File Name: Java_Inheritance_Method_Overriding.pdf

Method Overriding in Java

1. Introduction

Method Overriding হলো **child class** এর ability **parent class er method ke redefine** kora jabe **same signature** (name, parameters) diye। এটি **runtime polymorphism** support kore, এবং parent class er behavior ke child class specific behavior diye replace korte help kore।

---

2. Key Points

1. Method overriding হলো **runtime polymorphism**।
2. **Child class** parent class er method ke **same signature** diye override kore।
3. Overriding method er **return type** parent method er **same or covariant** hote hobe।
4. Overriding method **cannot reduce visibility** (e.g., parent method public → child method cannot be private)।
5. `@Override` **annotation** use kora best practice, compile time error prevent korte help kore।

---

3. Syntax

```java
class Parent {
    void display() {
        System.out.println("This is parent class method");
    }
}

class Child extends Parent {
    @Override
    void display() {
        System.out.println("This is child class method");
    }
}

public class TestOverriding {
    public static void main(String[] args) {
        Parent p = new Parent();
        p.display();  // Parent method

        Child c = new Child();
        c.display();  // Overridden method
        Parent p2 = new Child();
        p2.display(); // Runtime polymorphism
    }
}
```

Output:

```
This is parent class method
This is child class method
This is child class method
```

---

4. Explanation

1. `Child` class er `display()` method **parent er display() ke override kore।**
2. `Parent p2 = new Child();` → **Parent reference, child object**, method call **child er overridden method execute** kore (runtime polymorphism)।
3. Overriding er jonno **method name, parameters** same thakte hobe।

---

5. Rules for Method Overriding

1. **Same method name and parameter list**।
2. **Return type:** same or covariant।
3. **Access modifier:** cannot be more restrictive।
4. **Static methods cannot be overridden** (static binding)।
5. **Private methods cannot be overridden** (because they are not inherited)।

---

6. Example: Polymorphism with Overriding

```
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
```

```java
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

public class TestPolymorphism {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        Animal a2 = new Cat();

        a1.sound();  // Dog barks
        a2.sound();  // Cat meows
    }
}
```

Output:

```
Dog barks
Cat meows
```

Explanation: - Same parent type reference `Animal` but **child object er overridden method execute** kore। - এটি হলো **dynamic (runtime) polymorphism**।

---

7. Summary Table

Feature	Method Overriding
Purpose	Modify parent class behavior in child class
Occurs at	Runtime (dynamic polymorphism)
Signature	Must be same as parent method
Return Type	Same or covariant
Access Modifier	Cannot be more restrictive
Static/Private	Cannot override

---

8. Points to Remember

1. Method overriding **runtime polymorphism er base**।
2. Overridden method call depends on **object type**, not reference type।
3. `@Override` annotation always use kora best practice।
4. **Static, private, final methods** cannot be overridden।