

Support de cours

Cours:

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

Vidéo:

### W15-03-interfacesintro-JAVA-pt2

Concepts (extraits des sous-titres générés automatiquement) :

**Variable de type interface. Fait possible. Nouvelle classe. Méthode gestionclic. Nouveau type. Ambiguïté possible. Objet d'une classe. Dernière petite remarque. Interface graphique. Méthodes abstraites. Étiquette d'interface. Objet de type balle. Variable de type graphique. Interface. Composant possible d'une interface.**



[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/>

page 1/7

## Interfaces (Partie 2)

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

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



Une interface attribue un type supplémentaire à une classe d'objets, on peut donc :

- ▶ Déclarer une variable de type interface
- ▶ Y affecter un objet d'une classe qui implémente l'interface
- ▶ (éventuellement, faire un transtypage explicite vers l'interface)

```
Graphique graphique;  
Balle balle = new Balle(..);  
graphique = balle;  
Entite entite = new Balle(..);  
graphique = (Graphique) entite; // transtypage indispensable !
```

En déclarant une interface dans un programme java vous déclarez en fait un nouveau type, comme c'est le cas lorsque vous déclarez une nouvelle classe. Il est donc tout à fait possible de déclarer une variable de type interface comme ceci par exemple, et d'y affecter un objet mais attention, pas n'importe lequel, l'objet d'une classe qui implémente cette interface. C'est le cas ici, donc nous affectons à une variable de type Graphique, un objet de type Balle préalablement instancié. pour que ceci soit possible il faut, évidemment, que la classe Balle implémente l'interface Graphique. Il peut être intéressant dans certaines situations de manipuler des objets sous leur étiquette d'interface plutôt que sur leur étiquette propre, par exemple ici, plutôt manipuler une balle comme un objet graphique, et dans ce cas, ça justifie ce genre d'affectation. Dernière petite remarque : sachant qu'on peut parfaitement affecter un objet de type sous-classe à une variable de type super-classe, cette variable contenant un objet de type Balle donc, implémentant l'interface Graphique. Est-ce qu'il est possible d'affecter cette variable à un objet de type Graphique ? Alors, ceci est possible mais à une condition près, il faut ici rassurer votre compilateur sur le fait que ce qui est contenu dans la variable

notes

résumé

0m 1s



Une interface est un moyen d'attribuer des composants communs à des classes non-liées par une relation d'héritage :

- ☞ Ses composants seront disponibles dans chaque classe qui l'implémente

Composants possibles : (Java < 8)

1. Variables statiques finales (*assez rare*)

- ☞ Ambiguïté possible, nom unique exigé

2. Méthodes abstraites (*courant*)

- ☞ Chaque classe qui implémente l'interface sera obligée d'implémenter chaque méthode abstraite déclarée dans l'interface si elle veut pouvoir être instanciée
- ☞ Une façon de garantir que certaines classes ont certaines méthodes, sans passer par des classes abstraites
- ☞ Aucune ambiguïté car sans instructions

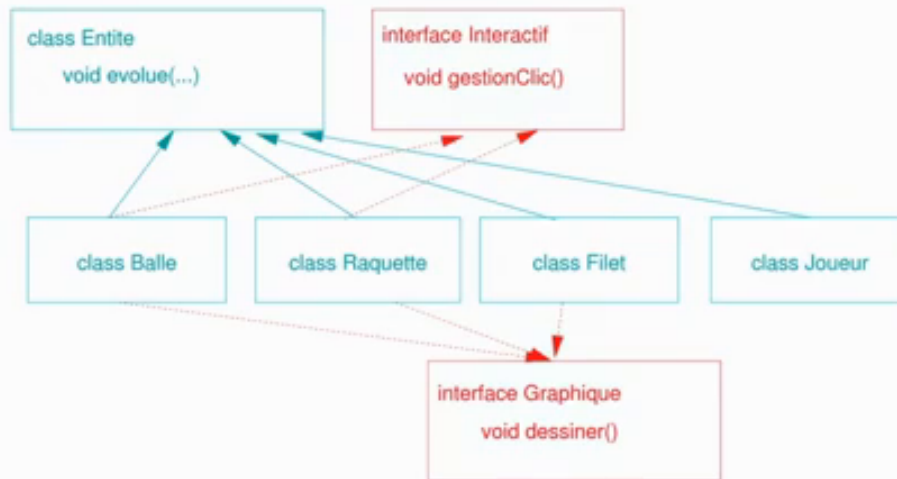
est bel et bien un objet qui implémente l'interface, en question, l'interface Graphique. Et pour se faire on a recours, à une conversion de type à un transtypage, ici.

notes

résumé

1m 25s





- Interface  $\neq$  Classe
- Une interface permet d'imposer à certaines classes d'avoir un contenu particulier sans que ce contenu ne fasse partie d'une classe.

Pour résumer, la fonction essentielle d'une interface est d'attribuer des composants communs à des classes qui ne sont pas liées par une relation d'héritage. En clair dans notre exemple, l'interface Interactif nous a permis ici d'imposer un contenu commun, une méthode gestionClic à deux classes, Balle, Raquette par exemple, qui ne sont pas liées entre elles par un lien d'héritage. Nous n'avons pas Balle héritant de Raquette ou vice versa.

notes

résumé

1m 35s



Une interface est un moyen d'attribuer des composants communs à des classes non-liées par une relation d'héritage :

- ☞ Ses composants seront disponibles dans chaque classe qui l'implémente

Composants possibles : (Java < 8)

### 1. Variables statiques finales (assez rare)

- ☞ Ambiguïté possible, nom unique exigé

### 2. Méthodes abstraites (courant)

- ☞ Chaque classe qui implémente l'interface sera obligée d'implémenter chaque méthode abstraite déclarée dans l'interface si elle veut pouvoir être instanciée
- ☞ Une façon de garantir que certaines classes ont certaines méthodes, sans passer par des classes abstraites
- ☞ Aucune ambiguïté car sans instructions

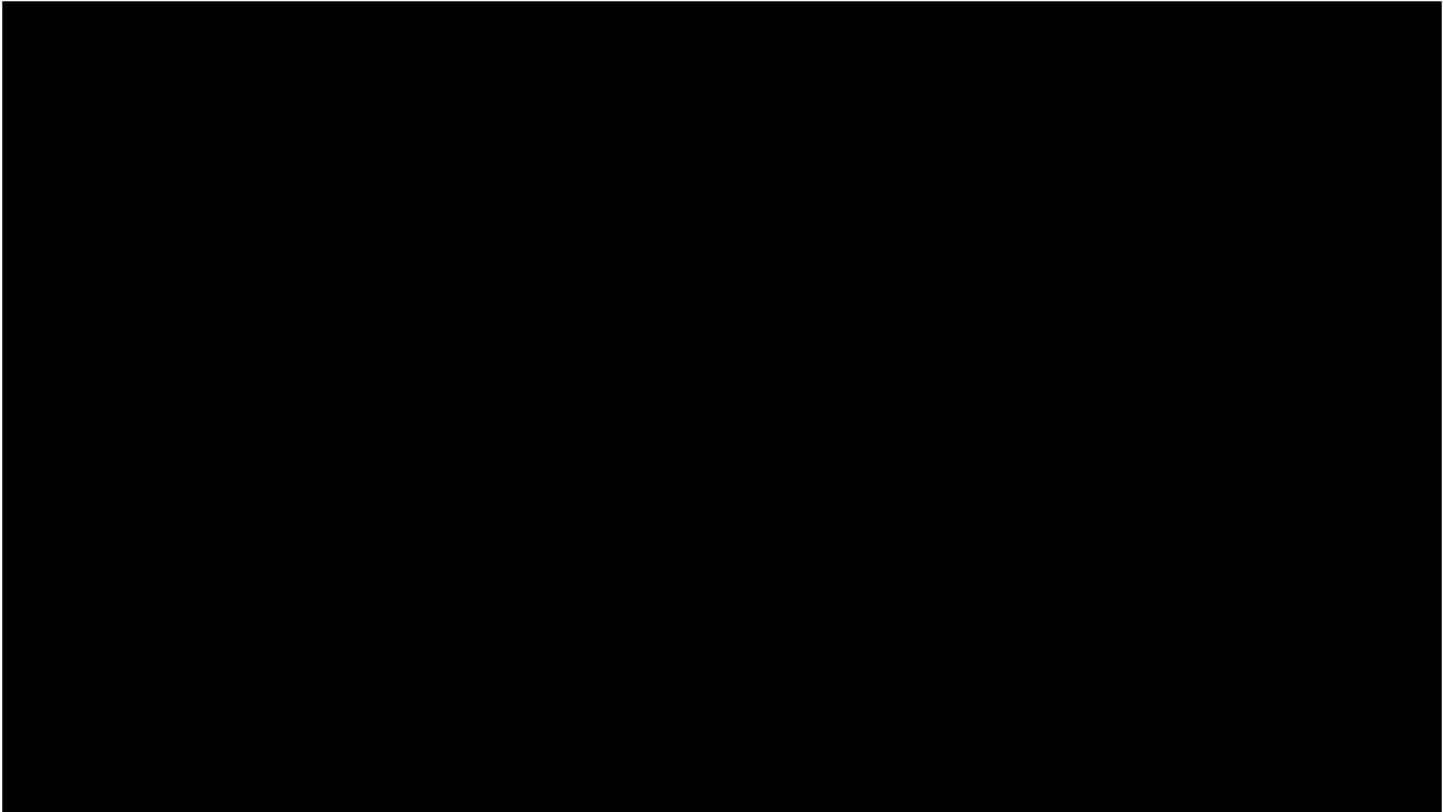
Une interface peut comporter soit des constantes c'est à dire, des variables finales, statiques, publiques. C'est assez rare et exige une certaine discipline. Pourquoi ? Parce que il peut y avoir une ambiguïté possible, si par exemple, une classe implémente deux interfaces qui chacune comporte la même constante. Supposez par exemple, que dans un programme nous ayons une première interface I1 qui contienne la définition d'une certaine constante et une seconde interface I2 qui contiendrait une constante du même nom. Avec possiblement, une autre valeur mais pas forcément. Si maintenant nous voulons déclarer une classe C qui implémenterait les deux interfaces, alors ici clairement, il y a un problème, il y a une ambiguïté : laquelle des deux constantes veut-on utiliser ? Et ceci sera refusé par le compilateur. Autre composant possible d'une interface et le plus couramment rencontré : les méthodes abstraites, et là nous avons vu que toute classe qui implémente l'interface, si elle veut pouvoir être instanciée, doit fournir une définition concrète de la méthode, sinon elle ne pourra pas être instanciée. Les interfaces permettent donc, d'imposer à certaines classes de fournir un certain contenu sans forcément, avoir recours à la notion de classe et de méthode abstraite. Contrairement au cas des variables

notes

résumé

2m 1s





comme les méthodes abstraites sont sans instructions il n'y a pas d'ambiguïté possible, par exemple, si dans une interface I1 il y a une méthode abstraite m et que dans une autre, interface I2, il y a également une méthode abstraite m alors une classe C peut parfaitement implémenter les deux interfaces sans que cela ne fasse réagir le compilateur. le compilateur.

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

3m 25s



.....

.....

.....

.....

.....