Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-01-4-Ecriture-pt4

Concepts (extraits des sous-titres générés automatiquement) :

Lire des variables. Type entier. Lecture de la valeur d'une variable. Lecture d'un entier nextint. Ensemble de lignes. Fonction nextline. Méthode nextint. Exemple précédent. Expression suivante. Expression b. Lire toute. Fonction nextdouble. Ligne. Nombre décimal. Variable n.

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

vers la vidéo

EPFL

Variables : lecture/écriture
(Partie 4)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

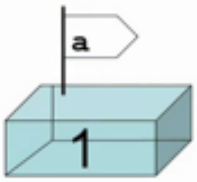
...

notes

résumé

0m 0s

```java
int a = 1;
int b = 2;

a = b;
b = a;

System.out.println(a + ", " + b);
```



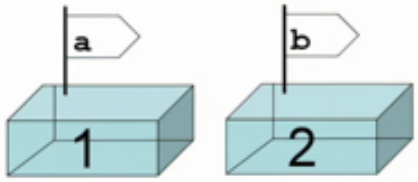
Regardons le déroulement pas-à-pas du code de la question précédente.

notes

résumé

0m 1s

```
int a = 1;
int b = 2;

a = b;
b = a;

System.out.println(a + ", " + b);
```



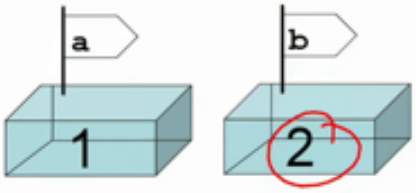
On commence par déclarer une variable a de type entier, que l'on initialise à la valeur 1. Puis on déclare une variable b de type entier que l'on initialise à la valeur 2

notes

résumé

0m 6s

```
int a = 1;
int b = 2;

a = b;
b = a;

System.out.println(a + ", " + b);
```



puis on arrive à une affectation où l'on évalue l'expression b qui retourne 2,

notes

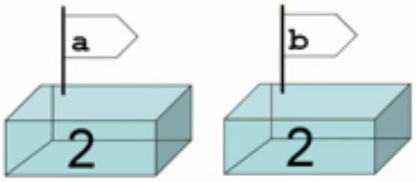
résumé

0m 21s

```
int a = 1;
int b = 2;

a = b;
b = a;

System.out.println(a + ", " + b);
```

Affiche:

2, 2



on va mettre 2 dans a, puis on évalue l'expression suivante, a et on va recopier cette expression dans b

notes

résumé

0m 26s

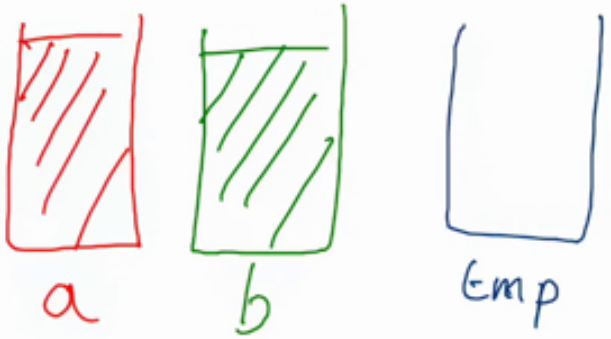
Supposons qu'on ait déclaré et initialisé deux variables `a` et `b`.

Comment échanger leurs valeurs ?

Les instructions:

```
a = b;
b = a;
```

ne **marchent pas**, comme le montre l'exercice précédent.

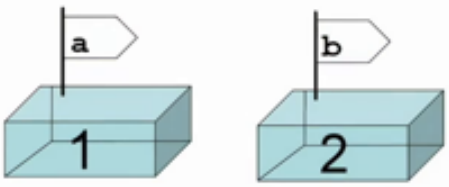


ce qui fait qu'au final, à l'affichage, on aura évaluation de l'expression a, 2, évaluation telle quelle de la chaîne ", " puis évaluation de l'expression b qui vaut 2 et l'affichage sera "2, 2". À ce stade, on peut se demander comment échanger la valeur de deux variables puisque dans l'exemple précédent, on n'a pas échangé les deux variables mais simplement recopié la valeur de b dans les deux variables, a et b. On peut se demander comment procéder pour échanger le contenu de deux verres. Supposons que vous ayez un verre de grenadine et un verre de menthe. Le verre de menthe va s'appeler b le verre de grenadine, a. Comment faire pour échanger le contenu des deux verres ? La bonne solution est d'introduire un troisième verre appelé tmp. Ce n'est pas plus difficile en programmation.

notes

résumé

0m 39s



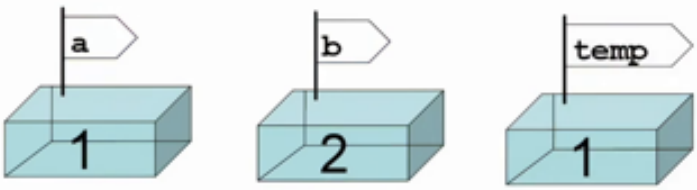

```
int a = 1;
int b = 2;
int temp = a;

a = b;
b = temp;
```

Pour échanger le contenu de deux variables, il suffit d'introduire une variable temporaire temp qui va permettre de transférer les valeurs.

notes

résumé

1m 37s



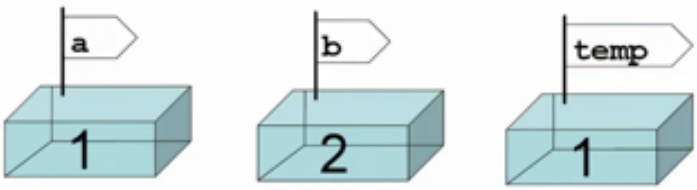

```
int a = 1;
int b = 2;
int temp = a;

a = b;
b = temp;
```

Par exemple, a est initialisée à 1 et b initialisé à 2

notes

résumé

1m 46s




```
int a = 1;
int b = 2;
int temp = a;

a = b;
b = temp;
```

et on introduit une variable temporaire temp

notes

résumé

1m 54s

```java
import java.util.Scanner;

class DemanderVariable
{
  public static void main(String [] args)
  {
    Scanner keyb = new Scanner(System.in);
    System.out.println("Entrez une valeur pour n");
    int n = keyb.nextInt();

    int nCarre = n * n;

    System.out.println("La variable n contient " + n);
    System.out.println("Le carre de " + n + " est " + nCarre + ".");
    System.out.println("Le double de n est " + 2 * n + ".");
  }
}
```

dans laquelle on recopie la valeur de a puis on recopie la valeur de b dans a. Puisque la valeur de a est sauvegardée dans temp, on peut remplacer dans a la valeur initiale par celle de b Enfin, on recopie la valeur sauvegardée dans temp dans la variable b, ce qui met ici la valeur 1 dans la variable b. Ce qui fait bien qu'au début on avait 1 dans a et 2 dans b, on a échangé le contenu initial des deux variables a et b. Passons maintenant à la lecture de la valeur d'une variable depuis le clavier. Prenons notre exemple favori du calcul et de l'affichage du carré ou du double de n. Supposons qu'on veuille non pas ce programme pour n = 4 mais plutôt demander la valeur de n au clavier à l'utilisateur. Comment faire ? Pour cela, on va simplement remplacer la ligne qui affectait la valeur 4 à n par un ensemble de lignes que

notes

résumé

2m 4s

Lire une valeur au clavier

Il faut d'abord importer la classe Scanner pour la rendre visible au compilateur:

```
import java.util.Scanner;
```

On peut maintenant créer un objet `Scanner`, par exemple:

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

clavier

`keyb` est une variable qu'on peut utiliser pour demander des valeurs au clavier.
Par exemple:

```
int n = keyb.nextInt();
```

l 'on va détailler dans la suite et qui vont permettre d'affecter une valeur lue au clavier à la variable n. Il faut commencer par importer ce qui s'appelle techniquement «la classe Scanner». En ajoutant cette ligne une fois au début du programme : import java.util.Scanner; Cette ligne ajoutée, vous pouvez déclarer une variable keyb de type Scanner et de l'initialiser en la liant au clavier, ce qu'on appelle techniquement l'entrée standard, in comme input, entrée représentée par System.in. Tout ce que vous avez à faire, c'est d'écrire cette ligne : Scanner avec un nom de variable, keyb, = new Scanner (System.in) ; et à partir de là, vous pouvez utiliser la variable keyb pour lire des variables au clavier. Cette ligne doit être écrite une fois et une seule pour tout le programme. Vous n'avez besoin que de faire une fois le lien vers le clavier. À ce stade du cours, vous l'écrirez dans le main et à partir de cette ligne, vous pourrez utiliser la variable keyb pour pouvoir lire les valeurs. On peut vouloir lire un entier au clavier pour initialiser une variable comme ici : je déclare une variable n de type entier que j'initialise avec une lecture au clavier qui s'écrit de la façon suivante : keyb, la variable Scanner que j'avais précédemment déclarée

notes

résumé

3m 13s

Lire une valeur au clavier

```
int n = keyb.nextInt();
```

Cette instruction

1. Arrête le programme momentanément;
2. Attend que l'utilisateur entre une valeur au clavier et appuie sur la touche `return`;
3. Affecte la valeur entrée par l'utilisateur à la variable `n`,

puis le programme continue.

`nextInt()` est une méthode de l'objet `Scanner`. Il existe une autre méthode pour Demander une valeur de type `double`:

```
double x = keyb.nextDouble();
```

keyb point et une lecture d'un entier nextInt() pour lire le prochain entier et parenthèses et on termine l'instruction d'initialisation de notre variable par un point virgule. Tout ceci est peut-être un peu technique, mais il suffit de suivre la procédure expliquée ici en trois étapes que je vous résume encore une fois. D'abord, vous ajoutez cette ligne une fois au tout début de votre programme. Puis cette ligne, une seule fois là où vous en avez besoin, ici dans le main pour déclarer une variable de type Scanner. Et vous utilisez autant de fois que nécessaire la lecture d'un entier avec nextInt(). La méthode nextInt() fonctionne ainsi : elle commence par arrêter momentanément le programme et elle va ensuite attendre que l'utilisateur saisisse une valeur et appuie sur la touche entrée ou return puis elle lit la valeur comme un entier la convertit en entier et le résultat sera la valeur de type entier de ce qui a été tapé au clavier affectée à la variable n à l'étape n°3 puis le programme va continuer. nextInt() est ce qu'on appelle une méthode, une fonction, un traitement associé à l'objet Scanner. Il est possible de lire plein d'autres choses, plein de valeurs d'autres types. Par exemple, si l'on veut lire une valeur décimale, une valeur de type double, on va utiliser la fonction nextDouble(). Pour initialiser une variable x de type double avec une valeur lue au clavier

notes

résumé

4m 49s

Déroulement du programme pas-à-pas

EPFL

```
Scanner keyb = new Scanner(System.in);
System.out.println("Entrez une valeur pour n");
int n = keyb.nextInt();

int nCarre = n * n;

System.out.println("La variable n contient " + n);
```

Ce qui s'affiche:

on écrira la déclaration double x et l'initialisation = keyb.nextDouble(); qui lira au clavier une valeur de type double, décimale.

notes

résumé

6m 37s

Lecture d'une ligne complète

```
Scanner keyb = new Scanner(System.in);
String s = keyb.nextLine();
System.out.println("Vous avez saisi : " + s);
```

`s` est une chaine de caractères. Après l'exécution de l'instruction:

```
String s = keyb.nextLine();
```

`s` contient tous les caractères tapés par l'utilisateur, jusqu'à l'appui de *return*.

Prenons un exemple pas-à-pas : on commence avec la ligne de déclaration usuelle par créer un Scanner qui va associer le clavier au Scanner keyb. Ensuite, il faut avertir l'utilisateur qu'il doit entrer une valeur. On va lui afficher, avec la technique vue précédemment, un message : "Entrez une valeur pour n" On va lire la valeur au clavier avec la méthode nextInt() de l'objet que l'on a créé keyb Supposons qu'alors l'utilisateur tape au clavier la valeur 2. Cette méthode va s'évaluer et lire la valeur 2 au clavier. 2 va se mettre comme valeur initiale dans la variable n. À l'étape suivante, on va évaluer n*n qui va faire 4. On va mettre 4 dans la variable nCarre. Puis on va afficher le message "La variable contient ", puis évaluation de l'expression n ; ce qui nous permettra d'afficher la valeur de n.

notes

résumé

6m 47s

Attention avec nextLine() !

```
int i = keyb.nextInt();
String s1 = keyb.nextLine();
String s2 = keyb.nextLine();
```

Si on tape:

25 francs
23 francs

➔ i contient 25, s1 contient francs, s2 contient 23 francs

Si on tape:

14
euros
43

➔ i contient 14, s1 est **vide**, s2 contient euros !

car nextLine() lit ce qui suit 14 jusqu'à la fin de la ligne, c'est-à-dire rien !

En plus de nextInt() qui lit un entier et nextDouble() qui lit un nombre décimal, une fonction qui pourrait être utile est la fonction nextLine() qui permet de lire d'un seul coup toute une ligne complète. Prenons un exemple, comme d'habitude nous déclarons un Scanner, keyb associé au clavier. Une fois cette déclaration faite, on peut faire un appel, keyb.nextLine(); qui va permettre de lire toute une ligne entière. Si on a tapé un message comme "bonjour tout le monde !", l'appel à la fonction nextLine() va alors lire toute la ligne tapée par l'utilisateur, tous les caractères rentrés jusqu'à ce qu'il ait tapé sur la touche «entrée» ou «return». La question est «dans quelle variable stocker ce message ?» Quel doit être son type ? Pour l'instant, en anticipant sur ce qui sera présenté dans une vidéo ultérieure, je vous demande de recopier la ligne où String est le type de la variable s utilisée pour sauvegarder le message et s son nom. Lorsqu'on a tapé la ligne String s = keyb.nextLine(); s va contenir tous les caractères qui auront été tapés par l'utilisateur jusqu'à l'appui sur la touche return. Attention cependant avec l'utilisation de la méthode nextLine() peu compatible avec nextInt() et nextDouble() Par exemple, si je déclare une variable i de type entier et que je l'initialise avec une lecture au clavier au travers de nextInt() et que juste après je veuille lire un message, une ligne entière nextLine(), que je mets dans s1 puis je lis une deuxième ligne de la même façon, que je mets dans l'initialisation d'une variable s2. L'idée est de lire un entier, puis de lire la fin de la ligne et une autre ligne. Si on tape le message 25 francs - retour à la ligne - suivi de 23 francs, ce qui va se passer, c'est

notes

résumé

8m 13s

Attention avec `nextLine()` !

```
int i = keyb.nextInt();
String s1 = keyb.nextLine();
String s2 = keyb.nextLine();
```

Si on tape:

```
25 francs
23 francs
```

➔ `i` contient 25, `s1` contient `francs`, `s2` contient `23 francs`

Si on tape:

```
14
euros
43
```

➔ `i` contient 14, `s1` est **vide**, `s2` contient `euros` !

car `nextLine()` lit ce qui suit 14 jusqu'à la fin de la ligne, c'est-à-dire rien !

qu'on va d'abord lire un entier, donc i contiendra 25, puis ensuite on va lire la fin de la ligne, ici jusqu'à francs donc s1 contiendra effectivement francs Puis enfin, on lira toute une ligne, donc on lira 23 Francs Jusque là, pas de problème. Si on tape : 14 ; euros ; 43 ; que se passe-t-il ? Regardons pas à pas. On demande de lire un entier : i contient 14 puis de lire la fin de la ligne : ici elle est vide. s1 sera vide et on lira la ligne suivante, euros ira dans s2. Donc s2 contiendra «euros». Ce qu'on voit ici, c'est que la combinaison de nextInt() suivi de nextLine() ne fonctionne pas toujours très bien.

notes

résumé

En particulier parce que l'utilisateur a tapé un retour à la ligne derrière le nombre entier 14 et la lecture de nextLine() après nextInt() tapé au clavier va aller lire la fin de la ligne qui correspond au retour à la ligne, à la touche return, qu'a tapé l'utilisateur, parce que nextInt() arrête sa lecture juste après le nombre entier, ici 14 et ne lit pas lui-même le caractère de retour, la touche return. Comment faire en pratique ? A chaque fois que vous devez lire un nombre avec nextInt() ou nextDouble() et que la lecture suivante est une ligne il faut anticiper la non lecture du retour à la ligne par nextInt() ou nextDouble() en rajoutant une lecture de cette fin de ligne un nextLine() supplémentaire inséré entre la lecture d'un entier ou d'un nombre décimal et celle du message suivant que vous voulez vraiment. Vous rajoutez une lecture dans une variable déclarée au préalable String s = keyb.nextLine(); On pourra bien sûr, plus tard, lorsque l'on connaîtra les structures de contrôle et surtout les branchements conditionnels ou les boucles conditionnelles, on pourra vérifier que cette lecture de s est bien celle d'une chaîne vide et sinon, la prendre comme message à lire ou si oui, prendre la lecture suivante comme le message que l'on désire. comme le message que l'on désire.

notes

résumé

11m 25s