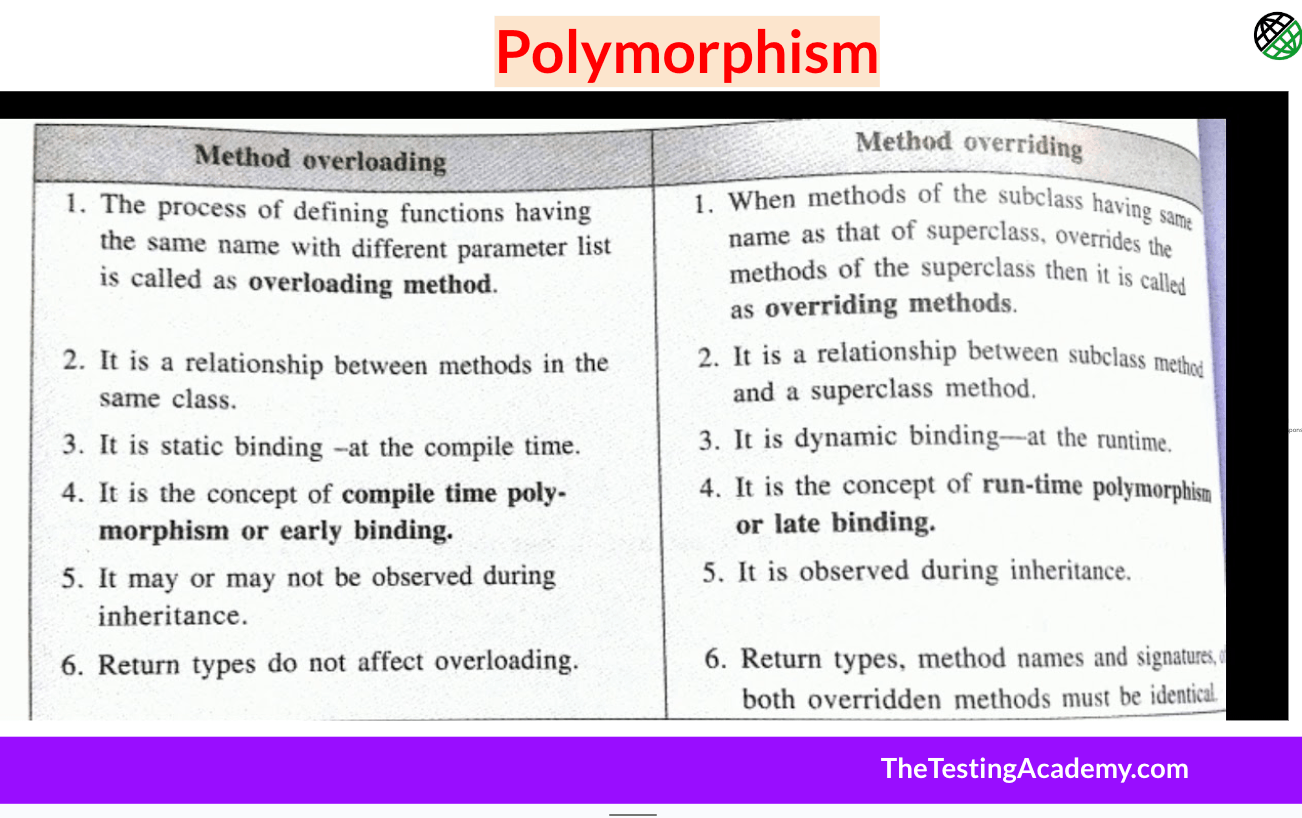
**Polymorphism**

* Polymorphism is the ability of an object to take on many forms.
* **Poly means ‘many’** and **morphism means ‘forms’.** when a parent class reference is used to refer to a child class object.
* Runtime Time
  + **Method Overriding**
* Compile Time
  + **Method Overloading**
  + Operator Overloading - This is not supported in Java

Declaring multiple method in same class with same method name but different in parameter can consider method loading.



**✅ Rules for Method Overriding**

**1. Method Signature Must Be the Same**

* The **method name**, **return type**, and **parameters** must be exactly the same in both parent and child classes.

✅ **Correct Example**

java

CopyEdit

class Parent {

void display() {

System.out.println("Parent method");

}

}

class Child extends Parent {

@Override

void display() { // Same method signature

System.out.println("Overridden method in Child");

}

}

❌ **Incorrect Example (Different Method Signature)**

java

CopyEdit

class Parent {

void display(int x) { // Different parameter

System.out.println("Parent method");

}

}

class Child extends Parent {

@Override

void display() { // ❌ No parameter, different signature (not overriding)

System.out.println("Overridden method in Child");

}

}

👉 **This is method overloading, NOT overriding.**

**2. Method Must Be Inherited (Not private or static)**

* **private methods** are **not inherited**, so they **cannot be overridden**.
* **static methods** are not overridden; they are **hidden** instead (method hiding).

❌ **Cannot Override a private Method**

java

CopyEdit

class Parent {

private void show() { // Private method, not inherited

System.out.println("Private method in Parent");

}

}

class Child extends Parent {

@Override

void show() { // ❌ This is NOT overriding, it's a new method

System.out.println("Method in Child class");

}

}

❌ **Cannot Override a static Method (Method Hiding)**

java

CopyEdit

class Parent {

static void display() {

System.out.println("Static method in Parent");

}

}

class Child extends Parent {

@Override

static void display() { // ❌ This is method hiding, not overriding

System.out.println("Static method in Child");

}

}

👉 **Static methods belong to the class, not an instance, so they are not overridden but hidden.**

**3. Access Modifier Cannot Be More Restrictive**

* The **access level** of the overridden method **cannot be more restrictive** than the method in the parent class.
* The visibility order in Java is:

java

CopyEdit

private → default (package-private) → protected → public

✅ **Allowed Access Modifier Changes**

java

CopyEdit

class Parent {

protected void display() {} // Protected method

}

class Child extends Parent {

@Override

public void display() {} // ✅ Increased access (protected → public)

}

❌ **Not Allowed (More Restrictive)**

java

CopyEdit

class Parent {

public void display() {}

}

class Child extends Parent {

@Override

private void display() {} // ❌ Compilation Error (Cannot reduce visibility)

}

👉 **The method in Child cannot have a more restrictive modifier than in Parent.**

**4. Return Type Must Be the Same or a Covariant Type**

* The return type **must be the same** or a **subtype (covariant return type)** of the parent method's return type.

✅ **Same Return Type**

java

CopyEdit

class Parent {

String getMessage() { return "Parent"; }

}

class Child extends Parent {

@Override

String getMessage() { return "Child"; } // ✅ Same return type

}

✅ **Covariant Return Type (Subtype)**

java

CopyEdit

class Parent {

Number getValue() { return 10; }

}

class Child extends Parent {

@Override

Integer getValue() { return 20; } // ✅ Integer is a subclass of Number

}

❌ **Different Return Type (Not a Subtype)**

java

CopyEdit

class Parent {

Number getValue() { return 10; }

}

class Child extends Parent {

@Override

String getValue() { return "Hello"; } // ❌ Compilation Error (String is not a subclass of Number)

}

**5. Method Must Be Non-Final**

* **Final methods cannot be overridden** because they are meant to be **unchangeable**.

❌ **Cannot Override a final Method**

java

CopyEdit

class Parent {

final void show() { // Final method

System.out.println("Final method in Parent");

}

}

class Child extends Parent {

@Override

void show() { // ❌ Compilation Error

System.out.println("Trying to override");

}

}

**6. super Can Be Used to Call Parent's Method**

* The super keyword is used inside the child class to call the parent class's overridden method.

java

CopyEdit

class Parent {

void display() {

System.out.println("Parent method");

}

}

class Child extends Parent {

@Override

void display() {

super.display(); // Calls Parent's method

System.out.println("Child method");

}

}

public class Main {

public static void main(String[] args) {

Child obj = new Child();

obj.display();

}

}

**Output:**

sql

CopyEdit

Parent method

Child method

**7. Overriding is Only for Instance Methods (Not Constructors)**

* **Constructors cannot be overridden**.
* However, the child class constructor can call the parent constructor using super().

java

CopyEdit

class Parent {

Parent() {

System.out.println("Parent Constructor");

}

}

class Child extends Parent {

Child() {

super(); // Calls Parent constructor

System.out.println("Child Constructor");

}

}

public class Main {

public static void main(String[] args) {

new Child(); // ✅ Calls Parent constructor first

}

}

**Output:**

nginx

CopyEdit

Parent Constructor

Child Constructor

**Summary of Overriding Rules**

| **Rule** | **Description** |
| --- | --- |
| **Method Signature** | Must be exactly the same (name + parameters) |
| **Inheritance** | The method must be inherited (not private or static) |
| **Access Modifier** | Cannot be more restrictive than the parent method |
| **Return Type** | Must be the same or a **subtype (covariant return type)** |
| **Final Methods** | Cannot override a final method |
| **Static Methods** | Cannot override, only hidden (method hiding) |
| **Constructors** | Cannot be overridden |