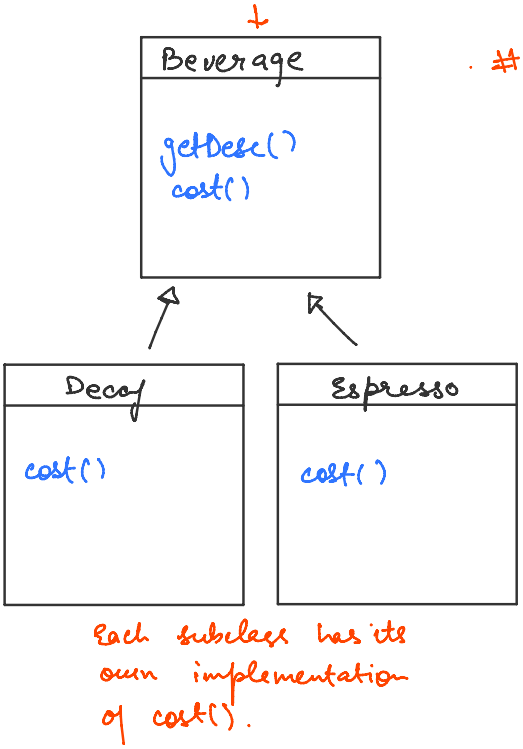
Decorator Pattern

04 September 2020 11:02 AM

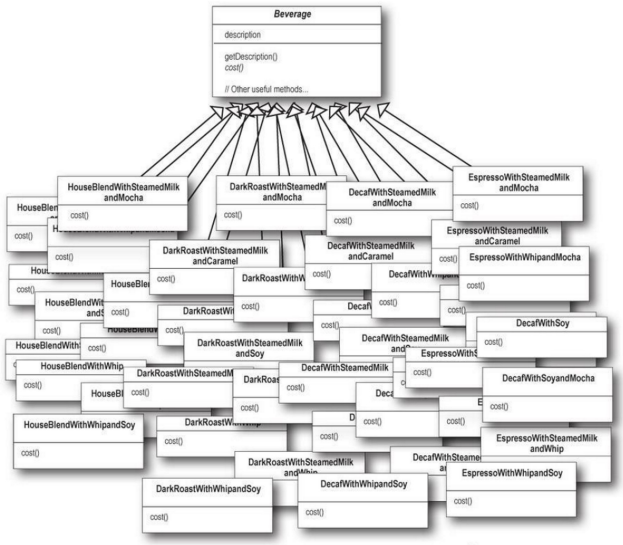
Q. Why do we need Decorator Pattern?

Abstract class, so we can write basic implementations.

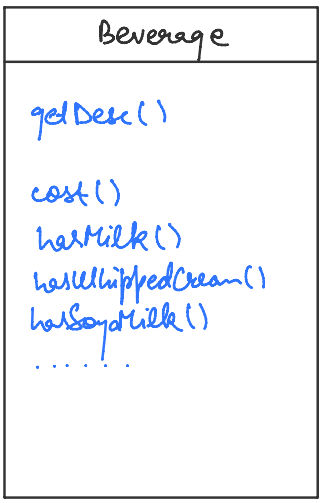


Each subclass has its own implementation of cost().

On observing we can notice that there can be many combinations of coffee and implementing all the classes will lead to class explosion.



#first solution (but a bad one)

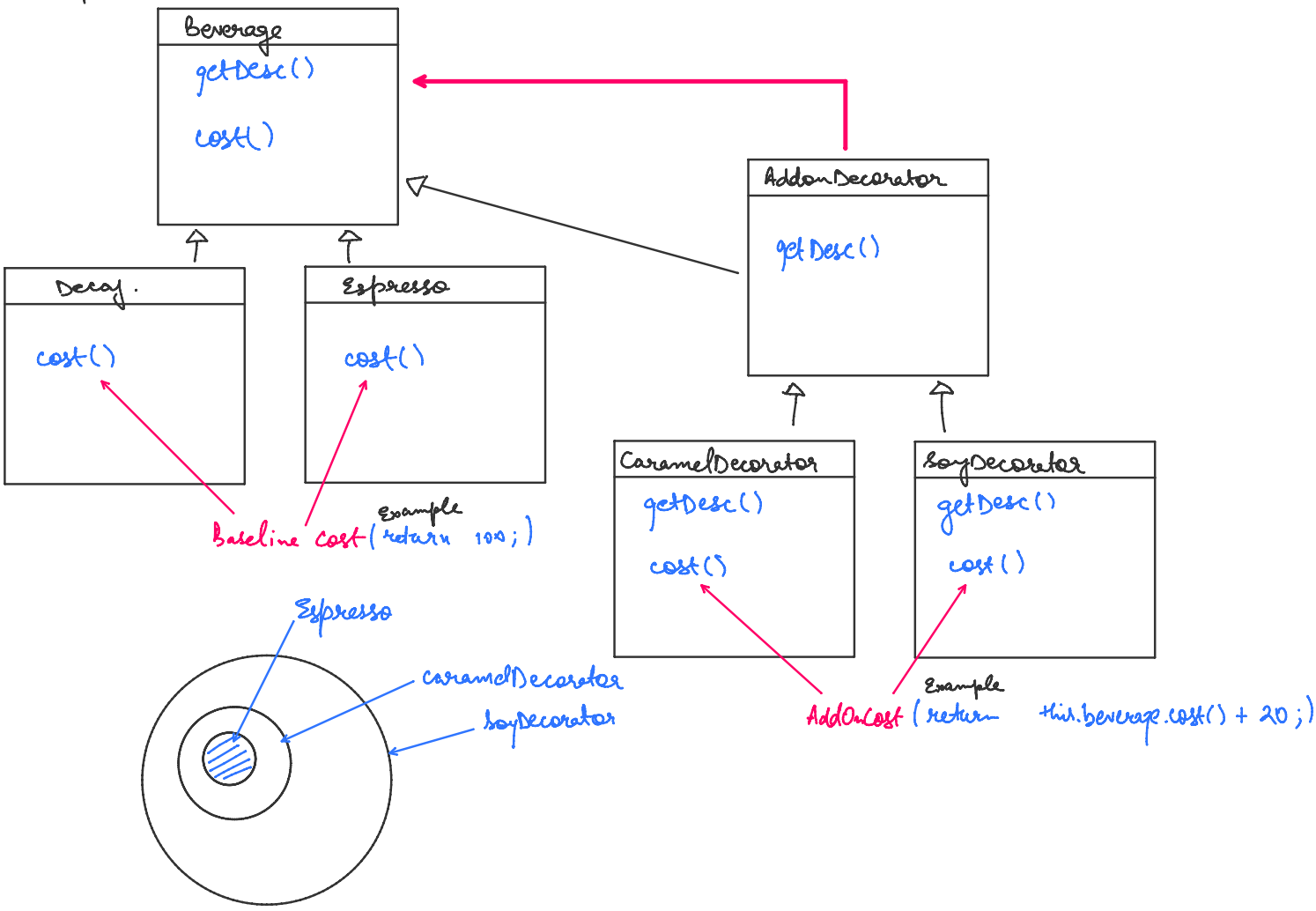


Booleans

We can get all the combinations of coffee but, each boolean will require some sort of if-else which will create chaos if new types of coffees are to be added.

1. source code needs to be changed to add new items.
2. subclasses inheriting all the unnecessary methods

Using the Pattern.



Decorator pattern allows a user to add new functionality to an existing object without altering its structure. This type of design pattern comes under structural pattern as this pattern acts as a wrapper to existing class.

This pattern creates a decorator class which wraps the original class and provides additional functionality keeping class methods signature intact.

We are demonstrating the use of decorator pattern via following example in which we will decorate a shape with some color without alter shape class.