**DECORATOR**

→ Conceito:

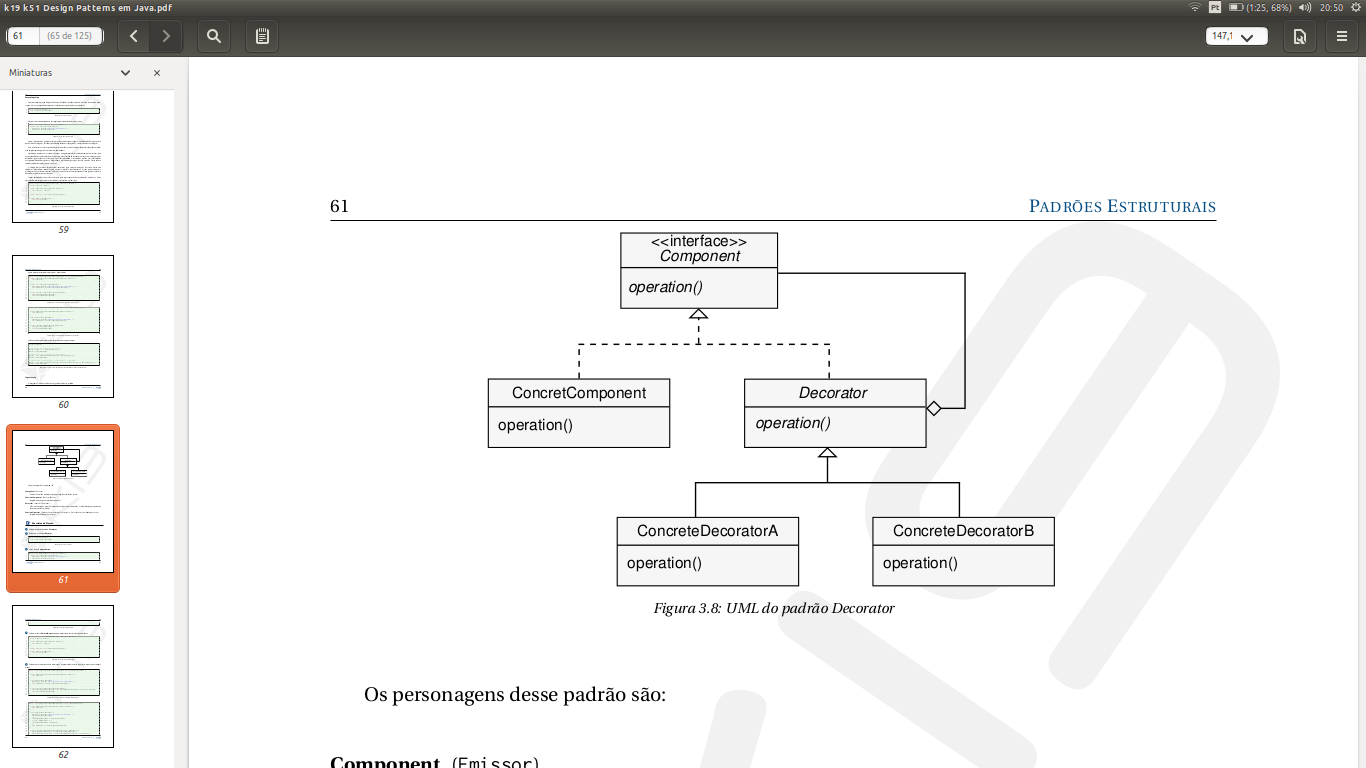
O padrão de projeto Decorator tem como objetivo adicionar funcionalidade a objetos individuais dinamicamente e não a uma classe, para não alterar a classe que define um objeto padrão é necessário que para cada funcionalidade adicional (decoração) a criação de um novo objeto (decorador). Principais características são:

* Os decoradores são do mesmo tipo que os objetos que eles decoram;
* e assim é possível usar um objeto decorador ao invés do objeto original;
* É possível usar mais de um decorador para um objeto;
* o decorador implementa o comportamento que é desejado para o sistema;
* os objetos podem ser decorados dinamicamente, em tempo de execução, e sem limite de decoradores que podem ser utilizados.

→ Aplicação:

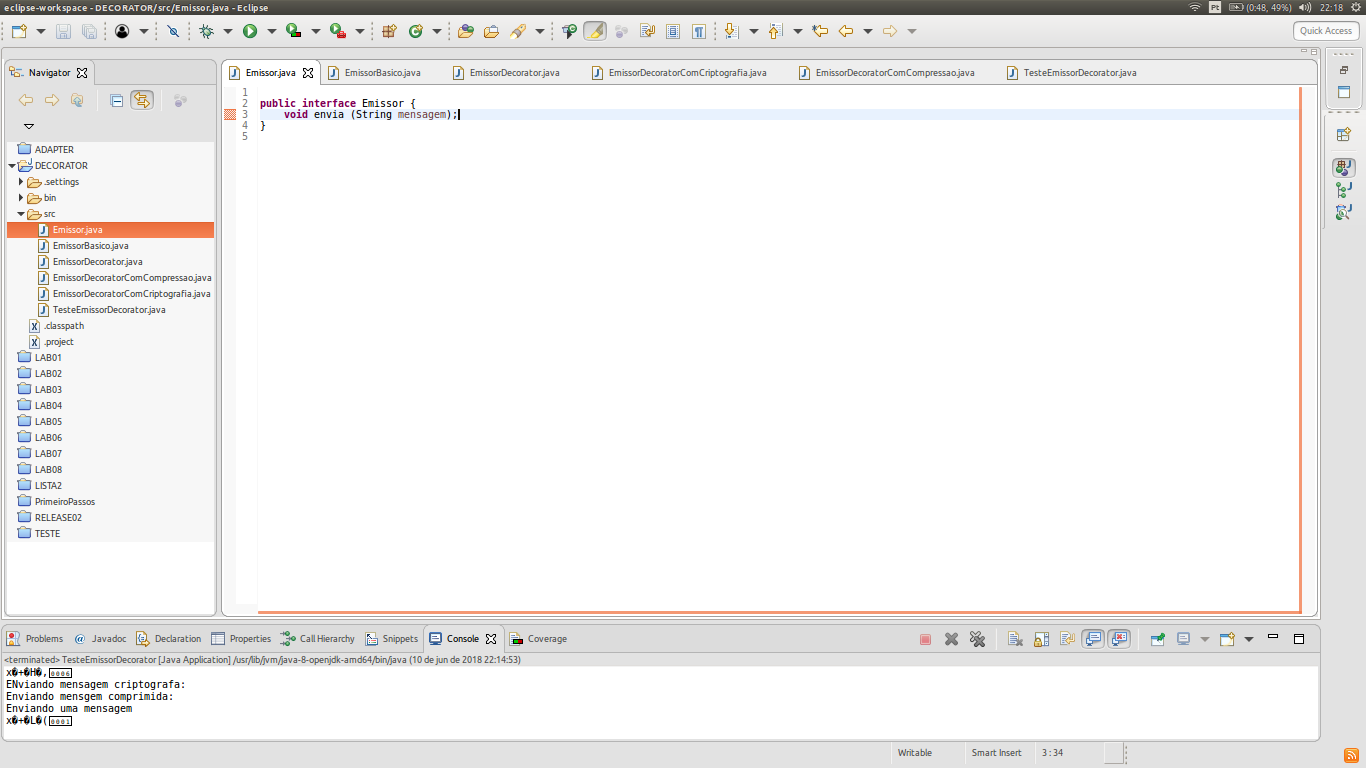
* incluir um comportamento a um objeto, sem afetar outros objetos;
* o padrão decorator é usado se deseja implementar comportamentos que possam ser removidos;
* quando não é possível extender várias subclasses para aplicar seus métodos então é necessário ter classes independentes que suportem cada combinação.

→ Diagrama UML Padrão Decorator

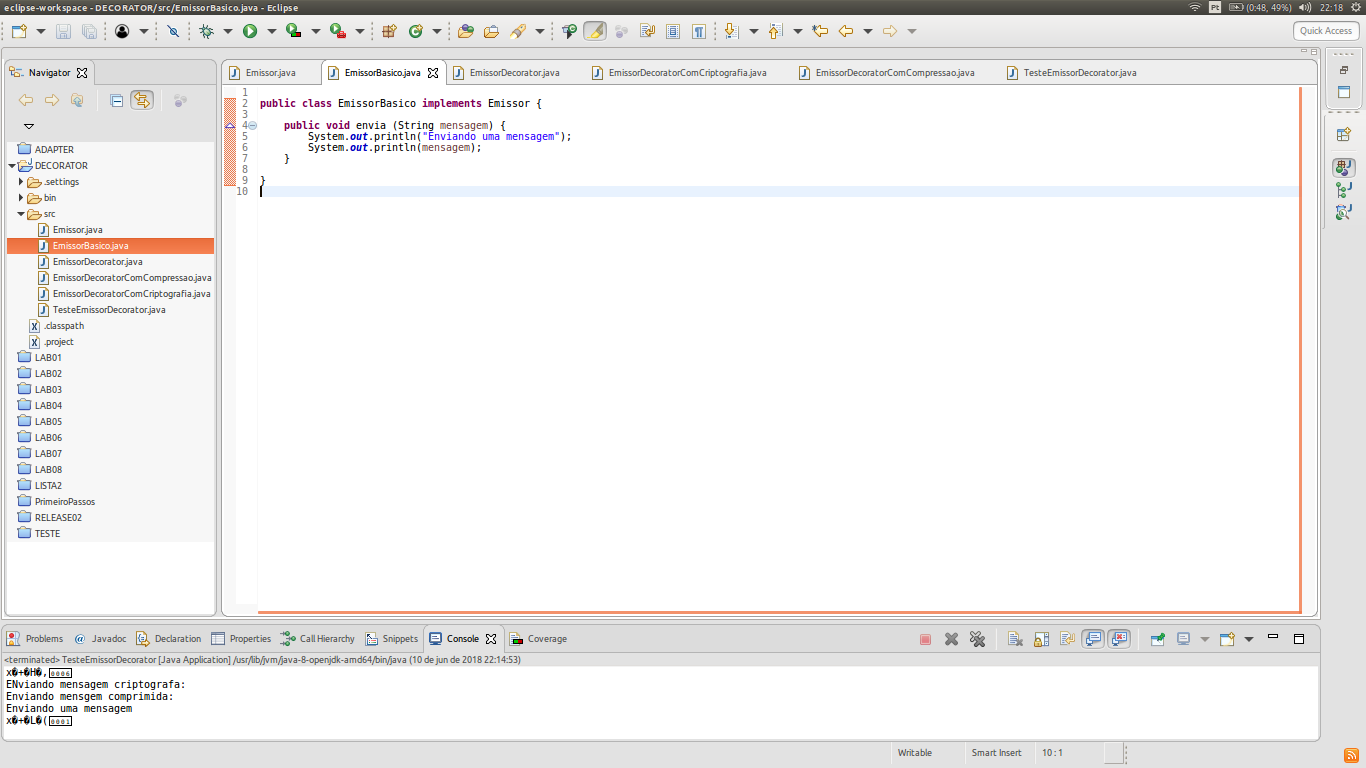


→ Exemplo

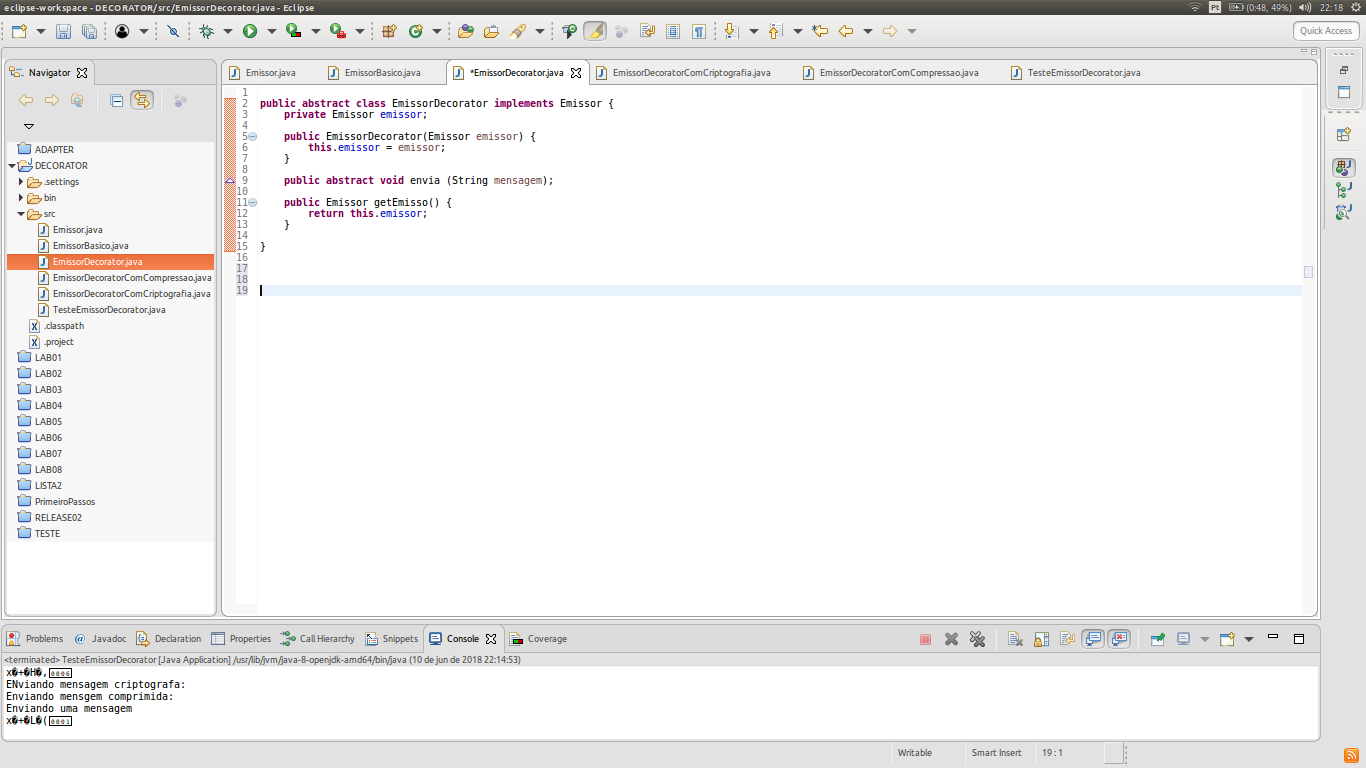
**Component:** Define a interface para objetos que podem ter responsabilidades acrescentadas dinamicamente.



**ConcreteComponent**: Define um objeto que pode ter responsabilidades atribuídas.

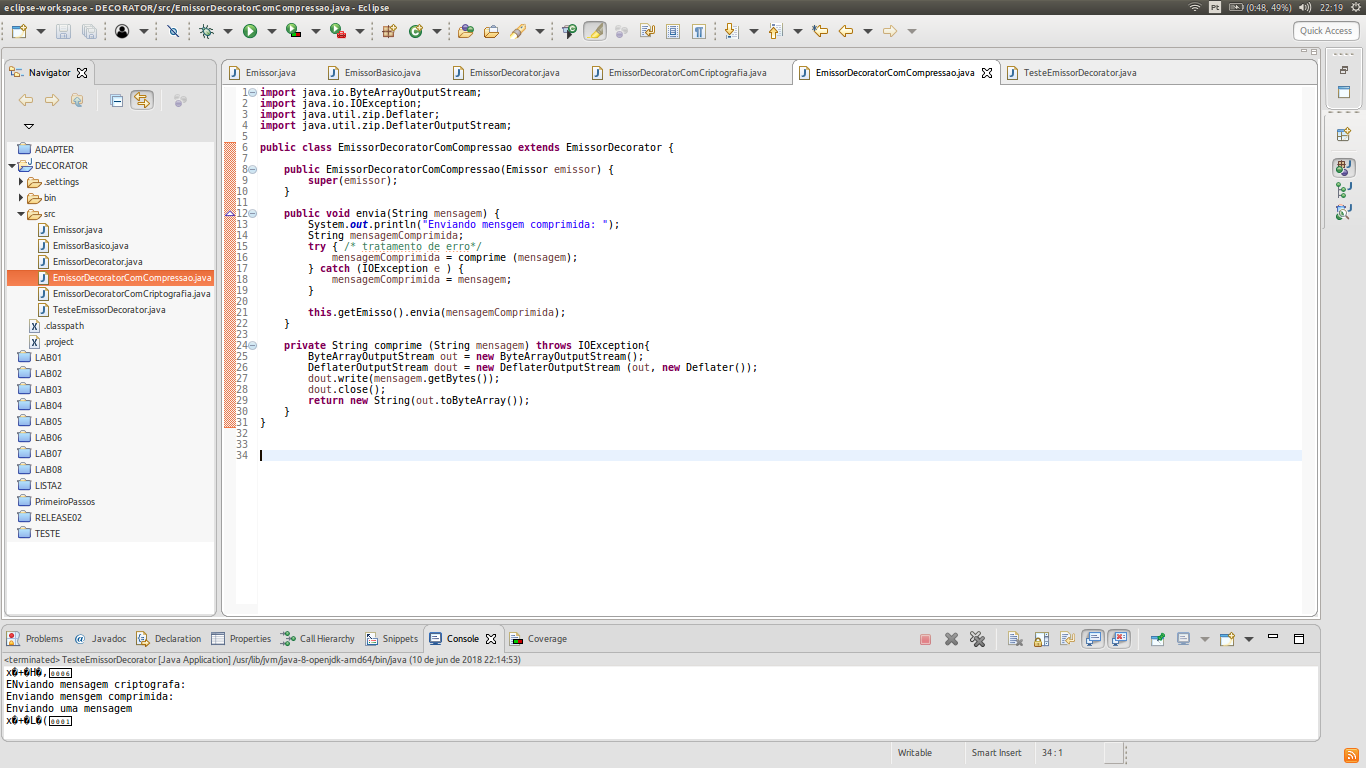


**Decorator**: Classe abstrata que mantém uma referência para um Component e será utilizado para padronizar os objetos decoradores.

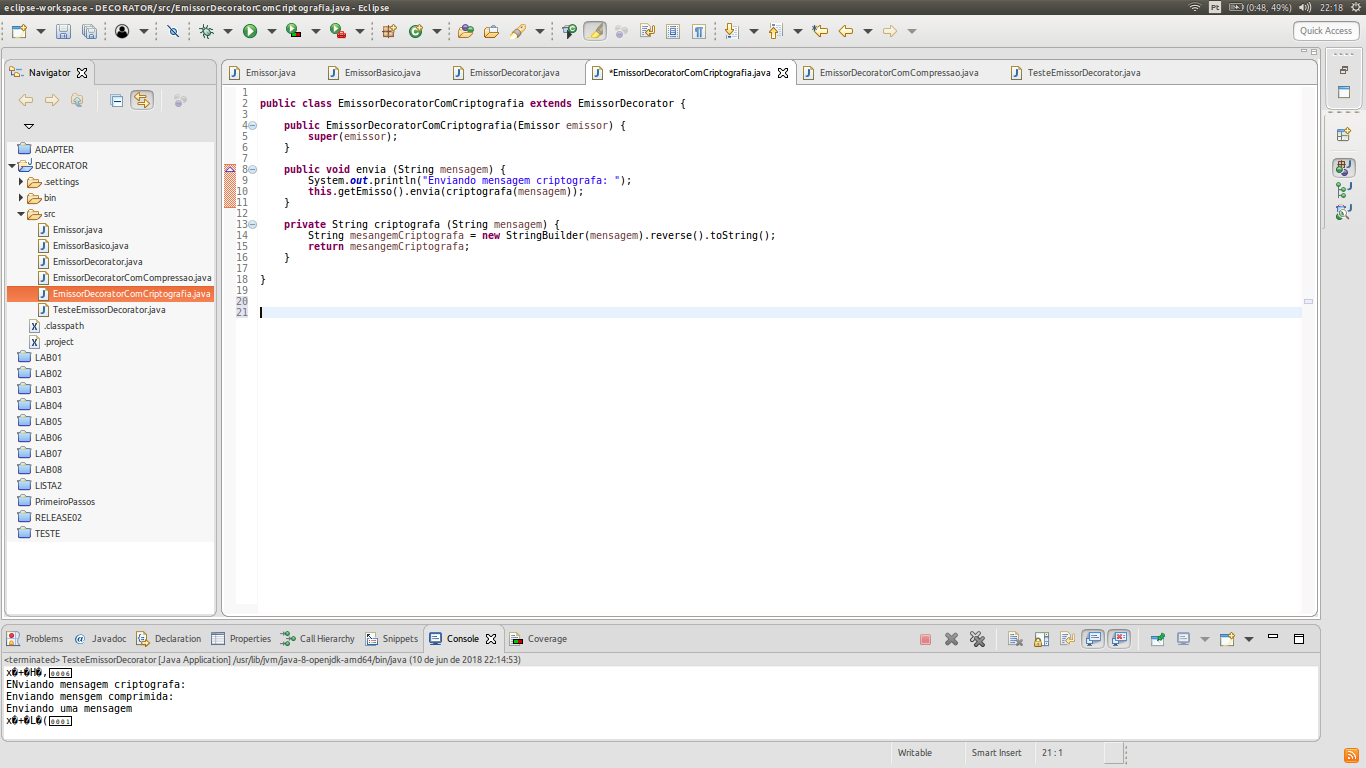


**ConcreteDecorator**: Acrescenta responsabilidades aos componente, implementação de um decorator.

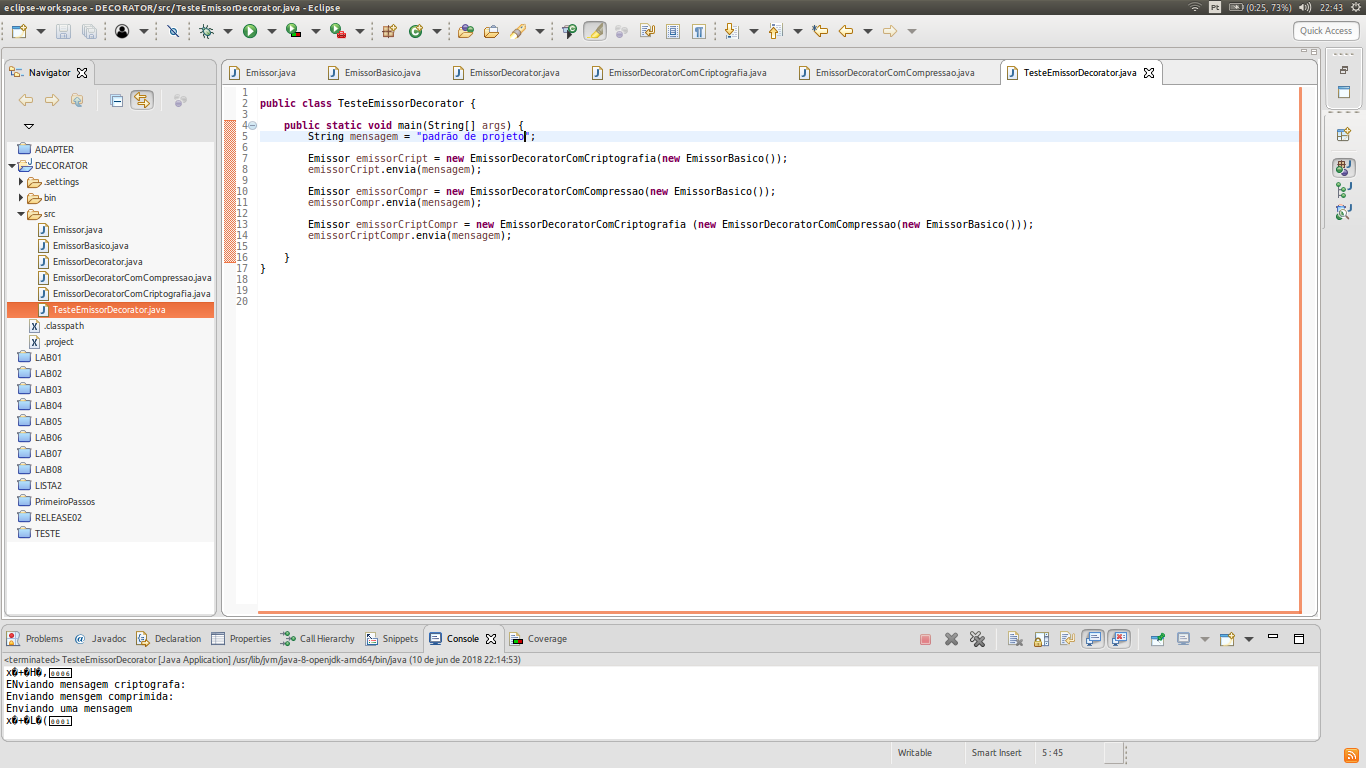
DecoratorA:



DecoratorB:



Exemplo de uma classe cliente:



→ Decorator no Java:

Na biblioteca java.io é usado o padrão de projeto estrutural decorator, que possui os decorador concreto LineNumberInpuntStream (sua principal função é contar a quantidade de linhas de um arquivo), BufferInputStream (coloca a entrada em buffer, e contém o método readLine() que lê a entrada de caracteres linha por linha), FileInputStream que está sendo decorado (ConcreteComponent) e oferece componente básico.

→ Processo

Decorator usa a herança apenas para ter referência ao tipo de objeto, quando for montar um decorador com um componente é acrescentada um novo comportamento e não herdado. Esse padrão se torna mais flexível para compor novos métodos a um objeto em tempo de execução diferente do uso de herança que ocorre em tempo de compilação.

→ Bibliografia

<<https://www.devmedia.com.br/padrao-de-projeto-decorator-em-java/26238>>acessado: 07/06/2018.

K19 TREINAMENTOS. Desgn Patterns em Java, USP.

GAMA,E.; HELM, R.; JOHNSON, R.; VLISSIDES,J. Padrões de Projeto Soluções Reutilizáveis de Software Orientado a Objetos.2000