PADRÕES DE PROJETO

DESIGN PATTERNS

GoF

Gang of Four

- Erich Gamma,
- Richard Helm,
- Ralph Johnson,
- John Vlissides

MARCO AURÉLIO REGIS

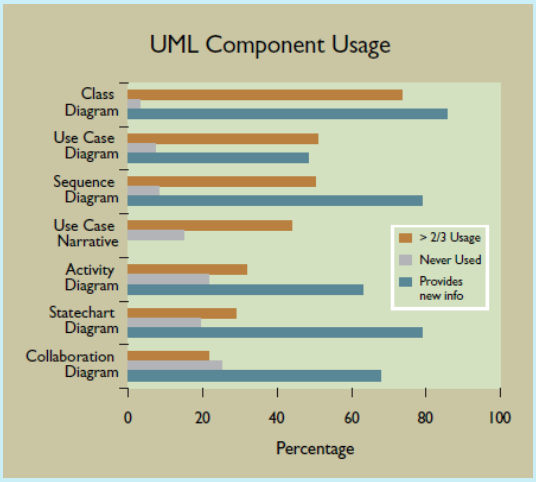
PADRÕES DE PROJETO

DESIGN PATTERNS

- **UML - *Unified Modeling Language***

MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS



[Dobing B., Parsons J., *Communications of the ACM*, Canada, 2006]

PADRÕES DE PROJETO
DESIGN PATTERNS

- **DIAGRAMA DE CLASSES**
  - Mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos. Os digramas de classes abrangem a visão estática do projeto de um sistema; um diagrama que mostra a coleção de elementos declarativos (estáticos).

[BOOCH G., et al. 2000]

PADRÕES DE PROJETO
DESIGN PATTERNS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **ASSOCIAÇÃO ENTRE CLASSES**
  - Um relacionamento estrutural que descreve um conjunto de vínculos, em que o vínculo é uma conexão entre objetos.
- Associações podem ser:
  - Unárias
  - Binárias
  - Múltiplas

PADRÕES DE PROJETO
DESIGN PATTERNS

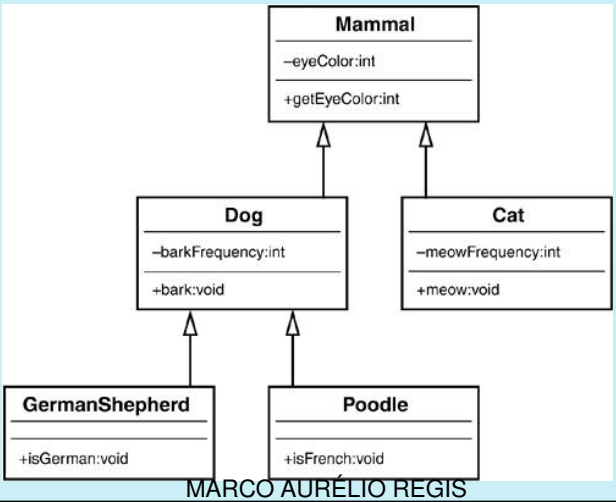
- **CARDINALIDADE ou MULTIPLICIDADE**
  - Determina quantos objetos no sistema são possíveis em cada vértice da associação.
- **NAVEGAÇÃO**
  - Se é possível para cada objeto acessar outro objeto da mesma associação.

PADRÕES DE PROJETO
DESIGN PATTERNS

- **HERANÇA**
  - É um relacionamento do tipo generalização / especialização, onde uma classe pode ser derivada (subclasse) de outra mais geral (superclasse), absorvendo todas as características fundamentais e adicionando outras novas características, de tal modo a torná-la mais especializada.

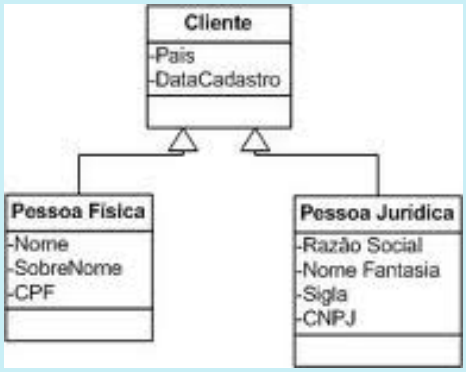
PADRÕES DE PROJETO
DESIGN PATTERNS

- **HERANÇA**



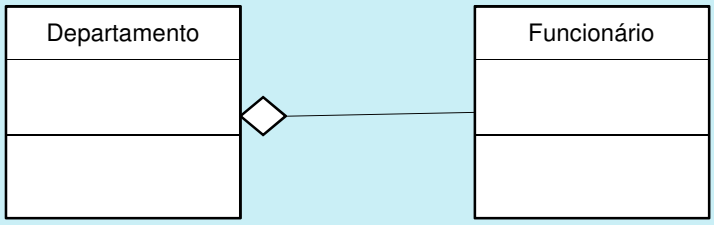
PADRÕES DE PROJETO
DESIGN PATTERNS

- **HERANÇA**
  - Identificando especializações:
    - Pessoa Física **é um** Cliente
    - Pessoa Jurídica **é um** Cliente



PADRÕES DE PROJETO
DESIGN PATTERNS

- **AGREGAÇÃO**
  - Tipo de relacionamento com características todo-parte.
  - Certo grau de independência entre as classes.

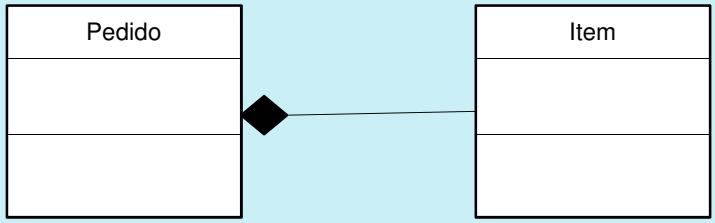


PADRÕES DE PROJETO
DESIGN PATTERNS

```
class Departamento{
   //Array de referências
   private Empregado empregados[ ] =
         new Empregado[10];

   public void addEmpregado(Empregado emp){
   }
}

class Empregado{
}
```

PADRÕES DE PROJETO
DESIGN PATTERNS

- **COMPOSIÇÃO**
  - Tipo de relacionamento com características todo-parte.
  - Alto grau de coesão entre o todo e as partes.



MARCO AURÉLIO REGIS

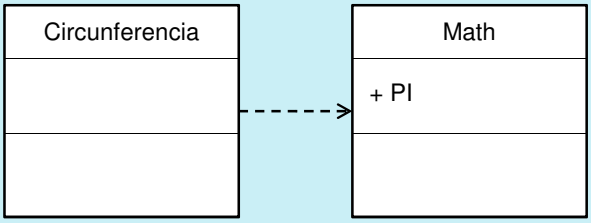
PADRÕES DE PROJETO
DESIGN PATTERNS

```
class Pedido{
    //Array de referências
    private Item itens[ ] = new Item[10];

    public Pedido() {
        for(int i=0; i < itens.length; i++){
            itens[ i ] = new Item();
        }
    }
}

class Item{
}
```

MARCO AURÉLIO REGIS

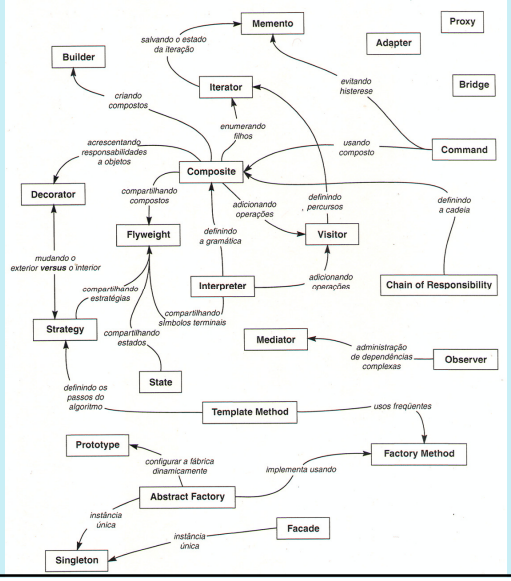
PADRÕES DE PROJETO
DESIGN PATTERNS

- **DEPENDÊNCIA**
  - Relacionamento entre duas classes: cliente (*client*) e fornecedor (*supplier*).
  - Classe cliente usa / depende da classe fornecedor.
  - A alteração da classe independente poderá afetar a semântica da classe dependente.



MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS



PADRÕES DE PROJETO

DESIGN PATTERNS

- Descrevem soluções para problemas recorrentes no desenvolvimento de sistemas ***orientado a objetos***.
  - possui um nome
  - define o problema
  - define a solução
  - quando aplicar esta solução
  - consequências

PADRÕES DE PROJETO

DESIGN PATTERNS

- **GoF (Gang of Four) (23)**
  - Padrões de criação (“criacionais”)
  - Padrões estruturais
  - Padrões comportamentais

PADRÕES DE PROJETO

DESIGN PATTERNS

- **PADRÕES DE CRIAÇÃO (“criacionais”) - 5**
  - Aplicam-se em situações que envolvem a criação de objetos.
  - Ajudam a fazer um sistema independente de como seus objetos são criados, compostos e representados (*Erich Gamma, et al.*).

“A word of warning and encouragement: Don't worry if you don’t understand this book completely on the first reading. We didn’t understand it all on the first writing! Remember that this isn't a book to read once and put on a shelf.”

MARCO AURÉLIO REGIS

PADRÕES DE PROJETO

DESIGN PATTERNS

- **PADRÕES ESTRUTURAIS - 7**
  - De que maneira as classes e objetos são compostos para a formação de grandes estruturas (*Erich Gamma, et al.*).

MARCO AURÉLIO REGIS

PADRÕES DE PROJETO

DESIGN PATTERNS

- **PADRÕES COMPORTAMENTAIS - 11**
  - Preocupam-se com algoritmos e atribuição de responsabilidades entre objetos. Descrevem, também, um padrão de comunicação entre classes ou objetos (*Erich Gamma, et al.*).

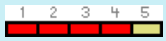
PADRÕES DE PROJETO

DESIGN PATTERNS

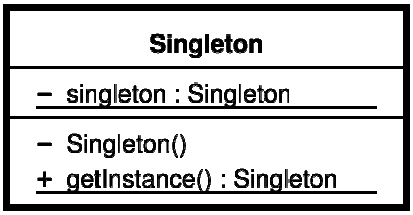
- **PADRÕES DE CRIAÇÃO (“criacionais”) - 5**
  - Eles disponibilizam uma maneira para criar objetos ocultando os detalhes da sua criação, ao invés de utilizar o operador ***new*** diretamente. Isto dá mais flexibilidade na hora de decidir quais objetos precisam ser criados para um determinado caso.

PADRÕES DE PROJETO
DESIGN PATTERNS

- **SINGLETON**

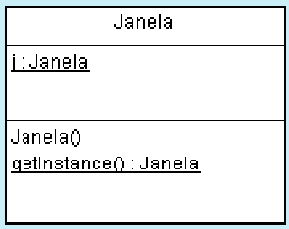


  - **Intenção:** garantir que uma determinada classe tenha uma, e somente uma instância, mantendo um ponto global de acesso para a mesma.



PADRÕES DE PROJETO
DESIGN PATTERNS

- **SINGLETON - *exemplo***



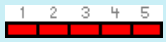
PADRÕES DE PROJETO
DESIGN PATTERNS

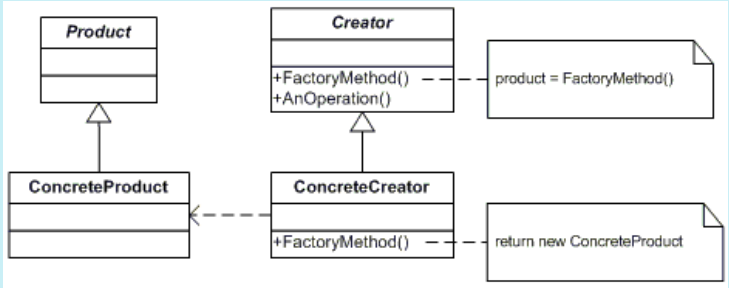
- **FACTORY METHOD**

  - **Intenção:** definir uma interface para criar um objeto, mas deixar as subclasses decidirem que classe instanciar.
  - Cria uma instância de várias classes derivadas.

PADRÕES DE PROJETO
DESIGN PATTERNS

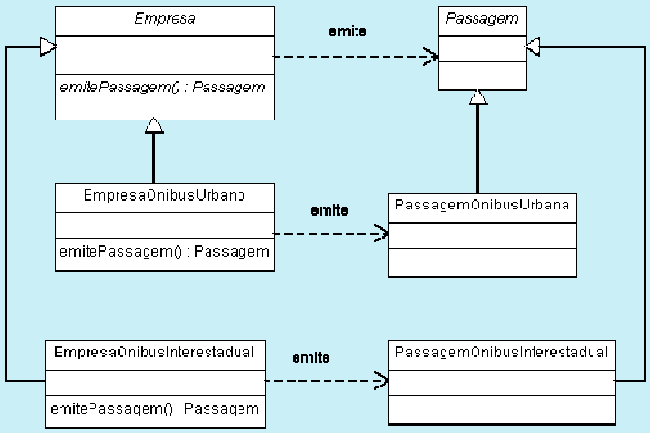
- **FACTORY METHOD**





PADRÕES DE PROJETO
DESIGN PATTERNS

- **FACTORY METHOD - *exemplo***



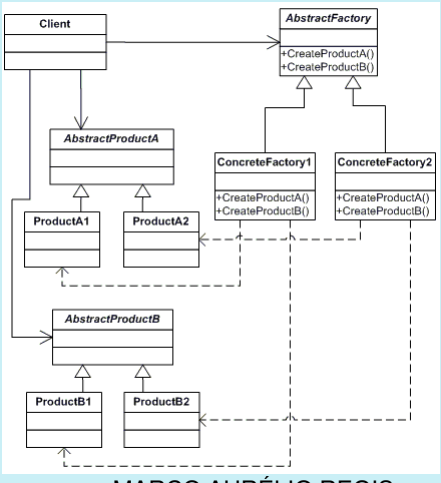
PADRÕES DE PROJETO
DESIGN PATTERNS

- **ABSTRACT FACTORY**
  - **Intenção:** fornecer uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
  - Cria uma instância de várias famílias de classes.

PADRÕES DE PROJETO

DESIGN PATTERNS

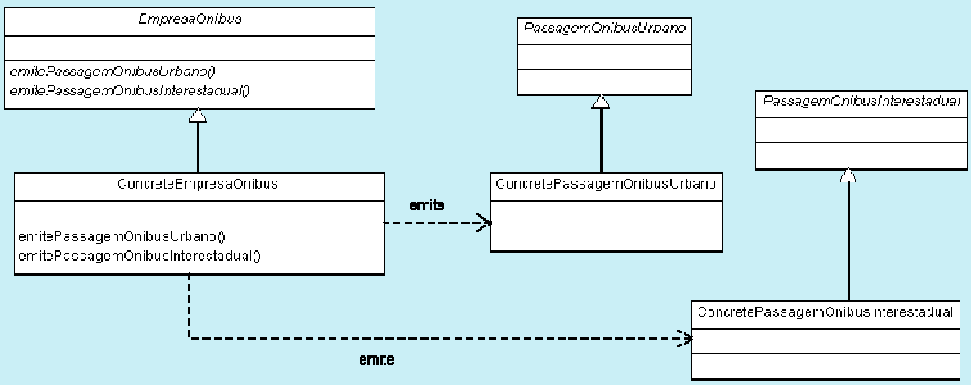
- **ABSTRACT FACTORY**



PADRÕES DE PROJETO

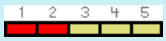
DESIGN PATTERNS

- **ABSTRACT FACTORY - *exemplo***



PADRÕES DE PROJETO
DESIGN PATTERNS

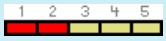
- **BUILDER**

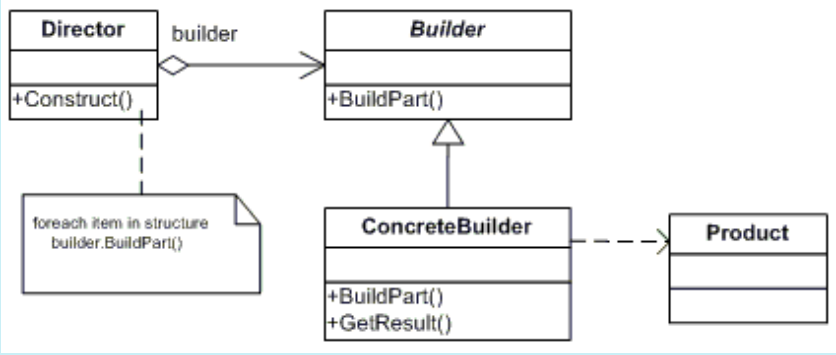


  - **Intenção:** separar a construção de um objeto complexo da sua representação de modo que o mesmo processo de construção possa criar diferentes representações.
  - Separa a construção do objeto de sua representação.

PADRÕES DE PROJETO
DESIGN PATTERNS

- **BUILDER**

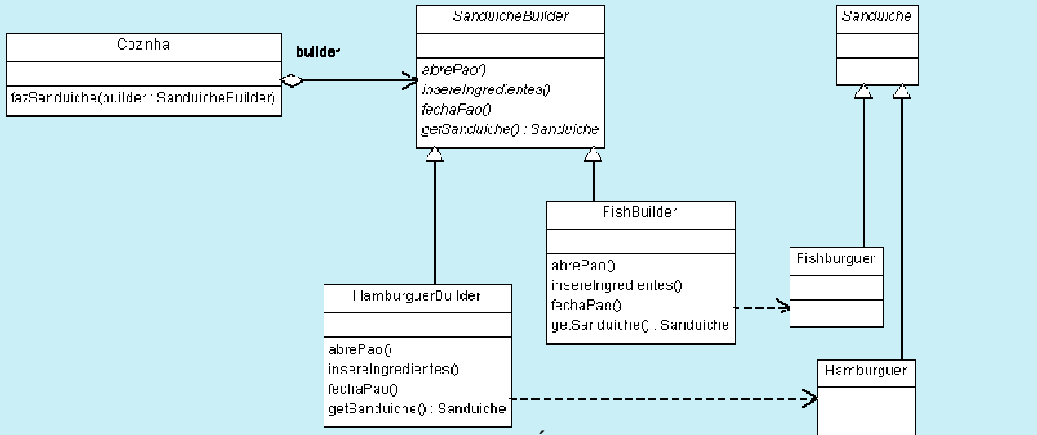




MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **BUILDER - *exemplo***

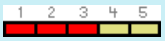


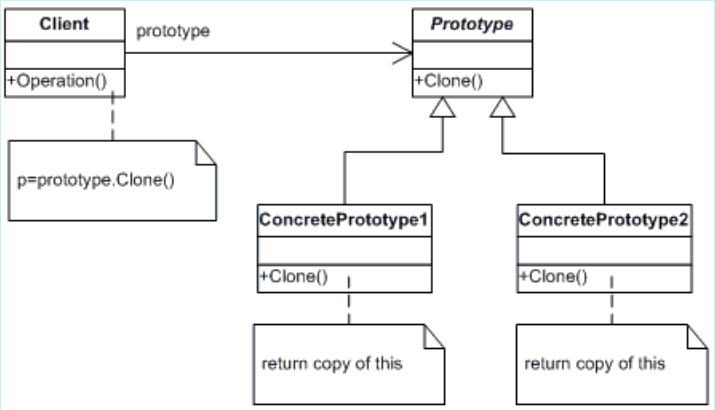
PADRÕES DE PROJETO
DESIGN PATTERNS

- **PROTOTYPE**
  - **Intenção:** especificar os tipos de objetos a serem criados usando uma instância protótipo e criar novos objetos pela cópia deste protótipo.
  - Uma instância inicializada a ser copiada ou clonada.

PADRÕES DE PROJETO
DESIGN PATTERNS

- **PROTOTYPE**

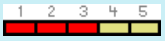


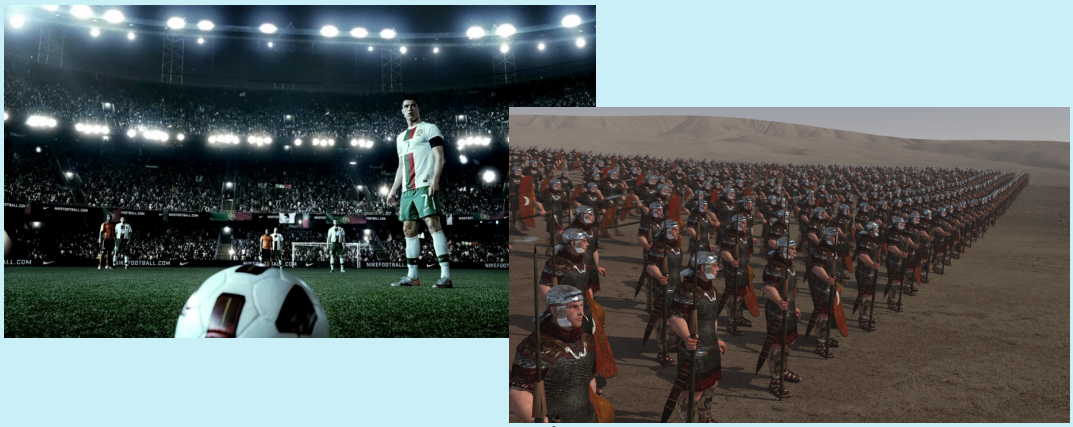


MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **PROTOTYPE - *exemplo***

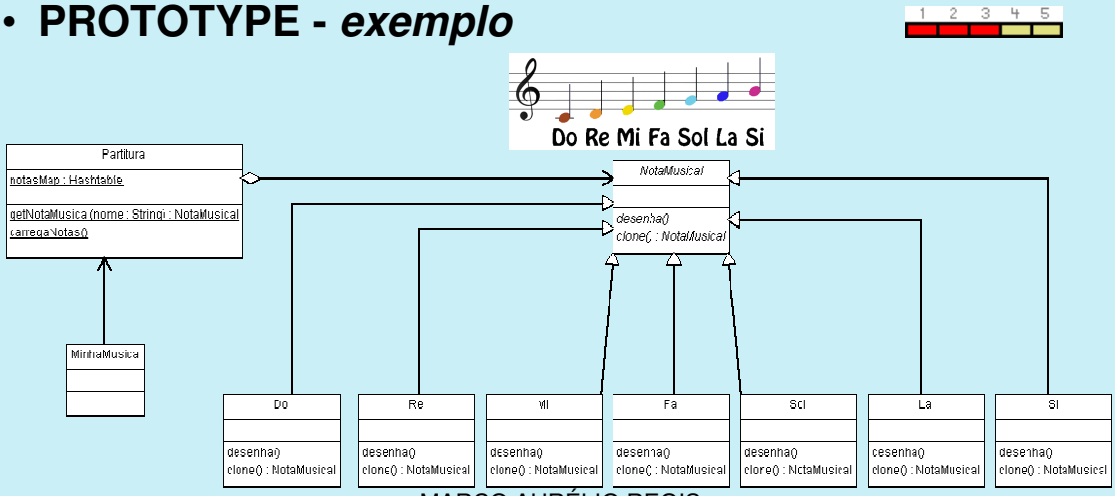




MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **PROTOTYPE - *exemplo***

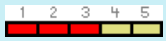


PADRÕES DE PROJETO
DESIGN PATTERNS

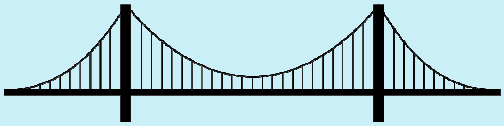
- **PADRÕES ESTRUTURAIS - 7**
  - Eles se preocupam com a composição das classes e seus objetos. O conceito de herança é largamente utilizado para compor interfaces e definir maneiras para compor objetos e obter novas funcionalidades.

PADRÕES DE PROJETO
DESIGN PATTERNS

- **BRIDGE**



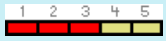
  - **Intenção:** desacoplar uma abstração da sua implementação, de modo que as duas possam variar independentemente.
  - Separa a interface do objeto de sua implementação.

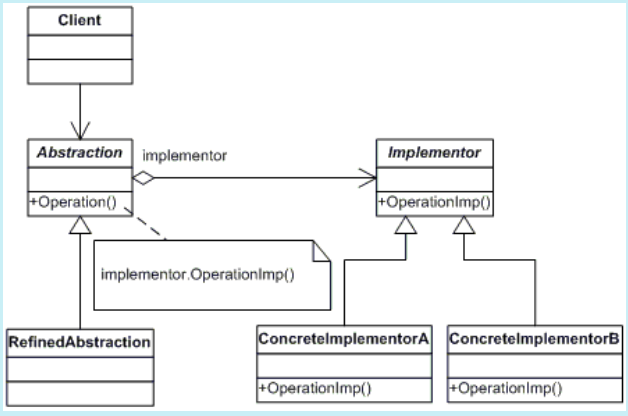


MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **BRIDGE**

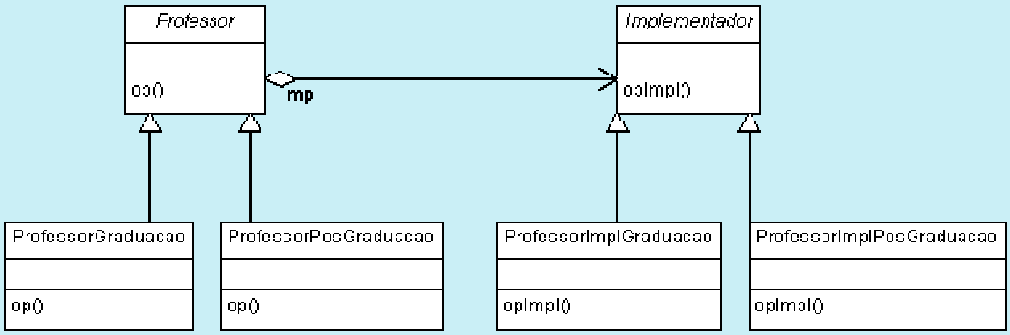




MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

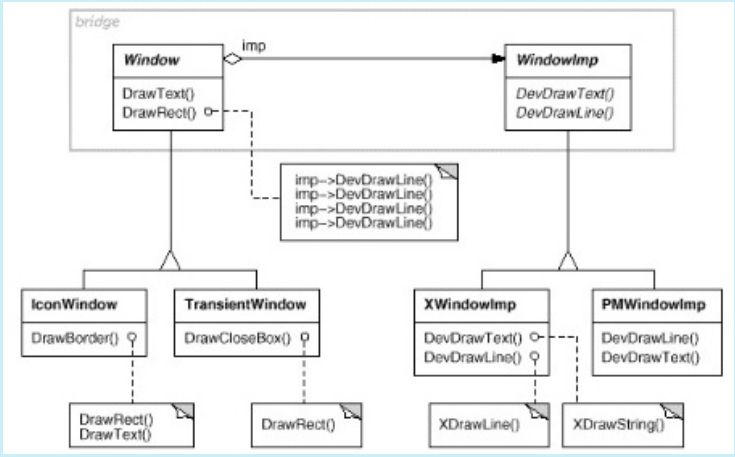
- **BRIDGE - *exemplo***



MARCO AURÉLIO REGIS

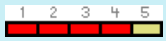
PADRÕES DE PROJETO
DESIGN PATTERNS

- **BRIDGE – *exemplo 2***



PADRÕES DE PROJETO
DESIGN PATTERNS

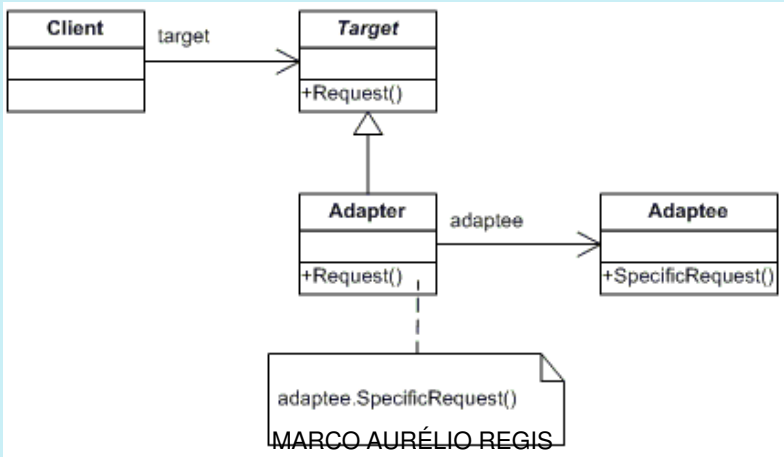
- **ADAPTER**



  - **Intenção:** converter a interface de uma classe em outra interface, esperada pelos clientes. O ***Adapter*** permite que classes com interfaces incompatíveis trabalhem em conjunto o que, de outra forma, seria impossível.
  - Equiparar interfaces de diferentes classes.

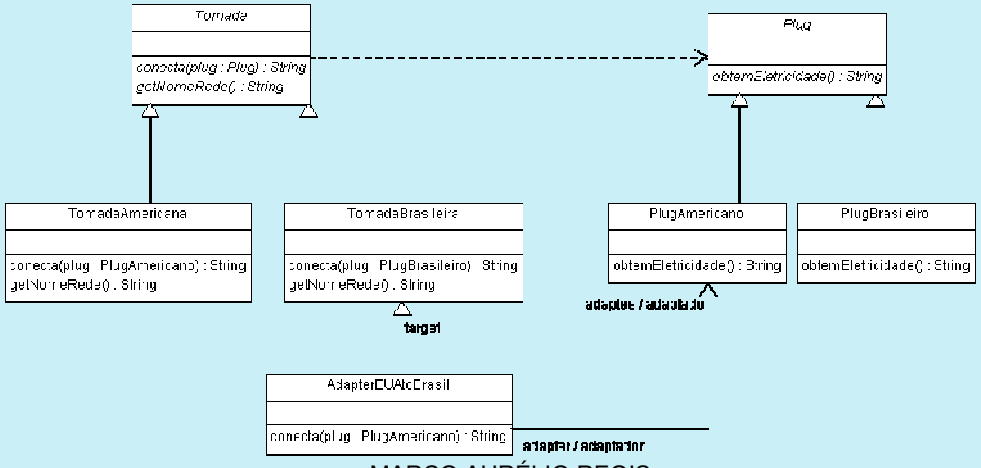
PADRÕES DE PROJETO
DESIGN PATTERNS

- **ADAPTER**



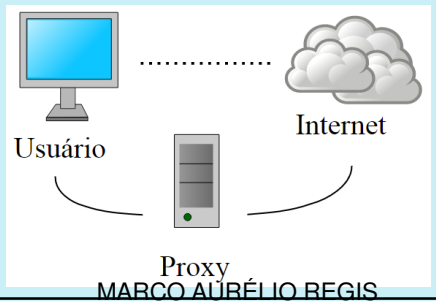
PADRÕES DE PROJETO
DESIGN PATTERNS

- **ADAPTER - *exemplo***



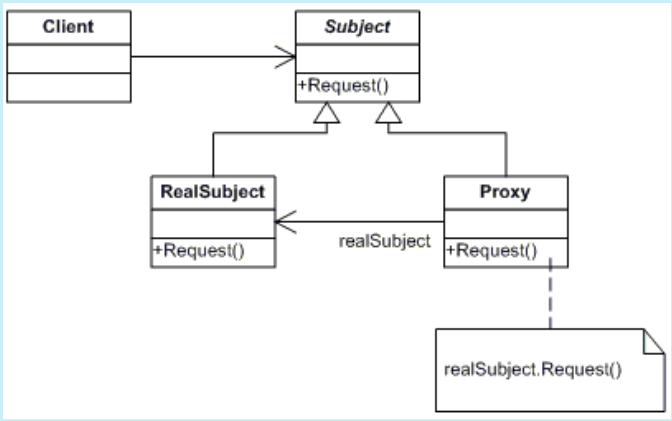
PADRÕES DE PROJETO
DESIGN PATTERNS

- **PROXY**
  - **Intenção:** fornece um substituto (*surrogate*) ou marcador da localização de outro objeto para controlar o acesso ao mesmo.
  - Um objeto representando um outro objeto.



PADRÕES DE PROJETO
DESIGN PATTERNS

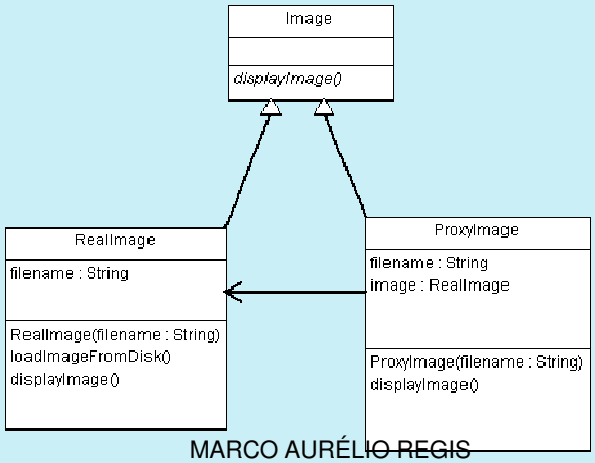
- **PROXY**



MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

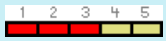
- **PROXY - *exemplo***



MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

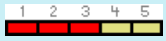
- **DECORATOR**

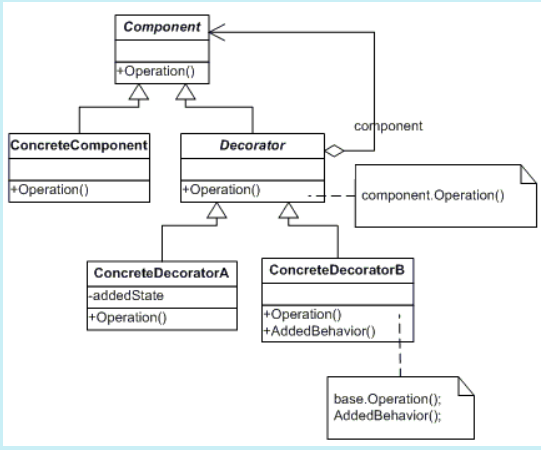


  - **Intenção:** agregar dinamicamente responsabilidades adicionais a um objeto.

PADRÕES DE PROJETO
DESIGN PATTERNS

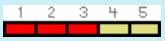
- **DECORATOR**

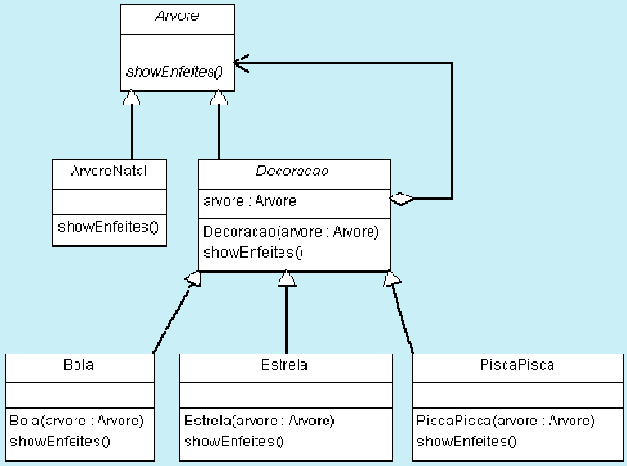




PADRÕES DE PROJETO
DESIGN PATTERNS

- **DECORATOR - *exemplo***





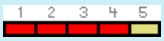
PADRÕES DE PROJETO
DESIGN PATTERNS

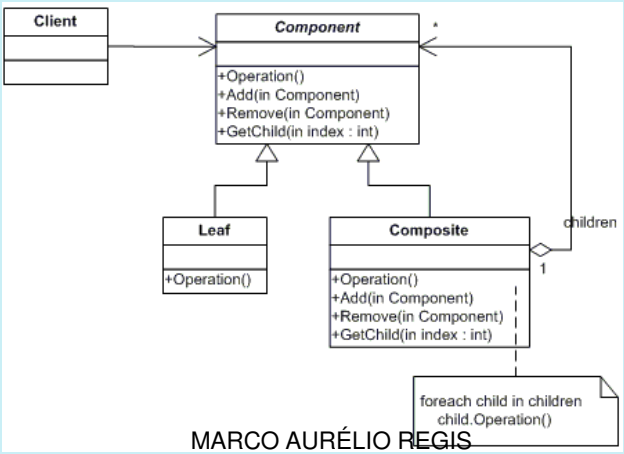
- **COMPOSITE**

  - **Intenção:** compor objetos em estruturas de árvore para representarem hierarquias todo-parte.

PADRÕES DE PROJETO
DESIGN PATTERNS

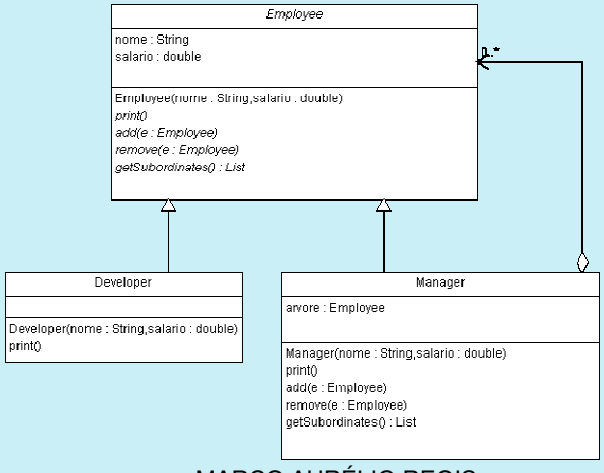
- **COMPOSITE**





PADRÕES DE PROJETO
DESIGN PATTERNS

- **COMPOSITE - *exemplo***



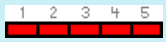
PADRÕES DE PROJETO
DESIGN PATTERNS

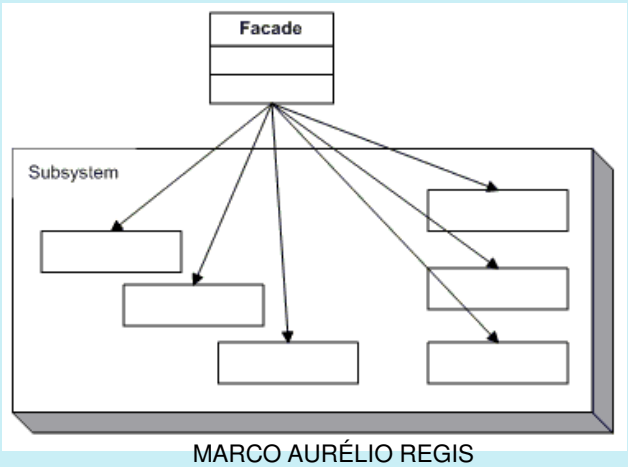
- **FAÇADE (FACADE)**

  - **Intenção:** fornecer uma interface unificada para um conjunto de interfaces em um subsistema. **Façade** define uma interface de nível mais alto que torna o subsistema mais fácil de ser usado.
  - Uma única classe representa um subsistema inteiro.

PADRÕES DE PROJETO
DESIGN PATTERNS

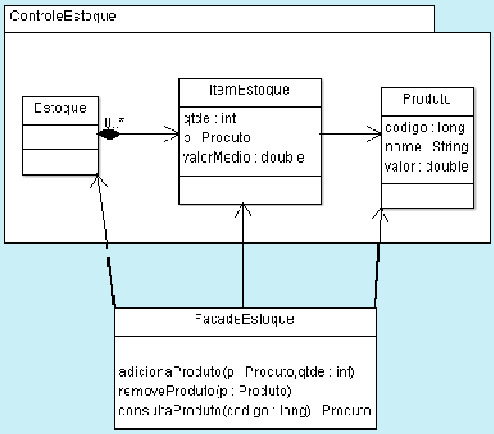
- **FAÇADE**





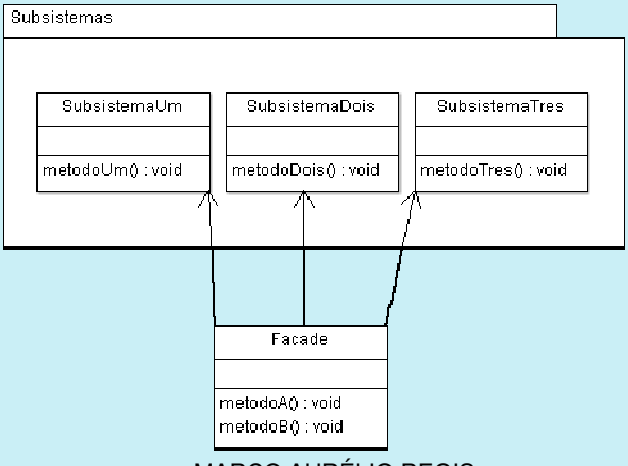
PADRÕES DE PROJETO
DESIGN PATTERNS

- **FAÇADE - *exemplo***



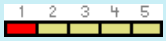
PADRÕES DE PROJETO
DESIGN PATTERNS

- **FAÇADE – *exemplo 2***



PADRÕES DE PROJETO
DESIGN PATTERNS

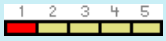
- **FLYWEIGHT**

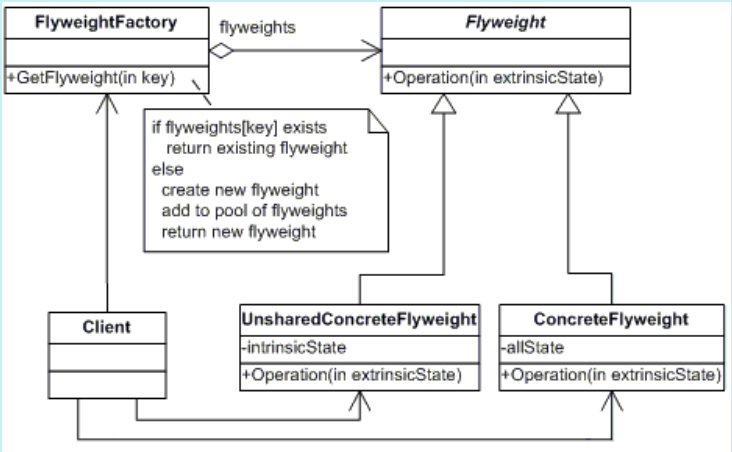


  - **Intenção:** usar compartilhamento para suportar eficientemente grandes quantidades de objetos de granularidade fina.

PADRÕES DE PROJETO
DESIGN PATTERNS

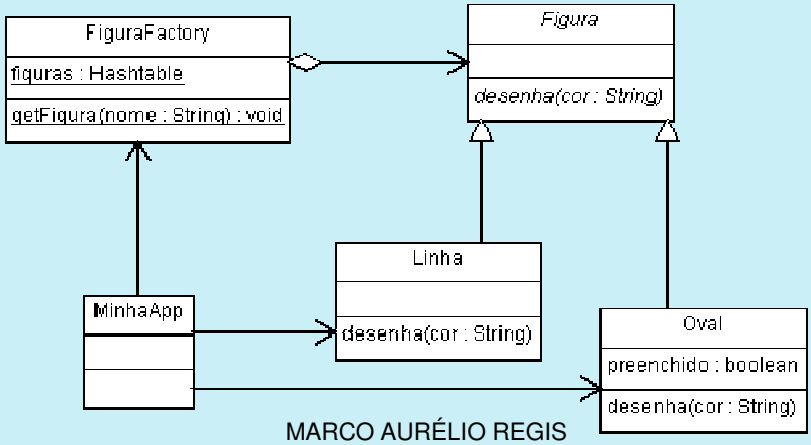
- **FLYWEIGHT**





PADRÕES DE PROJETO
DESIGN PATTERNS

- **FLYWEIGHT - *exemplo***

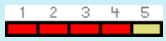


PADRÕES DE PROJETO
DESIGN PATTERNS

- **PADRÕES COMPORTAMENTAL - 11**
  - Eles são especificamente relacionados com a comunicação entre objetos.

PADRÕES DE PROJETO
DESIGN PATTERNS

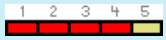
- **COMMAND**

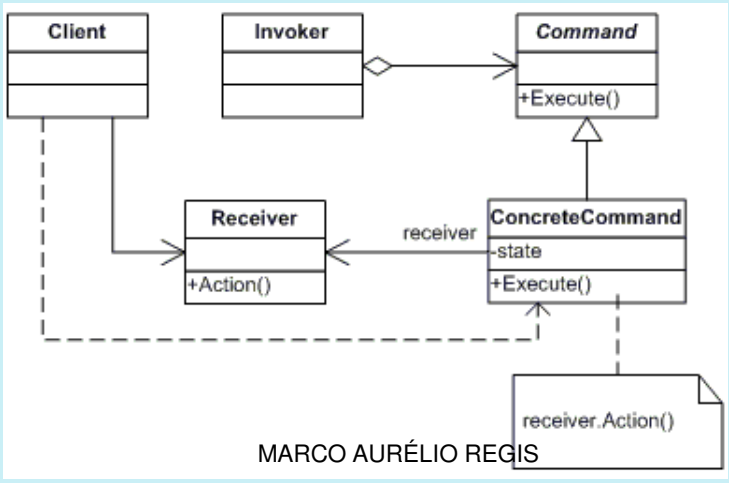


  - **Intenção:** encapsular uma solicitação como um objeto, desta forma permitindo parametrizar clientes com diferente solicitações, enfileirar ou fazer o registro (log) de solicitações e suportar operações que podem ser desfeitas.
  - Encapsular comandos como um objeto.

PADRÕES DE PROJETO
DESIGN PATTERNS

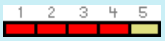
- **COMMAND**

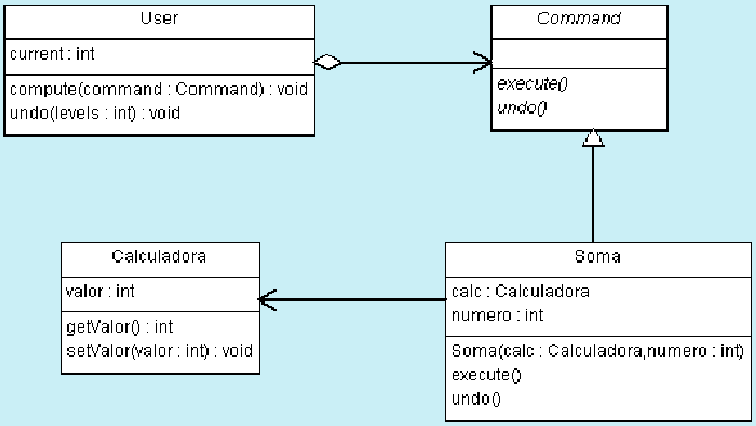




PADRÕES DE PROJETO
DESIGN PATTERNS

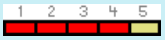
- **COMMAND - *exemplo***





PADRÕES DE PROJETO
DESIGN PATTERNS

- **STRATEGY**

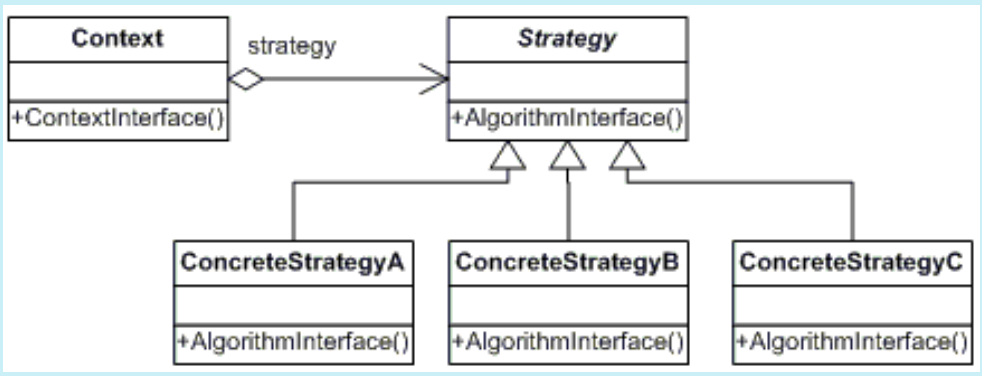


  - **Intenção:** definir uma família de algoritmos, encapsular cada uma delas e torná-las intercambiáveis. Strategy permite que o algoritmo varie independentemente dos clientes que o utilizam.
  - Encapsular algoritmos (“estratégias”) como um objeto.

PADRÕES DE PROJETO

DESIGN PATTERNS

- **STRATEGY**

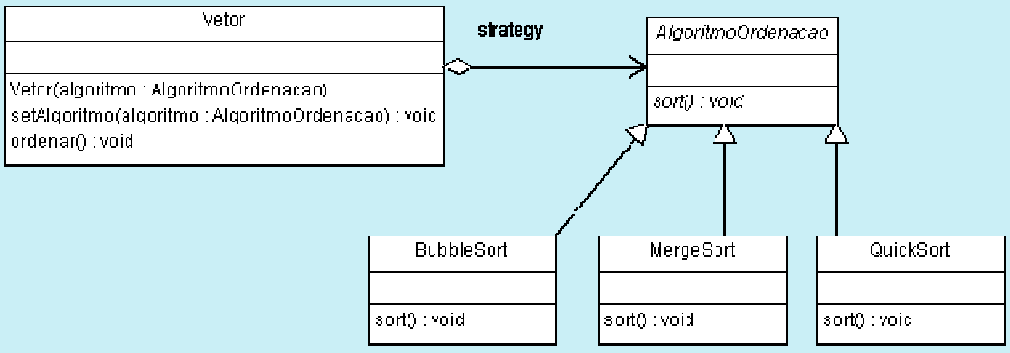


MARCO AURÉLIO REGIS

PADRÕES DE PROJETO

DESIGN PATTERNS

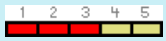
- **STRATEGY - *exemplo***



MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

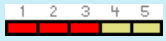
- **STATE**

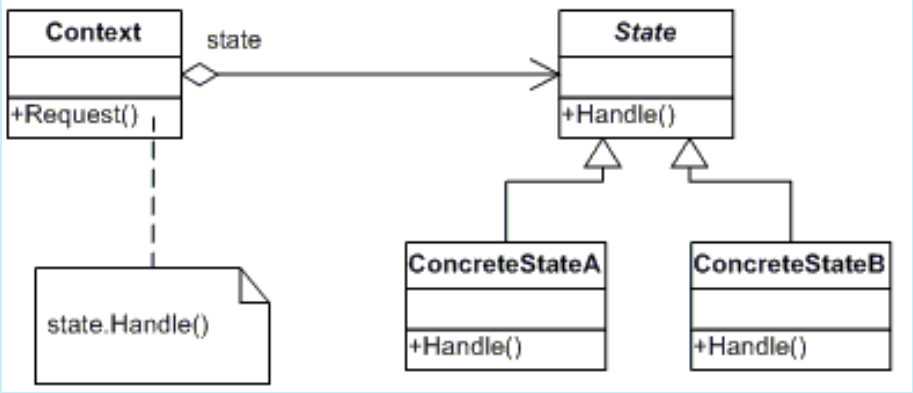


  - **Intenção:** permite a um objeto alterar seu comportamento quando o seu estado interno muda. O objeto parecerá ter mudado sua classe.
  - Alterar o comportamento de um objeto quando seu estado muda.

PADRÕES DE PROJETO
DESIGN PATTERNS

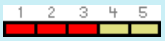
- **STATE**

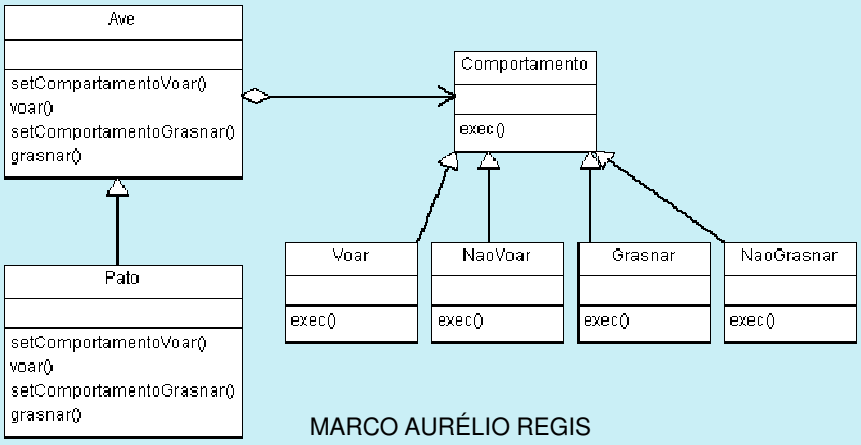




PADRÕES DE PROJETO
DESIGN PATTERNS

- **STATE - *exemplo***

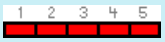




MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **OBSERVER**

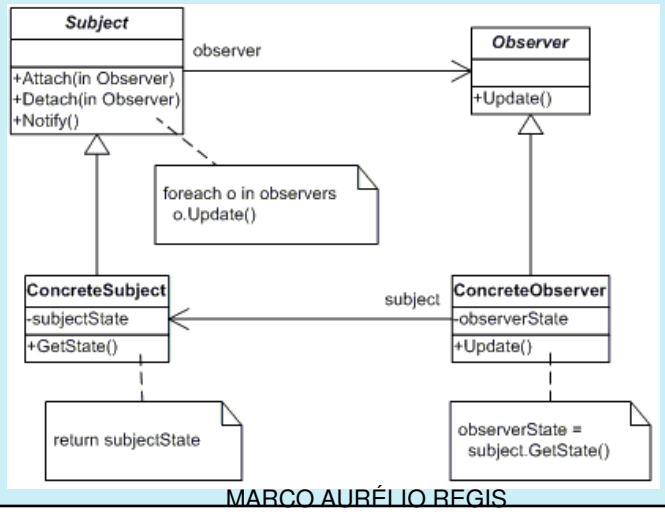


  - **Intenção:** definir uma dependência um-para-muitos entre objetos, de maneira que quando um objeto muda de estado todos os seus dependentes são notificados e atualizados automaticamente.

MARCO AURÉLIO REGIS

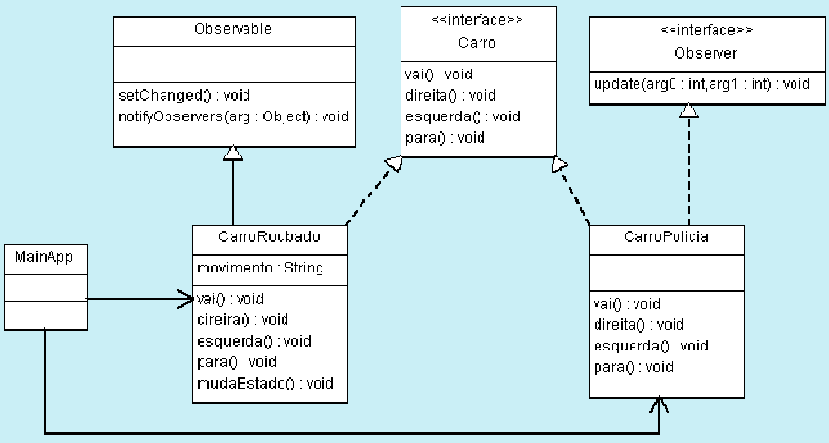
PADRÕES DE PROJETO
DESIGN PATTERNS

- **OBSERVER**



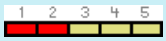
PADRÕES DE PROJETO
DESIGN PATTERNS

- **OBSERVER - *exemplo***



PADRÕES DE PROJETO
DESIGN PATTERNS

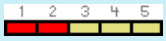
- **MEDIATOR**

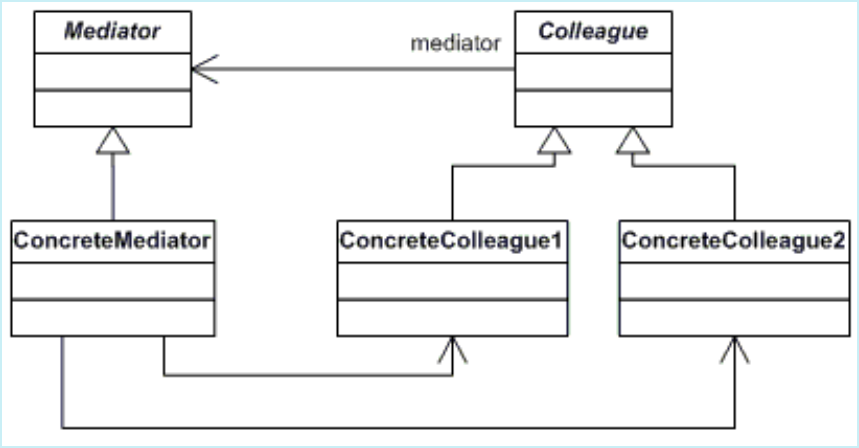


  - **Intenção:** definir um objeto que encapsula a forma como um conjunto de objetos interage O **Mediator** promove o acoplamento fraco ao evitar que os objetos se refiram uns aos outros explicitamente e permite variar suas interações independentemente.
  - Definir uma comunicação simplificada entre as classes.

PADRÕES DE PROJETO
DESIGN PATTERNS

- **MEDIATOR**

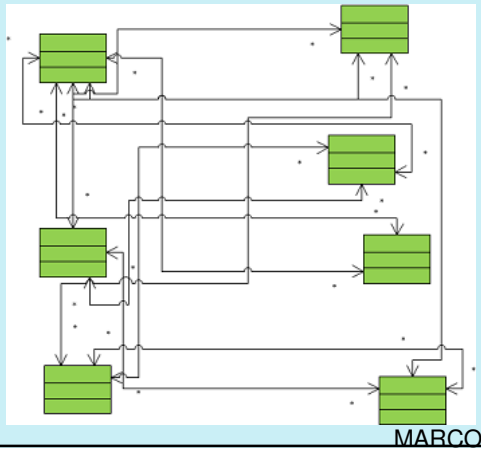




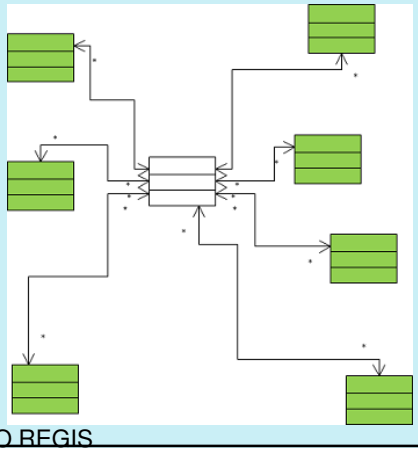
PADRÕES DE PROJETO
DESIGN PATTERNS

- **MEDIATOR**

Sem Mediator



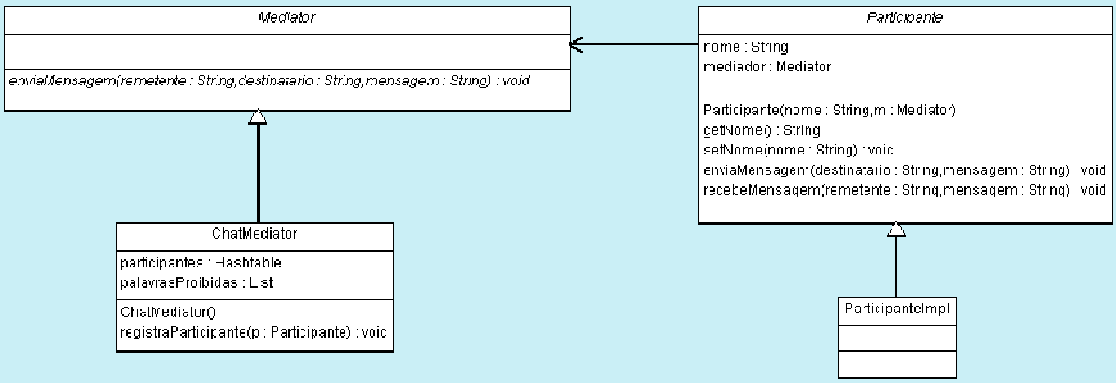
Com Mediator



MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

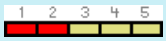
- **MEDIATOR - *exemplo***



MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **CHAIN OF RESPONSIBILITY**

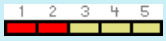


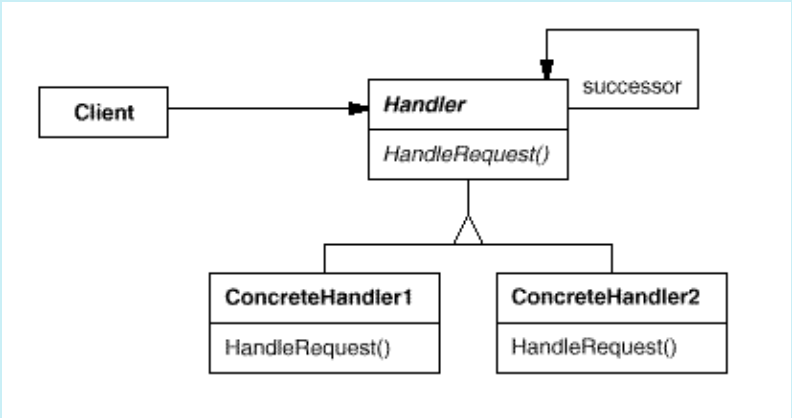
  - **Intenção:** evitar o acoplamento do remetente de uma solicitação ao seu receptor, ao dar a mais de um objeto a oportunidade de tratar a solicitação. Encadear os objetos receptores, passando a solicitação ao longo da cadeia até que um objeto a trate.
  - Uma maneira de passar uma requisição entre uma cadeia de objetos.

MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **CHAIN OF RESPONSIBILITY**





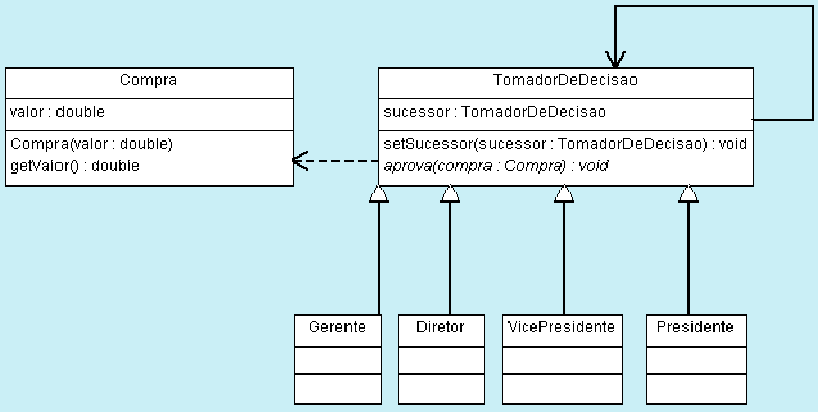
MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

- **CHAIN OF RESPONSIBILITY - *exemplo***

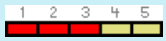
PADRÕES DE PROJETO
DESIGN PATTERNS

- **CHAIN OF RESPONSIBILITY - *exemplo***



PADRÕES DE PROJETO
DESIGN PATTERNS

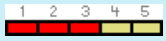
- **TEMPLATE METHOD**

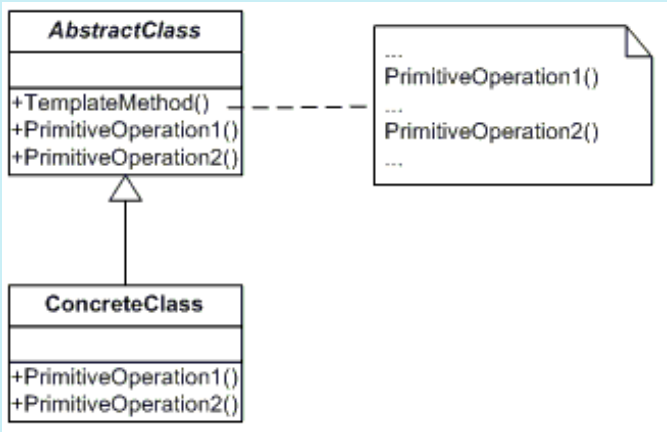


  - **Intenção:** definir o esqueleto de um algoritmo em uma operação, postergando (*deferring*) alguns passos para subclasses. Template Method (Gabarito de Método) permite que subclasses redefinam certos passos de um algoritmo sem mudar a estrutura do mesmo.

PADRÕES DE PROJETO
DESIGN PATTERNS

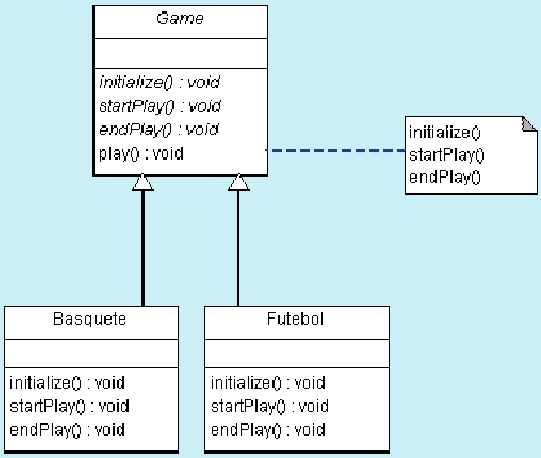
- **TEMPLATE METHOD**





PADRÕES DE PROJETO
DESIGN PATTERNS

- **TEMPLATE METHOD - *exemplo***

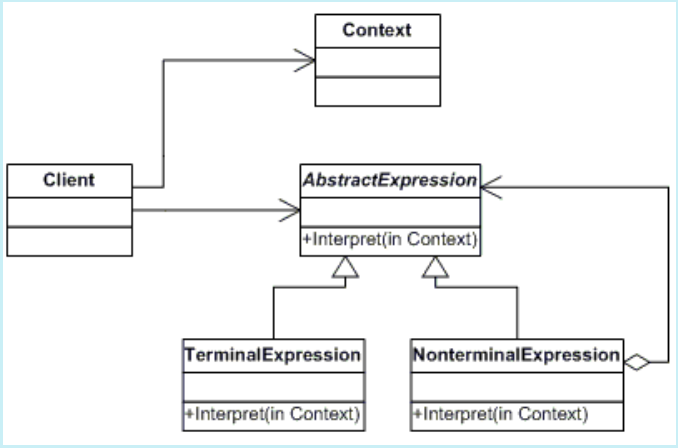


PADRÕES DE PROJETO
DESIGN PATTERNS

- **INTERPRETER**
  - **Intenção:** dada uma linguagem, definir uma representação para a sua gramática juntamente com um interpretador que usa representação para interpretar sentenças da linguagem.
  - Uma forma de incluir elementos da linguagem em um programa.
  - Usa classes para representar cada regra de uma gramática *(expressão regular).*

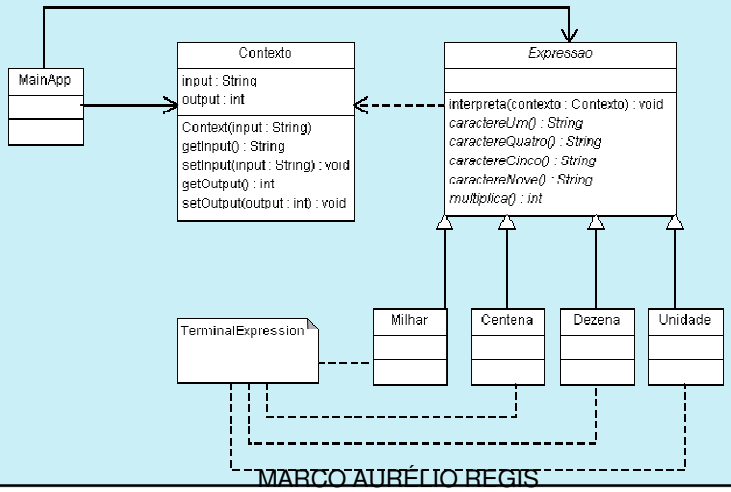
PADRÕES DE PROJETO
DESIGN PATTERNS

- **INTERPRETER**



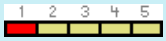
PADRÕES DE PROJETO
DESIGN PATTERNS

- **INTERPRETER**



PADRÕES DE PROJETO
DESIGN PATTERNS

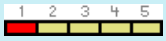
- **MEMENTO**

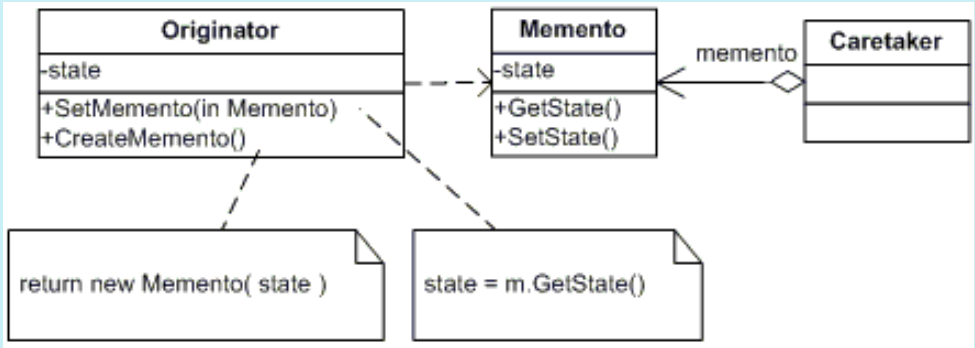


  - **Intenção:** sem violar o encapsulamento, capturar e *externalizar* um estado interno de um objeto, de maneira que o objeto possa ser restaurado para este estado mais tarde.

PADRÕES DE PROJETO
DESIGN PATTERNS

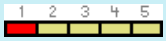
- **MEMENTO**

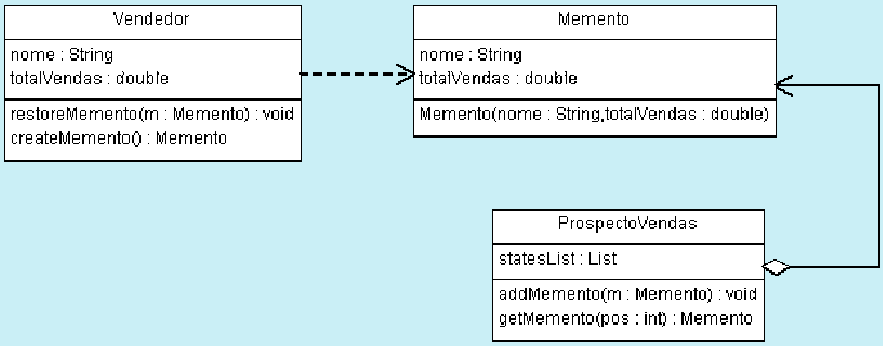




PADRÕES DE PROJETO
DESIGN PATTERNS

- **MEMENTO**





PADRÕES DE PROJETO
DESIGN PATTERNS

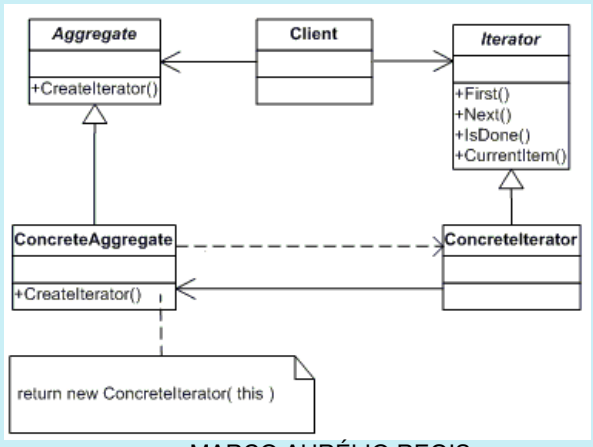
- **ITERATOR**

  - **Intenção:** fornecer um meio de acessar, sequencialmente, os elementos de um objeto agregado sem expor a sua representação.

PADRÕES DE PROJETO

DESIGN PATTERNS

- **ITERATOR**

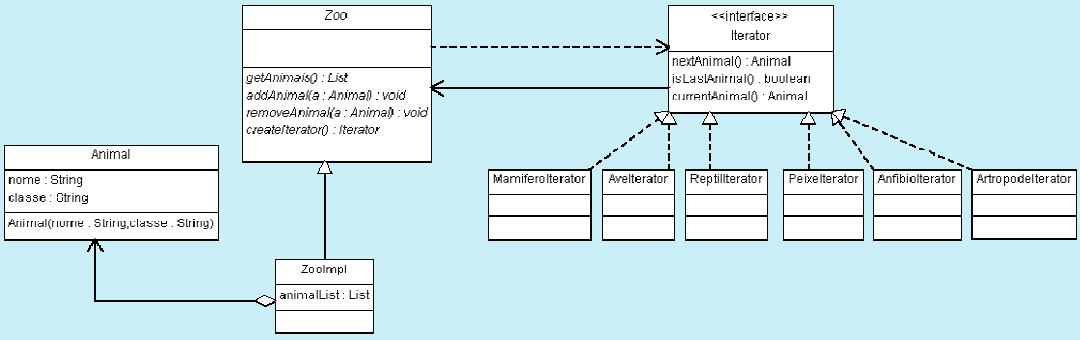


MARCO AURÉLIO REGIS

PADRÕES DE PROJETO

DESIGN PATTERNS

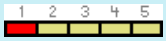
- **ITERATOR**



MARCO AURÉLIO REGIS

PADRÕES DE PROJETO
DESIGN PATTERNS

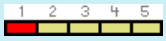
- **VISITOR**

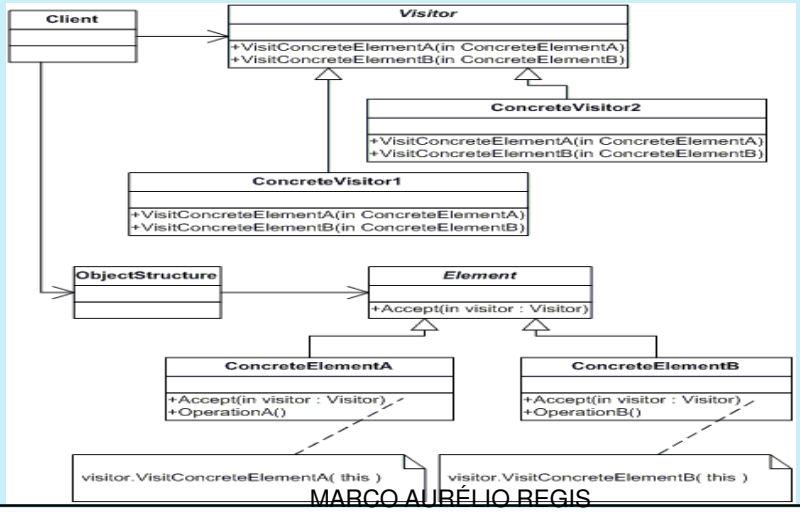


  - **Intenção:** representar uma operação a ser executada nos elementos de uma estrutura de objetos. Visitor permite definir uma nova operação sem mudar as classes dos elementos entre os quais opera.
  - Define uma nova operação a uma classe sem alterá-la

PADRÕES DE PROJETO
DESIGN PATTERNS

- **VISITOR**

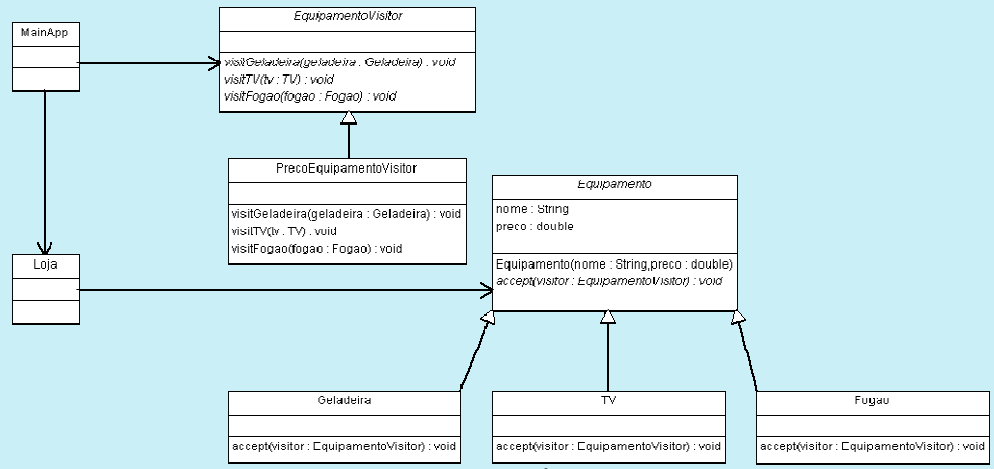




PADRÕES DE PROJETO

DESIGN PATTERNS

- **VISITOR**



MARCO AURÉLIO REGIS

PADRÕES DE PROJETO

DESIGN PATTERNS

OBRIGADO !

"A word of warning and encouragement: Don't worry if you don't understand this book completely on the first reading. We didn't understand it all on the first writing! Remember that this isn't a book to read once and put on a shelf."

MARCO AURÉLIO REGIS