CSE443
OBJECT ORIENTED ANALYSIS AND DESIGN
HW2
REPORT

Hakkı Erdem Duman
151044005

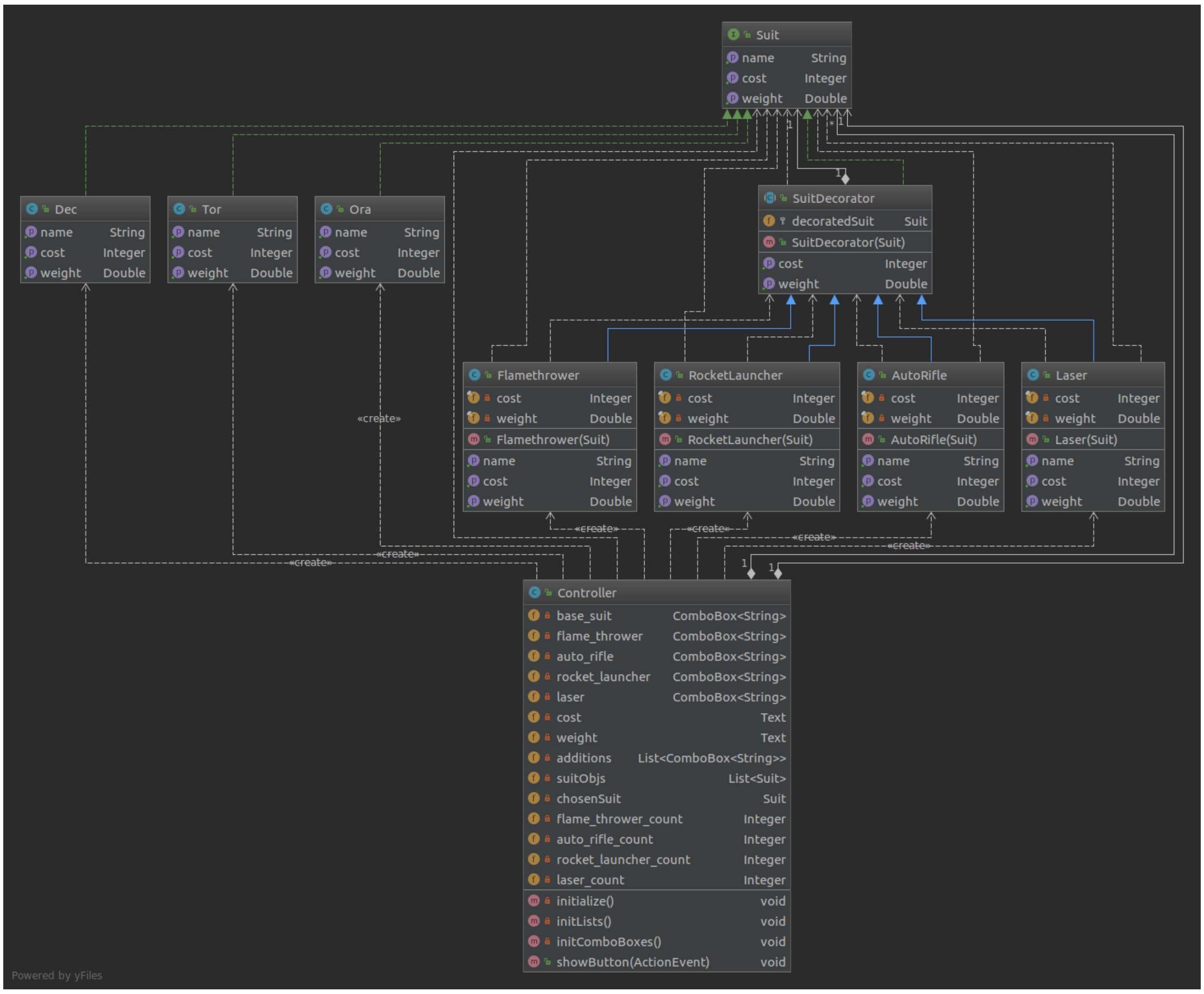
Question 1

1) Object’s clone method does not allow to get clone by default, it throws “CloneNotSupportedException”. In Oracle documents, it says “First, if the class of this object does not implement the interface Cloneable, then a CloneNotSupportedException is thrown”.
https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#clone()

2) Cloneable interface is implemented and clone method is overridden to throw “CloneNotSupportedException” exception.

3) If Cloneable interface is implemented, singleton object can be deep copied and that’s not the thing that we want. We can prevent it by throwing “CloneNotSupportedException”.

Question 2



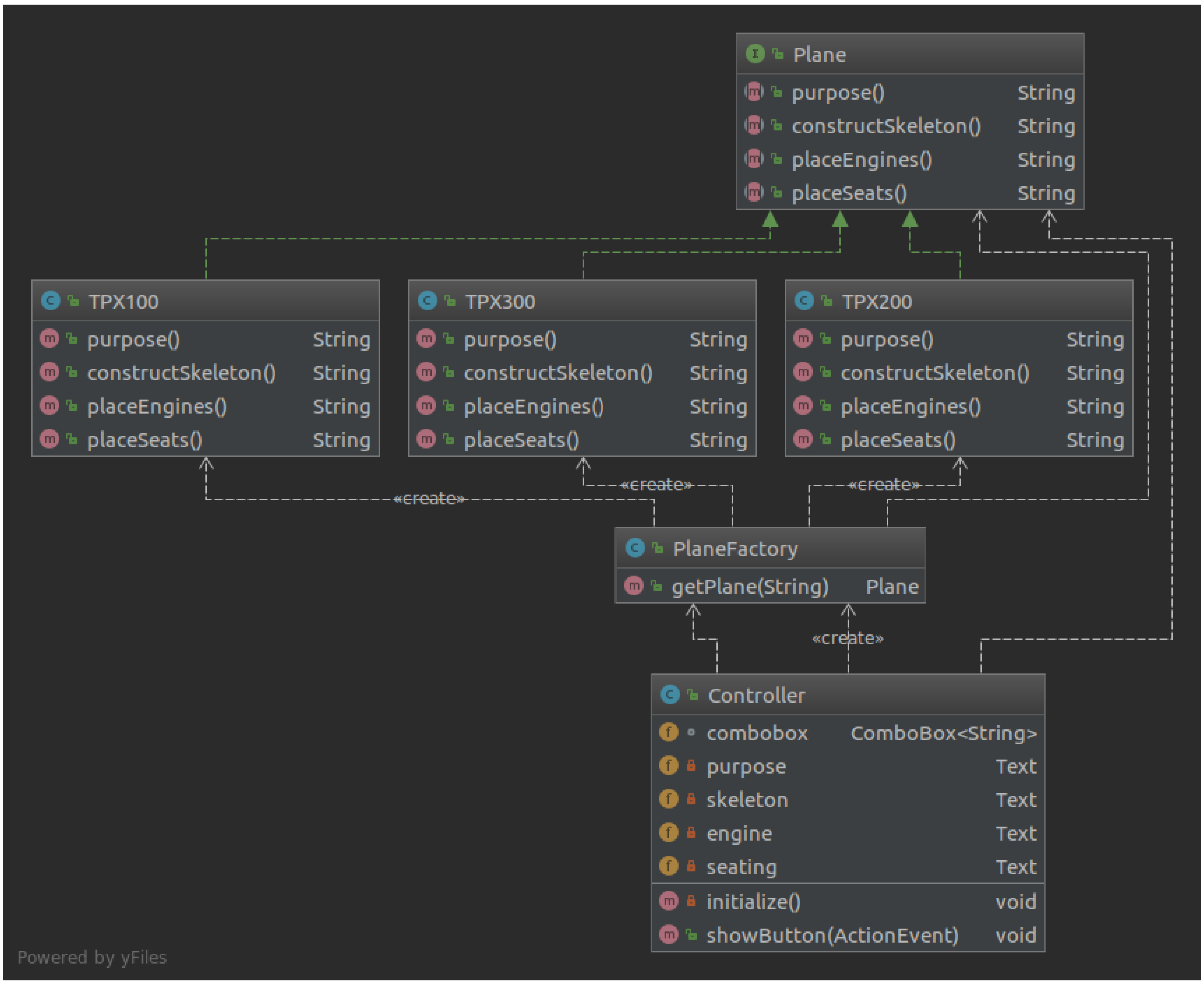
In this question, decorator pattern is used. Suit types (dec, ora, tor) implement "Suit" interface and "SuitDecorator" abstract class implements it to provide some functions for its extenders (FlameThrower, RocketLauncher, AutoRifle, Laser).

In user interface, user can change suit type and number of decoration items. "Show" button is needed to be clicked to see the result.

Whenever "show" button is clicked, suit type and number of decors are gotten and objects are created in for loops.

Jar file is "HW2_Part2/out/artifacts/JavaFXApp/hw2part2.jar"

Question 3.1

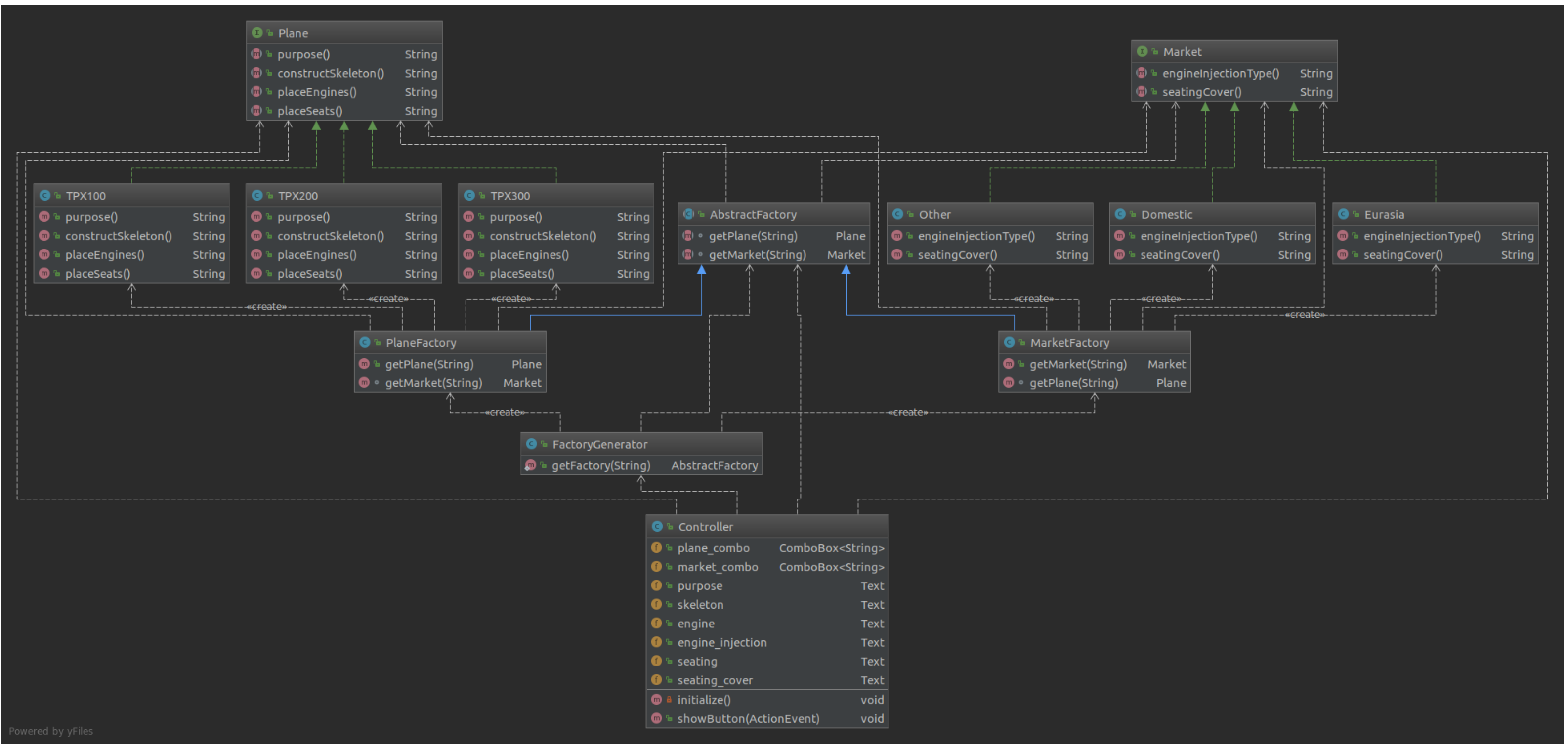


In this part, factory pattern is used. Logic of factory pattern is generating object according to given parameter. There is an interface called “Plane” and this interface is implemented by three classes that symbolize three type of planes. In “PlaneFactory” class, there is a method called “getPlane” and this method generates an object of one of three types of planes, according to its parameter.

In user interface, type of planes is chosen and “produce” button is clicked. It generates the construction phase of the chosen plane.

Jar file is “HW2_Part3_1/out/artifacts/JavaFXApp/hw2part3_1.jar”

Question 3.2



In this part, abstract factory pattern is used. Difference between this part and the previous part is getting two different input from user. Since these inputs' types are different, there should be a class that generates a factory before getting an object. This is what "FactoryGenerator" class does. Its method called "getFactory" gets a parameter and generates either plane or market factory. After that, PlaneFactory and MarketFactory can generate their objects according to their string parameters. "AbstractFactory" class should be used to gather PlaneFactory and MarketFactory under a single roof.

In user interface, user choose plane type and market. After the button is clicked, the plane's production steps and market details are shown.

Jar file is HW2_Part3_2/out/artifacts/JavaFXApp/hw2part3_2.jar