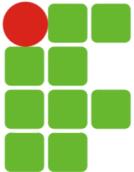


INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
RIO GRANDE DO NORTE  
Campus Natal - Central

# Análise e Projeto Orientados a Objetos

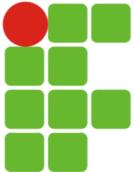
## Padrões de Projeto

Prof. Fellipe Aleixo (*fellipe.aleixo@ifrn.edu.br*)



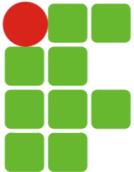
# Padrões de Projeto

- Um padrão de projeto é o projeto de solução um problema comum (que se repete no desenvolvimento de sistemas)
- Catalogados pela GoF (Gamma, Helm, Johnson, Vlissides)
- Categorias:
  - Criacionais
  - Estruturais
  - Comportamentais



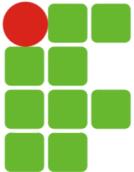
# Padrões Criacionais

- Procuram separar a operação de uma aplicação de como os seus objetos são criados
- Catalogados pela GoF:
  - Abstract Factory, Builder, **Factory Method**, Prototype e Singleton



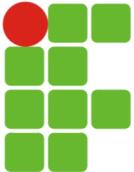
# Padrões Estruturais

- Propõem estruturas genéricas para a solução de um problema, visando sua extensão
- Catalogados pela GoF:
  - Adapter, Bridge, **Composite**, Decorator, **Facade**, Flyweigth e Proxy



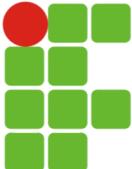
# Padrões Comportamentais

- Utilizam os conceitos de herança, agregação e composição para construir comportamento complexo a partir de componentes simples
- Catalogados pela GoF:
  - Chain of Responsibility, Command, Interpreter, Iterator, **Mediator**, Memento, **Observer**, State, **Strategy**, Template Method e Visitor



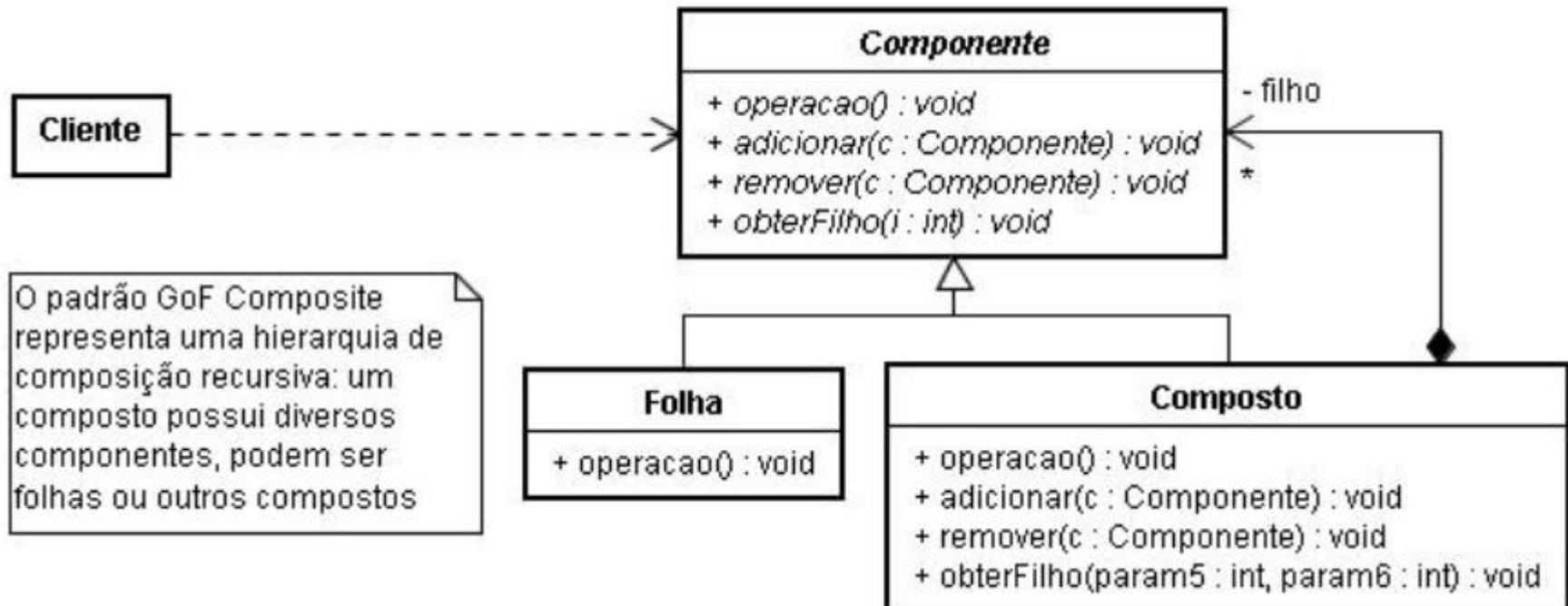
# Padrão *Composite*

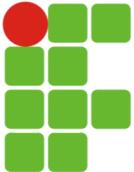
- Problema considerado:
  - Como definir uma relação de hierárquica entre objetos de tal forma que tanto o objeto “todo” quanto seus objetos parte sejam equivalentes em certos aspectos?
- Exemplo:
  - Montagem de peças – uma peça é composta de diversas outras peças, e essas peças componentes são elas próprias compostas, e assim por diante



# Padrão *Composite*

- Estrutura:





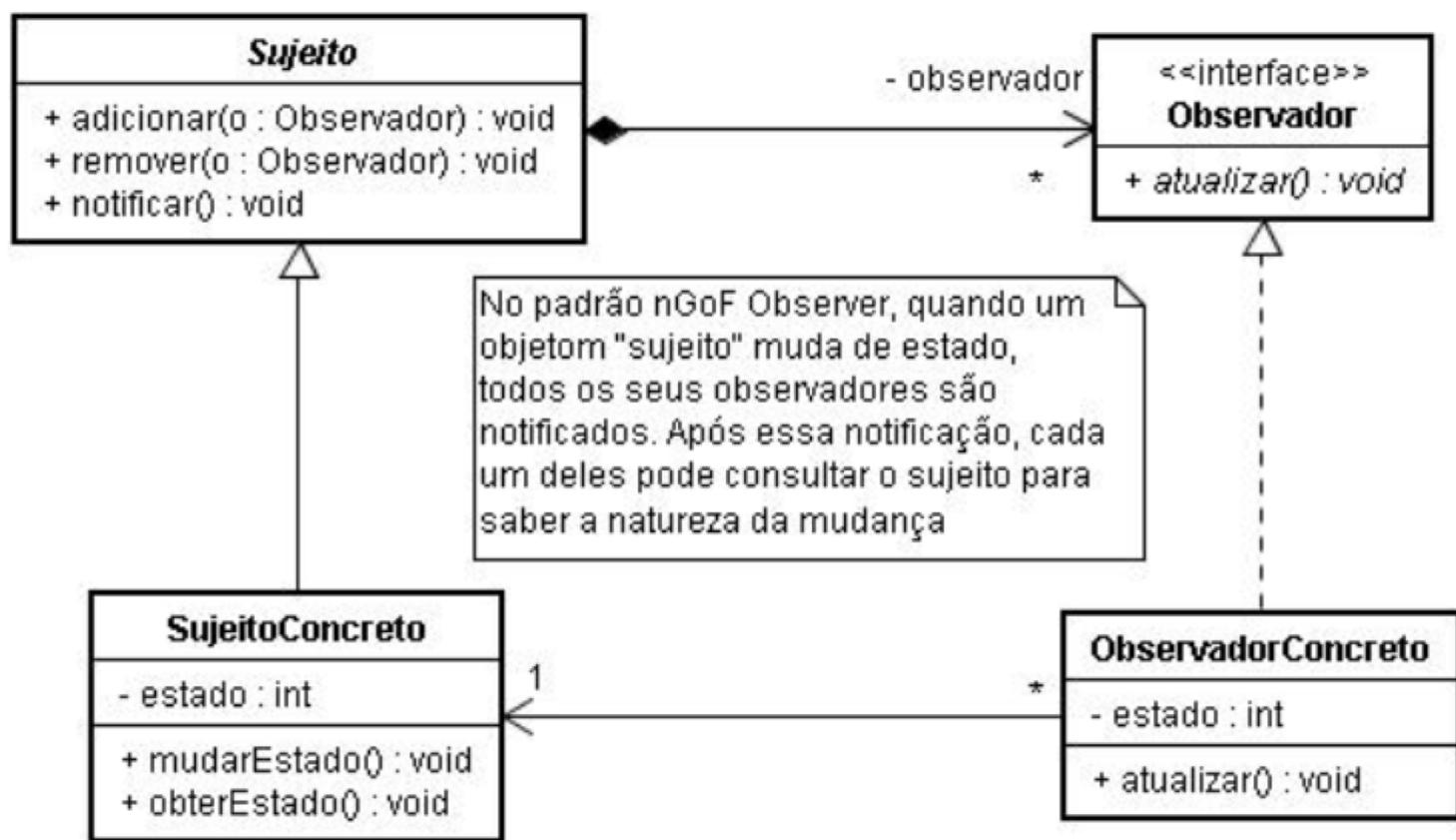
# Padrão *Observer*

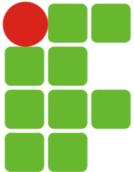
- Problema considerado:
  - Definir uma forma flexível uma dependência “um para muitos” entre objetos
  - Se houver alguma modificação do objeto central, todos dependentes são notificados
  - Necessita-se que um objeto central seja capaz de enviar mensagens de notificação aos seus dependentes sem conhecê-los diretamente



# Padrão *Observer*

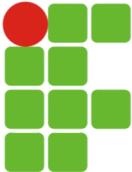
- Estrutura (Java: Observable e Observer):





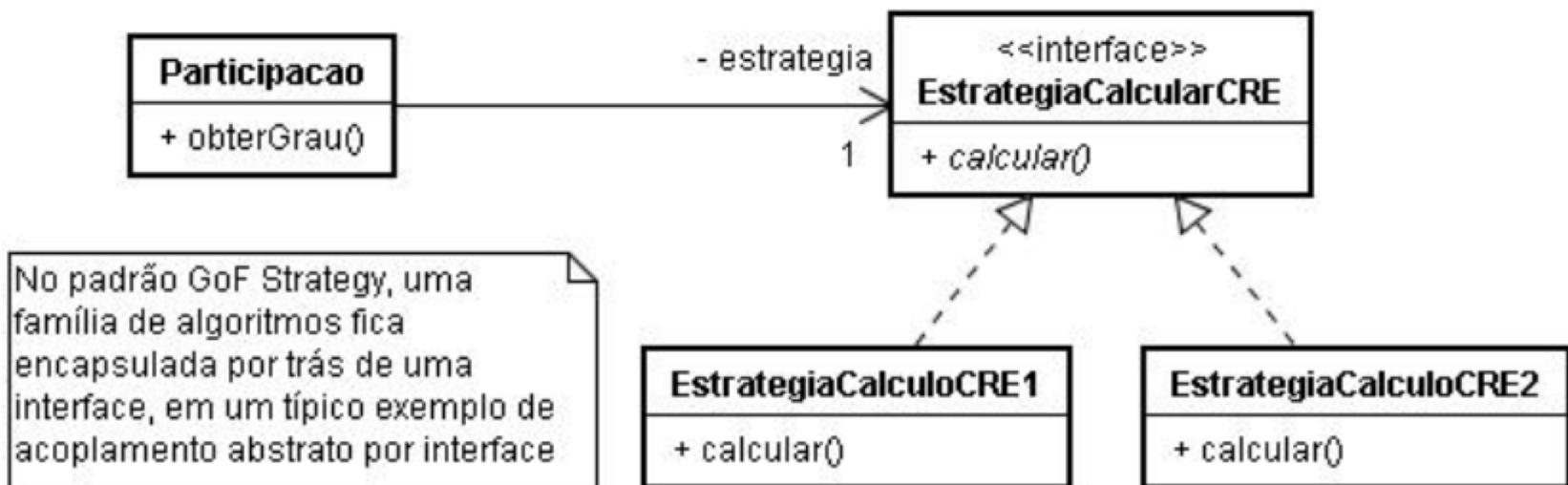
# Padrão *Strategy*

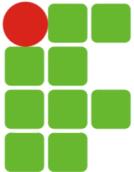
- Problema considerado:
  - Encapsular diferentes algoritmos para realização de alguma tarefa computacional
  - Permitir que os clientes possam utilizar qualquer desses algoritmos sem precisar ser modificado
- Exemplo:
  - Sistema de controle acadêmico – calcular o grau final (conceitos de A a E) de um aluno em uma disciplina cursada



# Padrão *Strategy*

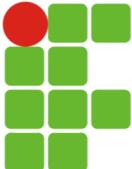
- Estrutura do exemplo:
  - Participação – representa participações de alunos em uma disciplina e suas correspondentes avaliações (notas)





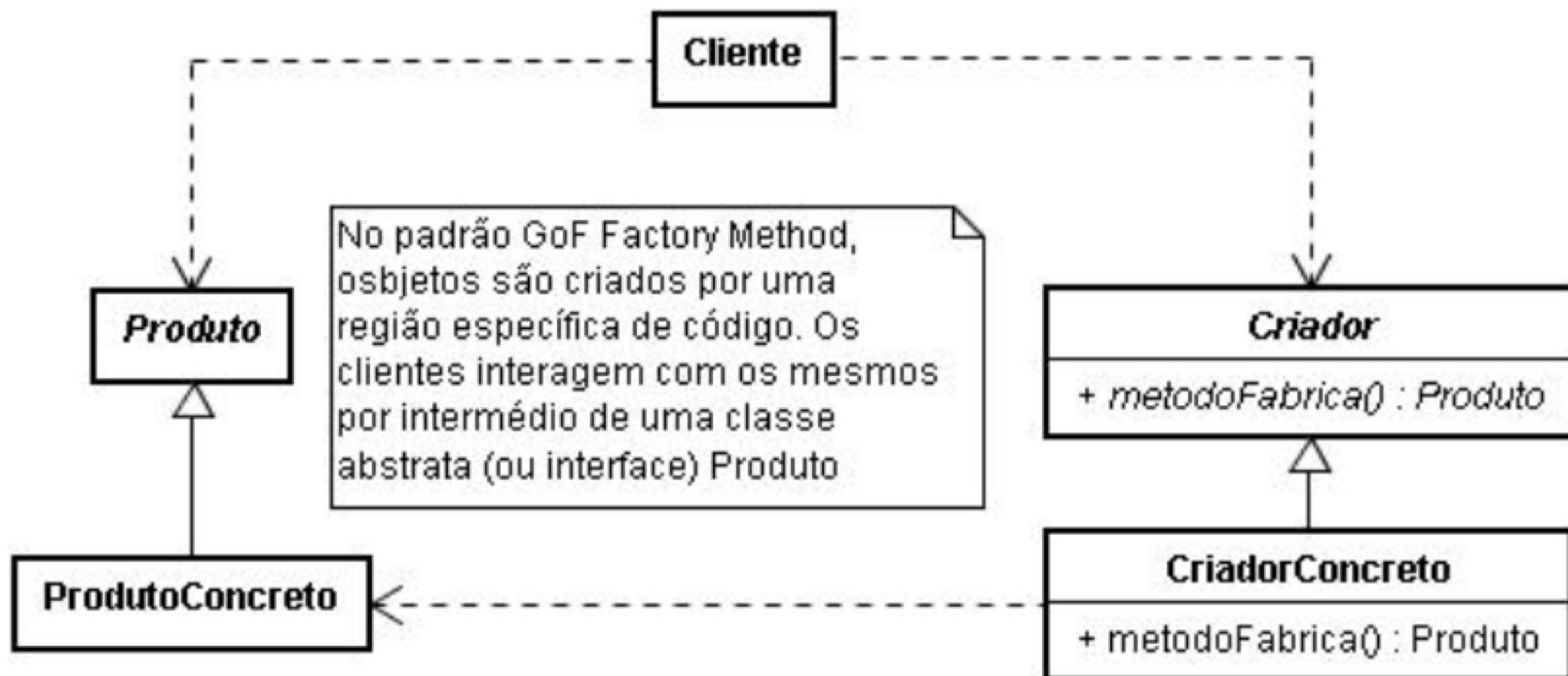
# Padrão *Factory Method*

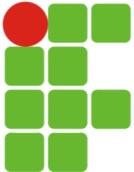
- Problema considerado:
  - Mesmo com um acoplamento abstrato, para se trabalhar com uma referência é necessário antes instanciar um objeto – conhecer a classe
  - O objetivo então é delegar a uma parte específica do código a criação de instâncias – fábrica
  - Uma fábrica cria instâncias de objetos quando requisitada, geralmente retornando uma referência a uma superclasse abstrata ou interface



# Padrão *Factory Method*

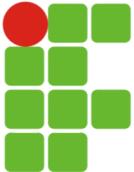
- Estrutura:





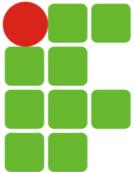
# Padrão *Mediator*

- Problema considerado:
  - Permitir que um grupo de objetos interaja entre si, mantendo um acoplamento fraco entre os componentes
- Solução:
  - Definir um objeto mediador – encapsula a interação entre grupos de objetos
- Exemplo:
  - As classes de “controle” são um exemplo típico de mediadores



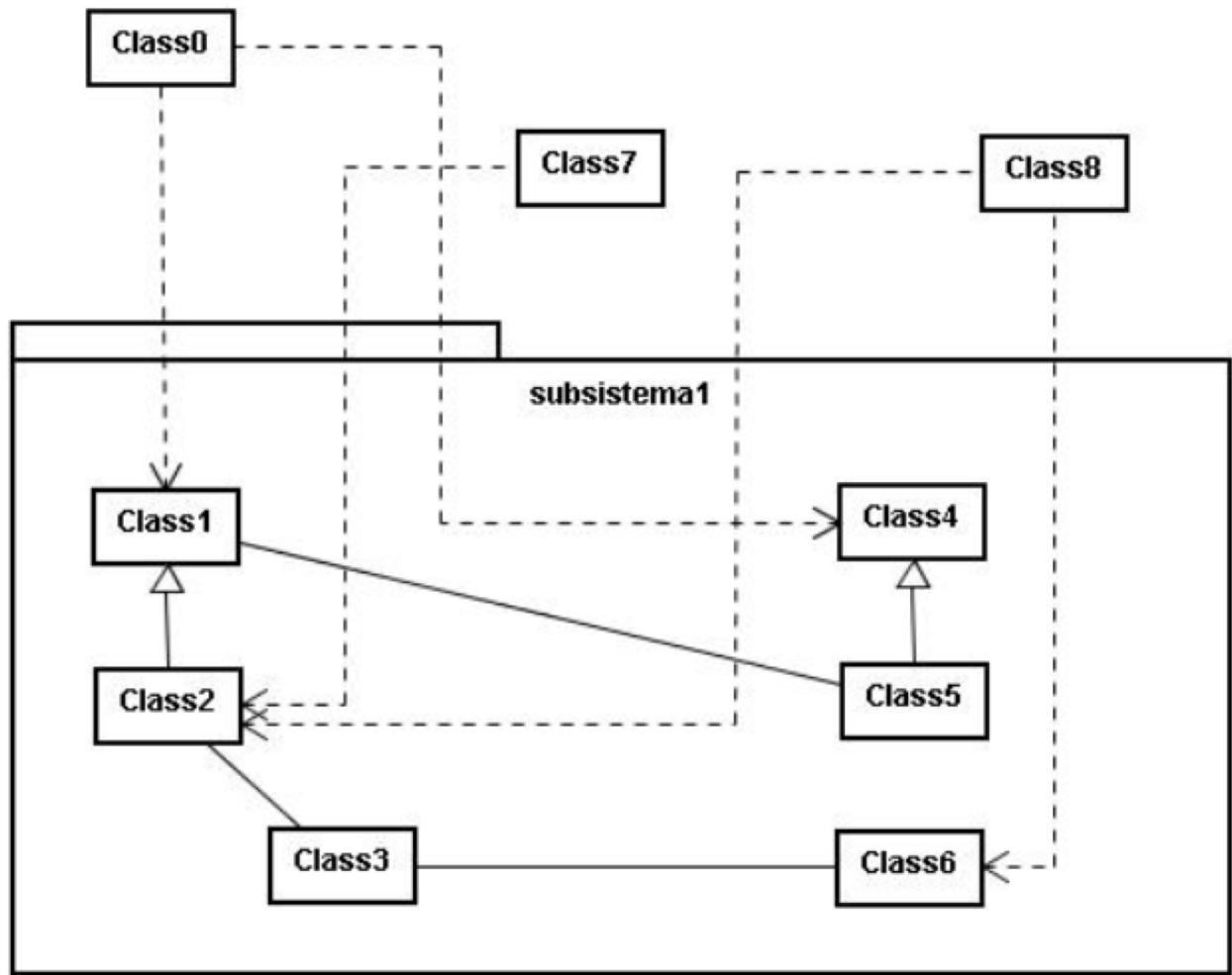
# Padrão *Facade*

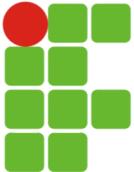
- Problema considerado:
  - Sistema dividido em vários subsistemas, que interagem entre si
  - “como definir uma interface de alto nível que torna um subsistema mais fácil de ser utilizado?”



# Padrão Facade

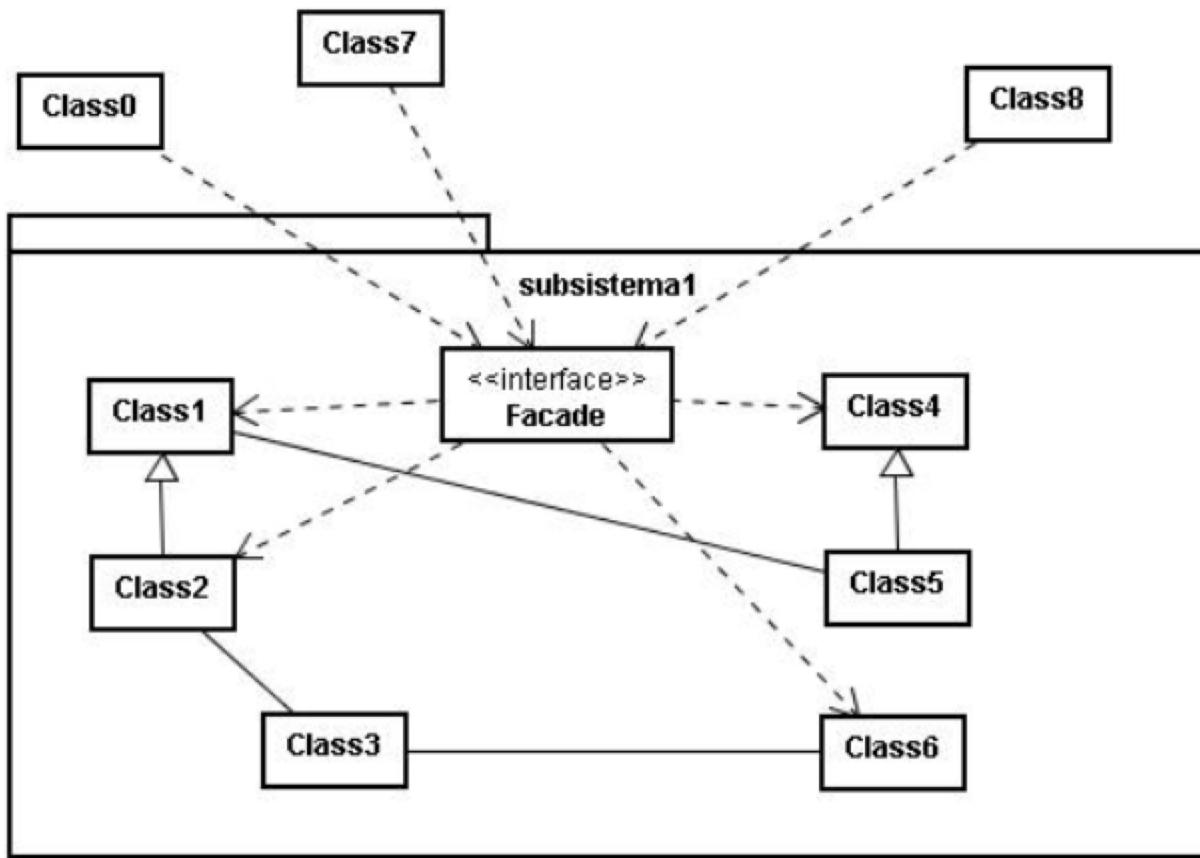
- Exemplo:

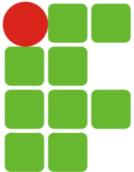




# Padrão Facade

- Estrutura:





# Maiores Referências

- GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLASSIDES, John. **Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos**, 2<sup>a</sup> edição. Bookman, 2000.
- ALUR, Deepak; MALKS, Dan; CRUPI, John. **Core J2EE Patterns: As Melhores Práticas e Estratégias de Design**, 2<sup>a</sup> edição. Campus/Elsevier, 2003.