# Annexe 1B : Pourquoi apprendre une méthode de programmation orientée-objet ?

## Conception de logiciels

Un très grand nombre de projets de conception de logiciels échouent ...

Pourquoi ?

* domaine en perpétuel changement ( tech, employés )
* complexité des systèmes
* mauvaise analyse des exigences des usagers
* projets trop longs à mettre en oeuvre, déjà dépassés

solution : OO car...

* structure simplifiée ( héritage )
* réutilisation ( transmission auto vs transmission manuelle )
* meilleure analyse ( choix des objets, des classes )

## Maintenance de systèmes et / ou de logiciels

Tâches :

1.Comprendre le logiciel et les changements à y apporter

* *étudier la doc*
* *trouver la source des problèmes*

2.Modifier le logiciel en y ajoutant les changements nécessaires

* *implantation des nouveaux éléments*
* *modifier la doc si nécessaire*

3.Revalider le logiciel avec tests

## Problèmes :

**1. programmes sont difficiles à comprendre**

* code mal structuré

**solution OO 1er concept fondamental:** l’heritage

Ex. 1 :

public classe point{

private int x,y;

}

sous-classe IPhone ( toutes les fonctionnalités d'un Ipod + de nouvelles )

Ex. 2 :

Public class pixel extends point{

String couleur; // x et y sont heriter

* accès à la documentation rapide : javadoc , [API Java](https://docs.oracle.com/en/java/javase/12/docs/api/java.base/module-summary.html) ( toutes les classes , méthodes qu'on peut utiliser, réutiliser )
* Nous utilisons JDK 12

**2. Programmes sont difficiles à modifier**

* couplage : un changement dans une composante se répercute à d'autres endroits multipliant les résultats erronés/inattendus

**solution OO 2ème concept fondamental: *encapsulation de donnes***

déf : le client externe à l'objet ne peut pas bousiller l'état interne d'un objet , il peut juste lui envoyer des messages

les seules méthodes pouvant modifier l'état interne d'un objet sont les méthodes de cet objet ; elles pourront être appelées par d’autres objets

pas nécessaire que les clients externes utilisant l'objet connaisse toute la mise en oeuvre de l'objet utilisé ( conduire une voiture sans connaître le fonctionnement du moteur )

exemples :

* retirer argent de notre compte en multiple de 20$
* impossible de connaitre le solde d’un autre objet
* pas de droit, ou droit en lecture seulement
* par défaut, toutes les variables d’instance seront private.

**3eme concept fondamental :** *polymorphisme : un objet peut prendre différentes formes (different type) dependant du context du projet*

Exemples :

calculerConsommationEssence message envoyé à un objet Mini ou à un camion 18 roues, on peut être confiant que les méthodes ont une manière de calculer différente

calculerAssurance sur une Honda CIvic Si ou sur une Corolla ...

dessiner un carré ou un cercle

On peut également désigner le polymorphisme par rapport à un objet ayant deux ou plusieurs types :

On peut considérer ma voiture comme un objet de type Voiture ou de la classe Toyota ou de la classe MoyenDeTransport.

3. Dur à valider : tests JUnit