

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W15-02-methodstat-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Méthodes statiques. Méthodes de classe. Attributs statiques. Instance concrète de la classe. Méthode de classe. Méthode d'instance. Méthode usuelle. Méthode statique. Mot clé static. Nom de la méthode statique. Telle méthode. Exemple d'utilisation de cette méthode. Classe a. Nom de la classe. Attribut usuel.



[vers la recherche de séquences vidéo](#)

(dans Introduction à la programmation orientée objet (en Java).)



[vers la vidéo](#)

Center for Digital Education. Plus de matériel de soutien pédagogique ici :

<https://www.epfl.ch/education/educational-initiatives/cede/educational-technologies-gallery/boocs-en/>

Méthodes statiques

(Partie 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



Similairement, si l'on ajoute `static` à une méthode on peut alors y accéder *sans aucun objet*

```
class A {
    static void methode1() {
        System.out.println("Méthode 1");
    }
    void methode2() {
        System.out.println("Méthode 2");
    }
}
class ExempleMethodeStatique {
    public static void main(String[] args) {
        A.methode1(); // OK
        A.methode2(); // Non !
        A v = new A();
        v.methode1(); // OK, alternative
        v.methode2(); // OK (comme d'habitude)
    }
}
```

Après les attributs statiques, nous allons dans cette séquence vidéo nous intéresser aux méthodes statiques, méthodes de classe. De la même façon que pour les attributs, si on ajoute le mot clé `static` devant une méthode, alors cette méthode devient une méthode de classe, c'est-à-dire une méthode que l'on peut appeler sans aucun objet, sans aucune instance concrète de la classe en question. Donc prenons un exemple : nous avons ici une classe A dans laquelle est définie une méthode, « `methode1` », qui est une méthode de classe, une méthode statique. Et puis nous avons une méthode usuelle, méthode d'instance ici, qui s'appelle « `methode2` ». Dans l'exemple d'utilisation de cette méthode de classe, nous voyons ici que nous pouvons tout à fait appeler la « `methode1` », méthode de classe, sans aucune instance à ce stade là de la classe A. La syntaxe est simplement : « nom de la classe, point ('.'), nom de la méthode statique/ nom de la méthode de classe ». Ceci est tout à fait possible, ça va appeler la « `methode1` » et afficher le message « Méthode 1 ». Bien sûr pour une méthode usuelle, méthode d'instance qui n'est pas déclarée avec le mot clé « `static` », on n'a pas le droit d'utiliser cette syntaxe ; ça n'a pas de sens ! « `methode2` » doit être attachée à une instance. On crée donc ici, pour continuer l'exemple une instance `v` de la classe A. Alors on peut bien sûr appeler la méthode de classe avec la syntaxe « `v.methode1` ». C'est exactement la même chose qu'avoir écrit « `A.methode1` ». C'est juste une forme alternative. Je vous recommande d'ailleurs plutôt d'utiliser le nom de la classe. le nom de la méthode pour des méthodes de classe, pour bien montrer l'intention et rappeler que c'est une méthode de classe, même si la

notes

résumé

0m 1s



Similairement, si l'on ajoute `static` à une méthode on peut alors y accéder *sans aucun* objet

```
class A {
    static void methode1() {
        System.out.println("Méthode 1");
    }
    void methode2() {
        System.out.println("Méthode 2");
    }
}
class ExempleMethodeStatique {
    public static void main(String[] args) {
        A.methode1(); // OK
        A.methode2(); // Non !
        A v = new A();
        v.methode1(); // OK, alternative
        v.methode2(); // OK (comme d'habitude)
    }
}
```

notation méthode d'instance est possible,

notes

résumé

Puisqu'une méthode statique peut être appelée avec ou sans objet :

- ▶ Le compilateur ne peut pas être sûr que l'objet `this` existe pendant l'exécution de la méthode
- ▶ Il ne peut donc pas admettre l'accès aux variables/méthodes d'instance (car elles dépendent de `this`)

Conclusion pour les accès dans la même classe :

- ▶ Une méthode statique peut *seulement* accéder à d'autres méthodes statiques et à des variables statiques



elle est à mon avis un peu plus ambiguë, et je trouve celle-ci beaucoup plus claire. Et puis évidemment comme d'habitude, on peut appeler la méthode2, qui une méthode d'instance sur l'instance v. Puisqu'une méthode statique peut être appelée sans instance, sans objet, alors une telle méthode ne peut pas s'appuyer sur l'existence d'un tel objet, et ne peut donc pas appeler des variables d'instance ni des méthodes d'instance. La référence « `this` » n'existe pas dans une telle méthode puisque on ne peut pas garantir l'existence d'un objet. Donc une telle méthode, une méthode statique ne peut appeler que des autres méthodes statiques

notes

résumé

1m 49s



```
class A {
    int i;
    static int j;
    void methode1() {
        System.out.println(i); // OK
        System.out.println(j); // OK
        methode2();           // OK
    }
    static void methode2() {
        System.out.println(i); // Faux
        System.out.println(j); // OK
        methode1();           // Faux
        methode2();           // OK (sauf recursion infinie)
        A v = new A();
        v.methode1();          // OK
    }
}
```

et ne peut avoir accès qu'à des membres de classe, qu'à des attributs statiques. Illustrons ceci sur un exemple, nous avons donc ici une classe A, qui contient donc un attribut usuel, non statique i, et puis un attribut de classe, un attribut statique j, une méthode usuelle, non statique, méthode d'instance, « methode1 », et une méthode statique, « methode2 », méthode de classe. Dans la méthode usuelle, « methode1 », on essaie d'accéder à l'attribut i, on n'a aucun souci. Une méthode liée ici à une instance peut accéder aussi bien aux attributs d'instance, que donc ici accéder à l'attribut de classe j. On peut accéder aux deux. On a le droit bien sûr dans la « methode1 » d'appeler la « methode2 », qui est une méthode de classe. Donc tout est permis bien sûr dans une méthode d'instance. Par contre dans une méthode de classe, on ne peut accéder qu'aux attributs et aux méthodes de classe. Si ici on essaie de faire la même chose que dans la « methode1 » et d'accéder à l'attribut d'instance i, alors à ce moment là, on va avoir une erreur de compilation, on n'a pas le droit de faire ceci puisqu'on est dans une méthode de classe. On ne peut pas garantir l'existence d'un attribut, d'un objet de la classe A. Par contre, on peut tout à fait accéder à l'attribut statique j, j est un attribut de classe, attribut statique. On a le droit de faire ceci. On n'a pas le droit non plus dans la « methode2 », méthode statique, d'appeler une méthode non statique, la « methode1 », donc ceci générerait aussi une erreur du compilateur. Par contre on peut tout à fait appeler des méthodes statiques, y compris elles-mêmes. Cet appel est syntaxiquement correct, il n'y a pas de soucis. Alors évidemment

notes

résumé

2m 25s

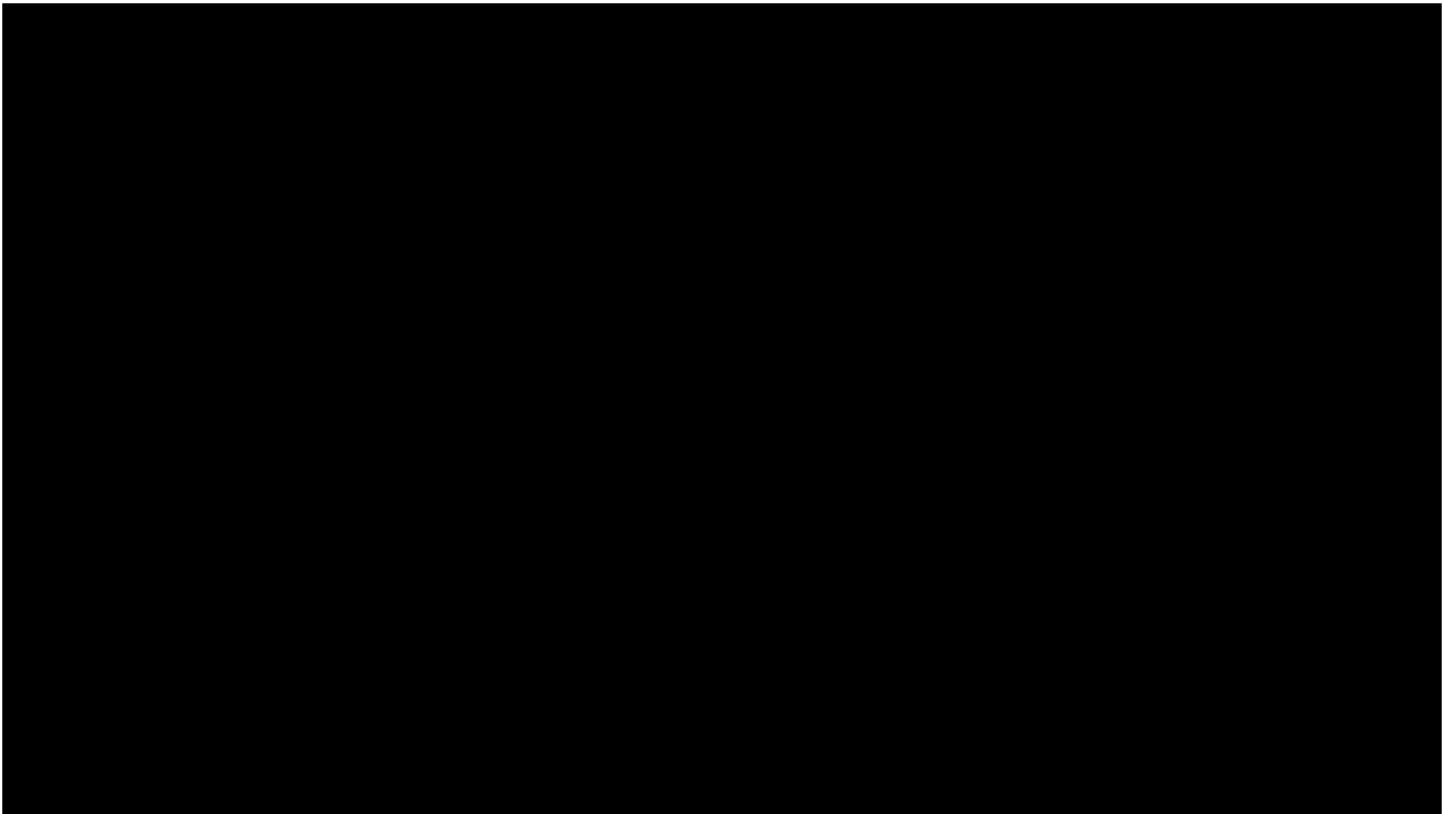


```
class A {  
    int i;  
    static int j;  
    void methode1() {  
        System.out.println(i); // OK  
        System.out.println(j); // OK  
        methode2();           // OK  
    }  
    static void methode2() {  
        System.out.println(i); // Faux  
        System.out.println(j); // OK  
        methode1();           // Faux  
        methode2();           // OK (sauf recursion infinie)  
        A v = new A();  
        v.methode1();          // OK  
    }  
}
```

ici, comme on a appelé la même méthode,

notes

résumé



il faudrait juste faire attention et prendre des précautions pour que ça ne donne pas une récursion infinie, et protéger cet appel par une condition de terminaison. Mais ça, ça nous emmène un peu trop loin on voulait juste signaler ici, que nous pouvons appeler dans une méthode statique, une autre méthode statique. Evidemment si on crée une instance comme ceci dans une méthode statique, on peut tout à fait créer une instance comme ça dans une méthode statique, alors bien sûr on peut appeler ici la méthode1 sur l'instance v de la classe A. de la classe A.

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

4m 13s



.....

.....

.....

.....

.....