@Override in Java - Why Use It?

Q: But without mentioning @Override it's working same, then why mention it?

A: Great observation bro! You're absolutely right - even without writing @Override, the program works the same. So your doubt is very genuine:

"If everything works without @Override, then why should we use it?"

Let me give you a clear and practical answer.

First: What does @Override do?

@Override is an annotation - a special instruction to the Java compiler.

It tells the compiler: "Hey compiler, I am trying to override a method from the parent class."

So what if we don't use it?

Yes, it will still work as long as:

- The method name in the child class exactly matches the parent method
- The parameters match
- The return type matches

But if you make a small mistake, Java will not catch it unless you use @Override.

Example: Without @Override, Bug Goes Undetected

Let's say you accidentally type the method name wrong:

```
    void showdetails() {   // typo: small 'd'
        super.showDetails();
        System.out.println("Department: " + department);
    }
```

Now:

- Java thinks this is a new method, not an override.
- m.showDetails(); will call the parent class version, not this one.
- You won't get any error, but your method won't run - and you won't even realize it.

But if you used @Override:

```
    @Override
    void showdetails() {   // typo
        ...
    }
```

Java compiler will immediately show an error:

```
    Method does not override method from its superclass
```

So now you know you made a mistake - and can fix it quickly!

Summary: Why to Use @Override

Reason	Benefit
Catches typos	You avoid silent bugs like wrong method names
Better readability	Anyone reading your code knows you're overriding a parent method
Safer code	Compiler helps you avoid mistakes
Recommended	Clean and maintainable code is important in real-world projects

Final Answer:

Even though the program works without @Override, you should always use it when overriding a method to avoid bugs and make your code cleaner and safer.