FeMASS
FACULDADE PROF. MIGUEL ÂNGELO DA SILVA SANTOS

Projeto de Sistemas II

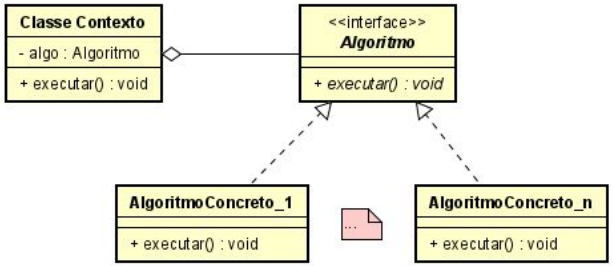
Faculdade Prof. Miguel Ângelo da Silva Santos

Material 4 - Padrão Estratégia (Strategy)

Professor: Isac Mendes Lacerda
e-mail: isac.curso@gmail.com

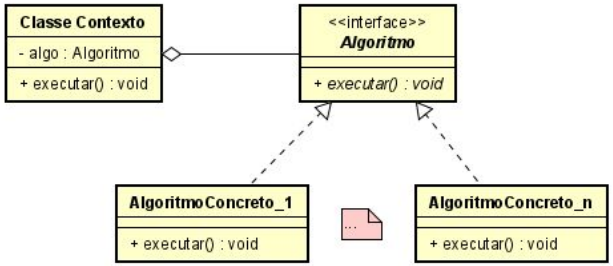
O que é e quando usar?

- ***Estratégia (Strategy)*** é um padrão comportamental incluído no livro do GoF.
- É **utilizado** quando a **classe de contexto** precisa **variar o comportamento** em tempo de execução.
- Organiza **algoritmos em famílias**.
- **Delega comportamento** e faz **uso de composição** que permite **intercambiar algoritmos**.



Motivação

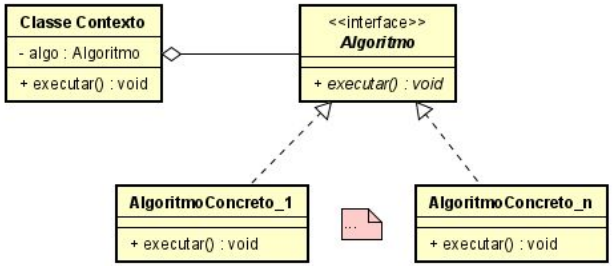
- **Melhora da manutenção** do código, especialmente com a **variação dos requisitos**.



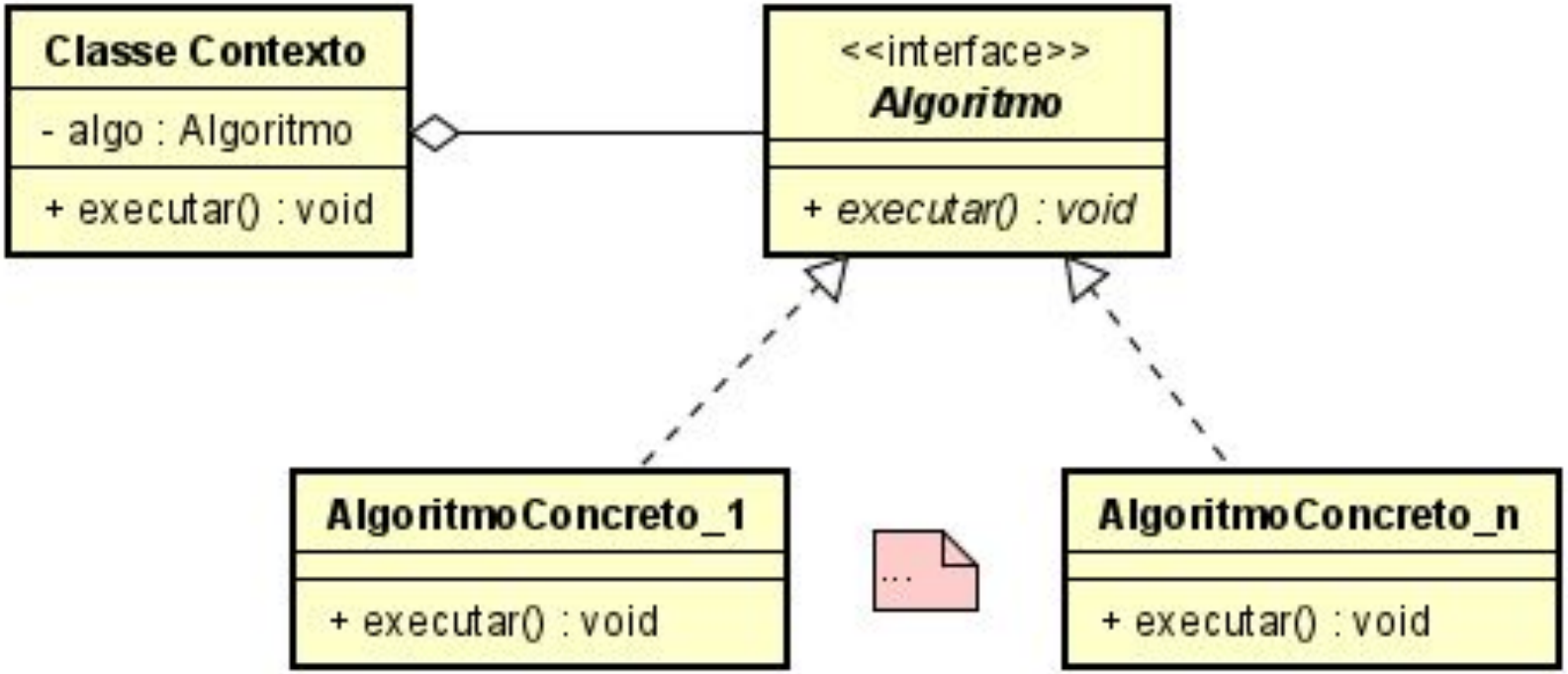
Estrutura conceitual

- A proposta consiste em **delegar responsabilidade** para **instâncias** que **compõem a classe de contexto**.

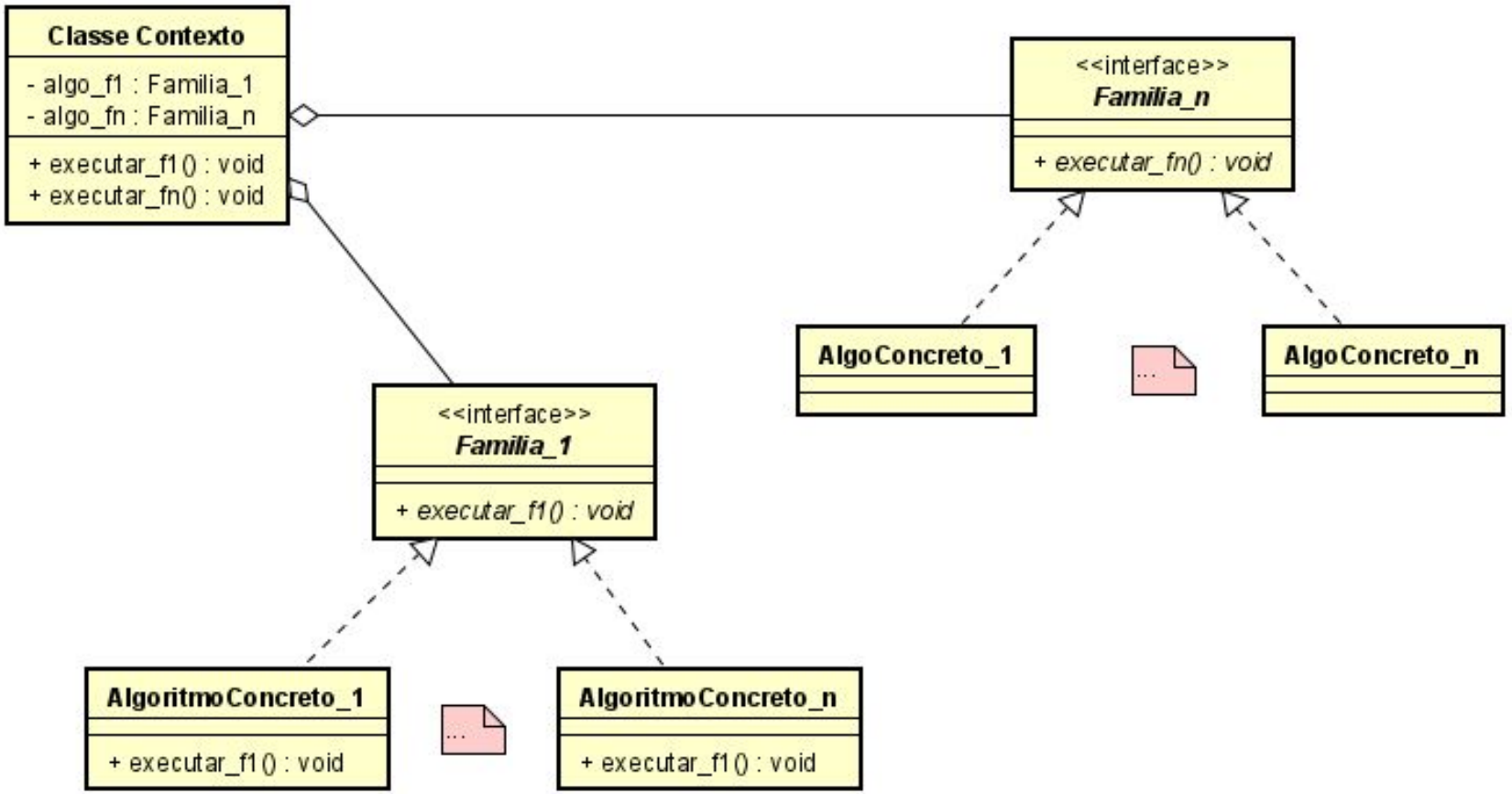
- Inclui: **classe de contexto**, **interface** ou **classe abstrata** e **classes concretas**.



Estrutura conceitual



Estrutura conceitual (famílias)



Estrutura conceitual (contexto e interface)

```
12  public class Contexto {
13      Algoritmo algoDefault;
14
15      Contexto (){
16          this.algoDefault = new AlgoritmoConcreto_1();
17      }
18      void set_algo(Algoritmo algo){
19          this.algoDefault = algo;
20      }
21
22      void executar(){
23          this.algoDefault.executar();
24      }
25
26      void executar(Algoritmo algo){
27          algo.executar();
28      }
29  }
```

```
    public interface Algoritmo {
13
        void executar();
15
16  }
```

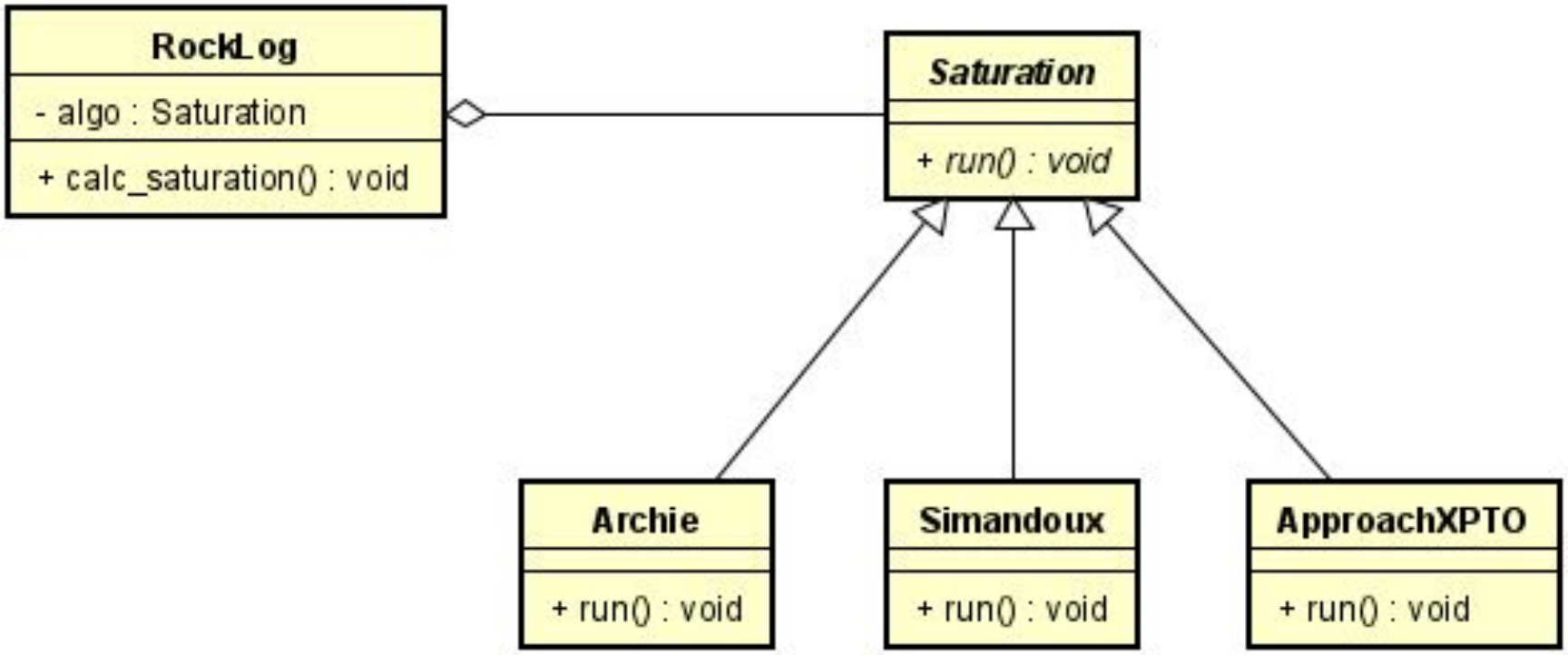
Estrutura conceitual (algoritmos 1 e 2)

```
12  public class AlgoritmoConcreto_1 implements Algoritmo{
13
14      /** Algoritmo 1 ...3 lines */
17      @Override
        public void executar(){
19          System.out.println("Olá, eu sou o algoritmo 1 e fui executado!");
20      }
21
22  }
```

```
12  public class AlgoritmoConcreto_2 implements Algoritmo{
13
14      /** Algoritmo 2 ...3 lines */
17      @Override
        public void executar(){
19          System.out.println("Olá, eu sou o algoritmo 2 e fui executado!");
20      }
21  }
```

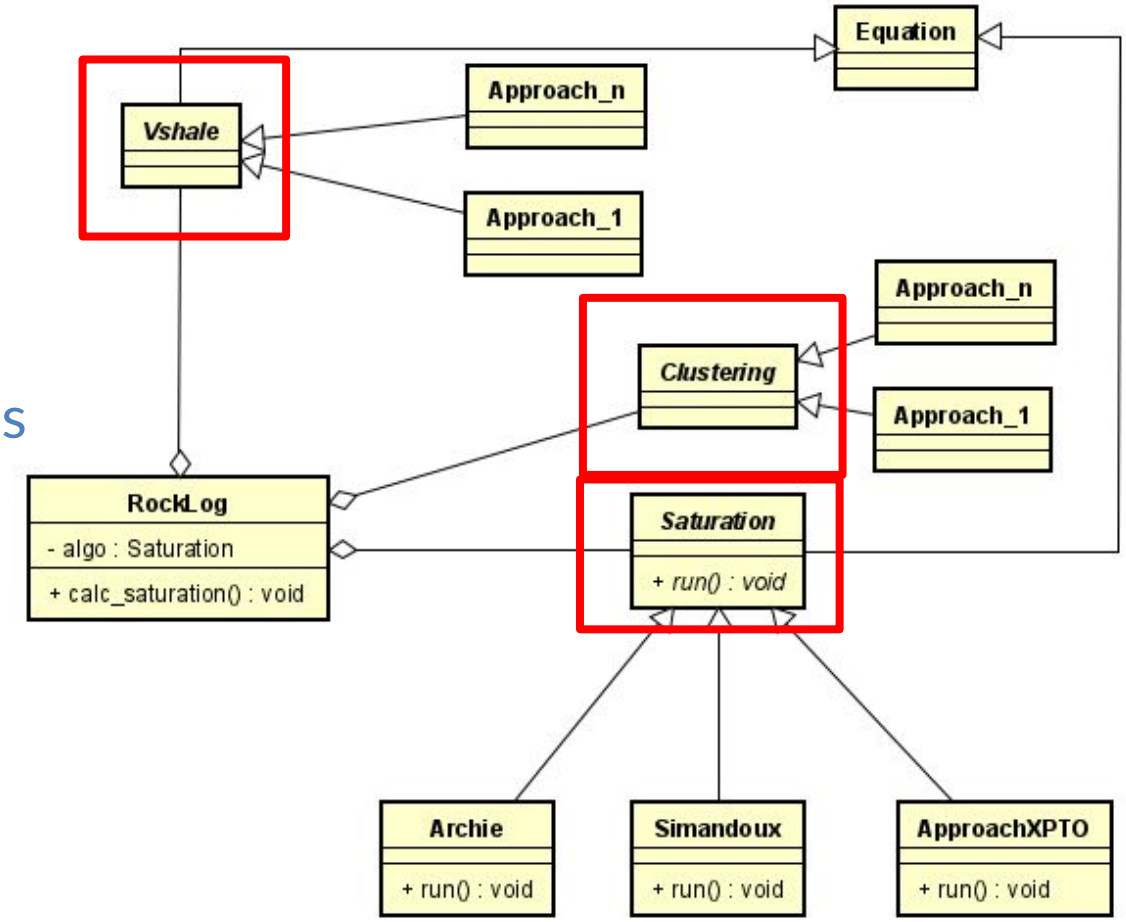
Um exemplo real

Strategy: no GRIPy



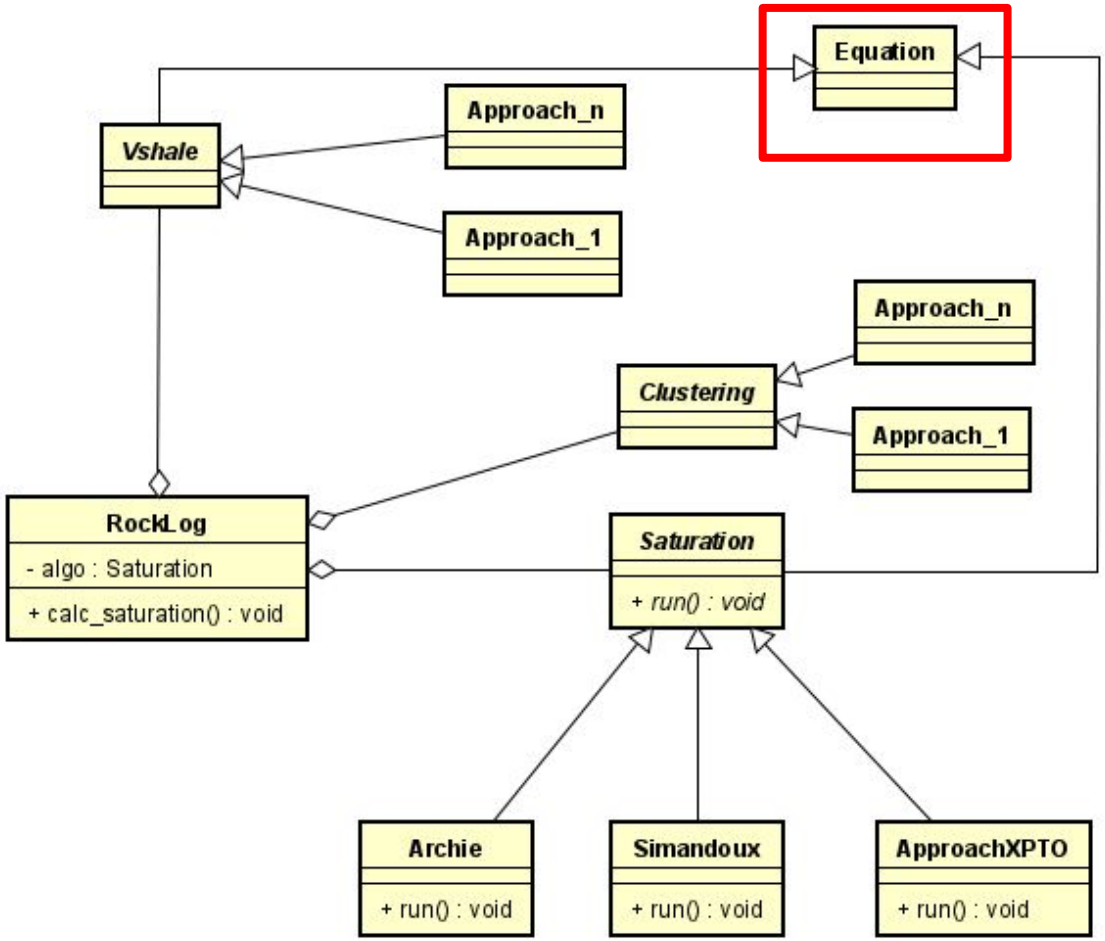
Strategy: n0 GRIPy

Inclusão de várias famílias de algoritmos

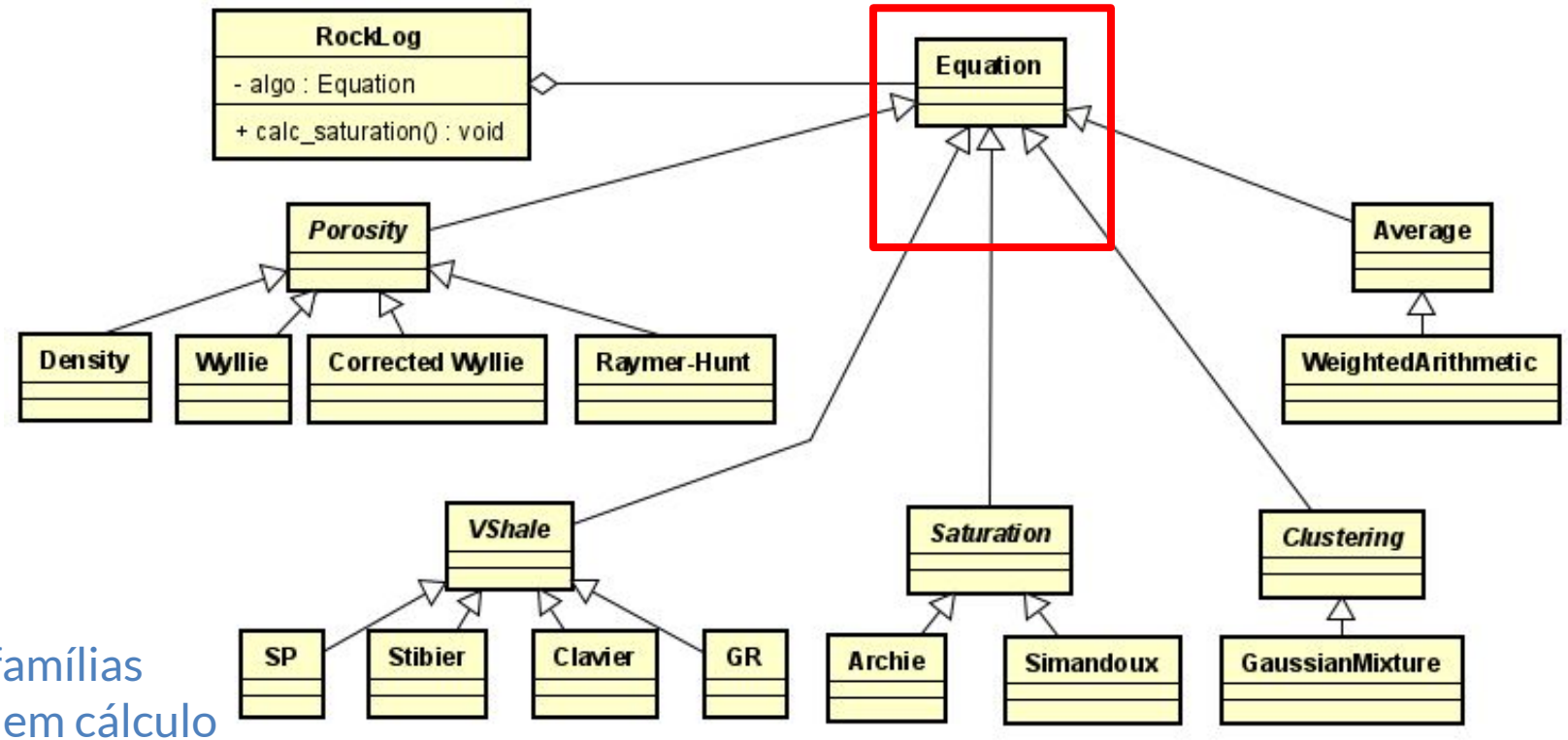


Strategy: n0 GRIPy

Todas as famílias baseadas em cálculo podem herdar de *Equation*



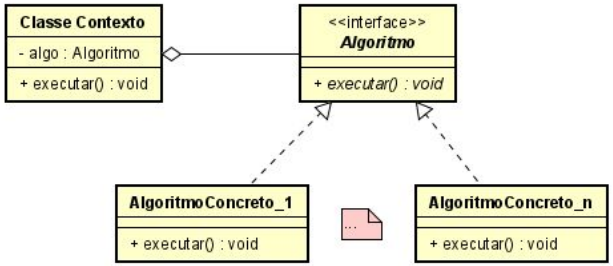
Strategy: n0 GRIPy



Todas as famílias baseadas em cálculo podem herdar de *Equation*

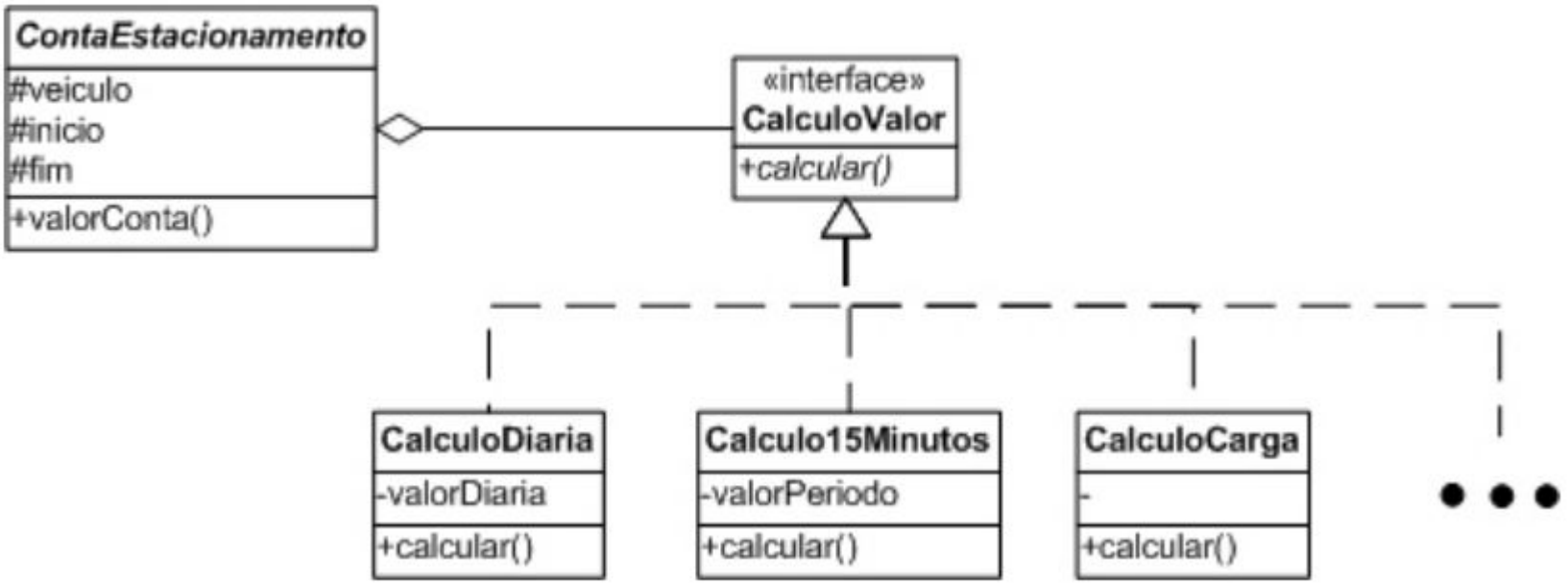
Vantagens e desvantagens

- Posso **mudar o algoritmo dinamicamente sem mudar** a estrutura de **classe**.
- **Novas implementações** podem ser **introduzidas posteriormente**.
- **Menos lógica condicional** na classe principal.
- Troca dinâmica de comportamento.
- **Aumento do número de classes**: há uma para cada algoritmo.



Exercício 1

Baseado no modelo parcialmente elaborado abaixo, inclua ao menos mais dois algoritmos hipotéticos relacionados a regras de cálculo de estacionamento. Implemente seu modelo completo baseado no padrão *Strategy* (se necessário, inclua atributos).



Exercício 2

Baseado no modelo parcialmente elaborado abaixo, inclua ao menos mais três algoritmos hipotéticos relacionados a regras de cálculo de desconto para vendas de livros. Implemente seu modelo completo baseado no padrão *Strategy* (se necessário, inclua atributos).