Approche orienté objet

1. Introduction
2. Notions objet et classe

L’approche orienté objet (o.o.) induit une nouvelle culture de programmation. Elle nécessite une rupture avec les pratiques de la programmation traditionnelle (procédurale).

Dans l’approche procédurale, les données et les traitements sont séparés.

Dans l’approche objet, les données et les traitements concernant ces données sont regroupés dans des entités appelées objets.

Approche procédurale Approche objet

Traitement

Traitement

Traitement

Donnée

Donnée

Donnée

programme

Traitement

(Fct/procéd)

Donnée

Que doit faire le programme ? Sur quoi porte le programme ?

Un objet est un élément identifiable du monde réel qui est soit concret (stylo, voiture …) soit abstrait (entreprise, compte …).

Un objet est caractérisé par :

* Ce qu’il est (sont état) : c.-à-d. les données sur lui-même
* Ce qu’il sait faire (son comportement) : c.-à-d. les traitements qu’il peut réaliser
* Un objet fait partie d’une catégorie d’objet appelée classe.
* La classe est le type de l’objet.
* Différentes manières de dire la même chose
* Un objet est une instance de la classe
* Un objet est une variable dont le type est la classe
* Une classe est modèle d’objets
* Une classe est un générateur d’objets

Une classe est composée de deux parties :

* Les attributs (propriétés, champs caractéristiques…)
* Les méthodes (fonctions ou procédures)

En UML :

Compte

Nom classe

+ débiter ()

+ créditer ()

+ infos ()

* Nom
* Titulaire
* Solde

Les Attributs

Les méthodes

En Java : Compte.java

public class Compte {

//attributs

private int num ;

private String titulaire ;

private double solde ;

//méthodes

public void créditer (double somme){

solde=solde+somme ;

}

public void débiter (double somme){

solde=solde-somme ;

}

public string infos(){

string res ;

res=”Numero:”+num+”\n”;

res+=”Titulaire:”+titulaire+ ”\n”;

res+=”Solde:”+solde+”\n”;

return res;

}

} //Fin classe

1. Déclaration et manipulation d’un objet
2. Déclaration

Syntaxe

NomClasse nomObjet;

Exemple:

Compte c1;

c1 est un objet de type Compte.

* Instanciation :

Contrairement à une variable, un objet doit être instancié (crée, construit…) avant d’être utilisé.

Pour cela on fait appel à une méthode particulière appelée « constructeur ».

Syntaxe :

Compte c1 ; //déclaration

c1= new Compte() ; //Déclaration + instanciation

1. Accès aux membres d’un objet

/ ! \ Un objet ne peut être manipulé sans être instancié.

L’accès aux membres d’un objet (attribut ou méthode) se fait grâce à l’opérateur « . »

Syntaxe :

Obj.attribut

Obj.membre =>

Obj.méthode()

Exemple :

c1.crediter(200) ;

c1.num=24 ;

c1.titulaire= « Dupont » ;

1. Concept d’abstraction et d’encapsulation des données
2. Abstraction

C’est un principe que permet de masquer la complexité d’un objet à l’utilisateur. C’est grâce à l’abstraction qu’un utilisateur peut utiliser un objet sans savoir comment il fonctionne.

Un objet est considéré comme une boite noire.

Un objet possède une interface (constitué de l’ensemble des méthodes) visibles et manipulable par l’utilisateur et une implémentation (code) qui n’est pas visible par l’utilisateur.

1. Encapsulation

Ce principe découle directement du principe d’abstraction. Un objet ne peut être manipulé que via les méthodes qui lui sont associées lors de cette abstraction. La structure interne de l’objet (ses attributs) est inaccessible directement : il faut passer par des méthodes qui constituent l’interface.

De ce fait, les attributs sont qualifiés de privé et les méthodes de publiques.

* Pour chaque attributs déclaré privé, on peut définir 2 méthodes publiques : un set et un get.

La syntaxe est la suivante :

Pour l’attribut num de Compte :

Private int num ;

//set

Public void setNum (int unNum){

Num=unNum

}

//get

Public void getNum(){

Return num;

}

Au lieu de:

C1.num=24;

On écrira :

C1.setNum(24) ;

System.out.println(c1.num) ;

System.out.println(c1.getNum());