Декоратор представляет структурный паттерн, который позволяет динамически подключать к объекту дополнительную функциональность.

Наш код демонстрирует использование паттерна Decorator для обработки и сохранения сообщений с использованием различных методов хеширования (MD5, SHA512) и шифрования (RSA).

class Decorator - собственно декоратор имеет тот же базовый класс, что и декорируемые объекты, что позволяет ему взаимодействовать с другими объектами IWriter в цепочке вызовов.

Класс декоратора также хранит ссылку на декорируемый объект в виде объекта IWriter и реализует связь с базовым классом как через наследование, так и через отношение агрегации.

Он также предоставляет виртуальный метод Save, который просто делегирует вызов методу Save оборачиваемого объекта.

Другие структурные паттерны:

1. Адаптер: предназначен для преобразования интерфейса одного класса в интерфейс другого
2. Фасад: позволяет скрыть сложность системы с помощью предоставления упрощенного интерфейса для взаимодействия с ней
3. Компоновщик: группы объектов в древовидную структуру по принципу "часть-целое и позволяет клиенту одинаково работать как с отдельными объектами, так и с группой объектов
4. Заместитель: предоставляет объект-заместитель, который управляет доступом к другому объекту.
5. Мост: позволяет отделить абстракцию от реализации таким образом, чтобы и абстракцию, и реализацию можно было изменять независимо друг от друга.
6. Приспособленец: способ оптимизации работы с большим количеством похожих объектов, чтобы экономить память и ресурсы, используя общие данные.

Паттерн Decorator - это структурный паттерн проектирования, который позволяет добавлять новое поведение или функциональность объекту динамически, не изменяя его основного интерфейса. Он используется для расширения функциональности класса без необходимости изменения его кода.

В данном коде паттерн Decorator используется для добавления дополнительной функциональности к классам, реализующим интерфейс `IWriter`, который имеет метод `Save(string? message)` для сохранения сообщения. Основной класс `Decorator` является абстрактным и реализует интерфейс `IWriter`. Он имеет в себе ссылку на другой объект `IWriter`, который является "оборачиваемым" объектом.

Конкретные декораторы (`DecMD5`, `DecSHA512`, `DecRSA`) расширяют функциональность базового класса `Decorator`, добавляя свой функционал к методу `Save`. Например, `DecMD5` вычисляет хеш MD5 для сообщения перед его сохранением, `DecSHA512` вычисляет хеш SHA512, а `DecRSA` шифрует хеш с использованием RSA перед его сохранением.

При создании объекта декоратора (например, `DecMD5`), ему передается другой объект `IWriter`, к которому он добавляет свою функциональность. Таким образом, можно комбинировать различные декораторы для получения различных комбинаций функциональности.

В основной программе (`Program.Main`) показаны различные примеры использования декораторов для сохранения сообщений с различной дополнительной обработкой (MD5, SHA512, RSA-шифрование).

В данном случае, расширение функциональности достигается за счёт использования композиции и полиморфизма.

1. \*\*Композиция\*\*: Основной класс `Decorator` содержит в себе объект типа `IWriter`, который он оборачивает. При создании экземпляра декоратора передается уже существующий объект `IWriter`, который он будет использовать для выполнения базовой функциональности. Таким образом, новый функционал добавляется, не затрагивая существующий код.

2. \*\*Полиморфизм\*\*: Все декораторы реализуют общий интерфейс `IWriter`, что позволяет им заменять друг друга в программе. Таким образом, клиентский код может работать с любым декоратором, не зная об их конкретной реализации. Например, в основной программе можно использовать любую комбинацию декораторов для обработки сообщений, и клиентский код останется неизменным.

Итак, расширение функциональности достигается за счёт создания новых классов-декораторов, которые оборачивают существующий функционал, добавляя к нему новые возможности. При этом они сохраняют совместимость с существующими интерфейсами, что позволяет использовать их в существующем коде без изменений.