Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-06-5-Definitions-pt4

Concepts (extraits des sous-titres générés automatiquement) :

Cas particuliers de méthodes. Reste du programme. Moment précis. Return n. Niveau des déclarations des variables. Méthode main. Boucle do while. Cas de n inférieur. Type particulier de retour. Corps de la méthode. Code suivant. Racine d'un paramètre de type. Seule fois. Variable n. Instruction return.

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

vers la vidéo

EPFL

Fonctions : définitions
(Partie 4)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Remarques sur l'instruction `return` (4/4)

Le compilateur doit être sûr de toujours pouvoir exécuter un `return`:

```
static double lire() {
   System.out.print("Entrez un nombre: ");
   Scanner keyb = new Scanner(System.in);
   double n = keyb.nextDouble();
   if (n > 0){
      return n; // Erreur
   }
}
```

Plus précisément, puisque l'on va commencer par demander la valeur

notes

résumé

0m 1s

Remarques sur l'instruction `return` (4/4)

Le compilateur doit être sûr de toujours pouvoir exécuter un `return`:

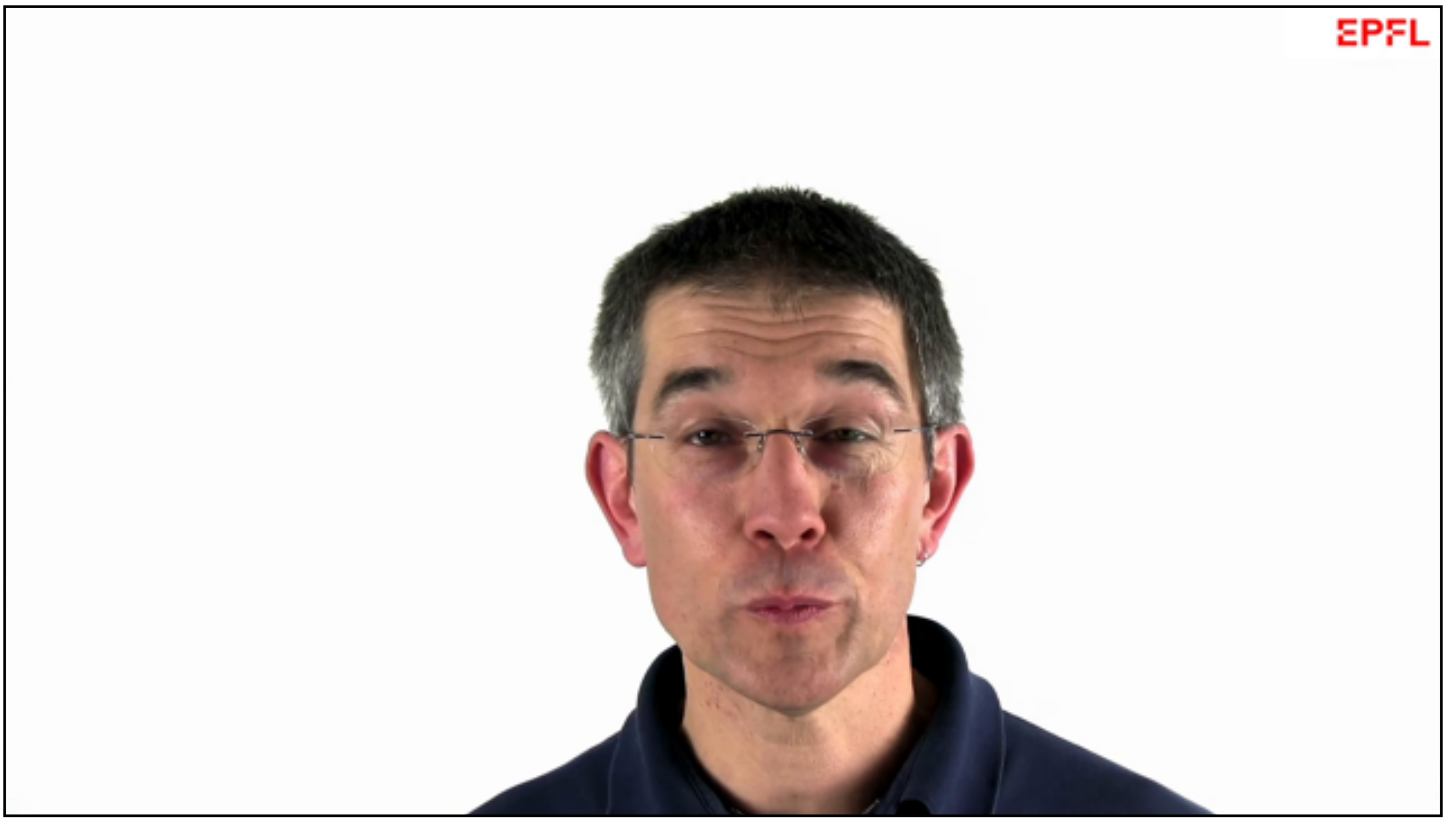
```
static double lire() {
  Scanner keyb = new Scanner(System.in);
  double n = 0.0;
  do {
      System.out.print("Entrez un nombre strictement positif : ");
      n = keyb.nextDouble();
  } while (n <= 0.0);
  return n;
}
```

à l'utilisateur avant de pouvoir tester si cette valeur est positive ou nulle, il s'agira concrètement d'une boucle do while, donc on va pouvoir écrire ici la boucle do while, ensuite on va s'intéresser à la condition, donc la condition était ici de retourner, de s'arrêter quand on a un nombre strictement positif, ce qui fait que l'on va donc vouloir boucler, répéter la question tant que le nombre est cette fois-ci bien sûr négatif ou nul, on répète tant que l'on n'a pas atteint la condition que l'on voulait pour faire notre return. Okay, donc on va ici rajouter le return n et tant que n n'est pas celui qu'on veut, tant que n est négatif ou nul, on va répéter, boucler sur donc ici bien sûr d'abord afficher le message, donc cette ligne, puis ensuite lire la réponse, donc on va mettre cette ligne ici. Au niveau des déclarations des variables, il faudra déclarer le scanner et déclarer n, toutes ces lignes-là vont se déplacer hors de la boucle puisque le scanner on n'a pas besoin de le répéter dans la boucle et n on le déclare aussi qu'une seule fois. Donc plus concrètement ça donne le code suivant : on va commencer par déclarer le scanner une fois pour toutes pour la lecture sur clavier, ensuite on déclare la variable n, il faut la déclarer hors de la boucle puisque on va l'utiliser ici pour la retourner, on l'utilise ici dans le test qui est hors de la boucle en question, donc on déclare la variable on l'initialise à une valeur qui fait sens, disons par exemple zéro, puis on va ensuite écrire le bloc ici de

notes

résumé

0m 6s



répétition, le bloc sur lequel on veut boucler, dans lequel on va afficher le message qui demande à l'utilisateur d'entrer un nombre et là on peut même lui spécifier exactement nos intentions pour qu'il sache quoi faire et dire que l'on entrer un nombre strictement positif, puis on lit ce nombre dans n depuis le clavier et donc on boucle tant que l'on n'a pas la bonne réponse, tant que n est négatif. Voilà donc comment on aurait pu corriger l'erreur du return qui manquait en cas de n inférieur ou égal à zéro.

notes

résumé

2m 1s

Méthodes sans paramètre

Il est aussi possible de définir des méthodes **sans paramètre**.
Il suffit, dans l'entête, d'utiliser une liste de paramètres vide : ()

Exemple :

```
private static Scanner clavier = new Scanner(System.in);

public static void main(String[] args)
{
  int val = saisieEntier();
  System.out.println(val);
}

static int saisieEntier()
{
  int i;
  System.out.println("entrez un entier: ");
  i = clavier.nextInt();
  return i;
}
```

Présentons maintenant quelques cas particuliers de méthodes, en commançant par les méthodes sans valeur de retour. Je dois d'abord vous rappeler qu'une méthode au sens java du terme n'a rien à voir avec une fonction mathématique, une méthode en Java c'est simplement un bout de code que l'on a réservé pour pouvoir l'utiliser, éviter la répétition, et en ce sens, ce bout de code n'a pas forcément besoin de retourner quelque chose au reste du programme. Si c'est le cas, alors on va spécifier le fait que la méthode ne retourne rien en utilisant comme type particulier de retour, le type void pour indiquer que l'on ne retourne rien au reste du programme. Dans ce cas-là, l'instruction return est alors optionnelle, on peut tout à fait ne placer aucun return dans le corps de la méthode, ou alors si vraiment on en a besoin, par exemple dans un if, d'arrêter à un moment précis l'exécution du corps, alors on pourra à ce moment-là utiliser l'instruction return. Prenons un exemple, avec une méthode que je vais ici appeler afficheRacine et dont le but est d'afficher la racine d'un paramètre de type double reçu ici et que j'appelle a. Si a est négatif, alors vous savez qu'on ne peut pas calculer la racine et donc il n'y a rien à afficher, donc dans ce cas-là, si la condition a négative est vraie, le if ici va se brancher sur cette instruction return, il n'y a que l'instruction return ici qui va donc mettre fin à l'exécution de cette méthode. Notez que comme le type de retour ici de la méthode afficheRacine est void, ça veut dire que afficheRacine n'a rien à renvoyer au reste du programme et vous voyez ici que le return n'a aucune expression qui le suit, ça n'aurait pas de sens, on ne sait pas quoi mettre derrière ce return. Si par contre, a est positif ou nul, à

notes

résumé

2m 30s

Méthodes sans paramètre

EPFL

Il est aussi possible de définir des méthodes **sans paramètre**.
Il suffit, dans l'entête, d'utiliser une liste de paramètres vide : ()

Exemple :

```
private static Scanner clavier = new Scanner(System.in);

public static void main(String[] args)
{
  int val = saisieEntier();
  System.out.println(val);
}

static int saisieEntier()
{
  int i;
  System.out.println("entrez un entier: ");
  i = clavier.nextInt();
  return i;
}
```

ce moment-là la condition du if est fausse bien entendu et à ce moment-là donc le bloc if est simplement ignoré et le programme va continuer après le if par exécuter l'affichage de la racine carrée de a, et on arrive donc ensuite à la fin de la méthode où on n'a pas mis de return, la méthode va se terminer, il n'est ici pas du tout nécessaire de mettre un return, un tel return est totalement optionnel. Autre cas particulier de méthodes, les méthodes sans paramètres. Nous en avons déjà un petit peu parlé mais je voudrais juste ici récapituler, ce sont des méthodes dont on n'a pas besoin de fournir depuis le reste du programme de valeurs pour qu'elles puissent fonctionner. Alors à ce moment-là, il suffit simplement dans l'entête d'indiquer la liste de paramètres vide par simplement une parenthèse ouvrante et une parenthèse fermante collées sans rien au milieu. Prenons ici un exemple d'une méthode dont le seul but est de demander un entier à l'utilisateur et donc qui doit retourner un entier au reste du programme. Un exemple d'appel pourrait être le suivant : ici on a une variable de type entier que l'on initialise avec le résultat de l'appel à saisieEntier, et cette méthode saisieEntier n'a rien besoin de recevoir du reste du programme, elle peut travailler simplement de façon autonome en déclarant ici localement pour elle-même une variable dans laquelle elle va stocker la réponse de l'utilisateur en affichant la question entrer un entier à l'utilisateur, en lisant l'entier lu par l'utilisateur au clavier et en le mettant dans la variable temporaire, locale, ici i, et en retournant donc au reste du programme la variable i. Vous voyez bien que pour faire tout ceci, elle n'a pas eu besoin de recevoir d'informations du reste du programme, c'est une méthode sans paramètres. Une aparté pratique ici, le scanner jusqu'à maintenant, nous en avions

notes

résumé

Méthodes sans paramètre

EPFL

Il est aussi possible de définir des méthodes **sans paramètre**.
Il suffit, dans l'entête, d'utiliser une liste de paramètres vide : ()

Exemple :

```
private static Scanner clavier = new Scanner(System.in);

public static void main(String[] args)
{
  int val = saisieEntier();
  System.out.println(val);
}

static int saisieEntier()
{
  int i;
  System.out.println("entrez un entier: ");
  i = clavier.nextInt();
  return i;
}
```

besoin que dans la méthode main, nous n'avions qu'une seule méthode jusque là et nous utilisions le scanner dans la méthode main, c'est pour ça que nous le déclarions dans cette méthode. Ici nous en avons besoin dans une autre méthode, la méthode saisieEntier, alors bien sûr on pourrait avoir une déclaration comme ça d'un scanner dans la méthode saisieEntier, mais c'est pas une bonne chose en soi parce que chaque fois que vous allez appeler saisieEntier, l'idée de la fonction c'est quand même d'être appelée plusieurs fois, chaque fois que vous allez comme ça faire un appel à saisieEntier, si cette ligne de déclaration du scanner est dedans vous allez à chaque fois redéclarer un clavier, or on a qu'un seul clavier, d'où l'idée c'est de partager ce clavier pour tout le programme et donc de mettre cette ligne hors dans la classe, hors de toute méthode ici, donc on retrouve la déclaration visuelle ici, static scanner clavier etc avec le lien ici au clavier System.in mais on rajoutera ici pour des raisons qui seront exposées dans le cours programmation orientée objet, le mot-clé private dedans pour

notes

résumé

La méthode `main`

`main` est aussi une méthode avec un nom et un entête imposés.

Par convention, tout programme Java doit avoir une méthode `main`, qui est appelée automatiquement quand on exécute le programme.

L'entête autorisée pour `main` est :

```
public static void main(String[] args)
```

les bonnes pratiques. En l'état, si vous avez besoin du scanner plus largement que dans la méthode main, je vous encourage donc à suivre cette syntaxe ici.

notes

résumé

7m 25s

Pour terminer, je voudrais parler de la méthode main, c'est aussi une méthode, c'est simplement une méthode dont le nom et l'entête ont été imposé, c'est la méthode avec laquelle tout programme Java va démarrer, c'est la première méthode qui sera exécutée lorsqu'on démarre le programme. L'entête imposée pour main est le suivant, où sans trop entrer dans les détails on voit ici que main reçoit sous forme d'un tableau de chaîne de caractères, une liste d'arguments qui viendraient de l'extérieur, qui viendraient du programme qui appelle un autre programme mais ceci nous emmènerait un petit peu trop loin, nous ne l'utiliserons pas dans ce cours, tout ce qu'il faut savoir c'est que le prototype de main doit être celui indiqué ici. indiqué ici.

notes

résumé

7m 37s