



Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W14-03-final-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Méthodes final. Méthode final. Séquence vidéo. Personnages possibles du jeu. Raison du fait. Message d'erreur du compilateur. Cadre de l'héritage. Super-classe. Méthodes. Contexte nouveau. Hiérarchie d'une méthode. Classes. Grandes lignes. Durée de vie du personnage. Déclaration de la classe.



[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/>

Le modificateur `final` (Partie 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



Ce modificateur permet d'indiquer les éléments du programme qui ne **doivent pas être modifiés/redéfinis/étendus**

- ▶ Possible pour les classes, méthodes, attributs, variables
- ▶ Utile surtout pour les variables
- ▶ Moins courant pour les méthodes et les classes

Cette séquence vidéo a pour but de présenter quelques compléments sur le modificateur « `final` », notamment dans le cadre de l'héritage. Le modificateur « `final` » peut s'appliquer à des variables pour indiquer qu'elles ne doivent pas être modifiées, peut s'appliquer à des classes pour indiquer qu'elles ne peuvent pas être étendues et peut être appliqué à des méthodes pour indiquer qu'elles ne peuvent pas être redéfinies dans le cadre d'une relation d'héritage. Il est très utile pour les variables et moins courant pour les méthodes et les classes. Le modificateur « `final` » a déjà été présenté pour les variables dans les grandes lignes

notes

résumé

0m 1s



Si l'on ajoute `final` à une méthode :

- Impossible de la redéfinir dans une sous-classe

Exemple : on aimerait toujours appliquer la méthode `vieillir` de `Personnage`

```
class Personnage
{
    //...
    final void vieillir() {
        --dureeVie;
    }
}
```



- message d'erreur du compilateur si la classe `Sorcier` essaie de redéfinir la méthode `vieillir`

dans le cadre de notre MOOC précédant sur l'introduction à la programmation en Java. Nous allons donc, dans ce qui suit, commencer par son application dans un contexte nouveau, à savoir l'application de « final » aux méthodes et aux classes. Commençons par les « méthodes final ». Une « méthode final » est une méthode déclarée comme telle au moment de sa définition dans la classe. Donc, il suffit de lui imposer le modificateur « final ». Ce que cela signifie est que la méthode en question ne peut pas être redéfinie dans une sous-classe. Reprenons l'exemple avec notre hiérarchie de « Personnage », supposons que l'on souhaite doter notre hiérarchie d'une méthode vieillir et que l'on souhaite faire en sorte que cette méthode soit toujours la même pour tous les personnages possibles du jeu. Par exemple, la méthode vieillir consisterait à décrémenter d'une unité à chaque fois la durée de vie du personnage. En définissant comme « final » la méthode vieillir dans la super-classe « Personnage », on empêche toute sous-classe de « Personnage » comme par exemple la sous-classe « Sorcier » de redéfinir cette méthode de façon spécifique. Si nous essayons par exemple de faire vieillir notre « Sorcier » un peu plus vite que les autres personnages, en redéfinissant la méthode vieillir dans cette sous-classe,

notes

résumé

0m 37s



Si l'on ajoute `final` à une classe :

- Impossible d'étendre la classe par une sous-classe

Exemple : on aimerait que la classe `Sorcier` n'ait jamais de sous-classe

```
final class Sorcier extends Magicien {  
    //...  
}
```

```
class MageNoir extends Sorcier { ..}  
// illicite!!
```

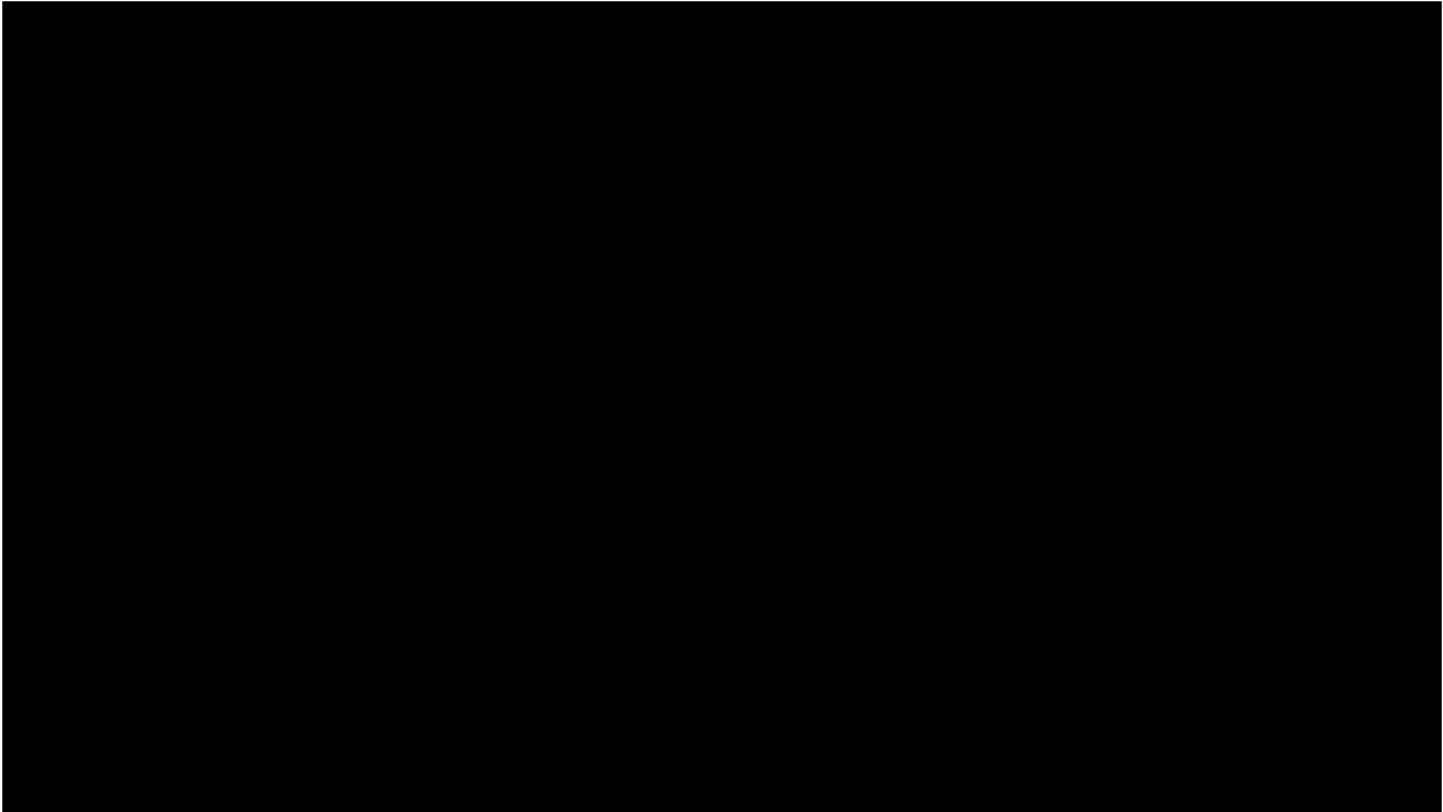
et bien, on aura un message d'erreur du compilateur nous indiquant que ceci n'est pas licite en raison du fait que la méthode vieillir est définie comme « final » dans la super-classe. Le modificateur « final » peut aussi s'appliquer aux classes. Dans ce cas, il indique qu'il est impossible de définir une sous-classe à cette classe, donc impossible d'étendre la classe par héritage. Déclarer une classe comme « final » se fait en mettant le mot réservé « final » devant la déclaration de la classe qui pour le reste se rédige comme nous avons l'habitude de le faire. Dans le cadre de notre exemple, si l'on souhaite pour une raison ou une autre que la classe « Sorcier » n'ait jamais de sous-classe, alors il suffit de la

notes

résumé

1m 49s





déclarer comme « final », ce qui signifie que si on tente par la suite de créer une classe qui étend la classe « Sorcier », donc qui hérite de la classe « Sorcier », et bien, nous aurons un message d'erreur du compilateur. du compilateur.

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

.....

.....

.....

.....

.....

2m 37s

