Soyut Fabrika(Abstract Factory) Tasarım Deseni

Nedir?

Soyut Fabrika tasarım deseni, creational tasarım desenlerinden biridir. Bu tasarım deseni birbiriyle alakalı veya bağımlı nesnelerin somut sınıflarını belirtmeden, yaratılması için gereken bir arayüz sağlar. Ayrıca bu desene fabrikaların fabrikası(**factories of the factory**) da denir.

Ne zaman Kullanılır?

Product'ları yaratan fabrika sınıfından somut nesne yaratma işlemini çıkarmak istiyorsak, bu tasarım desenini kullanmak gereklidir. Ayrıca temel fabrika deseninde bulunan if-else yapısından da kurtulmak istiyorsanız soyut fabrika tasarım desenini kullanabilirsiniz.

Nasıl Kullanılır?

Her bir Product alt sınıfları için bir fabrika sınıfı oluşturmak gereklidir. Bu oluşturulacak fabrika sınıfları ise türü **interface** veya **abstract** olan bir süper fabrika sınıfından türümelidir.

Faydaları Nedir?

1. Client sınıfına, bir **abstract** arayüz kullanmasını sağlayarak, gerçekte üretilecek ilişkili Product sınıflarını bilmeden veya önemsemeden oluşturulmasına olanak tanır.
2. if-else yapısından kurtararak daha anlaşılır kod yazmayı sağlar.

Gerekenler

Türü **abstract, interface** veya normal sınıf olan bir süper sınıf
En az bir tane alt sınıf
Bir tane süper **abstract** factory sınıfı
En az bir tane alt factory sınıfı
Bir tane Fabrika üretici sınıfı
Test sınıfı

Örnek Kullanım Alanları

1. **javax.xml.parsers.DocumentBuilderFactory#newInstance()**
2. **javax.xml.transform.TransformerFactory#newInstance()**
3. **javax.xml.xpath.XPathFactory#newInstance()**

Örnek Uygulama

Super(Üst) Sınıfı:

```
/**
 * Bu bizim super sinifimiz. Dikkat edin interface olarak tanimlandi. Normal sinif veya soyut ta olabilirdi
 */
public interface Shape {
    public double getArea();
    public double getSize();
}
```

Alt Sınıflar:

```
/**
 * Basit bir Circle sinifi
 */
public class Circle implements Shape {
    private double radius;

    @Override
    public double getArea() {
        return Math.PI*radius*radius;
    }

    @Override
    public double getSize() {
        return 2*Math.PI*radius;
    }

    public void setRadius(double radius) {
        this.radius = radius;
    }
}
```

```
/**
 * Basit bir Rectangle sinifi
 */
public class Rectangle implements Shape {
```

```
    private double width;
    private double height;

    @Override
    public double getArea() {
        return width*height;
    }

    @Override
    public double getSize() {
        return 2*(width+height);
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public void setHeight(double height) {
        this.height = height;
    }
}
```

Super(Üst) Abstract Fabrika Sınıfı:

```
/**
 * Abstract fabrika sinifimiz. Bu sinif süper fabrika sinifi
 */
public interface ShapeAbstractFactory {
    public Shape createShape();
}
```

Alt Somut Fabrika Sınıfları

```
/**
 * Circle sinifi icin fabrika sinifi
 */
public class CircleFactory implements ShapeAbstractFactory {
    @Override
    public Shape createShape() {
        return new Circle();
    }
}
```

```
/**
 * Rectangle sinifi icin fabrika sinifi
 */
public class RectangleFactory implements ShapeAbstractFactory {
    @Override
    public Shape createShape() {
        return new Rectangle();
    }
}
```

Fabrika Sınıf Nesne Üretilmesi

```
/**
 * Somut fabrika siniflarinin turune gore Shape nesneleri uretilmesini saglar
 */
public class ShapeFactory {
    public static Shape getShape(ShapeAbstractFactory factory){
        return factory.createShape();
    }
}
```

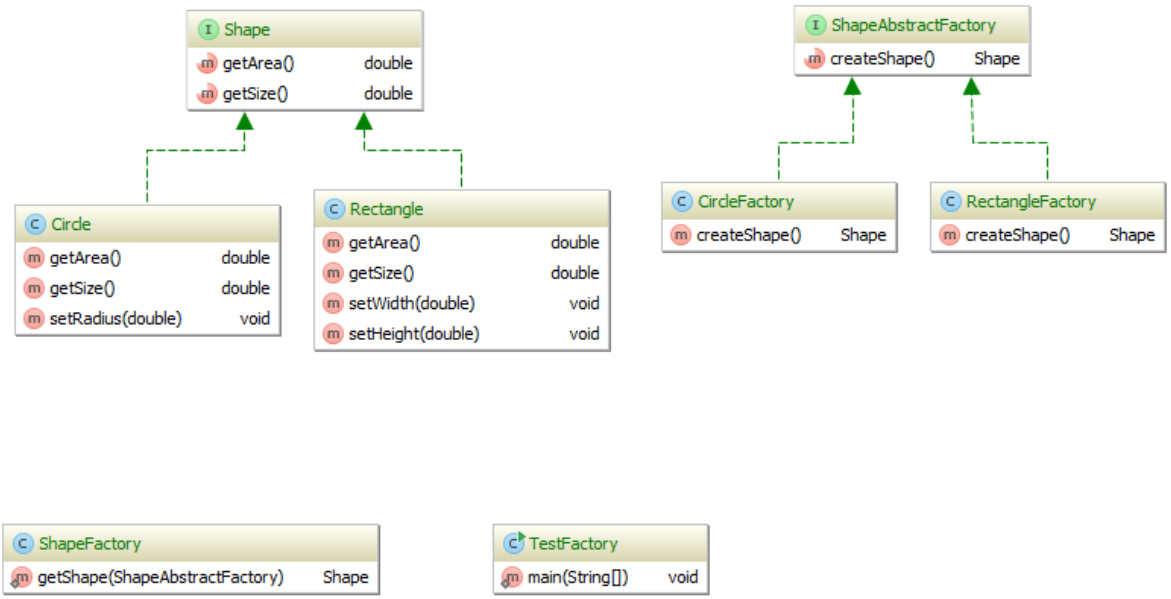
İstemci(Client) Sınıfı

```
/**
 * Test sinifi.
 */
public class TestFactory {
    public static void main(String[] args) {
        Shape rectangle = ShapeFactory.getShape(new RectangleFactory());
        ((Rectangle) rectangle).setWidth(13);
        ((Rectangle) rectangle).setHeight(5);

        Shape circle = ShapeFactory.getShape(new CircleFactory());
        ((Circle) circle).setRadius(4);

        System.out.println("Rectangle area: "+rectangle.getArea()+" and size: "+rectangle.getSize());
        System.out.println("Circle area: "+circle.getArea()+" and size: "+circle.getSize());
    }
}
```

Örneğimizin UML Diagramı



Fabrika Metod Tasarım Deseni'nin Şematik Gösterimi