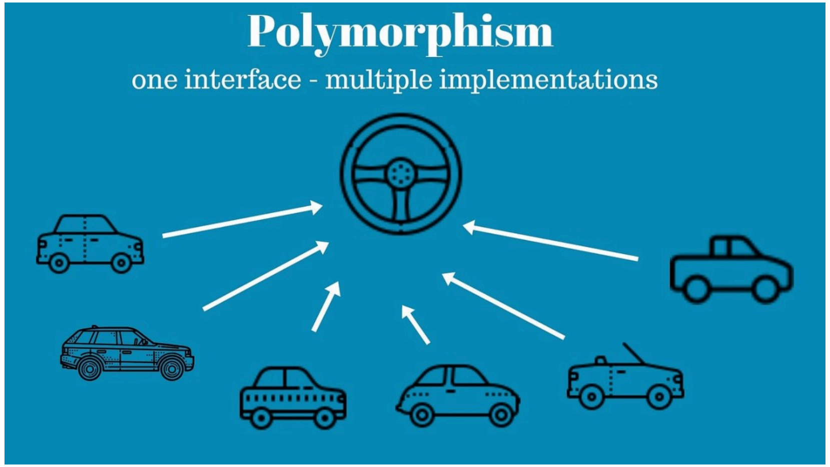
Java OOP: Polymorphism, Method Overloading and Overriding

Press Space for next page →

Agenda

Polymorphism

- Polymorphism is derived from **2 Greek words**: poly and morphs. The word **"poly" means many** and **"morphs" means forms**. So polymorphism means many forms.
- Polymorphism allows objects of different classes to be treated as objects of a common super class.
- The most common use of polymorphism in Java is when a parent class reference is used to refer to a child class object.

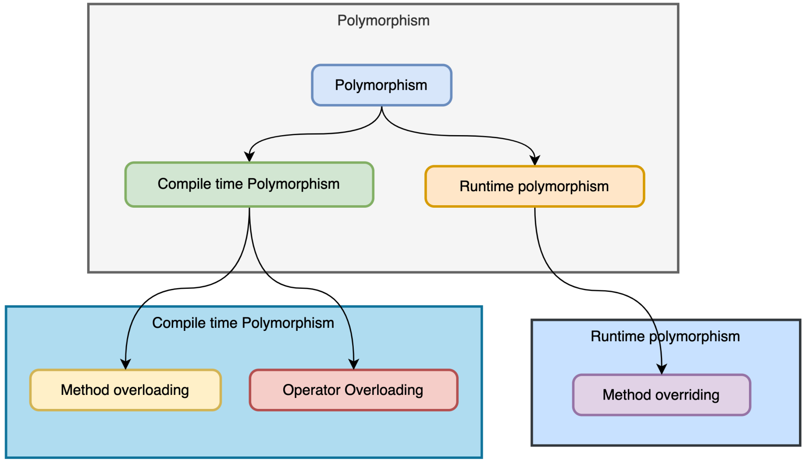


Types of Polymorphism

There are two types of polymorphism in Java

- Compile-time polymorphism
- Run-time polymorphism

1. We can perform **compile-time polymorphism by method overloading**.
2. We can perform **run-time polymorphism by method overriding**.
3. ***Java does not support operator overloading.***



Early Binding (Compile-time Polymorphism)

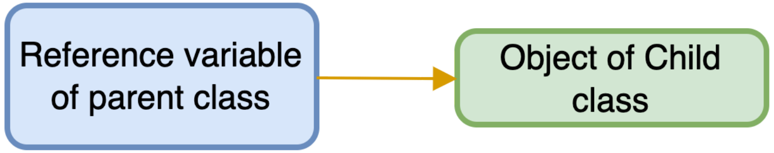
- Achieved through **method overloading**, where multiple methods have the same name but different parameters within the same class.
- The method to be executed is determined at **compile time** based on the method signature.

Runtime Polymorphism

- **Runtime polymorphism** or **Dynamic Method Dispatch** is a process in which a call to an overridden method is resolved at runtime rather than compile-time.
- In this process, an overridden method is called through the **reference variable of a superclass**.

Upcasting

> Upcasting is the typecasting of a child object to a parent object.



```java
class A{}
class B extends A{}
A a = new B(); // Upcasting
```

Code Example of Runtime Polymorphism

```java
class Animal {
    void sound() {
        System.out.println("Animals make sound");
    }
}
class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}
class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}
public class Main {
    public static void main(String[] args) {
        Animal myAnimal;

        myAnimal = new Dog();
        myAnimal.sound();  // Calls Dog's sound method

        myAnimal = new Cat();
        myAnimal.sound();  // Calls Cat's sound method
    }
}
```

Method Overloading

> If a class has multiple methods having **same name but different in parameters**, it is known as Method Overloading.

There are two ways to overload the method in Java:

1. By changing the number of arguments
2. By changing the data type

Code Example of Method Overloading

Changing the number of arguments

```
class Adder {
    static int add(int a, int b) {
        return a + b;
    }
    static int add(int a, int b, int c) {
        return a + b + c;
    }
}
```

Changing the data type

```
class Adder {
    static int add(int a, int b) {
        return a + b;
    }
    static double add(double a, double b) {
        return a + b;
    }
}
```

Method Overriding

> If subclass (child class) has the **same method as declared in the parent class**, it is known as method overriding in Java.

Usage of Java Method Overriding:

- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

- The method **must have the same name** as in the parent class
- The method **must have the same parameter** as in the parent class.
- There **must be an IS-A relationship** (inheritance).

Thank you ❤️

qa-june-2024-automation-with-java-slides