Gebze Technical University
Computer Engineering

CSE443 - Object Oriented Analysis and Design
Fall 2018 - 2019

HOMEWORK 2 REPORT

EMRE ÇELİK
141044024

Part 1

1-)

Comparable arayüzünün implement edilmemiş olması nedeniyle clone() metodu çağrılamayacak ve kod compile olmayacaktır. Object sınıfının clone() metodu(protected method) bir şekilde çağrılıp(public metot yardımıyla) bu işlem denenir ise bu durumda object' in clone metodu default olarak CloneNotSupportedException fırlatacaktır.

2-)

Cloneable interface i implement edilir, clone() override edilir, ve CloneNotSupportedException gibi bir exception fırlatılır.

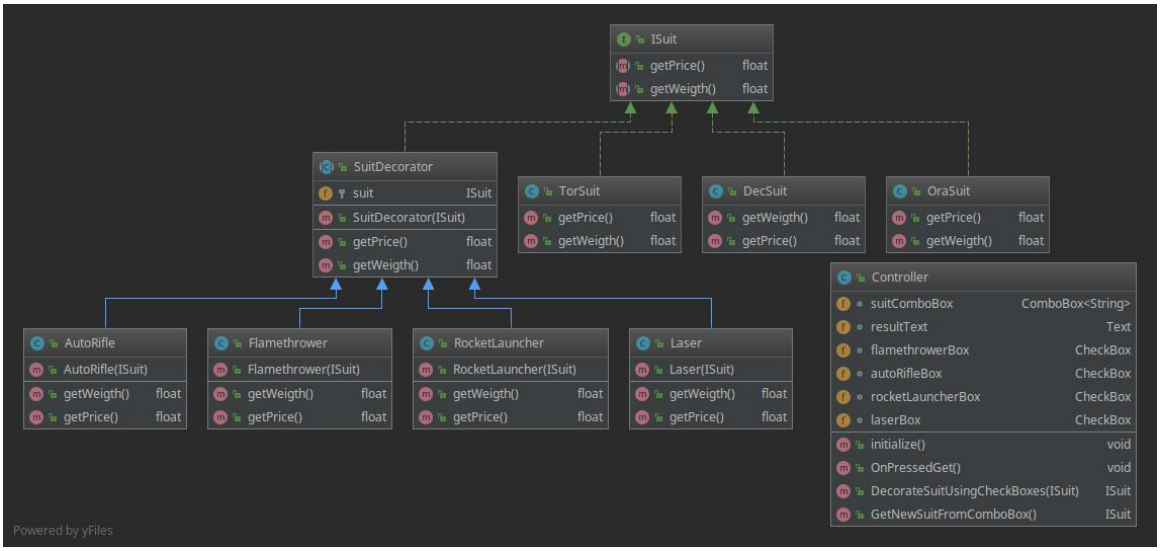
3-)

1. Soruya göre bakacak olursak, eğer Parent clone() metodunu override ettiğinde exception fırlatmışsa, clone alınmayacak, eğer parent clone()' u tamamen implement etmişse objenin kopyası alınmış olup, deep copy yapılmış olacak(objenin kopyası alınmış olacak).
2. Soruya göre bakacak olursak eğer clone methodunda exception fırlatılmışsa bu soruyu doğrulayacak ve clone edilmeycek. Fakat implement edilmişse deep copy yapılacak ve singleton' un kopyalanması önlenemeyecek.

Part 2

Temelde 3 farklı suit türü olduğundan, ISuit isminde bir interface oluşturulup, price ve weight' i veren getter' lar tanımlandı. Aksesuarların dinamik olarak eklenmesi için Decorator Pattern' i kullanıldı. SuitDecorator sınıfı burada decorator görevi görüp, ISuit tipinde bir obje saklayıp, Constructor' da bu objeyi alınıp atama gerçekleştirildi. Suit olan objeler metot override ederken kendi price ve weight' lerini döndürdüler. Decorator olanlar ise super' in weight ve price' larının üzerine kendi miktarlarını ilave ederek override işlemlerini tamamladılar. Aksesuarlar SuitDecorator' dan türetildi. UI ile etkileşimi FXML ile sağlanıp SUIT tipleri ComboBox ile seçildi ve aksesuarların checkbox' lar ile eklenmesi sağlandı.

Class Diagram

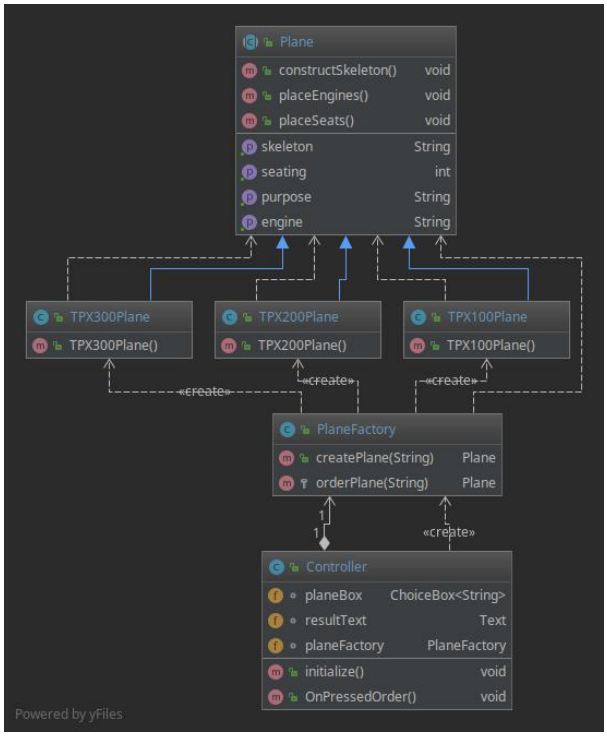


Part 3

Part 3.1

Plane üretimi için Factory Pattern' i kullanıldı. PlaneFactory isimli bir sınıf oluşturulup createPlane ve orderPlane metodları ile alınan String type kullanılarak istenilen modelde uçak üretimi sağlandı. Üretilen uçak Plane isimli abstract bir sınıf ile implement edilip, constructSkeleton, placeEngines ve placeSeats metodlarının içerisinde simülasyon anlamı katılması için yapılan işlemler ve işlemin yapıldığı memberlar(uçağın özellikleri) System.out' a bastırıldı. Farklı modellerdeki uçaklar bu sınıftan türetilip, constructor' da member' lar modele göre atandı. Bu sayede Factory' de yeni bir uçak üretildiği zaman, çıktı olarak önce purpose bastırıldı, sonra ise uçağın construct ve place gibi üretim metodları çağrıldı. Simülasyon adımları System.out' dan okunup UI' a yansıtıldı.

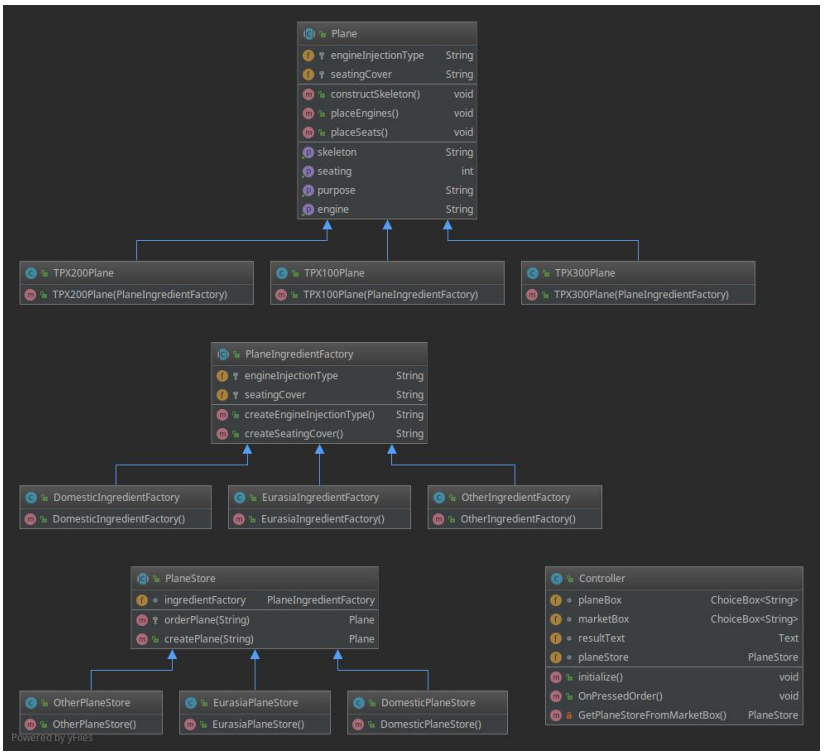
Class Diagram



Part 3.2

Farklı marketlerden uçak üretimi için Abstract Factory Pattern' i kullanıldı. PlaneFactory -> PlaneStore olarak değiştirilip abstract yapıldı. Uçağa ek özellikler katılması için ingredientFactory nesnesi tutuldu. createPlane ve orderPlane ingredientFactory nesnesini kullanarak abstract class' da implement edildi. Farklı store' lar açmak için bu abstract sınıf türetilip Eurasia, Domestic ve Other olarak 3 store oluşturuldu. IngredientFactory abstract sınıfında engineInjectionType ve seatingCover member' ları tutulmuş olup, create metodlarında bu member' ların üretimi sağlandı. 3 Store için bu abstract sınıf inherit edilip, bu member' lar constructor' da atandı. Böylelikle, Store' lar oluşturulduğunda, Constructor' da ingredientFactory member' larına kendi ingredientFactory' lerinin ataması yapılıp abstract metodların implement edilmesi sayesinde market' lere özel uçaklar üretildi. IngredientFactory, Plane' lere parametre olarak üretim aşamasında verildi. Plane ingredient' i constructor' da kullanarak engineInjectionType ve seating Cover(yeni memberlar)' ın atamasını yaptı. Plane' e eklenen engineInjectionType ve seating Cover, place metotlarında simülasyon aşamasında belirtildi.

Class Diagram



Class Diagram (Detaylı)