1. Introduction à la programmation orientée-objet avec le langage Java

[02.10.2017]

A. **Qu'est-ce que Java [1]?**
B. **Pourquoi Java ?**
C. **Modèle en cascade pour le développement logiciel**
D. **Langages procéduraux et fonctionnels vs orientés-objet**
E. **Concepts de programmation orientée-objet**
F. **Programmer avec Java**

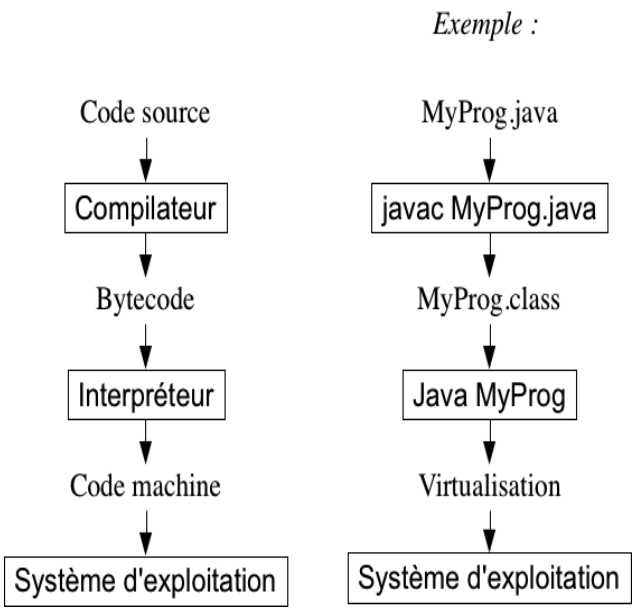
[1] Il ne faut pas confondre Java: langage orienté-objet avec JavaScript: langage de scripts utilisé principalement sur les sites web.

A. Qu'est-ce que Java ? JAVA ≠JavaScript

Java est :

- **un langage indépendant de la plate-forme**
- **un langage orientée objet**
  La programmation orientée objet est une façon d'organiser des programmes comme **collection d'objets,** dont *chacun représente une instance d'une* ***classe***.
- **un langage compilé**

Java est un langage ***interprété*** → un programme compilé n'est pas directement exécutable par le système d'exploitation ***MAIS*** il doit être interprété par un autre programme, qu'on appelle ***interpréteur***



Un programmeur Java écrit :

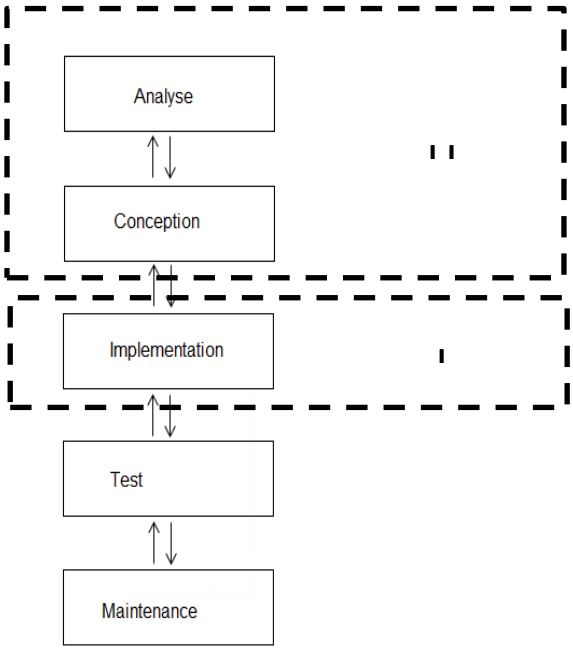
- ✓ son code source, sous la forme de classes,
- ✓ l'extension est .java. Ce code source est alors compilé par le compilateur javac
- ✓ en un langage appelé bytecode et enregistre le résultat dans un fichier dont l'extension est .class.
- ✓ Il doit être interprété par la machine virtuelle de Java qui transforme alors le code compilé en code machine compréhensible par le système d'exploitation.

B. Pourquoi Java ?

Le langage Java a l'avantage d'être :

- ✓ **Modulaire/ Réutilisable** : on peut écrire des portions de code génériques, c'est-à-dire utilisables par plusieurs applications. Le même code est utilisable en tant que composant de différents systèmes dans diverses applications.
- ✓ **Rigoureux** : la plupart des erreurs se produisent à la compilation et non à l'exécution
- ✓ **Portable** : un même programme compilé peut s'exécuter sur différents environnements. La portabilité est la capacité du logiciel à fonctionner avec un changement minimal sur différentes plates-formes matérielles et systèmes d'exploitation.
- ✓ **Adaptable :** Les applications modernes, telles que les navigateurs Web et les moteurs de recherche Internet, impliquent habituellement de grands. Le logiciel doit donc pouvoir évoluer.
  Java reste souple et puissant, malgré sa taille et sa conception simple. Il ne possède ni pointeur, ni arithmétique sur les pointeurs (comme en C). Ses chaînes et ses tableaux sont de véritables objets. Sa gestion de la mémoire est automatique.

C. Modèle en cascade pour le développement logiciel

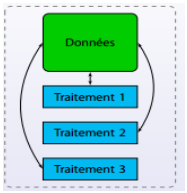


Ce modèle identifie 5 phases :

- ✓ **Analyse** : Un cahier des charges est défini pour identifier les besoins de l'application.
- ✓ **Conception** : L'architecture du système est décrite. Différentes composantes du logiciel sont identifiées puis détaillées
- ✓ **Implémentation** : Le modèle issu de la conception est implémenté.
- ✓ **Test** : Des tests unitaires et globaux permettent de corriger des erreurs faites lors des précédentes étapes.
- ✓ **Maintenance** : C'est le suivi du logiciel après sa livraison. Les retours des utilisateurs (correction d'erreurs).

D. Langages procéduraux et fonctionnels vs orientés-objet

Langages procéduraux et fonctionnels



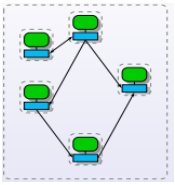
Plusieurs **procédures/fonctions** :

- ❑ qui effectuent un traitement sur des données (procédure)
- ❑ qui retournent une valeur après leur invocation (fonction)

Exemples: C, Fortran, etc.

Langages orientés-objet

Un programme est composé de :



Plusieurs **objets** qui contiennent:

- ❑ des données "internes"
- ❑ des traitements manipulant ces données internes ou d'autres données. Les données d'un objet sont appelés ses attributs et ses traitements sont ses méthodes /opérations

Exemples : Java, C++, Python, Ruby, etc.

E. Concepts de programmation orientée-objet

La programmation orienté objet (POO) est une autre manière de programmer.
C'est un paradigme de programmation.
Jusqu'à présent nous avons étudié la programmation procédurale, avec les notions suivantes :

- les variables
- les instructions et les blocs d'instructions
- les structures conditionnelles : if, while, for.
- les fonctions.

Avec la programmation objet, nous accédons à un niveau supérieur de raisonnement, plus abstrait que la programmation procédurale.
Lorsque que l'on code en objet, on ne pense pas machine, on pense tout simplement objet.
Pour coder objet, il faut penser objet.

Dans cette première partie, nous allons définir cette notion d'objet.

a. Objet

Comment définir la Programmation orientée objet ?

- un objet est une brique logicielle,
- un objet représente une idée, un concept, une entité,
- pour représenter un objet, on utilise un modèle,
- un objet possède une structure :
    - ce sont ses propriétés, ou attributs,
    - les propriétés décrivent ses caractéristiques, ce qui le compose,
    - les propriétés sont des variables propres à un objet,
    - les propriétés ont une valeur (comme une variable),
    - cette valeur peut être :
        - nulle,
        - un booléen
        - un numérique
        - une chaîne de caractères
        - un objet
    - les valeurs sont propres à un objet
- un objet a un comportement :
    - ce sont ses méthodes,
    - les méthodes décrivent une action, un état, donc un verbe,
    - les méthodes ont pour équivalence les fonctions dans la programmation procédurale,
    - le comportement est le même pour tous les objets regroupés dans le même modèle,
- On utilise les objets :
    - dès qu'on a un besoin,
    - sans limite de nombre (si ce n'est la capacité de la mémoire),
    - à partir d'un modèle,
    - pour une durée limitée,
- les objets interagissent entre eux,
- un objet peut hériter des caractéristiques d'un parent,
- un objet a une vie :
    - il naît,
    - il meurt,

- pour concevoir des objets, on utilise un langage de modélisation : UML, et tout particulièrement le diagramme des classes,

Approche procédurale vs orientée-objet

Procédurale	**Orientée-objet**
‘‘Que doit faire mon programme ?”	‘‘De quoi doit être composé mon programme ?”

Le livre de Java premier langage

Le canard Enchaine

Directeur de banque

Programmer en Java

Le monde

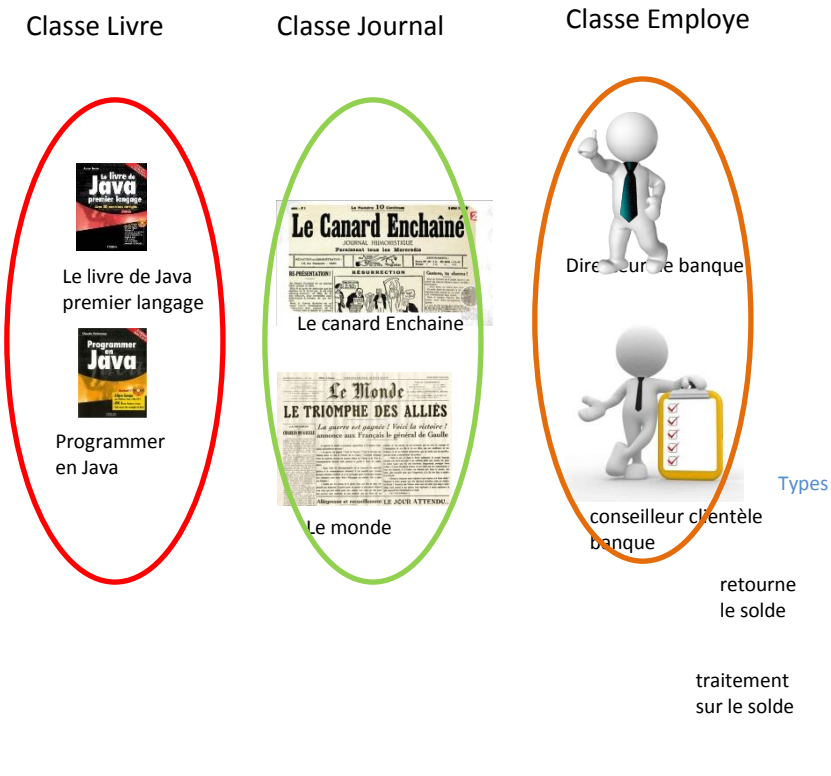
conseilleur clientèle banque

b. Classe

Une classe définit un ensemble d'entités, d'objets.
C'est un modèle, une description.
Elle définit la structure et le comportement.
La classe est statique, elle ne peut pas muter, donc changer de comportement.
Pour muter, il faudra créer une nouvelle classe héritant de la première.
Une classe contient des objets qu'on appelle des instances. Elle contient de 0 à n objets.
La création d’un objet se fait à partir d'une classe.

Des objets similaires peuvent être informatiquement décrits par une même abstraction : *une* ***classe***

- ❑ même structure de données et méthodes de traitement
- ❑ valeurs différentes pour chaque objet



Une classe est composée de plusieurs membres dont chacun est soit :

- ✓ un attribut : variable typée
- ✓ une méthode: ensemble d'instructions de traitement

Exemple

```
class CompteBancaire {
    String proprietaire;
    double solde;                          attributs

    double getSolde() {
        return solde;                      méthode
    }                                      (fonction)
    void credite(double val) {
        solde = solde + val;               méthode
    }                                      (procédure)
}
```

c. Abstraction :

Abstraction. L'abstraction est un processus où vous "cachez" les détails inutiles d'un objet.

Par exemple: une voiture est un objet bien défini, qui se compose de plusieurs autres objets comme un système d'engrenage, un mécanisme de direction, un moteur, qui ont de nouveau leurs propres sous-systèmes. Mais pour les humains, la voiture est un objet unique, qui peut être géré par l'aide de ses sous-systèmes, même si leurs détails intérieurs sont inconnus.

d. Encapsulation :

Lors de la conception d'un programme orienté-objet, le programmeur doit identifier les objets et les données appartenant à chaque objet mais aussi des droits d'accès qu'ont les autres objets sur ces données. L'encapsulation de données dans un objet permet de cacher ou non leur existence aux autres objets du programme.

On cache les propriétés et les méthodes internes au fonctionnement de l'objet.
On rend visible uniquement les méthodes qui doivent être vues de l'extérieur.
On parle de **visibilité**.

Une donnée peut être déclarée en accès :

public : les autres objets peuvent accéder à la valeur de cette donnée ainsi que la modifier

privé : les autres objets n'ont pas le droit d'accéder directement à la valeur de cette donnée (ni de la modifier). En revanche, ils peuvent le faire indirectement par des méthodes de l'objet concerné (si celles-ci existent en accès public).

e. Héritage (Inheritance) :

Une classe est décrite par des propriétés et des méthodes. Chaque classe a ses spécificités.
Le but de l'héritage est de regrouper les propriétés et les méthodes communes à plusieurs classes dans une nouvelle classe (la classe parente) puis d'hériter de cette classe.

On parle aussi de généralisation.
Ainsi, pour définir une nouvelle classe, il suffit de la faire hériter d'une classe existante et de lui ajouter de nouvelles propriétés/méthodes.

De cette façon, les classes héritées forment une hiérarchie descendante, au sommet de laquelle se situe la classe de base (superclasse). On appelle également la classe héritée la sous-classe et la classe parente la super-classe.

Une classe se définit tout simplement grâce au mot-clé *class* de la manière suivante par exemple :

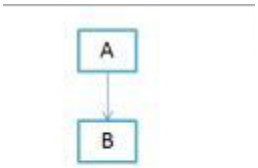
```
public class Animal{
   ...
   public void sound(){
      System.out.println("Animal is making a sound");
   }
}
```

Pour faire hériter une classe d'une superclasse, Java fournit le mot-clé **extends**.
Par exemple :

```
public class Cat extends Animal{
...
    @Override
    public void sound(){
        System.out.println("Meow");
    }
}
```

Types d'héritage en Java:

Single : Le diagramme ci-dessous montre que la classe B étend une seule classe qui est A. Ici A est la classe parente (super-classe) et B serait une sous-classe.



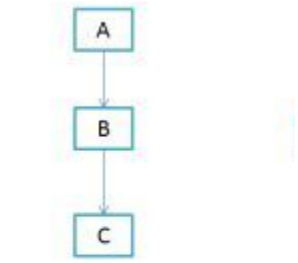
(a) Single Inheritance

En Java :

```
Class A
{
   public void methodA()
   {
     System.out.println("Base class method");
   }
}

Class B extends A
{
   public void methodB()
   {
     System.out.println("Child class method");
   }
   public static void main(String args[])
   {
     B obj = new B();
     obj.methodA(); //calling super class method
     obj.methodB(); //calling local method
  }
}
```

Multilevel : Classe C est la sous-classe de classe B and classe B la sous-classe de A. A est la super-classe.

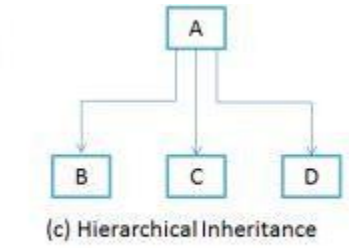


(d) Multilevel Inheritance

En Java :

```
Class X
{
   public void methodX()
   {
     System.out.println("Class X method");
   }
}
Class Y extends X
{
public void methodY()
{
System.out.println("class Y method");
}
}
Class Z extends Y
{
   public void methodZ()
   {
     System.out.println("class Z method");
   }
   public static void main(String args[])
   {
     Z obj = new Z();
     obj.methodX(); //calling grand parent class method
     obj.methodY(); //calling parent class method
     obj.methodZ(); //calling local method
  }
}
```

Hiérarchique (Hierarhical) : Lorsqu'une classe a plus d'une sous-classe/ plusieurs sous-classes ont la même classe parent (super- classe), ce type d'héritage est connu comme un héritage hiérarchique.



(c) Hierarchical Inheritance

En Java :

```
class A
{
   public void methodA()
   {
      System.out.println("method of Class A");
   }
}
class B extends A
{
   public void methodB()
   {
      System.out.println("method of Class B");
   }
}
class C extends A
{
  public void methodC()
  {
     System.out.println("method of Class C");
  }
}
class D extends A
{
  public void methodD()
  {
     System.out.println("method of Class D");
  }
}
class JavaExample
{
  public static void main(String args[])
  {
     B obj1 = new B();
     C obj2 = new C();
     D obj3 = new D();
     //All classes can access the method of class A
     obj1.methodA();
     obj2.methodA();
     obj3.methodA();
  }
}
```

f. Polymorphisme (Polymorphism) :

poly comme plusieurs et morphisme comme forme.

Le polymorphisme traite de la capacité de l'objet à posséder plusieurs formes.

Cette notion intervient sur le comportement de l'objet, donc les méthodes.

Le comportement de l'objet devient donc modifiable.

G. Programmer avec Java

Avant de commencer :

- ✓ tous les programmes Java sont composés d'au moins une classe qui doit contenir une méthode main (méthode principale du programme)
- ✓ Qu'est-ce qu'une méthode ? une suite d'instructions à exécuter.
- ✓ Qu'est-ce qu'elle contient ?
  - une entête : la carte d'identité de la méthode
  - un corps : le contenu de la méthode (Délimité par des accolades {})
  - une valeur de retour : le résultat que la méthode va retourner
    **pour les méthodes de type void : renvoient rien

Plus loin:

• Ken Arnold and James Gosling: *The Java Programming Language,* Addison-Wesley.
• Java Technology Home Page: http://java.sun.com.
• Le tutorial Java, http://docs.oracle.com/javase/tutorial/
• Le site JavaWorld, http://www.javaworld.com
• http://www.javamug.org/mainpages/Java.html