

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W13-02-2-protectedmasquage-JAVA-pt2

Concepts (extraits des sous-titres générés automatiquement) :

Redéfinition de méthode. Objet de type guerrier. Super-classe. Éventuelles sous-classes de personnages. Hiérarchie de classe. Méthode générale. Tournure syntaxique. Langages de programmation orienté. Syntaxe particulière. Méthode d'une super-classe. Guerrier de quelques bonnes manières. Super-classe personnage. Méthode. Terme de jargon. Sous-classe magicien.



[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/>

Héritage : masquage

(Partie 2)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s

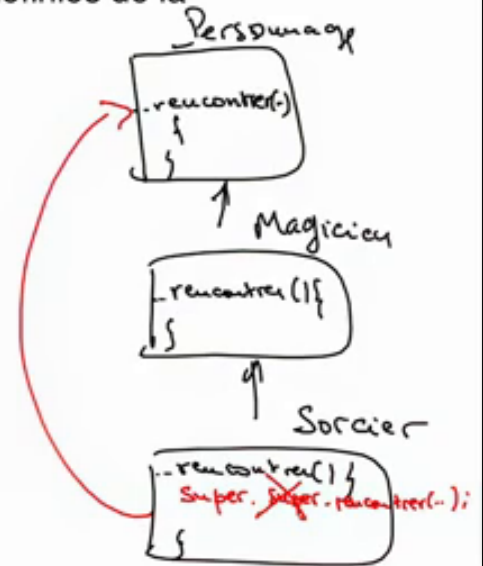


Accès à une méthode masquée (2)

Pour accéder aux attributs masqués et aux méthodes redéfinies de la super-classe :

- ▶ on utilise le mot réservé **super**
- ▶ Syntaxe : **super**. méthode ou attribut
- ▶ Exemple :

```
class Guerrier extends Personnage {
    //...
    public void rencontrer (Personnage perso) {
        super.rencontrer(perso); // salutation d'usage !!
        frapper(perso);
    }
}
```



notes

résumé

0m 1s

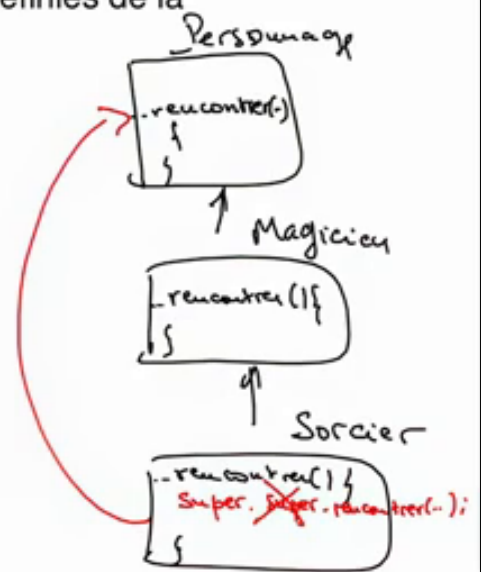


Accès à une méthode masquée (2)

Pour accéder aux attributs masqués et aux méthodes redéfinies de la super-classe :

- ▶ on utilise le mot réservé `super`
- ▶ Syntaxe : `super.méthode` ou `attribut`
- ▶ Exemple :

```
class Guerrier extends Personnage {
    //...
    public void rencontrer (Personnage perso) {
        super.rencontrer(perso); // salutation d'usage !!
        frapper(perso);
    }
}
```



notes

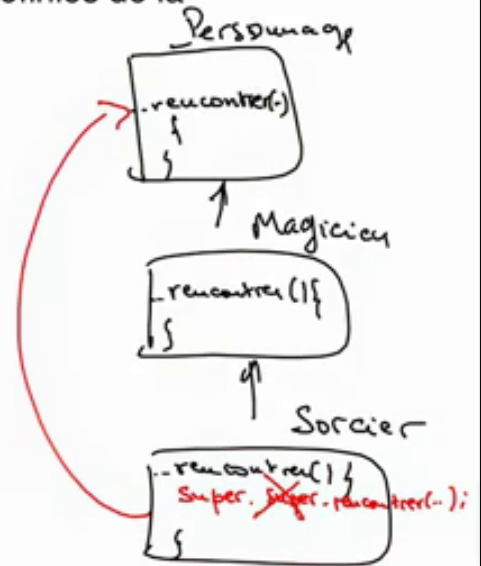
résumé

Accès à une méthode masquée (2)

Pour accéder aux attributs masqués et aux méthodes redéfinies de la super-classe :

- ▶ on utilise le mot réservé **super**
- ▶ Syntaxe : **super**. méthode ou attribut
- ▶ Exemple :

```
class Guerrier extends Personnage {
    //...
    public void rencontrer (Personnage perso) {
        super.rencontrer(perso); // salutation d'usage !!
        frapper(perso);
    }
}
```



objet de type "super-super-classe", que comme un objet de sa classe parente directe, ce qui peut être vu comme un affaiblissement

notes

résumé



de la relation "est-un", de la relation d'héritage. Notez que cette limitation n'est pas présente dans tous les langages de programmation orienté objet, certains langages orienté objet autorisent tout à fait ce genre de tournures, c'est-à-dire qu'une sous-sous-classe fasse appel à une méthode de la super-super-classe directement. Notez que si la méthode rencontrer n'avait pas été redéfinie au niveau intermédiaire ici, alors un appel à "super.rencontrer" va simplement rechercher la méthode rencontrer dans la super-classe la plus proche où elle se trouve, et donc aurait bel et bien été chercher la méthode rencontrer dans la classe Personnage. Et ceci conclut cette séquence. Et ceci conclut cette séquence.

notes

résumé

4m 37s

