

UNIVERSITE DE CORSE

2023-2024

Licence SPI 2ème année  
Parcours Informatique

## UE Programmation Orientée Objet

Introduction

CH 1 - Paradigme Orienté Objet

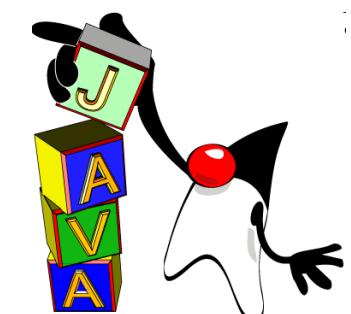
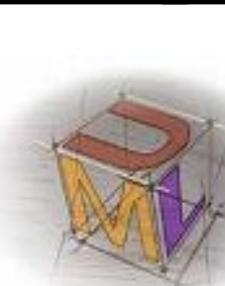


Evelyne VITTORI

[vittori\\_e@univ-corse.fr](mailto:vittori_e@univ-corse.fr)

Paul PINA-GHERARDI

[PINA-GHERARDI\\_p@univ-corse.fr](mailto:PINA-GHERARDI_p@univ-corse.fr)



# Programmation Orientée objet

## Plan du Cours

### CH1 – PARADIGME ORIENTE OBJET

1.1 - Notion de Paradigme Orienté Objet

1.2 - Introduction au langage Java

1.3 - Modélisation Objet et UML



CH2 – OBJETS et CLASSES

CH3 – COMMUNICATION ENTRE OBJETS

CH4 – HERITAGE et POLYMORPHISME

# 1.1 – Notion de Paradigme Orienté Objet



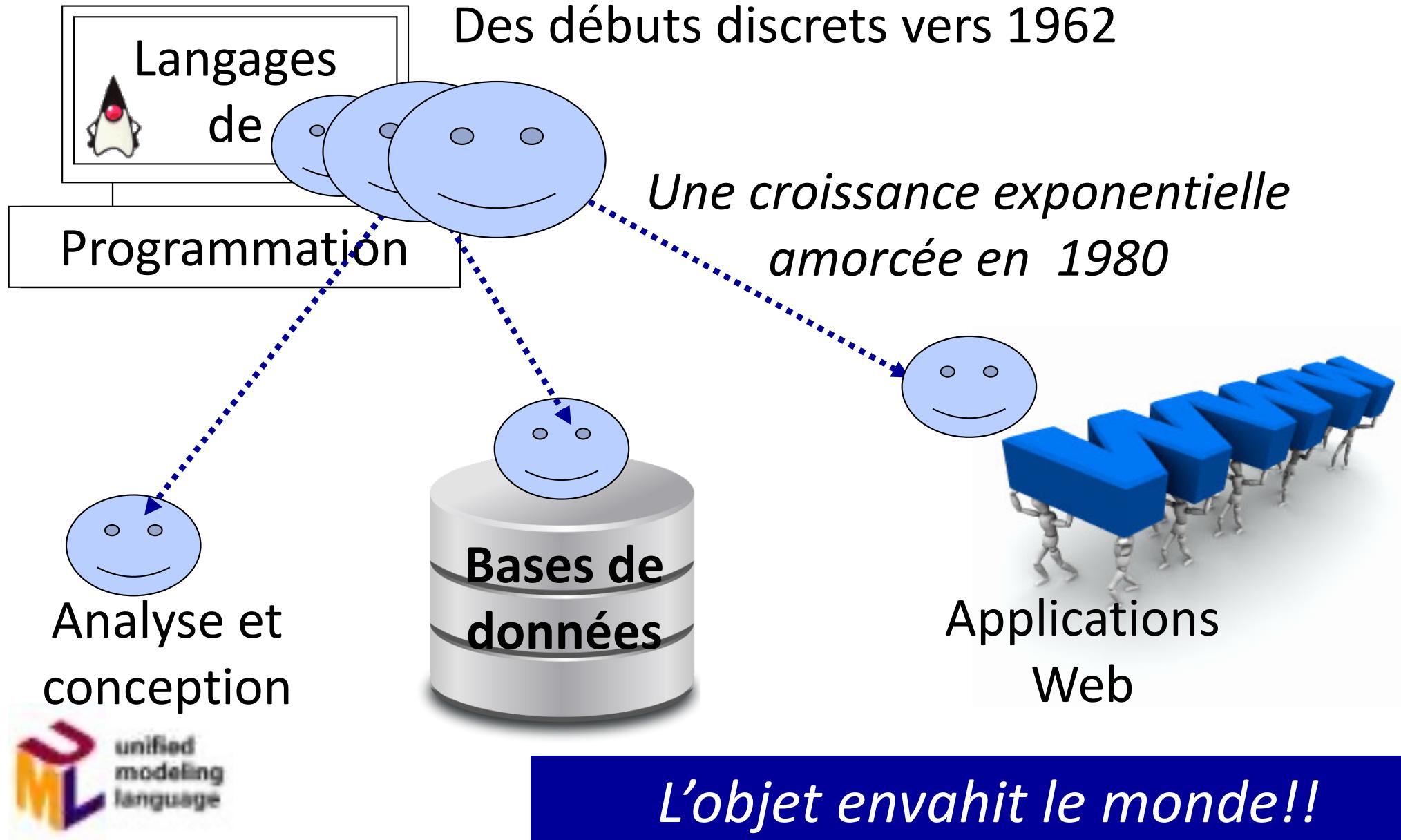
## 1. L'odyssée de l' «Objet »

- Un peu d'histoire
- Origines de la POO
- Principes de base
- Langages orientés objets



## 2. Une nouvelle manière de voir le monde

# L'odyssée de l'objet



# Aujourd’hui, les objets sont partout!!

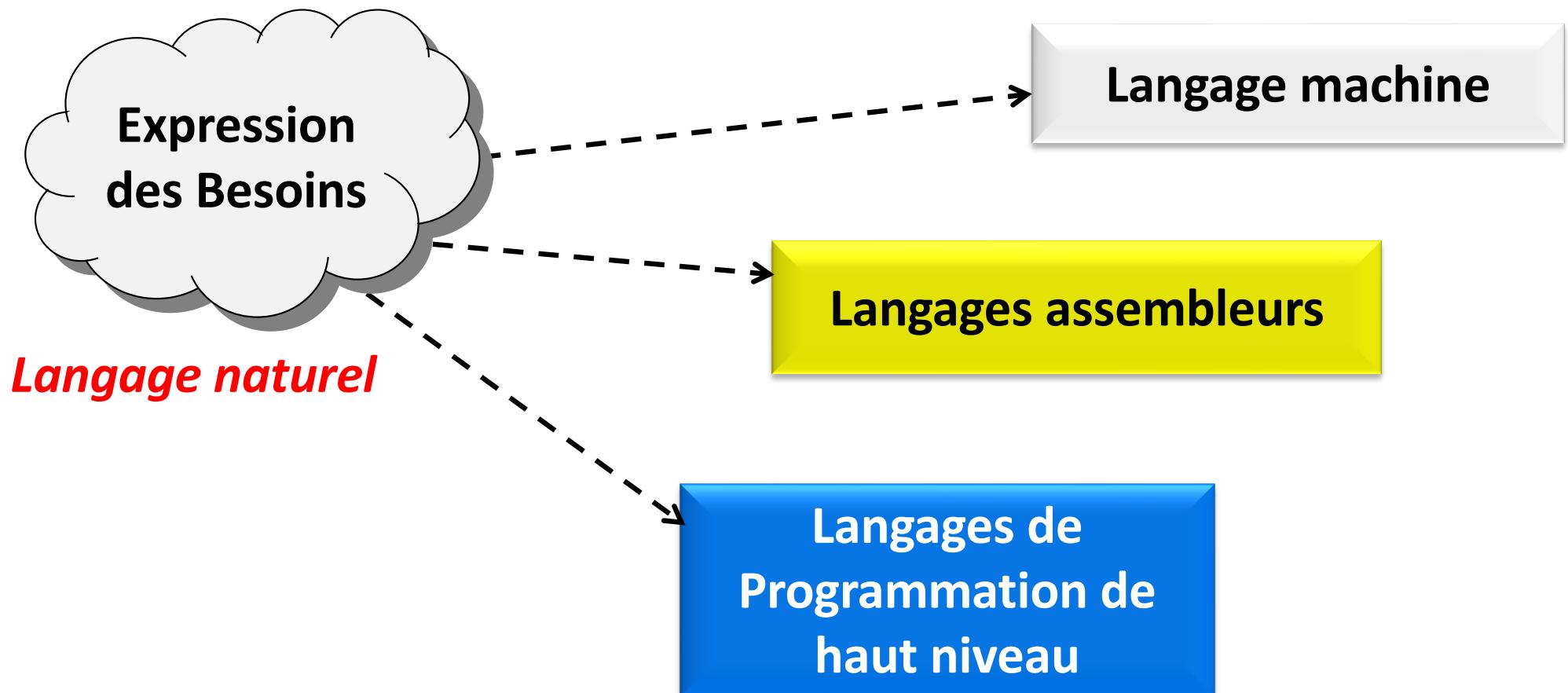
## On parle de Paradigme Orienté Objets

Paradigme

*"Modèle théorique de pensée qui oriente la recherche et la réflexion scientifiques"* (Larousse)

# Un peu d'histoire ...

## Evolution des langages de programmation

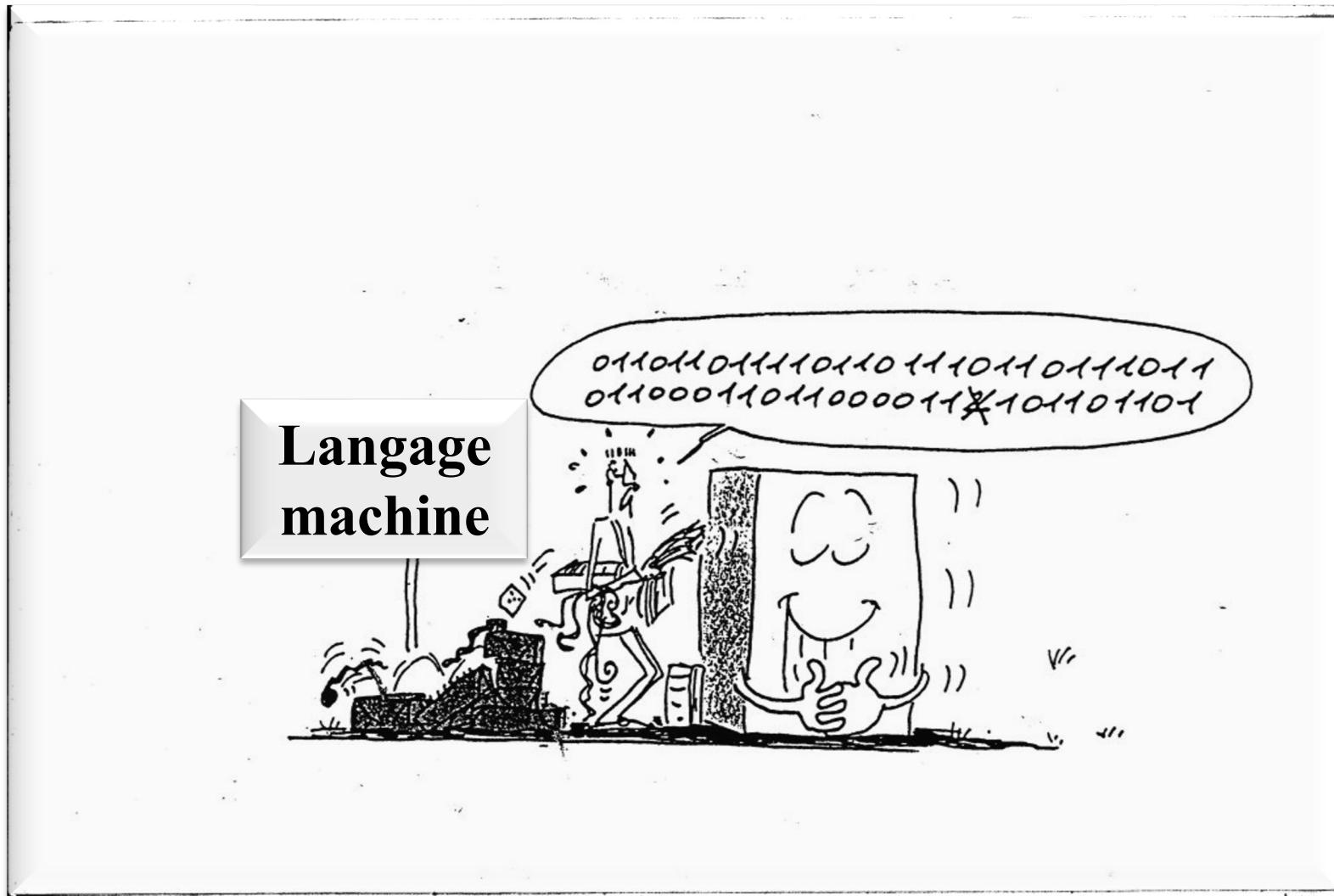


1ère tendance: «se libérer des contraintes matérielles»



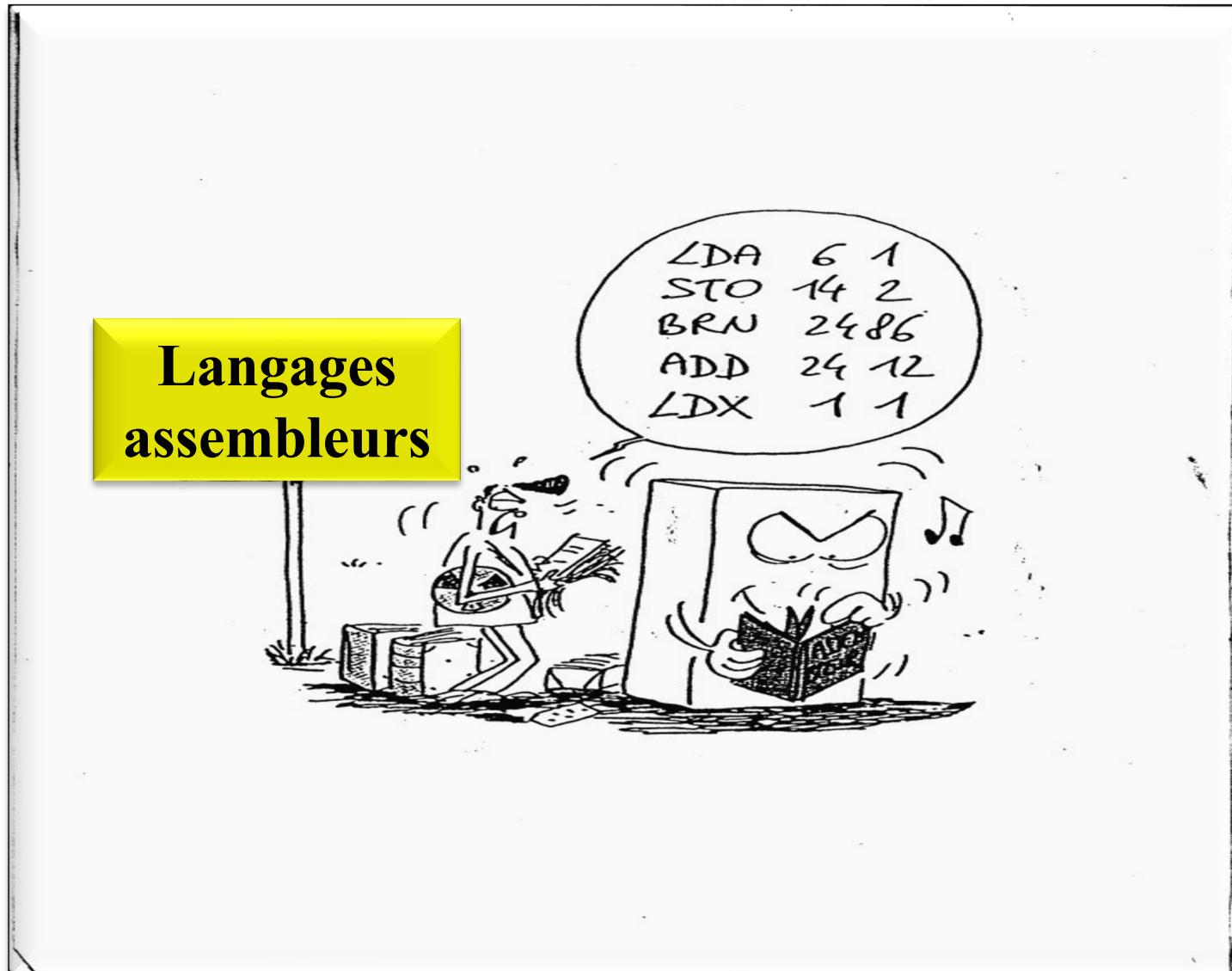
# Un peu d'histoire ...

## Evolution des langages de programmation



# Un peu d'histoire ...

## Evolution des langages de programmation



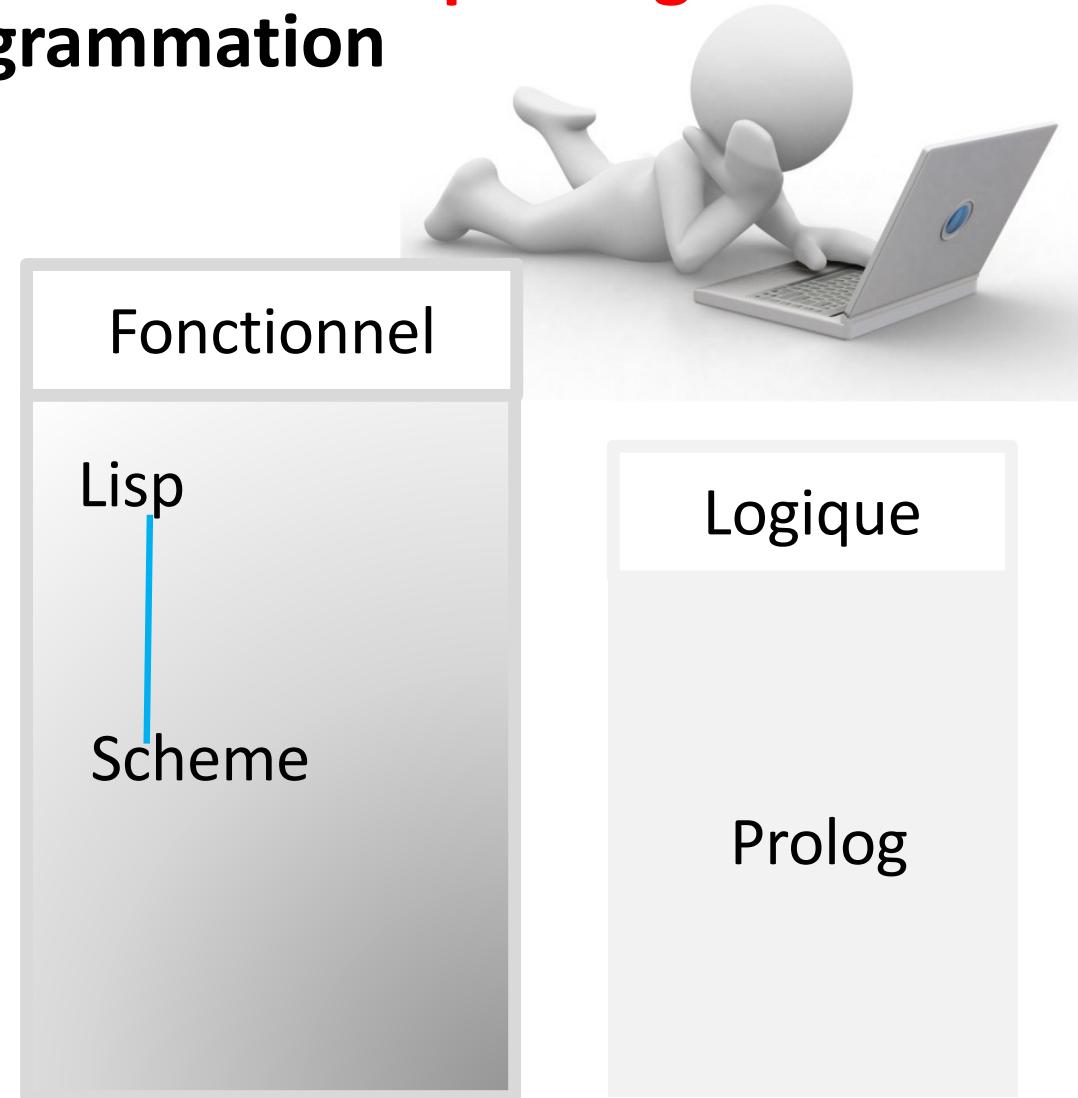
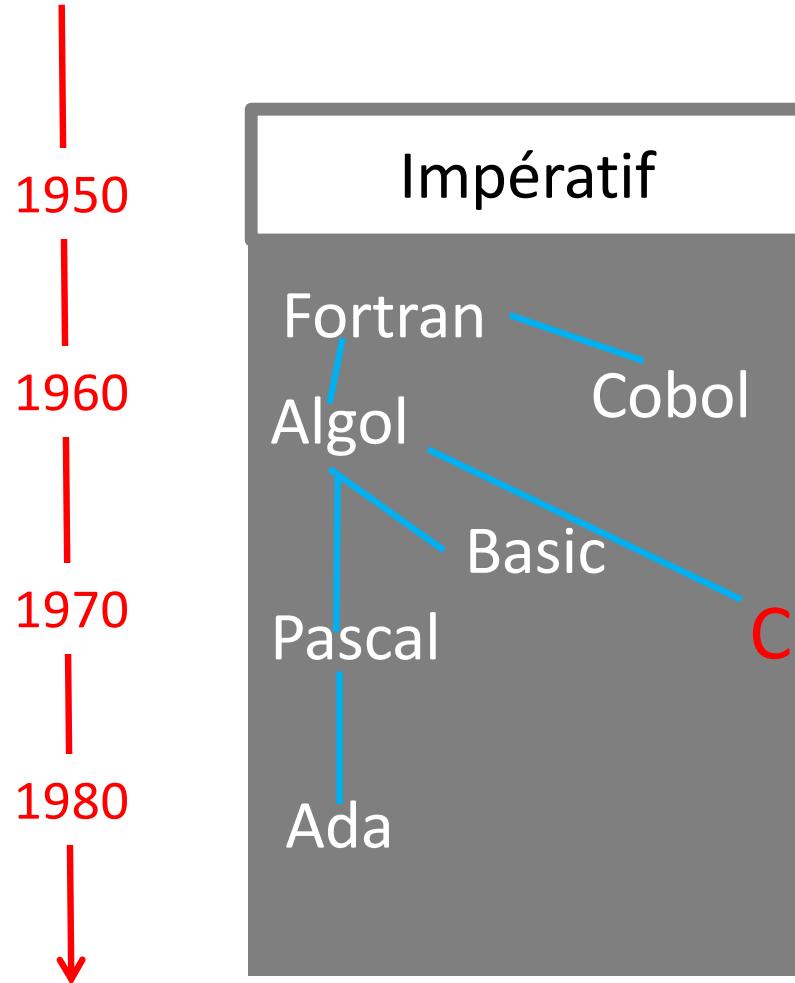
# Un peu d'histoire ...

## Evolution des langages de programmation

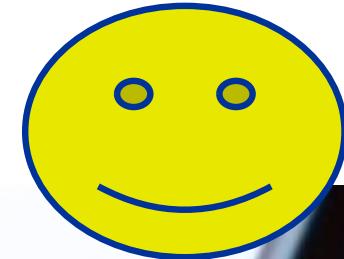


# Un peu d'histoire ...

Depuis les années 50 : Différents **paradigmes** de programmation



# Origines de la POO



Programmation  
structurée (impérative)

Algorithmique de base

variables typées

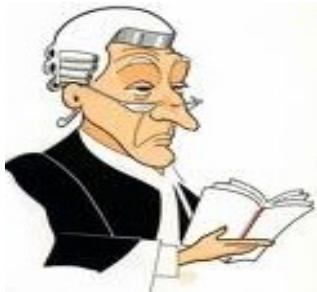
Structures de contrôle

Procédures et Fonctions

Types Abstraits  
de Données  
Structure de données  
+ opérations associées

Programmation  
Orientée Objet

Modules  
Unité de Programme  
+ Interface



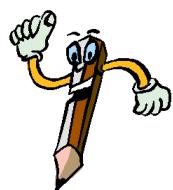
# Principes de base de la POO

## Abstraction



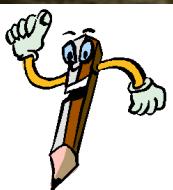
*Objets et Classes,  
Mécanisme d'instanciation*

## Encapsulation



*Attributs, méthodes, visibilité*

## Hiérarchie d'héritage



C'est l'objectif de ce  
cours!!

## Polymorphisme

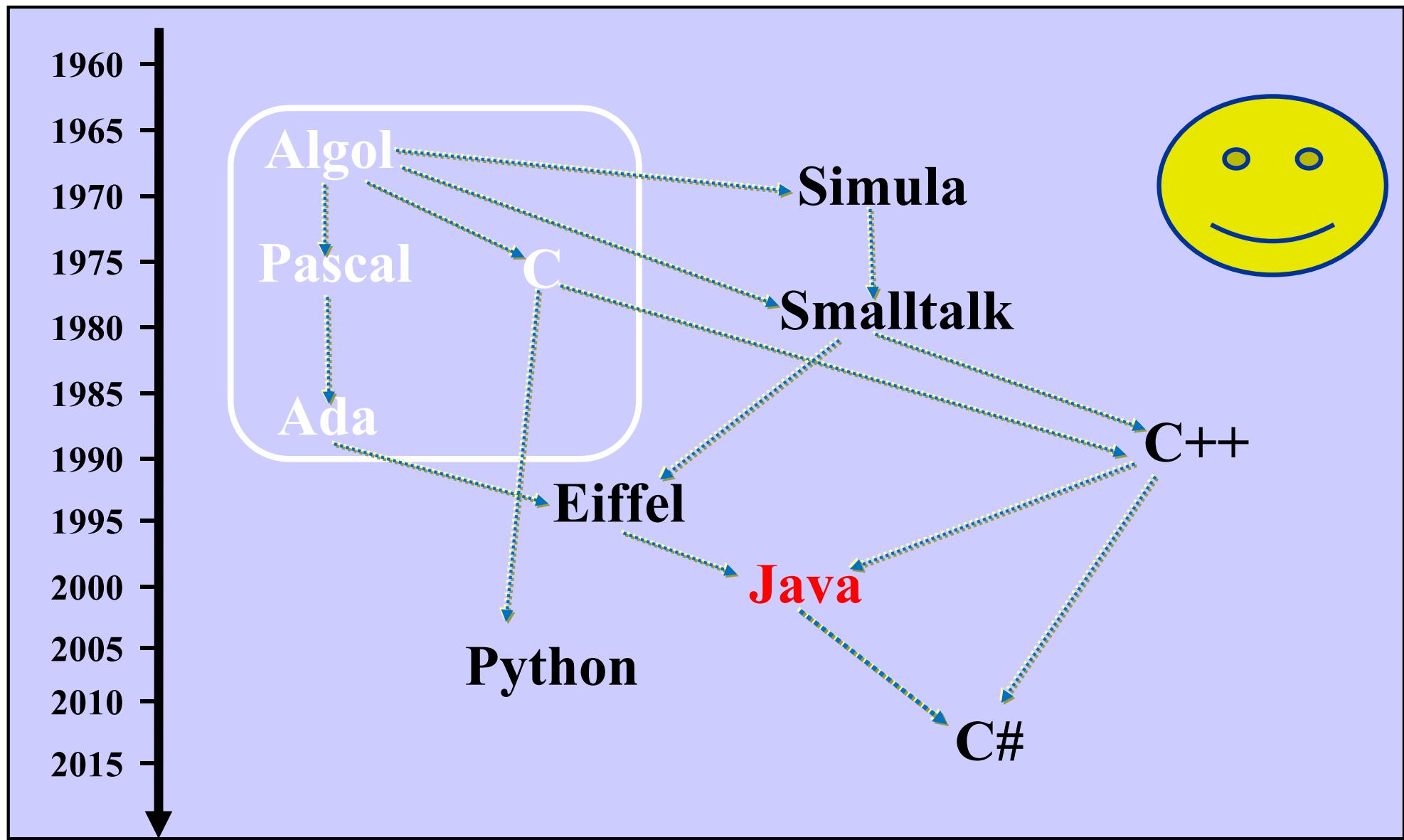


# Classification des LOO

- Langages « totalement » OO
  - Tout est objet! Smalltalk, Java, C#
  - Prise en compte de tous les principes de la POO
- Langages OO « hybrides » C++, Python, php
  - Langages multi-paradigmes
- Langages « partiellement » OO JavaScript
  - Certains principes OO ne sont pas pris en compte

Même dans un langage totalement OO, il est toujours possible de ne pas véritablement programmer OO!!

# Les langages Orientés Objets



# Les prémisses ....: En norvège, il y a plus de 40 ans!!!

## Simula 67 (1962-67)

Ole-Johan Dahl, Kristan Nygaard

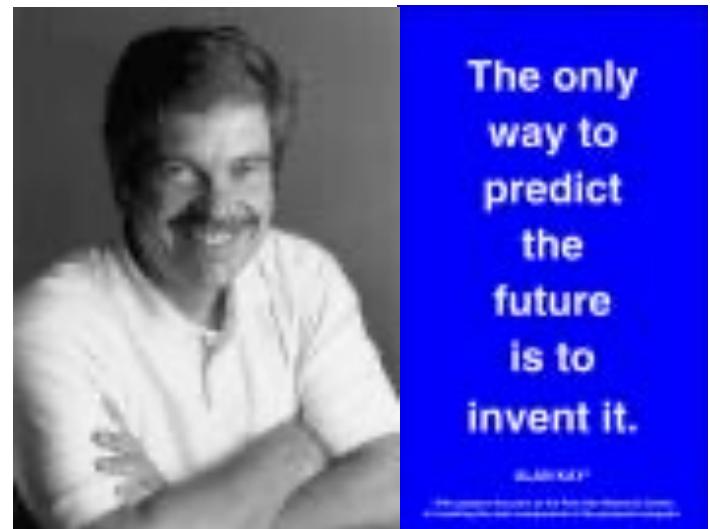


- 1962: Simulation de systèmes complexes  
Ex: *Distribution statistique des durées de traitement des passagers dans un aéroport*
- Langage universel (Simula 67)
- Notions de classes, d'instances, de fonctions (méthodes) associées aux objets, de hiérarchie d'héritage.

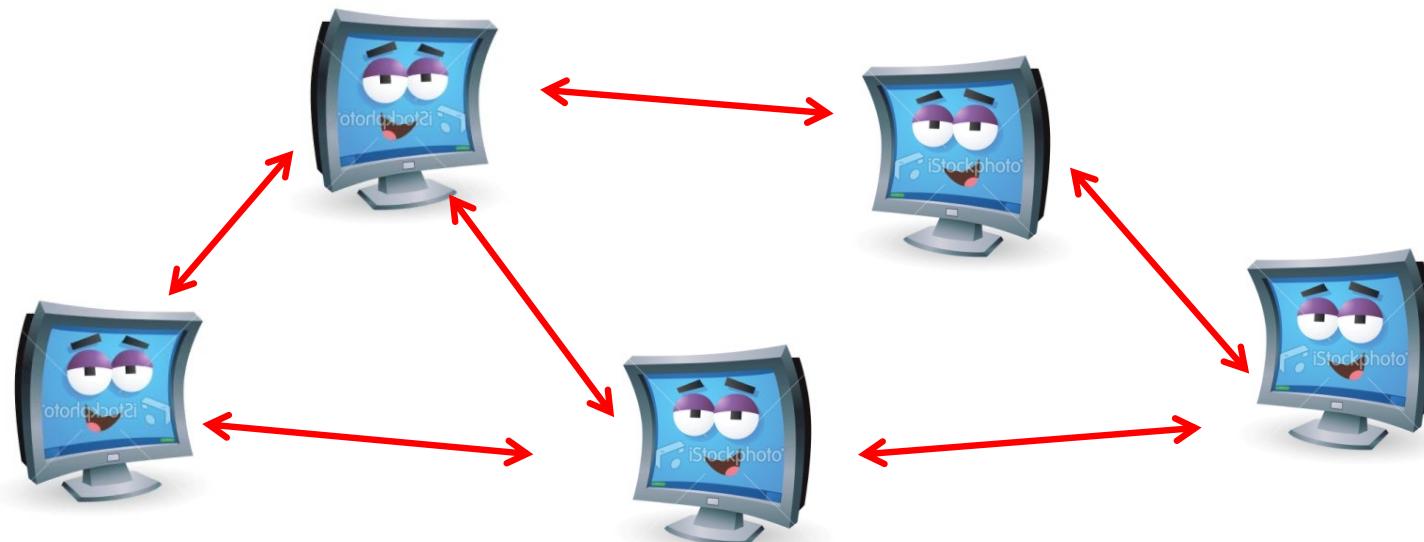
# La recherche à Xerox Parc ...

**Smalltalk - 1972**

*Alan Kay*



*“Chaque objet est comme un petit ordinateur qui interagirait avec d'autres ordinateurs”*



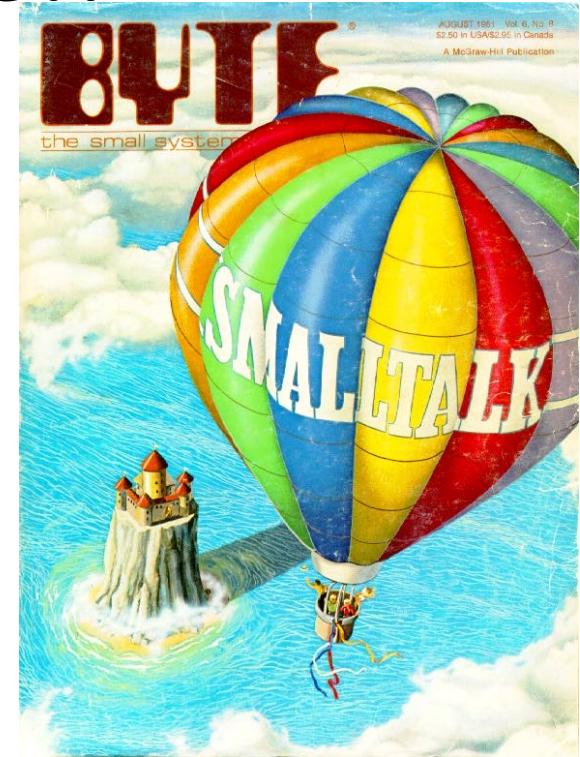
# SMALLTALK s'envole!!

## Smalltalk 80- 1980

(version industrielle) Alan Kay

Une référence en POO:

- langage totalement orienté objet
- code généré **portable** (bytecode)
- pas d'allocation et désallocation explicite de la mémoire
- maîtrise difficile malgré une syntaxe simple
- Programmes nécessitant des ressources systèmes importantes

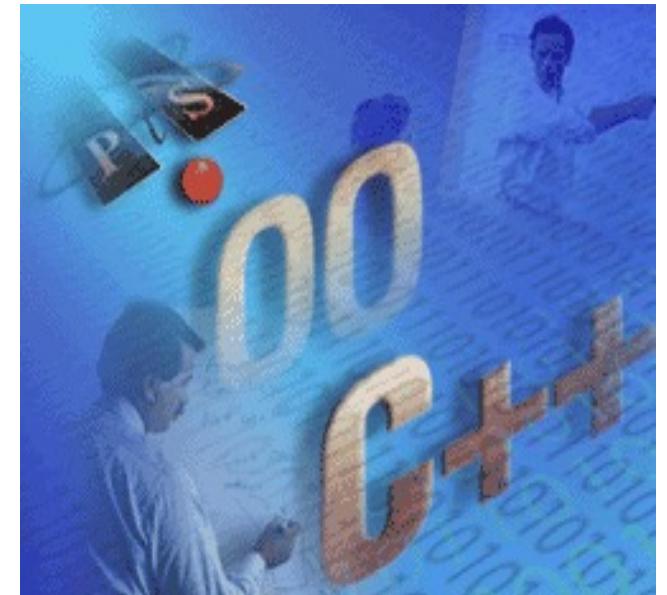


# C++: le LOO le plus « pro »

## C++ (1981-1986)

*Bjarne Stroustrup (ATT&T Bell)*

- Extension orientée objet de C
- Programmes rapides et efficaces
- Programmation « bas niveau » possible
  - Utilisation de pointeurs
  - Allocation et désallocation explicite de la mémoire



«C with classes»

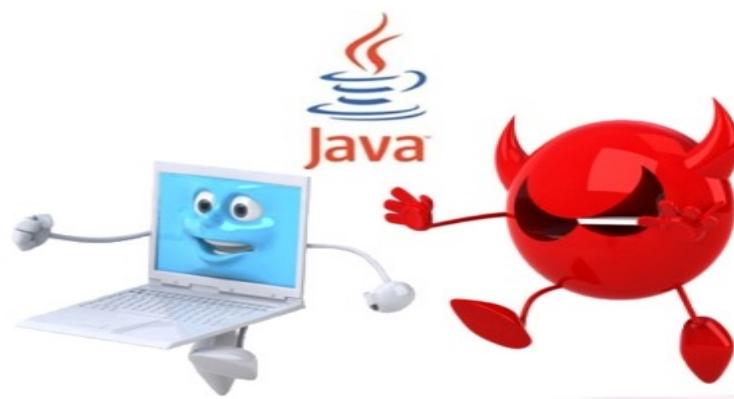
# Et le WEB arrive ...

## JAVA – (1995)

*James Gosling (SUN)*



- Langage entièrement orienté objet
- Technologie portable (byte-code) adaptée à Internet
- Langage **simple** et fiable
- Syntaxe familière proche du C/C++ mais
  - Sans pointeur
  - Sans allocation et désallocation explicite de la mémoire



ORACLE®

Rachat en  
2009

# La riposte de Microsoft

**C# - 2000(C-sharp),**

*Anders Hejlsberg (Microsoft)*

- Langage de base de la plateforme .NET
- Puissance du C++
- Facilité de développement du Visual Basic
- Élégance de Java
  - Compilé et interprété(**Common Intermediate Language (CIL)** )

Langage de base de la plateforme  
de développement Web .Net





# Python

## Python (1990- Guido van Rossum)

- Objectifs : Simplicité et puissance
- Programmation multi-paradigme (fonctionnel, objet, ...)
- Lisibilité du code
- Développement rapide d'application:  
Langage interprété
- Facilité de fonctionnement avec d'autres langages



# Autres langages spécialisés web

## PHP (1994)

« Personal Home Pages » (1<sup>ère</sup> signification)

« PHP *Hypertext Preprocessor* »

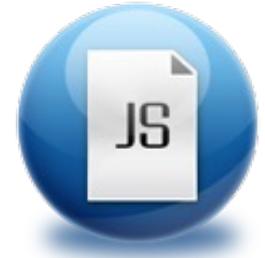
- Langage dédié aux applications Web
- Permet de créer des pages web dynamiques liées à des bases de données
- Scripts exécutés côté serveur (résultats intégrés dans des pages HTML).
- Orienté objet uniquement depuis php5 (2007)

# Autres langages spécialisés Web

## JavaScript

- Scripts intégrés dans des pages HTML et exécutés côté client
- Très « Partiellement » orienté objet

```
<!DOCTYPE html>
<html lang="en">
  <head> <title>Some Page</title>
        <script type="text/javascript">
          alert("Hello World!");
        </script>
  </head>
  <body>
    <p>The content of the web page.</p>
  </body>
</html>
```



A ne pas confondre avec Java !!

# 1.1 – Notion de Paradigme Orienté Objet

## 1. L'odyssée de l' «Objet »

- Un peu d'histoire
- Origines de la POO
- Principes de base
- Principaux langages orientés objets



## 2. Une nouvelle manière de voir le monde

# Approches de décomposition de problèmes



*Approches  
méthodiques de  
décomposition*

**Génie Logiciel**  
“Art de concevoir, réaliser et faire évoluer avec des moyens et des délais raisonnables des programmes de qualité”



# Du fonctionnel à l'objet

## Evolution des démarches



Décomposition fonctionnelle  
(1970 ...)



Décomposition orientée objet

# Approche Fonctionnelle

- Démarche **centrée sur les traitements**
- Analyse **descendante**
  - Décomposition d'un programme en sous-programmes moins complexes ...
  - Hiérarchie de fonctions
- Approche associée aux langages de **programmation structurés** (ou procéduraux)

# Du fonctionnel à l'objet: un exemple

## Programme de pilotage de machines à café



Machine à  
Café moulu



Machine à  
Café en grains



Machine à  
Café soluble<sup>28</sup>

# Approche fonctionnelle: Analyse

- **Que doit faire** le programme?
  - Faire du café
  - Provoquer l'éjection des gobelets
  - ...
- **Comment** réaliser ces actions?
  - De 3 manières différentes selon le type de machine

# Approche fonctionnelle: Programmation

- Implémentation de trois fonctions (une pour chaque type de machine)

*numéro de la  
machine*

```
def faireLeCafeGrain(numMachine : int)-> int :
```

```
def faireLeCafeMoulu(numMachine : int)-> int :
```

```
def faireLeCafeSoluble(numMachine : int)-> int :
```

# Approche fonctionnelle: Programmation

**tabMac = liste d'entiers**

*Tableau indiquant le type  
de chaque machine pilotée*

⋮

0	1	2	3	4
CAFEMOULU	CAFESOLUBLE	CAFESOLUBLE	CAFEGRAIN	CAFEMOULU

**N = len(tabType)**

*Entier Nombre de machines*

**Constantes**

CAFEGRAIN=1;

CAFESOLUBLE=2;

CAFEMOULU=3;

**La machine n°3 est de type 1 (CAFEGRAIN)**

# Approche fonctionnelle: Programmation

```
# Initialisation de la liste tabTyp
tabMac = [CAFESOLUBLE, CAFEMOULU, CAFEMOULU, CAFEGRAIN, CAFESOLUBLE]
for numMachine in range(len(tabMac)) :
    if tabMac[numMachine] == CAFEGRAIN:
        res = faireLeCafeGrain(numMachine)
    elif tabMac[numMachine] == CAFEMOULU:
        res = faireLeCafeMoulu(numMachine)
    elif tabMac[numMachine] == CAFESOLUBLE:
        res = faireLeCafeSoluble(numMachine)
    else:
        res = 9 # Erreur, type inconnu
    if res == 0:
        print("Le cafe est pret")
    else:
        print("Machine en panne")
```

Programme  
gestionMachines

# Approche fonctionnelle: Limites



nouveau type  
de machine à café  
Machine à Capsules



Comment  
modifier le  
programme  
pour prendre  
en compte ce  
nouveau type  
de machine

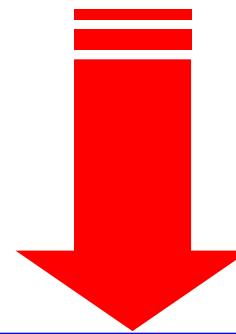
# Approche fonctionnelle: Limites



nouveau type  
de machine à café



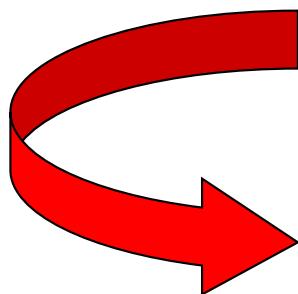
Modification du programme existant:  
✓ nouvelle condition  
✓ Autres ....



- Recherche fastidieuse dans le code
- Risques d'introduction d'erreurs

# Approche fonctionnelle: Limites

- Code peu robuste
- Représentation éloignée de notre manière de penser



- Difficulté de maintenance
- Evolutivité restreinte

# Du fonctionnel à l'objet

## Evolution des démarches

Décomposition fonctionnelle  
(1970 ...)



Décomposition orientée objet  
(1990 ...)



# Approche Orientée Objets

- Démarche centrée sur la **recherche des « Objets »** (données) à travers leurs propriétés caractéristiques et leur comportement
- Un programme est constitué d'objets collaborant entre eux par envois de messages
- Objectif initial: Gérer la complexité

Réutilisabilité et Extensibilité

# Approche orientée Objet: Analyse

- Que doit manipuler le programme?  
*Des machines à café !!*
- Qu'est-ce qui caractérise une machine à café?

*Réservoir d'eau*  
*Réservoir de café*  
*Réservoir de sucre*  
*Réservoir de gobelets*  
*Résistance chauffante*  
*Bouton sélecteur de la dose de sucre*  
*Bouton Marche/arrêt*  
...

# Approche orientée Objet: Analyse

- Que peut-on demander à une machine à café?

*De faire du café*

*Des informations sur son état*

- *quantité d'eau dans le réservoir*
- *état du bouton de sélection de la dose de sucre*
- *...*

*D'éjecter les gobelets*

*...*

# Approche Orientée Objet: Programmation

- Mais il existe différentes sortes de machines à café!

*Oui, mais toutes les machines à café ont des points communs: c'est ce qui doit nous intéresser pour les piloter.*

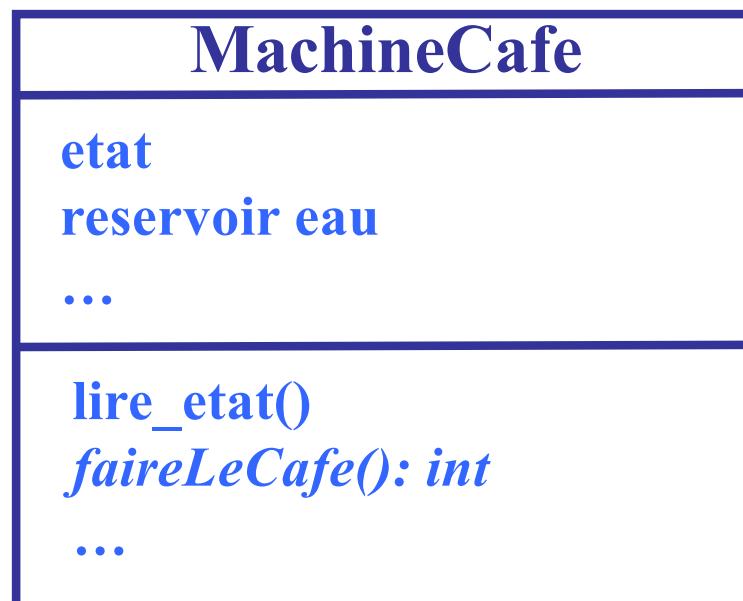
*Ensuite, on représente les spécificités de chaque type de machine à café.*

# Approche Orientée Objet: Programmation

- Représentation des machines à café

*Machine à café (points communs)* -

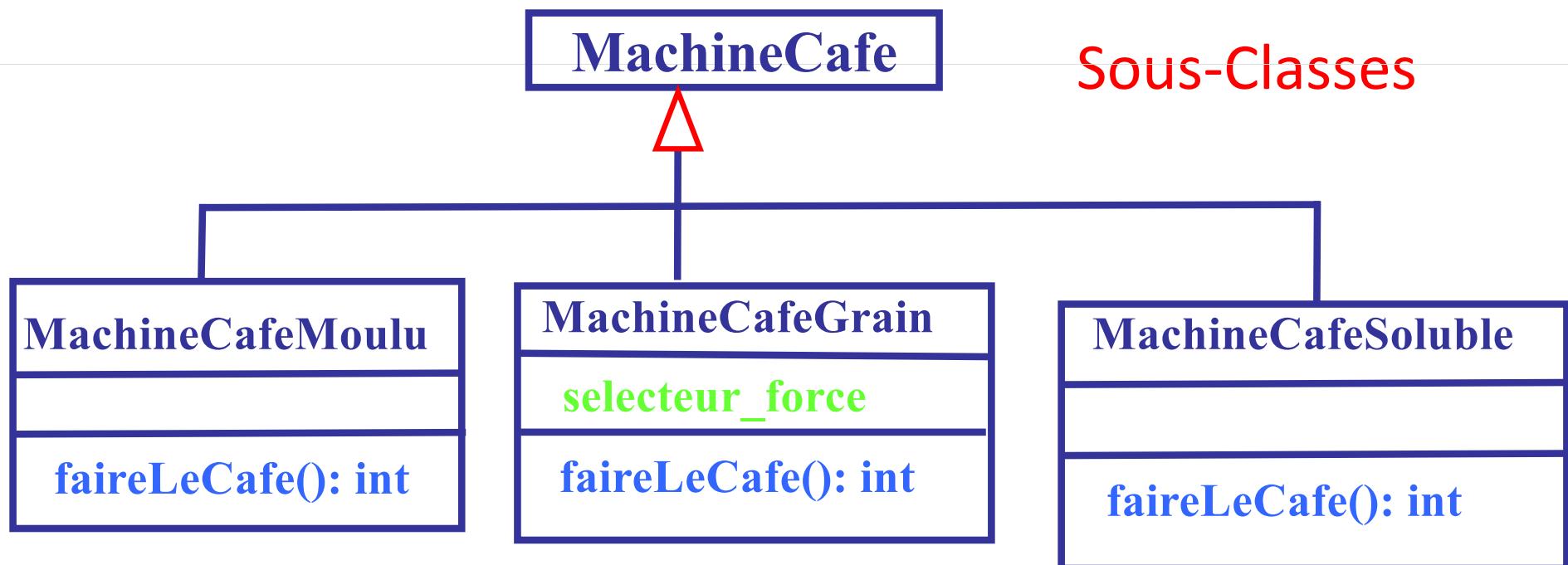
Classe MachineCafe



# Approche Orientée Objet: Programmation

- Représentation des machines à café

*machine à café (cas particuliers)*



# Approche Orientée Objet: Programmation des classes

```
abstract class MachineCafe {  
    abstract int faireLeCafe();  
    //Méthode abstraite  
}  
  
class MachineCafeGrain extends MachineCafe {  
    int faireLeCafe() {  
        //Simulation de faire le café à partir de grains  
        return ...  
    }  
}  
  
class MachineCafeMoulu extends MachineCafe {  
    int faireLeCafe() {  
        //Simulation de faire le café moulu  
        return ...  
    }  
}  
  
class MachineCafeSoluble extends MachineCafe {  
    int faireLeCafe() {  
        //Simulation de faire le café soluble  
        return ...  
    }  
}
```

version Java

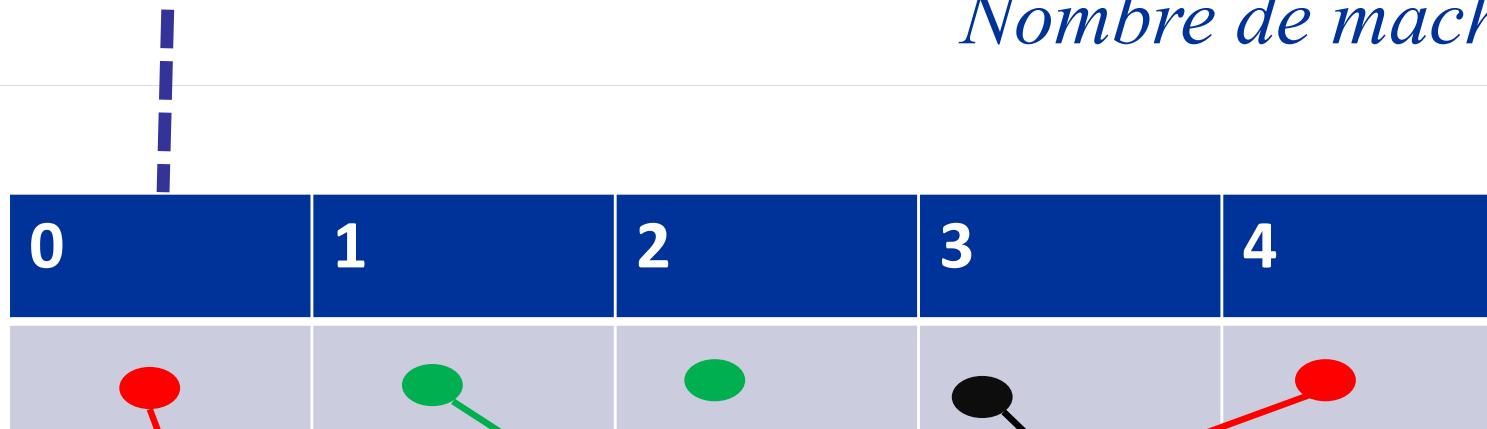
# Approche Orientée Objet: Programmation

**MachineCafe tabMac [N] ;**

*Tableau de machines*

**N = entier**

*Nombre de machines*



**Objet de la classe  
MachineCafeMoulu**

**Objet de la classe  
MachineCafeSoluble**

**Objet de la classe  
MachineCafeGrain**

# Approche Orientée Objet: Programmation

- Programme gestionMachines

version java

```
for (MachineCafe machine: tabMac) {  
  
    res= machine.faireLeCafe();  
  
    if (res == 0)  
        System.out.println("le cafe est pret\n");  
    else  
        System.out.println("Machine en panne\n");  
}
```

# Approche Orientée Objet:

## Programmation

### ■ Définition des classes

version python

```
class MachineCafe:  
    def faireLeCafe(self): pass  
        #Méthode générique redéfinie dans les sous-classes  
  
class MachineCafeGrain(MachineCafe):  
    def faireLeCafe(self): return 0  
        #Simulation de faire le café à partir de grains  
class MachineCafeMoulu(MachineCafe):  
    def faireLeCafe(self): return 0  
        #Simulation de faire le café moulu  
class MachineCafeSoluble(MachineCafe):  
    def faireLeCafe(self): return 0  
        #Simulation de faire le café soluble
```

# Approche Orientée Objet:

## Programmation

- Programme gestionMachines (centre de contrôle des objets)

```
# Initialisation de la liste tabTyp avec des instances des classes de machines à café
tabMac = [MachineCafeSoluble(),
MachineCafeMoulu(), MachineCafeMoulu(),
MachineCafeGrain(), MachineCafeSoluble()]

for machine in tabMac:
    res = machine.faireLeCafe()
    if res == 0:
        print("Le cafe est pret")
    else:
        print("Machine en panne")
```

version python

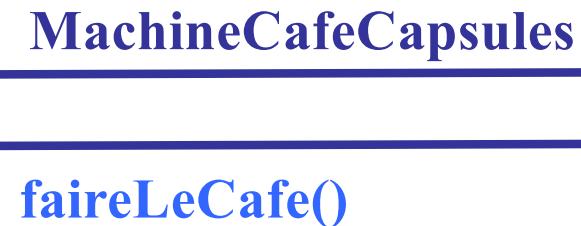
# Approche Orientée Objet: Atouts



nouveau type  
de machine à café



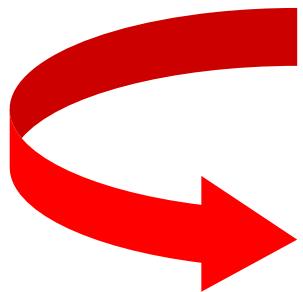
Ajout d'une nouvelle sous-classe



- Aucune modification du programme existant
- Pas de risque d'introduction d'erreurs dans le code existant

# Approche Orientée Objet: Atouts

- Code plus robuste
- Code plus clair car plus proche de la pensée humaine

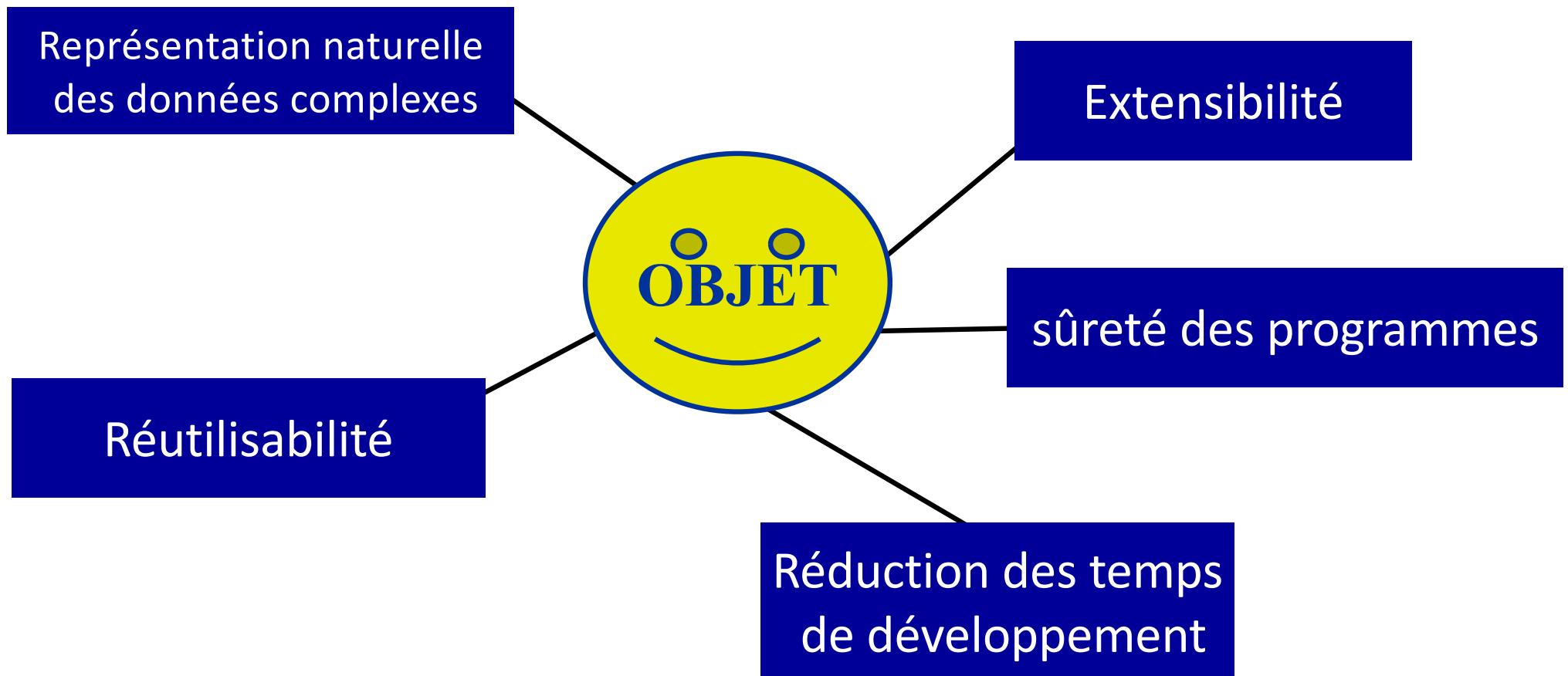


- Maintenance facilitée
- Evolutivité

# Du fonctionnel à l'objet: un exemple

## Conclusion

### Atouts de l'approche Orientée Objets



# La POO: une autre manière de structurer les programmes

- Démarche centrée sur la **recherche des « Objets »** (données) à travers leurs propriétés caractéristiques et leur comportement
- Un programme est **constitué d'objets** collaborant entre eux par envois de messages

## Programmation structurée classique

- Ensembles de **fonctions** qui s'appellent entre elles

## Programmation orienté Objets

- Ensemble de **classes** permettant de définir des objets

# Programmation Orientée objet

## Plan du Cours

### CH1 – PARADIGME ORIENTE OBJET

1.1 - Notion de Paradigme Orienté Objet

1.2 - Introduction au langage Java

1.3 - Modélisation Objet et UML



### CH2 – OBJETS et CLASSES

### CH3 – COMMUNICATION ENTRE OBJETS

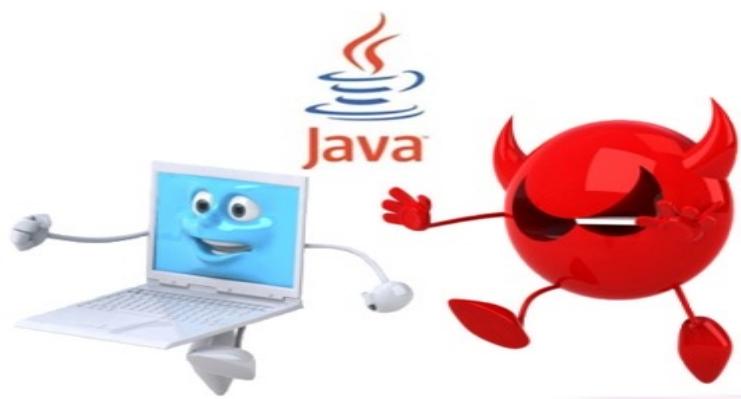
### CH4 – HERITAGE et POLYMORPHISME

# Naissance de java ...

## JAVA – (1995)

*James Gosling (SUN)*

- Langage entièrement orienté objet
- Technologie portable (byte-code) adaptée à Internet
- Langage **simple** et fiable
- Syntaxe familière proche du C/C++ mais
  - Sans pointeur
  - Sans allocation et désallocation explicite de la mémoire



ORACLE®

Rachat en  
2009

# Historique du langage JAVA

- 1995 : Premières versions de Java rendues publiques
- Une nouvelle version tous les deux ans depuis 1995
  - JDK 1.0 en 1995
  - JDK 1.1 en 1997
  - J2SE 1.2 en 1999 (Java 2, version 1.2)
  - J2SE 1.3 en 2000 (Java 2, version 1.3)
  - J2SE 1.4 en 2002
  - **J2SE 5.00 (ou 1.5) en 2004**
  - Java SE 6.00 en 2006
  - **Java SE 7.00 en 2011**
  - **Java SE 8.00 en 2014**
  - Java SE 9.00 en septembre 2017
  - Java SE 10.00 en mars 2018
  - Java SE 11 en septembre 2018
  - ....
  - **Java SE 20 en mars 2023** → Version actuelle
- Evolution très rapide et succès du langage



Une version annoncée tous les 6 mois à partir de java9!

# Présentation de JAVA

JAVA est une "plateforme"

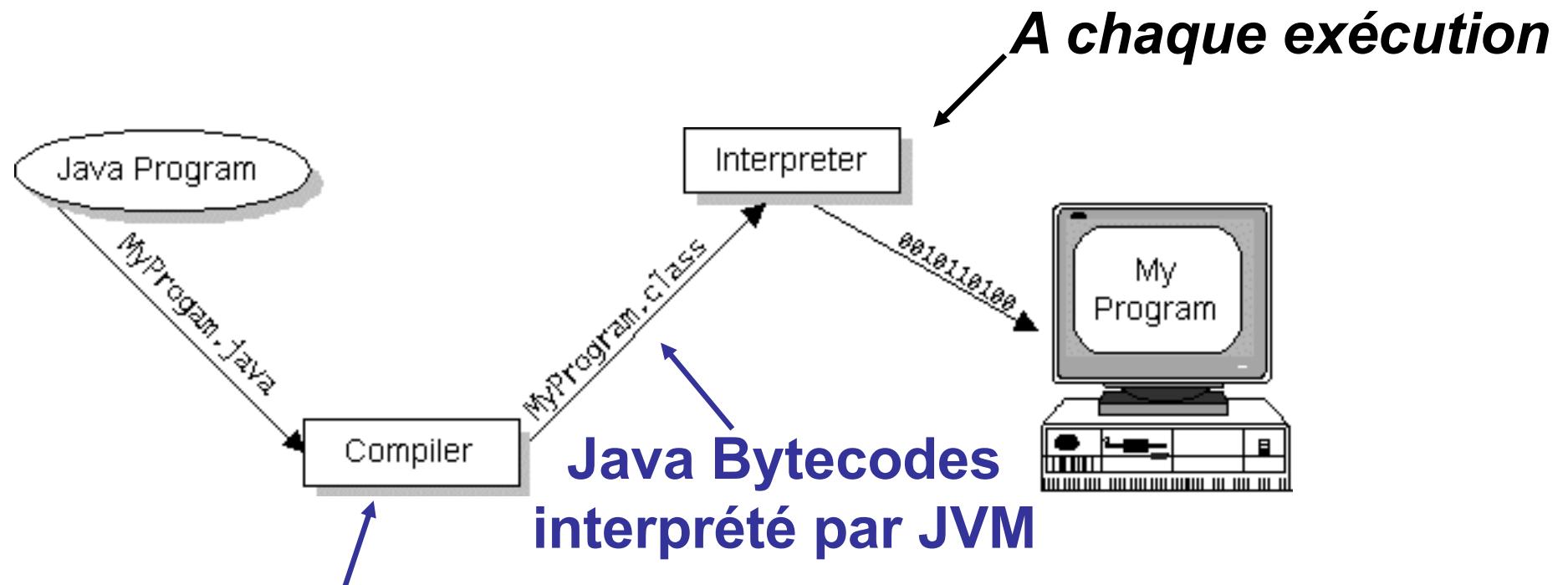
- Un langage de programmation orienté objets
  - Un ensemble de classes standards réparties dans différents **packages (API )**
  - Un ensemble d'outils (le **JDK,...**)
- Un environnement d'exécution JRE (Java Runtime Environment):  
la **machine virtuelle Java(JVM)**





# Présentation de JAVA

- Java est compilé et interprété



Fait une seule fois  
« compile once, run everywhere »

# Présentation de JAVA

## Un langage portable

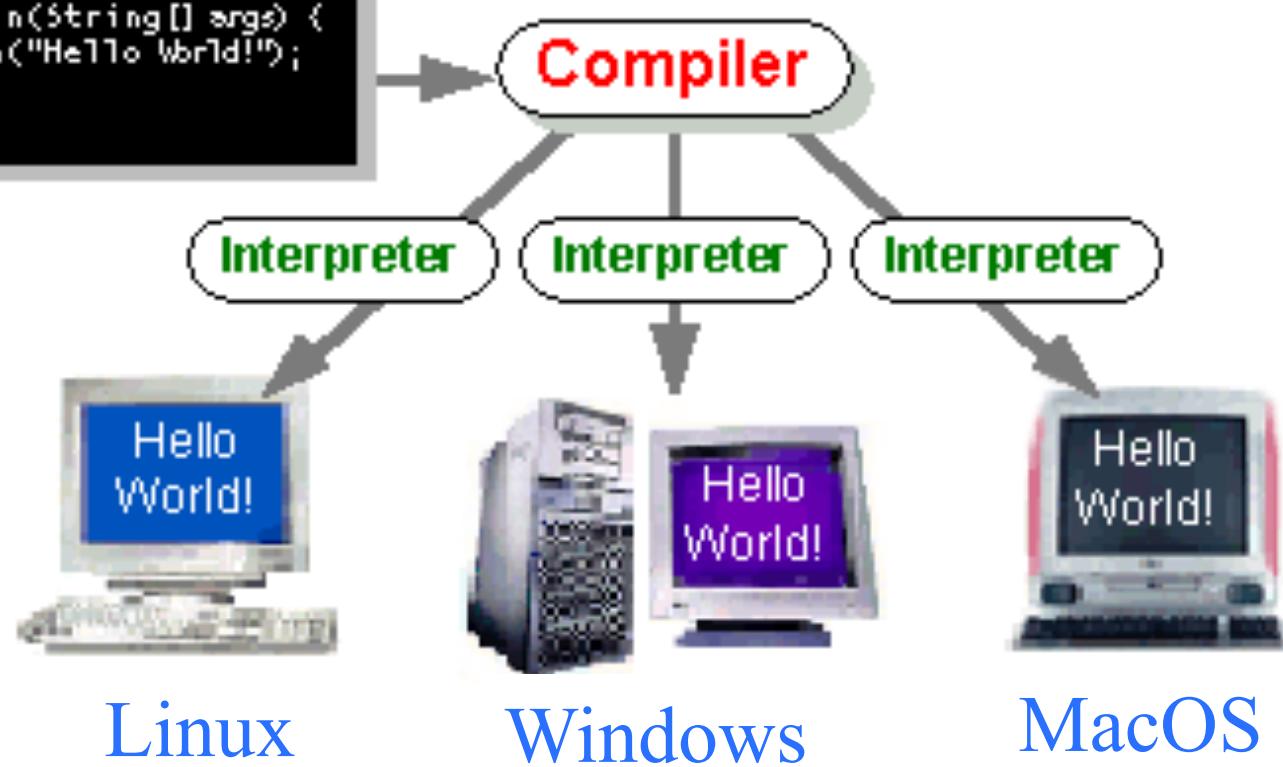


- Indépendant de la plate-forme

### Java Program

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

HelloWorldApp.java



# Présentation de JAVA

- Plusieurs éditions différentes disponibles gratuitement sur

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

- Java Standard Edition (JavaSE)

Version de base du Java Development Kit (JDK)



- Java Enterprise Edition (JavaEE)

Développement d'applications d'entreprise distribuées et articulées autour du Web.

- Java Embedded Edition (JavaME Enbedded)

Environnement optimisé pour les secteurs de l'embarqué (cartes à puce, téléphones portables, organisateurs personnels, etc.)

A partir de janvier 2019, tout n'est plus gratuit pour les entreprises!

# Présentation de JAVA

## Le JDK

- Environnement de développement fourni par Oracle : Java Development Kit
- Il contient :
  - les classes de base de l'API java (plusieurs centaines),
  - la documentation au format HTML
  - le compilateur : **javac**
  - Le JRE (Java RunTime Environment) contenant l'implémentation de la JVM (machine virtuelle Java)
  - le visualiseur d'applets : appletviewer
  - le générateur de documentation : javadoc
  - etc.



# Présentation de JAVA

## Les constituants de l'API (ou les APIs)

- API (Application and Programming Interface)

*Interface pour la programmation d'applications*

- Les classes de l'API sont regroupées, par catégories, en paquetages (ou "packages").

- Java 1 (1996) : 170 classes
- Java 2 (2000) : 1732 classes
- Java SE 5 (2004): 3270 classes
- Java SE 7(2011): 8000 classes

# Présentation de JAVA

## Documentation des classes

- Documentation standardisée au format HTML :
  - classes de l'API
  - possibilité de génération automatique avec l'outil **Javadoc**.
  - intérêt de l'hypertexte pour naviguer dans la documentation
- Accessible en ligne : <http://docs.oracle.com/javase/8/docs/api/index.html>  
ou téléchargeable gratuitement

# Présentation de JAVA

## Quelques APIs

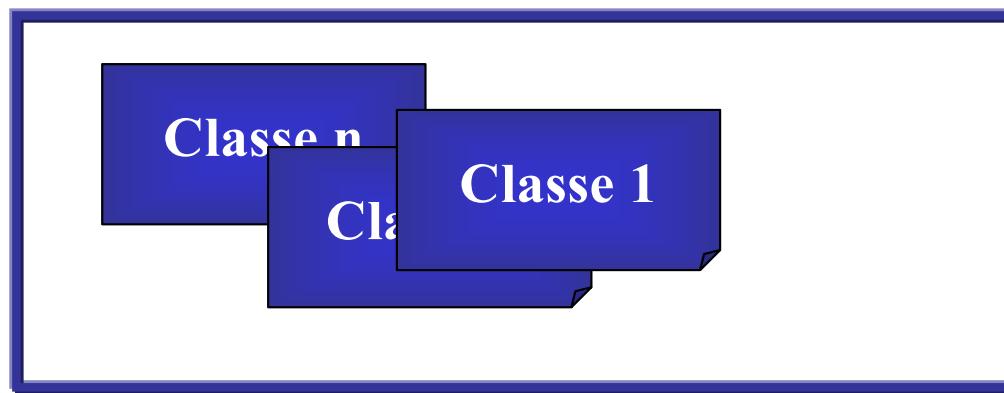
- **java.lang** : Types de bases, Threads, ClassLoader, Exception, Math,...
- **java.util** : Hastable, Vector, Date, Stack,...
- **java.applet**
- **java.io** : les entrées/sorties
- **java.net** : les réseaux

Core Apis

- **java.awt** : interface graphique
- **javax.swing** : interface graphique (nouvelle génération)

# Présentation de JAVA

- Programme Java = ensemble de classes



En général, on a une classe par fichier. Ce n'est pas une obligation mais c'est préférable pour des raisons de clarté

# Présentation de JAVA

## Types de programmes Java

- Les Applications indépendantes
  - Programmes autonomes (stand-alone)



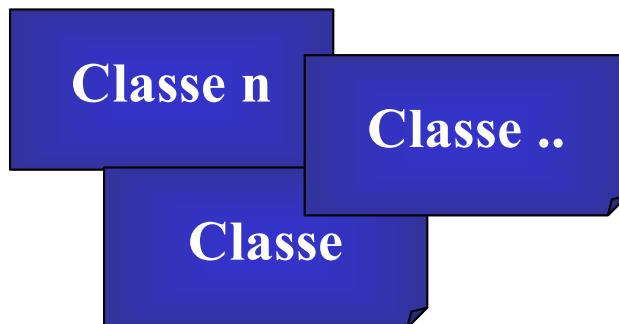
- Les Applets
  - Programmes exécutées dans l'environnement d'un navigateur Web et chargés au travers de pages HTML

*Seuls diffèrent les contextes d'invocation et d'exécution*

- Les droits des applets et des applications ne sont pas les mêmes

# Présentation de JAVA

- Application Java indépendante= une classe doit contenir la méthode « main »



## Classe Depart

```
public static void  
main(String args[ ])  
{  
.../...  
}
```

# Création d'une application

- Prenez votre éditeur de texte préféré...

```
/**  
 * La classe HelloWorldApp implémente une  
application qui  
 * affiche simplement "Hello World!" sur la sortie  
standard.  
 */  
class HelloWorldApp {  
    public static void main(String[ ] args) {  
        // Affichage du message  
        System.out.println("Hello World!");  
    }  
}
```

Même les MAJUSCULES sont importantes :  
 $A \neq a$

- Sauvegardez le fichier sous le nom  
**HelloWorldApp.java**

# TPCours



- Récupérer le sujet de TPCours
- Téléchargez et installez l'environnement de développement Eclipse en suivant la démarche indiquée dans le sujet
- Créez votre premier programme « hello world » en ajoutant une classe HelloWorld dans votre package exercice0

# Présentation de JAVA

## Création d'une applet

- Prenez un éditeur de texte quelconque ...

Pour  
information

```
/**  
 * La classe HelloWorld l'applet la plus simple  
 */  
import java.applet.Applet;  
import java.awt.Graphics;  
  
public class HelloWorld extends Applet {  
    public void paint(Graphics g) {  
        g.drawString("Hello world!", 50, 25);  
    }  
}
```

- Sauvegardez le fichier sous le nom **HelloWorld.java**

# Présentation de JAVA

## Création d'une applet

- Reprenez votre éditeur de texte préféré...

Pour  
information

```
<HTML>
<HEAD>
<TITLE>Un petit exemple</TITLE>
</HEAD>
<BODY>
Voici la sortie de mon programme :
<APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

- Sauvegardez le fichier sous le nom Hello.html

# Création d'une applet

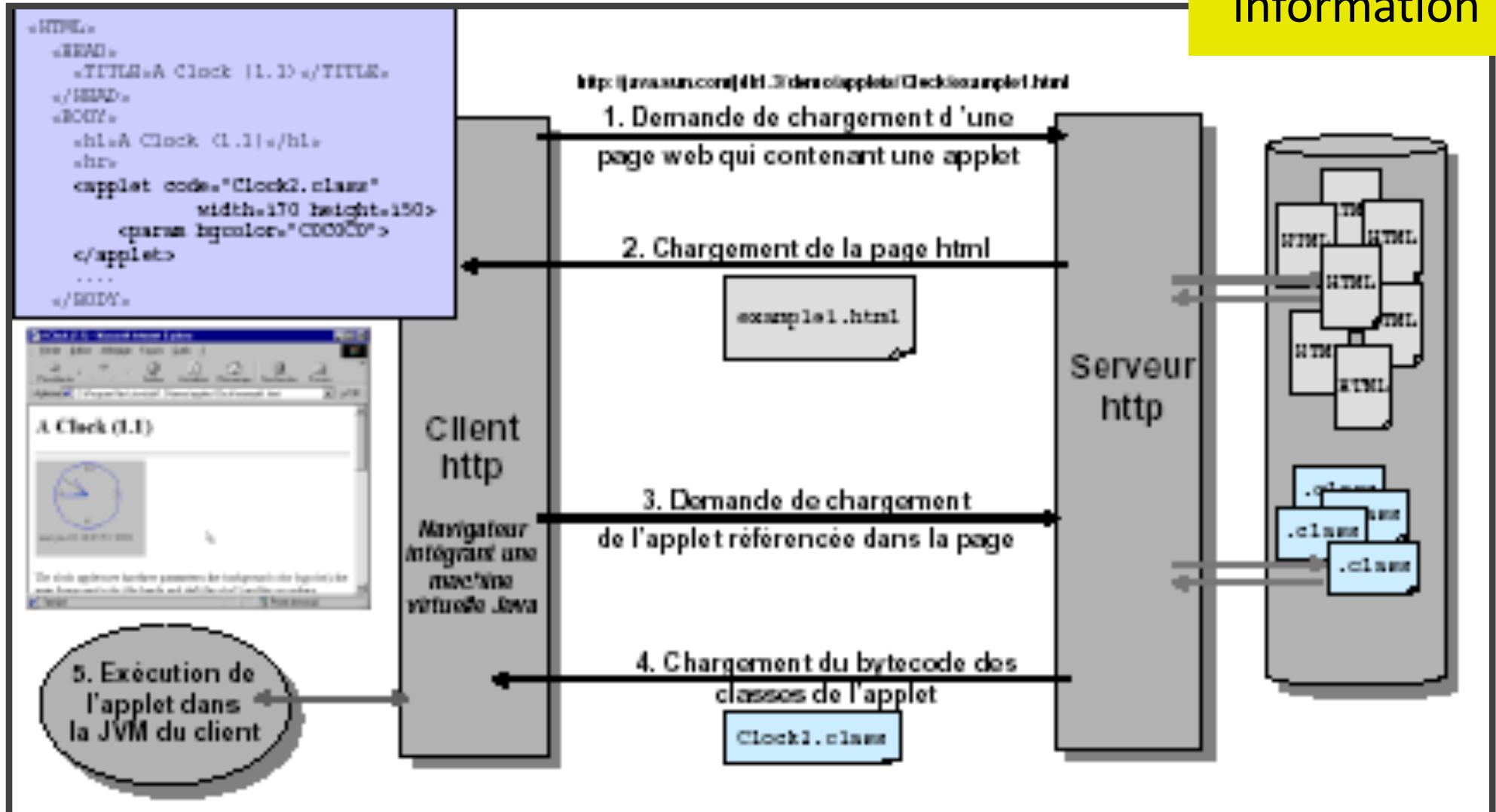
- Compilez le fichier HelloWorld.java
- Exédez le via :
  - C:\>appletviewer Hello.html
  - Ou Votre navigateur préféré...

Pour  
information



# Principe de fonctionnement d'une applet

Pour information



# Programmation Orientée objet

## Plan du Cours

### CH1 – PARADIGME ORIENTÉE OBJET

1.1 - Notion de Paradigme Orienté Objet

1.2 - Introduction au langage Java

1.3 - Modélisation Objet et UML



### CH2 – OBJETS et CLASSES

### CH3 – COMMUNICATION ENTRE OBJETS

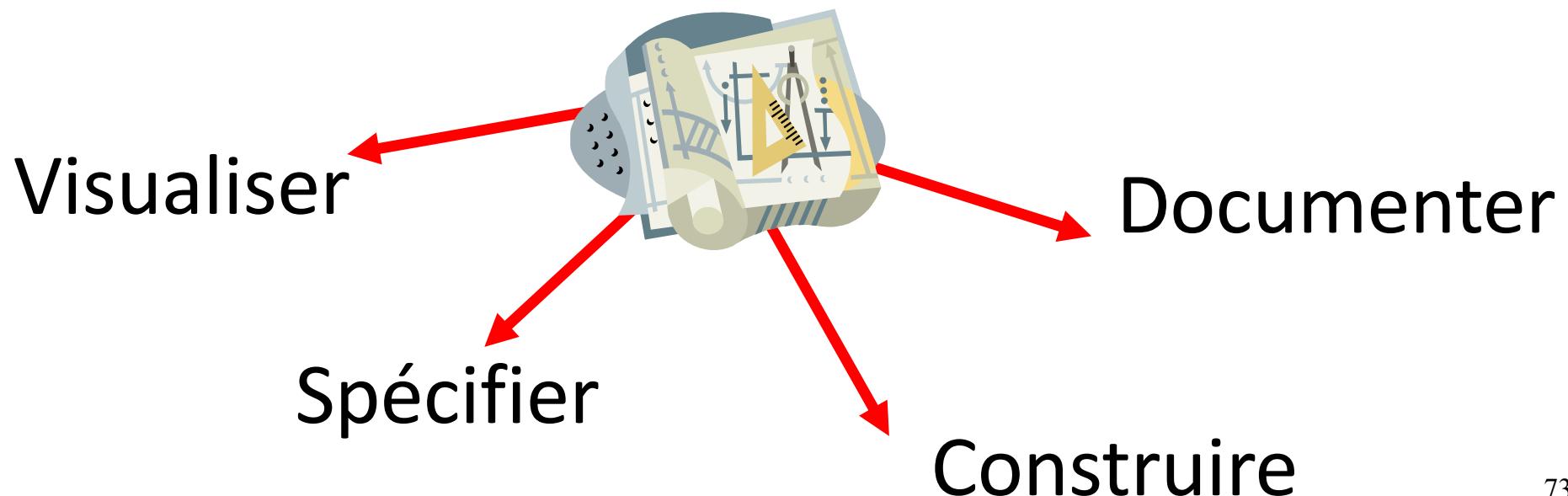
### CH4 – HERITAGE et POLYMORPHISME

# Modélisation OO et UML

## Qu'est ce qu'UML?

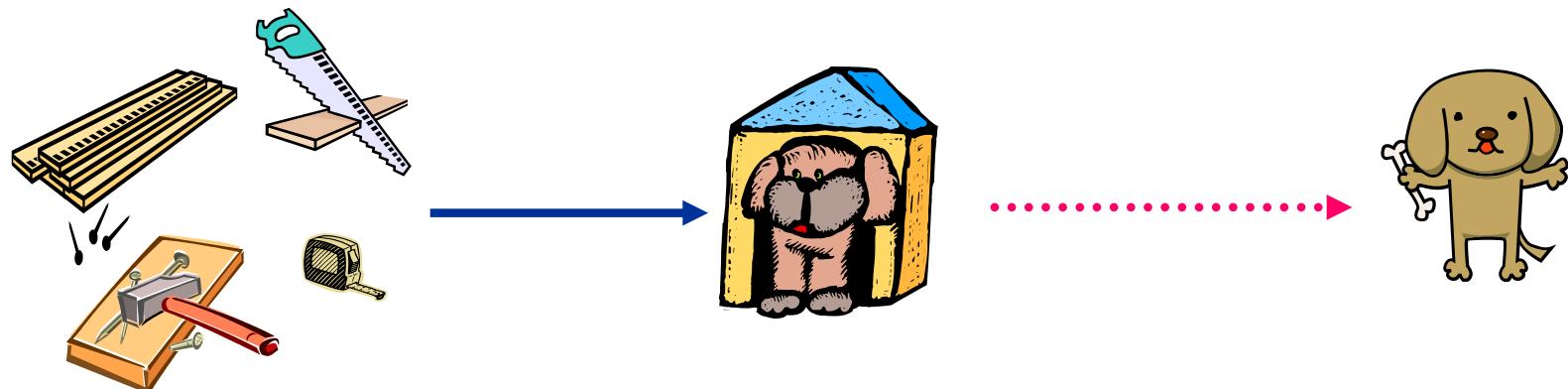
**Unified Modeling Language**  
*Langage uniifié pour la modélisation*

Langage **standard** pour l'écriture  
de **plans** d'élaboration de logiciels.

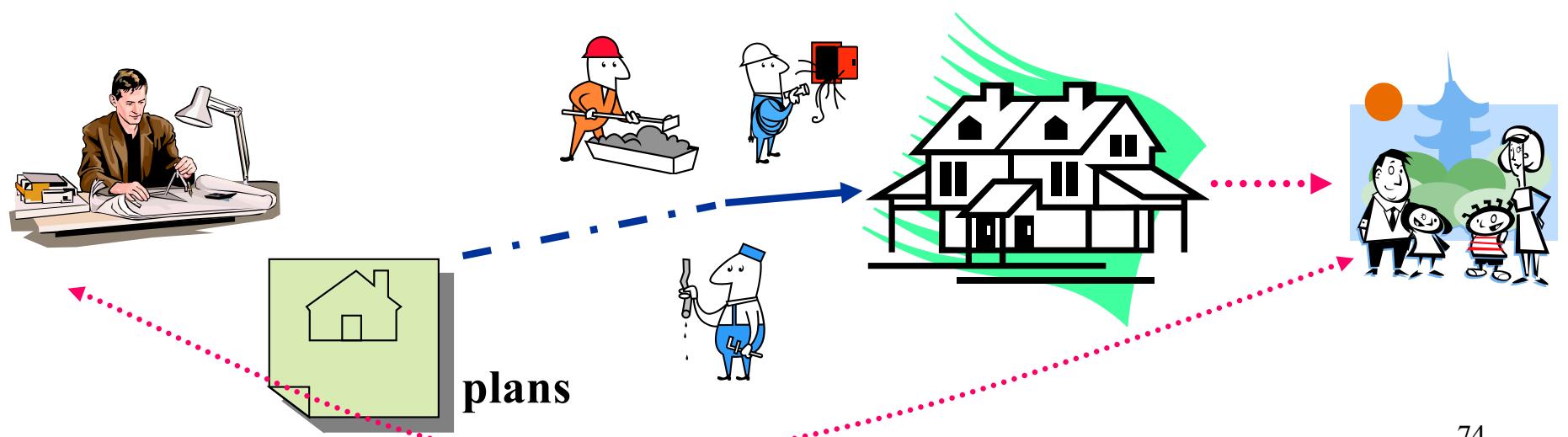


# Pourquoi la modélisation?

## Construction d'une niche

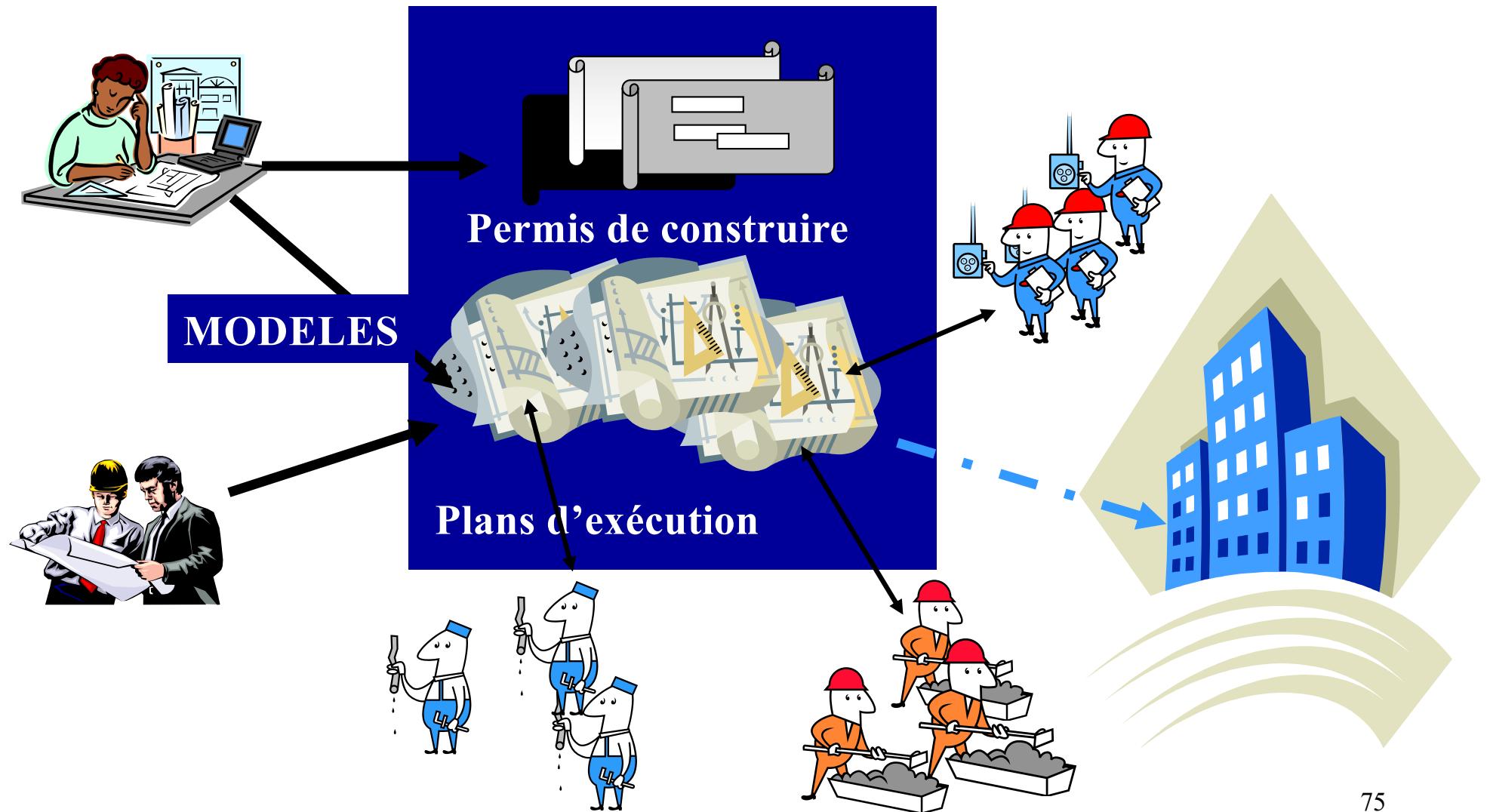


## Construction d'une maison



# Pourquoi la modélisation?

## Construction d'un immeuble



# Modélisation OO et UML

## Qu'est ce qu'UML?

- Outil de **Communication**,  
**d'aide au développement** et  
à la **maintenance**
  - Indépendant de la méthode
  - Indépendant des langages de programmation
  - Adapté à toutes les phases du développement
  - Compatible avec toutes les techniques de réalisation
- Notation **graphique** Simple, Générale et Flexible



# Historique d'UML

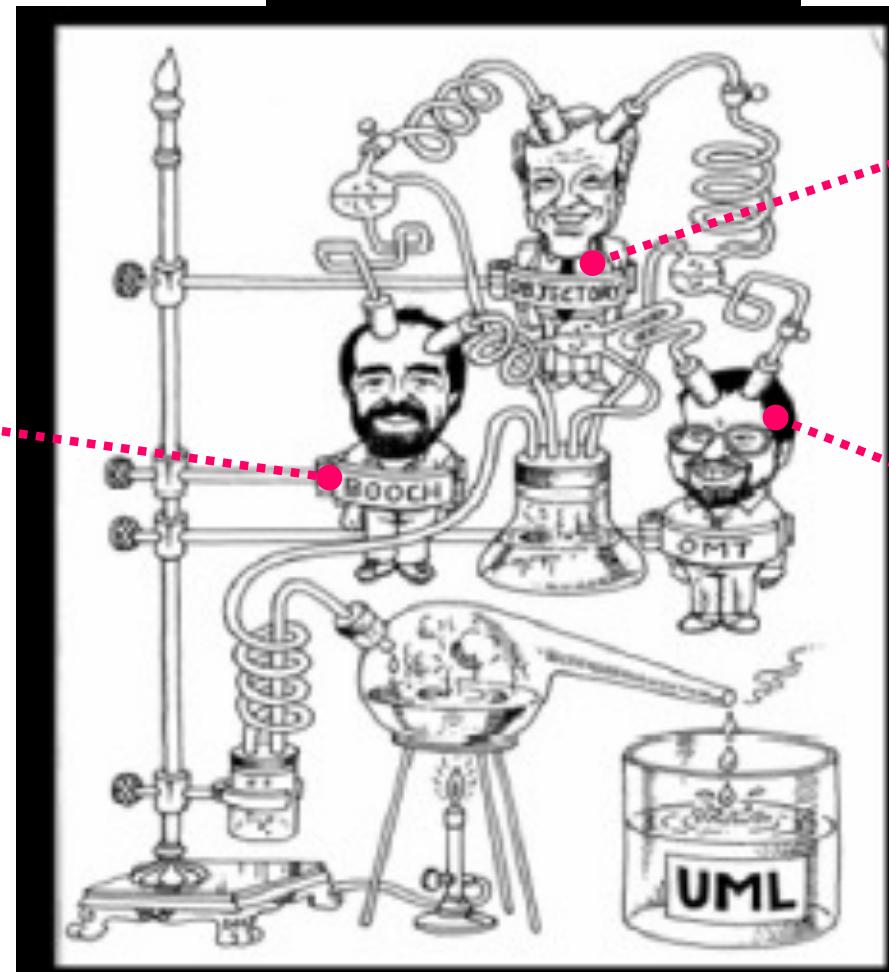
## Les « pères fondateurs » (1995)

Rational. software

### Les 3 amigos



Grady BOOCH



Ivar JACOBSON



James RUMBAUGH

# Historique d'UML

## Les « batisseurs » (depuis 1996)

- Organisme à but non lucratif
- Crée en 1989
  - Initiative: HP, Sun, Unisys, American Airlines, F
  - Mission :
    - ✓ Essor des technologies objet
    - ✓ Promotion de standards
    - ✓ Garantie de l'interopérabilité

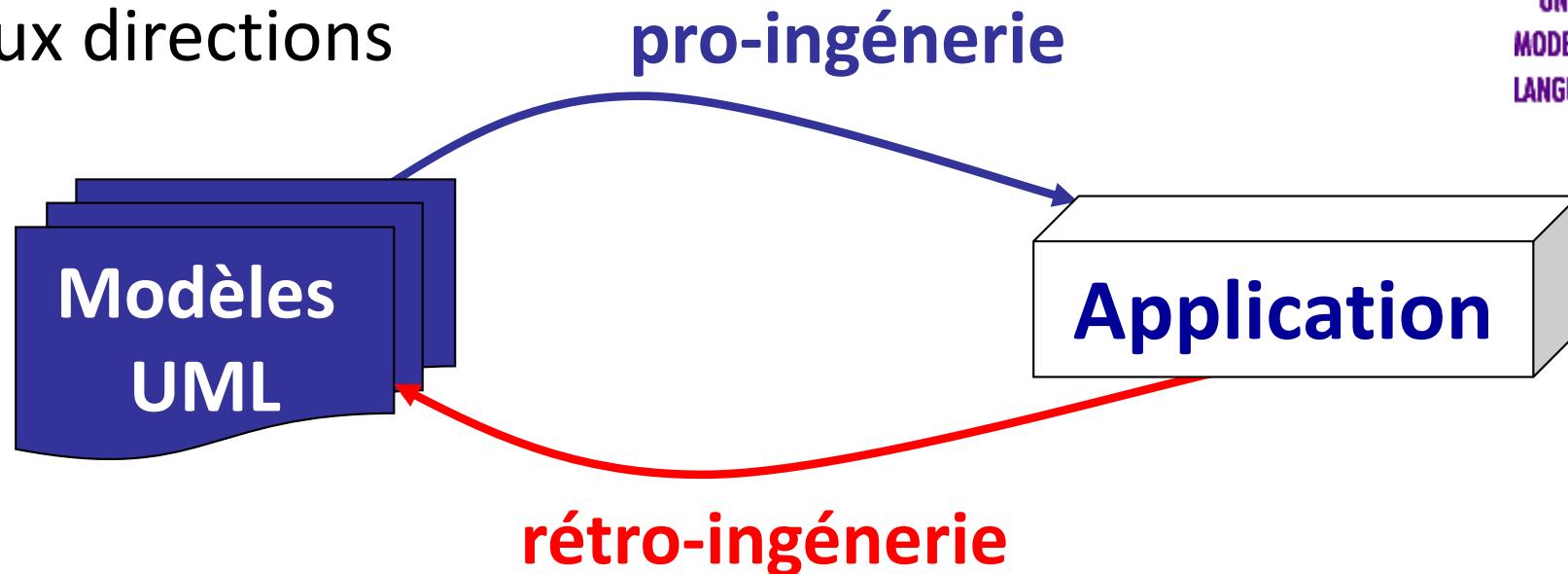


***UML 1.1 (1997)***

***UML 2.0 (2005)***

# Modes d'utilisation d'UML

- Deux directions

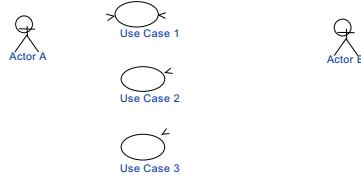


Trois modes d'utilisation

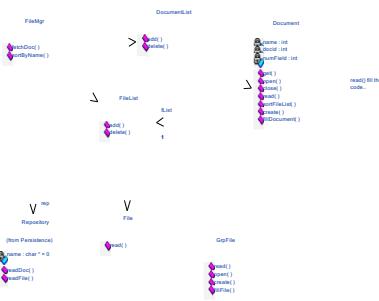
- Esquisse
- Plan
- Langage de programmation

# La « boîte à outils » UML

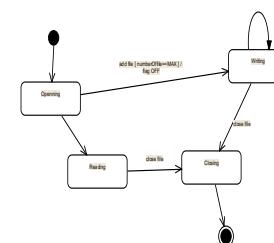
**Diagramme  
des cas d'utilisation**



**Diagramme  
de classe**

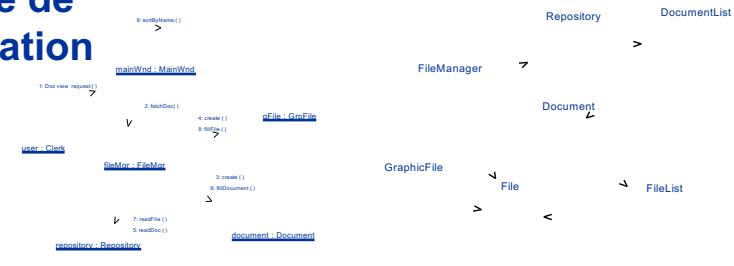


**Diagramme de  
machines d'état**

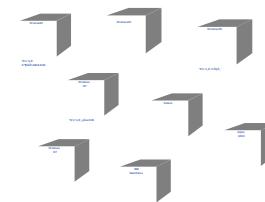


## Diagrammes

**Diagramme de  
communication**



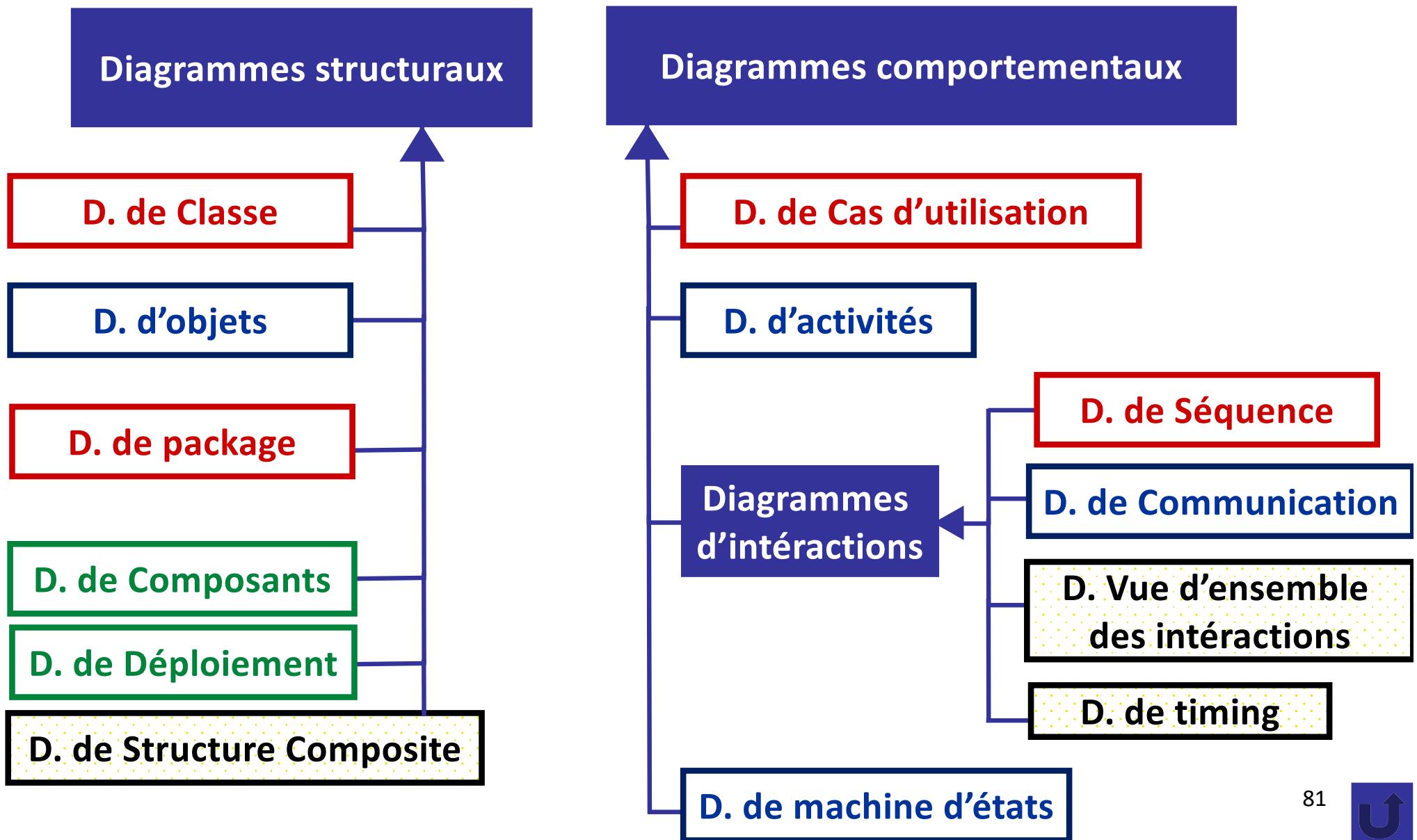
**Diagramme  
de déploiement**



**Diagramme de  
composants**

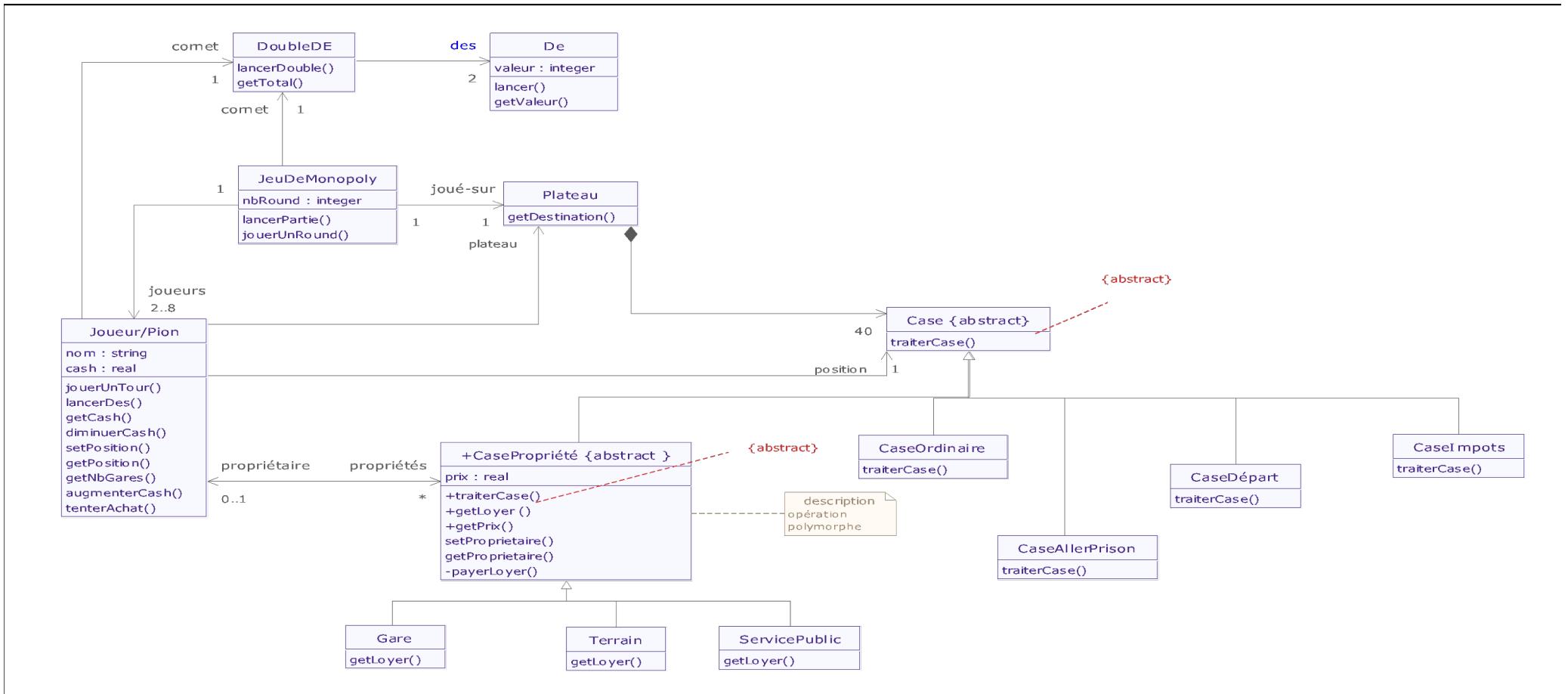
**Diagramme  
de séquence**

# Les 13 Diagrammes UML2



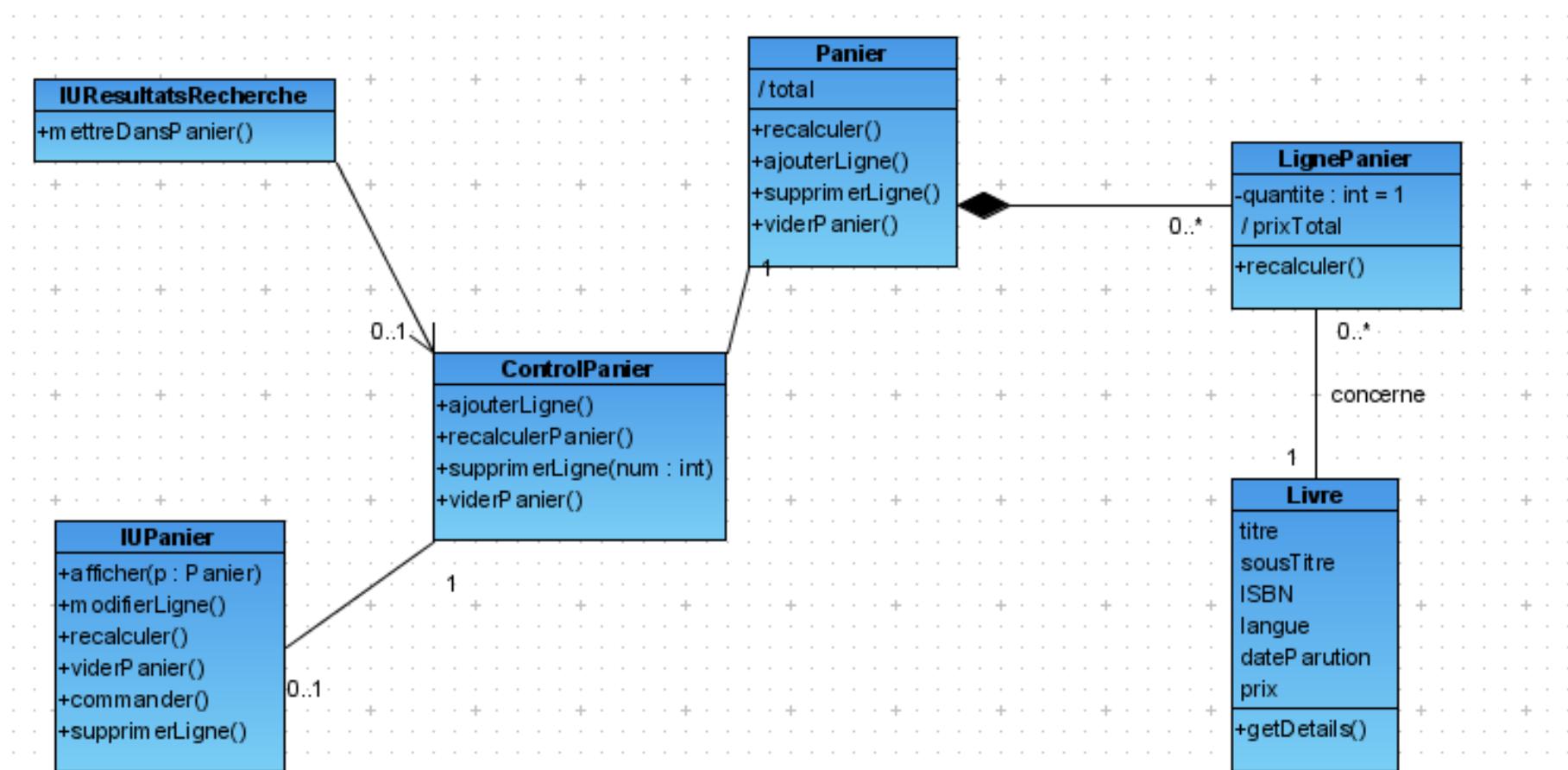
# Diagramme de classes

Description de la structure Statique de l'application



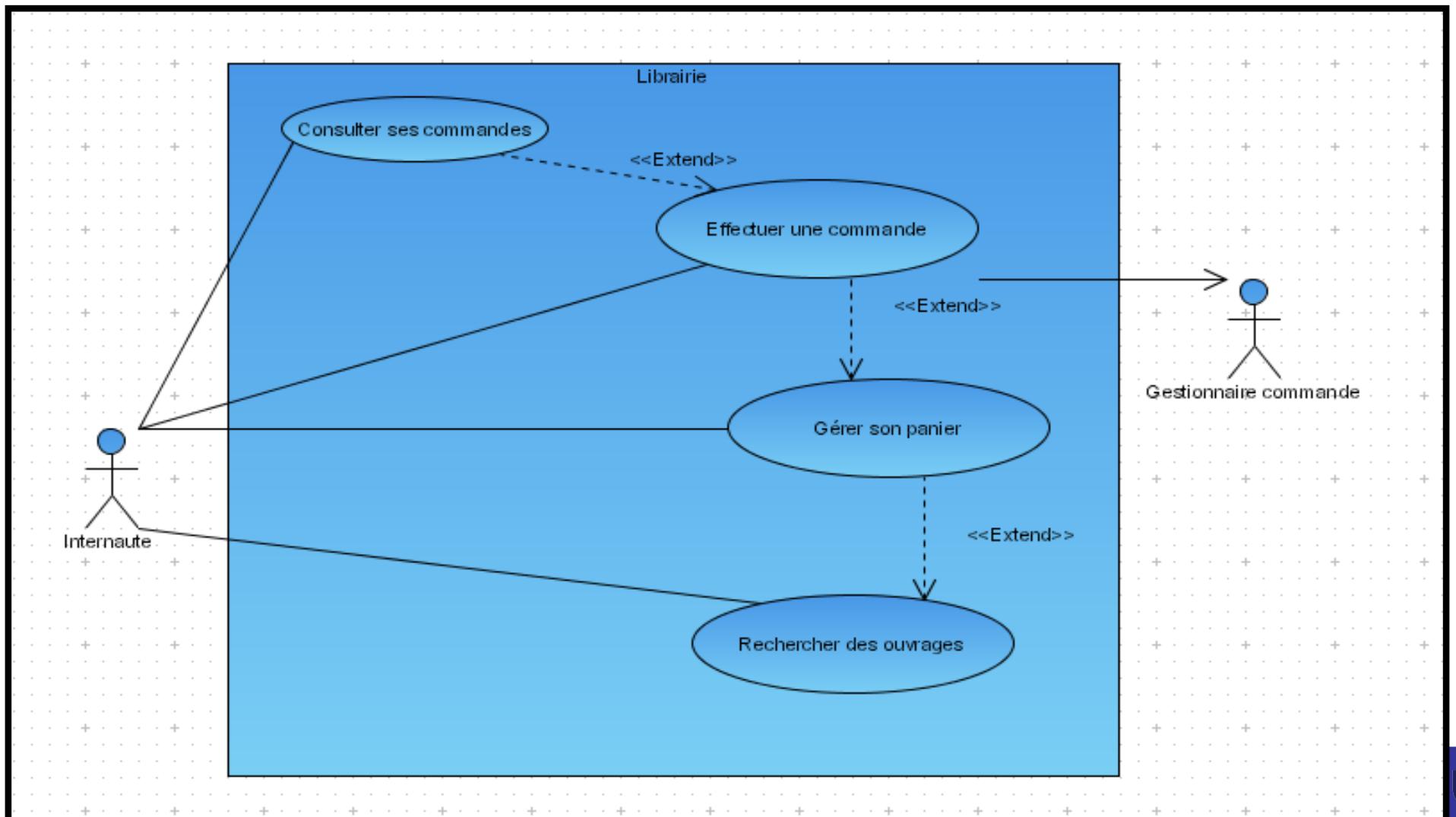
# Diagramme de classes

Description de la structure Statique de l'application



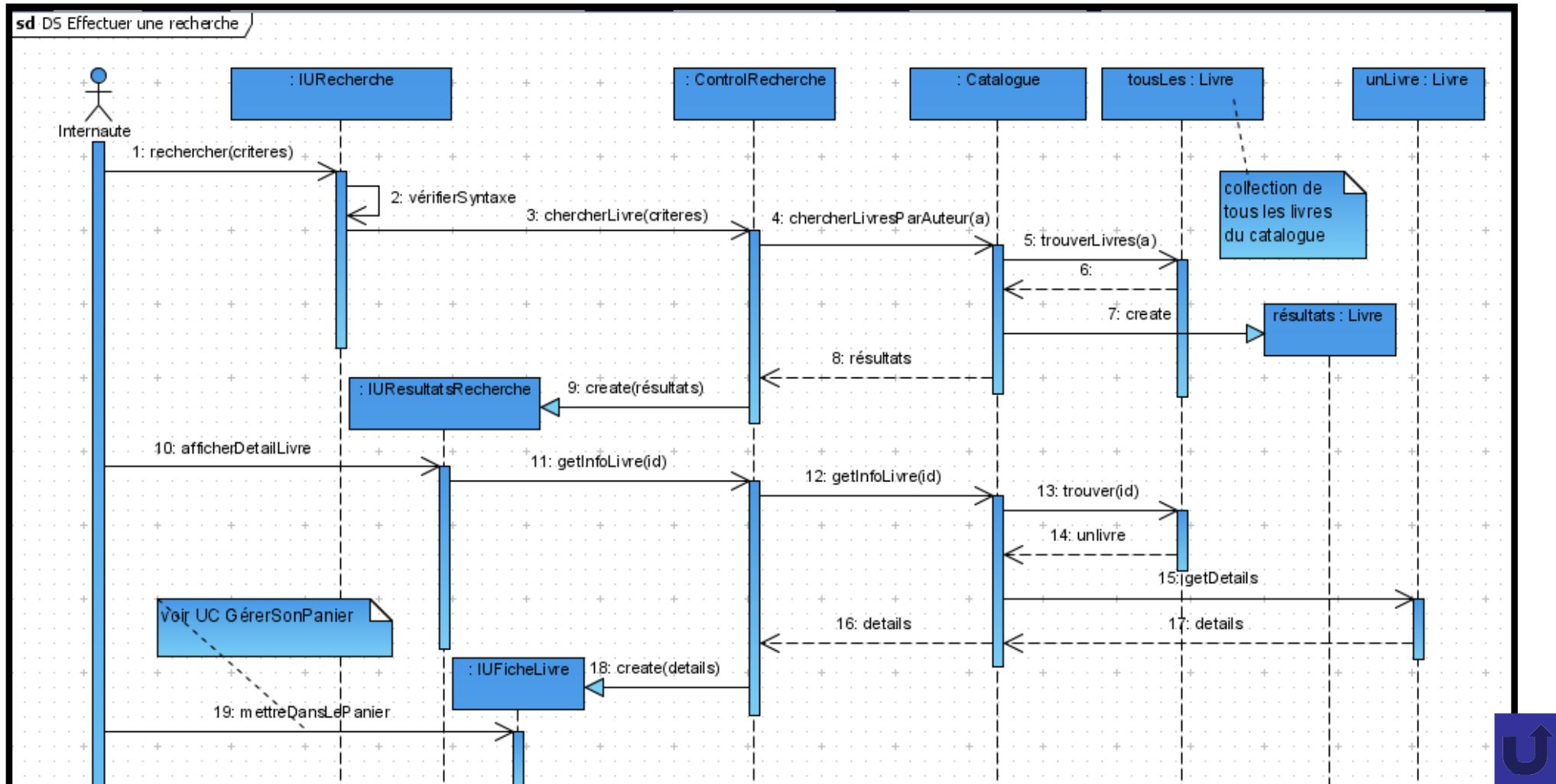
# Diagramme de cas d'utilisation

Visualisation de l'interaction de l'application  
avec le monde extérieur



# Diagramme de séquence

Description des enchaînements d'envois de messages  
pour réaliser une fonctionnalité de l'application  
(point de vue temporel)



# Références



## Sites sur Java

- Le Tutorial officiel en ligne :  
<http://docs.oracle.com/javase/tutorial/index.html>
  
- Le site très pédagogique de Jean-Michel Doudoux (mis à jour régulièrement)  
<http://www.jmdoudoux.fr/java/dej/indexavecframes.htm>