

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W13-04-polymorphisme-JAVA-pt3

Concepts (extraits des sous-titres générés automatiquement) :

Classe jeu. Attribut de type personnage. Résolution statique des liens. Méthode publique. Type tableau personnage. Résolution dynamique des liens. Petit exemple analogue. Façon suivante. Méthode de la classe jeu. Joueur principal du jeu. Méthode rencontrer. Résolution dynamique. Type apparent des variables. Façon générale de rencontre. Première entrée.



[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/>

Polymorphisme : introduction

(Partie 3)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



Résolution dynamique des liens – Exemple (1)

```
class Jeu {
    private Personnage joueur;
    private Personnage[] adversaires;
    // ..
    public void tourDeJeu() {
        for (int i = 0; i < adversaires.length; ++i)
        {
            adversaires[i].rencontrer(joueur);
        }
    }
    // ...
}
```

Que se passe-t-il si :

```
adversaires[0] = new Sorcier(...);
adversaires[1] = new Guerrier(...);
// ...
leJeu.tourDeJeu();
```

Pour finir, résumons et synthétisons sur un petit exemple analogue à celui de notre exemple d'introduction. Supposons que nous disposons d'une classe Jeu dans laquelle il y aurait un attribut de type Personnage qui serait le joueur principal du jeu, et un autre attribut, de type tableau Personnage, qui, lui, représentera les adversaires qui vont rencontrer le joueur. La classe Jeu contient une méthode publique tourDeJeu qui fait en sorte que chaque adversaire rencontre tour à tour le joueur. Supposons que dans une méthode de la classe Jeu, par exemple un des constructeurs, le tableau d'adversaire soit rempli de la façon suivante : la première entrée contiendrait un sorcier et la seconde un guerrier. On suppose aussi, comme dans les exemples vus précédemment, qu'un sorcier a la façon générale de rencontrer un autre personnage, c'est-à-dire qu'il va simplement le saluer et que le guerrier, lui, rencontre le personnage d'une façon spécialisée, c'est-à-dire en le frappant. Si un programmeur utilisateur de la classe Jeu déclare un variable leJeu

notes

résumé

0m 1s



- ▶ Avec la résolution « statique » des liens, dans `tourDeJeu`, ce serait toujours `rencontrer(Personnage)` de `Personnage` qui serait appelé (**c'est le type apparent des variables qui décide**)
 - ☞ Le personnage principal du jeu se fait saluer deux fois : une fois par le guerrier et une autre fois par le sorcier
- ▶ Avec la résolution « dynamique » des liens, dans `tourDeJeu`, `rencontrer(Personnage)` de `Personnage` est appelée pour le sorcier mais `rencontrer(Personnage)` de `Guerrier` est appelée pour le guerrier (**c'est le type effectif qui décide**)
 - ☞ Le personnage principal du jeu se fait saluer par le sorcier ... mais frapper par le guerrier !
 - ☞ C'est ce qui va se passer en Java

de type `Jeu` et qu'il invoque la méthode `Tour de jeu` sur cette variable, que va-t-il se passer ? Avec la résolution statique des liens, si toutefois elle était possible, c'est le type apparent des variables qui va décider, donc le type `Personnage` qui va décider. Et à ce moment-là, le joueur va se faire saluer deux fois : une fois par le guerrier, une autre fois par le sorcier. Avec la résolution dynamique des liens, la méthode `Rencontrer` va s'adapter au contenu de la variable de type `Personnage`, c'est-à-dire de l'entrée du tableau. C'est le type effectif qui décide, et dans ce cas-là, le joueur va se faire saluer par le sorcier, mais frapper par le guerrier.

notes

résumé

1m 1s





La résolution dynamique est toujours ce qui va se passer en Java. La seconde option. Voilà, ceci conclut notre séquence d'introduction sur le polymorphisme en Java. Dès la séquence suivante, nous aurons l'occasion d'y revenir plus en profondeur et plus en détail. plus en détail.

notes

résumé

1m 37s

