**O Problema**

Digamos que você tenha que fazer um sistema pra um bar especializado em coquetéis, onde existem vários tipos de coquetéis cadastrados. O coquetéis são feitos de uma bebida com vários outros adicionais. As bebidas podem ser cachaça, vodka, tequila, etc.... E os adicionais podem ser limão, leite condensado, açúcar, gelo, etc.... Então pode-se fazer diversas misturas que dependeriam do cliente. Dessa forma, teríamos que criar várias classes para prever o que um possível cliente solicitaria e isso seria inviável. E é aí que entra o decorator...

**Padrão Decorator**

* **O que é?**

**Decorator** é um padrão de projeto de software que permite adicionar um comportamento a um objeto já existente em tempo de execução, ou seja, agrega dinamicamente responsabilidades adicionais a um objeto. Ao contrário da herança que aplica funcionalidades a todos os objetos dela, o padrão decorator permite aplicar funcionalidades apenas a um objeto específico.

* **Características**
* Os decoradores têm o mesmo supertipo que os objetos que eles decoram;
* Você pode usar um ou mais decoradores para englobar um objeto;
* Os objetos podem ser decorados a qualquer momento, então podemos decorar os objetos de maneira dinâmica no tempo de execução com quantos decoradores desejarmos.

**Consequências**

**Impactos positivos**

* Mais flexibilidade do que herança
  + Adição ou remoção de responsabilidades em tempo de execução
  + Adição da mesma propriedade mais de uma vez
* Evita o excesso de funcionalidades nas classes
* Quando utiliza padrão de projeto torna o cod mais universal facilitando a compreensão e a manutenção por ouros programadores

**Impactos negativos**

* Decorator e seu componente não são idênticos
  + Comparações tornam-se mais complexas
  + Pode diminuir a eficiência do projeto.
* Resulta em um design que tem vários pequenos objetos, todos parecidos