programmation objet

Ludovic Liétard

Introduction

- La programmation objet se caractérise par des concepts et des notions particulières
- Pour palier les inconvénients de la programmation procédurale qui se caractérise par :

Des fonctions définies par rapport aux SDD Peu ou pas du tout de modularité Une difficile réutilisation des composants

Introduction

 La programmation procédurale se focalise sur les traitements :

Quels traitements ?
Ils amènent à définir des SDD

 La programmation objet se focalise sur les données:

> Quelles sont les données ? Comment interagissent-elles ?

Les Objets

- Un objet est une donnée et toute donnée est soit une variable objet, soit une variable d'un type\primitif
- Un objet est une donnée abstraite c'est-à-dire que sa structure interne, ses composants ne sont pas connus du programmeur (connus uniquement par le développeur)
- On manipule un objet en lui envoyant des messages

Les Objets

■ Exemple :

Personne p ;

// p est un objet qui décrit une personne

... Initialisation de la personne...

//on envoie\à p le message *obtenirNom()* qui // délivre son nom

System.out.println("Nom = "+p.obtenirNom());

Les Classes

- Une classe est un modèle à partir duquel on crée des objets
- Une classe a un nom
- L'objet créé est une instance de la classe

Personne p1, p2;

// p1 et p2\sont deux instances de la classe
// Personne

... Initialisation et utilisation de p1 et p2

Les Classes

- Une classe décrit le contenu des objets par des variables d'instance (qui peuvent être des objets)
- Une classe décrit <u>les méthodes d'instance</u> qui représentent le code des messages que l'on envoie aux instances de la classe
- Une méthode d'instance particulière, <u>le</u> <u>constructeur</u>, est utilisée pour créer les instances (il existe un constructeur par défaut)
- On représente une classe par un diagramme (inspiration UML)

Les Classes

 Les classes peuvent hériter d'autres classes : mécanisme d'héritage

(cf. plus loin...)

Les Classes

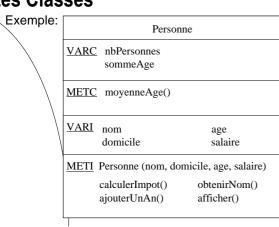
 Exemple (le constructeur a le même nom que la classe) :

\ <u></u>			
Personne			
VARI			
nom	age		
domicile	salaire		
METI Personne (nom, domicile, age, salaire)			
calculerImpot()	obtenirNom()		
ajouterUnAn()	afficher()		
l s			

Les Classes

- Les variables de classe sont des variables qui caractérisent la classe (elles ne sont pas des caractéristiques des objets).
- Les méthodes de classe sont des méthodes qui s'appliquent à la classe.

Les Classes



Les Classes

Les classes peuvent être <u>prédéfinies</u> : on utilise le nom de la classe,

ses méthodes (d'instance ou de classe)

Les classes peuvent être <u>définies</u>:
 on décide des VI, VC, MI, MC et leur implémentation, on nomme la classe on utilise le nom de la classe, ses méthodes

Les Classes

Exemple (Classe Personne supposée déclarée):

// déclaration d'une instance de Personne

Personne p;

// initialisation de la personne (constructeur Personne())

p = new Personne();

// Personne p = new Personne() est équivalent

// utilisation de la méthode de classe moyenneAge()

System.out.println("Nb = "+Personne.moyenneAge());

// utilisation de la méthode d'instance obtenirNom()

System.out.println("Nom = "+p.obtenirNom());

L' Encapsulation

 L'encapsulation constitue un principe fondamental de l'approche objet

C 'est l'interdiction formelle d'accéder aux valeurs des variables d'instance d'un objet sans passer par l'envoi de messages (l'exécution de méthodes).

 Malheureusement il est possible de passer outre mais il faudra respecter ce principe

L' Encapsulation

■ L'interêt est double :

l'accès aux contenus des objets est contrôlé le contenu des objets est caché aux utilisateurs

 On se focalise ainsi sur le comportement des objets et non pas sur leur implémentation. La maintenance du logiciel est rendue plus aisée.

par exemple la Classe Pile. On utilise les MI empiler, dépiler sans se soucier de l'implémentation. Un changement d'implémentation n'aura aucune incidence sur les logiciels utilisant des piles.

L' Encapsulation

Exemple

Personne			
<u>VARI</u>	nom domicile	age salaire	
METI Personne (nom, domicile, age, salaire)			
	obtenirNom() obtenirDomiciled obtenirAge() obtenirSalaire() calculerImpot() ajouterUnAn () afficher()	0	

La surcharge

- Une classe peut définir plusieurs méthodes ayant le même nom du moment que les types des arguments ou leur nombre diffèrent (le résultat des méthodes doit être identique).
- On dit alors que la méthode est surchargée

La surcharge

Exemple

	Personne		
	VARI	nom domicile	age salaire
	<u>METI</u>	Personne (nom, Personne(nom,	domicile, age, salaire)
\	obtenirNom() obtenirDomicile() obtenirAge() obtenirSalaire() modifier(nouveauSalaire) modifier(nouvelAge, nouveauDomicile)		

Le polymorphisme

- Le <u>polymorphisme</u> signifie que le même nom peut être donné à deux méthodes différentes, lorsqu'elles s 'appliquent à des objets différents
- Cela implique que ces méthodes visent au même but
- Exemple : une méthode d'instance de nom affiche peut être définie dans la classe Voiture et une méthode d'instance de nom affiche peut également être définie dans la classe Personne

Conclusion

- S'intéresser aux données et à leur comportement
- Définir des classes pour des données qui peuvent être concrètes (une voiture) ou abstraites (un transfert de compte bancaire à compte bancaire)
- Faire de l \encapsulation, utiliser du polymorphisme et de la surcharge
- Des langages pour la programmation objet:
 SmallTalk, C++, Java...
- Dans le cadre du cours : langage Java