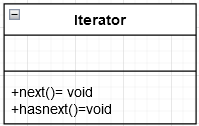
**Pesquisa Padrões de Projetos**

**Padrão Iterator**

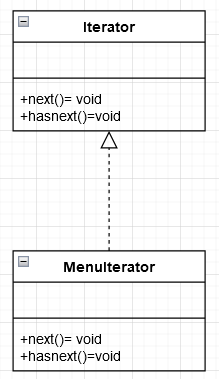
Objetivo: O Projeto Padrão Iterator visa encapsular uma iteração ele é um ótimo padrão para fornecer navegação sem expor a estrutura de um objeto. Permite a você percorrer elementos de uma coleção sem expor as representações dele (lista, pilha, árvore, etc.)

O padrão Iterator depende de uma interface chamada Iterator



O método hasnext() verifica se existe mais elementos para serem iterados, já o next() retornara o próximo objeto na iteração.

Depois disso, já temos a interface que pode implementar iterações para qualquer coleção de objetos: matrizes, listas, tabelas de hash, etc. Para cada conjunto de objetos que você deseja encapsular a iteração cria uma implementação para a interface Iterator definido acima. Por exemplo, para encapsular uma iteração de um menu teria a classe abaixo:



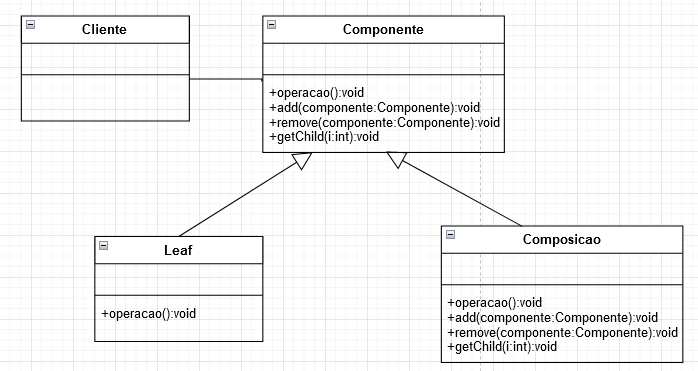
O padrão Iterator permite acessar os elementos de um array sem saber como eles estão sendo representados, por isso torna-se irrelevante se a coleção de objetos é um ArrayList, HashTable ou qualquer outra coisa. Além disso, o padrão Iterator assume a responsabilidade de acessar os elementos sequencialmente e transfere essa tarefa para o objeto Iterator.

**Padrão Composite**

O **Composite** é um padrão de projeto estrutural que permite que você componha objetos em estruturas de árvores e então trabalhe com essas estruturas como se elas fossem objetos individuais. Usar o padrão Composite faz sentido apenas quando o modelo central de sua aplicação pode ser representado como uma árvore. O padrão Composite descreve como usar a composição de forma que os clientes não precisem distinguir objetos simples de estruturas complexas.

Problema: cliente precisa tratar de maneira uniforme objetos individuais e composições desses objetos

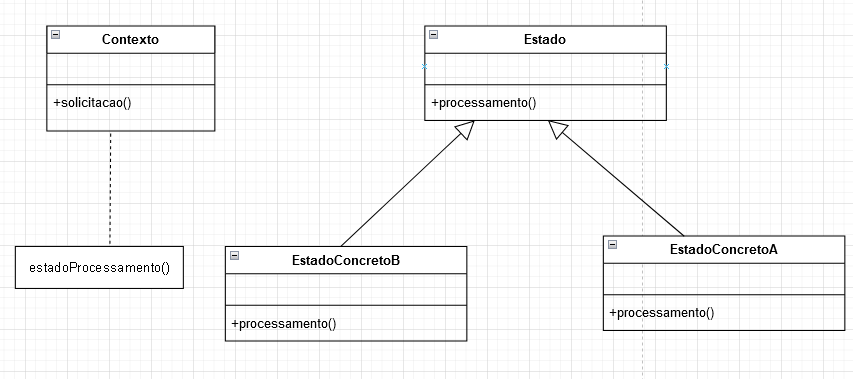
Solução: Tratar os grupos e indivíduos diferentes através de uma única interface



**Padrão State**

O padrão State permite que um objeto altere o seu comportamento quando o seu estado interno muda. O objeto parecerá ter mudado de classe.

O padrão encapsula os estados em classes separadas e delega as tarefas para o objeto que representa o estado atual, nós sabemos que os comportamentos mudam juntamente com o estado interno.



O contexto é a classe que pode ter vários estados internos diferentes, interface estado define uma interface comum para todos os estados concretos, estado concretos (podemos ter vários estados concretos) lidam com as solicitações provenientes do contexto

**Padrão Proxy**

No padrão Proxy, um procurador funciona como um substituto ou um marcador para outro objeto. Um proxy (procurador) pode ser usado de várias formas. Ele pode atuar como um  
representante local para um objeto situado num espaço de endereço remoto. Pode  
representar um grande objeto que deveria ser carregado por demanda. Pode proteger o  
acesso a um objeto sensível. Proxies fornecem um nível de referência indireta a  
propriedades específicas de objetos. Daí eles poderem restringir, aumentar ou alterar  
estas propriedades. O Proxy define um representante ou “procurador” para outro objeto e não muda a sua interface

Estrutura: Ao definir uma interface Subject, a presença do objeto Proxy no lugar do RealSubject é transparente para o cliente.

