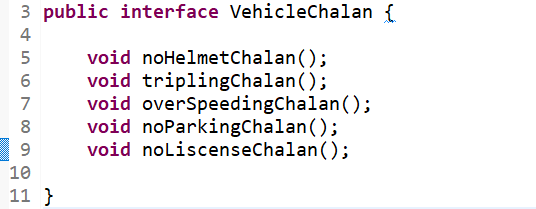
Segregation means keeping things separated, and the Interface Segregation Principle is about separating the interfaces.

The principle states that many client-specific interfaces are better than one general-purpose interface. Clients should not be forced to implement a function they do no need.

Let’s take an example,

Suppose we have an interface called **VehicleChalan**



Now, we have a class **BikeChalan** which is implementing **VehicleChalan**

Graphical user interface, text, application

Description automatically generated

As you can see that whatever method coming from the **VehicleChalan** interface is going to be use in **BikeChalan**. Because whatever chalan is mentioned in **VehicleChalan** interface is belong to bike.

But if some other class i.e. **CarChalan** is implements the **VehicleChalan**

Graphical user interface, text, application

Description automatically generated

Car do not have Helmet Chalan and Tripling Chalan restrictions, so this two method should not be there in this class. But if this **CarChalan** class implements **VehicleChalan** then unwillingly it has to override the **noHelmetChalan()** and **triplingChalan()** methods too.

And by doing this, a developer will be break the **ISP.**

**Instead what should be do ?**

We should segregate the **VehicleChalan** interface into two some other interfaces.

We will create two different interfaces i.e. **TwoWheelerChalan** and **FourWheelerChalan**

**Text, letter

Description automatically generated**

**Text

Description automatically generated**

Now, **Car** class should extends **FourWheelerChalan** and **Bike** class should extends **TwoWheelerChalan**

Now we are following **Interface Segregation Principle**