



Support de cours

Cours:

## Introduction à la programmation orientée objet (en Java)

Vidéo:

### W16-05-JAVA-pt3

Concepts (extraits des sous-titres générés automatiquement) :

**Règle générale. Différences majeures. Méthodes abstraites. Notion de classe abstraite. Classes abstraites. Introduction des définitions. Principale raison. État du cavalier. Monture du cavalier. Aspect comportemental. Ajout de nouvelles méthodes. Notion d'état. Interfaces existantes. Introduction des méthodes. Héritage multiple.**



[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/>

# **Interfaces : Nouveautés depuis Java 8**

## **(Partie 3)**

**Introduction à la programmation orientée objet (en Java)**

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





Avec l'introduction des définitions par défaut pour les interfaces le contenu de ces dernières se rapproche désormais beaucoup de celui des classes abstraites.

notes

---

---

---

---

---

---

---

---

---

---

résumé

0m 1s



---

---

---

---

---

## Interface ou classe abstraite ?

↑  
methode abstraite  
+  
methode avec definition (par default)  
↑

En effet, on peut désormais mettre dans une interface aussi bien des méthodes abstraites, ce qui était le cas avant Java 8, et on peut également ajouter des méthodes qui ont désormais une définition concrète par défaut. Ce qui les rapproche beaucoup de la notion de classe abstraite.

notes

résumé

0m 13s



La différence majeure est que les interfaces ne permettent pas de modéliser des **états** (attributs).

La question peut donc a priori se poser de quand choisir une option, plutôt qu'une autre.

notes

résumé

0m 37s



La différence majeure est que les interfaces ne permettent pas de modéliser des états (attributs).

```
interface Cavalier {  
    Monture m;  
}
```

Eh bien une des différences majeures est qu'une interface ne peut pas spécifier d'état, elle ne peut pas contenir d'attributs. Par exemple, il n'est pas possible de définir quelle est la monture du cavalier dans l'interface, Cavalier. Ici, je ne pourrai pas mettre un attribut de type, Monture, dans le Cavalier.

notes

résumé

0m 42s



La différence majeure est que les interfaces ne permettent pas de modéliser des états (attributs).

```
interface Cavalier {  
    Monte m;  
}
```

Le fait de pouvoir spécifier un attribut permettrait de spécifier un

notes

résumé

1m 1s



La différence majeure est que les interfaces ne permettent pas de modéliser des états (attributs).

~~Constructeur~~

```
interface Cavalier {  
    Monture m;  
}
```

état du cavalier, dépendamment de quelle monture il chevauche, il serait dans tel ou tel état ; ceci n'est pas possible dans une interface. L'autre différence, qui est inhérente à la notion d'état, est qu'il n'y a pas de constructeur ;

notes

résumé

1m 3s





Comportemental / fonctionnel  
indépendant d'un état

La différence majeure est que les interfaces ne permettent pas de modéliser des **états** (attributs).

☞ Elles sont à privilégier s'il n'y a pas d'état.

ajout de nouvelles méthodes

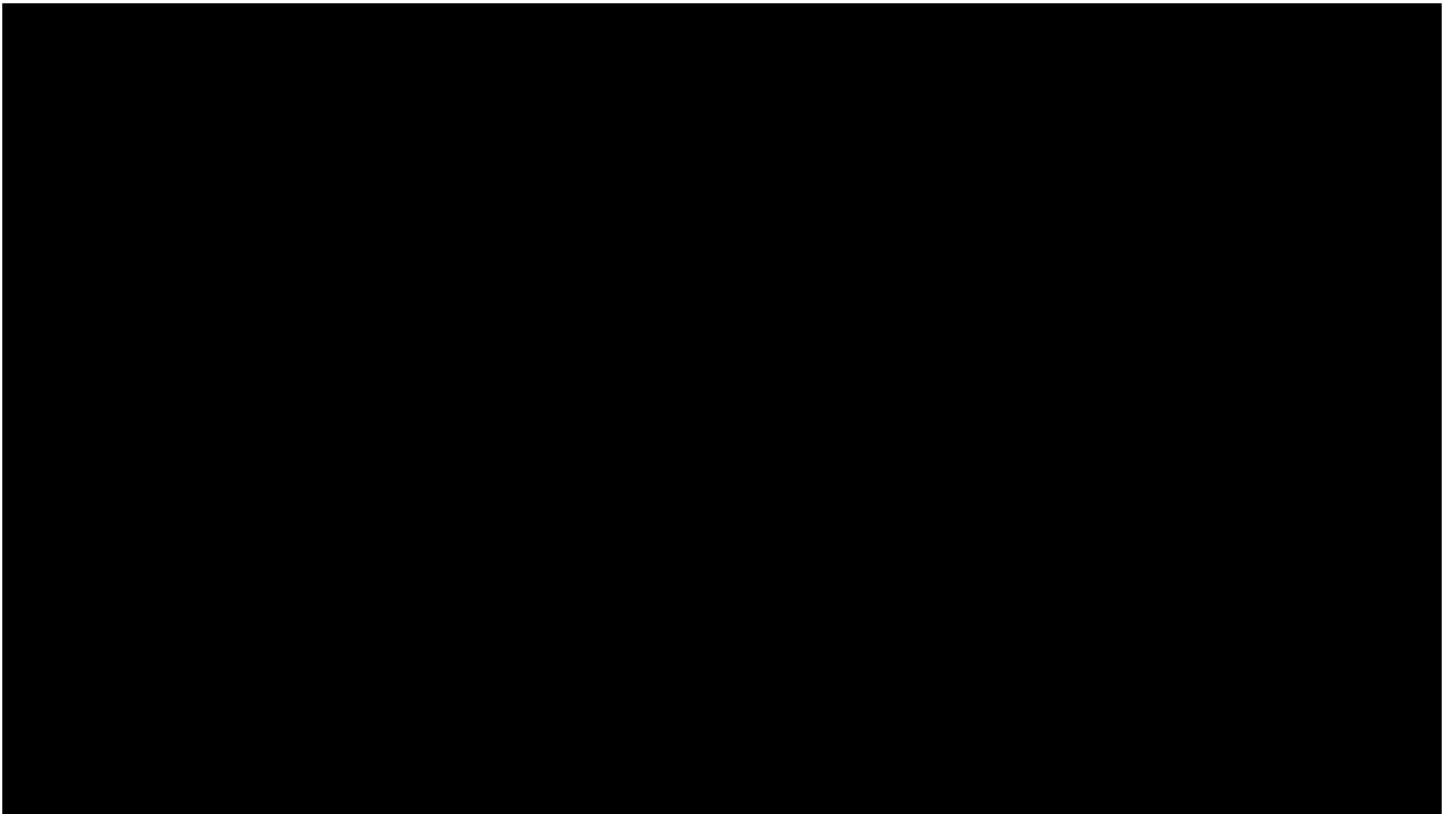
cette notion est inhérente à la notion d'état puisque le constructeur a pour vocation d'initialiser les attributs, or il n'y en a pas. Pour résumer, donc, pas d'état, pas de constructeur. La règle générale à suivre est que si ce que l'on souhaite modéliser est un aspect comportemental, fonctionnel, comme se déplacer, indépendant d'un état, d'objet, alors on va privilégier l'usage des interfaces. Le lien, implements, est en effet plus flexible, moins contraignant que celui de l'héritage au niveau de la conception. Rappelons enfin que la principale raison d'être de l'introduction des méthodes avec définition par défaut dans les interfaces, n'est pas tant de mimiquer l'héritage multiple, ou d'apporter une alternative aux classes abstraites, mais elle a surtout été de permettre l'ajout de nouvelles

notes

résumé

1m 25s





méthodes à des interfaces existantes, ... sans pénaliser l'existant, c'est-à-dire sans avoir à retoucher une classe implémentant ces interfaces. Et ceci conclut cette vidéo sur les spécificités post Java 7 des interfaces. spécificités post Java 7 des interfaces.

#### notes

---

---

---

---

---

---

---

---

---

---

#### résumé

2m 25s



---

---

---

---

---