Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-04-1-tableaux-intro-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Tableau de données. Tableaux de taille. Scores de joueurs. Manipule des données. Tableaux de données. Type de valeurs. Façon générale. Taille du tableau. Petit programme. Programme java capable. Moyenne de l'ensemble de ces scores. Utilité des tableaux. Dernier point. Données de type élémentaires. Nombre de joueurs.

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

vers la vidéo

EPFL

Tableaux : introduction
(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

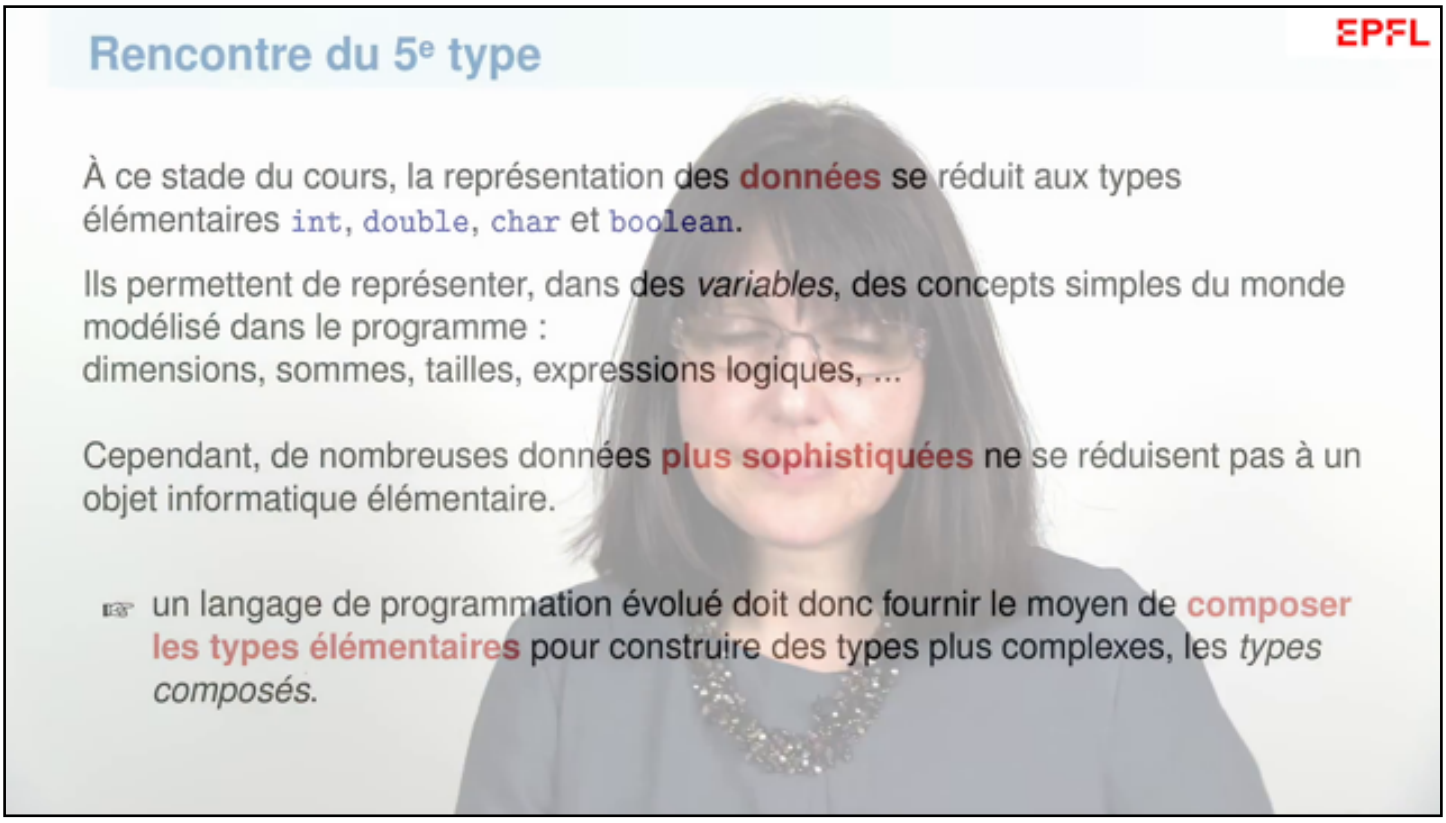
0m 0s

Nous avons vu jusqu'ici que tout programme manipule des données

notes

résumé

0m 1s



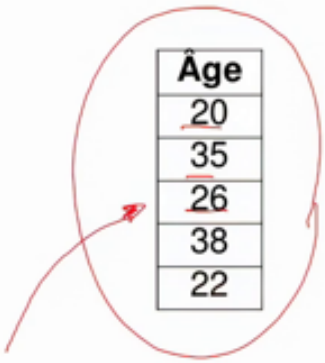
et que ces données sont stockées dans ce qu'on appelle des variables. Nous avons également vu que à toute variable en java doit être associé un type qui représente le type de valeurs qu'on veut stocker dans la variable. Jusqu'à présent les types que nous avons examinés sont des types simples, par exemple, des entiers, des doubles ou des booléens qui permettent effectivement de modéliser des données de type élémentaires comme par exemple, des dimensions, des expressions logiques et des choses de cette nature. Dans de nombreuses situations, il est nécessaire de représenter des données de type plus sophistiqué, de manipuler des ensembles de données comme un tout, et c'est l'objectif de la vidéo d'aujourd'hui.

notes

résumé

0m 6s

Exemples de données structurées



Âge
20
35
26
38
22

Nom	Taille	Âge	Sexe
Dupond	1.75	41	M
Dupont	1.75	42	M
Durand	1.85	26	F
Dugenou	1.70	38	M
Pahut	1.63	22	F

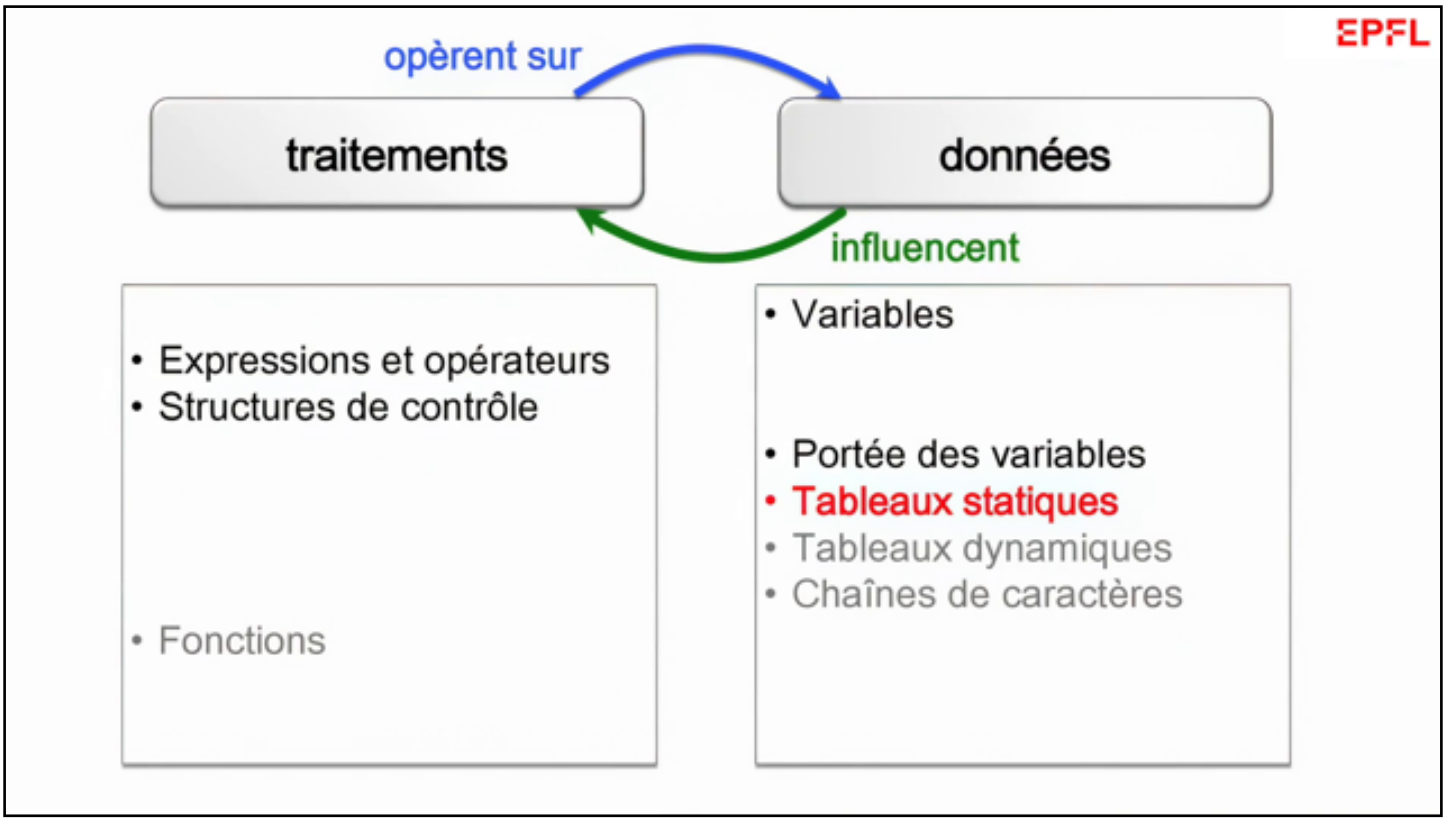
- tableaux
- structures de données hétérogènes
  (par exemple, « un enregistrement » dans le tableau de droite ci-dessus)
- chaînes de caractères
  (par exemple, le « nom »)
- ...

L'objectif du cours d'aujourd'hui est donc d'aller au delà de l'utilisation des types simples comme les entiers, les doubles, les caractères ou les booléens et apprendre à manipuler des données plus sophistiquées en composant des données de type élémentaire. En programmation, de façon générale, un type de données structurées, composées ou évoluées va permettre essentiellement de manipuler un ensemble de données comme un tout. Par exemple, si dans un programme j'ai besoin de modéliser un ensemble d'âge, qui est un ensemble de données homogènes de type entiers ici, il est naturel de penser à quelque chose qui ressemblerait à un tableau de données que je peux manipuler comme un tout.

notes

résumé

0m 45s



Je peux imaginer des représentations sophistiquées de cette nature également où la nature des données est hétérogène, comme par exemple ici, une chaîne de caractères, un double et un entier. De façon plus simple, on peut aussi considérer qu'une chaîne de caractères est un type composé puisqu'une chaîne de caractères est un ensemble de caractères, donc permet de manipuler un ensemble de caractères comme un tout également.

notes

résumé

1m 25s

Petit exemple introductif

Supposons que l'on souhaite écrire un programme de jeu à plusieurs joueurs.

Score	Écart à la moyenne
1000	-1860
1500	-1360
2490	-370
6450	3590
...	...

Nous allons donc aujourd'hui, continuer à explorer les briques de base qui permettent en Java de construire un programme en terme de traitement et de données. Aujourd'hui nous allons travailler plutôt sur le volet des données et examiner comment on peut travailler avec des tableaux de données, plus précisément, nous allons travailler avec des tableaux de taille fixe pour la séquence qui nous occupe aujourd'hui.

notes

résumé

1m 51s

Petit exemple introductif

Supposons que l'on souhaite écrire un programme de jeu à plusieurs joueurs.

Pour illustrer l'utilité des tableaux en programmation, voici un petit exemple d'introduction. Imaginez par exemple que vous souhaitiez écrire un petit programme qui manipule des scores de joueurs. Le but est simplement ici, de permettre d'afficher le score de tous les joueurs, de calculer la moyenne de l'ensemble de ces scores et d'afficher pour chacun des scores, l'écart à la moyenne. Nous allons dans un premier temps essayer d'écrire un programme Java capable de réaliser ces traitements mais en utilisant uniquement les moyens dont nous disposons jusqu'ici, c'est à dire, des données de type élémentaire.

notes

résumé

2m 15s

Solution avec les moyens actuels

```
...
Scanner keyb = new Scanner(System.in);

// Lecture des donnees et calculs
System.out.println ("Score Joueur 1:");
int score1 = keyb.nextInt();
System.out.println ("Score Joueur 2:");
int score2 = keyb.nextInt();
// Calcul de la moyenne
double moyenne = (score1 + score2);
moyenne /= 2;
// Affichages
System.out.println("Score     Ecart Moyenne");
System.out.println(score1 + " " + (score1 - moyenne));
System.out.println(score2 + " " + (score2 - moyenne));
...
```

Nous n'allons pas être très ambitieux pour commencer, nous allons imaginer que nous avons simplement deux joueurs à représenter.

notes

résumé

2m 49s

Solution avec les moyens actuels (2)

```
System.out.println ("Score Joueur 1:");
int score1 = keyb.nextInt();
System.out.println ("Score Joueur 2:");
int score2 = keyb.nextInt();
System.out.println ("Score Joueur 3:");
int score3 = keyb.nextInt();
System.out.println ("Score Joueur 4:");
int score4 = keyb.nextInt();
System.out.println ("Score Joueur 5:");
int score5 = keyb.nextInt();

// calcul de la moyenne
double moyenne = (score1 + score2 + score3 + score4 + score5);
moyenne /= 5;
```

Comment faire les affichages ?

Nous avons donc à modéliser les scores de deux joueurs, ce que l'on peut faire avec les moyens existants au travers de deux variables de type entier : la variable score 1, la variable score 2. On imagine que dans ce programme, ces valeurs sont obtenues au travers d'une interaction clavier, comme ici. Une fois que ces valeurs sont obtenues, le calcul de la moyenne peut se faire simplement en sommant les deux scores et en divisant la somme par deux. Et ensuite nous pouvons procéder à l'affichage des scores et de l'écart à la moyenne pour chacun des scores. Ici nous affichons le score 1 et l'écart à la moyenne du score 1 et nous faisons pareil pour le score 2. Une question qui va naturellement se poser ici c'est, comment procéder si on se trouve avec davantage de joueurs ? Bien évidement, il faudra ajouter autant de variables que l'on a de joueurs et typiquement dans le cadre de cet exemple où on a cinq joueurs, on se trouverait dans l'obligation de déclarer cinq variables 'score', une pour chaque joueur, ce qui est déjà relativement laborieux. Le calcul de la moyenne se complexifie puisque désormais on doit sommer les valeurs de cinq variables et puis, comment faire les affichages ? On se dit qu'on doit répéter pour chacun des scores la ligne d'instruction qui permet d'afficher

notes

résumé

2m 59s

Solution avec les moyens actuels : limites

EPFL

Mais

1. comment l'écrire (`scorei` n'est pas correct) ?
2. comment faire si on veut considérer 100, 1000... joueurs ?
3. comment faire si le nombre de joueurs n'est pas connu au départ ?

le score et l'écart à la moyenne. On aimerait bien contourner ce côté un peu laborieux de répéter chacune des instructions et donc on va penser naturellement à utiliser une boucle. Comment faire si on souhaite procéder à l'affichage des différents scores et des écarts à la moyenne en utilisant une boucle ? Une solution naturelle ici serait d'imaginer d' employer une boucle for, qui nous permettrait d'itérer sur chacun des scores et pour chacun de ces scores réaliser les affichages souhaités. On peut donc imaginer écrire cette boucle for dans cet esprit. On aurait un compteur qui itérerait du premier score jusqu'au dernier et pour chacun des scores, on réaliserait les affichages souhaités. Si l'on examine cependant ce programme un peu plus en détail, donc la boucle for et même l'intégralité du programme, on se rend compte cependant, que cette approche ne répond pas de façon complètement satisfaisante à tous les besoins qu'on peut envisager dans ce contexte. Imaginons par exemple, que je souhaite procéder avec plus de joueurs que 5, c'est à dire 100 ou 1000. Dans ce cas-là le programme prendrait des dimensions ingérables, il faudrait introduire un millier de variables par exemple, complexifier le calcul de la moyenne, ensuite si on regarde la façon que nous avons eu d'écrire le score ici pour accéder au ième score de la séquence, en réalité cette syntaxe n'est pas une syntaxe valable en Java pour désigner le ième élément d'une séquence. Et puis, dernier point, l'approche que nous avons utilisée ne répond pas non plus de façon satisfaisante à la situation où le nombre de joueurs n'est pas connu au départ. Donc, si l'on veut écrire un programme de cette nature

notes

résumé

4m 13s

Solution avec tableau

```
System.out.print ("Donnez le nombre de joueurs:");
int n = keyb.nextInt();
if (n > 0) {
   double moyenne = 0;
   int scores [] = new int[n];

    // Lecture des scores
   for (int i = 0; i < n; ++i) {
      System.out.println ("Score Joueur " + i + " :");
      scores[i] = keyb.nextInt();
      moyenne += scores[i];
   }
   moyenne /= n; // calcul de la moyenne

    // Affichages
   System.out.println(" Score " + " Ecart Moyenne");
   for (int i = 0; i < n ; ++i) {
      System.out.println(scores[i] + " " + (scores[i] - moyenne));
   }
}
```

l'outillage dont nous disposons jusqu'ici ne répond pas de façon satisfaisante à nos besoins dans ce contexte et les tableaux sont le moyen de remédier à cela.

notes

résumé

5m 49s

Les tableaux

EPFL

Un tableau est une collection de valeurs *homogènes* (= même type)

Exemple : Tableau `scores` contenant 4 `int`

1000	1500	2490	6450
`scores[0]`	`scores[1]`	`scores[2]`	`scores[3]`

Utilisation des tableaux : *stockage* de *plusieurs* variables de *même* type

On pourra définir des tableaux d'`int`, de `double`, de `char`, ...

Pour vous donner une idée concrète de ce que c'est que la notion de tableau voici comment s'écrirait le programme précédant en utilisant ce concept. Donc l'idée serait d'utiliser un nouveau type de données, le type ici "tableau d'entiers", dont nous verrons la syntaxe plus en détails dans le cadre de cette vidéo ; et, principalement, lorsque l'on déclare un tableau, on peut manipuler une séquence de données comme un tout et accéder au ième élément de la séquence au moyen de l'indexation. Donc nous voyons que le programme précédent s'écrit de façon beaucoup plus concise, nous pouvons notamment gérer toute la partie qui lit les scores de façon unifiée au travers d'une boucle for. Ensuite, peu importe le nombre de scores, nous écrirons le programme de la même façon que nous aillons à faire à 5 ou à 1000 joueurs et puis nous pouvons également désormais gérer le fait que le nombre de joueurs n'est pas connu au départ et donc lire le nombre de scores avant d'exécuter le calcul que nous voulons réaliser sur les scores. Un tableau en programmation est donc un type de données qui nous permet de manipuler une collection de valeurs homogènes comme un tout. Les valeurs sont toutes du même type dans un tableau, ici vous avez sous les yeux un exemple représentant un tableau d'entiers. Toutes les valeurs contenues dans le tableau score sont de type 'int'.

notes

résumé

6m 1s

Les différentes sortes de tableaux

Il existe en général quatre sortes de tableaux :

		taille initiale connue *a priori* ? non	taille initiale connue *a priori* ? oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	1.	2.
	non	3.	4.

2 à 7
5

On peut bien sûr définir en Java des tableaux de n'importe quel type connu, tableau d'entiers, de doubles, de char et de n'importe quel autre type connu y compris des tableaux de tableaux. En programmation de façon général on peut considérer qu'il existe a priori quatre types de tableaux, ceci en fonction de la réponse aux questions suivantes. Première question : est-ce que la taille du tableau est connue a priori ? C'est à dire, au moment où je commence à écrire le programme je sais quelle taille a le tableau, ou je ne sais pas quelle taille a le tableau. Donc on peut avoir deux réponses possibles à cette question, et, deuxième question : est-ce que la taille peut varier lors de l'utilisation du tableau ? C'est à dire qu'une fois que j'ai fixé une taille à mon tableau, est-ce que je peux faire varier cette taille pendant que le programme s'exécute ou est-ce que cette taille va rester fixée une fois pour toutes ? Le type de tableau le plus souple, le plus général qui répond au maximum de besoins est sans aucun doute, celui qui se trouve dans cette configuration, c'est à dire, un tableau dont la taille initiale n'est pas connue au moment où je rédige le programme. On peut fixer cette taille au moment où le programme s'exécute mais je ne la connais pas quand j'écris le programme et cette taille peut également varier pendant que le programme s'exécute. Si l'on imagine par exemple que je souhaite écrire un programme qui gère les âges des inscrits à ce cours. Donc ce nombre n'est pas connu au moment où j'écris le programme mais va être déterminé au moment où je lance l'exécution du programme avec le nombre d'inscrits effectifs que j'ai à ce moment là, et on peut imaginer qu'en cours d'exécution, j'ai de nouveaux inscrits, donc le tableau va augmenter de taille.

notes

résumé

7m 13s

Les différentes sortes de tableaux

Il existe en général quatre sortes de tableaux :

		taille initiale connue *a priori* ?	
		non	oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	1.	2.
	non	3.	4.

2 à 7

5

Je peux également imaginer qu'il y ait eu des gens qui se désinscrivent aux cours, et à ce moment là, la taille du tableau va diminuer. À l'autre extrême, il existe des tableaux dont la taille est connue au moment où j'écris le programme et dont la taille n'est pas amenée à changer en cours d'exécution du programme, donc cette configuration là. Par exemple, si je veux écrire un programme qui manipule des vecteurs à 2 dimensions, donc je sais que mon vecteur à 2 dimensions a deux composantes exactement et pas plus. Donc, une composante pour X et une composante pour Y et je sais qu'un vecteur à 2 dimensions va rester un vecteur à 2 dimensions pendant toute l'exécution du programme et donc sa taille ne va pas changer pendant que j'exécute le programme. Situation intermédiaire, le cas où un tableau a une taille qui n'est pas connue au moment où je rédige le programme, mais une fois que cette taille est fixée elle ne va plus évoluer pendant que le programme s'exécute, donc ce cas de figure là. Par exemple on peut imaginer que j'écris un jeu qui se joue de 2 à 7 joueurs. Au moment où j'exécute le programme, je ne sais pas combien de joueurs vont intervenir exactement dans le jeu. Donc, quand le programme s'exécute il va demander combien de joueurs interviennent. Par example, il y en a 5.

notes

résumé

Une fois que ces 5 joueurs interviennent dans la partie, la partie doit se dérouler avec les 5 joueurs et pas plus. Donc la taille du tableau n'est pas amenée à évoluer pendant que le programme s'exécute. Dernier cas de figure enfin, peut-être plus difficile à illustrer, un peu moins intuitif, celui où la taille du tableau est connue a priori au moment où j'écris le programme et dont la taille peut varier pendant que j'exécute le programme. Par exemple, je veux représenter le nombre de cantons d'un pays. Donc, ce nombre de cantons est fixé, connu a priori. Donc au moment où j'écris le programme, je sais, par exemple qu'il y en a 26. Et puis on peut imaginer que à un moment donné un nouveau canton se crée dans le pays ou que deux cantons fusionnent en un seul. Donc la taille du tableau va être amenée à augmenter ou diminuer pendant que le programme s'exécute. Ce dernier cas de figure est sans doute le moins naturel et le moins fréquent dans la pratique. La plupart des langages de programmation ne vont pas offrir un outil spécifique, dédié pour la gestion de chacun des cas qui se présentent ici. Par exemple, en Java nous disposons principalement de 2 outillages, celui qui permet de gérer des tableaux de taille dynamique. Ici par exemple, au moyen d'un type prédéfini de Java qu'on appelle les «ArrayList», et qui répond aux besoins des situations 1 et 2. Et puis les tableaux de taille fixe, qui vont nous occuper dans le cadre de la vidéo d'aujourd'hui, et qui répondent aux besoins des situations 3 et 4. des situations 3 et 4.

notes

résumé

10m 1s