In Java, **all methods in an interface are implicitly abstract** and **public**, **unless they are**:

1. **Static methods** (introduced in Java 8)
2. **Default methods** (introduced in Java 8)
3. **Private methods** (introduced in Java 9, for reuse inside default/static methods)

So, to clarify:

* **Before Java 8**: Yes, all methods in an interface were abstract and public.
* **From Java 8 onward**: Interfaces can also have:
  + default methods with a body.
  + static methods with a body.
* **From Java 9 onward**: You can also define private methods for internal use in the interface.

**Example:**

java

CopyEdit

public interface MyInterface {

void abstractMethod(); // Implicitly public and abstract

default void defaultMethod() { // Has a body

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

}

static void staticMethod() { // Has a body

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

}

private void helperMethod() { // Java 9+

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

}

}

Would you like to see how these are used in a class that implements the interface?

4o

**You said:**

complasory override all the method?

**ChatGPT said:**

In Java, **it is not compulsory to override all methods of an interface in every case**—it depends on the type of methods in the interface and how the class is defined.

Here’s a breakdown:

**✅ You MUST override:**

* All **abstract methods** in the interface, **if your class is not abstract**.

java

CopyEdit

interface A {

void method1(); // abstract

}

class B implements A {

public void method1() {

System.out.println("Overridden");

}

}

**🚫 You do NOT need to override:**

* **default methods** – these have a body already.

java

CopyEdit

interface A {

default void method2() {

System.out.println("Default implementation");

}

}

class B implements A {

// Not required to override method2

}

**📝 If you don’t override all abstract methods:**

* Then the class must be declared abstract.

java

CopyEdit

abstract class B implements A {

// method1 not overridden, so class must be abstract

}

**Summary:**

| **Interface Method Type** | **Must Override?** | **Notes** |
| --- | --- | --- |
| Abstract method | ✅ Yes | Unless the class is abstract |
| Default method | ❌ No | Optional to override |
| Static method | ❌ No | Cannot be overridden |
| Private method | ❌ No | Only usable within the interface |

Would you like a real-world example combining all these types?

4o

**You said:**

📝 If you don’t override all abstract methods: You MUST override:give me example of both

**ChatGPT said:**

Sure! Here are **two clear examples** to explain:

**✅ Case 1: You MUST override all abstract methods (for a non-abstract class)**

java

CopyEdit

interface Vehicle {

void start(); // abstract method

void stop(); // abstract method

}

class Car implements Vehicle {

@Override

public void start() {

System.out.println("Car started");

}

@Override

public void stop() {

System.out.println("Car stopped");

}

}

* ✔️ All abstract methods are overridden.
* ✅ This compiles fine because Car is a concrete class and has implemented everything.

**📝 Case 2: If you don’t override all abstract methods, your class must be abstract**

java

CopyEdit

interface Vehicle {

void start(); // abstract method

void stop(); // abstract method

}

abstract class Bike implements Vehicle {

@Override

public void start() {

System.out.println("Bike started");

}

// ❌ stop() is NOT overridden here

// So this class must be abstract

}

* ⚠️ If you remove the abstract keyword from Bike, you’ll get a **compilation error**:

*Bike is not abstract and does not override abstract method stop() in Vehicle*

**Summary:**

| **Class Type** | **Overridden All Abstract Methods?** | **Compilation** |
| --- | --- | --- |
| Concrete class | ✅ Yes | ✅ OK |
| Concrete class | ❌ No | ❌ Error |
| Abstract class | ❌ No | ✅ OK |

Would you like an example using default methods as well?