Programação Orientada a Objetos

UniRitter
Laureate International Universities

Alguns Padrões de Projetos GoF

UniRitter
Laureate International Universities

Aplicando Padrões de Projeto

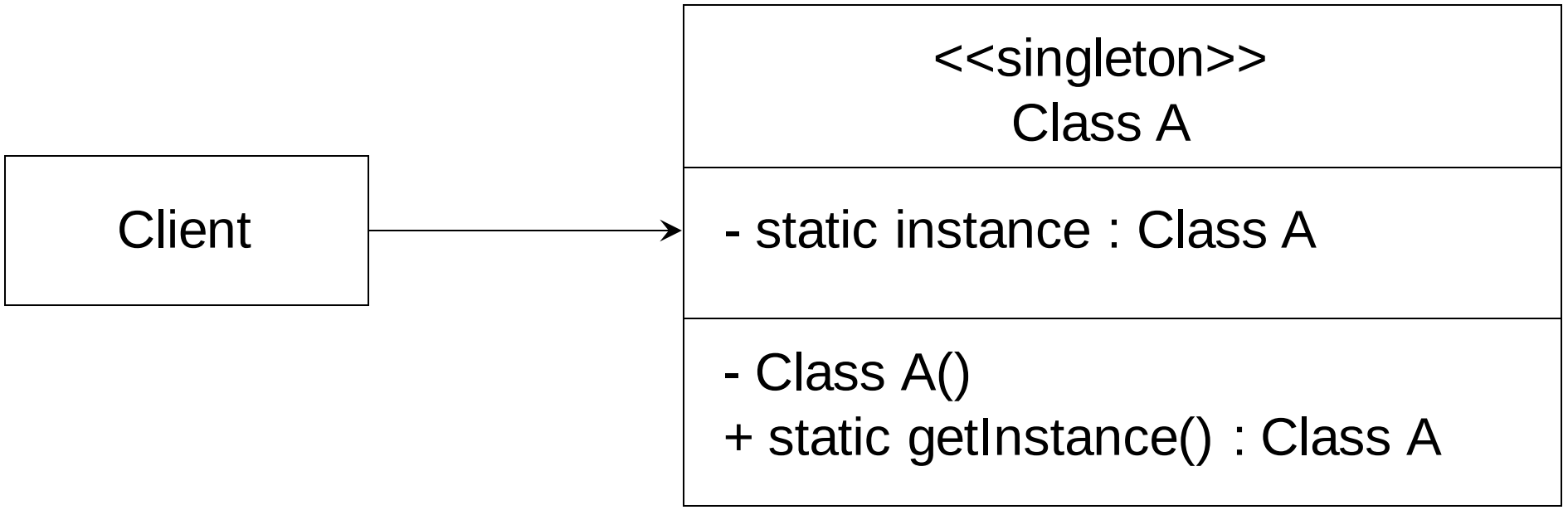
Singleton – Criacional

Intenção:

- Garantir que uma classe tenha somente uma instância e fornecer um ponto de acesso à instancia

quando:

- Deve haver exatamente uma instância da classe
- Deve ser facilmente acessível aos clientes em um ponto de acesso bem conhecido



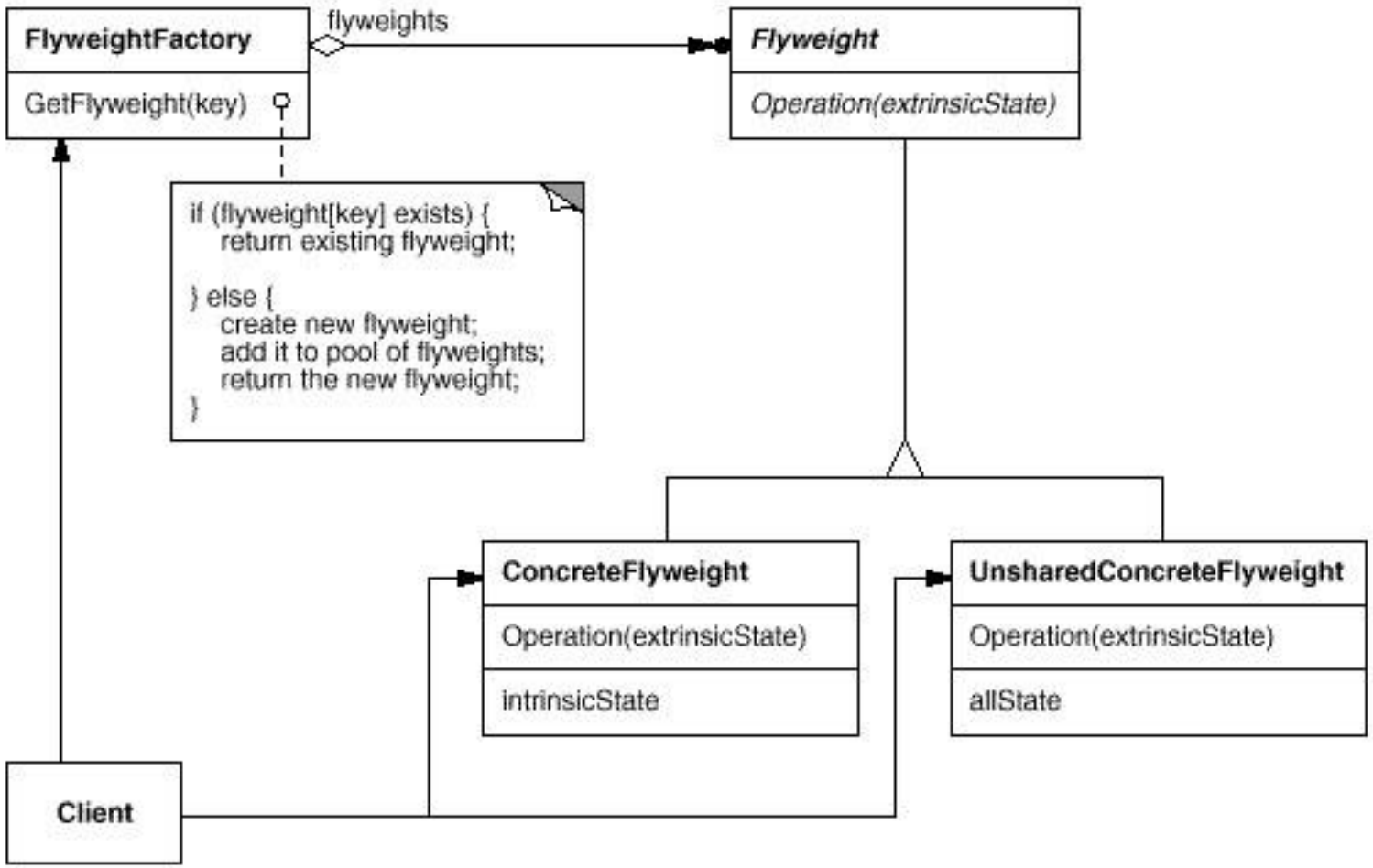
Flyweight

- Peso-mosca;
- Permite representar um grande número de objetos de forma eficiente;
- Aplicação se onde há diversas instâncias similares da mesma classe;
- Propõe reutilizar a mesma instância em todo o lugar que a necessitar;
- Ex: Status de Pedido, Categoria e grupos de produtos, etc.

Flyweight

- Utiliza uma “fábrica”;
- Normalmente a fábrica utiliza uma chave para identificar a instância;
- A fabrica pode criar os elementos de forma “ansiosa” ou “preguiçosa”.

Flyweight



Flyweight

- Uso de Singleton;
- É importante garantir a imutabilidade dos elementos “flyweight”;
- Não adicione métodos setters
- Todo atributo deve ser private e final;
- Impeça a criação de subclasses (final);
- Não retorne atributos como objetos mutáveis. Crie uma cópia e a retorne;

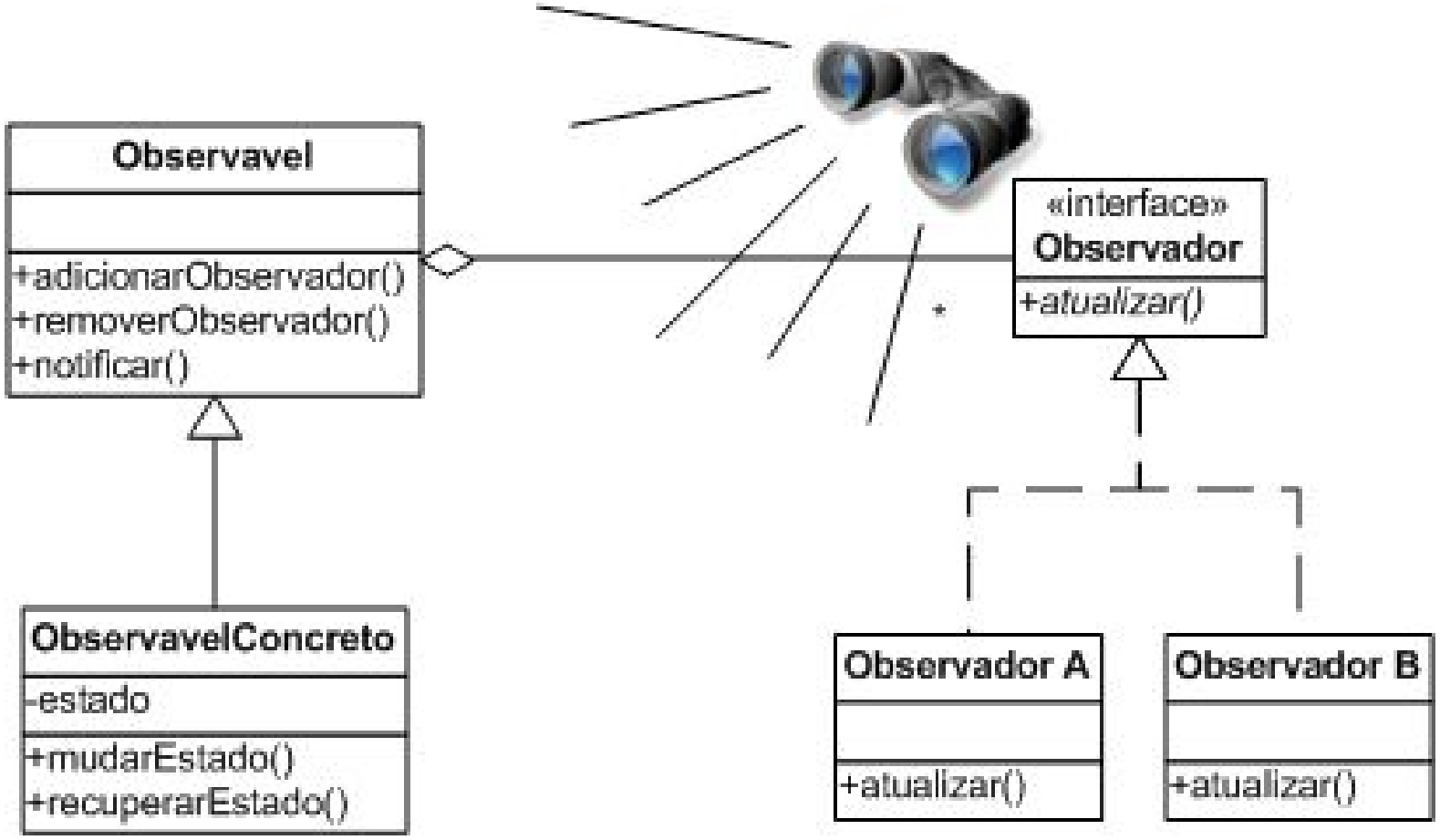
Flyweight

- Cuidado ao receber objetos para criação. Estes objetos pode ser alterados;
- Cuidado quando se usa frameworks de persistência (tipo JPA) pois normalmente eles criam novos objetos, assim como serialização;
- Anotações @Transient, PrePersistent, @PostLoad

Observer

- Utilizado quando os eventos de um objeto precisam ser observados/monitorados por outros objetos;
- Cria-se uma interface Observador, que será implementada pelas classes que que desejam receber os eventos;
- O objeto “observado” deve conhecer seus observadores e notifica-los de seus eventos (addObservador())

Observer

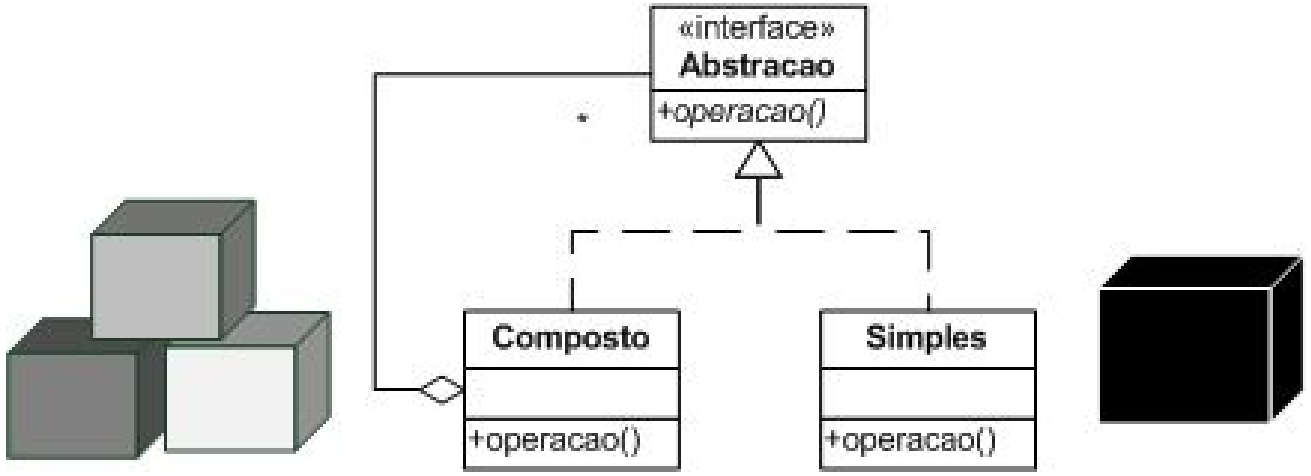


Observer

- API do Java
  - Interface Observer
  - Classe Observable

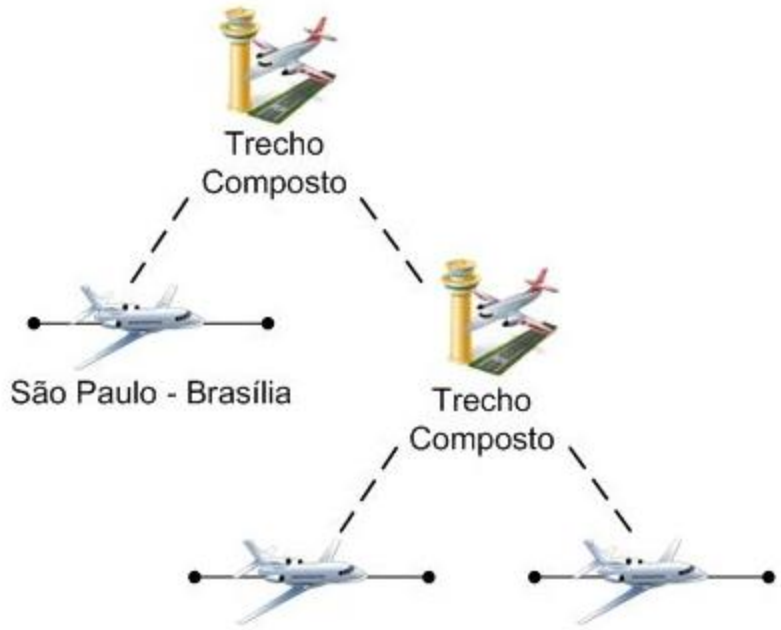
Composite

- Quando se necessita que um conjunto de objetos compartilhe da mesma abstração que um elemento do conjunto;



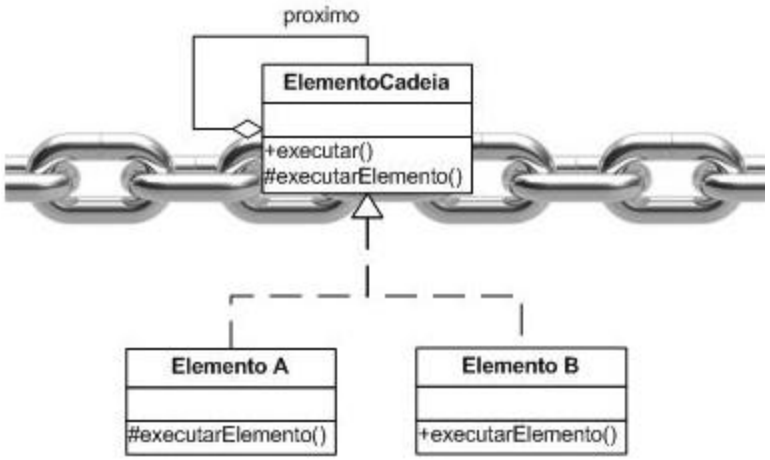
Composite

- Ex.: Trecho aéreo



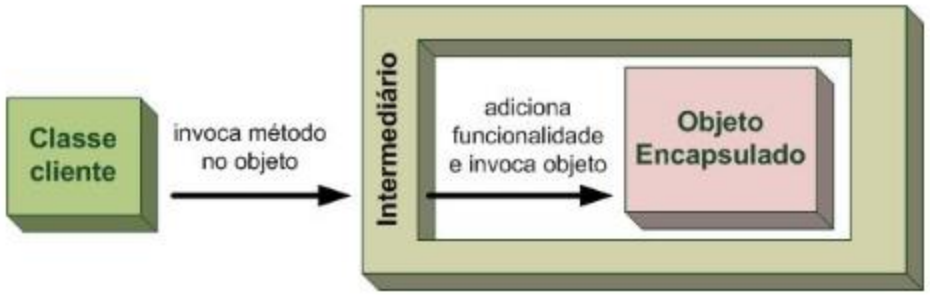
Chain of Responsibility

- Cria uma cadeia de execução de operações onde cada elemento realiza sua operação/processa a informação e a delega ao próximo elemento da sequência/fila;



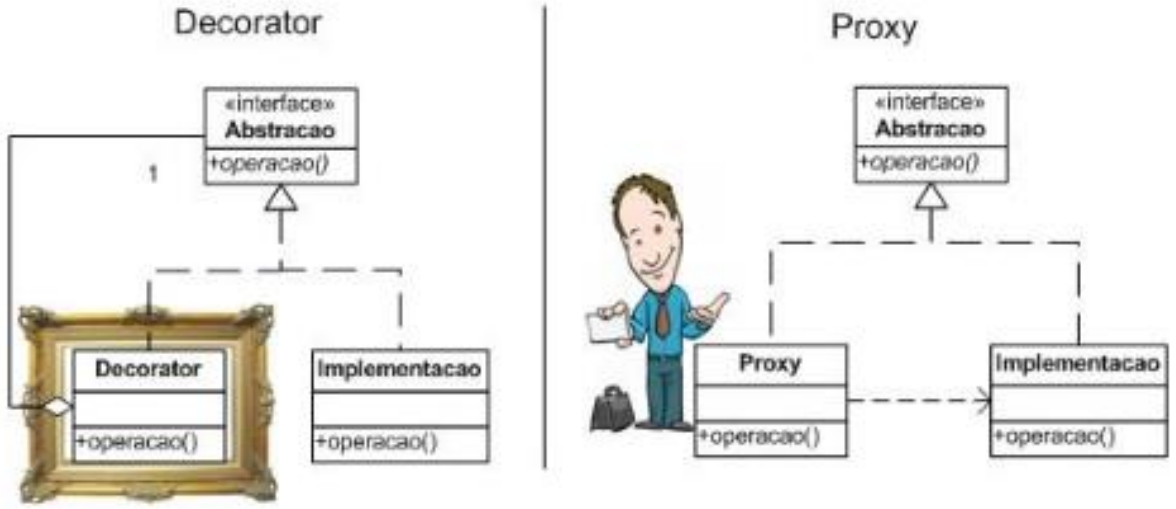
Proxy e Decorator

- Criar uma classe que envolva outra para que ela se passe por outra.



Proxy e Decorator

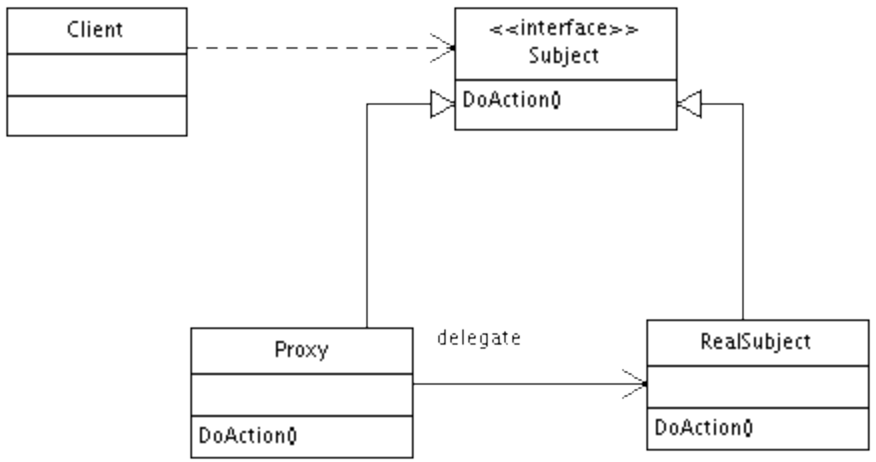
- Utilizam composição recursiva, ou seja é composto por uma classe que possui a mesma abstração que ele;

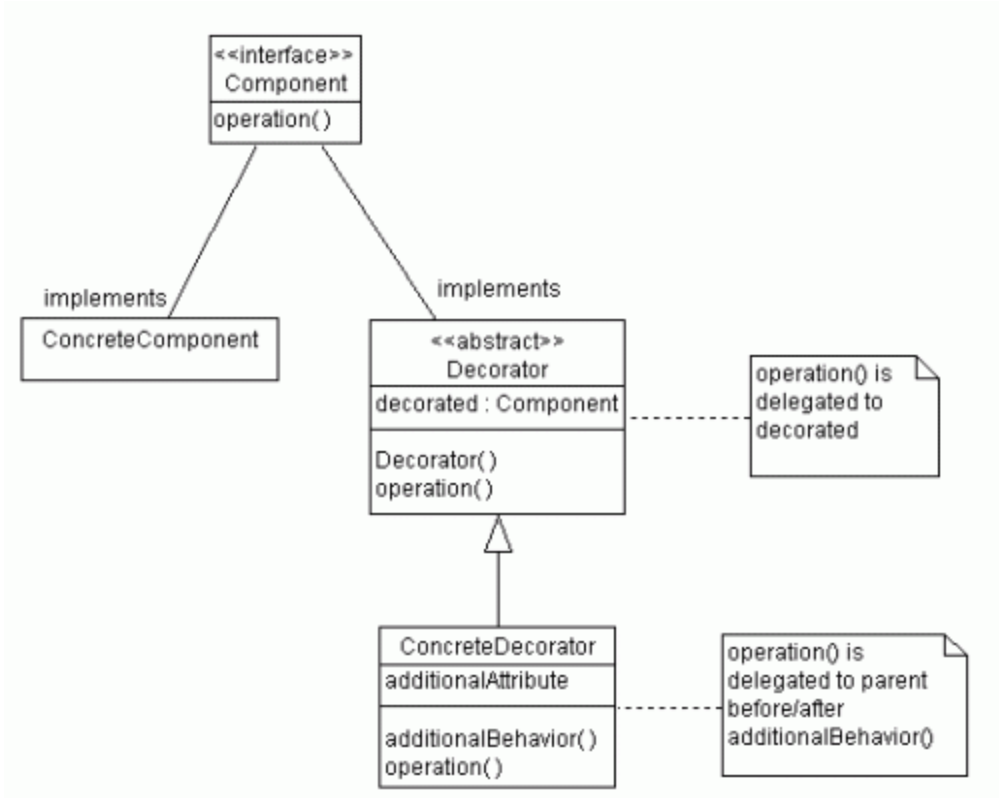


Proxy e Decorator

- O padrão Decorator adiciona mais funcionalidades ao objeto;
- Já o Proxy provem interconexão, principalmente para objetos em servidores remotos;
- Por exemplo encapsular a lógica de comunicação

Proxy e Decorator





Proxy e Decorator

- Diferenças:
- Decorator: adicionar novas funcionalidades
- Proxy: Proteger o objeto original