Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-06-6-Methodologie

Concepts (extraits des sous-titres générés automatiquement) :

Première étape. Type de retour. Deuxième étape. Premier aspect. Nom de la méthode. Méthode. Valeur de retour. Troisième étape. En-têtes des méthodes. Méthode du reste du programme. Niveau de ce cours. Tel appel. Exemple de la méthode racine carrée. Exemple concret. Type de retour de la méthode.

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

vers la vidéo

EPFL

Fonctions : méthodologie

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Pour résumer : Méthodologie pour construire une méthode

1. clairement identifier ce que **doit faire** la méthode
   (ce point n'est en fait que conceptuel, on n'écrit aucun code ici !)
   - ☞ ne pas se préoccuper ici du ~~*comment*~~, mais bel et bien du **quoi** !

     Les instructions dans le corps de la méthode dont la finalité n'est pas le calcul de la valeur de retour, ou qui modifient des objets extérieurs à la méthode (non passés en paramètre), sont appelées des « **effets de bord** ».

Résumons maintenant la méthodologie que nous vous conseillons de suivre lorsque vous souhaitez créer une méthode. Le tout premier aspect est de se focaliser d'abord sur ce que doit faire la méthode, c'est à dire sur le "quoi" et non pas sur "comment" faire ce que doit faire la méthode. Donc bien se préoccuper sur qu'est-ce que doit faire la méthode, et en particulier éviter que la méthode fasse des effets qui ne sont pas désirés, ce que l'on appelle des "effets de bord", comme nous avions vu dans la vidéo sur les en-têtes des méthodes

notes

résumé

0m 1s

Pour résumer : Méthodologie pour construire une méthode

1. clairement identifier ce que **doit faire** la méthode
2. quels **arguments** ?
   ☞ que doit recevoir la méthode pour faire ce qu'elle doit ?

avec l'exemple de la méthode racine carrée sqrt. La première étape est donc de se focaliser sur le "quoi" plutôt que de mettre en avant le "comment".

notes

résumé

0m 37s

Pour résumer : Méthodologie pour construire une méthode

EPFL

1. clairement identifier ce que **doit faire** la méthode
2. quels **arguments** ?
3. quel type de retour ?
4. (maintenant, et seulement maintenant) Se préoccuper du *comment* :
   comment faire ce que doit faire la méthode ?
   ☞ c'est-à-dire écrire le corps de la méthode

Une fois que ceci est clair, vous pouvez alors passer à la deuxième étape qui est : qu'est-ce que doit recevoir la méthode du reste du programme pour pouvoir fonctionner ? Pour pouvoir faire ce qu'elle a à faire. Par exemple si la méthode s'appelle f et que je fais un appel ici, qu'est-ce que le reste du programme doit passer à cette méthode pour pouvoir fonctionner ? Troisième étape : vous demandez quel est le type de retour. Qu'est-ce que retourne la méthode au reste du programme ? Et pour ça posez-vous la question : est-ce que ça a un sens d'écrire un appel comme ceci z = f, f étant le nom de la méthode, est-ce que ça a un sens d'écrire z = f de quelque chose ? Si ça a un sens, alors le type de retour sera le type de z. Par exemple, écrire z = sqrt(2.0) a un sens, et donc le type de retour de la méthode sqrt doit être le même que le type de z. Dans cet appel, ça serait donc typiquement un double. Si ça n'a pas de sens d'écrire z = appel de méthode, alors la méthode ne doit rien retourner, et donc son type de retour, entre guillemets, serait le type void. Ensuite quatrième étape :

notes

résumé

0m 47s

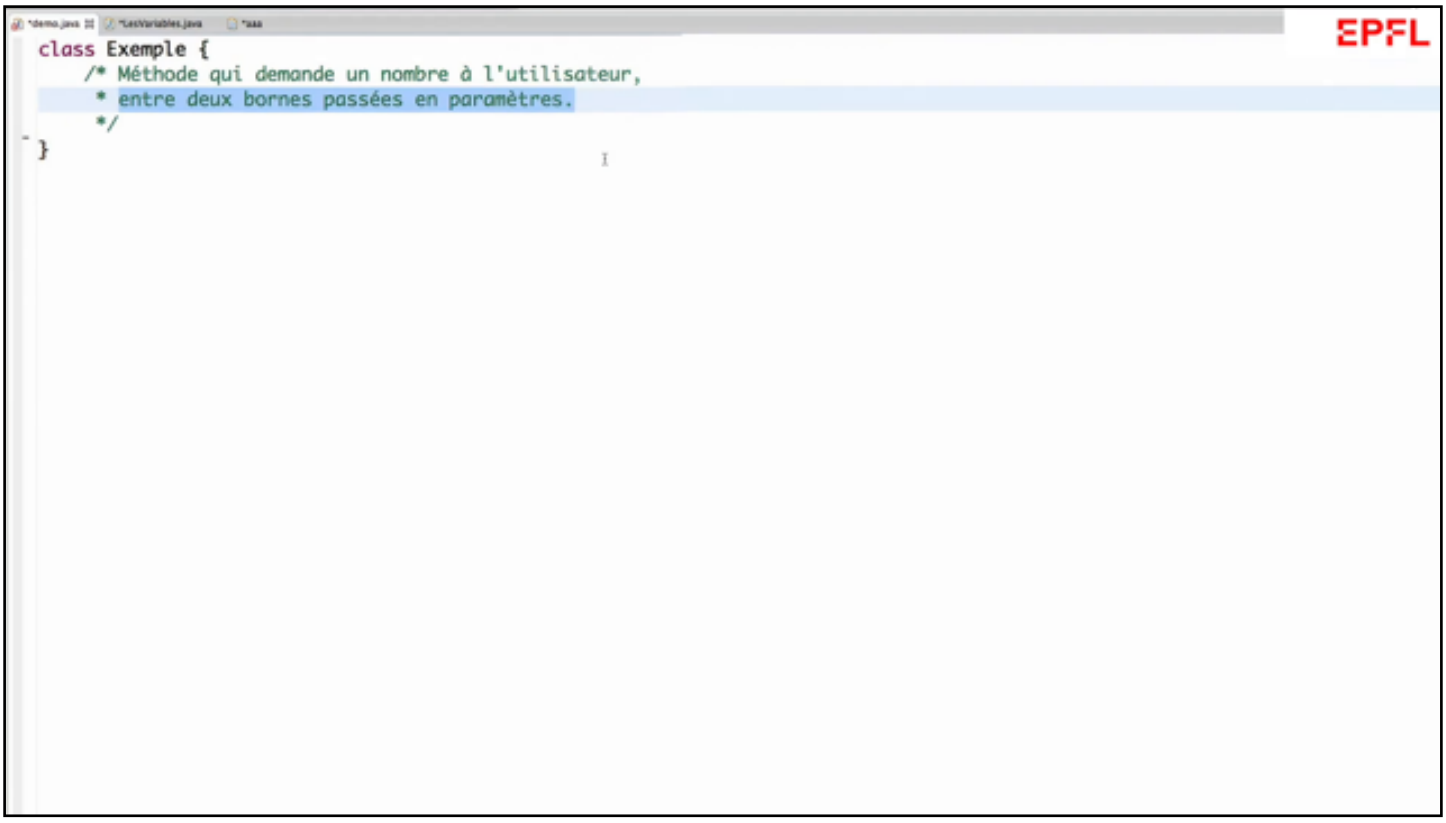
```
class Exemple {

}
```

maintenant et seulement maintenant, vous devez vous préoccuper du "comment". Comment j'écris le code ? Comment je construis cette valeur de retour que la méthode doit retourner au reste du programme ?

notes

résumé

2m 1s



Prenons maintenant un exemple concret. Supposons que l'on veuille par exemple créer une méthode qui demande un nombre à l'utilisateur, et par exemple que ce nombre soit compris entre deux bornes. Donc on demanderait à l'utilisateur d'entrer un nombre entre 1 et 10. En suivant la méthodologie, on commence donc par bien se poser la question de ce que l'on veut faire, et à ce stade-là donc bien concevoir la méthode, et là je vous conseille par exemple de commencer par taper des commentaires qui indiqueraient clairement ce que ce que vous voulez faire.

notes

résumé

2m 16s

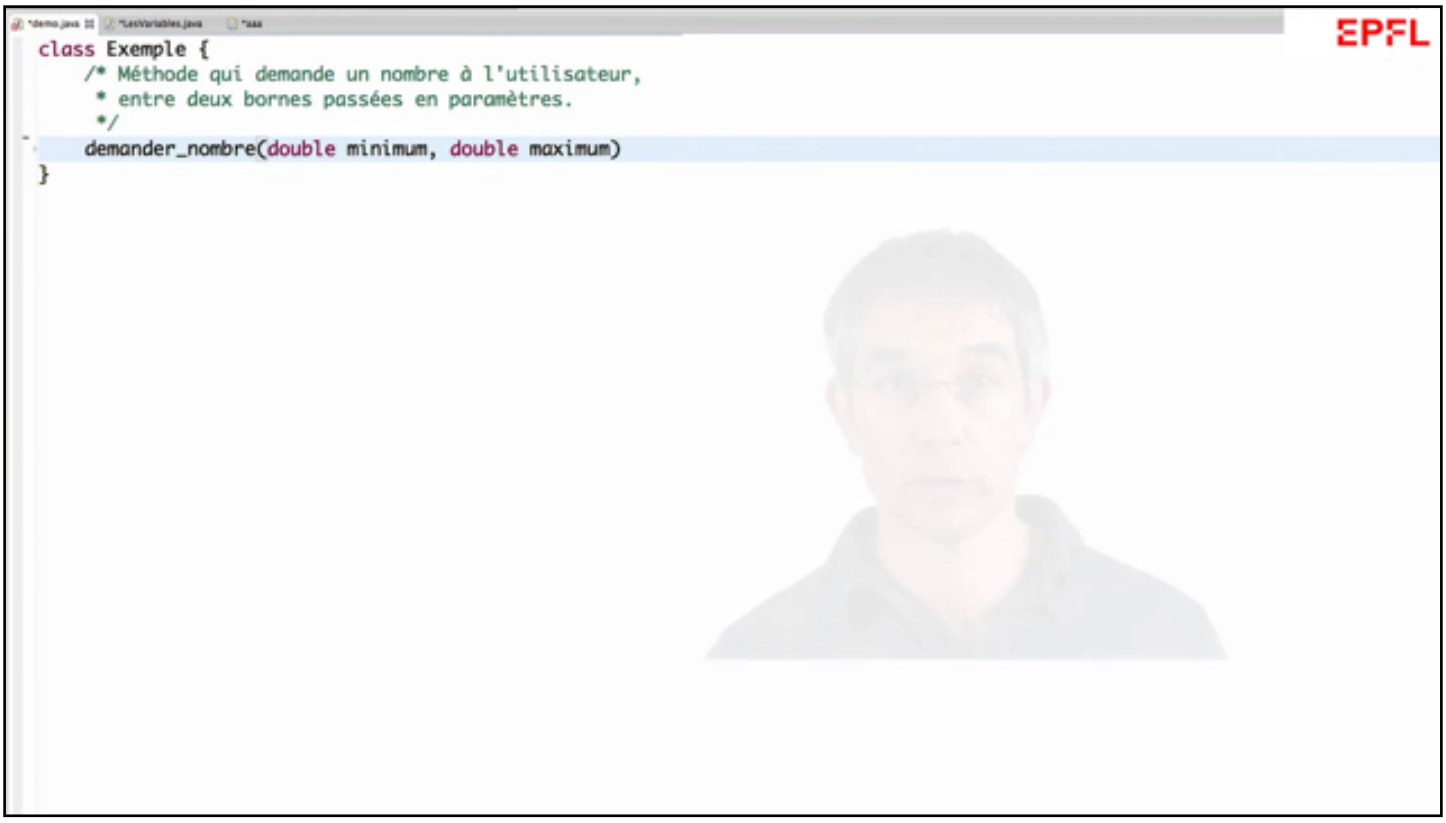
```java
class Exemple {
    /* Méthode qui demande un nombre à l'utilisateur,
     * entre deux bornes passées en paramètres.
     */
    demander_nombre()
}
```

Donc par exemple ici, clairement indiquer que c'est une méthode qui demande un nombre à l'utilisateur et que ce nombre doit être compris entre 2 bornes que l'on passera en paramètres. A ce stade-là on peut aussi commencer par fixer le nom, choisir le nom, donc par exemple le nom de la méthode pourrait être demander_nombre. Cette étape qui consiste à clairement spécifier ce que doit faire la méthode, qui s'intéresse donc au "quoi", ce que doit faire la méthode et non pas au "comment", est essentiellement conceptuelle, mais elle est fondamentale pour le bon développement de la méthode. Et donc elle n'est pas à négliger. On peut alors passer à la deuxième étape qui consiste à se demander quels paramètres doit recevoir la méthode en question.

notes

résumé

2m 49s



Donc ici on a décidé qu'elle recevrait deux paramètres : un paramètre qui indiquera la borne inférieure, donc par exemple on va l'appeler minimum ; et puis un paramètre qui reçoit la borne supérieure, donc par exemple qu'on peut appeler maximum. Dans notre exemple où nous voulions demander un nombre à l'utilisateur compris entre 1 et 10, on passerait donc la valeur "1" comme premier argument, et la valeur 10 comme deuxième argument. Une fois qu'on a terminé avec cette deuxième étape, qui s'intéressait aux paramètres, on passe à la troisième étape, qui se demande quel doit être le type de la valeur de retour de la méthode.

notes

résumé

3m 37s

```java
class Exemple {
    /* Méthode qui demande un nombre à l'utilisateur,
     * entre deux bornes passées en paramètres.
     */
    double demander_nombre(double minimum, double maximum)

            z = demander_nombre(1.0, 10.0); ???
}
```

Qu'est-ce que la méthode doit renvoyer au reste du programme ? Et pour cela je vous conseille donc d'écrire une affectation du genre : z = "appel de la méthode" donc ici z = demander_nombre dans le cas précédent, par exemple, on va mettre entre les bornes 1 et 10. Et se demander si un tel appel a un sens. Ici pour nous clairement un tel appel a un sens et la valeur ici stockée dans z, serait celle que l'utilisateur nous aurait fournie. Et donc quand un tel appel a un sens, à ce moment-là le type de retour de la méthode ici, est le type de z dans une telle expression. Donc ici on s'attend à avoir un double. Donc le type de retour est double.

notes

résumé

4m 13s

```java
class Exemple {
    /* Méthode qui demande un nombre à l'utilisateur,
     * entre deux bornes passées en paramètres.
     */
    static double demander_nombre(double minimum, double maximum)

}
```

Au niveau de ce cours on n'oubliera pas de rajouter le mot clé static. Une fois qu'on a terminé avec cette troisième étape, qui s'intéressait au type de retour de la méthode, maintenant et seulement maintenant on va s'intéresser au "comment". Comment la méthode doit travailler, comment elle doit faire ce qu'elle a à faire ?

notes

résumé

5m 1s

C'est à dire uniquement maintenant, on va se préoccuper du corps de la méthode et donc ajouter ici un bloc, c'est à dire deux accolades, pour pouvoir y mettre ce que la méthode doit faire ici, pour demander un nombre à l'utilisateur. pour demander un nombre à l'utilisateur.

notes

résumé

5m 20s