GEBZE TECHNICAL UNIVERSITY

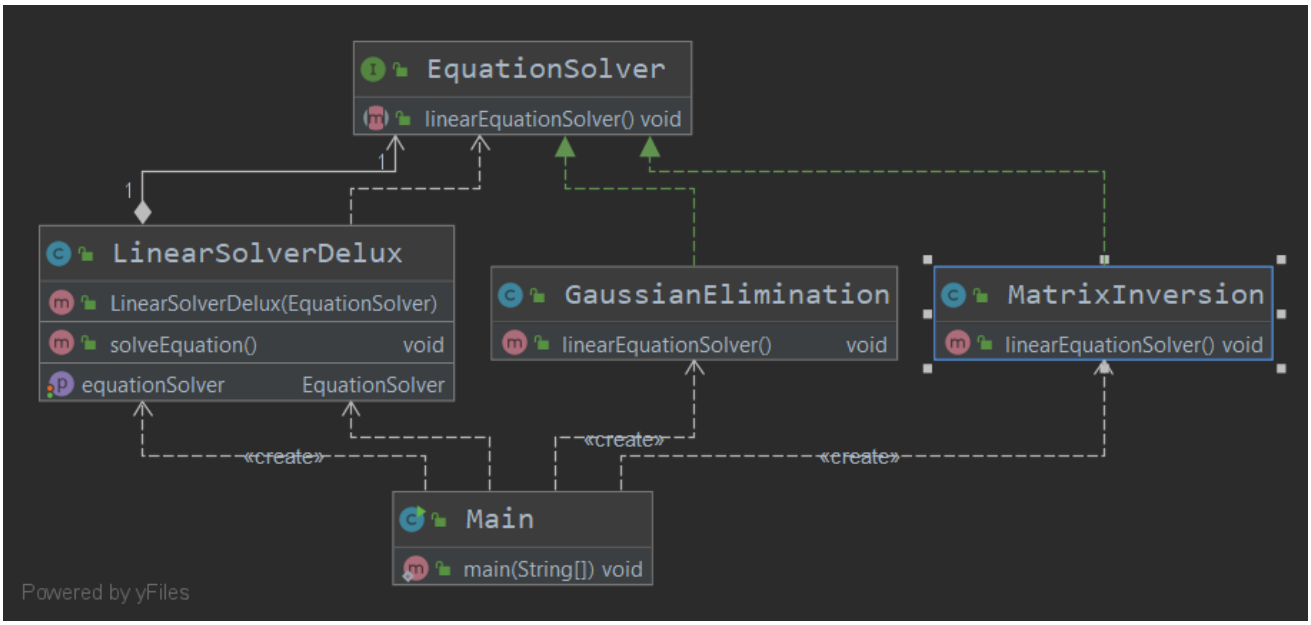
CSE443 – Object Oriented Analysis and Design

Homework 1 Report

İslam Göktan SELÇUK – 141044071

Part 1

Verilen problemi çözmek için sınıfları oluştururken Strategy örüntüsü kullanıldı. İki çözüm metodunun dinamik olarak değiştirilebilmesi için EquationSolver interface'inden çözüm metotlarının her biri için bir sınıf türetildi ve bu sınıfların kullanılacağı sınıf içerisinde, yani LinearSolverDelux sınıfı içersinde EquationSolver tipinde bir değişken oluşturuldu. EquationSolver sınıfı sarmalanarak LinearSolverDelux sınıfından soyutlandı. Böylelikle EuqationSolver objesi ve setter metotu yardımıyla istenen lineer denklem çözüm metodu dinamik olarak ayarlanarak kullanılabildi.

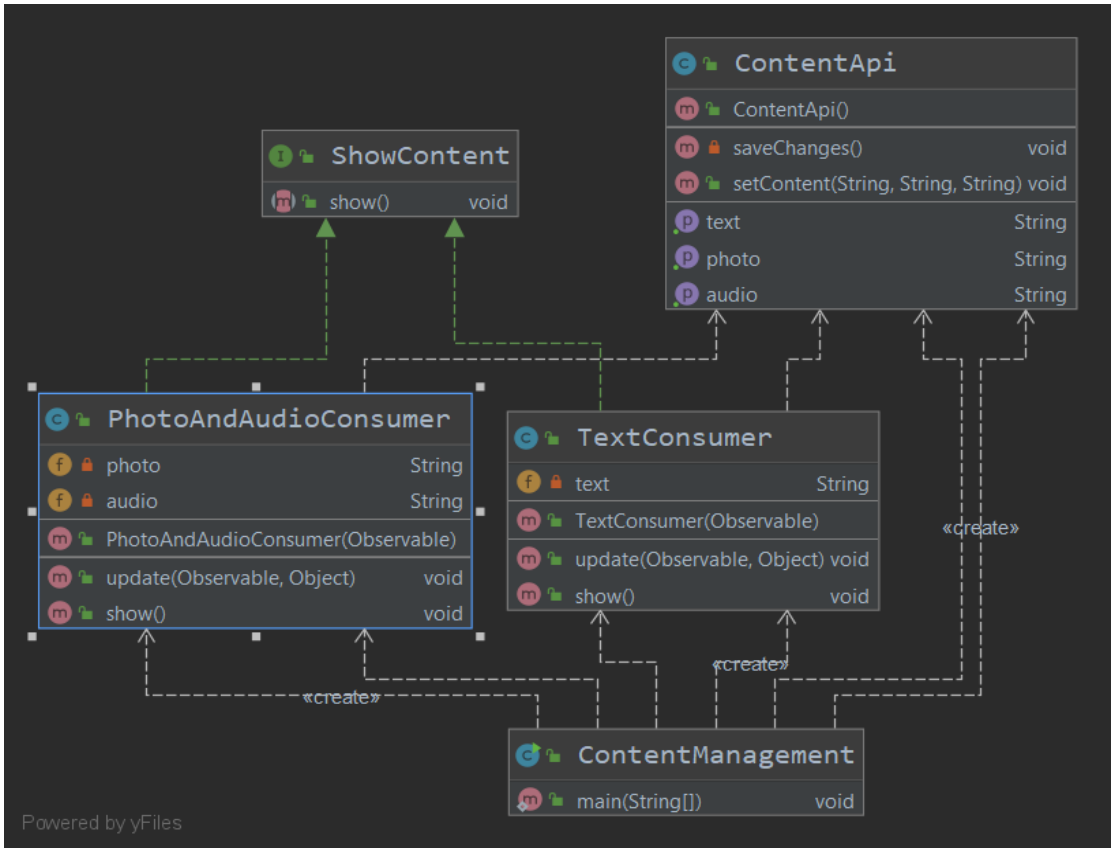



```
    public static void main(String[] args) {
        System.out.println("-------- Part 1 --------");
        LinearSolverDelux linearSolverDelux = new LinearSolverDelux(new MatrixInversion());
        linearSolverDelux.solveEquation();
        linearSolverDelux.setEquationSolver(new GaussianElimination());
        linearSolverDelux.solveEquation();

    }
}
```

Dinamik olarak çözüm metodunun değiştirilebilmesi sağlanmış oldu.

Part 2

İstenen program için Observer örüntüsü kullanıldı. Bir application programming interface(ContentApi) ve bu interface'den içerik alması beklenen müşteri rolündeki sınıflar oluşturuldu(TextConsumer, PhotoAndAudioConsumer). Müşteri rolündeki sınıflar Java'nın Oberver interface'inden ve api sınıfı Observable sınıfından türetildi. Böylelikle Java'ya entegre olan Observer örüntüsü kullanılmış oldu. Müşterilerin ihtiyaçlarının değişebileceği içim api Observer sınıflarını beslerken, farklı müşterilere farklı verileri besleyebilecek şekilde geliştirildi. Böylelikle müşteri ihtiyacı olan veriyi apiden çekebilecek ve yeni bir müşteri sınıfı oluşturulduğunda api'den o müşteriye özel verilerin çekilebilmesi sağlandı. ContentApi'de oluşabilecek herhangi bir içerik değişimi ile tüm müşterilerin update() metodu çağırılarak, müşterilere ait verilerin güncelliği korundu.




```
    System.out.println("-------- Part 2 --------");

    ContentApi contentApi = new ContentApi();
    TextConsumer textConsumer = new TextConsumer(contentApi);
    PhotoAndAudioConsumer photoAndAudioConsumer = new PhotoAndAudioConsumer(contentApi);


    contentApi.setContent( text: "Text1",  photo: "Photo1",  audio: "Audio1");
    contentApi.deleteObserver(textConsumer);
    contentApi.setContent( text: null,  photo: "photo2",  audio: "audio2");
}
```

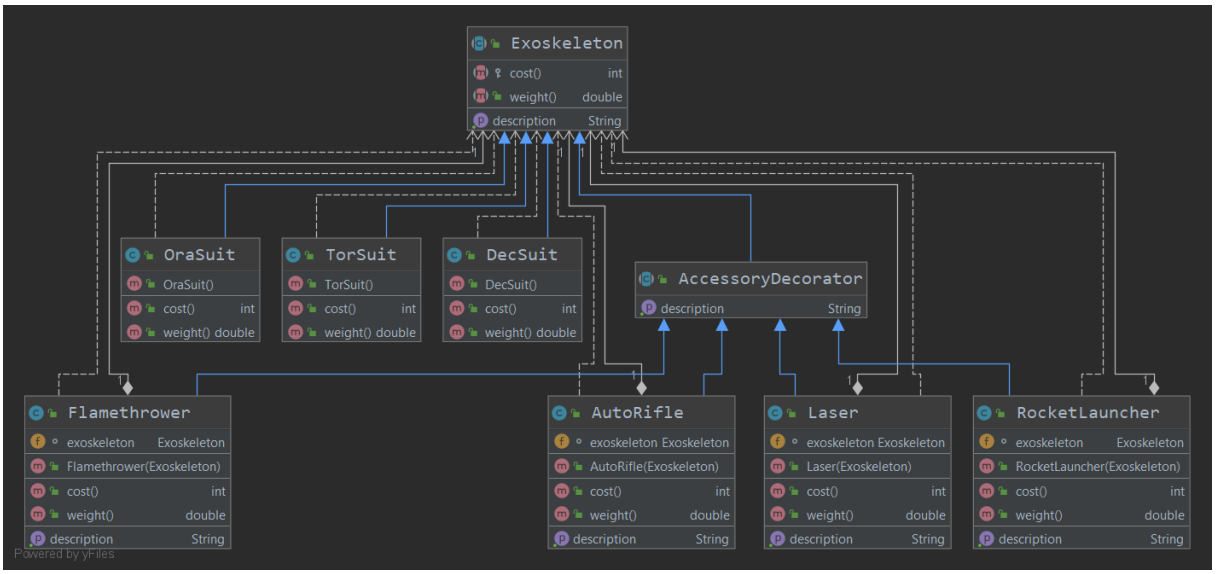
Api kullanılarak observer objeleri oluşturuldu.

Api'de olan veri değişikliği setContent() aracılığı ile ilgili müşterileri bilgilendi

Part 3

Problem çözümü için Decorator örüntüsü kullanıldı. Problem bir zırh geliştirme şirketi için dinamik olarak bu zırh oluşturulurken, zırhın ağırlığının ve maliyetinin hesaplanmasını istemekteydi.

Üç temel zırh çeşidi olduğu için bunlar doğrudan Exoskeleton üzerinden türetildi ve bu zırhların aksesuarları decorator örüntüsünün uygulanmasını sağlayacak şekilde AccessoryDecorator sınıfından türetildi.



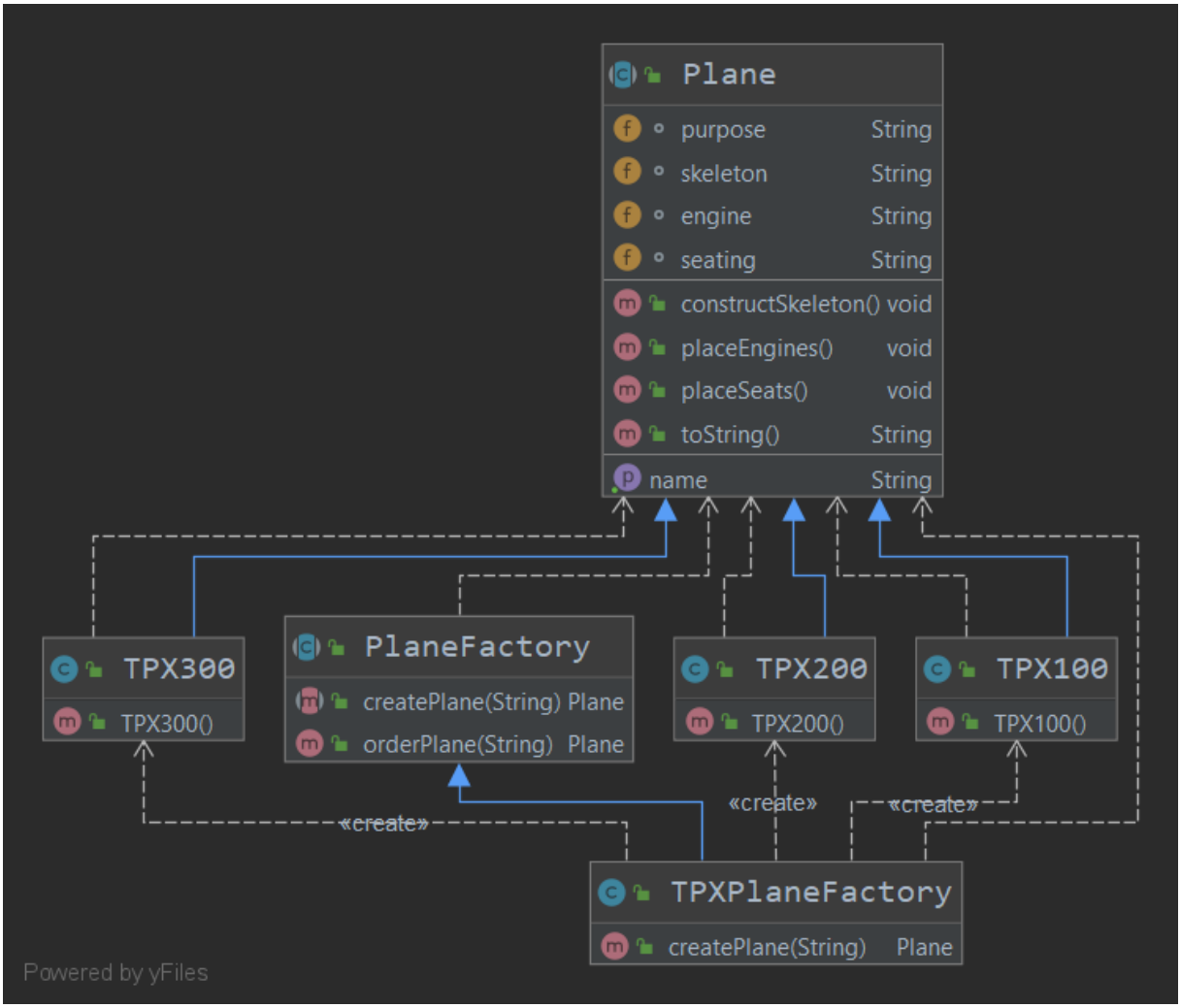

```java
public static void main(String[] args) {
    System.out.println("-------- Part 3 --------");

    Exoskeleton exoskeleton = new DecSuit();
    exoskeleton = new Flamethrower(exoskeleton);
    exoskeleton = new AutoRifle(exoskeleton);
    exoskeleton = new AutoRifle(exoskeleton);
    exoskeleton = new RocketLauncher(exoskeleton);

    System.out.println(exoskeleton.getDescription()
        + "\nCost: " + exoskeleton.cost() + "000TL"
        + "\nWeight: " + exoskeleton.weight() + "kg");


}
```

Öncelikle üç temel zırhdan birisi oluşturuldu. Sonrasında eklenen her aksesuar sarmalanarak oluşturuldu.

Sarmalanarak obje oluşturulması sayesinde maliyet ve ağırlık dinamik bir şekilde yeni oluşturulan objeler üzerinden hesaplanabilmiş oldu.

Part 4 – Birinci Kısım

Problem çözümünde Factory Method örüntüsü kullanıldı. PlaneFactory abstract sınıfından türetilen TPXPlaneFactory aracılığı ile üç farklı türde uçak createPlane() factory method'u ile üretilebilmiş oldu. Böylelikle TPX türü dışında oluşturulacak farklı türdeki uçaklar'da PlaneFactory'den türetilerek createPlane() factory method'u sayesinde üretilebilecek şekilde proje tasarlanmış oldu.



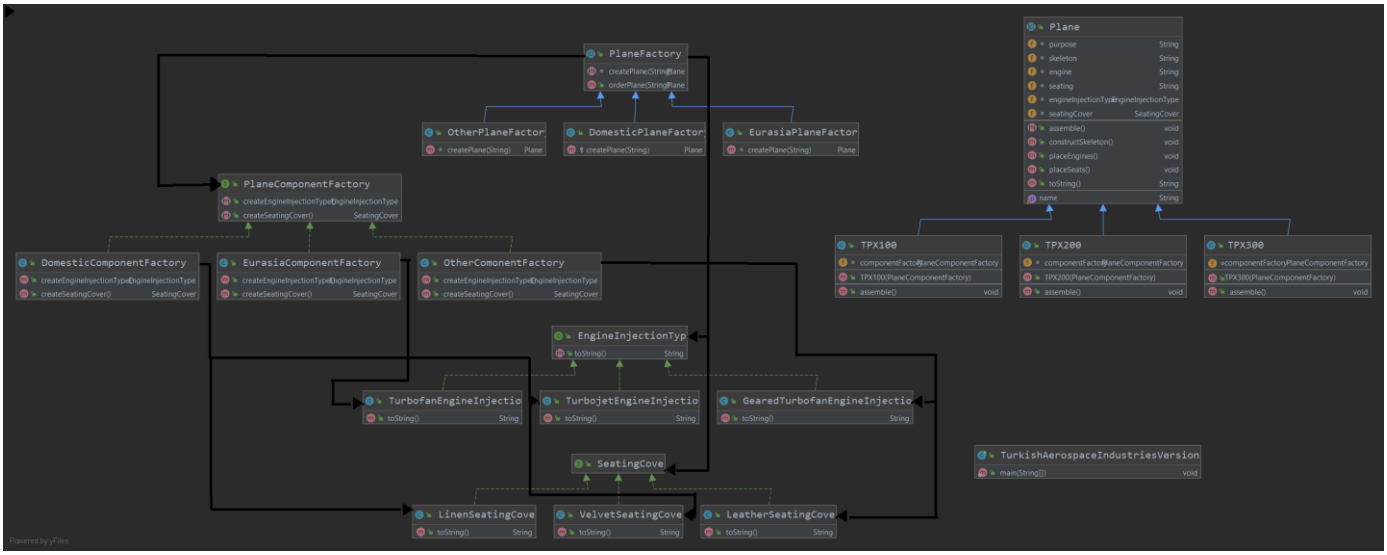

```
    public static void main(String[] args) {
        System.out.println("-------- Part 4a(Factory Method) --------");
        PlaneFactory tpxFactory = new TPXPlaneFactory();

        Plane plane = tpxFactory.orderPlane( type: "TPX100");
        System.out.println(plane.toString());

        plane = tpxFactory.orderPlane( type: "TPX200");
        System.out.println(plane.toString());

        plane = tpxFactory.orderPlane( type: "TPX300");
        System.out.println(plane.toString());
    }

}
```

Farklı türden TPX tipinde uçaklar TPXPlaneFactory aracılığı ile oluşturulmuş oldu.

Part 4 – İkinci Kısım

Bu tasarım oluşturulurken uçak türlerinin bölgelere göre değişim gösterebileceği parçalar için interface'ler oluşturuldu. Bölgeler için ayrı factory sınıfları oluşturuldu ve bu sınıflar bölgelere göre istenen parçaları oluşturabilecek şekilde tasarlandı.

TPX tipinde uçaklar için part 4'ün ilk kısmındaki tasarımın büyük kısmı korundu. (Plane sınıfı ve bundan türetilen sınıflar)




```java
    public static void main(String[] args) {
        System.out.println("-------- Part 4b(Abstract Factory) --------");

        PlaneFactory domesticFactory = new DomesticPlaneFactory();
        PlaneFactory eurasiaFactory = new EurasiaPlaneFactory();

        Plane plane = domesticFactory.orderPlane( type: "TPX200");
        System.out.println(plane.toString());

        plane = eurasiaFactory.orderPlane( type: "TPX300");
        System.out.println(plane.toString());
    }
}
```

Farklı bölgeler için farklı factory sınıfları kullanıldı.

Tüm Projenin Çıktısı (DesignPatternsHw1.jar Dosyasının Çıktısı)

```
-------- Part 1 --------
Matrix Inversion Method
Gaussian Elimination Method

-------- Part 2 --------
Waiting for photo data
Photo: Photo1
Waiting for audio data
Audio: Audio1
Waiting for text data.
Text: Text1
Waiting for photo data
Photo: Photo1
Photo: photo2
Waiting for audio data
Audio: Audio1
Audio: audio2

-------- Part 3 --------
Dec Suit, Flamethrower, Auto Rifle, Auto Rifle, Rocket Launcher
Cost: 760000TL
Weight: 37.5kg

-------- Part 4a(Factory Method) --------
>>> Building a TPX 100 <<<
The Aluminum Alloy skeleton was made.
The Single Jet Engine was made.
The 50 seats were made.
Name: TPX 100
Purpose: Domestic Flights
Skeleton: Aluminum Alloy
Engine: Single Jet Engine
Seating: 50 seats

>>> Building a TPX 200 <<<
The Nickel Alloy skeleton was made.
The Twin Jet Engines was made.
The 100 seats were made.
Name: TPX 200
Purpose: Domestic and short international flights
Skeleton: Nickel Alloy
Engine: Twin Jet Engines
Seating: 100 seats

>>> Building a TPX 300 <<<
The Titanium Alloy skeleton was made.
The Quadro Jet Engines was made.
The 250 seats were made.
Name: TPX 300
Purpose: Transatlantic Flights
Skeleton: Titanium Alloy
```

```
Engine: Quadro Jet Engines
Seating: 250 seats


-------- Part 4b(Abstract Factory) --------
-------------------------------------------------------
>>> Building a TPX200 Plane for DOMESTIC Market <<<
The Nickel Alloy skeleton was made.
The Twin Jet Engines was made.
The 100 seats were made.
Assembling TPX200 Plane for DOMESTIC Market

Name: TPX200 Plane for DOMESTIC Market
Purpose: Domestic and short international flights
Skeleton: Nickel Alloy
Engine: Twin Jet Engines
Seating: 100 seats
/// Market specific components \\\
Engine Injection Type: Turbojet Engine Injection
Seating Cover: Velvet Seating Cover
-------------------------------------------------------
-------------------------------------------------------
>>> Building a TPX300 Plane for EURASIA Market <<<
The Titanium Alloy skeleton was made.
The Quadro Jet Engines was made.
The 250 seats were made.
Assembling TPX300 Plane for EURASIA Market

Name: TPX300 Plane for EURASIA Market
Purpose: Transatlantic Flights
Skeleton: Titanium Alloy
Engine: Quadro Jet Engines
Seating: 250 seats
/// Market specific components \\\
Engine Injection Type: Turbofan Engine Injection
Seating Cover: Linen Seating Cover
```
