Penser en Java

Résumé

PolyMorphisme : des objets interchangeables

Il arrive qu'on veuille traiter un objet non en tant qu'objet du type spécifique qu'ilest, mais en tant qu'objet de son type de base. Cela permet d'écrire du code

indépendant des types spécifiques. Dans l'exemple de la forme géométrique, les

fonctions manipulent des formes génériques sans se soucier de savoir si ce sont des

cercles, des carrés, des triangles ou même des formes non encore définies. Toutes les

formes peuvent être dessinées, effacées, et déplacées, donc ces fonctions envoient

simplement un message à un objet forme, elles ne se soucient pas de la manière

dont l'objet traite le message.

Un tel code n'est pas affecté par l'addition de nouveaux types, et ajouter de

nouveaux types est la façon la plus commune d'étendre un programme orienté objet

pour traiter de nouvelles situations. Par exemple, on peut dériver un nouveau type

de forme appelé pentagone sans modifier les fonctions qui traitent des formes

génériques. Cette capacité à étendre facilement un programme en dérivant de

nouveaux sous-types est important car il améliore considérablement la conception

tout en réduisant le coût de maintenance.

Abstract

Dans une modélisation, il est souvent souhaitable qu'une classe de base ne présente

qu'une interface pour ses classes dérivées. C'est à dire qu'on ne souhaite pas qu'il

soit possible de créer un objet de cette classe de base, mais seulement pouvoir

surtyper jusqu'à elle pour pouvoir utiliser son interface. Cela est possible en rendant

cette classe *abstraite* en utilisant le mot clef **abstract**. Le compilateur se plaindra si

une tentative est faite de créer un objet d'une classe définie comme **abstract**. C'est

un outil utilisé pour forcer une certain conception.

Le mot clef **abstract** est aussi utilisé pour décrire une méthode qui n'a pas encore

été implémentée - comme un panneau indiquant « voici une fonction de l'interface

dont les types dérivés ont hérité, mais actuellement je n'ai aucune implémentation

pour elle ». Une méthode **abstract** peut seulement être créée au sein d'une classe

**abstract**. Quand cette classe est dérivée, cette méthode doit être implémentée, ou la

classe dérivée devient **abstract** elle aussi. Créer une méthode **abstract** permet de

l'inclure dans une interface sans être obligé de fournir une portion de code

éventuellement dépourvue de sens pour cette méthode.

Le mot clef **interface** pousse le concept de classe **abstract** un cran plus loin en

évitant toute définition de fonction. Une **interface** est un outil très pratique et très

largement répandu, car il fournit une séparation parfaite entre l'interface et

l'implémentation. De plus, on peut combiner plusieurs interfaces, alors qu'hériter de

multiples classes normales ou abstraites est impossible.