Open-Closed Principle(OCP)

A class should be open for extension but closed for modification. This principle helps in making the code more maintainable and flexible.

It can be implemented by creating classes that can be extended by adding new functionality without modifying the existing code. This is achieved by using inheritance,polymorphsim, and interfaces.

Let's consider an example of a Shape class hierarchy that includes different types of shapes, such as Rectangle, Circle, and Triangle. Each shape has a different area calculation method.

```
Class Shape
{
  public:
     virtual double area()=0;
};

class Rectangle : public Shape
{
  private:
    double height;
    double width;
  public:
     Rectangle( double width, double height);
     double area();
};
class Circle : public Shape
{
  private:
    double radius;
  public:
     Circle(double radius)
     double area();
};
class Triangle : public Shape
{
   private:
       double base;
       double height;
   public:
         Triangle(double base, double height);
         double area();
};
```

Now, let's suppose that we want to add a new shape, such as a Square, to our hierarchy.

```
class Square : public shape
{
  private:
   double side;
  public:
     Square(double side);
     double area();
};

Square square(5.0);
double  squarearea = square.area();
```

By following open closed principle we can create code that is more maintainable,flexible, and easier to extend. This allows us to add new functionality to our code without modifying the existing code which reduces the risk of introducing bugs and makes the code more modular.