Programmation Objet

Classe et objet

Avec Java, la notion d'objet est omniprésente et nécessite un minimum d'apprentissage. Nous allons donc voir dans un premier temps les principes de la programmation objet et le vocabulaire associé, puis nous verrons comment mettre cela en application avec Java.

Dans un langage procédural classique, le fonctionnement d'une application est réglé par une succession d'appels aux différentes procédures et fonctions disponibles dans le code. Ces procédures et fonctions sont chargées de manipuler les données de l'application qui sont représentées par les variables de l'application. Il n'y a aucun lien entre les données et le code qui les manipule.

Programmation procédurale : une série d'étapes à réaliser

Dans un langage objet on va au contraire essayer de regrouper le code. Ce regroupement est appelé une classe. Une application développée avec un langage objet est donc constituée de nombreuses classes représentant les différents éléments manipulés par l'application. Les classes vont décrire les caractéristiques de chacun des éléments. C'est ensuite l'assemblage de ces éléments qui va permettre le fonctionnement de l'application.

Programmation objet : block de code classé par concept.

Ce principe est largement utilisé dans d'autres domaines que l'informatique. Dans l'industrie automobile, par exemple, il n'existe certainement pas chez aucun constructeur un plan complet décrivant les milliers de pièces constituant un véhicule. Par contre, chaque sous-ensemble d'un véhicule peut être représenté par un plan spécifique.

- Le chassis
- La boite de vitesse
- Le moteur
- …

Chaque sous-ensemble est également décomposé jusqu'à la pièce élémentaire

- Un boulon
- Un piston
- Un pignon
- …

C’est l’assemblage de tous ces éléments qui permet la fabrication d’un véhicule.

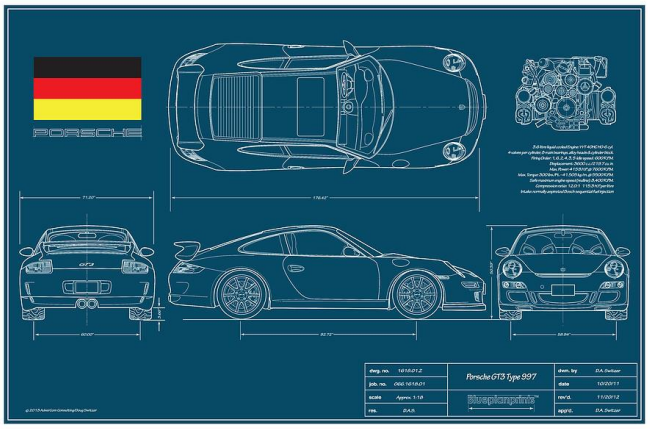
Attention : Ce n’est pas l’assemblage des plans qui permet la construction du véhicule, mais l’assemblage des pièces fabriquées à partir de ces plans.

Dans une application informatique c’est l’assemblage des objets créés à partir des classes qui va permettre le fonctionnement de l’application.

Les deux termes “classe” et “objet” sont souvent confondus mais ils représentent des notions bien distinctes.

La classe décrit la structure d’un élément alors que l’objet représente un exemplaire créé sur le modèle de cette structure. Après sa création, un objet est indépendant des autres objets construits à partir de la même classe. Par exemple la carrosserie d’une voiture pourra être peinte d’une couleur différente des autres voitures fabriquées selon le plan.

Classe = Plan



Objet = Pièce fabriquée

Par contre si un plan vient à être modifié, toutes les voitures fabriquées après la modification du plan bénéficieront des changements apportés au plan. Avec le risque de ne plus être compatible avec les anciennes versions.

Exo 1 : Voiture.java

1. Créer une classe Voiture, qui aura trois attributs :
    - Couleur
    - Nombre de portière
    - Marque
2. Par une saisie clavier, l'utilisateur doit pouvoir saisir les informations de sa voiture.
3. Imaginons un utilisateur qui possède plusieurs voitures, comment stocker les informations de toutes ses voitures ?
4. Créer une fonction qui permet d'afficher les informations de la ou des voiture(s) d'un utilisateur.

   > " Punto Orange 3 portes"

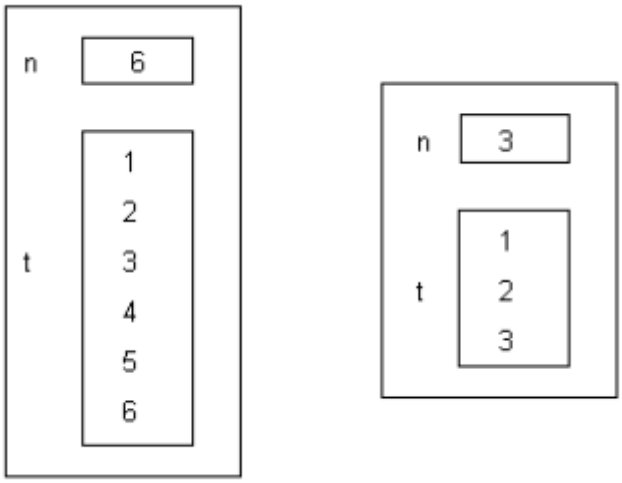
5. Dans le parking du triangle de Montpellier, comment stocker les informations de tous les adhérents et de leur véhicule. Afficher les informations pour vérifier.

Classes et instances

On souhaite créer des objets qui comporte :

- Un nombre n
- Un tableau t de n entiers 1, 2, 3, …, n

Exemple de deux tels objets :



Exo 2 : ObjetTableau.java

1. Créer une telle classe.
2. Créer un constructeur de cette classe
3. Créer des objets de la classe *ObjetTableau* **dans une autre classe** *ProgObjet* qui sert de classe principale.

```
Public void affiche() {
     System.out.println("affichage d'un objet");
     System.out.println("n vaut "+n);
     for (int i = 0; i < n; i++){
          System.out.print(t[i]+" * ");
     }
     System.out.println("");
}
```

Voici le code d'une méthode *affiche()*. Ce code doit être copié dans votre classe *ObjetTableau*.

La méthode affiche ne peut être exécutée que par une instance. Elle permet d'afficher un objet à l'écran, mais elle ne calcule rien, d'où son type de valeur de retour : void. La méthode *affiche()* est une **méthode d'instance**.

Un objet comporte des attributs d'instances et dispose de méthodes d'instances, définis dans la classe.

Ainsi, dans la classe *ObjetTableau*, n et t sont des **attributs d'instance** la méthode affiche() est une **méthode d'instance**.

Une classe peut aussi comporter des attributs de classe et des méthodes de classe. Ils sont signalés par le mot-clé **static**

Ils existent dès le début de l'application, même si aucun objet de la classe n'a encore été créé.

Seule exception : le constructeur est une méthode de classe, même s'il n'est pas précédé de static.

Exo 2 : ObjetTableau.java (suite)

4. Créer une fonction qui permet de compter les objets créés, et surtout vérifier que ça marche.
5. Noter dans le tableau suivant si la méthode appelée est une méthode de classe ou d'instance, puis le résultat de son exécution :

Instruction	Méthode : classe ou instance ?	Affichage
`ObjetTableau.nombreObjets();`		
`ObjetTableau obj1 = new ObjetTableau(6);`		
`obj1.affiche();`		
`ObjetTableau.nombreObjets();`		
`ObjetTableau obj2 = new ObjetTableau(3);`		

`obj2.affiche();`		
`ObjetTableau.nombreObjets();`		

Exo 3 : ObjetTableau.java (suite)

1. Ajouter à la classe *ObjetTableau* deux méthodes d'instance ***int maximum()*** et ***int minimum()*** renvoyant respectivement le plus grand et le plus petit élément du tableau t
2. Ajouter à la classe *ObjetTableau* une méthode d'instance ***int somme()*** renvoyant la somme des éléments du tableau t.
3. Ajouter à la classe *ObjetTableau* une méthode d'instance i***nt egal(int p)*** comptant le nombre d'éléments de t qui sont égaux à p
4. Ajouter à la classe *ObjetTableau* une méthode d'instance i***nt compris(int p, int q)*** comptant le nombre d'éléments de t qui sont strictement compris entre p et q. Attention il faut vérifier que p < q.
5. Ajouter à la classe *ObjetTableau* une méthode d'instance i***nt premier(int p)*** renvoyant le premier élément.
6. Ajouter à la classe *ObjetTableau* une méthode d'instance i***nt dernier(int p)*** renvoyant le dernier élément.
7. Ajouter à la classe *ObjetTableau* une méthode d'instance ***void inverse()*** inverse l'ordre des éléments d'un tableau.

Projet récréatif : Créér un tamagochi !!!

Cf wiki.