Modelagem de Sistemas Orientado a Objetos com UML.

Capítulo 5

Modelagem Estática
Diagrama de Classes
Conceitos Avançados
Parte III

Ana Paula Gonçalves Serra, Dr.

Onde Estamos na Disciplina de Modelagem de Sistemas Orientado a Objetos com UML?

2

1 Conceitos fundamentais de orientação a objetos.

2 Estruturação e modelagem de sistemas.

3 Diagramas de classes – Parte III de III.

4 Diagrama de sequência.

5 Realização de Casos de Uso.

6 Diagrama de estados.

7 Diagrama de atividades.

8 Diagramas de Implementação (Pacote, Componente e Distribuição)

3

Objetivos do Capítulo

- Este capítulo tem por objetivo apresentar ao alunos os seguintes conceitos:

1. Discussão do Exercício - Relacionamentos
2. Dependências
3. Classes Abstratas
4. Classes de Interface
5. Polimorfismo
6. Exercício

4

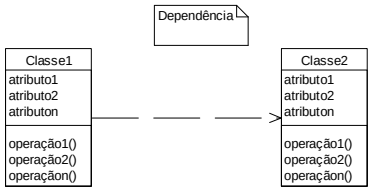
1. Relacionamentos

Complete:

5

2. Dependência

- Conexão semântica entre 2 objetos:
  - um independente e outro dependente;
- Alterações no objeto independente causam alterações no elemento dependente;
  - Por exemplo:
    - uso de atributos e operações;
    - Uma classe como parâmetro.



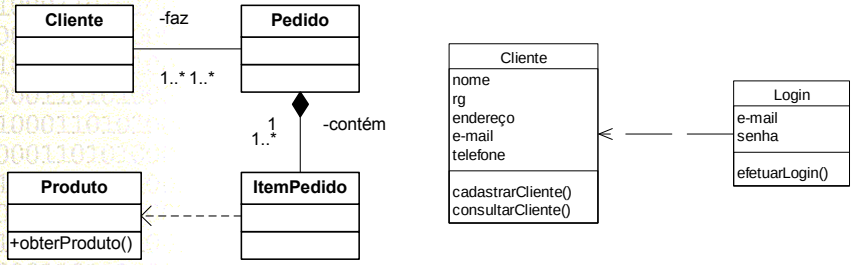
- Pode ser usada mesmo que as classes não se relacionem;
- Podem violar o conceito de encapsulamento (mecanismo usado para ocultar dados, estrutura interna e detalhes de implementação de algum elemento).

Pós-Graduação em Eng. de Software
Universidade São Judas Tadeu
Modelagem de Sistemas
Orientado a Objetos com UML
Ana Paula G. Serra

6

2. Dependência

Exemplo:



Pós-Graduação em Eng. de Software
Universidade São Judas Tadeu
Modelagem de Sistemas
Orientado a Objetos com UML
Ana Paula G. Serra

7

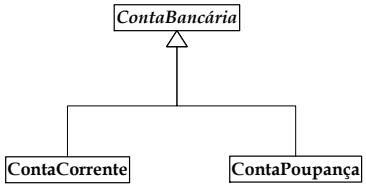
3. Classes Abstratas

- São classes que não podem criar instâncias, porque algumas operações não são implementadas e existem somente em hierarquia;
- Classes abstratas são usadas para especificar um contrato (define responsabilidades e pós-condições de uma operação ou é usado para fazer referência a um conjunto de condições de uma interface) que todas as instâncias devem respeitar;
- Uma classe pode possuir tanto operações abstratas quanto operações concretas.
- Uma classe que contém uma operação abstrata é chamada de *classe abstrata*.

8

3. Classes Abstratas

- Uma classe abstrata pode ter subclasses abstratas, mas nos níveis mais baixos (especializados) devem estar classes concretas;
- Podem herdar de classes abstratas e não abstratas;
- Podem possuir dados (atributos);
- Em UML, classes e operações abstratas são representadas com o nome em *itálico*. Também pode-se utilizar o estereótipo *<<abstract>>*

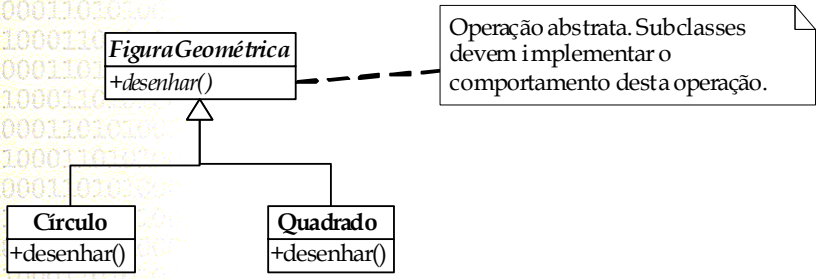


3. Classes Abstratas

- Uma subclasse herda todas as propriedades de sua superclasse que tenham visibilidade pública ou protegida.
- Entretanto, pode ser que o comportamento de alguma operação herdada seja diferente para a subclasse.
  - Nesse caso, a subclasse deve redefinir o comportamento da operação.
  - A assinatura (nome e parâmetros) da operação pode ser reutilizada.
  - A implementação da operação (método) é diferente.

3. Classes Abstratas

Exemplo

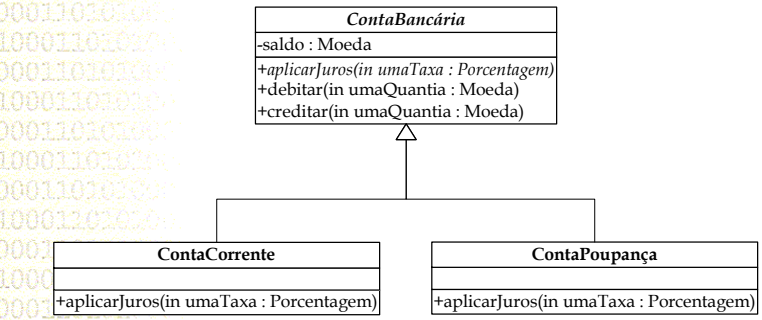


As classes Círculo e Quadrado são concretas, pois fornecem implementação para a operação abstrata herdada.

11

3. Classes Abstratas

Exemplo

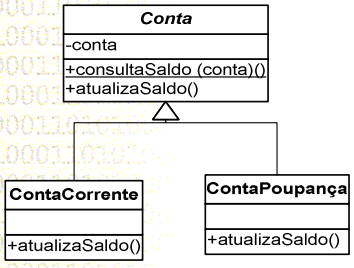


Classes ContaCorrente e ContaPoupança redefinem a operação aplicarJuros.

12

3. Classes Abstratas

Exemplo




```
public abstract class  Conta
private int conta;
{
  public consultaSaldo (int conta)
  {
    código
  }
  public abstract atualizaSaldo()
}
```

13

4. Classes de Interface

- Uma interface é uma classe abstrata que tem unicamente operações abstratas;
- Nenhum método implementado e nenhum atributo;
- Uma interface é uma coleção de operações utilizadas para especificar um serviço de uma classe e estabelece o comportamento das classes que a implementa;

14

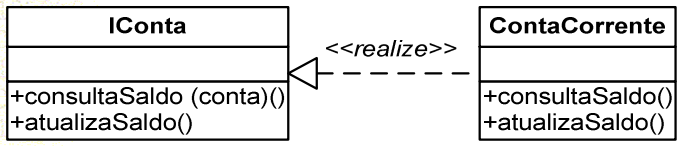
4. Classes de Interface

- Interfaces podem ser representadas como classes, com um estereótipo <<interface>>.
- A UML sugere incluir um "I" antes do nome da interface.

15

4. Classes Interface

Exemplo




```
public interface IConta
{
  public consultaSaldo (int conta)
  public atualizaSaldo()
}
```


16

4. Diferenças entre Classes Abstratas e Classes de Interface

- Interface é usada quando não há necessidade das subclasses herdarem métodos já implementados;
- Por definição, interfaces não permitem a declaração de atributos.

17

5. Polimorfismo

- É o conceito pelo qual duas ou mais subclasses podem invocar métodos que têm a mesma assinatura, mas comportamentos distintos, especializados para cada subclasse;
- A **assinatura de um método** é composta pelo tipo e pelos parâmetros que recebe;
- A assinatura é repetida na(s) subclasse(s) para enfatizar a redefinição de implementação.

18

5. Polimorfismo

- Para haver polimorfismo, é necessário que os métodos tenham exatamente a mesma assinatura, sendo utilizado o mecanismo de redefinição de métodos;
- Esse mecanismo de redefinição não deve ser confundido com o mecanismo de **sobrecarga de métodos**.

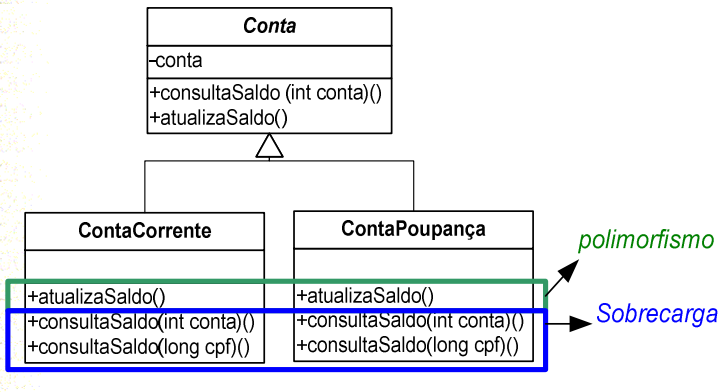
19

5. Polimorfismo

- Sobrecarga de métodos:
  - No mecanismo de sobrecarga (overloading), dois métodos de uma mesma classe podem ter o mesmo nome, desde que sua assinatura seja diferente;
  - Tal situação não gera conflito, pois o compilador é capaz de detectar qual método deve ser escolhido a partir da análise dos tipos de argumentos do método.

20

5. Polimorfismo



21

6. Discussão Exercício – Comércio Eletrônico

(Solução)

22

7. Exercício

Desenvolva o modelo de classes do Sistema de Hotel

Ver especificação em: ExercicioHotel.pdf

Referências Bibliográfica

- UML – Guia do Usuário. 2ª. ed. Grady Booch; James Rumbaugh; Ivar Jacobson. Editora Campus. 2006. ISBN: 10-85-352-1784-3.
- Larman, Craig; Utilizando UML e Padrões, 2a. Edição. Bookman, 2003. ISBN: 85-363-0358-1.
- UML Essencial. 3ª. ed. Martin Fowler; Kendall Scott. Editora Bookman. 2000. ISBN: 85-363-0454-5.