Orientação a Objetos
Introdução

Prof. Lucas Boaventura

Paradigmas de Programação

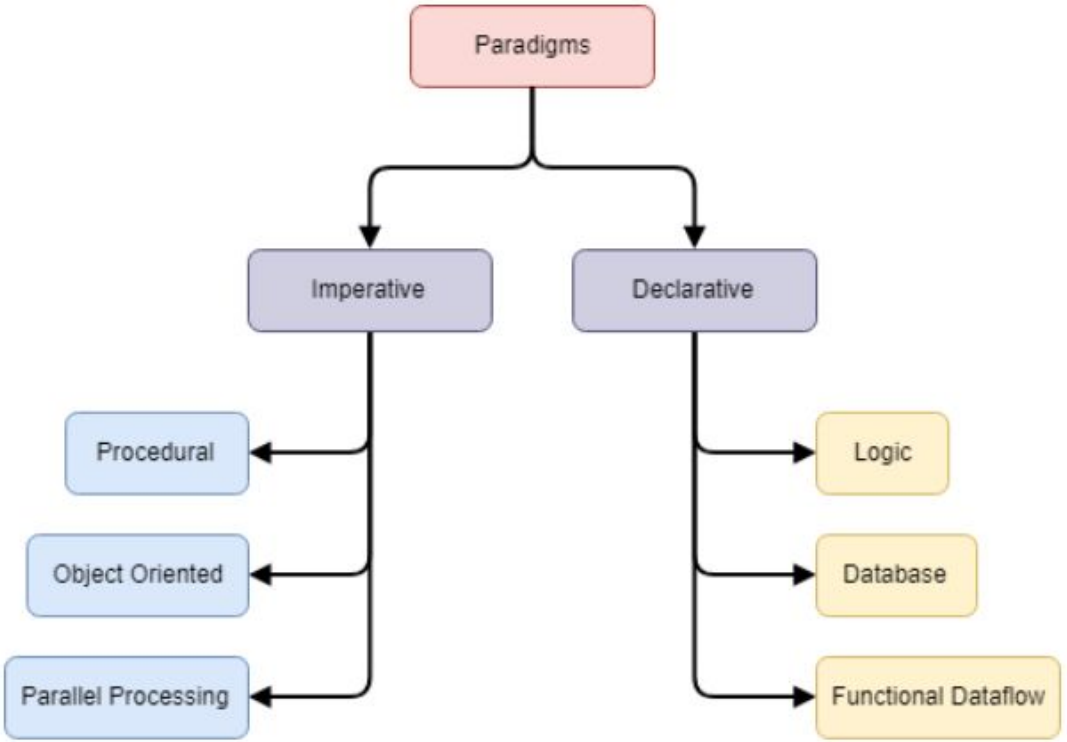
- Sem estrutura
  - Programas pequenos e muitas vezes com acesso a variáveis globais
  - O mesmo código deve ser repetido para executar um procedimento mais de uma vez.

- Procedural
  - O programa fica mais estruturado e pode ser visto como uma sequência de procedures. Um programa com várias partes.

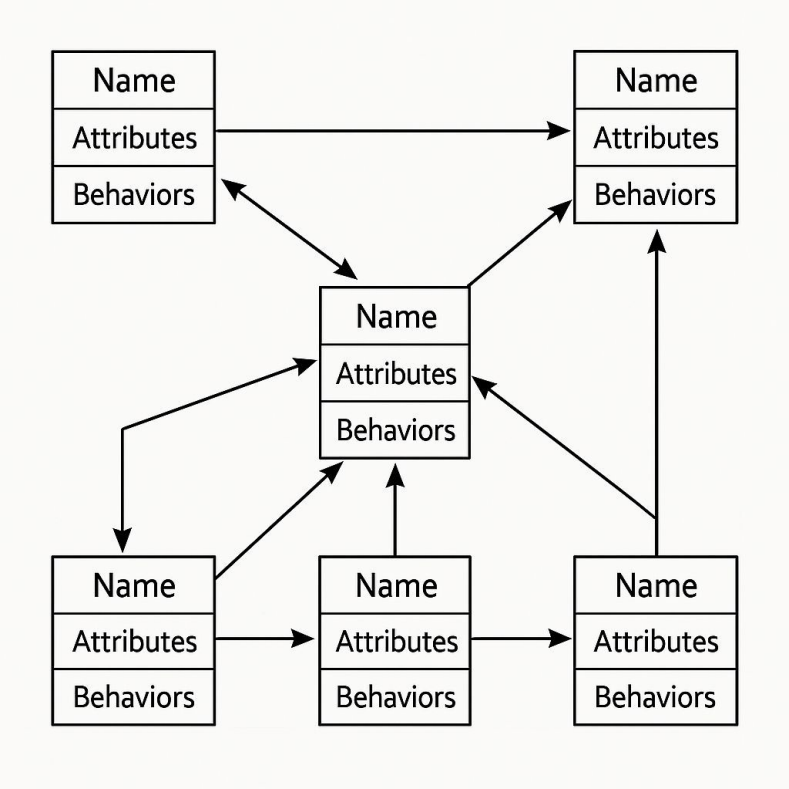
Paradigmas de Programação

- Modular
  - Funcionalidades são agrupadas em módulos.
  - Cada módulo é único.
- Funcional
  - Imutabilidade
  - Recursão em vez de iteração
- **Orientada a Objetos**

Paradigmas de Programação



Orientação a Objetos



Conceitos Básicos

- Programação Procedural (Código contínuo)
- Criação de Objetos para melhorar a organização
- Conceito para aumentar a aderência com a realidade
- Orientação a Objetos é um Paradigma de programação.

Orientação a Objetos

- **O que é?**
  - Paradigma de Programação baseado em Objetos
  - Baseada no princípio da reutilização de componentes

Orientação a Objetos

- **Para que serve?**
  - Organização
  - Colaboração
  - Reaproveitamento / Reusabilidade
  - Flexibilidade
  - Melhoria da Manutenabilidade do código

Linguagens Orientadas a Objeto

- C++
- C#
- Java
- JavaScript
- Perl
- PHP
- Python
- Objective-C
- Ruby
- Swift
- Kotlin
- Dart

Fundamentos de OO

- Abstração
  - Objeto
  - Classe
- Encapsulamento
- Herança
- Polimorfismo

Abstração

- Extrair a essência de objetos da realidade
- Capacidade de compreender o contexto ao qual cada objeto pertence e definir as características essenciais do mesmo para este determinado contexto
- Exemplo:
  - Carro (venda) vs. Carro (fábrica)

Objeto

- O que é um objeto?
  - Algo que existe e tem identidade própria
  - Um objeto pode conter outro objeto
  - Cada objeto possui características ou atributos que descrevem seu estado
  - A maioria dos objetos possui vários atributos
  - Além disso, um objeto tem comportamento

Objeto

- Exemplos:
  - Objetos concretos: Copo, Caneta, Mochila, Carro
  - Objetos abstratos: conta de banco, conexão de rede.
- Uma maneira de identificar se algo pode ser caracterizado como objeto é ver se ele é descrito por um substantivo.

Classe

- Usada para criar objetos
- Descreve o que o objeto será, é um modelo do objeto, uma planta do objeto.
- Composta por:
  - Nome
  - Atributos (Propriedades, Dados, Variáveis Internas)
  - Métodos (Operações, Comportamento)

Atributos

- Atributos são as características de um objeto. É a estrutura de dados que vai representar a classe.
- Ex: Classe Pessoa - Atributos: nome, endereço, telefone, CPF,...; Classe Carro - Atributos: nome, marca, ano, cor, ...; Classe Livro - Atributos: autor, editora, ano.
- O valor do atributo identifica o objeto e informa seu estado.

Métodos

- Definem os comportamentos dos objetos.

- São normalmente são públicos, sendo assim os meios de interação da entre classes.

Métodos

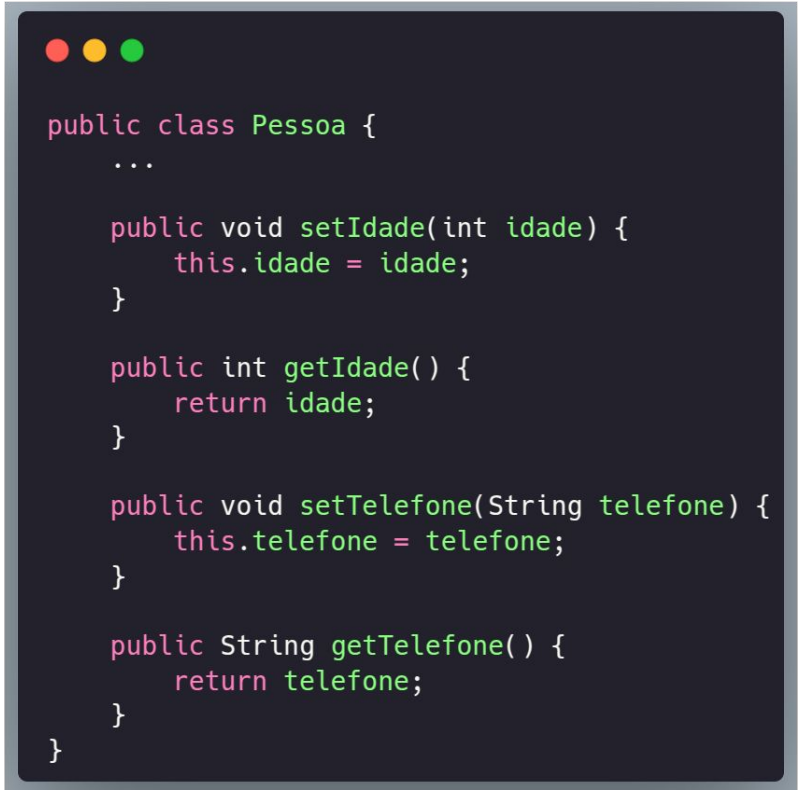
- **Construtores:** responsáveis pela alocação de memória e inicialização de dados, sendo sempre chamados automaticamente na declaração um novo objeto
- **Destrutores:** chamados quando o objeto é destruído. Liberam a memória, fecham arquivos, conexões, etc.
- Métodos de acessores (**Get** e **Set**)
- Outros

Exemplo: Classe Pessoa em Java


```java
public class Pessoa {
    private String nome;
    private int idade;
    private String telefone;

    public Pessoa();

    public void setNome(String nome);

    public String getNome();

    public void setIdade(int idade);

    public int getIdade();

    public void setTelefone(String telefone);

    public String getTelefone();
}
```

Exemplo: Classe Pessoa em Java

```java
public class Pessoa {
    private String nome;
    private int idade;
    private String telefone;

    public Pessoa() {
        this.nome = "";
        this.idade = 0;
        this.telefone = "";
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getNome() {
        return nome;
    }
}
```



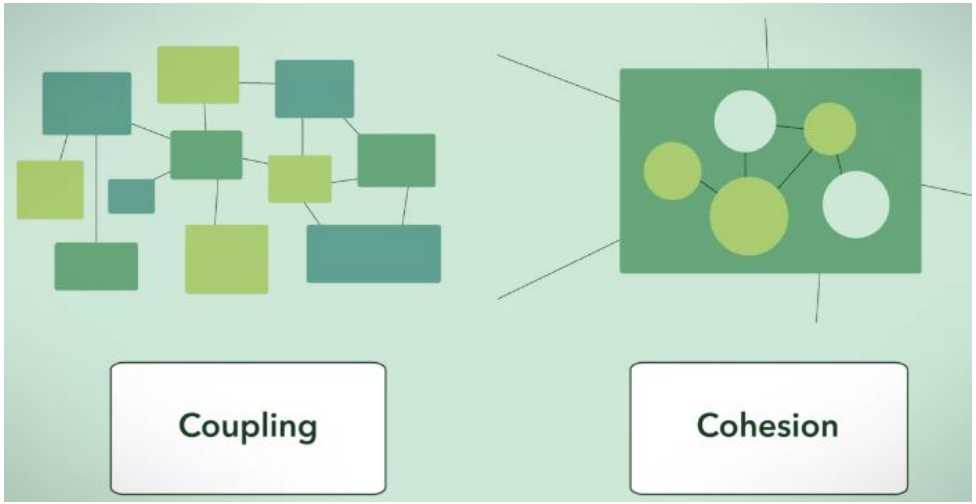
```java
public class Pessoa {
    ...

    public void setIdade(int idade) {
        this.idade = idade;
    }

    public int getIdade() {
        return idade;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getTelefone() {
        return telefone;
    }
}
```

Encapsulamento

- **Controla a visibilidade/acesso à Atributos e Métodos**
  - **Público:** podem ser acessados por qualquer entidade no programa.
  - **Privado:** tem acesso restrito aos membros da própria classe e as classes amigas (friends).
  - **Protegido:** tem acesso restrito aos membros da própria classe, as classes filhas (herança) e as classes amigas (friends).

Encapsulamento

- Diagrama de Acoplamento e Desacoplamento de Classes
  - Acoplamento e Coesão
  - ...



Ref. : https://www.coursera.org/lecture/object-oriented-design/1-3-1-coupling-and-cohesion-q8wGt

Encapsulamento

- **Diagrama de Acoplamento e Desacoplamento de Classes**
  - Acoplamento e Coesão (Grau de dependência e ligação)

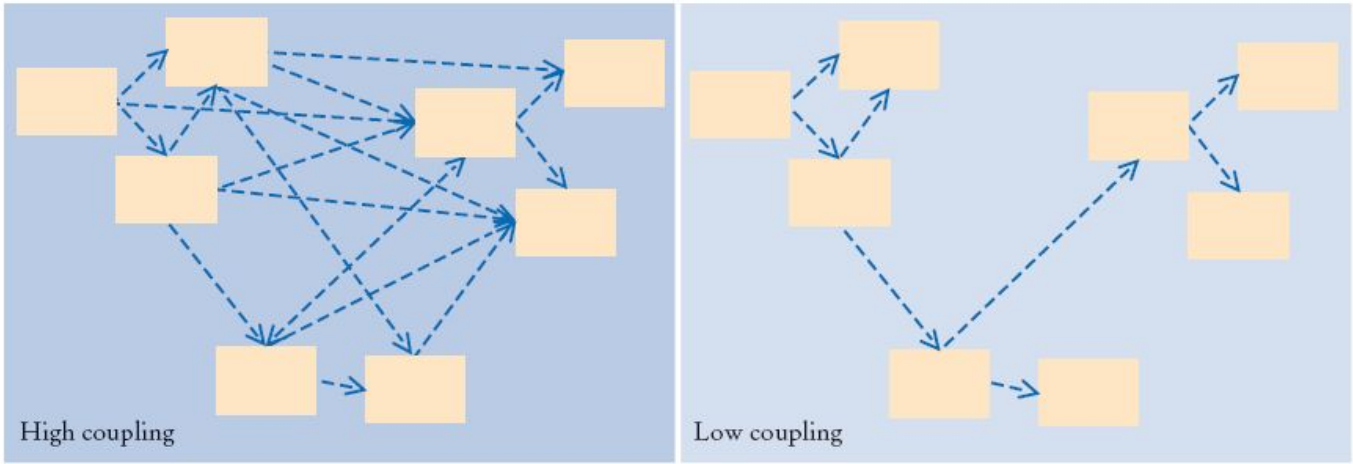


Figure 2 High and Low Coupling Between Classes

Ref. : https://www.iro.umontreal.ca/~pift1025/bigjava/Ch09/ch09.html

Alta (focada em uma única tarefa)

Encapsulamento

- Coesão e Acomplamento

CONCEITO	SIGNIFICA	O IDEAL É...
Coesão	O quanto as partes de uma classe estão relacionadas	**Alta** (focada em uma única tarefa)
Acoplamento	O quanto uma classe depende de outra	**Baixo** (pouca dependência entre classes)

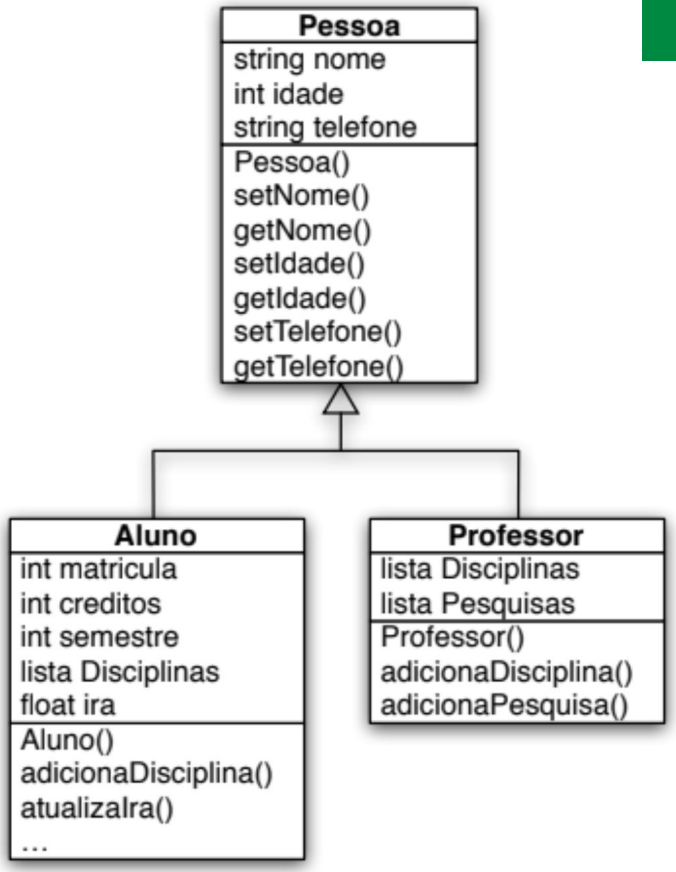
Alta coesão + Baixo acoplamento = Código bonito, limpo e fácil de manter.

Herança

- Princípio que permite classes compartilharem atributos e métodos.
- Viabiliza o reaproveitamento e especialização de classes.
- **Tipo:**
  - Herança Simples
  - Herança Múltipla
  - Herança em múltiplos níveis

Herança

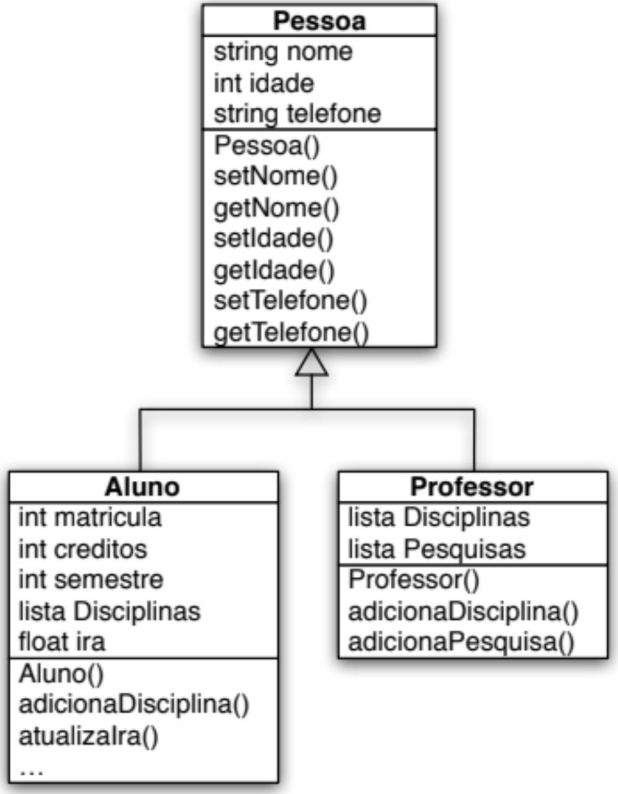
- A Classe "filha" ou “derivada" herda atributos e métodos da classe "pai"



Herança

Aluno.java

```java
public class Aluno extends Pessoa {
    private int matricula;
    private int quantidadeDeCreditos;
    private int semestre;
    private float ira;

    public Aluno() {}
    public Aluno(String nome, int idade, String telefone, int matricula){}
    public void setMatricula(int matricula) {}
    public int getMatricula() {}
    public void setQuantidadeDeCreditos(int creditos) {}
    public int getQuantidadeDeCreditos() {}
    public void setSemestre(int semestre) {}
    public int getSemestre() {}
    public String getSemestreString() {}
    public void setIra(float ira) {}
    public float getIra() {}
    public void imprimirAluno() {}
}
```



Polimorfismo

- É um meio de prover uma interface única para entidades de tipos diferentes.
- Tipos:
  - **Sobrecarga:** quando métodos ou operadores de mesmo nome em uma classe recebem parâmetros diferentes.
  - **Sobrescrita:** quando métodos de classes derivadas possuem mesma assinatura do método da superclasse (classe “pai”) porém funcionam de maneiras distintas.

Polimorfismo - Exemplos

- Sobrecarga de métodos:

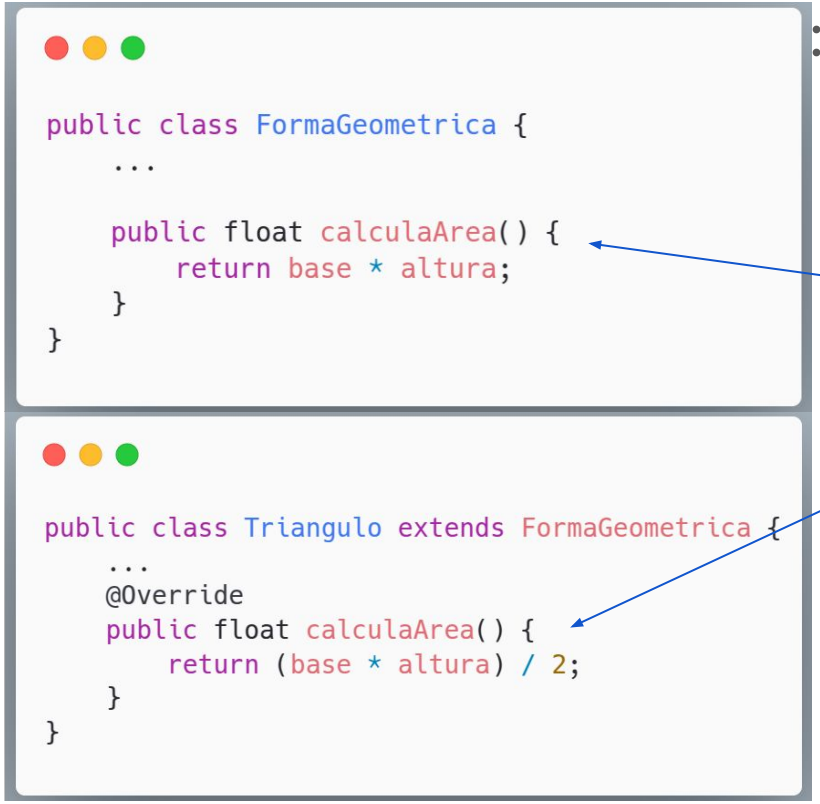
métodos com mesmo nome...

```java
public Pessoa() {
    this.nome = "";
    this.idade = 0;
    this.telefone = "";
}

public Pessoa(String nome, int idade, String telefone) {
    this.nome = nome;
    this.idade = idade;
    this.telefone = telefone;
}
```

```java
public int calculaArea(int base, int altura) {
    return base * altura;
}

public int calculaArea(Double base, Double altura) {
    return base * altura;
}
```

...com parâmetros distintos

Polimorfismo - Exemplos



```java
public class FormaGeometrica {
    ...

    public float calculaArea() {
        return base * altura;
    }
}
```

```java
public class Triangulo extends FormaGeometrica {
    ...
    @Override
    public float calculaArea() {
        return (base * altura) / 2;
    }
}
```

métodos com mesmo assinatura...

mas com implementação distintas.

Polimorfismo - Exemplos

- Sobrecarga de Operadores:

```
int x, y, z;
    z = x + y;

float f1, f2, f3;
    f3 = f1 + f2;

string s1, s2, s3;
    s3 = s1 + s2;

complexo c1, c2, c3 => no formato (a + bi)
    c3 = c1 + c2;
```

Polimorfismo - Exemplos

- Sobrecarga de Operadores - sintaxe

Orientação a Objetos

Dúvidas?