



# 02

Chapitre

## Un paradigme : conception orientée objet

**1I2AC1 : Génie logiciel et Conception orientée objet**

Régis Clouard, ENSICAEN - GREYC

« N'importe quel programmeur peut écrire du code que l'ordinateur comprend. Les bons programmeurs écrivent du code que les humains peuvent comprendre. »

**Martin Fowler**

# Plan du chapitre

---

1

Le paradigme  
objet

# Paradigmes de programmation

---

- Donnez quelques paradigmes de programmation

# Paradigme procédural vs Paradigme objet

4

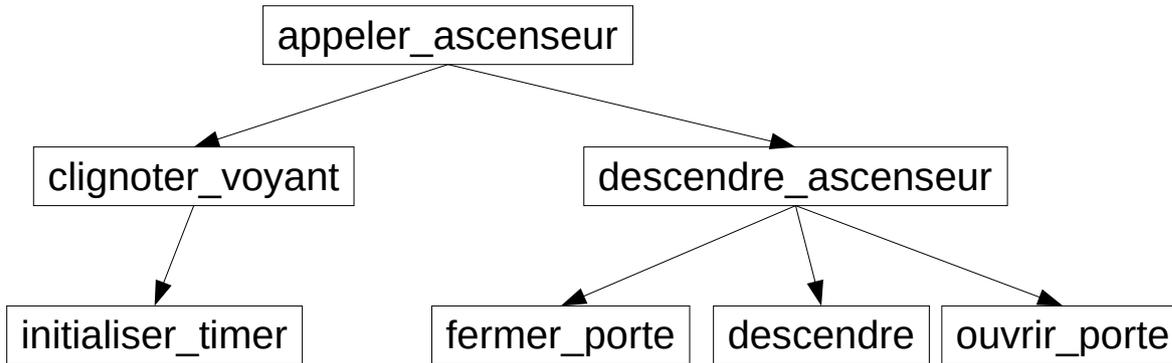
- Deux points de vue sur le problème → Points de vue duaux

- **Procédural**

- Point de vue sur les opérations
- Les données sont inertes

- **Objet**

- Point de vue sur les données
- Les données sont animés



Grappe d'appels

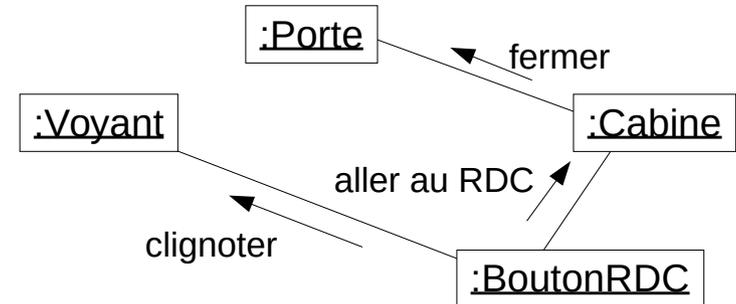


Diagramme de collaboration

# Paradigme procédural vs Paradigme objet

---

5

**Exemple**

**Comptage des étudiants  
présents en cours**

# Conception procédurale vs Conception objet

---

6

## ■ Algorithmique

- Question à résoudre : **Que veut-on faire ?**
- Réponse : la séquence d'appels des procédures

## ■ Modélisation

- Question à résoudre : **De quoi parle t-on ?**
- Réponse : la liste des objets avec les bons services

# Conception orientée objet (COO)

## Conception orientée objet :

*Si je disposais d'un chapeau magique, quel type de données voudrais-je voir sortir du chapeau pour m'aider à résoudre le problème ?*



# Conception orientée objet

---

## Exemple

**Guichet automatique de billets  
(GAB ou *ATM*)**

# Conception procédurale vs Conception objet

9

## ■ Algorithmique

- Avantages
  - ▶ Proche de la représentation machine
- Limites
  - ▶ Inaccessible aux clients
  - ▶ Inadaptée à la complexité des gros logiciels

## ■ Modélisation

- Avantages
  - ▶ Adapter aux gros logiciels : approche cartésienne de la conception
  - ▶ Implémentation repoussée le plus tard possible
  - ▶ Accessible aux clients
- Limite
  - ▶ Vision fractionnée du logiciel

# Programmation procédurale vs Programmation orientée objet

---

10

- La différence ne concerne que quelques mots clés
  - ▶ 6 mots clés suffisent pour passer du C au Java :  
`class`, `extends`, `implements`, `interface`, `new`, `public`
  - Mais ce sont deux paradigmes différents
- Conséquence :
  - Le paradigme objet ne s'apprend pas par le langage

# Concepts de la conception objet

---

- La conception orientée objet s'appuie sur 5 concepts :
  - 1) Objet et principe d'encapsulation
  - 2) Classe
  - 3) Associations
  - 4) Héritage
  - 5) Polymorphisme

# Plan du chapitre

1  
Le paradigme  
objet

2  
Les objets  
(encapsulation)

# Notion d'objet

- Concrètement
  - **Objet = structure en C incluant des pointeurs sur des procédures**
- Par exemple une voiture

```
at_013_sr: Car
color = blue
quantity= 42 l
power = 100 hp
move()
stop()
refuel()
```

```
typedef struct s_car {
    int color;
    int quantity;
    int power;

    void (*move)();
    void (*stop)();
    void (*refuel)(int);
} Car;
```

```
Car at_01_sr;

at_01_sr.power = 110;
at_01_sr.refuel(10);
```

# Notion d'objet

14

## ■ Conceptuellement

- **Objet = propriétés + services**
- **Propriété (Attribut) : donnée membre**
  - ▶ Possédant une valeur
  - ▶ Pouvant évoluer au cours du temps
- **Service (Méthode) : procédure membre attachée à l'objet**
  - ▶ Utilisant potentiellement les données membres pour fonctionner
  - ▶ Déclenchée par appel explicite à partir de l'objet

at-013-sr: Car

color = blue  
quantity= 42 l  
power = 100 hp

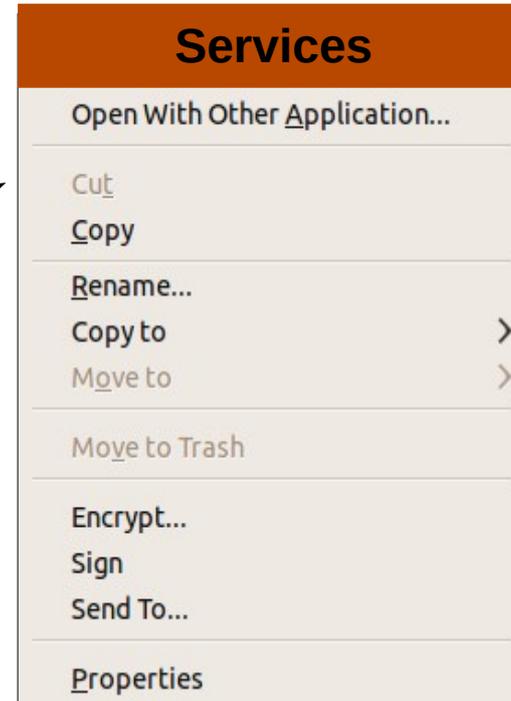
move()  
stop()  
refuel()

# Principe d'encapsulation

- objet = fournisseur de services  
≠ structure de données



CD/DVD Drive



# Principe d'encapsulation

- Exemple : parcours d'une chaîne de caractères pour la mettre en majuscule
- En C
  - En Java

# Principe d'encapsulation

- Éprouvez la différence essentielle entre :

1) `at_01_sr.color = "blue";`

2) `at_01_sr.setColor("blue");`

# Principe d'encapsulation

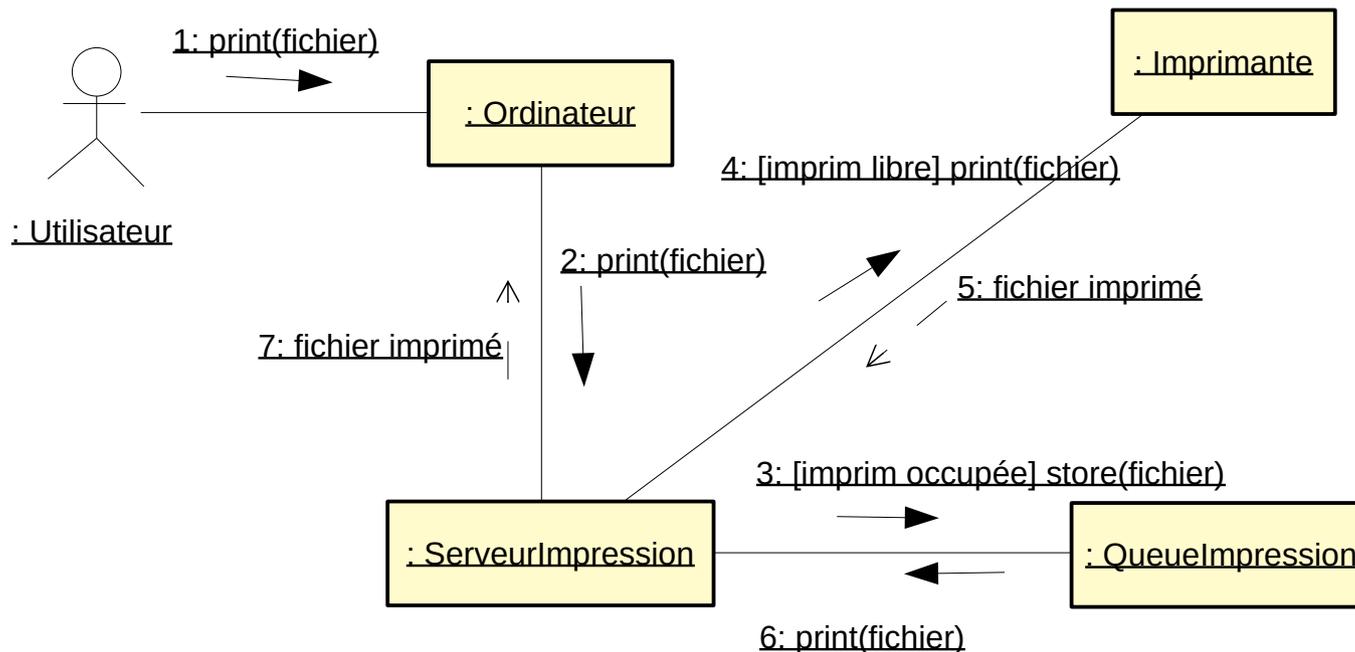
18

- Les attributs ne sont pas une préoccupation de la conception mais de la programmation
- Un attribut n'existe que parce qu'un service en a besoin

**Ne me parlez plus d'attributs en modélisation  
(sauf à ma demande)**

# Association

- Un objet ne doit pas être omniscient mais au contraire spécialisé
  - Sinon cela revient à faire de la conception procédurale
- Il doit donc faire appel aux services d'autres objets qu'il connaît par la liste de ses associations



# Plan du chapitre

---

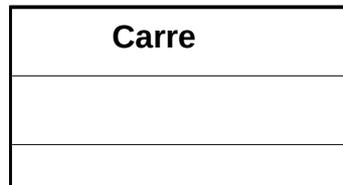
1  
Le paradigme  
objet

2  
Les objets

3  
Les classes

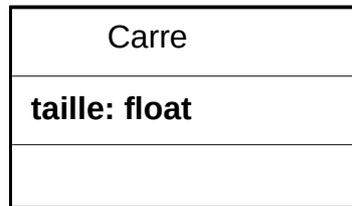
# Classe

- Intention
  - Représente un concept du domaine
  - Génératrice d'objets
- Nom : substantif **au singulier**
- Casse : PascalCase
- Représentation UML



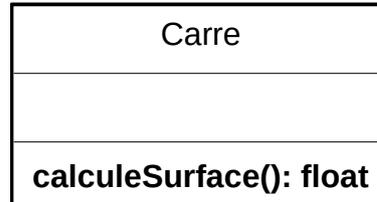
# Attribut / Propriété

- Intention
  - Porte une propriété de l'objet
- Nom : substantif
- Type : primitifs ou assimilés
- Casse : camelCase
- Représentation UML



# Méthode / Service

- Intention
  - Propose un service
- Nom : verbe
- Casse : camelCase
- Représentation UML



# Implémentation en Java

---

24

**Exemple**

**Cas de Voiture**