Ödev 2

2. Creational Design Pattern'lar incelenmelidir. Örneklerle anlatınız. (20 PUAN)

2.1. Abstract Factory

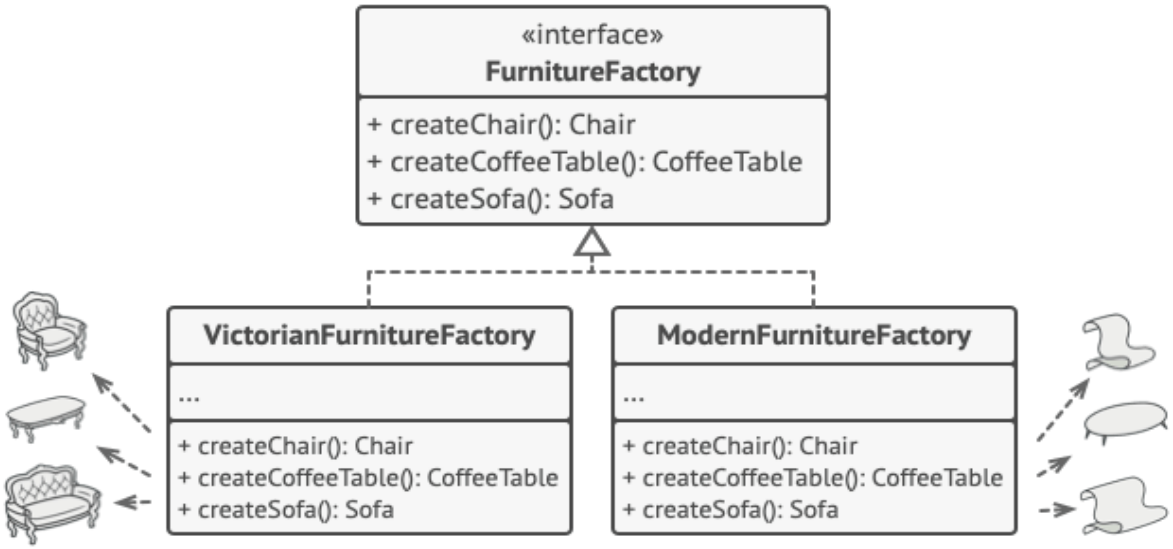
Birden fazla ürün grubu ile çalışmak durumunda kaldığımızda kullanılır. Ürün gruplarının oluşumunu istemci tarafından ayırarak, karar verme koşulu olmadan, esnek ve geliştirilebilir bir yapı kurmamızı sağlar.

Örnek: Bir mobilya şirketinde sandalye , koltuk ve sehpa olsun. Hepsi için de modern, victorian gibi tarzlar olsun. Modern bir koltuk ile victorian sehpa uyumlu olmaz. Yeni ürünler eklendiğinde kodun tamamen değişmesini istemeyiz.

Bu yüzden her ürün tipi için bir arayüz oluşturmamız gerekli. Tüm koltuk varyantları için koltuk arayüzünü, tüm sehpa varyantları için sehpa arayüzünü kullanmalıyız.

Sonraki adım ; ürün ailesine ait ürünleri oluşturmak için gereken tüm methodları içeren Abstract factory tanımlamaktır.

Örneğin createChair ,createSofa ,createCoffeeTable gibi.



İstemci kod, factory'ler ve ürünlerle çalışırken ilgili abstract interface'i kullanmalıdır. Bu size istemci kod'a pasladığınız factory sınıfı ve ürün varyantını asıl koda dokunmadan kolayca değiştirme olanağı sağlar.

2.2. Singleton

Bir sınıftan yalnızca 1 nesne oluşturmak için ortaya konulmuş bir yapıdır. Normal bir nesne üretirken istediğimiz yerde üretebiliriz. İlk olarak bunu engelliyoruz. Her yerden üretilemeyecek. Nesne üretmek isteyenler sınıfa başvuruyor. Sınıf da diyor ki bende var bunu kullan.

Faydaları; Birden fazla sınıfın içinde yani farklı farklı sınıfların içinden aynı instance a erişebiliriz.

Aynı nesneye bütün uygulamanın her yerinden erişebiliriz. Bir objenin tek olmasını garanti edebiliriz.

```
public class Singleton
{
        private string text:
        private static Singleton instance = nulls;
        private Singleton ()
        {
                text = "Hello World!";
        }

        public static Singleton GetInstance ()
        {
                if (instance == null)
                        instance = new Singleton():
                return instance;
        }
        public string GetText ()
        {
                return text;
        }
}
```

2.3.Builder Pattern

Bir classın özelliklerini üretmek istiyoruz. Fakat bazı özellikler gerekli bazı özellikler isteğe bağlı. Örneğin Starbucks clasımız olsun. Kahve boyutunu muhakkak bilmeliyiz fakat her kahve yumuşak içim değildir veya laktozlu değildir.

```
public static class Builder
{
private final String kahveBoyutu;
private String laktozsuzSutMiktarı;
private String yumusatici;

        public Builder(String kahveBoyutu)
        {
                super();
                this.kahveBoyutu= kahveBoyutu;
        }
        public Builder laktozsuzTayfa(String laktozsuzSut)
        {
                laktozsuzSutMiktarı = laktozsuzSut;
                return this;
        }
        public Builder yumusakIcenler(String yumusakDeger)
        {
                yumusatici = yumusakDeger;
                return this;
        }
        public starbucksBuilder build()
        {
                return new StarbuckBuilder(this);
        }
}
public StarbucksBuilder(Builder builder) {
         kahveBoyutu= builder.kahveBoyutu;
        laktozsuzSutMiktarı= builder. laktozsuzSutMiktarı;
```

```
        yumusatici = builder. yumusatici;
    }
```

Bunu mainde çağırırken ;

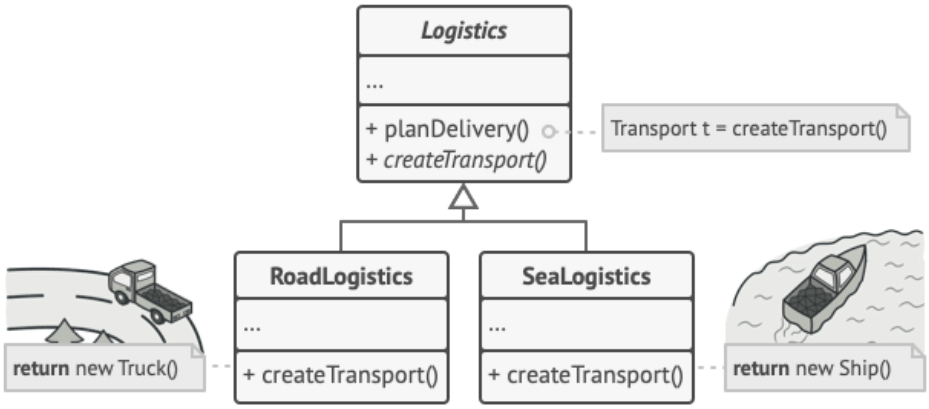
```
StarbucksBuilder sb= new StarbucksBuilder.Builder("tall").yumusakIcenler("istiyorum")build();
```

4.Factory Method

Factory Method, üst sınıfta nesneler oluşturmak için bir arabirim sağlayan, ancak alt sınıfların oluşturulacak bu nesne türünü değiştirmesine izin veren bir türdür.

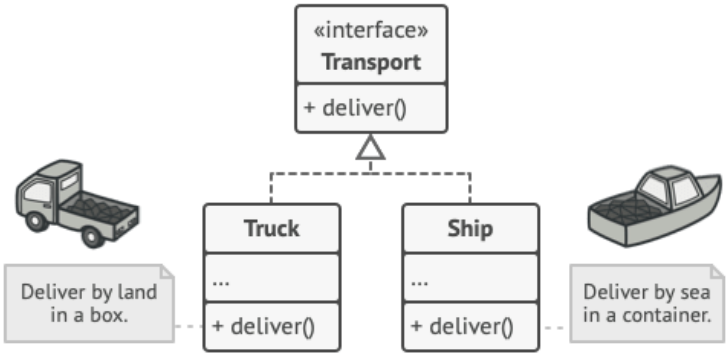
Bir lojistik firmasında sadece kamyon nakliyesi yapılıyor. Her şey kamyon sınıfıına göre ayarlanmış durumda. Deniz taşımacılığı da yapmak istersek kodun tamamı başka sınıf içinde olduğu için çok kolay olmayacaktır.

Çözüm; Fabrika Metodu/Factory Method deseni doğrudan nesne oluşturma çağrılarını (new operatörü kullanarak), özel bir fabrika metodu çağrısına dönüştürmeyi öneriyor. Nesneler hala new opeartörü ile oluşturuluyor ama fabrika metodu içerisinden çağrılıyor. Fabrika metodunun döndürdüğü nesneler de *ürün* olarak adlandırılıyor.



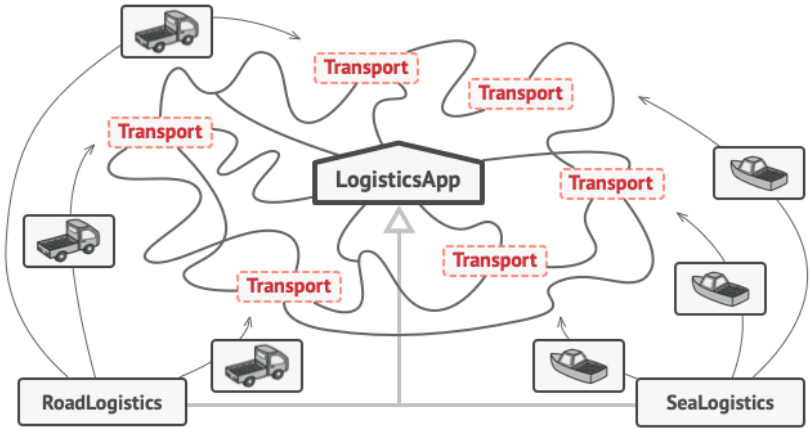
Alt sınıflar fabrika metodunun döndürdüğü nesne sınıfını değiştirebilir.

Örneğin hem Kamyon hem de Gemi sınıfları tasi adında bir metod içeren Transport Interface'ini esas almalı. Her iki sınıfta metodu farklı esas alacaktır; kamyonlar malı karadan taşırken gemiler deniz yoluyla taşıyacaktır. KaraLojistigi sınıfı kamyon nesnesi döndürürken DenizLojistigi sınıfı gemi döndürecektir.



Tüm ürünler aynı ara yüzü paylaşmak zorunda.

Örneğin hem Kamyon hem de Gemi sınıfları tasi adında bir metod içeren Transport Interface'ini esas almalı. Her iki sınıfta metodu farklı esas alacaktır; kamyonlar malı karadan taşırken gemiler deniz yoluyla taşıyacaktır. KaraLojistigi sınıfı kamyon nesnesi döndürürken DenizLojistigi sınıfı gemi döndürecektir.



Tüm ürün sınıfları ortak bir ara yüz paylaştığı sürece ilgili objeleri içiniz rahat şekilde çağıran koda döndürebilirsiniz.

Örneğin hem Kamyon hem de Gemi sınıfları tasi adında bir metod içeren Transport Interface'ini esas almalı. Her iki sınıfta metodu farklı esas alacaktır; kamyonlar malı karadan taşırken gemiler deniz yoluyla taşıyacaktır. KaraLojistigi sınıfı kamyon nesnesi döndürürken DenizLojistigi sınıfı gemi döndürecektir.

2.5.Prototype

Prototype (Clone) bir objeyi, kodunuz onun sınıflarına bağımlı hale gelmeden kopyalamayı sağlayan bir tasarım desenidir.

Bir nesneniz olduğunu ve onun birebir kopyasını oluşturmamız mümkün değildir. Çünkü bazı alanlar private tanımlanmış ve objenin dışından görünmüyor olabilir.

Prototype tasarım deseni klonlama sürecini klonlanan nesneye delege eder ve bunu klonlama destekleyen tüm nesneler için ortak bir arayüz belirleyerek yapar. Bu arayüz kodunuzu bu nesnenin sınıfına bağımlı hale getirmeden nesneyi klonlamanızı sağlar. Böyle bir arayüz genellikle sadece bir `clone` metodu içerir.

4. Java dünyasındaki framework'ler ve çözdükleri problemler nedir? Kod Örneklendirini de içermelidir. (10 Puan)

SPRING: Kurumsal alanda en çok kullanılan Java Framework'ü olarak Spring'i örnek verebiliriz. Spring Framework'ü hem .net hem de Java için geliştirilmiş Java EE uygulamaları yapmamızı kolaylaştıran harika bir Framework'tür. Bu framework model-view-controller katmanlarını kontrol ederek ihtiyacımız olan paket ve sınıfları ekleyebildiğimiz ve bu paketleri kullanabilmemizi sağlayan bir Framework'tür. Spring ile çok karmaşık uygulamaların yanında oldukça basit uygulamalar yapmak da mümkündür.

"Spring, Java programlamayı herkes için; daha hızlı, daha kolay ve daha güvenli hale getirir. Spring'in hız, basitlik ve üretkenliğe odaklanması onu dünyanın en popüler Java framework'ü haline getirdi." gibi açıklamalar ile tanıtmaktadır.

JSF (Java Server Faces): Bir web sitesini Java ile yapmak mümkündür. Ancak Java Server Faces Framework'ü dinamik web sayfaları oluşturmamıza yaramaktadır. JSP kodları HTML dilinin içine yazılır ve kendine özel etiket sistemi vardır. Bu sayede HTML ile karışmadan kendi içinde güzel ve düzenli bir performans sunarak kaliteli siteler yapma konusunda yazılımcıya yardımcı olmaktadır.

MAVEN: Bir proje geliştirirken proje içinde bir standart ve düzen oluşturmayı, geliştirme sürecini daha basitleştirmemizi sağlayan Java Framework'üdür. Geliştirdiğiniz projenin kütüphane bağımlılığını ve IDE bağımlılığını bu framework sayesinde ortadan kaldırabildiği gibi projenin daha kolay geliştirilmesi hakkında bize birçok kolaylık sağlamaktadır.

HIBERNATE: Hibernate bir ORM kütüphanesidir. Veritabanı üzerinde yapılan işlemleri kolaylaştıran bu framework için Java sınıflarının veritabanı dönüşümünü yapıyor diyebiliriz. Aynı zamanda veri çekme ve veri sorgulama işlerinde de oldukça yardımı dokunan bir Framework'tür.

5. Spring frameworkünün kullandığı design patternlar neler?

1. Singleton pattern
2. Factory Method pattern
3. Proxy pattern
4. Template pattern

6. SOA - Web Service - Restful Service - HTTP methods kavramlarını örneklerle açıklayınız. (5 Puan)

SOAP(en: Simple Access Protocol ,tr: Basit Nesne Erişim Protokolu) en temel anlamda, internet üzerinden küçük miktarda bilgileri yada mesajları aktarma protokoludur. SOAP mesajları XML formatındadırlar ve genellikle HTTP(Hyper Text Transfer Protocol) protokolu(bazende TCP/IP) kullanılarak gönderilirler. SOAP ,XML tabanlı kullanıma mecbur bırakır. Bu konuda esnek değildir.

WEB SERVICE: Web servisler platform bağımsız olmak üzere bir çok uygulama, cihaz ya da nodeun birbiri ile iletişim kurmalarını sağlayan yapılardır.

Restful web servisleri: REST mimarisi temel alınarak geliştirilmiş oldukça hafif, genişletilebilir ve basit servislerdir. Restful servislerin amacı client-server arasındaki veri akışını platform bağımsız olarak gerçekleştirebilmek ve veri

akışını en az yükle sağlayabilmektir.
Restful servisleri response tipi olarak JSON, HTML, XML gibi bir çok formatta çalışabilirler. Yapısının az yer kaplaması ve bir çok mobil platformda kullanışlı olmasından dolayı response tipi olarak JSON kullanılmaktadır. Restful servisleri esnek ve kolay geliştiriebilir bunun yanında dil ve platform bağımsızdırlar. SOAP servislerinin aksine ekstra bir kütüphaneye ihtiyaç duymadan çalışabilirler.

KısacaRest servisler ;

1.Platform bağımsızlar. (Client'ın Windows, Server'ın Linux olmasının hiçbir önemi yok)

2.Dil bağımsızlar .

3.HTTP üzerinden çalışıyorlar.

4.Esnekler ve çok kolay genişletilebilirler.

HTTP methodları?

HTTP methodları arasında GET, PUT, DELETE Idempotent methodlardır. POST değildir.