Interface Segmented Principle

Interface Segmented Principle also is one of the principles among the SOLID Principles. It works on the efficient implementation of Interface. We can define the formal definition of the principle as per below:

Interfaces should be implemented in such a way that client should not implement the unnecessary functions which are not needed to implement.

We will understand this principle using the simple example as per below:

Suppose there is an interface implementation of Restaurant Employees as per below:

```
//interface of restaurant employee
interface ResEmp
{
    //various methods of interface
    public void CookFood();
    public void ServeCustomers();
    public void WashDishes();
}
```

Now we will suppose that there is a class named Waiter which implements the interface of ResEmp.

```
class Waiter implements ResEmp
{
    public void CookFood()
    {
        // not job of waiter
    }

    public void ServeCustomers()
    {
        system.out.println("Serving the food");
    }

    public void WashDishes()
    {
        //not job of waiter
    }
}
```

Here we can say that the Waiter class has to implement the methods of **CookFood() and WashDishes(),** which is not the need of implementation because it is not the job of the Waiter and it is unnecessary to implement those methods. This is the reason it doesn't follow the **Interface Segmented Principle.**

The solution to this problem can be considered efficient segmentation of Interface in such a way that unnecessary implementation of methods can be avoided.

Example: Let us define two interfaces

```
public interface WaiterJob
{
    public void serveCustomers();
    public void AssignSeats();
}

public interface ChefJob
{
    public void CookFood();
    public void DecideMenu();
}
```

Here in these interfaces, we have only implemented the necessary methods. Now Chef class can implement ChefJob interface and Waiter class can implement WaiterJob interface. Which can lead to efficient implementation of classes and interfaces.

```
public class Waiter implements WaiterJob
{
    public void serveCustomers()
    {
        //logic
    }
    public void AssignSeats()
    {
        //logic
    }
}

public class Chef implements ChefJob
{
    public void CookFood()
    {
        //logic
    }
    public void DecideMenu()
    {
        //logic
    }
```

This is how we can understand the Interface Segmented Principle and can implement it for better understandability and development of software.