Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-05-4-tabdyn-pt3

Concepts (extraits des sous-titres générés automatiquement) :

Exemples de tableaux dynamiques. Valeurs de type évolué. Type chaîne de caractères. Contenu d'un tableau dynamique. Valeurs de type élémentaire. Type évolué. Dernier élément. Déclaration du tableau. Type de base. Seconde valeur. Valeur d'indice. Tableaux dynamiques d'entiers. Nombre de conventions. Premier test. Premières versions de java.

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

vers la vidéo

EPFL

Tableaux dynamiques
(Partie 3)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Dans tous les exemples de tableaux dynamiques montrés jusqu'ici

notes

résumé

0m 1s

Que faire pour des types de base ?

EPFL

En Java, à chaque type de base correspond un type évolué prédéfini :

- `Integer` est le type évolué correspondant à `int`
- `Double` est le type évolué correspondant à `double`
- etc. . . .

☞ Utiles dans certains contextes (typiquement les `ArrayList`)

☞ La conversion du type de base au type évolué se fait automatiquement

le contenu était de type chaîne de caractères c'est évidemment un choix délibéré qui répondait à la contrainte que le contenu d'un tableau dynamique en Java doit systématiquement être de type évolué. Il peut cependant se trouver que l'on ait besoin de travailler avec des tableaux dynamiques d'entiers de tables. Comment procéder ? xCe qu'il suffit de savoir à ce stade c'est que Java offre pour chaque type de base un type évolué correspondant. Par exemple pour le type « int » vous disposez du type « Integer », avec un « grand I » qui est équivalent et qui est de type évolué. De même pour le type « double » vous allez avoir un équivalent « Double », avec un « grand D » et ceci est valable pour chaque type de base. Ces types évolués d'un genre particulier s'avèrent particulièrement utiles dans certains contextes typiquement pour les « ArrayList ». Pourquoi ? Parce que nous avons vu que dans un « Arraylist » il n'était possible de stocker que des valeurs de type évolué. Je ne peux ainsi jamais déclarer dans un programme Java un « ArrayList double », avec un petit d, ou un « ArrayList int ». Par contre je peux tout à fait déclarer un « ArrayList » de « Integer ». Une fois la déclaration du tableau faite en respectant la contrainte que le contenu est de type évolué, je peux ensuite tout à fait mettre des valeurs de type élémentaire dans mon tableau pour une raison très simple, c'est que la conversion de types de base en types évolués, peut alors se faire automatiquement. Ce n'était pas le cas dans les premières versions de Java mais c'est désormais comme ça que cela fonctionne et je peux ainsi travailler avec des tableaux dynamiques d'entiers ou double en étant presque affranchi

notes

résumé

0m 6s

Que faire pour des types de base ?

EPFL

En Java, à chaque type de base correspond un type évolué prédéfini :

- `Integer` est le type évolué correspondant à `int`
- `Double` est le type évolué correspondant à `double`
- etc. . . .

☞ Utiles dans certains contextes (typiquement les `ArrayList`)

☞ La conversion du type de base au type évolué se fait automatiquement

de la contrainte de l'utilisation des types évolués. Voyons ça maintenant sur un exemple concret

notes

résumé

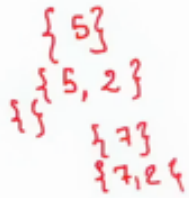
Exemple

Ecrivons un programme qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
Saisie de 3 valeurs :
Entrez la valeur 0 : 5
Entrez la valeur 1 : 2
Entrez la valeur 2 : 0
Entrez la valeur 0 : 7
Entrez la valeur 1 : 2
Entrez la valeur 2 : -4
Entrez la valeur 1 : 4
Entrez la valeur 2 : 12

-> 7 4 12
```



Nous souhaitons ici écrire un petit programme qui remplit un tableau dynamique d'entiers au moyen de nombres strictement positifs. On suppose que les valeurs vont être demandées à l'utilisateur et nous souhaitons également adhérer à un certain nombre de conventions si l'utilisateur introduit la valeur 0 ce sera pour nous la convention que l'on souhaite recommencer le remplissage depuis le début et si l'utilisateur introduit un nombre négatif ce sera pour nous la convention que l'on souhaite effacer le dernier élément introduit dans le tableau. Le déroulement du programme pourrait avoir l'allure suivante donc on va demander à l'utilisateur de saisir trois valeurs strictement positives, et on va lui demander chacune de ces valeurs en séquence, ici on lui demande d'introduire la valeur d'indice 0 et on suppose qu'il a introduit 5, cette valeur étant strictement positive, on va la stocker dans le tableau. On lui demande ensuite la seconde valeur, celle d'indice 1, cette valeur est strictement positive, on va l'introduire à nouveau dans le tableau. Ensuite on lui demande d'introduire la dernière valeur, la valeur d'indice 2, mais il se trouve que l'utilisateur entre un 0 ce qui correspond à la convention recommencer depuis le début, ce qui veut dire qu'à ce stade, on veut faire en sorte que le tableau se vide à nouveau ; donc puisque le tableau s'est vidé la prochaine valeur qu'on va demander à l'utilisateur, c'est de nouveau la valeur d'indice 0, ici l'utilisateur introduit un 7, nous allons donc avoir ce tableau il introduit ensuite un 2 nous allons avoir ce tableau.

notes

résumé

1m 44s

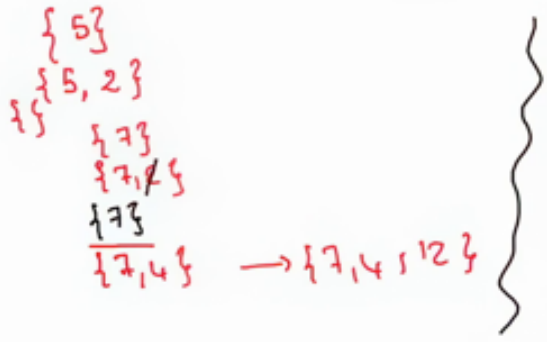
Exemple

Ecrivons un programme qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
Saisie de 3 valeurs :
Entrez la valeur 0 : 5
Entrez la valeur 1 : 2
Entrez la valeur 2 : 0
Entrez la valeur 0 : 7
Entrez la valeur 1 : 2
Entrez la valeur 2 : -4
Entrez la valeur 1 : 4
Entrez la valeur 2 : 12

-> 7 4 12
```



Ensuite il introduit pour la dernière valeur de nouveau, une valeur négative ce qui correspond pour nous à la seconde convention, celle d'effacer le dernier élément. Le dernier élément introduit est le 2 donc nous souhaitons faire en sorte que désormais le tableau soit ceci, et c'est pourquoi à la prochaine étape, on va demander à l'utilisateur d'introduire la seconde valeur puisque le tableau ne contient désormais qu'une seule valeur. A ce moment là l'utilisateur entre un 4, que nous allons stocker dans le tableau, il introduit ensuite en dernière instance un 12 et nous allons avoir ce tableau. Puisque nous avons bien les trois valeurs souhaitées, nous avons construit notre tableau tel que nous l'avons souhaité et nous allons au final aboutir à un tableau qui contient 7, 4 et 12, qui sont des valeurs strictement positives. Nous voyons ici que tout au long de l'exécution de ce programme,

notes

résumé

3m 13s

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
ArrayList<Integer> vect = new ArrayList<Integer>();

System.out.println("Donnez la taille voulue : ");
int taille = scanner.nextInt();
System.out.println("Saisie de " + taille + " valeurs :");
while (vect.size() < taille) {
  System.out.println("Entrez la valeur " + vect.size() + " : ");
  int val = scanner.nextInt();
  if ((val < 0) && (!vect.isEmpty())) { vect.remove(vect.size() - 1); }
  else if (val == 0)                  { vect.clear();                 }
  else if (val > 0)                   { vect.add(val);                }
}
```

la taille de notre tableau croît et décroît au gré de nos besoins. Donc c'est typiquement un cas où l'utilisation d'un tableau dynamique est particulièrement adaptée. Voyons maintenant comme tout cela se traduit sous la forme d'un programme Java. Les traitements montrés dans l'exemple précédent peuvent être mis en oeuvre au moyen d'un petit programme Java ayant cette allure. Nous commençons par déclarer un tableau dynamique d'entiers, comme nous ne pouvons pas créer de « ArrayList » de type « int », de type élémentaire, nous utilisons le type « Integer » ; notre tableau dynamique s'appelle « vect » et il est initialisé comme un tableau dynamique vide

notes

résumé

4m 13s

Exemple

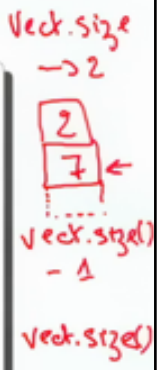
EPFL

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
ArrayList<Integer> vect = new ArrayList<Integer>();

System.out.println("Donnez la taille voulue : ");
int taille = scanner.nextInt();
System.out.println("Saisie de " + taille + " valeurs :");
while (vect.size() < taille) {
  System.out.println("Entrez la valeur " + vect.size() + " : ");
  int val = scanner.nextInt();
  if ((val < 0) && (!vect.isEmpty())) { vect.remove(vect.size() - 1); }
  else if (val == 0)                  { vect.clear();                 }
  else if (val > 0)                   { vect.add(val);                }
}
```



donc au final la variable « vect » contiendra la référence à un tableau dynamique vide d'entiers. Nous demandons ensuite à l'utilisateur combien de valeurs il veut introduire dans son tableau et nous saisissons la taille voulue par l'utilisateur pour le tableau au moyen d'une interaction clavier ; nous demandons ensuite à l'utilisateur de saisir les différentes valeurs pour remplir le tableau et donc pour cela nous allons utiliser une boucle puisqu'il va introduire potentiellement plusieurs valeurs. La boucle est telle que tant que la taille actuelle du tableau n'a pas atteint la taille voulue alors nous continuons à boucler. A chaque itération de la boucle nous allons demander à l'utilisateur de saisir une des valeurs du tableau. Mais quelle valeur précisément ? Donc ici nous voulons indiquer à l'utilisateur l'indice de la valeur à saisir. Supposons que l'état de remplissage du tableau soit le suivant. Donc à ce moment là « vect.size() » vaut 2 et le dernier élément a pour indice 1 c'est-à-dire « vect.size() - 1 », c'est-à-dire que la prochaine valeur du tableau que nous voulons saisir a bien pour indice « vect.size() » et c'est ce que nous retrouvons ici. Nous allons demander à l'utilisateur d'entrer la valeur qui est à la position « vect.size() » qui va occuper la position « vect.size() ». La valeur à mettre dans le tableau est à nouveau saisie via une interaction clavier. et c'est à ce moment que l'on commence à tester les différents cas pour adhérer aux différentes conventions exposées dans l'exemple de déroulement précédent. Une des conventions que nous voulions mettre en oeuvre est que si la valeur introduite par l'utilisateur est négative cela voudrait dire que nous voulons supprimer le dernier élément introduit dans le tableau. Donc concrètement, nous allons utiliser la méthode « remove » pour réaliser cette suppression. Et à ce moment là il faudrait supprimer

notes

résumé

4m 49s

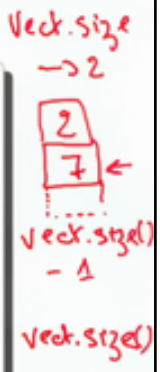
Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
ArrayList<Integer> vect = new ArrayList<Integer>();

System.out.println("Donnez la taille voulue : ");
int taille = scanner.nextInt();
System.out.println("Saisie de " + taille + " valeurs :");
while (vect.size() < taille) {
  System.out.println("Entrez la valeur " + vect.size() + " : ");
  int val = scanner.nextInt();
  if ((val < 0) && (!vect.isEmpty())) { vect.remove(vect.size() - 1); }
  else if (val == 0)                  { vect.clear();                 }
  else if (val > 0)                   { vect.add(val);                }
}
```



le dernier élément du tableau, l'élément d'indice « vect.size() - 1 » tel que nous l'avons vu ici. Cela étant, il faut prendre une précaution nous ne pouvons procéder à cette suppression que si le tableau dynamique n'est pas vide, et donc nous devons combiner ce premier test avec un second test qui va tester si le vecteur, le tableau dynamique, est vide ou non, donc nous n'allons procéder à la suppression que si la valeur est négative, et que le tableau n'est pas vide. Le fait qu'il soit vide est testé avec la fonctionnalité « isEmpty » le fait qu'il ne soit pas vide est exprimé avec la négation ici. Si la condition exprimée ici n'est pas vérifiée nous devons procéder ensuite à d'autres tests nous avions vu que nous voulions adhérer à la convention que si l'utilisateur introduit la valeur 0, cela signifie que nous voulons vider le tableau donc nous procédons à un test alternatif ici. Le fait de vider complètement un tableau dynamique se fait au moyen de la fonctionnalité « clear » donc ici simplement, si la valeur introduite par l'utilisateur est nulle, alors on vide complètement le tableau.

notes

résumé

Exemple

Ecrivons un programme qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
ArrayList<Integer> vect = new ArrayList<Integer>();

System.out.println("Donnez la taille voulue : ");
int taille = scanner.nextInt();
System.out.println("Saisie de " + taille + " valeurs :");
while (vect.size() < taille) {
  System.out.println("Entrez la valeur " + vect.size() + " : ");
  int val = scanner.nextInt();
  if ((val < 0) && (!vect.isEmpty())) { vect.remove(vect.size() - 1); }
  else if (val == 0)                  { vect.clear();                 }
  else if (val > 0)                   { vect.add(val);                }
}
```

Sinon, si nous atteignons ce stade de l'execution cela signifie que soit la valeur introduite est strictement positive, soit le vecteur est nul. Il n'est donc pas strictement garanti que la valeur introduite soit positive, il faut donc le retester, c'est ce que nous faisons ici. Si c'est le cas, si la condition est vérifiée nous pouvons alors ajouter la valeur à notre tableau dynamique et ceci se fait au moyen de la fonctionnalité « add ». Donc nous allons itérer ces traitements tant que le tableau dynamique n'a pas atteint la taille souhaitée par l'utilisateur. Remarquons ici un point assez important. Nous avons construit notre tableau nous l'avons déclaré en utilisant le type évolué « Integer » ceci signifie que dans notre tableau nous n'avons pas directement des valeurs entières comme ceci, et comme je l'ai montré très schématiquement dans le déroulement de l'exemple, mais pour être tout à fait exacte nous avons dans le tableau des références vers des entiers. Donc ici la situation serait la suivante : à la position 0 nous aurions une référence vers une zone mémoire contenant 12 et non pas directement la valeur 12. Comment se fait-il dans ce cas que nous ayons pu introduire une valeur de type « int » dans le tableau comme nous avons fait ici. En effet puisque notre tableau a été déclaré comme contenant des « Integers » nous nous attendons à ne pouvoir ajouter à ce tableau que des valeurs strictement du même type et donc pas directement un entier mais une référence vers un entier, ce que pour l'heure nous ne savons pas construire ou initialiser.

notes

résumé

8m 1s

Comparaison d'éléments

Attention : les éléments d'un tableau dynamique sont toujours des références

☞ Comparaison au moyen de `equals`

```
ArrayList<Integer> tab = new ArrayList<Integer>();

tab.add(2000);
tab.add(2000);

System.out.println(tab.get(0) == tab.get(1));

System.out.println((tab.get(0)).equals(tab.get(1)));
```

Nous avons cependant pu introduire directement un « int » Alors ceci est permis grâce à la conversion automatique qui se fait entre les types de base et les types évolués correspondants. Donc en réalité lorsque nous exécutons cette instruction ce que nous introduisons dans le tableau n'est pas directement la valeur « val » introduite par l'utilisateur, qui elle est de type « int », mais nous introduisons dans le tableau un « Integer » qui est une référence vers la valeur « val » et tout ceci se fait de façon complétement automatique et transparente, nous n'avons pas à nous préoccuper de cette conversion. Donc grace à cette conversion automatique, il devient complètement naturel de pouvoir travailler également avec des tableaux dynamiques d'entiers, de « doubles » ou de type de base; la seule précaution qu'il est indispensable de prendre est que, au moment de la déclaration du tableau, il faut absolument le déclarer en utilisant le type évolué correspondant. Cette conversion automatique entre un type de base et un type évolué correspondant, est ce que l'on appelle en termes techniques « l'autoboxing ». Nous avons vu qu'un « ArrayList » ne peut contenir que des éléments de type évolué, examinons pour finir l'incidence que cela a sur la comparaison des éléments d'un tel tableau. A la première instruction je vais avoir la conversion automatique d'une valeur de type de base en une valeur de type « Integer », mais ce que je vais en réalité ajouter dans mon tableau dynamique ça n'est pas directement la valeur 2000 mais une référence vers cette valeur. Donc je me trouve dans une situation en mémoire qui est la suivante.

notes

résumé

9m 37s

Ici j'ai stocké dans mon tableau non pas directement la valeur 2000 mais son adresse. Donc lorsque j'exécute la seconde instruction c'est exactement la même situation qui se produit, sauf qu'à ce moment j'ai 2 références. Il se trouve que ces références ont un contenu similaire, mais elles ont été créées par deux « add » distinctes et à ce moment là, il s'agit bien de deux zones mémoire distinctes, différentes. Donc ici, si je souhaite comparer le contenu de la première entrée du tableau avec la seconde, par une tournure de cette nature là, le résultat de l'exécution de cette instruction va être en réalité fausse. Pourquoi ? Parce que j'ai bel et bien à faire à 2 zones mémoire distinctes, et donc il ne s'agit pas du même emplacement et la comparaison va retourner fausse. S'il m'intéressait de comparer plutôt les contenus alors comme pour les chaines de caractères je ne dois pas recourir à un test d'égalité avec « == » mais plutôt avoir recours au un test d'égalité avec « equals » donc on accède au premier élément du tableau qui est un « Integer » et on applique sur cet élément la méthode « equals » pour le comparer au second élément du tableau. Et ici, le résultat de l'exécution de cette instruction sera bel et bien « true » car les valeurs sont identiques. Les valeurs pointées par les références sont identiques. les références sont identiques.

notes

résumé

11m 13s