mémoire suite à une telle déclaration ? Et bien ici j'ai déclaré un tableau à 3 entrées, qui se traduit

notes

résumé

1m 37s

Tableaux à plusieurs dimensions

EPFL

Comment déclarer un tableau à plusieurs dimensions ?

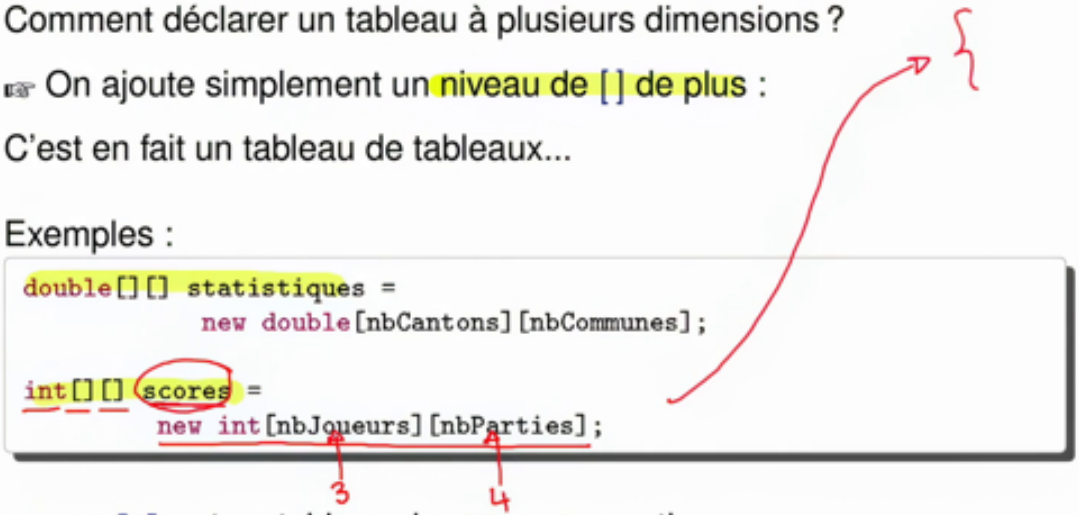
☞ On ajoute simplement un niveau de [ ] de plus :

C'est en fait un tableau de tableaux...

Exemples :

```
double[][] statistiques =
            new double[nbCantons][nbCommunes];

int[][] scores =
          new int[nbJoueurs][nbParties];
```



`scores[i]` est un tableau de `nbParties` entiers

☞ `scores` est bien un tableau de tableaux.

par ceci,

notes

résumé

Tableaux à plusieurs dimensions

EPFL

Comment déclarer un tableau à plusieurs dimensions ?

☞ On ajoute simplement un niveau de [ ] de plus :

C'est en fait un tableau de tableaux...

Exemples :

```
double[][] statistiques =
            new double[nbCantons][nbCommunes];

int[][] scores =
          new int[nbJoueurs][nbParties];
```

`scores[i]` est un tableau de `nbParties` entiers

☞ `scores` est bien un tableau de tableaux.

dont chacune des entrées est elle-même un tableau à 4 entrées, ce qui peut se traduire schématiquement par ceci. Donc ici, pour l'entrée « i », j'ai en fait un tableau qui peut contenir au plus 4 entrées, et chacune des entrées correspond en fait à un score. Donc ceci me permet de comptabiliser pour chaque joueur, le score obtenu à chacune des parties. Donc si je veux accéder au i-ème joueur, j'itère sur la première dimension. Une fois que j'ai accédé à cette dimension, je peux accéder à la seconde dimension qui va me donner les différents scores obtenus aux différentes parties. Donc schématiquement je peux tout à fait représenter ceci par un tableau à 2D,

notes

résumé

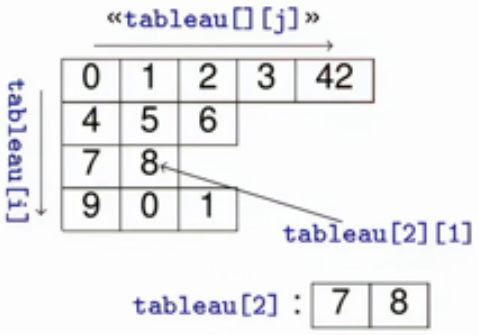
3m 25s

Tableaux à plusieurs dimensions

Les tableaux multidimensionnels peuvent également être initialisés lors de leur déclaration. Il faut bien sûr spécifier autant de valeurs que les dimensions et ceci pour chacune des dimensions.

Exemple :

```
int[][] tableau = {
    { 0, 1, 2, 3, 42 },
    { 4, 5, 6 },
    { 7, 8 },
    { 9, 0, 1 }
};
```



donc ici j'aurais un tableau à 3 lignes et à 4 colonnes. Donc chacune des lignes « i » correspond à un joueur, donc ici ce serait la ligne du joueur i. Et chacune des colonnes « j » correspond en réalité à un score pour la partie j. Donc si je veux connaître le score obtenu par le joueur i lorsqu'il a joué à la partie j, et bien je dois accéder à cette case du tableau à 2D. Les scores pour chaque partie sont en réalité des entiers, ce qui explique la raison pour laquelle j'ai déclaré mon tableau à 2D comme un tableau d'entiers. Si je connais toutes les valeurs du tableau au moment de sa déclaration, il est tout à fait possible de l'initialiser au moment de sa déclaration par une tournure de cette nature qui ressemble beaucoup à ce qu'on a fait pour les tableaux à 1D. Il faut bien sûr spécifier autant de valeurs que de dimensions, et ceci pour chacune des dimensions. Ce que vous avez ici sous les yeux correspond à un tableau de 4 lignes. Et nous voyons que chaque ligne correspond à un tableau, et que le nombre de colonnes n'a pas à être forcément le même, en JAVA, pour toutes les lignes. Donc nous pouvons parfaitement construire un tableau qui a cette allure, où pour chaque ligne, on a un nombre de colonnes différent. Donc ici nous voyons bien que nous avons affaire à un tableau de tableaux. Chaque entrée individuelle du premier tableau correspond à un tableau.

notes

résumé

4m 13s

Le mécanisme d'indexation qui me permet d'accéder aux différentes valeurs du tableau a exactement la même signification que pour les tableaux à 1D. Ici par exemple, si j'accède à « tableau de [2] », je suis en train d'accéder aux éléments de ce premier niveau de tableau. Donc ici, sachant que les éléments sont numérotés de 0 à « taille - 1 », ici j'ai l'élément d'indice 0, d'indice 1, d'indice 2, donc lorsque j'accède à l'élément d'indice 2, je suis sur ce tableau là précisément, ce que je retrouve ici. Donc sachant que « tableau [2] » est lui-même un tableau, je peux accéder maintenant à sa j-ème case par le second indice. Donc ici, sachant que les indices sont toujours numérotés de 0 à « taille - 1 », l'élément d'indice 1 sera celui-ci ; ce qui correspond dans le schéma à cela, et dans ma déclaration d'initialisation à cet élément-là. d'initialisation à cet élément-là.

notes

résumé

5m 49s