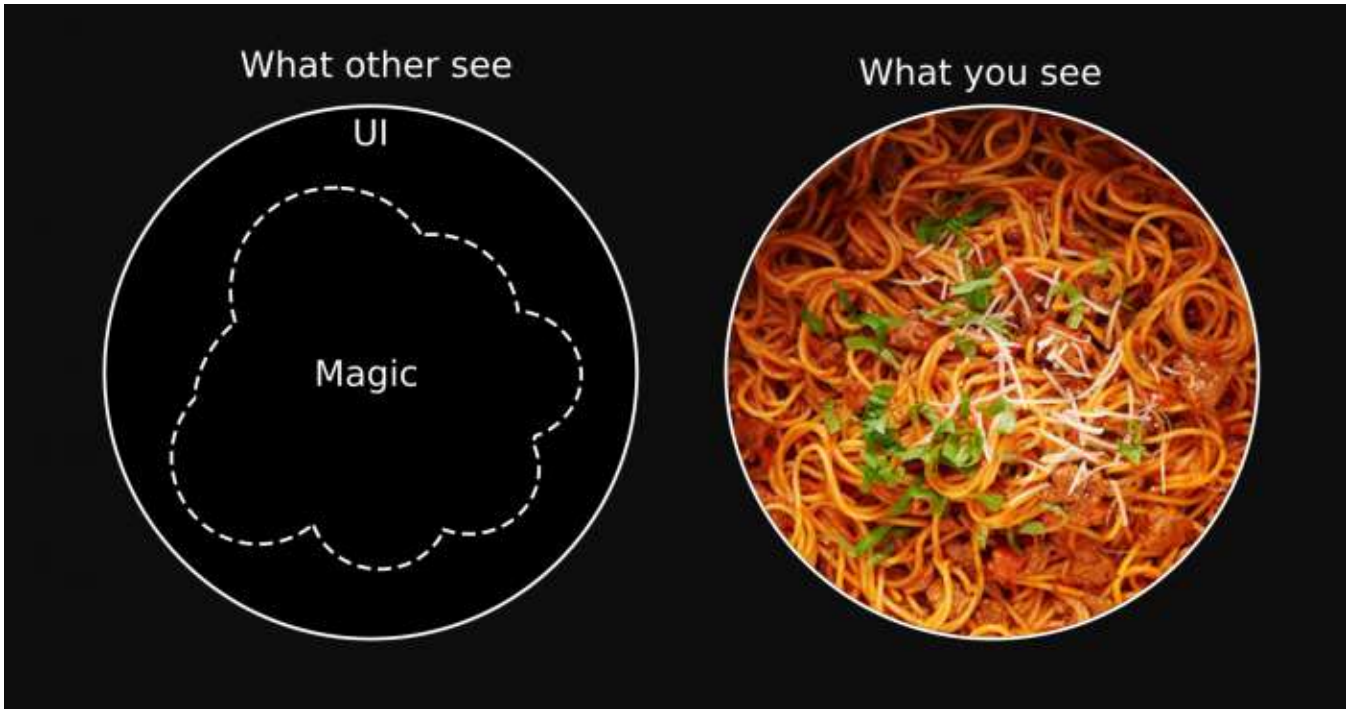
KD Knowledge Diet Follow
Apr 4 · 3 min read · Listen



# Interface Segregation Principle: Your only way to be a super competent developer




During development, the code tends to be long. At some point, the codebase becomes so bloated that you can't handle it. Nevertheless, senior developers make things work somehow. How do they do that? ***"All enterprise level developers actively use interfaces".*** So today, I am going to explain **Interface Segregation Principle** in one of SOL'**I**'D PRINCIPLE. **If you didn't know this, you should know that you are falling behind.**

## What is Interface Segregation Interface?

**Interface Segregation Principe** is also called **ISP** for abbreviation. When you write code, it's pretty easy to violate **ISP,** because your software evolves and you have to add more features. The aim of ISP is to minimize the side effects by splitting the software into independent parts. The point is **"YOU SHOULD ONLY DEFINE METHODS THAT ARE GOIND TO BE USED".**

## Bad Practices

```
interface IMultiFunction {
    public void print();
    public void getPrintSpoolDetails();
    public void scan();
    public void scanPhoto();
    public void fax();
    public void internetFax();
}
```

What do you think? Do you think the code example above has nothing wrong?

Firstly, you need to ask yourself if the methods defined inside `IMultiFunction` is cohesive. Obviously, `print()` and `getPrintSpoolDetails()` are cohesive. But how about `print()` and `scan()` ? Does all printing machines support for these features?

I'm going to show you another example.

```
class CanonPrinter implements IMultiFunction {

    @Override
    public void print() {}

    @Override
    public void getPrintSpoolDetails() {}

    /* This machine can't scan */
    @Override
```

```
    /* This machine can't fax */
    @Override
    public void fax() {}

     /* This machine can't fax on internet */
    @Override
    public void internetFax() {}

}
```

`CanonPrinter` can't `fax`. But it still needs to implement `fax()`.

In software development, ***"Unimplemented methods are almost always indicative of a poor design."***

So, what should you do?

## Good Practices

***ISP starts from splitting big interfaces into smaller interfaces.***

Let's refactor !

```
interface IPrint {
    public void print();
    public void getPrintSpoolDetails();
}

interface IScan {
    public void scan();
    public void scanPhoto();
}

interface IFax {
    public void fax();
    public void internetFax();
}
```


```
class HPPrintNScanner implements IPrint, IScan {

    @Override
    public void scan() {
        // TODO Auto-generated method stub

    }

    @Override
    public void scanPhoto() {
        // TODO Auto-generated method stub

    }

    @Override
    public void print() {
        // TODO Auto-generated method stub

    }

    @Override
    public void getPrintSpoolDetails() {
        // TODO Auto-generated method stub

    }

}
```

As you can see, `HPPrintNScanner` implemented only necessary methods.

## Techniques to identify violations

1. Find if your Interfaces have methods with low cohesion
2. Check Empty Method Implementations

## Conclusion

You've learned some great techniques that will increase the quality of your code. This is

## Other Articles for Solid Principles

**Single Responsibility Principle: What? working as “a software engineer” you don’t know it?!?!**

Have you ever experienced that code refactoring doesn’t make your code better? Adding spaghetti on top of spaghetti...

paigeshin1991.medium.com

**Open Closed Principle: Make your code cost-free and flexible**

Software is never dormant. It’s constantly changing. The annoying fact about the software development is that every...

paigeshin1991.medium.com

**Liskov Substitution Principle: Top Developer’s technique which improves 2.5x**

As you write code, doesn’t anyone ever question whether your code is logically correct or consistent enough? To solve...

paigeshin1991.medium.com

**Interface Segregation Principle: Your only way to be a super competent developer**

During development, the code tends to be long. At some point, the codebase becomes so bloated that you can’t handle it...

paigeshin1991.medium.com

**Dependency Inversion Principle: How Google Developers write code**

To become a high-paid developer, you need to learn TDD. Basically, you have to develop software with TDD to get into a...