

Petit cours de POO en JAVA : Leçon 1

par [Eric Reboisson](#)

Date de publication : 18/05/2006

Dernière mise à jour : 18/05/2006

Dernièrement j'ai aidé un stagiaire dans sa mission d'étude d'impact du remplacement d'un composant de génération graphique commercial vers une solution gratuite. Cette étude se faisait dans un environnement applicatif RCP avec utilisation de plugins ECLIPSE, avec un existant assez rude de prime abord (beaucoup de classes, d'interfaces, de nouveautés TIGER, etc...). Ainsi dans cet environnement, le stagiaire n'était pas des plus rassurés, surtout que son niveau de JAVA était proche du néant (désolé pour lui ...) Aussi nous avons décidé de regarder ensemble les différents aspects de la programmation object au travers de JAVA en partant de la base, pour arriver à un mini projet qui finalement couvrait beaucoup d'aspects intéressants. Historique du contenu : 19/05/2006 - Les propriétés dans une classe - L'initialisation dans les constructeurs 18/05/2006 - Création d'une classe simple - Constructeurs

- I - Quelques rappels
- II - Premier pas
- III - Constructeur(s) de classe
- IV - Les propriétés dans une classe et notion d'encapsulation
- V - Conclusion

I - Quelques rappels

Quelques renvois ici pour expliquer certains fondamentaux que je ne traiterai pas dans cet article :

- POO : **P**rogrammation **O**rientée **O**bjet
- [Comment compiler et exécuter mon application Java ?](#)

A noter que pour cet article, [JDK 1.5](#) (5.0 ou TIGER) doit être installé et fonctionner.

II - Premier pas

Commençons par créer une classe la plus simple qui soit, en créant le fichier *MoyenDeTransport.java* (Veiller à bien respecter la casse) dans un répertoire de travail *D:\data\workspace\learningjava1* , et pour la suite de l'article nous nommerons ce répertoire *%WORK%*

Puis ajouter le code suivant dans le fichier :

La classe la plus simple possible

```
public class MoyenDeTransport {  
}
```

Ensuite compiler le fichier en vous plaçant dans *%WORK%* :

Compilation de la classe MoyenDeTransport

```
javac MoyenDeTransport.java
```

Le résultat de la compilation est un fichier compilé nommé *MoyenDeTransport.class* généré dans le répertoire *%WORK%* et représentant une classe JAVA nommée *MoyenDeTransport* .

III - Constructeur(s) de classe

Un constructeur est une méthode particulière dans une classe, le constructeur porte le même nom que la classe, et permet de créer une instance de cette classe (allocation mémoire).

Une classe peut avoir plusieurs constructeurs, qui seront différenciés par leurs paramètres.

Un constructeur vide existe toujours dans une classe, ainsi dans l'exemple précédent ce constructeur implicite n'est pas présent dans le code mais existe pour créer une instance de la classe.

Un constructeur sert également à initialiser des propriétés/comportements dans une classe (nous aborderons ce point plus loin)

Ajoutons maintenant le code suivant et compilons :

3 exemples de constructeurs

```
public class MoyenDeTransport {  
  
    // "Le" constructeur vide , implicite en JAVA.  
    public MoyenDeTransport() {  
  
        System.out.println("Constructeur vide");  
    }  
  
    // Un 2nd constructeur avec 1 paramètre  
    public MoyenDeTransport(String pname) {  
  
        System.out.println("Constructeur avec 1 paramètre");  
    }  
  
    // Un 3rd constructeur avec 2 paramètres  
    public MoyenDeTransport(String pname, String ptype) {  
  
        System.out.println("Constructeur avec 2 paramètres");  
    }  
}
```

Créons maintenant une classe de test pour utiliser notre classe *MoyenDeTransport* :

Une classe de test pour utiliser MoyenDeTransport

```
public class Test {  
  
    // La méthode Main :  
    // C'est une méthode spéciale, elle est utilisée lors de  
    // l'exécution de la classe par la commande : java Test  
    public static void main(String[] args) {  
  
        // On crée ici 3 instances de la classe MoyenDeTransport  
        MoyenDeTransport vehicule1 = new MoyenDeTransport();  
        MoyenDeTransport vehicule2 = new MoyenDeTransport("MERCEDES");  
        MoyenDeTransport vehicule3 = new MoyenDeTransport("CESNA", "Avion");  
    }  
}
```

Compiler cette classe de test en veillant bien à indiquer le **CLASSPATH** car nous dépendons maintenant d'une autre classe (si vous suivez bien, le ./ permet d'indiquer le répertoire courant pour trouver *MoyenDeTransport*):

Compilation en indiquant le CLASSPATH

```
javac -classpath ./ Test.java
```

Executer la classe, le résultat doit être le suivant :

Execution en indiquant le CLASSPATH

```
D:\data\workspace\learningjava1>java -classpath ./ Test  
Constructeur vide  
Constructeur avec 1 paramètre  
Constructeur avec 2 paramètres
```

IV - Les propriétés dans une classe et notion d'encapsulation

Ajoutons maintenant 2 propriétés *name* et *type* à notre classe *MoyenDeTransport* et des méthodes pour modifier et récupérer ses propriétés (c'est ce qu'on appelle l' encapsulation).

Ajout de propriétés à une classe

```
public class MoyenDeTransport {

    private String name;
    private String type;

    // "Le" constructeur vide , implicite en JAVA.
    public MoyenDeTransport() {

        System.out.println("Constructeur vide");
    }

    // Un 2nd constructeur avec 1 paramètre
    public MoyenDeTransport(String pname) {

        System.out.println("Constructeur avec 1 paramètre");
    }

    // Un 3rd constructeur avec 2 paramètres
    public MoyenDeTransport(String pname, String ptype) {

        System.out.println("Constructeur avec 2 paramètres");
    }

    // Les accesseurs ( getter ou setter ) permettent de modifier ou récupérer les attributs d'une
    classe
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

Puis ajoutons un peu de code à notre classe *Test* pour voir le résultat :

```
public class Test {

    // La méthode Main :
    // C'est une méthode spéciale, elle est utilisée lors de
    // l'execution de la classe par la commande : java Test
    public static void main(String[] args) {

        // On crée ici 3 instances de la classe MoyenDeTransport
        MoyenDeTransport vehicule1 = new MoyenDeTransport();
        MoyenDeTransport vehicule2 = new MoyenDeTransport("MERCEDES");
        MoyenDeTransport vehicule3 = new MoyenDeTransport("CESNA", "Avion");

        // On affecte ici "RENAULT" à la propriété name de l'instance vehicule1
        vehicule1.setName("RENAULT");
        System.out.println(vehicule1.getName());
        System.out.println(vehicule2.getName());
        System.out.println(vehicule3.getType());
    }
}
```

Recompiler la classe *MoyenDeTransport.java* et *Test.java* et exécuter la classe *Test*, le résultat doit être :

Affichage des propriétés grâce aux accesseurs

```
D:\data\workspace\learningjava>java -classpath ./ Test
Constructeur vide
Constructeur avec 1 paramètre
Constructeur avec 2 paramètres
RENAULT
null
null
```

On remarque ici l'affichage de 2 valeurs nulles, car pour *vehicule2* et *vehicule3* ni le *name*, ni le *type* n'ont été renseignés.

Voyons comment les initialiser en utilisant les constructeurs de la classe MoyenDeTransport :

Initialisation des propriétés grâce aux constructeurs

```
public class MoyenDeTransport {

    private String name;
    private String type;

    // "Le" constructeur vide , implicite en JAVA.
    public MoyenDeTransport() {

        // this(0..N paramètres) , permet d'invoquer le constructeur selon son nombre de paramètres
        this("default");
    }

    // Un 2nd constructeur avec 1 paramètre
    public MoyenDeTransport(String pname) {

        this(pname, "default");
    }

    // Un 3rd constructeur avec 2 paramètres
    public MoyenDeTransport(String pname, String ptype) {

        // this : une référence sur l'objet courant
        this.name = pname;
        this.type = ptype;
    }

    // Les accesseurs ( getter ou setter ) permettent de modifier ou récupérer les attributs d'une
    classe
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

Recompiler la classe *MoyenDeTransport.java* et exécuter la classe *Test*, le

Affichage des propriétés initialisées dans les constructeurs

```
D:\data\workspace\learningjava>javac -classpath ./ MoyenDeTransport.java

D:\data\workspace\learningjava>java -classpath ./ Test
RENAULT
MERCEDES
Avion
```


V - Conclusion

Cet article en construction va évoluer pour essayer d'aborder les bases de la programmation objet en JAVA.

A venir...L'héritage...Les bienfaits de l'abstraction