Decorator ID

Student:

Izabela Kuźniar

Teacher:

Andrea Corradini

Course:

Software Design Patterns

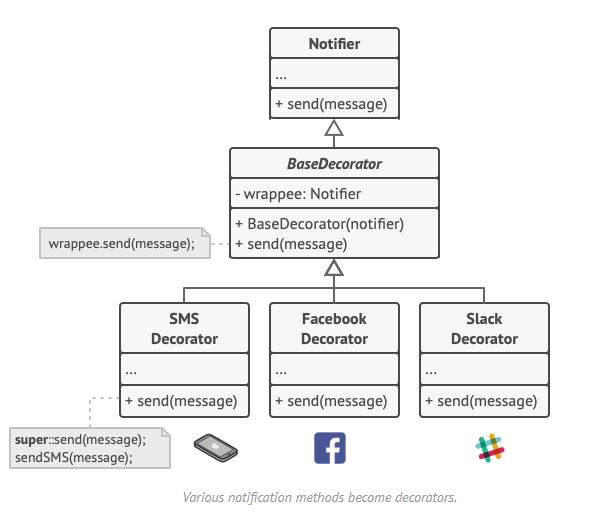
# Name and category

Decorator, also known as Wrapper is a structural design pattern.

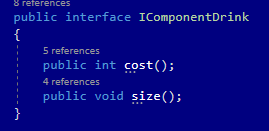
# Intent:

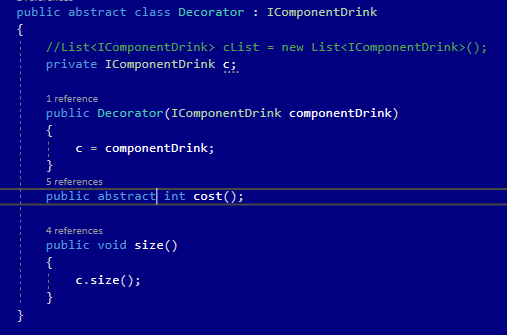
Lets attach new behaviors to objects by placing these objects inside special wrapper objects (that contains the behavior).

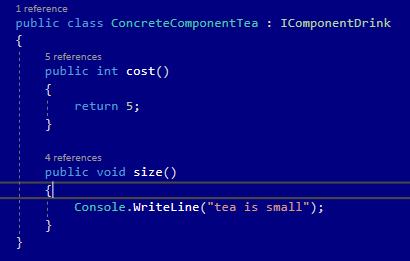
# Structure as a UML class diagram

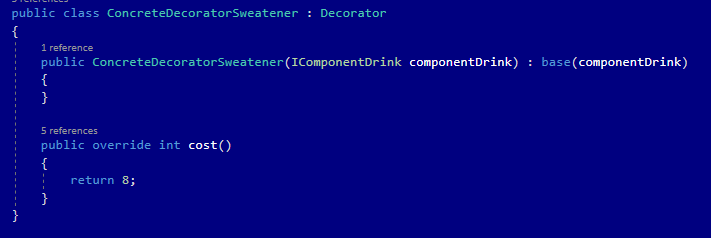


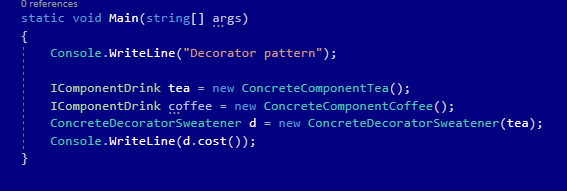
# Implementation:











# Consequences:

Benefits:

* Ability to extend behavior of the object without making a new subclass.
* Ability to add or remove responsibilities from an object at runtime.
* Ability to combine several behaviors by wrapping an object into multiple decorators.
* Single Responsibility Principle. You can divide a monolithic class that implements many possible variants of behavior into several smaller classes.

Drawbacks:

* Hard to remove a wrapper form the wrappers stack.
* Hard to implement the decorator such a way that its behavior doesn’t depend on the order in the decorators stack.

# Known uses

* Stream class in Java

# Related patterns

1. Adapter changes the interface of an existing object, while Decorator enhances an object without changing its interface. In addition, Decorator supports recursive composition, which Adapter does not.
2. Adapter provides a different interface to the wrapper object, Proxy provides it with the same interface, and Decorator provides it with an enhanced interfae.
3. Chain of Responsibility and Decorator have very similar structures. Both patterns rely on recursive composition to pass the execution through a series of objects. However, there are several crucial differences.
4. Composite and Decorator have a similar structure diagrams, both rely on recursive composition to organize open-ended number of objects.
5. Applying the prototype pattern allows to clone complex structures such as Decorator or Composite and not re-constructing them from scratch.
6. Decorator enables changes of the skin of an object, while Strategy lets you change the insides.
7. Decorator and Proxy have similar structures, but very different intents, Both patters are built on the composition principle, where one object is supposed to delegate some of the work to another. The difference is that a Proxy usually manages the life cycle of its service object on its own, whereas the composition of Decorators is always controlled by the client.