**The OCP states that a class should open for extension but close for modification.**

Modification means changing the code of an existing class, and extension means adding new functionality.

So what this principle wants to say is: We should be able to add new functionality without touching the existing code for the class. This is because whenever we modify the existing code, we are taking the risk of creating potential bugs. So we should avoid touching the tested and reliable (mostly) production code if possible.

But how are we going to add new functionality without touching the class. It is usually done with the help of interfaces and abstract classes.

Let’s look OCP into action,

Suppose we are working on project called **SHOPIFY**. Where we have a class called **Shop**

Text

Description automatically generated

In this class, you can see that we are selling the fruits as well as vegetables.

It’s seems okay for now but suppose in future the **SHOPIFY** project is going to sell clothes also then what developer will do is, he will add a new method called **puchaseCloth()**

And here developer is breaking the OCP. Because we should not modify the class.

Then what is better approach ?

We’ll create a interface called Shop

Graphical user interface, text

Description automatically generated

And we’ll separate all the different functionality that is purchasing fruit and vegetables into a separate class which will then implement the Shop interface.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

And now onwards every new category of product **SHOPIFY** wants to sell then developer just have create new class implementing the Shop interface.

Graphical user interface, text, application

Description automatically generated

Our main class

Graphical user interface

Description automatically generated with low confidence