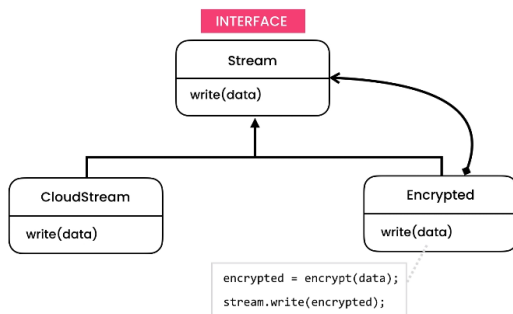# Decorator Design Pattern
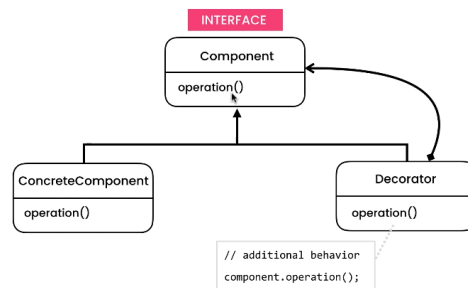
This pattern is used for adding additional behavior to an object.

Class Diagrams

Our Example	The pattern in Gang of Four's book

Our Example

INTERFACE
Stream
write(data)

CloudStream
write(data)

Encrypted
write(data)

```
encrypted = encrypt(data);
stream.write(encrypted);
```

The pattern in Gang of Four's book

INTERFACE
Component
operation()

ConcreteComponent
operation()

Decorator
operation()

```
// additional behavior
component.operation();
```

Explanation

Here we want to implement a class for storing data. The CloudStream class is responsible for reading and writing plain data. The EncriptedCloudStream class is responsible for encrypting the plain data. The CompressCloudStream class is responsible for compressing the plain data. Using the CloudStream object is needed for storing data and using EncriptedCloudStream and CompressCloudStream objects are optional. In the Stream interface, we declare operations we need to support in various types of streams. So. the CloudStream and EncriptedCloudStream and CompressCloudStream (it is not in the picture but it is in code) classes are three implementations of this interface.