Gebze Technical University

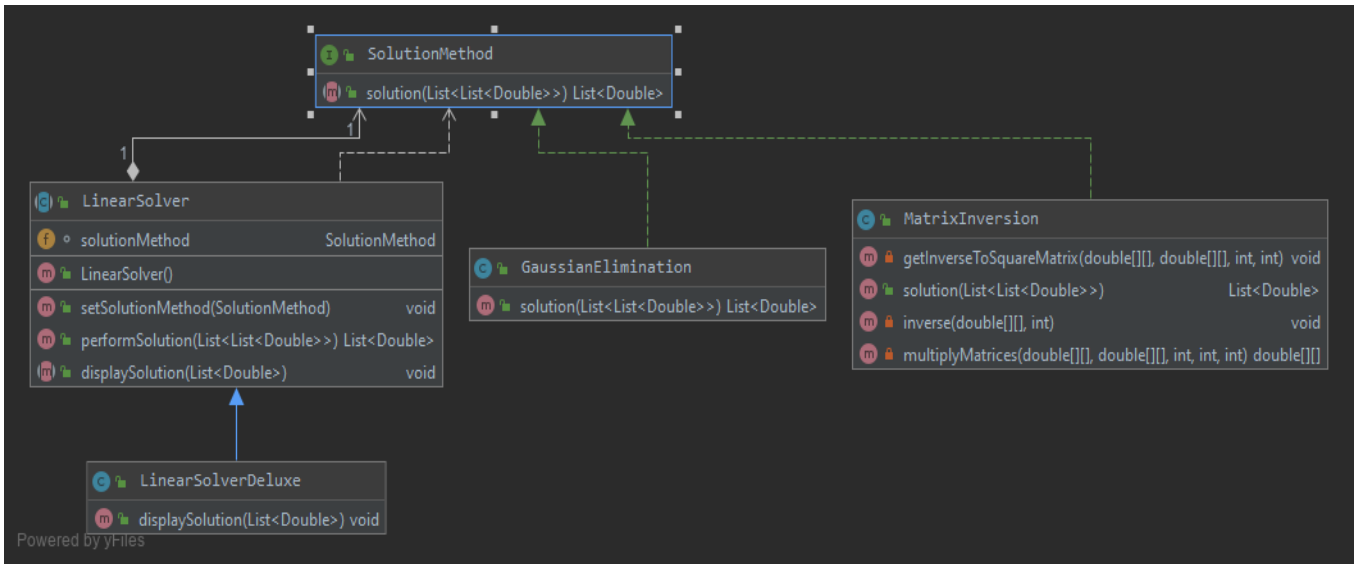
Computer Engineering

Object Oriented Analysis and Design
CSE443 – 2019

HOMEWORK 1 REPORT

Serkan Sorman
151044057

Q1)



Linear Solver Deluxe proje tasarımı için Strategy design patternı kullanıldı.Bu tasarımın kullanılmasının sebebi: iki farklı metod ile çözüm bulunmasına ihtiyaç olunması. Ayrıca daha sona yeni çözüm metodlarının projeye dahil edilmesi durumunda bakım maliyetlerini minimuma indirgemek ve çözüm metodları arasında dinamik geçiş olanağı sağlanması.

LinearSolver içinde delegate edilen solutionMetod nesnesi üzerinden solution metodu çağrılarak LinearSolvera set edilen solution tipine göre çözüm gerçekleştirilir. SetSolution() metodu ile metodlar arasında dinamik olarak geçiş sağlanır. Matrix inversion kullanılarak yapılan çözümde verilen matris coefficient ve constant matris olarak ayrılır. Daha sonra Gauss Jordan tekniği kullanılarak coefficient matrisin tersi alınır. Elde edilen inverse matris constant matris ile çarpılarak solution matrisi elde edilir. Bu solution matrisi bir Double array liste alınarak return edilir.

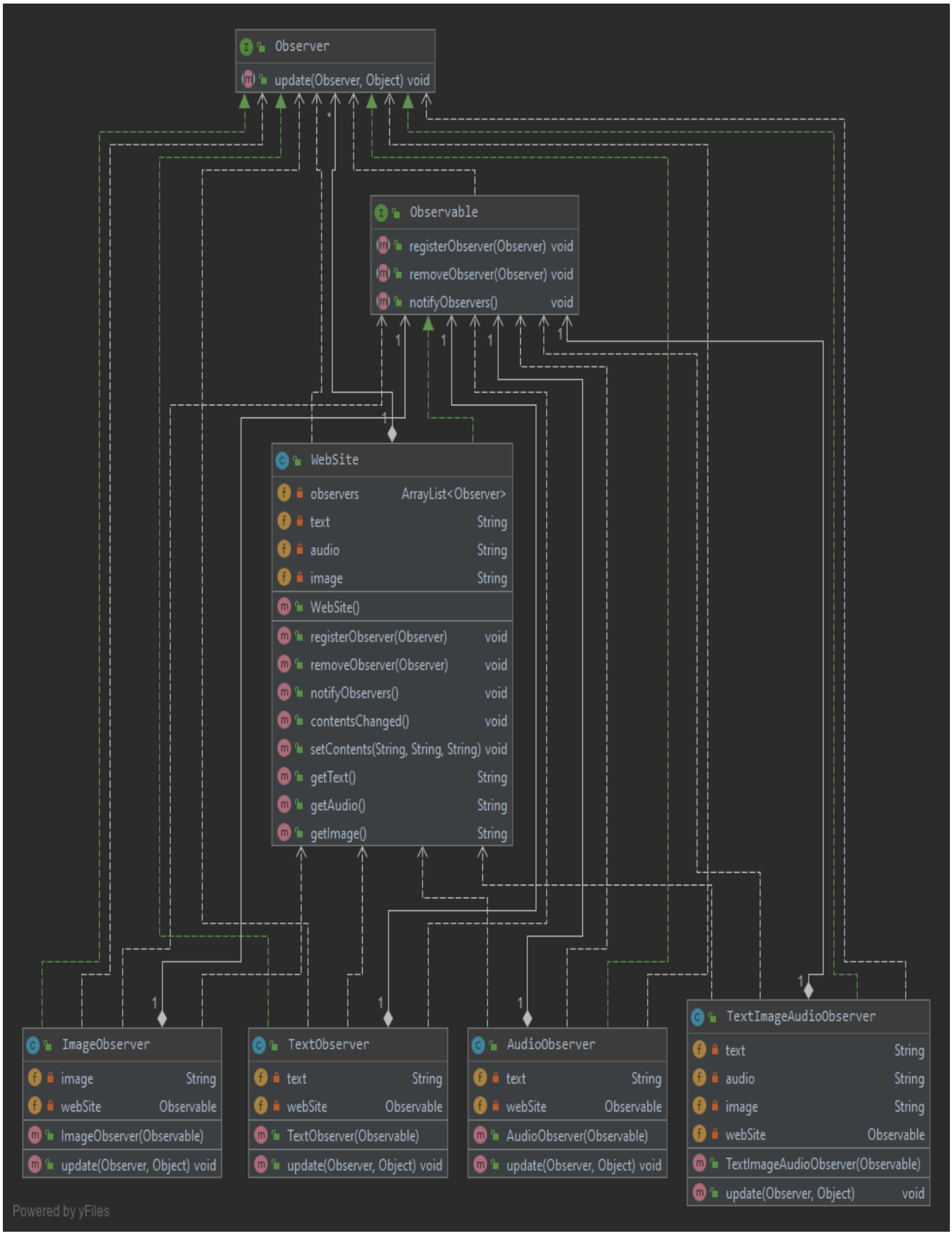
```
linearEquations.add(Arrays.asList(1.0, 2.0, 3.0,1.0)); // x + 2y + 3z = 1.0
linearEquations.add(Arrays.asList(4.0, 5.0, 6.0,1.0)); // 4x + 5y + 6z = 1.0
linearEquations.add(Arrays.asList(1.0, 0.0, 1.0,1.0)); // x + z = 1.0
```

Eşitlikler 2 boyutlu array listesi halinde kullanıcıdan alınır.

```
############ Gauss Elimination ##########
0,00
-1,00
1,00
############ Matrix inversion ##########
0,00
-1,00
1,00
```

Gauss elimination ve Matrix inversion yöntemleri arasında dinamik olarak geçiş yapılarak elde edilen solutionlar.

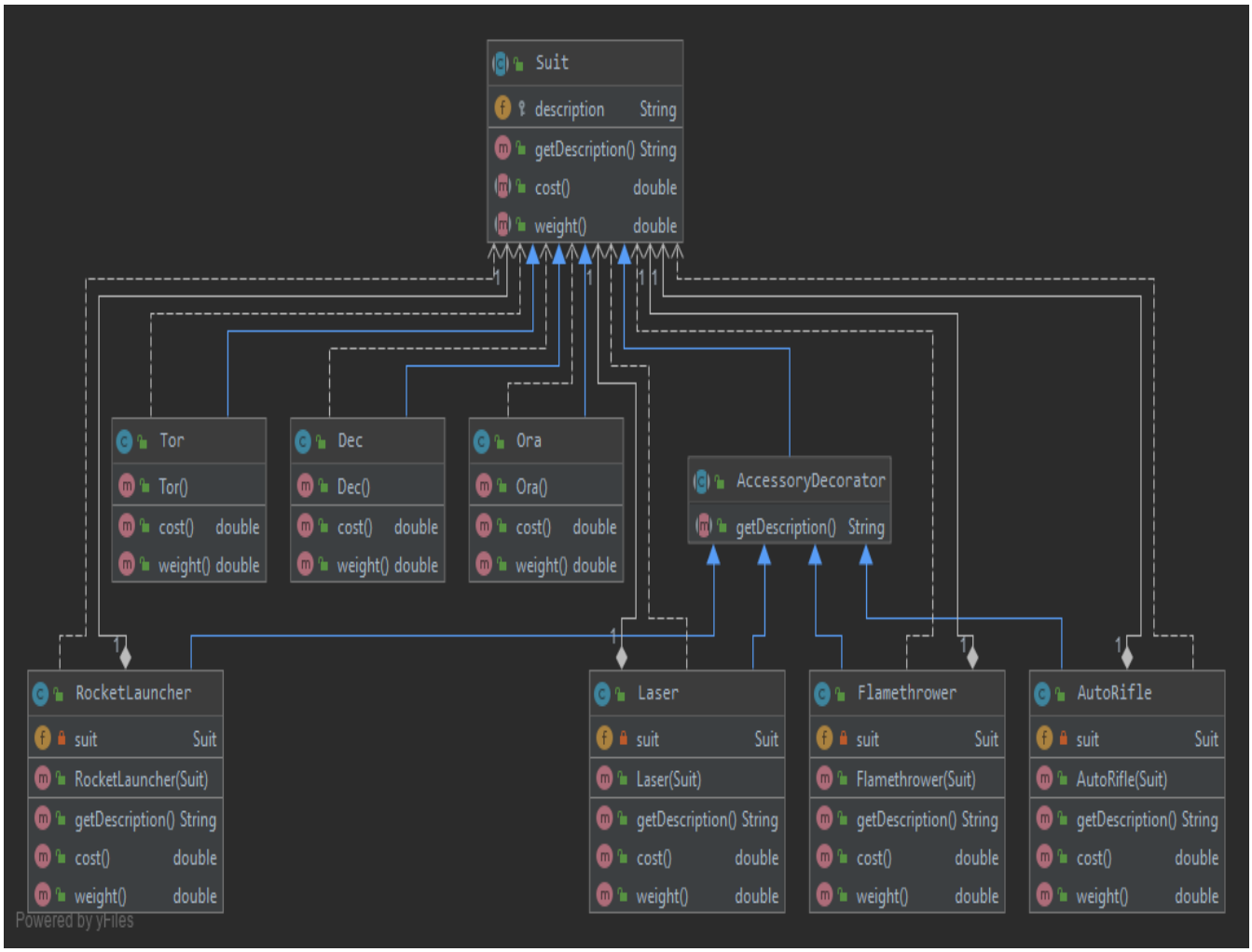
Q2)


Observer
update(Observer, Object) void
Observable
registerObserver(Observer) void
removeObserver(Observer) void
notifyObservers() void
WebSite
observers ArrayList<Observer>
text String
audio String
image String
WebSite()
registerObserver(Observer) void
removeObserver(Observer) void
notifyObservers() void
contentsChanged() void
setContents(String, String, String) void
getText() String
getAudio() String
getImage() String
ImageObserver
image String
webSite Observable
ImageObserver(Observable)
update(Observer, Object) void
TextObserver
text String
webSite Observable
TextObserver(Observable)
update(Observer, Object) void
AudioObserver
text String
webSite Observable
AudioObserver(Observable)
update(Observer, Object) void
TextImageAudioObserver
text String
audio String
image String
webSite Observable
TextImageAudioObserver(Observable)
update(Observer, Object) void
Powered by yFiles

Abonelerin favori web sitelerine abone olup. İstedikleri veri tipine göre (Text, Audio, Image ya da Tamamı) bildirim aldıkları bir sistem tasarlamak için Observer design patternı uygulandı. Ayrıca websitesinin birden fazla abonesi olabilir ve abone ekleme çıkarma ve bildirim gönderme gibi fonksiyonların olması gerekir. Observer design patternının burada kullanılmasının ana amacı bir nesnenin yani web sitesindeki contentlerde değişiklik olduğunda ona bağlı (abone) olan tüm observerlara bildirim gönderilip güncellenme işleminin gerçekleştirilmesidir. Yeni bir observer oluşturulduğunda class içinde tutulan websitesi üzerinden constructor içerisinde registerObserver() metodu çağrılarak websitesinde bulunan observer listesine eklenir.

WebSite da bulunan SetContents metodu ile web sitesinde yapılan değişiklikler set edilir ve contentsChanged() metodu içerisindeki notifyObservers metodu çağrılarak web sitesinin listesinde bulunan tüm observerların update() metodu çağrılır ve observerlar güncellenen contentleri alırlar.

Q3)



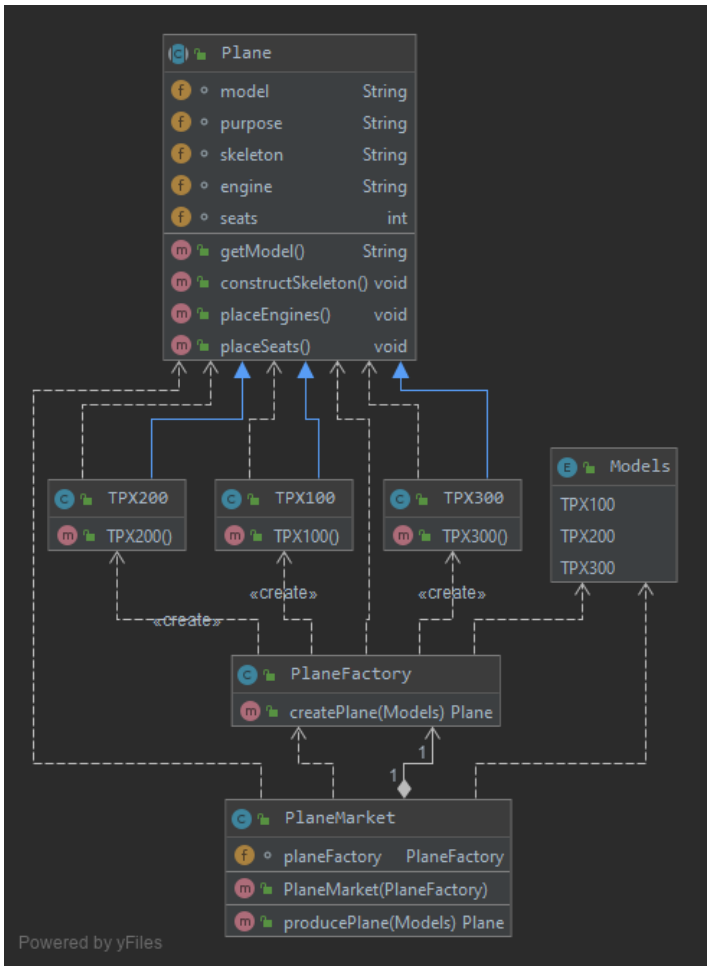
Projede Decorator design patternı kullanıldı. Bu patternın kullanılmasının sebebi farklı çeşit zırhlara run time'da ekipmanların eklenmesi. Zırhların sahip oldukları cost ve weightleri ile bu eklenen ekipmanların cost ve weightlerinin toplamının dinamik olarak hesaplanmasıdır. Ayrıca design patternın gereği olarak yeni zırh ve ekipmanların eklenme durumu için abstract Suit ve AccessoryDecorator yapılarına ihtiyaç duyulmuştur. AccessoryDecorator içersinde tutulan Suit objesi ve eklenen ekipman kullanılarak cost ve weight metodları içerisinde toplam cost ve weight hesaplanır.

```
Suit suit = new Dec();
System.out.println(suit.getDescription() + " Total Weight: " + suit.weight() + "kg Total Cost: "
        + suit.cost() + "k TL");

Suit suit2 = new Ora();
suit2 = new Flamethrower(suit2);
suit2 = new AutoRifle(suit2);
suit2 = new RocketLauncher(suit2);
suit2 = new Laser(suit2);

System.out.println(suit2.getDescription() + " Total Weight: " + suit2.weight() + "kg Total Cost: "
       + suit2.cost() + "k TL");

Suit suit3 = new Tor();
suit3 = new Flamethrower(suit3);
suit3 = new AutoRifle(suit3);
suit3 = new AutoRifle(suit3);
suit3 = new Laser(suit3);
System.out.println(suit3.getDescription() + " Total Weight: " + suit3.weight() + "kg Total Cost: "
       + suit3.cost() + "k TL");
```

```
Dec Suit Total Weight: 25.0kg Total Cost: 500.0k TL
Ora Suit, Flamethrower, AutoRifle, RocketLauncher, Laser Total Weight: 46.5kg Total Cost: 1930.0k TL
Tor Suit, Flamethrower, AutoRifle, AutoRifle, Laser Total Weight: 60.5kg Total Cost: 5310.0k TL
```

Birden fazla zırh tipi üzerine ekipman ekleme testi ve sonuçları.

Q4) Factory Method



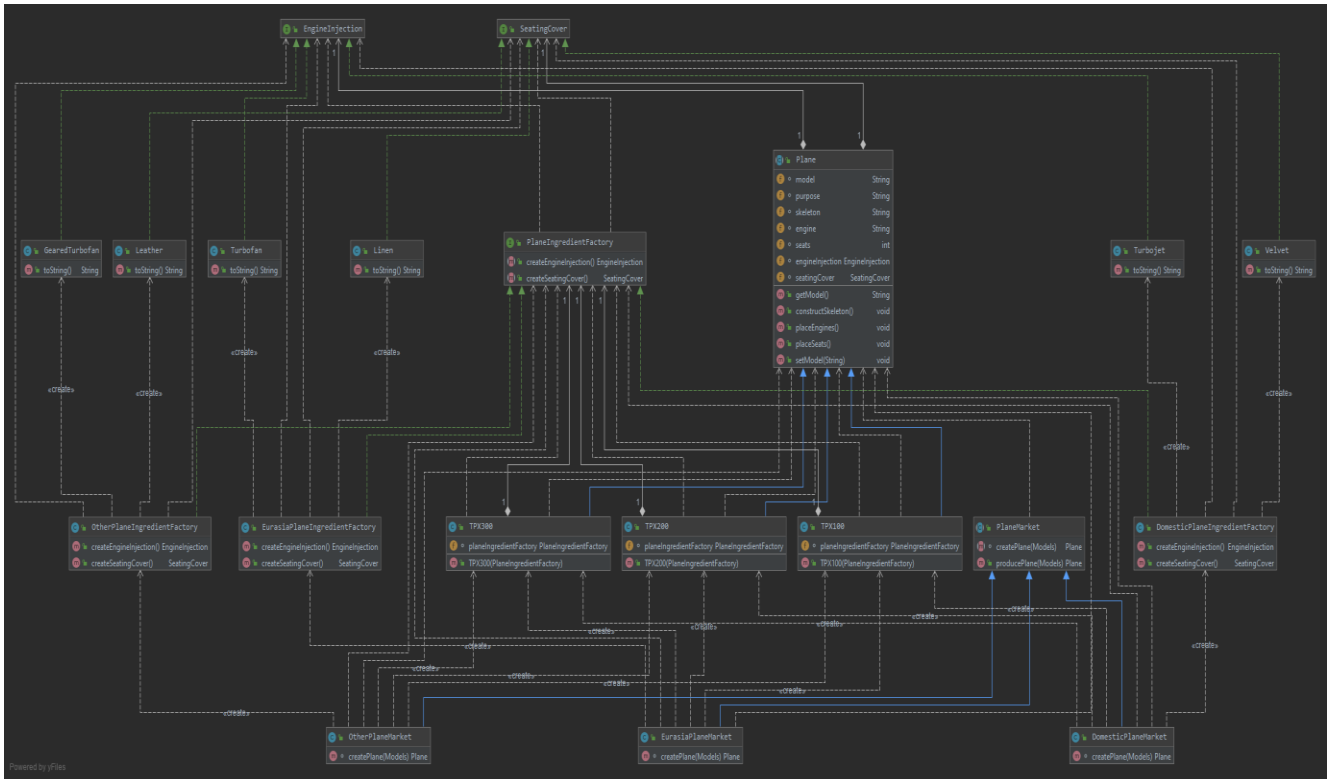
Birden fazla TPX modeli olduğu için abstract Plane kullanılmıştır. PlaneFactory içerisinde bulunan CreatePlane() metodunda parametrede gönderilen modele göre plane üretilir. Case sensitivtyi önlemek için modellerde enum kullanılmıştır. PlaneMarket içerisinde PlaneFactory objesi tutulur ve bu obje üzerinden producePlane() metodu içerisinde createPlane() metodu çağrılır.

```
PlaneFactory planeFactory = new PlaneFactory();
PlaneMarket planeMarket = new PlaneMarket(planeFactory);

Plane plane = planeMarket.producePlane(Models.TPX100);
System.out.println(plane.getModel()+ " is produced\n");

plane = planeMarket.producePlane(Models.TPX200);
System.out.println(plane.getModel()+ " is produced\n");

plane = planeMarket.producePlane(Models.TPX300);
System.out.println(plane.getModel()+ " is produced");
```

```
Constructing Skeleton Aluminum alloy
Placing Single jet engine
Placing Seats 50
TPX 100 is produced

Constructing Skeleton Nickel alloy
Placing Twin jet engines
Placing Seats 100
TPX 200 is produced

Constructing Skeleton Titanium alloy
Placing Quadro jet engines
Placing Seats 250
TPX 300 is produced
```

Main test ve sonuçları.

Q4) Abstract Factory



Class diagram net gözükmediği için image olarak ayrıca eklenmiştir

Aynı TPX uçak modelleri farklı marketlerde farklı içeriklerle bulunduğu için abstract PlaneMarket den türeyen concrete market sınıfları oluşturuldu. Ayrıca marketlere göre değişiklik gösteren EngineInjectionType ve SeatingCover için Interfaceler tanımlandı. Marktlere göre değişiklik gösteren içerikler bu interfaceleri implement etti ve Plane sınıfı içerisinde objeleri delegate edildi. Farklı içeriklerin üretimi marketlere göre değiştiği için her markete özel IngredientFactory tasarlandı.

Marketler içerisinde bulunan createPlane metodu içerisinde bulunulan markete göre IngredientFactory oluşturuldu. Oluşturulacak uçak modeline göre istenilen uçağın constructurına verildi ve uçak oluşturuldu.

```
PlaneMarket domesticPlaneMarket = new DomesticPlaneMarket();
PlaneMarket  eurasiaPlaneMarket = new EurasiaPlaneMarket();

Plane plane = domesticPlaneMarket.producePlane(Models.TPX100);
System.out.println("Plane " + plane.getModel() + " is produced\n");

plane = domesticPlaneMarket.producePlane(Models.TPX200);
System.out.println("Plane " + plane.getModel() + " is produced\n");

plane = domesticPlaneMarket.producePlane(Models.TPX300);
System.out.println("Plane " + plane.getModel() + " is produced\n");


plane = eurasiaPlaneMarket.producePlane(Models.TPX100);
System.out.println("Plane " + plane.getModel() + " is produced\n");

plane = eurasiaPlaneMarket.producePlane(Models.TPX200);
System.out.println("Plane " + plane.getModel() + " is produced\n");

plane = eurasiaPlaneMarket.producePlane(Models.TPX300);
System.out.println("Plane " + plane.getModel() + " is produced\n");
```

```
Constructing Skeleton Nickel alloy
Placing Twin jet engines Engine Injection Type: Turbojet
Placing Seats 100 Seating Cover: Velvet
Plane Domestic TPX 200 is produced

Constructing Skeleton Titanium alloy
Placing Quadro jet engines Engine Injection Type: Turbojet
Placing Seats 250 Seating Cover: Velvet
Plane Domestic TPX 300 is produced

Constructing Skeleton Aluminum alloy
Placing Single jet engine Engine Injection Type: Turbofan
Placing Seats 50 Seating Cover: Linen
Plane Eurasia TPX 100 is produced
```

Main test ve sonuçları