Java Method Overriding: Access Modifiers, Abstract, Interface, Final

Method Overriding and Access Modifiers

1. public method:
   - Can be overridden in the subclass.
   - Overridden method must be public.

2. protected method:
   - Can be overridden as protected or public.
   - Cannot override as private or default.

3. private method:
   - Cannot be overridden.
   - Not visible in the subclass.

4. default (no modifier) method:
   - Can only be overridden in same package.

Java Code Example (Access Modifiers)

```
class Parent {
    public void showPublic() {
        System.out.println("Parent: public method");
    }

    protected void showProtected() {
        System.out.println("Parent: protected method");
    }

    void showDefault() {
        System.out.println("Parent: default method");
```

```
    }

    private void showPrivate() {
        System.out.println("Parent: private method");
    }
}

class Child extends Parent {
    @Override
    public void showPublic() {
        System.out.println("Child: public method overridden");
    }

    @Override
    public void showProtected() {
        System.out.println("Child: protected method overridden");
    }

    @Override
    void showDefault() {
        System.out.println("Child: default method overridden");
    }

    public void showPrivate() {
        System.out.println("Child: new method with same name as parent's private");
    }
}
```

Overriding Abstract Methods

Java Method Overriding: Access Modifiers, Abstract, Interface, Final

```
abstract class Animal {
    public abstract void makeSound();
}

class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Dog says: Woof!");
    }
}
```

Overriding Interface Methods

```
interface Vehicle {
    void start(); // implicitly public abstract
}

class Car implements Vehicle {
    @Override
    public void start() {
        System.out.println("Car starts with key ignition");
    }
}
```

Final Methods - Cannot Override

```
class Parent {
    public final void greet() {
        System.out.println("Hello from Parent");
    }
```

```
}

class Child extends Parent {
    // Compile error: Cannot override final method
}
```

Java 8+ Interface Default and Static

```
interface Printer {
    default void print() {
        System.out.println("Default printing");
    }
}

class HPPrinter implements Printer {
    @Override
    public void print() {
        System.out.println("HPPrinter custom print");
    }
}

interface TestStatic {
    static void run() {
        System.out.println("Running static in interface");
    }
}
```

Summary Table

| Modifier/Type | Can Override? | Notes |

Java Method Overriding: Access Modifiers, Abstract, Interface, Final

abstract method	Yes (must)	Must override in subclass
Interface method	Yes (must)	Must be public
default (interface)	Yes	Can override
static (interface/class)	No	Can be hidden, not overridden
final method	No	Cannot override
private method	No	Not inherited
public/protected method	Yes	As long as access not reduced