**Dossier BLOC 2 ALGO.1 : Base Programmation Objet**

|  |
| --- |
| Objets d’un traitement informatique  |

# Introduction

Au lieu de programmer avec des procédures et fonction on va travailler avec des classes (méthode) et objet.

objectif :

* code + facile à faire évoluer (+ simple à reprendre)
* masquer la complexité : donner des outils très simple (boite noire où on appuie sur des boutons pour faire des fonctions)

La principale différence de la programmation orientée objet (POO) vient du fait que l'objet regroupe les données et les moyens de traitement de ces données.

Un objet rassemble de fait deux éléments de la programmation procédurale.

* Les **champs/attributs** : Les champs sont à l'objet ce que les variables sont à un programme
* Les **méthodes** : procédures ou fonctions destinées à traiter les données.

## Objet & Classe

Le concept de classe correspond à la généralisation de la notion de type des programmes structurés.

L'objet en lui-même est une instance de classe, plus simplement un exemplaire d'une classe, sa représentation en mémoire.

Par conséquent, on déclare comme type une classe, et on déclare des variables de ce type appelées des objets.



## Formalisme Algorithme

### Classe

**CLASSE** MaClasse

**DÉBUT**

 **Privé**

 champ1 : type du champ

 champ 2 : type du champ

 …

 **Public**

 champPrivé1 : type du champ

 …

 procédure nom\_de\_la\_procedure (*liste de paramètre : type*)

 DÉBUT

 VARIABLES

 action(s)

 FIN

 FONCTION nom\_de\_la\_fonction (*liste de paramètre : type*) **:** type

 DÉBUT

 VARIABLES

 action(s)

 retourner unevariable

 FIN

 …

**FIN** **CLASSE**

### Objet

**ALGO** MonProgrammePrincipale()

DÉBUT

 VARIABLES

 Mavariable : **Maclasse**

 Mavariable <- **new** Maclasse()

 Mavariable.nom\_de\_la\_procedure(*liste de paramètre : type*)

FIN

## Formalisme Java

### Classe

**public** **class** MaClasse

{

 **private** **int** mavar;

 **private** String monString;

 **public** **void** maProcedure()

 {

 System.***out***.println("test");

 }

 **public** **int** maFonction()

 {

 **return** 1;

 }

}

### Objet

**public** **class** principal

{

 **public** **static** **void** main(String[] args)

 {

 MaClasse MaVarClasse;

 MaVarClasse = **new** MaClasse();

 MaVarClasse.maProcedure();

 }

}

**Question 1** : Quelle est la différence entre les attributs privés et publics ?

# Création d'une classe

Contexte :

Le concessionnaire "Lapie" vend un grand nombre de véhicule neuf et d'occasion.

Exemple de véhicule vendu :

* Renault Mégane, occasion, 90000km, année 2010, couleur rouge, immat CBO 582 75
* Peugeot 306, neuve, année 2016, couleur noire

**Question 2** : Veuillez définir les différentes éléments qui caractérisent une voiture.

**Question 3 :** Créer la classe voiture

## Les constructeurs

Comme leur nom l'indique, les constructeurs servent à construire l'objet en mémoire.

 Un constructeur va donc se charger de mettre en place les données, d'associer les méthodes avec les champs

Un objet peut avoir plusieurs constructeurs 🡪 notion de polymorphisme

Le polymorphisme veut dire que le même service, aussi appelé opération ou méthode, peut avoir un comportement différent selon les situations.

### Formalisme

**Algo**

CLASSE MaClasse

DÉBUT

 Privé

 champ1 : entier

 champ 2 : Chaine de caractère

 …

 Public

 MaClasse()

 début

 champ1=0

 champ2=""

 fin

 MaClasse(Var : entier)

 début

 champ1=Var

 fin

 MaClasse(Var : entier ; Var2 : chaine de caractère)

 début

 champ1=Var

 champ2=Var2

 fin

FIN CLASSE

**ALGO** MonProgrammePrincipale()

DÉBUT

 VARIABLES

 Mavariable : **Maclasse**

 Mavariable2 : **Maclasse**

 Mavariable <- **new** Maclasse(1)

 Mavariable2 <- **new** Maclasse(3,"Martin")

FIN

**Java**

**public** **class** MaClasse

{

 **private** **int** mavar;

 **private** String monString;

 **public** MaClasse()

 {

 mavar=0;

 monString="";

 }

 **public** MaClasse(**int** var)

 {

 mavar=var;

 }

 **public** MaClasse(**int** var, String var2)

 {

 mavar=var;

 monString=var2;

 }

}

**public** **class** principal

{

 **public** **static** **void** main(String[] args)

 {

 MaClasse MaVarClasse,MaVarClasse2;

 MaVarClasse = **new** MaClasse(1);

 MaVarClasse2= **new** MaClasse(3,"Martin");

 }

}

**Question 4 :** Compléter la classe voiture avec les constructeurs (un constructeur avec rien renseigné et un constructeur avec tous les champs).

## les méthodes GET & SET

Un accesseur (GET) est une méthode qui va nous permettre d'accéder aux variables de nos objets en lecture, et un mutateur (SET) nous permettra d'en faire de même en écriture ! Grâce aux accesseurs, vous pourrez afficher les variables de vos objets, et grâce aux mutateurs, vous pourrez les modifier.

### Formalisme

**Algo**

ALGOMonProgrammePrincipale()

DÉBUT

 VARIABLES

 Mavariable : Maclasse

 MonString : Chaine de Caractère

 .....

 AFFICHER ("valeur de ma variable :",

 **Mavariable.get\_champ1()**)

 MonString=**Mavariable.get\_champ2()**

FIN

CLASSE MaClasse

DÉBUT

 Privé

 champ1 : entier

 champ2 : Chaine de caractère

 Public

 FONCTION get\_champ1() : entier

 début

 retourner champ1

 fin

 FONCTION get\_champ2() : Chaine de caractère

 début

 retourner champ2

 fin

FIN CLASSE

**Java**

**public** **class** MaClasse

{

 **private** **int** mavar;

 **private** String monString;

 **public** **int** getMavar() {

 **return** mavar;

 }

 **public** **void** setMavar(**int** var) {

 **this**.mavar = var;

 }

 **public** String getMonString() {

 **return** monString;

 }

 **public** **void** setMonString(String unString) {

 **this**.monString = unString;

 }

}

**public** **class** principal

{

 **public** **static** **void** main(String[] args)

 {

 **int** monEntier ;

 MaClasse MaVarClasse,MaVarClasse2;

 MaVarClasse = **new** MaClasse(1);

 MaVarClasse2= **new** MaClasse(3,"Martin"); System.***out***.println(MaVarClasse2.getMonString());

 monEntier=MaVarClasse2.getMavar();

 }

}

**Sous Eclipse, il est possible de générer automatiquement les get, set & constructeurs :**

****

****

**Question 5 :** Compléter la classe voiture avec les get et set nécessaires à l'utilisation de la classe.

**Question 6** : Faire le programme principal qui permet de créer les 2 voitures du contexte.

# Exercices pratiques

**Exercice 1** :

Lorsqu'une voiture d'occasion reste un an sans avoir été vendue, la peinture de la voiture est refaite à neuf. Non seulement les rayures disparaissent mais la couleur peut également être changée.

La voiture Renault Mégane est dans le garage depuis plus de 1 an. Celle-ci va donc être repeinte en "gris métallisé".

**Compléter le programme principal pour repeindre cette voiture.**

**Exercice 2 :**

Il arrive pour diverses raisons qu'une voiture change de garage. Lors du déplacement de la voiture, le kilométrage se voit augmenté. La voiture Renault Mégane va donc partir dans un garage situé à 250 km plus loin.

**Créer la méthode qui permet d'augmenter le kilométrage d'une voiture.**

**Modifier le programme principal pour augmenter le kilométrage de la Renault Mégane.**

**Exercice 3 :**

Lors d'une transaction, le propriétaire de la voiture est donc connu : nom, adresse, téléphone.

**Créer la classe propriétaire.**

**Modifier la classe voiture pour ajouter l'objet propriétaire (ajouter le constructeur et le get & set)**

Le propriétaire "Jacque MARTIN" (tél : 0102030405) vient d'acheter la Renault Mégane.

**Modifier le programme principal pour ajouter le propriétaire à la voiture Renault Mégane.**