ENSIAS

M.2.5.1. Paradigme Objet

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Programmation Orientée Objet

M.2.5.1 : Paradigme Objet (30%)

- Cours (8h) + TP (6h)
- Exam 30 min

M.2.5.2 : Programmation Objet (70%)

- Cours (12h) + TP (10h) + AP (14h)
- Exam 1h (60%) + AP (40%)

ENSIAS

Programmation Orientée Objet

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

M.2.5.1 : Paradigme Objet (30%)

- Cours (8h) + TP (6h)
- Exam 30 min

Leçon 1 : Du procédural à l'Objet

Leçon 2 : Concepts objet de base

Leçon 3&4 : Java - les premiers pas

M.2.5.1. Paradigme Objet

1. Du procédural à l'Objet
2. Concepts objet de base
3. Java : les premiers pas

Plan

- Paradigme objet
- Langages à objet

Qualité du logiciel…

1. **Valide**

   respecte le *cahier des charges*

2. **Fiable**

   fonctionne même dans des *conditions anormales*

3. **Flexible**, **Réutilisable**

   s'adapte à de nouvelles *applications*

4. **Extensible**

   s'adapte à de nouvelles *fonctionnalités*

5. **Portable** : peut être transféré dans *différents environnements* logiciels et matériels

Qualité du logiciel…

6. **Compatible** se combine *avec d'autres*
7. **Maintenable**

   modification, correction, adaptation (clarté du code, commentaires, choix des SDs)
8. **Facile d'utilisation** par un *client*
9. **Efficace**

   utilise de façon optimale les *ressources* disponibles
10. **Intègre** : protège son code et ses données contre des *accès non autorisés*

ENSIAS

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Histoire ...

VS

Procedural Programming

Object-Oriented Programming

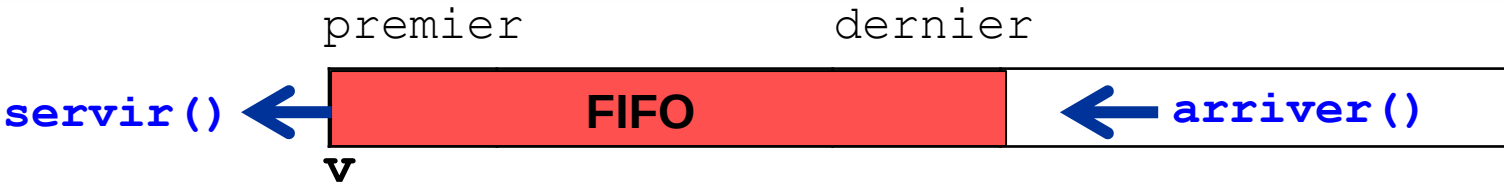
Structurée et modulaire

- Equation de **Niclauss Wirth** : **Pascal, C**
- **Module :** Données privées + Procédures

Fichier

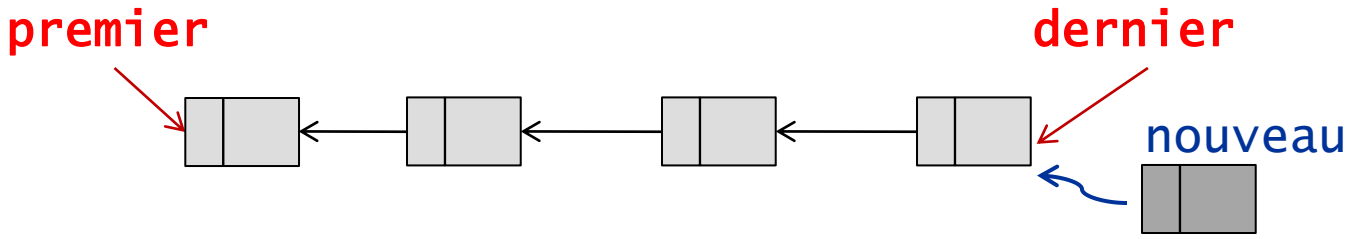
```
static client v[100]; //privée (locale)
static int premier, dernier;
void arriver(client x){
  if(! filePleine()) { dernier++; v[dernier]=x;}
}
int servir(){/* … */}
```

Représentation séquentielle



Structurée et modulaire

- **Validité : /à la spécification**
- **Accès à travers une interface**
- **Maintenabilité**



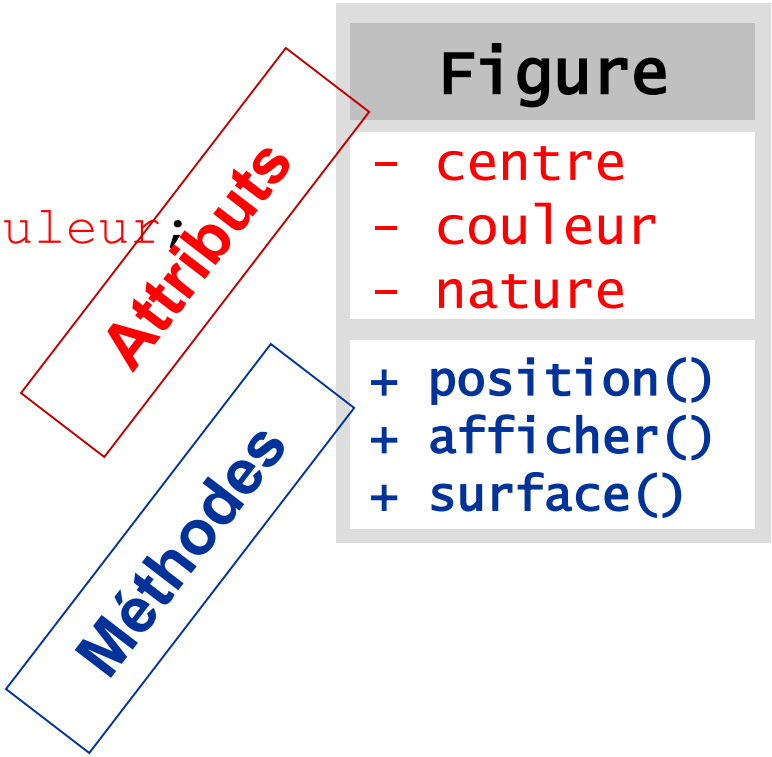

```
static client * premier, dernier;

void arriver(client * nouveau){
   nouveau -> suivant = dernier;
   dernier = nouveau;
}
```

Représentation en liste chaînée ?

Abstraction des données

On passe du module au Type = fabrique (moule)

```
enum Nature {cercle,triangle,rectangle}

class Figure {

  private :
     Point centre; Couleur couleur;
     Nature nature;

  public :

     Point position() {
        return centre;
     }

     void afficher(int);
     float surface();
}
```



Abstraction des données

On passe du module au Type = fabrique (moule)

Boite noire Il faut "connaître" la nature

```
float surface() {
  switch ( nature ) {
    case Cercle : /* code spécifique */
    case Triangle :/* code spécifique */
    case Rectangle: /* code spécifique */
  }
}
```

Figure

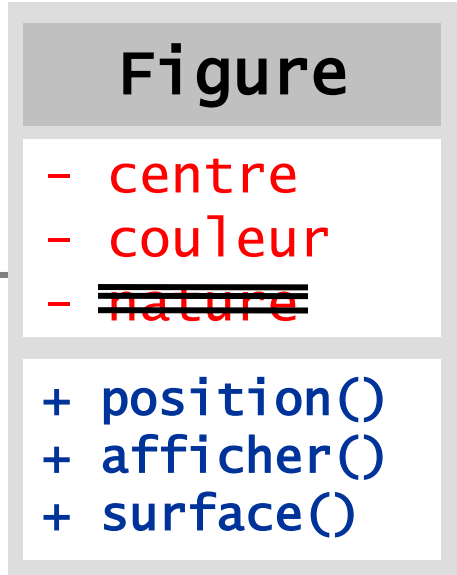
- centre
- couleur
- nature

+ position()
+ afficher()
+ surface()

Paradigme Objet

On ajoute la notion d'héritage

➔ ***D'abord Propriétés générales***




```
abstract class Figure {
    private Point centre;
    private Couleur couleur;
    public Point position() {
        return centre;
    }
   public abstract void afficher();
   public abstract float surface();
}
```

Paradigme Objet

On ajoute la notion d'héritage

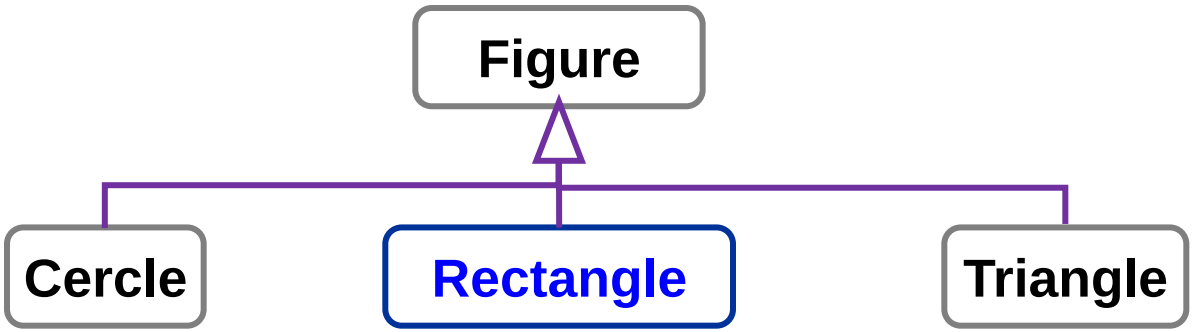
➔ ***Ensuite propriétés particulières***

```
class Rectangle extends Figure {
    private int longueur, largeur;
    float surface() {
        return longueur*largeur;
    }
}
```

Figure

- centre
- couleur

+ position()
+ afficher()
+ surface()



ENSIAS

Paradigme Objet

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Intérêt

➔ **Réduction du coût de développement**

Par Réutilisation du Code

➔ **Réduction du coût de maintenance**

Réduction des dépendances entre classes

Paradigme Objet

➔ **Décider quelles Procédures :**

Meilleurs algorithmes

➔ **Décider quels Modules :**

Masquer les données

➔ **Décider quels Types :**

Ensemble complet d'opérations

➔ **Décider quelles Classes :**

Expliciter les propriétés communes à l'aide de ***l'héritage***

Paradigme Objet

Démarche

1. **De quoi parle-t-on ?**
2. **Que veut-on faire ?**

➔ ***Ex : Simuler un carrefour***

- Rues, Feux, Véhicules, Piétons, Accidents
- Embouteillages, Manifestations

➔ ***Ouvrir()* une porte ?**

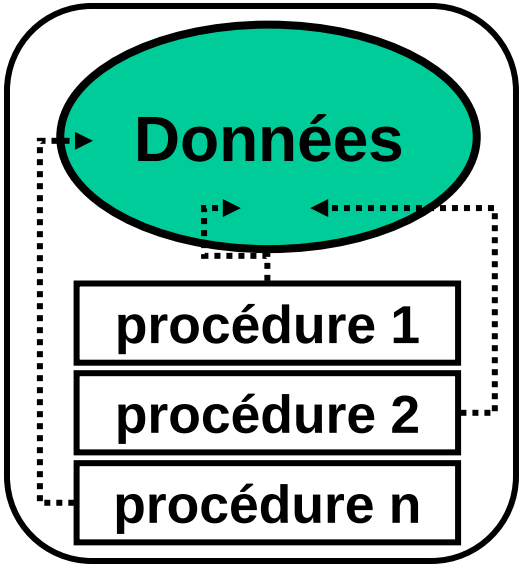
C'est la ***Porte*** *qui s'ouvre() !! (être vivant)*

- *Double battants*
- *Coulissante, à relever, …*

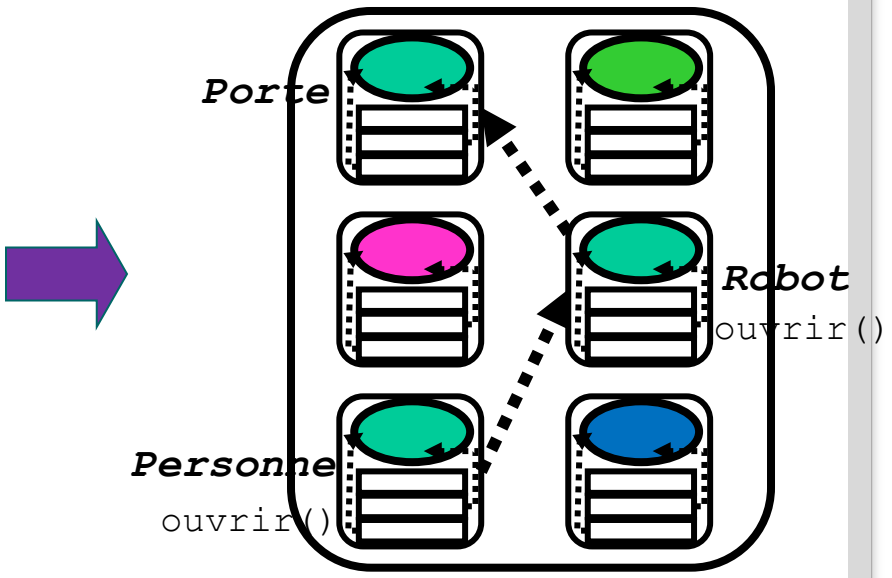
Porte d'Ali Baba

Paradigme Objet

Application Procédurale



Application Objet



Concepts

Classe → Instanciation → Objet

Envoi de messages

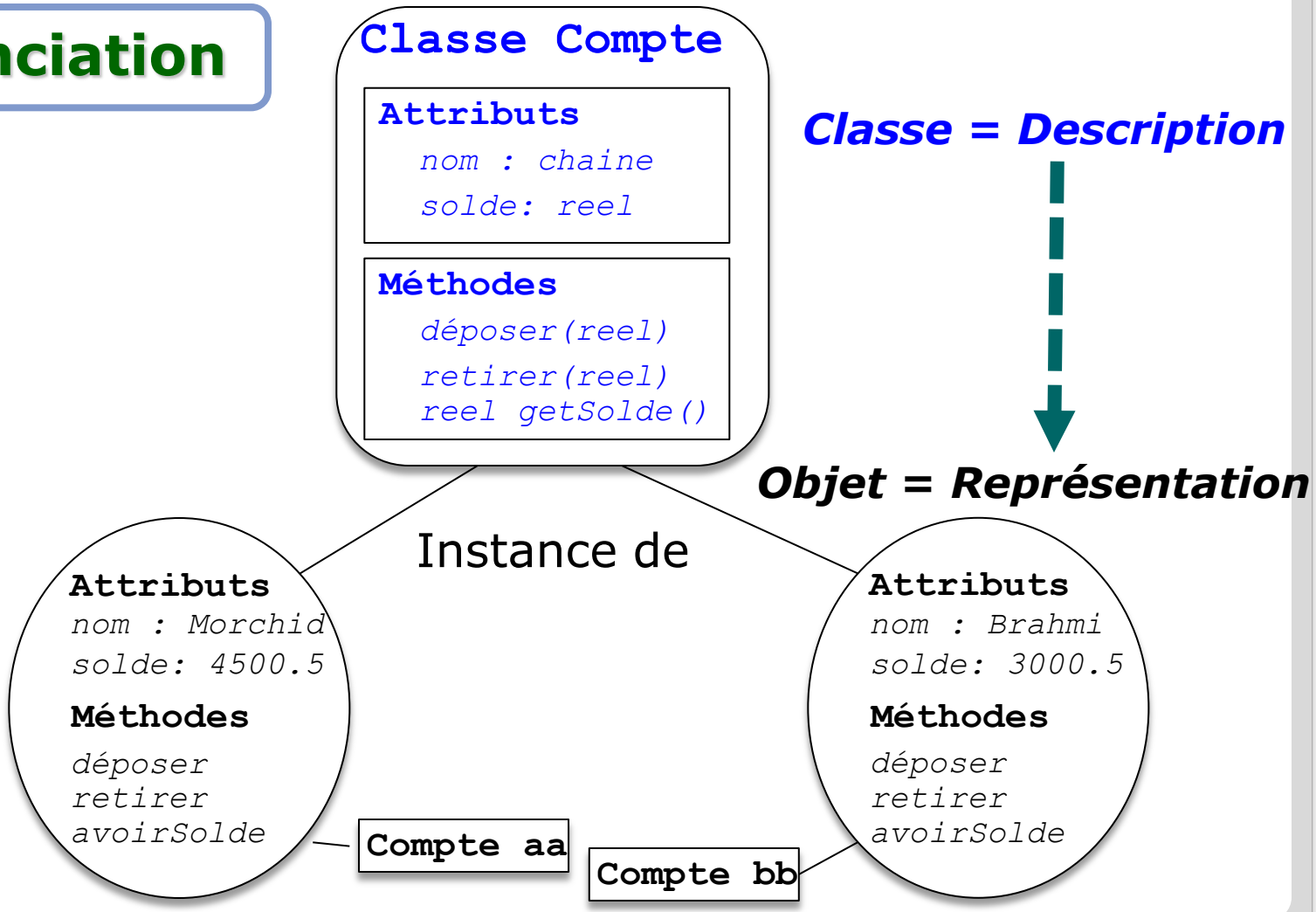
Héritage

Polymorphisme

Liaison dynamique

Paradigme Objet

Instanciation

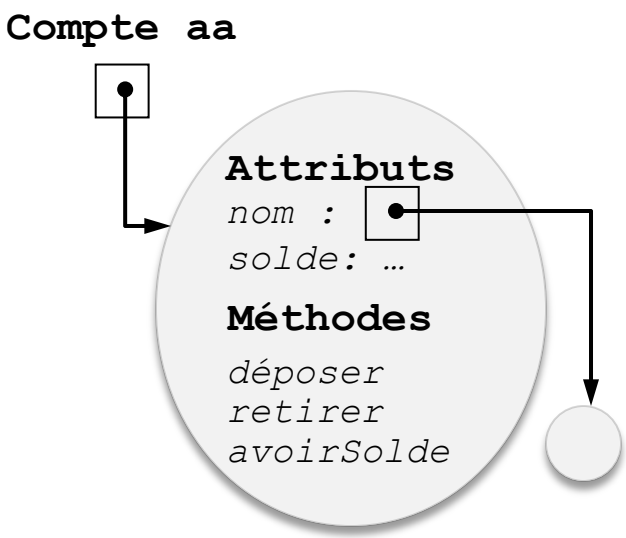


Paradigme Objet

Instanciation

➔ **Création des objets ?**

- Notion de constructeur



➔ **Destruction des objets?**

- Automatique ou non

Paradigme Objet

Langages OO

Langage Statique

- **Typage** = fort
- **Liaison statique** = oui
- **Liaison dynamique** = méthodes virtuelles
- **Usage** = applications finies

Langage Dynamique

- **Typage** = faible
- **Liaison statique** = non
- **Liaison dynamique** = oui
- **Usage** = prototypage, simulation graphisme

Simula 1960 | **Eiffel** 1980 | **C++** 1983 | **Java** 1995 | **SmallTalk** 1980 | **CLOS** 1980 | **LOOPS** 1983

Exercices d'examen

Ce qui sépare une classe d'un type de données abstrait est ..... C .....

A	*La notion de type*	**C**	*La notion d'héritage*
B	*L'encapsulation*	**D**	*Rien*

Dans une **démarche objet** ..... B .....

A	Le **`robot`** ouvre la **`porte`**	**C**	Le **`robot`** tourne le poignet pour ouvrir la **`porte`**
B	Le **`robot`** demande à la **`porte`** de s'ouvrir		

(**`robot`** et **`porte`** sont des objets de l'application)

ENSIAS

M.2.5.1. Paradigme Objet

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat