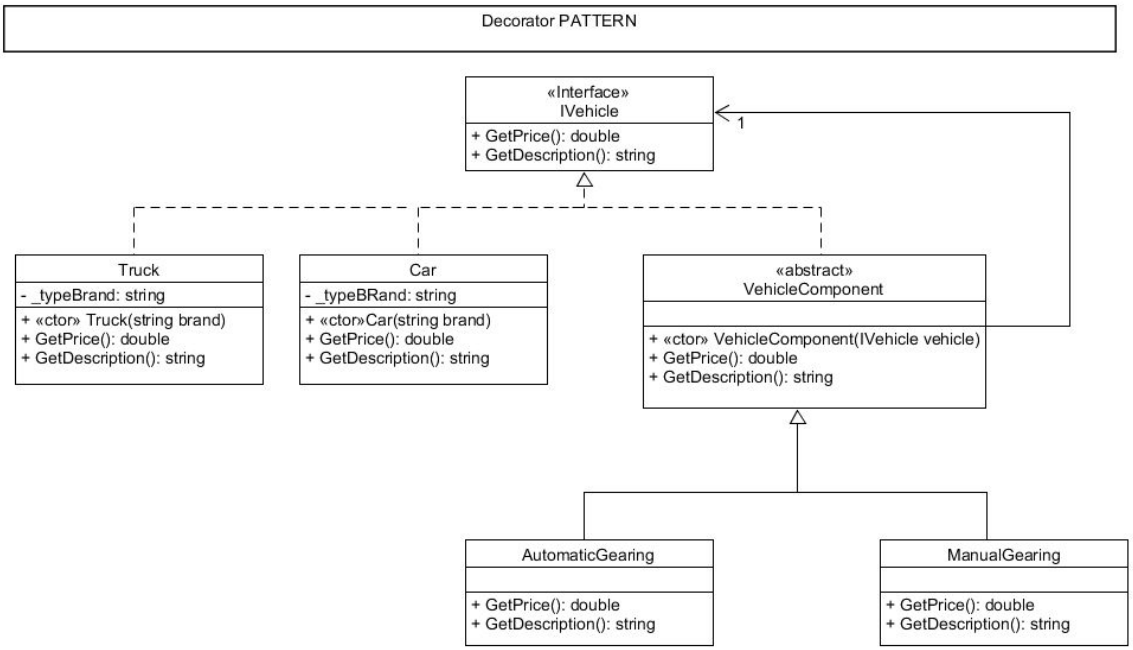
Decorator Pattern

Mrunal Patil - 3101851
Emil Karamihov - 2992884

Application Description: The application is built to manage the purchase of different Vehicles like Cars, Trucks, etc. It should also be possible to add extra features to the vehicle to customize it. Thus, Decorator design pattern is used to design this application. In the pattern every vehicle or customizable object implements GetDescription() and GetPrice() methods as these are common features between all of them. GetDescription() will give the total description of the Car. eg:-(Polo-Automatic Gearbox). GetPrice() will keep calculating the price depending on the objects which are added to customize the vehicle.



Maintainability, Reusability, Extensibility.

1. The decorator pattern allow more flexibility than only class inheritance. If there are a lot of different components which are the same and or needed to be added. It allows adding and removing behaviours at runtime. Based on its flexibility (reliving us from making multiple complicated objects), now with this pattern we make less and smaller object thanks to the interface. Making it easier to maintain the code and because of its reusability. In addition, it is very easy to extend the program with more objects they just need to implement the interface and its methods.

Pros

2. Much more flexible than class inheritance.

3. Allows adding and removing behaviors at runtime.
4. Allows combining several additional behaviors by using multiple components.
5. Allows composing complex objects from simple ones instead of having monolithic classes that implement every variant of behavior.

Cons

1. It is hard to configure a multi-wrapped object.
2. Lots of small classes.