POO - Le polymorphisme

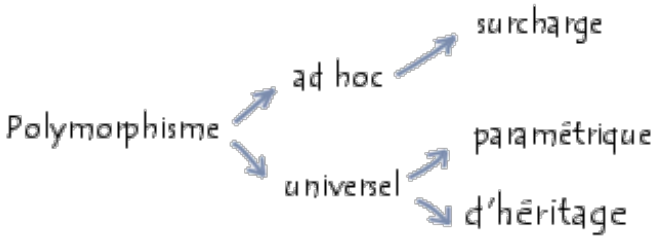
Septembre 2015

Définition du polymorphisme

Le nom de *polymorphisme* vient du grec et signifie *qui peut prendre plusieurs formes*. Cette caractéristique est un des concepts essentiels de la programmation orientée objet. Alors que l'héritage concerne les classes (et leur hiérarchie), le polymorphisme est relatif aux méthodes des objets.

On distingue généralement trois types de polymorphisme :

- Le polymorphisme ad hoc (également *surcharge* ou en anglais *overloading*)
- Le polymorphisme paramétrique (également *généricité* ou en anglais *template*)
- Le polymorphisme d'héritage (également *redéfinition*, *spécialisation* ou en anglais *overriding*)



Nous allons maintenant tenter de définir plus précisément tout cela, mais il est important de noter que beaucoup de confusions existent lorsqu'il s'agit de différencier tous ces types de polymorphisme.

Le polymorphisme ad hoc

Le polymorphisme ad hoc permet d'avoir des fonctions de même nom, avec des fonctionnalités similaires, dans des classes sans aucun rapport entre elles (si ce n'est bien sûr d'être des filles de la classe objet). Par exemple, la classe complexe, la classe image et la classe lien peuvent avoir chacune une fonction "afficher". Cela permettra de ne pas avoir à se soucier du type de l'objet que l'on a si on souhaite l'afficher à l'écran.

Le polymorphisme ad hoc permet ainsi de définir des opérateurs dont l'utilisation sera différente selon le type des paramètres qui leur sont passés. Il est donc possible par exemple de surcharger l'opérateur *+* et de lui faire réaliser des actions différentes selon qu'il s'agit d'une opération entre deux entiers (addition) ou entre deux chaînes de caractères (concaténation).

Le polymorphisme paramétrique

Le polymorphisme paramétrique, appelé généricité, représente la possibilité de définir plusieurs fonctions de même nom mais possédant des paramètres différents (en nombre et/ou en type). Le polymorphisme *paramétrique* rend ainsi possible le choix automatique de la bonne méthode à adopter en fonction du type de donnée passée en paramètre.

Ainsi, on peut par exemple définir plusieurs méthodes homonymes *addition()* effectuant une somme de valeurs.

- La méthode *int addition(int, int)* pourra retourner la somme de deux entiers
- La méthode *float addition(float, float)* pourra retourner la somme de deux flottants
- La méthode *char addition(char, char)* pourra définir au gré de l'auteur la somme de deux caractères
- etc.

On appelle *signature* le nombre et le type (statique) des arguments d'une fonction. C'est donc la signature d'une méthode qui détermine laquelle sera appelée.

Le polymorphisme d'héritage

La possibilité de redéfinir une méthode dans des classes héritant d'une classe de base s'appelle la **spécialisation**. Il est alors possible d'appeler la méthode d'un objet sans se soucier de son type intrinsèque : il s'agit du **polymorphisme d'héritage**. Ceci permet de faire abstraction des détails des classes spécialisées d'une famille d'objet, en les masquant par une interface commune (qui est la classe de base).

Imaginons un jeu d'échec comportant des objets *roi*, *reine*, *fou*, *cavalier*, *tour* et *pion*, héritant chacun de l'objet *piece*. La méthode *mouvement()* pourra, grâce au polymorphisme d'héritage, effectuer le mouvement approprié en fonction de la classe de l'objet référencé au moment de l'appel. Cela permettra notamment au programme de dire *piece.mouvement* sans avoir à se préoccuper de la classe de la pièce.

‹ Précédent

- 1
- 2
- 3
- 4
- 5
- 6

Réalisé sous la direction de Jean-François PILLOU,
fondateur de CommentCaMarche.net.