



PRÉSENTÉ À

Pr. Mariame Nassit

**3ème Année Licence Fondamentale ISI**

PRÉSENTÉ PAR

Anas Haddou

# Travaux pratique

Développement Orienté Objet : Programmation  
objet/Java

# Vue d'ensemble

- 1 TP : Classes et Objets

---

- 2 TP : Constructeurs

---

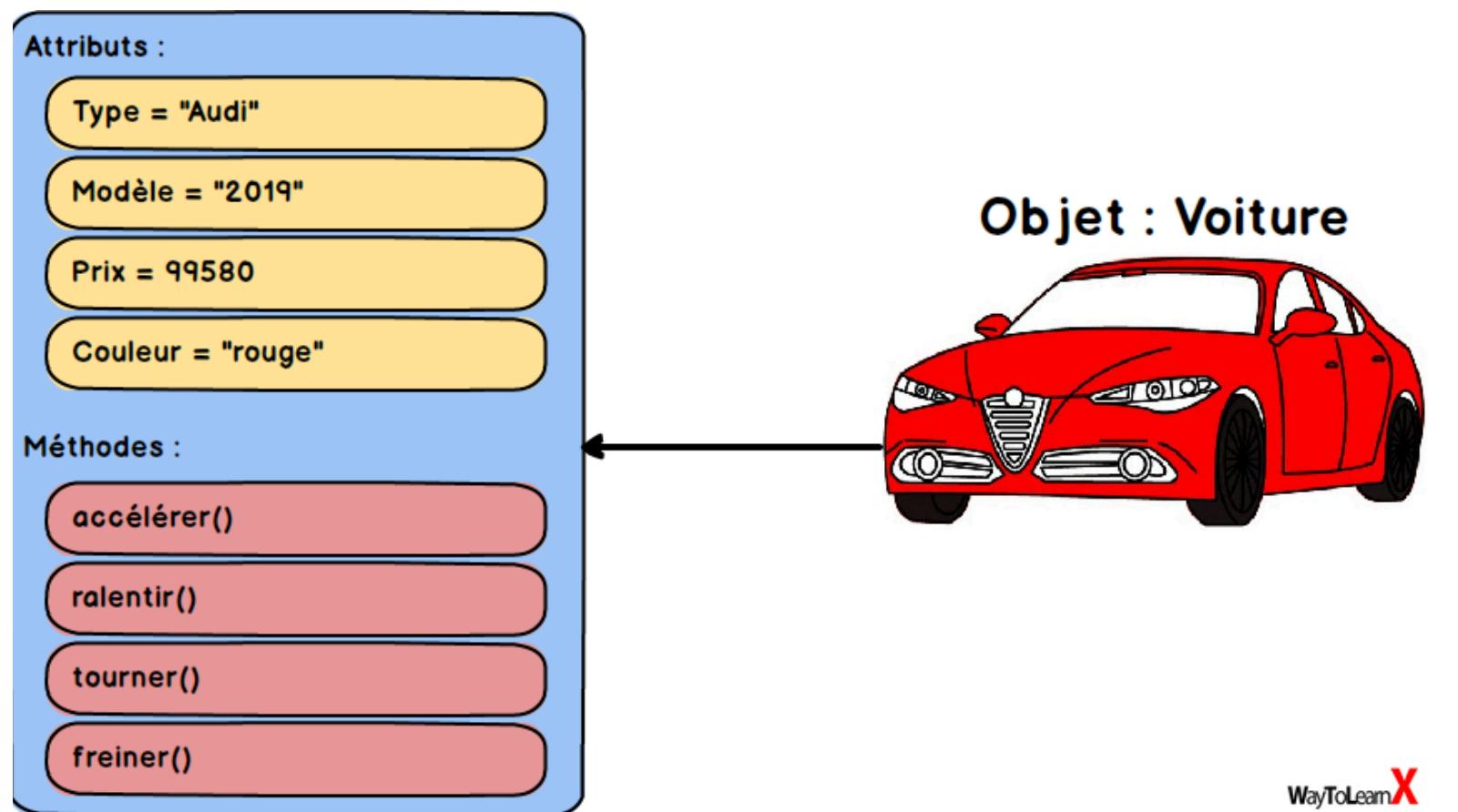
- 3 TP : Héritage

---

- 4 TP : Polymorphisme

---

# Classes et Objets



- Java est un langage orienté objet. Tout élément doit se trouver à l'intérieur d'une classe.

# EXERCICE 1 : SURCHARGE DE MÉTHODES EN JAVA



```
1  public class Calculatrice {  
2  
3      public int additionner(int a, int b) {  
4          return a + b;  
5      }  
6  
7      public int additionner(int a, int b, int c) {  
8          return a + b + c;  
9      }  
10  
11  
12     public void afficherResultat(int resultat) {  
13         System.out.println("Le résultat du calcul est : " + resultat);  
14     }  
15  
16     public static void main(String[] args) {  
17         Calculatrice calc = new Calculatrice();  
18  
19  
20         int resultat1 = calc.additionner(5, 10);  
21         calc.afficherResultat(resultat1);  
22  
23         int resultat2 = calc.additionner(5, 10, 15);  
24         calc.afficherResultat(resultat2);  
25     }  
26 }
```

## EXERCICE 2 : CLASSE PERSONNE AVEC ATTRIBUTS STATIQUES



```
1  public class Personne {  
2  
3      private static int compteur = 0;  
4  
5      private String nom;  
6  
7      public Personne(String nom) {  
8          this.nom = nom;  
9          compteur++;  
10     }  
11  
12     public void afficherInfo() {  
13         System.out.println("Le nom de la personne est : " + nom);  
14     }  
15  
16     public static void afficherCompteur() {  
17         System.out.println("Nombre total de personnes créées : " + compteur);  
18     }  
19  
20     public static void main(String[] args) {  
21  
22         Personne personne1 = new Personne("Alice");  
23         Personne personne2 = new Personne("Bob");  
24         Personne personne3 = new Personne("Charlie");  
25  
26         personne1.afficherInfo();  
27         personne2.afficherInfo();  
28         personne3.afficherInfo();  
29  
30         Personne.afficherCompteur();  
31     }}}
```

## EXERCICE 3: CALCUL DE L'AIRE D'UN CERCLE AVEC UNE MÉTHODE STATIQUE

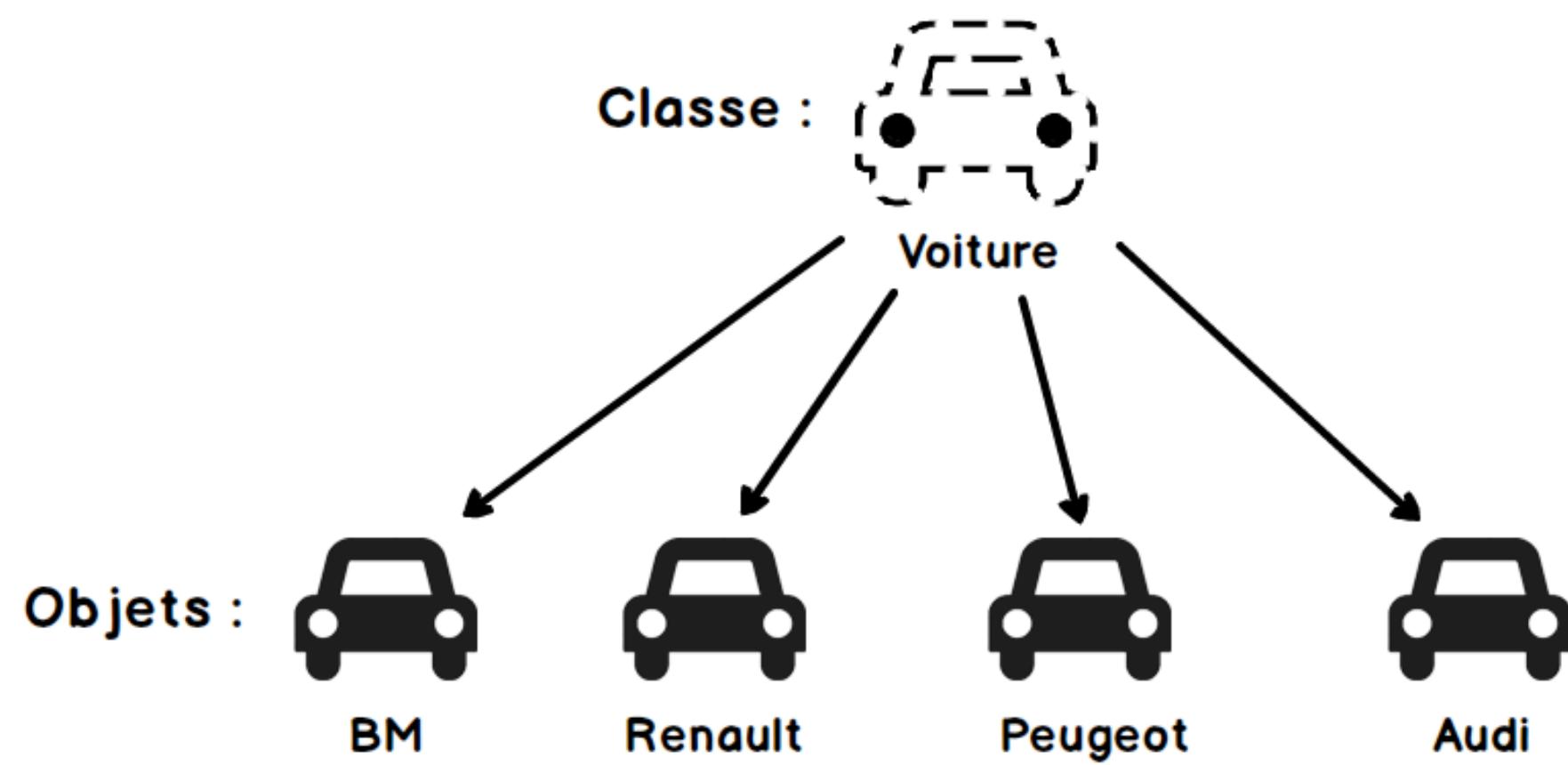


```
1 public class Cercle {  
2  
3     public static final double PI = 3.14159;  
4  
5     private double rayon;  
6  
7     public Cercle(double rayon) {  
8         this.rayon = rayon;  
9     }  
10  
11    public static double calculerAire(double rayon) {  
12        return PI * rayon * rayon;  
13    }  
14  
15    public void afficherInfos() {  
16        System.out.println("Cercle de rayon : " + rayon);  
17        System.out.println("Aire du cercle : " + calculerAire(rayon));  
18    }  
19  
20    public static void main(String[] args) {  
21        Cercle cercle = new Cercle(5.0);  
22  
23        cercle.afficherInfos();  
24  
25        double aire = Cercle.calculerAire(3.0);  
26        System.out.println("Aire d'un cercle de rayon 3.0 : " + aire);  
27    }  
28}
```

# EXERCICE 4 : GESTION D'UN COMPTE BANCAIRE AVEC ATTRIBUTS STATIQUES

```
● ○ ●  
1 public class CompteBancaire {  
2  
3     private static double tauxInteret = 0.05;  
4  
5     private double solde;  
6  
7     public CompteBancaire(double soldeInitial) {  
8         this.solde = soldeInitial;  
9     }  
10  
11    public void ajouterInteret() {  
12        double interet = solde * tauxInteret;  
13        solde += interet;  
14    }  
15  
16    public void afficherSolde() {  
17        System.out.println("Solde actuel : " + solde);  
18    }  
19  
20    public static void setTauxInteret(double nouveauTaux) {  
21        tauxInteret = nouveauTaux;  
22    }  
23  
24    public static void main(String[] args) {  
25  
26        CompteBancaire compte1 = new CompteBancaire(1000.0);  
27        CompteBancaire compte2 = new CompteBancaire(2000.0);  
28  
29        System.out.println("Soldes initiaux :");  
30        compte1.afficherSolde();  
31        compte2.afficherSolde();  
32  
33        compte1.ajouterInteret();  
34        compte2.ajouterInteret();  
35  
36        System.out.println("\nSoldes après ajout des intérêts :");  
37        compte1.afficherSolde();  
38        compte2.afficherSolde();  
39  
40        CompteBancaire.setTauxInteret(0.10);  
41  
42        compte1.ajouterInteret();  
43        compte2.ajouterInteret();  
44  
45        System.out.println("\nSoldes après modification du taux d'intérêt :");  
46        compte1.afficherSolde();  
47        compte2.afficherSolde();  
48    }  
49}
```

# Constructeurs



- Un constructeur est une méthode qui n'a pas de type de retour et qui porte le même nom que la classe.

## EXERCICE 1 :



```
1 public class Personne {  
2  
3     private String nom;  
4     private int age;  
5  
6     public Personne(String nom, int age) {  
7         this.nom = nom;  
8         this.age = age;  
9     }  
10  
11    public void afficherDetails() {  
12        System.out.println("Nom : " + nom);  
13        System.out.println("Âge : " + age);  
14    }  
15 }
```



```
1 public class TestPersonne {  
2  
3     public static void main(String[] args) {  
4         Personne personne1 = new Personne("Alice", 25);  
5         Personne personne2 = new Personne("Bob", 30);  
6  
7         System.out.println("Détails de la première personne :");  
8         personne1.afficherDetails();  
9  
10        System.out.println("\nDétails de la deuxième personne :");  
11        personne2.afficherDetails();  
12    }  
13 }
```

## EXERCICE 2 :



```
1 public class Rectangle {  
2  
3     private int longueur;  
4     private int largeur;  
5     public Rectangle() {  
6         this.longueur = 1;  
7         this.largeur = 1;  
8     }  
9     public Rectangle(int longueur, int largeur) {  
10        this.longueur = longueur;  
11        this.largeur = largeur;  
12    }  
13    public int calculerSurface() {  
14        return longueur * largeur;  
15    }  
16    public void afficherDetails() {  
17        System.out.println("Longueur : " + longueur);  
18        System.out.println("Largeur : " + largeur);  
19        System.out.println("Surface : " + calculerSurface());  
20    }  
21 }
```



```
1 public class TestRectangle {  
2  
3     public static void main(String[] args) {  
4         Rectangle rectangle1 = new Rectangle();  
5         System.out.println("Détails du premier rectangle  
(constructeur par défaut) :");  
6         rectangle1.afficherDetails();  
7  
8         Rectangle rectangle2 = new Rectangle(5, 10);  
9         System.out.println("\nDétails du deuxième rectangle  
(constructeur paramétré) :");  
10        rectangle2.afficherDetails();  
11    }  
12 }
```

## EXERCICE 3 :



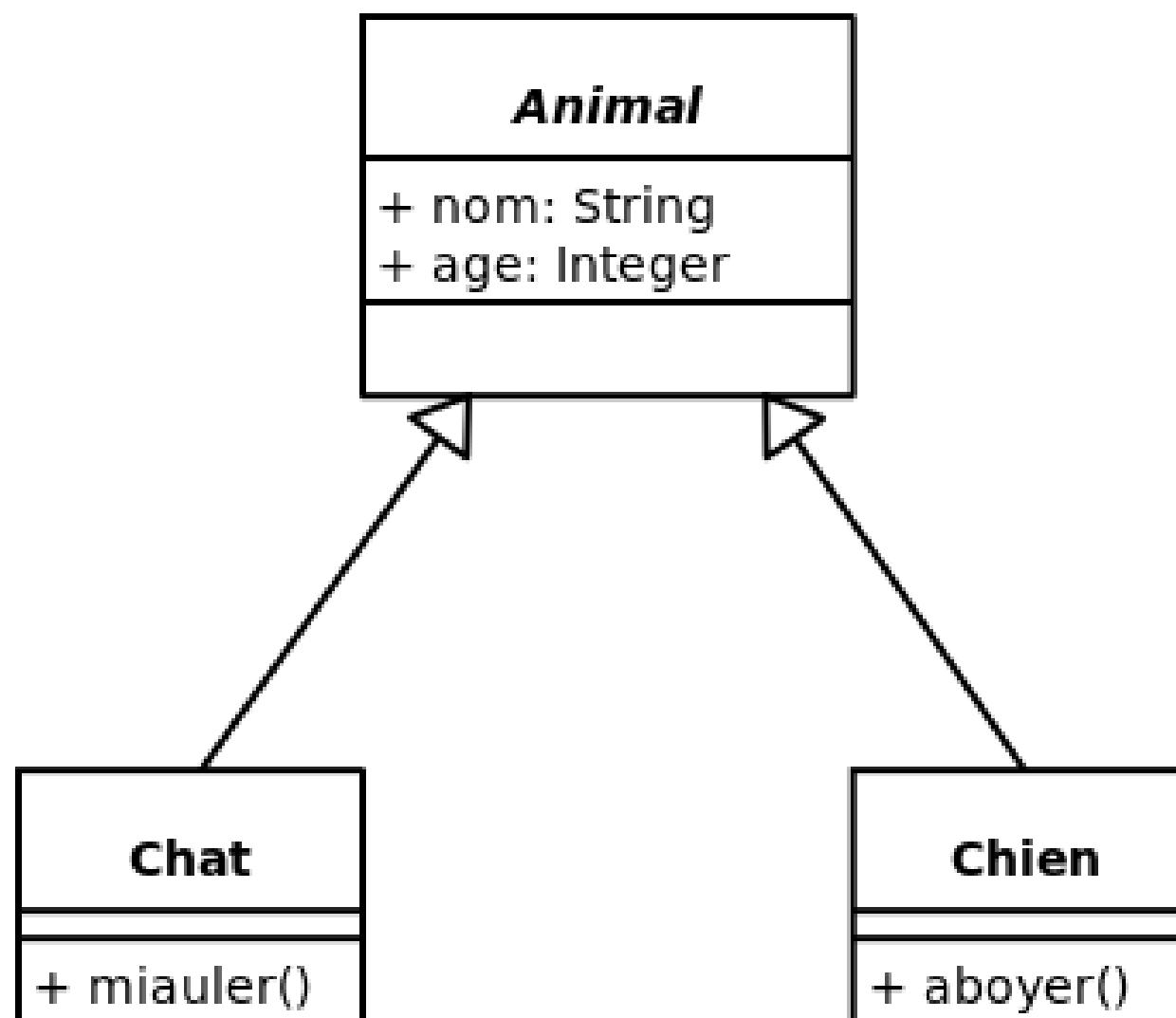
```
1  public class Point {  
2      private int x;  
3      private int y;  
4  
5      public Point() {  
6          this.x = 0;  
7          this.y = 0;  
8      }  
9  
10     public Point(int x, int y) {  
11         this.x = x;  
12         this.y = y;  
13     }  
14  
15     public Point(Point autrePoint) {  
16         this.x = autrePoint.x;  
17         this.y = autrePoint.y;  
18     }  
19  
20     public void afficherDetails() {  
21         System.out.println("Coordonnées du point : (" + x + ", " + y + ")");  
22     }  
23 }
```



```
1  public class TestPoint {  
2  
3      public static void main(String[] args) {  
4          Point point1 = new Point();  
5          System.out.println("Point 1 (constructeur par défaut) :");  
6          point1.afficherDetails();  
7  
8          Point point2 = new Point(3, 4);  
9          System.out.println("\nPoint 2 (constructeur paramétré) :");  
10         point2.afficherDetails();  
11  
12         Point point3 = new Point(point2);  
13         System.out.println("\nPoint 3 (constructeur de copie,"  
14             " copie de point2) :");  
15         point3.afficherDetails();  
16     }  
17 }
```

-

# Héritage



- Un constructeur est une méthode qui n'a pas de type de retour et qui porte le même nom que la classe.

## EXERCICE 1 :



```
1 public class Personne {  
2  
3     private String nom;  
4     private String prenom;  
5  
6     public Personne(String nom, String prenom) {  
7         this.nom = nom;  
8         this.prenom = prenom;  
9     }  
10  
11    public void afficher() {  
12        System.out.println("Nom : " + nom);  
13        System.out.println("Prénom : " + prenom);  
14    }  
15 }
```



```
1 public class Etudiant extends Personne {  
2  
3     private String cne;  
4  
5     public Etudiant(String nom, String prenom, String cne) {  
6         super(nom, prenom);  
7         this.cne = cne;  
8     }  
9  
10  
11    @Override  
12    public void afficher() {  
13        super.afficher();  
14        System.out.println("CNE : " + cne);  
15    }  
16 }
```



```
1 public class TestPersonneEtudiant {  
2  
3     public static void main(String[] args) {  
4         Personne personne = new Personne("Dupont", "Jean");  
5         System.out.println("Détails de la personne :");  
6         personne.afficher();  
7  
8         Etudiant etudiant = new Etudiant("Martin", "Alice", "A123456");  
9         System.out.println("\nDétails de l'étudiant :");  
10        etudiant.afficher();  
11    }  
12 }
```

## EXERCICE 2 :

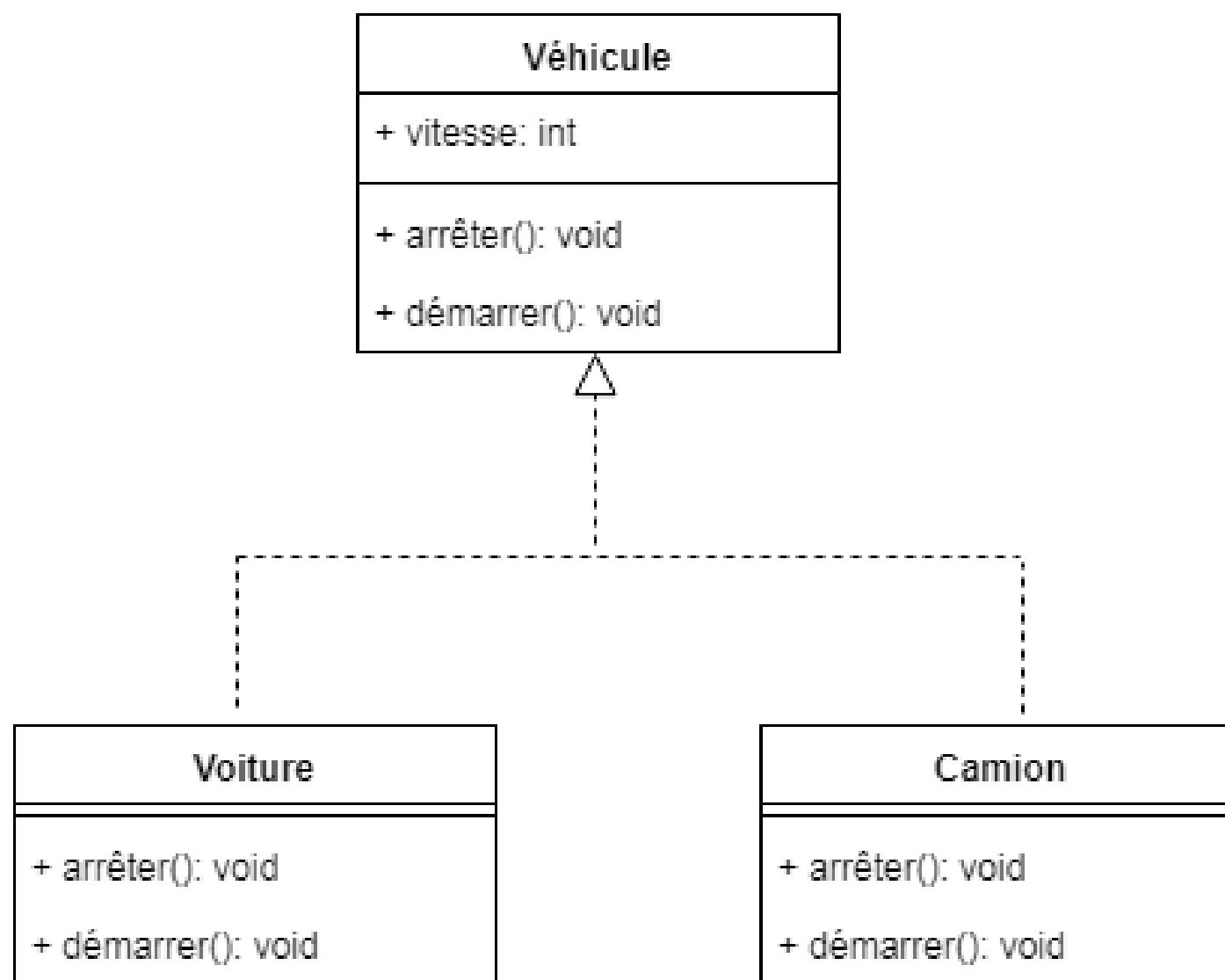
```
● ● ●  
1 public class ChevalCourse extends Cheval {  
2  
3     private int nombreCourses;  
4  
5     public ChevalCourse(String nom, String couleur,  
6                          int anneeNaissance,  
7                          int nombreCourses) {  
8         super(nom, couleur, anneeNaissance);  
9         this.nombreCourses = nombreCourses;  
10    }  
11  
12    public int getNombreCourses() {  
13        return nombreCourses;  
14    }  
15  
16    public void setNombreCourses(int nombreCourses) {  
17        this.nombreCourses = nombreCourses;  
18    }  
19  
20    @Override  
21    public void afficherDetails() {  
22        super.afficherDetails();  
23        System.out.println("Nombre de courses terminées : "  
24                    + nombreCourses);  
25    }  
26}
```

```
● ● ●  
1 public class Cheval {  
2     private String nom;  
3     private String couleur;  
4     private int anneeNaissance;  
5     public Cheval(String nom, String couleur, int anneeNaissance) {  
6         this.nom = nom;  
7         this.couleur = couleur;  
8         this.anneeNaissance = anneeNaissance;  
9     }  
10    public String getNom() {  
11        return nom;  
12    }  
13    public void setNom(String nom) {  
14        this.nom = nom;  
15    }  
16    public String getCouleur() {  
17        return couleur;  
18    }  
19    public void setCouleur(String couleur) {  
20        this.couleur = couleur;  
21    }  
22    public int getAnneeNaissance() {  
23        return anneeNaissance;  
24    }  
25    public void setAnneeNaissance(int anneeNaissance) {  
26        this.anneeNaissance = anneeNaissance;  
27    }  
28    public void afficherDetails() {  
29        System.out.println("Nom : " + nom);  
30        System.out.println("Couleur : " + couleur);  
31        System.out.println("Année de naissance : " + anneeNaissance);  
32    }  
33}
```



```
1
2 public class TestChevaux {
3
4     public static void main(String[] args) {
5         Cheval cheval = new Cheval("Bella", "Noir", 2015);
6         System.out.println("Détails du cheval :");
7         cheval.afficherDetails();
8
9         ChevalCourse chevalCourse = new ChevalCourse("Flash", "Blanc", 2018, 10);
10        System.out.println("\nDétails du cheval de course :");
11        chevalCourse.afficherDetails();
12
13        chevalCourse.setNombreCourses(15);
14        System.out.println("\nDétails du cheval de course après modification :");
15        chevalCourse.afficherDetails();
16    }
17 }
```

# Polymorphisme



- Une même méthode peut avoir des définitions différentes en fonction de la classe dans laquelle elle est utilisée.

## EXERCICE 1 :



```
1 public class TestPolymorphisme {
2
3     public static void main(String[] args) {
4         Animal[] animaux = new Animal[3];
5
6         animaux[0] = new Chien();
7         animaux[1] = new Chat();
8         animaux[2] = new Oiseau();
9
10        for (Animal animal : animaux) {
11            animal.parle();
12        // Polymorphisme : la méthode parle() appelée dépend du type réel de l'objet
13        }
14    }
15 }
```



```
1  public class Animal {
2
3      public void parle() {
4          System.out.println("L'animal fait un bruit.");
5      }
6  }
7
8  public class Chien extends Animal {
9
10     @Override
11     public void parle() {
12         System.out.println("Le chien aboie : Woof Woof !");
13     }
14 }
15
16 public class Chat extends Animal {
17
18     @Override
19     public void parle() {
20         System.out.println("Le chat miaule : Meow Meow !");
21     }
22 }
23
24 public class Oiseau extends Animal {
25
26     @Override
27     public void parle() {
28         System.out.println("L'oiseau chante : Cui Cui !");
29     }
30 }
```

## EXERCICE 2 :

```
● ● ●  
1 public class Employe {  
2     private String nom;  
3     private int id;  
4  
5     public Employe(String nom, int id) {  
6         this.nom = nom;  
7         this.id = id;  
8     }  
9  
10    public void afficheDetails() {  
11        System.out.println("Nom : " + nom);  
12        System.out.println("ID : " + id);  
13    }  
14}  
15  
16 public class Manager extends Employe {  
17  
18     private String departement;  
19  
20     public Manager(String nom, int id, String departement) {  
21         super(nom, id);  
22         this.departement = departement;  
23     }  
24  
25     @Override  
26     public void afficheDetails() {  
27         super.afficheDetails();  
28         System.out.println("Rôle : Manager");  
29         System.out.println("Département : " + departement);  
30     }  
31 }  
32 }
```

```
● ● ●  
1 public class Developpeur extends Employe {  
2  
3     private String langage;  
4  
5     // Constructeur  
6     public Developpeur(String nom, int id, String langage) {  
7         super(nom, id);  
8         this.langage = langage;  
9     }  
10  
11    @Override  
12    public void afficheDetails() {  
13        super.afficheDetails();  
14        System.out.println("Rôle : Développeur");  
15        System.out.println("Langage de programmation : " + langage);  
16    }  
17 }
```

```
● ● ●  
1  
2 public class TestEmployes {  
3  
4     public static void main(String[] args) {  
5         Employe employe1 = new Manager("Alice Dupont", 101, "Ressources Humaines");  
6         Employe employe2 = new Developpeur("Bob Martin", 102, "Java");  
7  
8         System.out.println("Détails du Manager :");  
9         employe1.afficheDetails();  
10        System.out.println("\nDétails du Développeur :");  
11        employe2.afficheDetails();  
12    }  
13 }
```

Merci  
à vous

