CSE443 Object Oriented Analysis and Design

Seyit Ahmet KARACA - 141044084

HW2 Rapor

1.Soru 1 Cevapları

1 – A

Clonable interface'ini implement edip bir class oluşturulup sonrasında yeni oluşturduğumuz singleton sınıfımızı bundan inherit edersek clone() methodunu ile singleton sınıfının kopyası oluşturulabilir.Böylece singleton konsepti bozulmuş oluyor.Bunu görmek için oluşturulan singleton sınıfından bir obje oluşturulur.Sonrasında yeni bir singleton objesi oluştulurken ilk objenin clone methodu ile yeni bir obje oluşturulabilir. Oluşturulmuş bu iki objenin hash kodu ekrana bastırılmak istendiğinde iki farklı hashcode gözlenir. Bunun anlamı iki farklı obje oluşturulmuştur.

1 - B

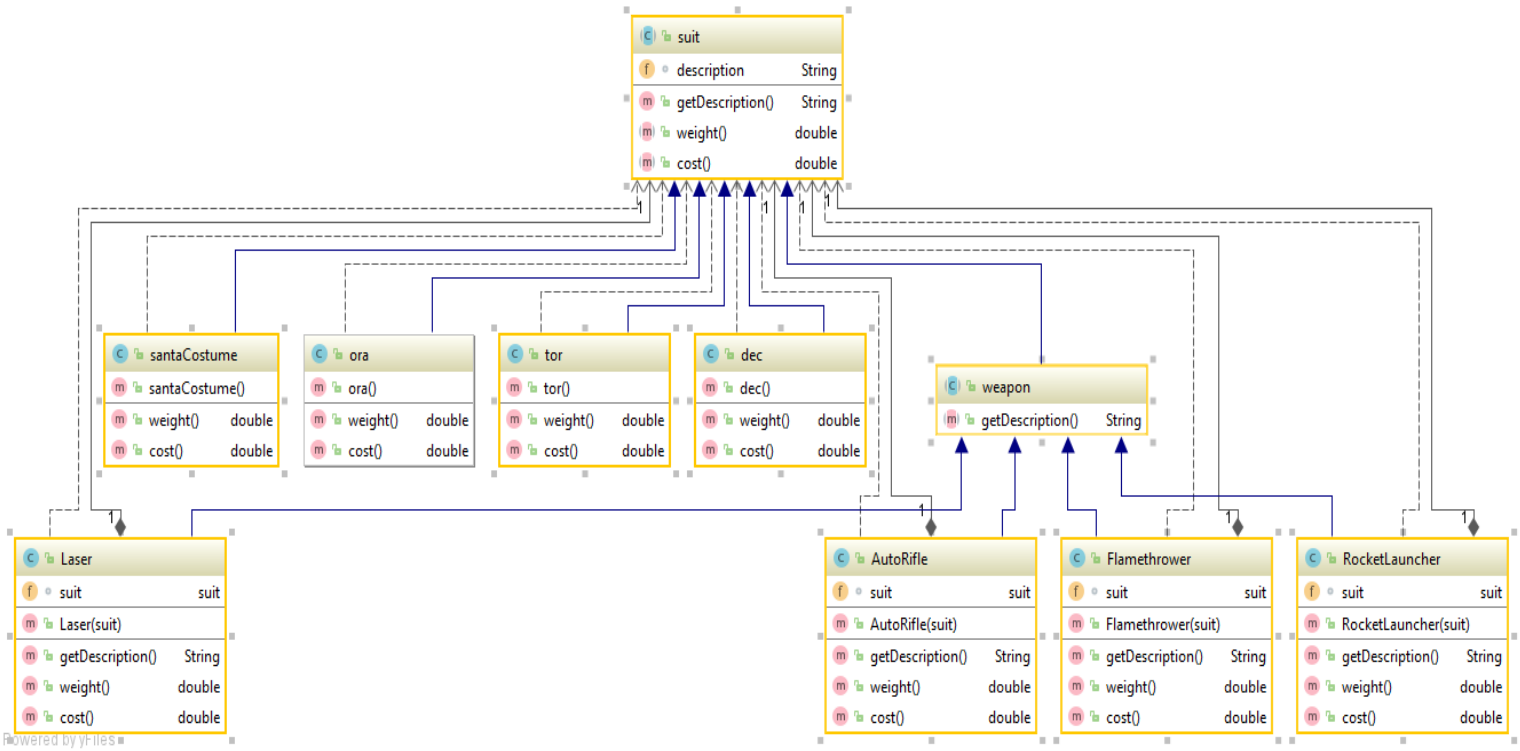
Bunu engellemek için clonable interface'i implement edilip clone methodu override yöntemi ile yeniden yazılır. İçerisine ClonableNotSupportedException gönderilmesi sağlanır.Böylece clone methodu çağrıldığında hata mesajı verilip singleton sınıfının kopyası oluşturulması engellenecektir.

1 – C

Üst sınıfta Clonable interface'ini implement etmiş sınıf olsa dahi override ile singleton sınıfında bu engellenebilir.

2.Zırhsan

2.1.Zırhsan UML Diyagramı



2.2.Zırhsan Sınıf Yapısı

Suit isminde soyut sınıf bulunmaktadır.Bu soyut sınıf sayesinde Decorator tasarım örüntüsü gerçeklenebilmektedir. Suit sınıfını extend ederek içerisinde bulunan weight ve cost methodlarını override eden 4 kostüm bulunmaktadır. Oluşturulan yapıda bu kostümlere yeni özellikler eklenerek bir savaş kostümü oluşturmak amaçlandığından bu 4 kostüm yapının temelidir. Sonrasında ise ne kadar uygun olmasa da "weapon" soyut sınıfı oluşturulup Suit sınıfından extend edilmiştir. Weapon sınıfını extend eden sınıflar ise kostümlere ek özellikler ekleyecek sınıflardır. Bu sınıflar suit sınıfında soyut olarak bırakılmış fiyat ve ağırlık methodlarını yeniden yazmak zorundadır. Bu yeniden yazılan methodlarda kurucu fonksiyonlarında aldıkları kostümün ağırlığına ve fiyatına eknecek şekilde düzenlenmiştir. Bu sayede Decorator tasarım örüntüsü gerçeklenmiş olup kıyafetlere dinamik olarak yeni özellikler eklenebilir.

2.3. Zırhsan Sonuçlar

```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
ora, Laser 1700000.0TL  35.5KG
HoHoHo, AutoRifle, AutoRifle, Laser 260003.0TL  9.5KG
Dec, RocketLauncher, AutoRifle, Laser 880000.0TL  39.5KG
```

Suit sınıfından ora objesi oluşturuldu ve suit sınıfından extend olmuş olan ora kostümü dinamik olarak atandı. Ora isimli kostüme bir tane lazer silahı eklendi ve yeni oluşan fiyat ve ağırlık ekrana bastırıldı.

Aynı yöntem ile kendi oluşturduğum santa kostümü oluşturuldu ve 2 tane auto rifle ile 1 tane lazer silahı eklendi.Yeni fiyat ve ağırlığı ekrana bastırıldı.

```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
ora, Laser 1700000.0TL  35.5KG
ora 1500000.0TL  30.0KG
```

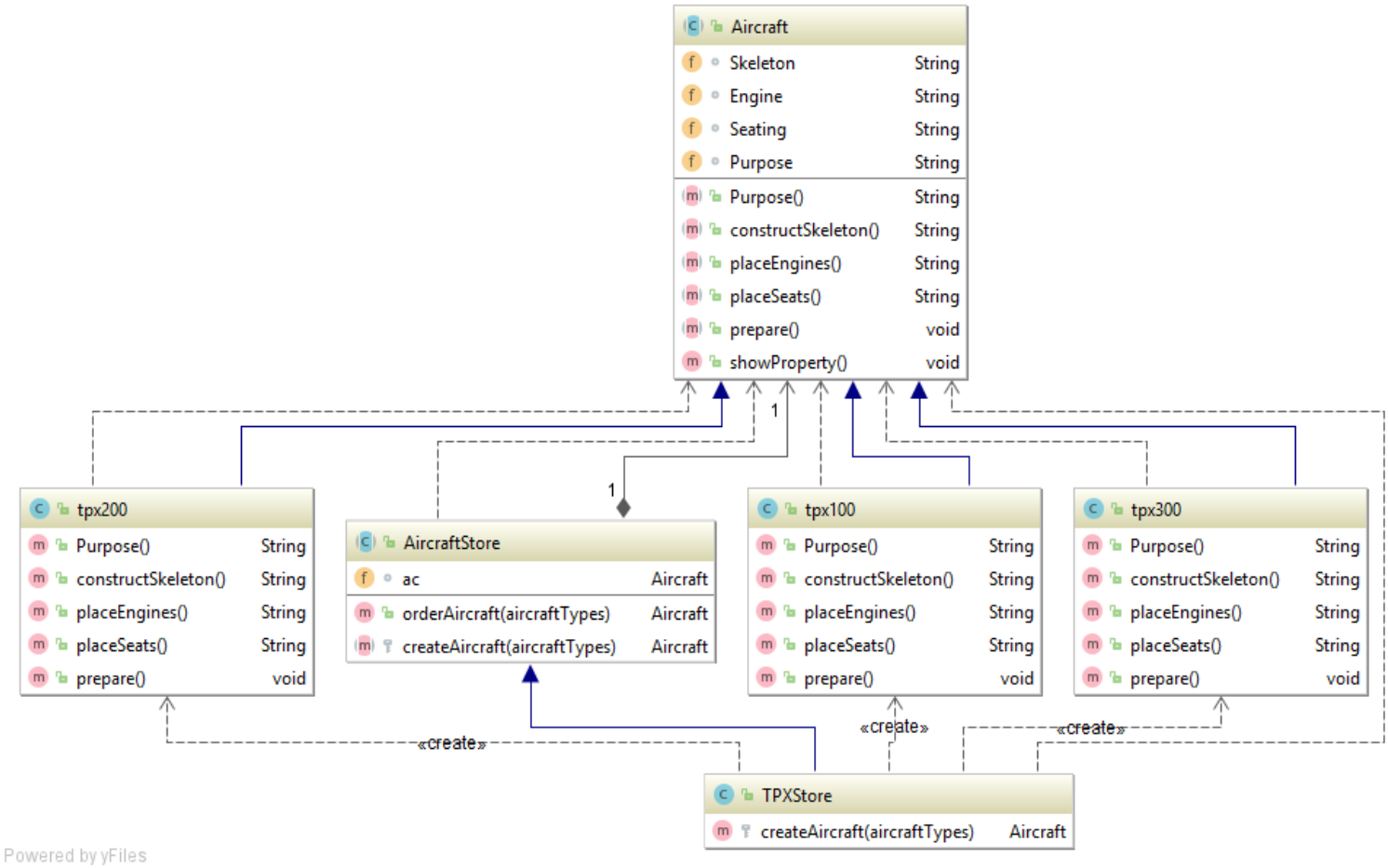
Ora kostümü oluşturuldu ve lazer özelliği eklenip yeni fiyat ve ağırlığı ekrana bastırıldı. Aynı şekilde ikinci bir ora kostümü oluşturulup herhangi bir özellik eklenmedi. Lazer özelliğinin fiyatı 20000 olup ağırlının 5.5KG olduğu olması gerektiği gibi gözlemlendi.

3.1. TAI Fabrika Method

3.1.1. TAI Fabrika Method Amacı

TPX serisi yolcu uçakları yapılacak. Bu serinin 3 farklı modeli bulunmaktadır. Farklı modeller olmalarından dolayı yapılma amaçları , iskeletleri , koltuk sayıları ve motorları farklıdır. Yapılması istenen kodlama ise bir uçak fabrikası olacak ve istenen tipte uçak hazırlanıp verilecek.

3.1.2. TAI Fabrika UML Diyagramı



3.1.3. TAI Fabrika Sınıf Yapısı

Bir tane soyut uçak sınıfı oluşturuldu.Sınıfta uçakların ortak özellikler ve methodları eklendi.Sonrasında ise uçakların istekte bulunabilmesi için bir AircraftStore oluşturuldu. Bu soyut sınıftan uçaksipariş etmek için bir method bulunmaktadır. Birde bu uçak fabrikasında her farklı modelin uçak oluşturabilmesi için soyut olarak bırakılmış bir createAircraft methodu bulunmaktadır. Fabrikada yeni bir uçak serisi üretilmek istenirse bu soyut store'dan extend edip yapması gereken sadece createAircraft methodunu implement etmek olacaktır. Bu projede sadece tek tip uçak serisi olduğundan sadece TPXStore bulunmaktadır. TPXStore , aitcraftstore'dan extend olmuş olup uçak tipi alan createAircraft methodunu implement etmiştir. Böylece istenen tipte uçak oluşturulabilmektedir.

Oluşturulan store'un sipariş verebilmek için kullanılan orderAircraft methoduna uçak tipi verilerek istenen uçak elde edilebilmektedir.

3.1.4. TAI Fabrika Çıktıları

```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
Amac        :Domestic flights
Motor       :Single jet engine
İskelet     :Aluminum alloy
Koltuk sayisi:50 seats
Amac        :Domestic and short international flights
Motor       :Twin jet engine
İskelet     :Nickel alloy
Koltuk sayisi:100 seats
Amac        :Transatlantic flights
Motor       :Quadro jet engine
İskelet     :Titanium alloy
Koltuk sayisi:250 seats

Process finished with exit code 0
```

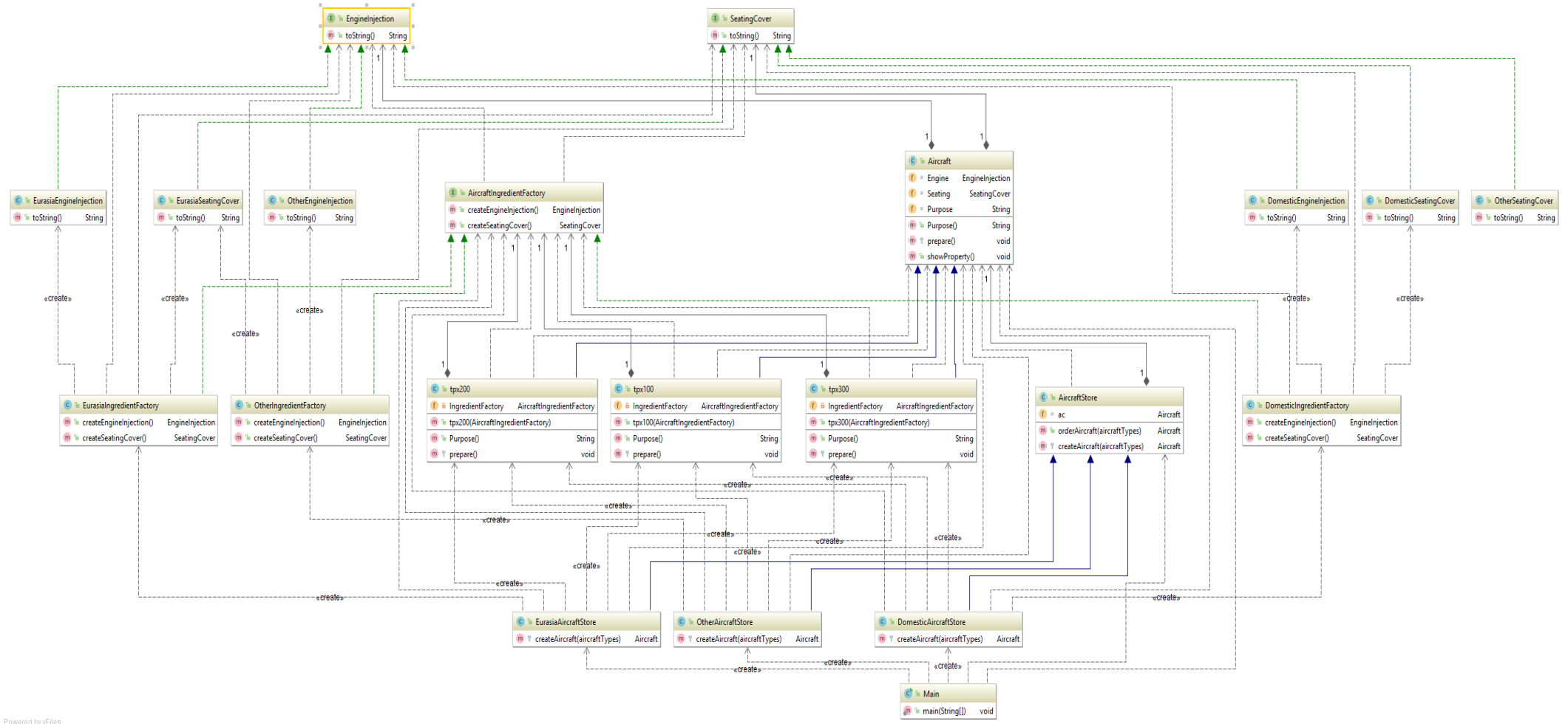
Bir tane TPXStore oluşturulup objesinden orderAircraft methodu ile 3 farklı uçak elde edilmiştir.

3.2.TAI Soyut Fabrika Method

3.2.1. TAI Soyut Fabrika Method Amacı

Önceki yapıda tek bir fabrika ve uçak modellerine göre üretim yapılıyordu. Soyut fabrika methodu ile farklı bölgelere ait özelliklerin de yapılabileceği bir fabrikaya dönüştürülmek istenmektedir. Böylece fabrikanın uçak modelleri aynı olmasına rağmen farklı uçuşlar için uçaklarda düzenlemeler yapılabilmektedir.

3.2.2. TAI Soyut Fabrika UML Diyagramı



3.2.3. TAI Soyut Fabrika Sınıf Yapısı

Önceki sınıf yapısına ek olarak AircraftStore'dan 3 farklı uçuş için farklı store oluşturuldu. Ve uçakların oluşturuldukları sınıflara kurucu methodlar eklendi. Bu yeni kurucu methodlara ingrendientFactory tipinde bir parametre eklendi. Farklı bölgelere uçuşlar yapacak uçakları üreten fabrikalar kendi özellikleri için motor ve döşemelerini yapabilecekleri özellik fabrikaları eklendi.Bu sayede farklı bölge uçuşları için üretilen uçaklara kendi istedikleri özellikleri verebilmektedir.Örneğin domesticAircraftStore , AircraftStore'den extend olduğu için order methodu bulunmakta ve AircraftStore'dan gelen createAircraft methodu ile uçak üretebilmektedir. Uçağı üretirken DomesticIngrendientFactory ile kendi özellklerinin bulundu sınıf ile uçakların ilklendirilmesini sağlamıştır. Bu özelliklerin bulunduğu fabrika ile uçak oluşurken bu özellik fabrikasına bakarak motor ve koltuk döşemelerini bu özellik sınıftan alarak oluşturulmaktadır. Bu sayede farklı bölgeler için farklı özellikte aynı model uçaklar üretilebilmektedir.

3.2.4. TAI Soyut Fabrika Sonuçlar

```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...

Amac        :Domestic and short international flights
Motor       :Turbofan
Koltuk sayisi:Linen

Amac        :Domestic and short international flights
Motor       :Geared turbofan
Koltuk sayisi:Linen

Amac        :Domestic and short international flights
Motor       :Turbojet
Koltuk sayisi:Velvet

Process finished with exit code 0
```

Görselde aynı amaç için üretilmiş farklı bölgelerdeki uçakların çıktısı bulunmaktadır.Amaçları aynı fakat uçakların özellikleri farklıdır.