

# Formation POE JAVA

## Journée #1: Introduction à la programmation orientée-objet avec le langage Java

*Athanasia Katsouraki*

# Déroulement du cours

119h : Développer les bases de la programmation en Java web

35h : Pratiquer le requêtage de base de données

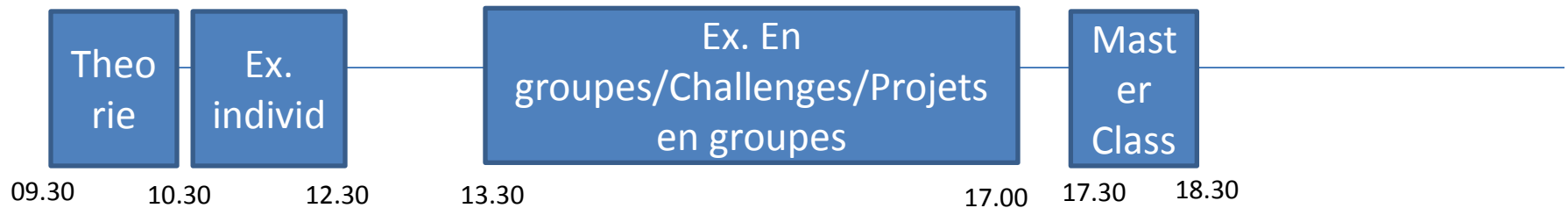
14h : S'initier aux outils de l'intégration continue

## -Une journée type –

1 – 1h30' : Théorie

1h30' -2h : Exercices individuels – Tutoriels

3h30' -4h : Exercices/Tutoriels en groupes



# Agenda

- ☐ Qu'est-ce que Java?
- ☐ Environnement Java
- ☐ Pourquoi Java?
- ☐ Modèle en cascade pour le développement logiciel
- ☐ Langages procéduraux et fonctionnels vs orientés-objet
- ☐ Concepts de programmation orientée-objet
  - ☐ Objet
  - ☐ Classe
  - ☐ Encapsulation
  - ☐ Héritage
    - ☐ Types d'héritage en Java
  - ☐ Polymorphisme

# Qu'est-ce que Java ?

Java est :

- ✓ un langage indépendant de la plate-forme
- ✓ un langage orientée objet
- ✓ un langage compilé

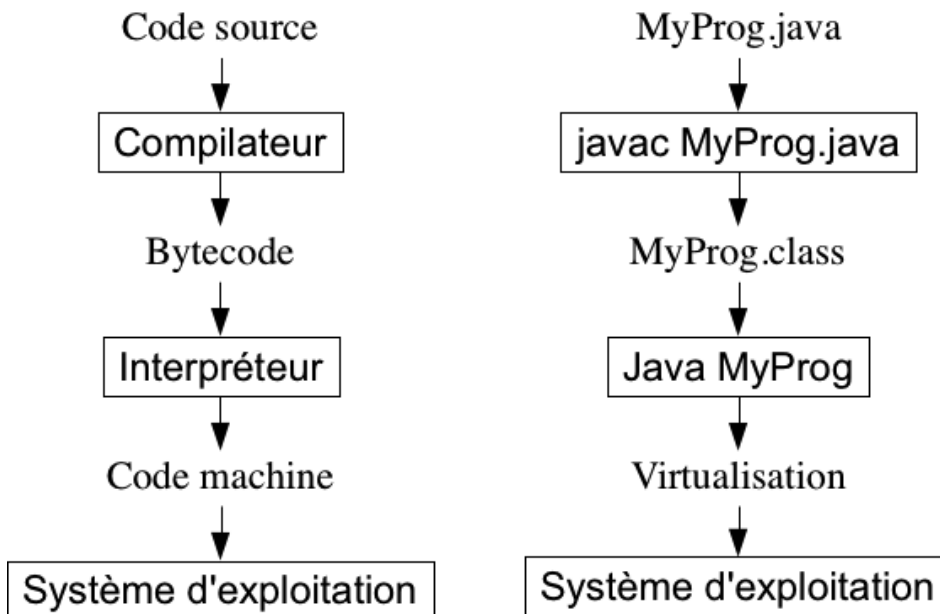
# Environnement Java

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**

*Exemple :*

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.



# Pourquoi?



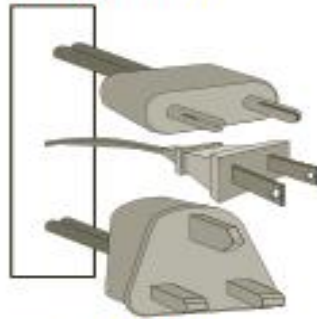
Modulaire/Reutilisable



Portable



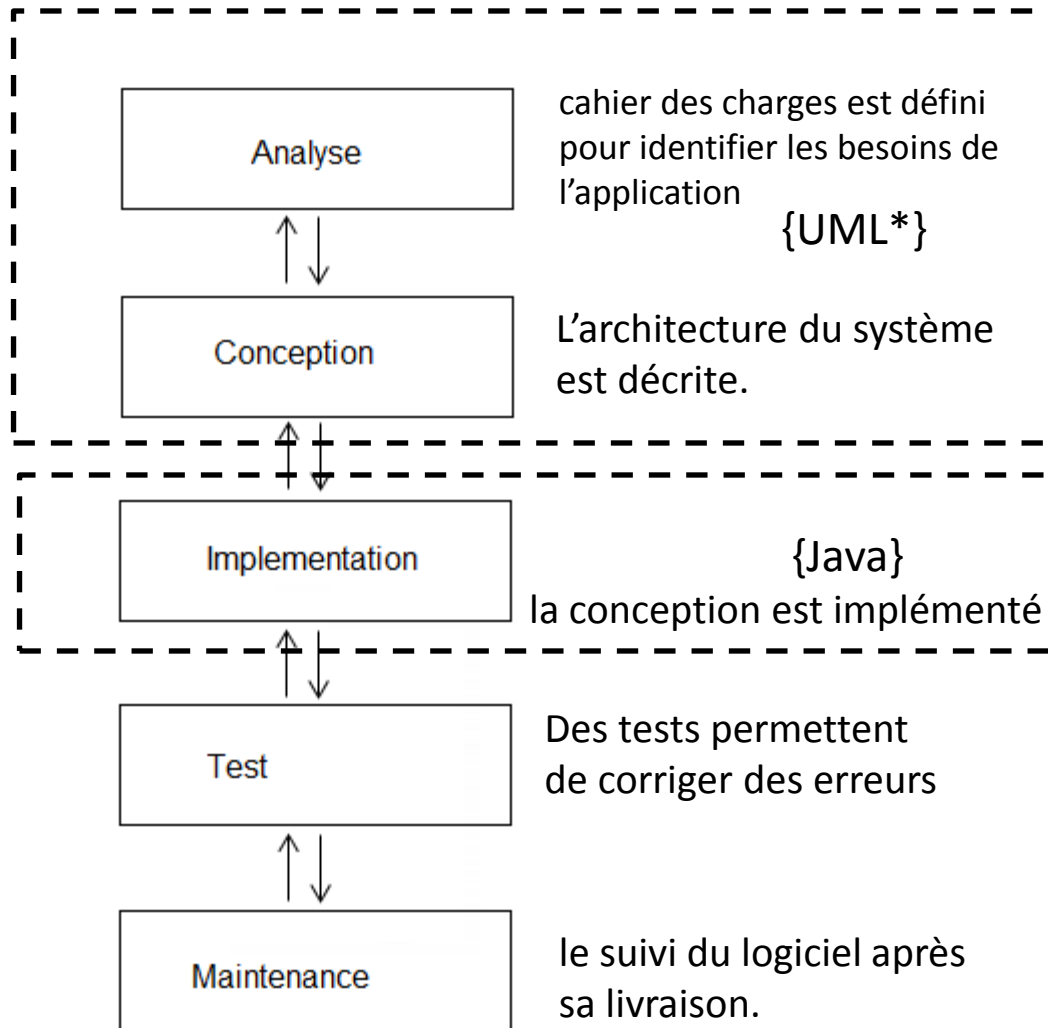
Rigoureux



Adaptable

- ✓ on peut écrire des portions de code génériques - utilisables par plusieurs applications.
- ✓ la plupart des erreurs se produisent à la compilation et non à l'exécution
- ✓ un même programme compilé peut s'exécuter sur différents environnements.
- ✓ Le logiciel doit évoluer en réponse aux conditions changeantes dans les systèmes différents .

# Modèle en cascade pour le développement logiciel

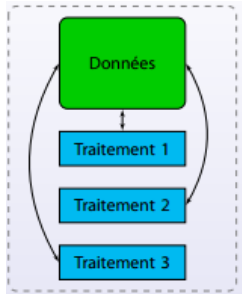


\*on utilise un langage de modélisation : UML – le diagramme des classes

# Langages procéduraux et fonctionnels vs orientés-objet

Un programme est composé de :

## Langages procéduraux et fonctionnels

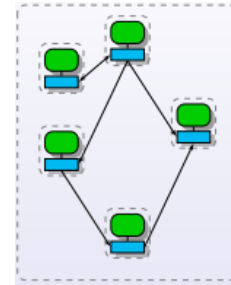


plusieurs **procédures/fonctions** :

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

Exemples: C, Fortran, etc.

## Langages orientés-objet



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.



# Concepts de programmation orientée-objet

La programmation orienté objet (POO) est une autre manière de programmer.



**Pour coder objet, il faut penser objet.**

- ☐ Objet
- ☐ Classe
- ☐ Encapsulation
- ☐ Heritage (+Types d' heritage)
- ☐ Polymorphisme

# Objet

❖ Approche procédurale vs orientée-objet

## Procédurale

“Que doit faire  
mon programme ?”

## Orientée-objet

“De quoi doit être  
composé mon programme ?”



Le livre de Java  
premier langage



Programmer  
en Java



Le canard Enchaîne



Le monde



Directeur de banque



conseiller clientèle  
banque

- ☐ un objet représente:
  - une idée, un concept, une entité
- ☐ un objet possède une structure :  
ce sont ses propriétés, ou attributs  
→ les propriétés décrivent :  
ses caractéristiques (variables /valeurs)
- ☐ un objet a un comportement :  
ce sont ses méthodes → décrivent une  
action, un état, donc un verbe
- ☐ pour concevoir des objets, on utilise  
un langage de modélisation : UML –  
le diagramme des classes

# 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

Classe Livre



Le livre de Java  
premier langage



Programmer  
en Java

Classe Journal



Le canard Enchaîné



Le monde

Classe Employe



Directeur de banque



conseiller clientèle  
banque

Types

retourne  
le solde

traitement  
sur le solde

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

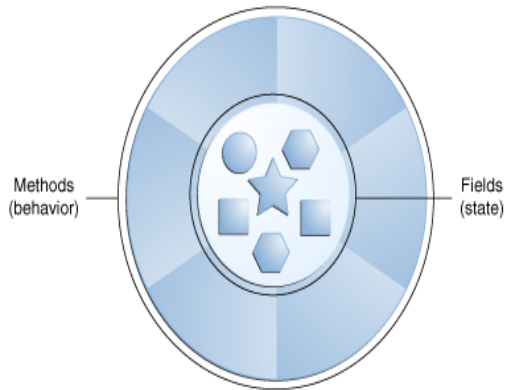
```
    double getSolde() {  
        return solde;
```

méthode  
(fonction)

```
    void credite(double val) {  
        solde = solde + val;
```

méthode  
(procédure)

# Exemple

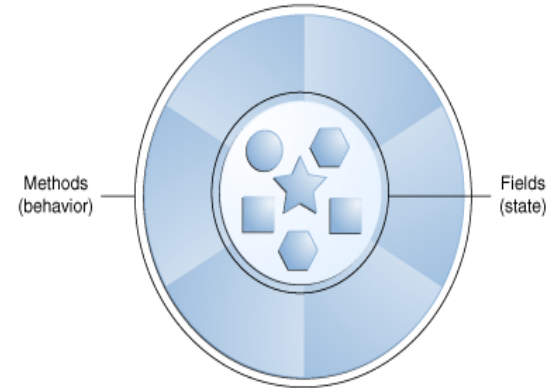


A software object.

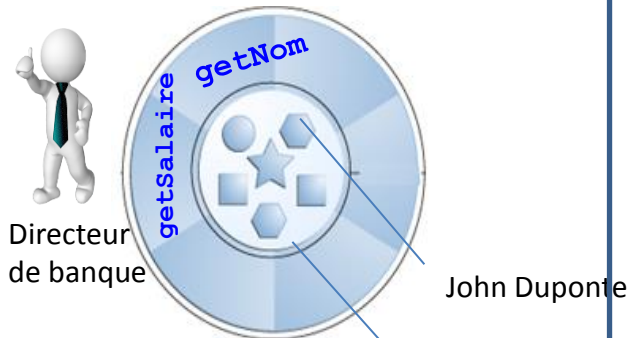
## Exemple

```
class Employe {
    String   name;
    double   salaire;
    double   getSalaire(){
        return salaire;
    }
}
```

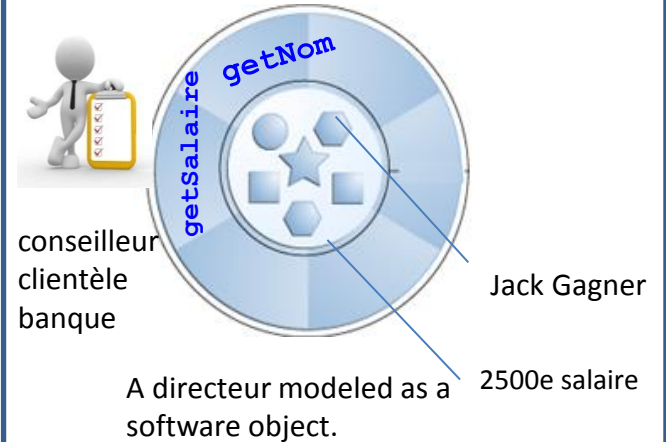
```
String getName() {
    return name;
}
```



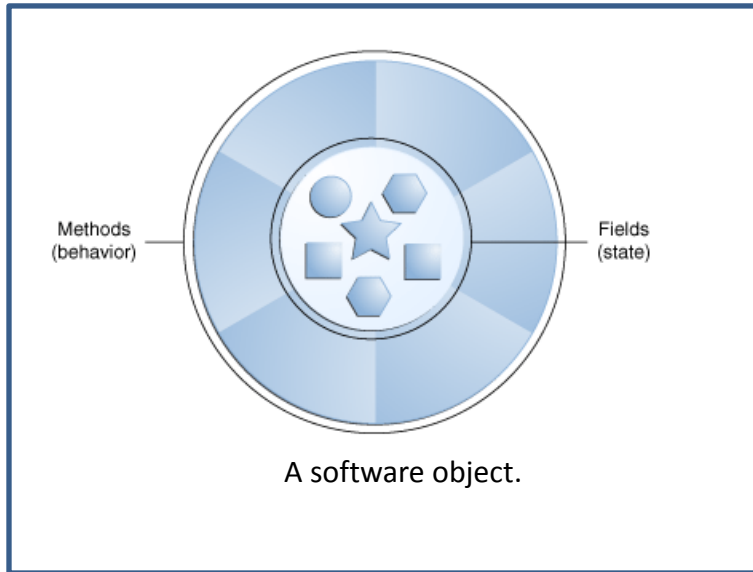
A software object.



A directeur modeled as a software object.



A directeur modeled as a software object.



## Classe Livre



Le livre de Java  
premier langage



Programmer  
en Java

## Exemple

```
class Livre {  
    String name;  
    double prix;  
    double getPrix() {  
        return prix;  
    }  
  
    String getName() {  
        return name;  
    }  
}
```

How the two books can be  
modeled as a  
software object?

# Encapsulation

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.

→ **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).

# Héritage (Inheritance)

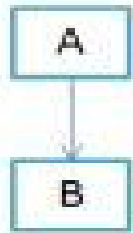
Le but de l'héritage est de regrouper les propriétés et les méthodes communes à plusieurs classes dans une nouvelle classe (la classe parente) puis d'hériter de cette classe.

→ **généralisation**

Les classes héritées forment une hiérarchie descendante, au sommet de laquelle se situe la classe de base (superclasse).

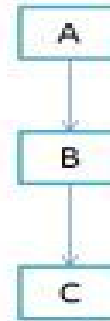
On appelle également ***la classe héritée*** la sous-classe et ***la classe parente*** la super-classe.

# Types d'héritage en Java



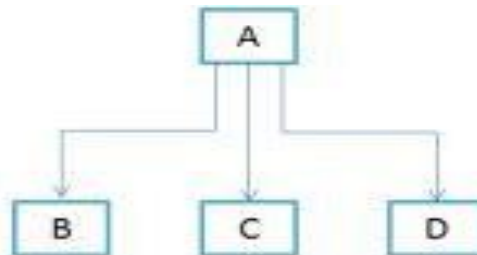
- ✓ A est la super-classe
- ✓ B la sous-classe de A

(a) Single Inheritance



- ✓ A est la super-classe
- ✓ B la sous-classe de A
- ✓ C est la sous-classe de B

(d) Multilevel Inheritance



(c) Hierarchical Inheritance

- ✓ plusieurs sous-classes ont la même classe parent (super- classe)



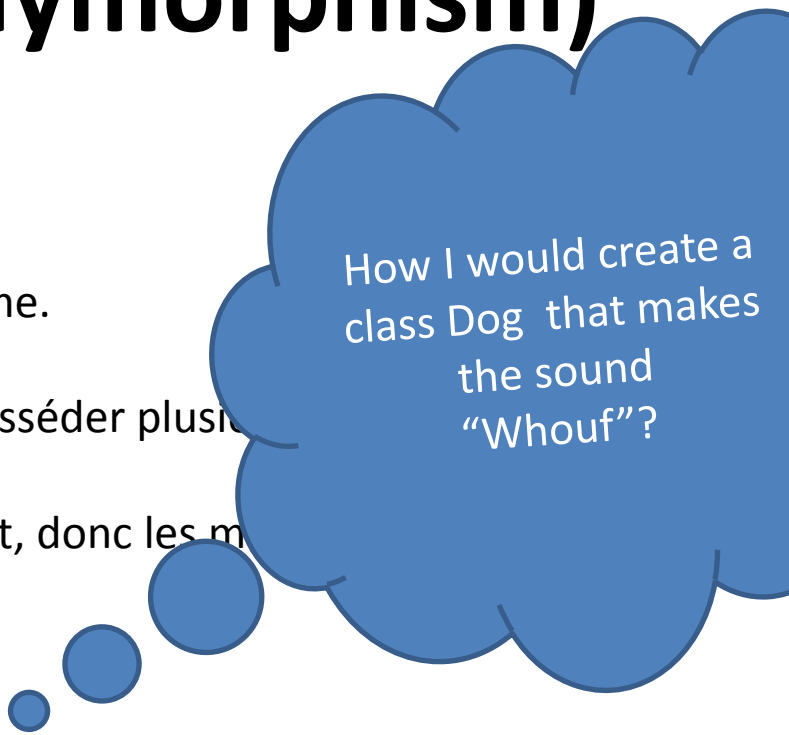
# Polymorphisme (Polymorphism)

**poly** comme plusieurs + **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.



How I would create a class Dog that makes the sound "Whouf"?

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

```
public class Cat extends Animal{  
    @Override  
    public void sound(){  
        System.out.println("Meow");  
    }  
    public static void main(String args[]){  
  
        Animal obj = new Cat();  
        obj.sound();  
    }  
}
```



<https://github.com/akatsouraki/Promo-P20.git>