**Lesson02 Open Closed Principle**

**Notes:-**

**1-the Open-Closed Principle, which states that classes should be open for extension, but closed for modification. In other words, you should extend functionality using interfaces and inheritance rather than jumping back into already-written/tested code and adding to it or changing it.**

**(which means that each filter types is created in separated class that implement interface which used as an argument in other interface implemented by our better Filter)**

It use ISpecification as an parameter on the IFilter interface which means, I accept any type that implement ISpecification interface

**ISpecification**

**IFilter**

**BetterFilter**

Color Specification

Size Specification

AndSpecification

**public enum Color{Red,Green,Blue}**

**public enum Size{Small,Meduim,Large,Yuge}**

**public class Product{**

**public string Name { get; set; }**

**public Color Color { get; set; }**

**public Size Size { get; set; }**

**public Product(string name, Color color, Size size){**

**if (name == null){throw new ArgumentNullException(paramName: nameof(name));}**

**Name = name;**

**Color = color;**

**Size = size;}}**

**public class ProductFilter{**

**public IEnumerable<Product> FilterBySize(IEnumerable<Product> products, Size size){**

**foreach (var prod in products)**

**if (prod.Size == size)**

**yield return prod;}**

**public IEnumerable<Product> FilterByColor(IEnumerable<Product> products, Color color){**

**foreach (var prod in products)**

**if (prod.Color == color)**

**yield return prod;}**

**public IEnumerable<Product> FilterByColorAndSize(IEnumerable<Product> products, Color color,Size size){**

**foreach (var prod in products)**

**if (prod.Color == color && prod.Size == prod.Size)**

**yield return prod;}}**

**//with using open closed principle, we separate into two interface, one for the filters ,**

**//with using Open Closed Principle we use the two interface one for the filters**

**//and the another that use that interface as argument and implemented by custom filter**

**public interface ISpecification<T>{**

**bool IsSatisfied(T obj);}**

**public interface IFilter<T>{**

**IEnumerable<T> Filter(IEnumerable<T> items, ISpecification<T> sepc);}**

**public class ColorSpecification : ISpecification<Product>{**

**public Color \_color { get; set; }**

**public ColorSpecification(Color color){this.\_color = color;}**

**public bool IsSatisfied(Product obj){**

**return obj.Color == \_color;}}**

**public class SizeSpecification : ISpecification<Product>{**

**public Size \_size { get; set; }**

**public SizeSpecification(Size size){this.\_size = size;}**

**public bool IsSatisfied(Product obj){**

**return obj.Size == \_size;}}**

**public class AndSpecification<T> : ISpecification<T>{**

**private ISpecification<T> first, second;**

**public AndSpecification(ISpecification<T> first,ISpecification<T> second){**

**this.first = first ?? throw new ArgumentNullException(paramName: nameof(first));**

**this.second = second ?? throw new ArgumentNullException(paramName: nameof(second));}**

**public bool IsSatisfied(T obj){**

**return first.IsSatisfied(obj) && second.IsSatisfied(obj);}}**

**public class BetterFilter : IFilter<Product>{**

**public IEnumerable<Product> Filter(IEnumerable<Product> items, ISpecification<Product> spec){**

**foreach (var item in items)**

**if (spec.IsSatisfied(item))**

**yield return item;}}**

**class Program{**

**static void Main(string[] args){**

**var apple = new Product("Apple", Color.Green, Size.Small);**

**var tree = new Product("Tree", Color.Green, Size.Meduim);**

**var house = new Product("House", Color.Blue, Size.Large);**

**var prods = new List<Product>(){ apple, tree, house };**

**//each time that we need new filter feature, we need to modify new filter feature in the , so //the solution is used with Open Closed Design Pattern**

**var pf = new ProductFilter();**

**Console.WriteLine("Green Products (old)");**

**foreach (var p in pf.FilterByColor(prods, Color.Green)){Console.WriteLine(p.Name);}**

**var bf = new BetterFilter();**

**Console.WriteLine("Better Filter");**

**foreach (var p in bf.Filter(prods, new ColorSpecification(Color.Green))){**

**Console.WriteLine(p.Name);}**

**Console.WriteLine("With Multiple Better Filter");**

**foreach (var p in bf.Filter(prods, new AndSpecification<Product>(new ColorSpecification(Color.Green),new SizeSpecification(Size.Meduim)))){**

**Console.WriteLine(p.Name);}**

**Console.ReadLine();}}**