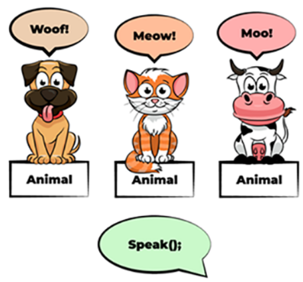
Le Polymorphisme



Le polymorphisme est un autre concept fondamental de la programmation orientée objet (OOP) en Java. Il permet aux objets de prendre plusieurs formes et d'utiliser les méthodes en fonction du contexte.

Le polymorphisme permet à une seule interface d'être utilisée pour représenter différentes types de données ou classes.

En Java, le polymorphisme peut être principalement de deux types : le polymorphisme à la compilation (polymorphisme statique) et le polymorphisme à l'exécution (polymorphisme dynamique).

Polymorphisme à la Compilation (Polymorphisme Statique)

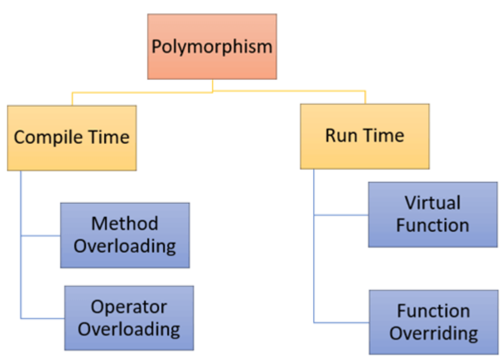


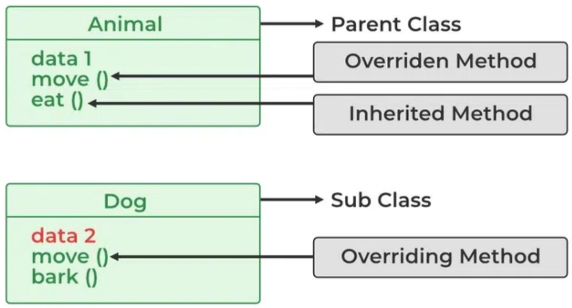
Figure 1: "All non-static functions are Virtual in Java"

Le polymorphisme statique est obtenu par la surcharge de méthodes. Cela signifie que plusieurs méthodes dans une classe peuvent avoir le même nom mais avec des paramètres différents (types et/ou nombre de paramètres).

Exemple de Surcharge de Méthodes

```java
public class MathUtil {
    // Surcharge de la méthode add pour deux entiers
    public int add(int a, int b) {
        return a + b;
    }

    // Surcharge de la méthode add pour trois entiers
    public int add(int a, int b, int c) {
        return a + b + c;
    }

    // Surcharge de la méthode add pour deux doubles
    public double add(double a, double b) {
        return a + b;
    }
}

public class Main {
    public static void main(String[] args) {
        MathUtil mathUtil = new MathUtil();

        // Appelle la première méthode add(int, int):
        System.out.println(mathUtil.add(5, 3));

    // Appelle la deuxième méthode add(int, int, int):
        System.out.println(mathUtil.add(5, 3, 2));

        // Appelle la troisième méthode add(double, double):
        System.out.println(mathUtil.add(5.0, 3.0));
    }
}
```

Polymorphisme à l'Exécution (Polymorphisme Dynamique)



Le polymorphisme dynamique est réalisé par le biais de l'héritage et de la substitution de méthodes (override).

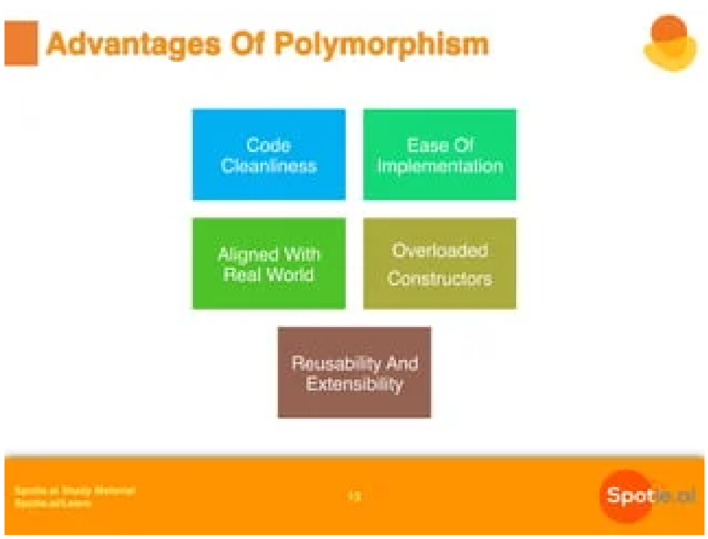
Une méthode de la classe parente est redéfinie dans une classe enfant, permettant à un objet de la classe enfant d'être traité comme un objet de la classe parente tout en utilisant la méthode redéfinie.

Exemple de Substitution de Méthodes

```java
// Classe de base (super-classe)
class Animal {
    void makeSound() {
        System.out.println("Some generic animal sound");
    }
}

// Sous-classe (classe dérivée)
class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Woof");
    }
}

// Autre sous-classe (classe dérivée)
class Cat extends Animal {
    @Override
    void makeSound() {
```

```java
            System.out.println("Meow");
        }
    }

public class Main {
    public static void main(String[] args) {
        Animal myAnimal;

        myAnimal = new Dog();
        myAnimal.makeSound();  // Appelle la méthode makeSound de Dog

        myAnimal = new Cat();
        myAnimal.makeSound();  // Appelle la méthode makeSound de Cat
    }
}
```

Avantages du Polymorphisme



1. **Flexibilité et Extensibilité** : Il permet d'écrire des programmes plus flexibles et extensibles. Par exemple, vous pouvez ajouter de nouvelles classes dérivées sans modifier le code existant.
2. **Réutilisation de Code** : Les méthodes et les classes peuvent être réutilisées plus facilement.
3. **Maintenance Simplifiée** : La maintenance du code est plus facile car les modifications dans la classe parente se répercutent automatique-

ment sur toutes les sous-classes.

Conclusion

En résumé, le polymorphisme est un outil puissant en Java qui permet d'écrire des programmes plus robustes, flexibles et faciles à maintenir.