

Programação Orientada a Objetos

Aula: 02/10/2023

Fundamentos de POO – Parte 1

Material elaborado em parceria com os professores Nádia F. F. da Silva, Juliana P. Félix, Guilherme S. Marques e Reinaldo de S. Júnior.

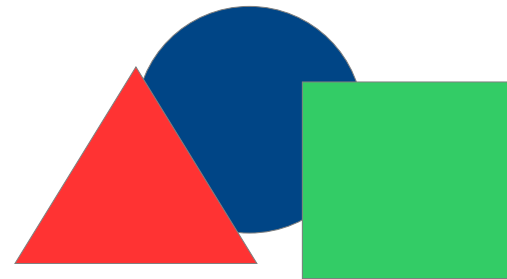
Prof. Dirson Santos de Campos
dirson_campos@ufg.br



Programação Orientada a Objetos

OO

Orientação a Objetos



Importância do Paradigma OO

A Computing Curricula Series Report
2020 December 31

Computing Curricula 2020

CC2020

Paradigms for
Global Computing Education

Importância do Paradigma OO

Computing Curricula: The Overview Report

CC2020 [\[4\]](#): Computing Curricula 2020: Paradigms for Global Computing Education.

1. Computer Engineering
2. Computer Science
3. Cybersecurity
4. Information Systems
5. Information Technology
6. Software Engineering
7. with data science

**Presente em todos os
Paradigmas
do Ensino na
Computação
atualmente no mundo**

Importância do Paradigma OO

Three and two-course sequences for each implementation strategies,

CC

Implemen. strategies	Three-course sequences	Two-course sequences
Imperative first	CS101I. Programming Fundamentals CS102I. The Object-Oriented Paradigm CS103I. Data Structures and Algorithms	CS111I. Introduction to Programming CS112I. Data Abstraction
Objects first	CS101O. Introduction to Object-Oriented Programming CS102O. Objects and Data Abstraction CS103O. Algorithms and Data Structures	CS111O. Object-Oriented Programming CS112O. Object-Oriented Design and Methodology
Functional first	CS111F. Introduction to Functional Programming CS112F. Objects and Algorithms	

A nova versão do Curricula
2023 ainda está em Beta

Curricula 2023

Version Beta

March 2023

The Joint Task Force on Computing Curricula
Association for Computing Machinery

(ACM)

IEEE-Computer Society

(IEEE-CS)

Association for Advancement of Artificial
Intelligence

(AAAI)



Association for
Computing Machinery



IEEE
COMPUTER
SOCIETY



Programação Orientada a Objetos

A **orientação a objetos** é um **paradigma** de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.

A **orientação a objetos** visa um pensamento o mais próximo possível da vida real.

Programação Orientada a Objetos

O que é um Paradigma?

- Um modelo, padrão, estilo, protótipo
 - É a representação de um padrão a ser seguido
 - Formas de abstração do problema
 - Estilo de programação
- Define a forma como um desenvolvedor lida com um problema – tanto na análise como na programação*

Programação Orientada a Objetos

Paradigma Imperativo

-Primeiro faça isso, depois faça aquilo

-uma sequência de comandos que o computador executará, passo a passo, modificando dados e variáveis a fim de chegar ao resultado esperado (Pascal, Cobol, C, etc).

-Derivados : Estruturado ou Procedural

Programação Orientada a Objetos

Paradigma Funcional

–Subdividir o problema em funções matemáticas e resolver cada uma separadamente, pois os resultados encontrados serão utilizados posteriormente.

–Ex: Lisp, R, Erlang

Programação Orientada a Objetos

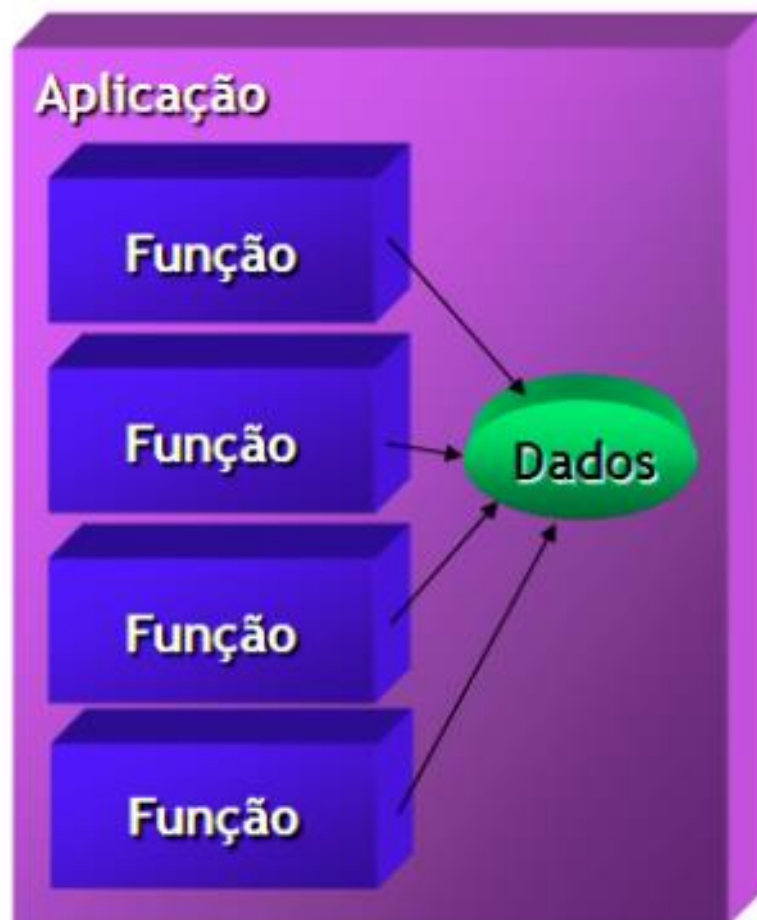
Paradigma Orientado a Objetos

- Um conjunto de **classes** (estruturas definidas) que gera **objetos** (instâncias) e, estes recebem ordens para executar tarefas através de troca de mensagens.
- O Software como um todo se forma com a interação entre os objetos através das trocas de mensagens.
- Vantagens: manutenção, flexibilidade e reutilização

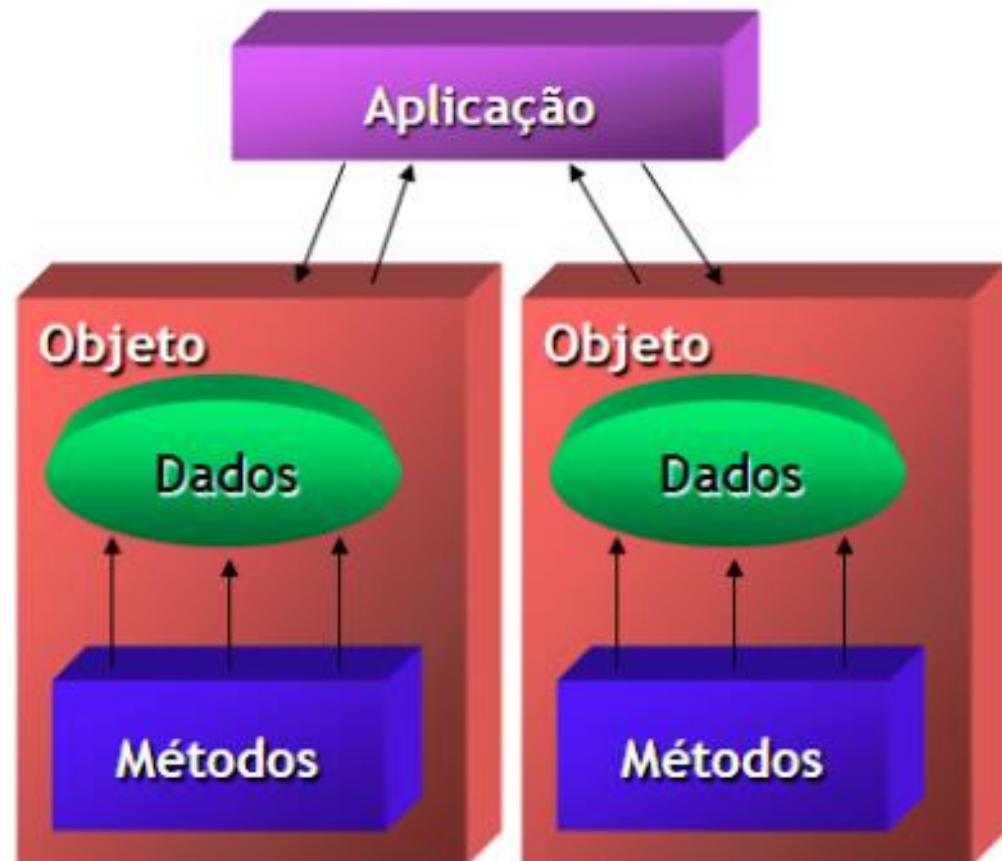
Programação Orientada a Objetos

Diferença entre os principais paradigmas

Estruturada



Orientação a Objetos



Programação Orientada a Objetos

Diferença entre os principais Paradigmas

Sequência de Passos x Objetos

•Programação como sequência de passos

–Paradigma estruturado (tradicional) onde um problema é resolvido a partir de um início e fim bem definidos e eventualmente dividido em sub-rotinas;

•Programação utilizando objetos

–O paradigma da orientação a objetos considera que os dados a serem processados e os mecanismos de processamento dos dados (operações) devem ser considerados em conjunto;

Programação Orientada a Objetos

Um pouco de história sobre POO

(Programação Orientada a Objetos)

- (1967) SIMULA - 1ª Linguagem Orientada a Objetos;
- Década de 70 surge a linguagem SmallTalk (considerada puramente orientada a objetos);
- Década de 80 ... Rápida evolução ... Surgimento de Ada e C++
- Década de 90 ... Java

Programação Orientada a Objetos

..mais história

- O uso da Tecnologia de Objetos como metodologia básica para desenvolvimento de sistemas (abrangendo todo o ciclo ... desde análise até o código) é uma prática que passou a ser difundida na década de 80 com a publicação dos trabalhos do pesquisador Grady Booch.
- A tecnologia de objetos veio para ficar!
- Seus conceitos e técnicas imprimem maior qualidade, produtividade e profissionalismo na construção de software

Linguagem Java atualmente

Java™

Paradigm	Multi-paradigm: generic, object-oriented (class-based), functional, imperative, reflective, concurrent
Designed by	James Gosling
Developer	Sun Microsystems
First appeared	May 23, 1995; 27 years ago ^[1]
Stable release	Java SE 20 ^[2]  / 21 March 2023; 34 days ago
Typing discipline	Static, strong, safe, nominative, manifest
Filename extensions	.java, .class, .jar, .jmod
Website	oracle.com/java/  , java

Linguagem Python atualmente



python™

Paradigma

Multiparadigma:
orientada a objetos •
imperativa • funcional

Surgido em

20 de fevereiro de 1991
(32 anos)^[1]

Última versão

3.11.3 (5 de abril de 2023; há
22 dias^[2])

Criado por

Guido van Rossum^[1]

Estilo de tipagem

dinâmica • forte • *duck* •
gradual (desde a versão 3.5)

**Principais
implementações**

CPython • IronPython •
Jython • PyPy • Stackless

Influenciada por

ABC^[3] • ALGOL 68 • C^[3] •
Haskell • Icon • Java • Lisp
• Modula-3^[3] • Perl •
Smalltalk

Influenciou

Boo • CoffeeScript • D •
Fantom • GDScript • Go •
Groovy • JavaScript • Julia
• Nim • Py • Ruby •
Squirrel • Swift

Licença:

Python Software
Foundation License^[4]

**Extensão do
arquivo:**

.py • .pyc • .pyd • .pyo •
.pyw • .pyz

Página oficial

www.python.org 

Programação Orientada a Objetos

Orientação a objetos é necessária?

- Nem sempre ...

- Há situações onde o modelo de uma tarefa a ser executada é tão simples que a criação de uma classe para representá-lo torna o problema mais complicado e confuso ...

- Exemplo: calcular as raízes de uma equação de segundo grau

$$x = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}$$

Fórmula de Bhaskara

Programação Orientada a Objetos

```
#include <stdio.h>
#include <math.h>

double calculaX1(int a, int b, int c) {
    double delta = (b*b)-4*(a)*(c);
    return -b + sqrt(delta) / 2*a;
}

double calculaX2(int a, int b, int c) {
    double delta = (b*b)-4*(a)*(c);
    return -b - sqrt(delta) / 2*a;
}

int main(int argc, char** argv) {
    int a, b, c;
    printf("Digite a, b e c ... \n");
    scanf("%d",&a);
    scanf("%d",&b);
    scanf("%d",&c);
    printf("X1 = %f", calculaX1(a, b, c));
    printf("X2 = %f", calculaX2(a, b, c));
    return 0;
}
```

Código estruturado em C

```
public class Bhaskara {
    private int a, b, c;

    Bhaskara(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public double calculaX1() {
        double delta = (b*b)-4*(a)*(c);
        return -b + Math.sqrt(delta) / 2*a;
    }

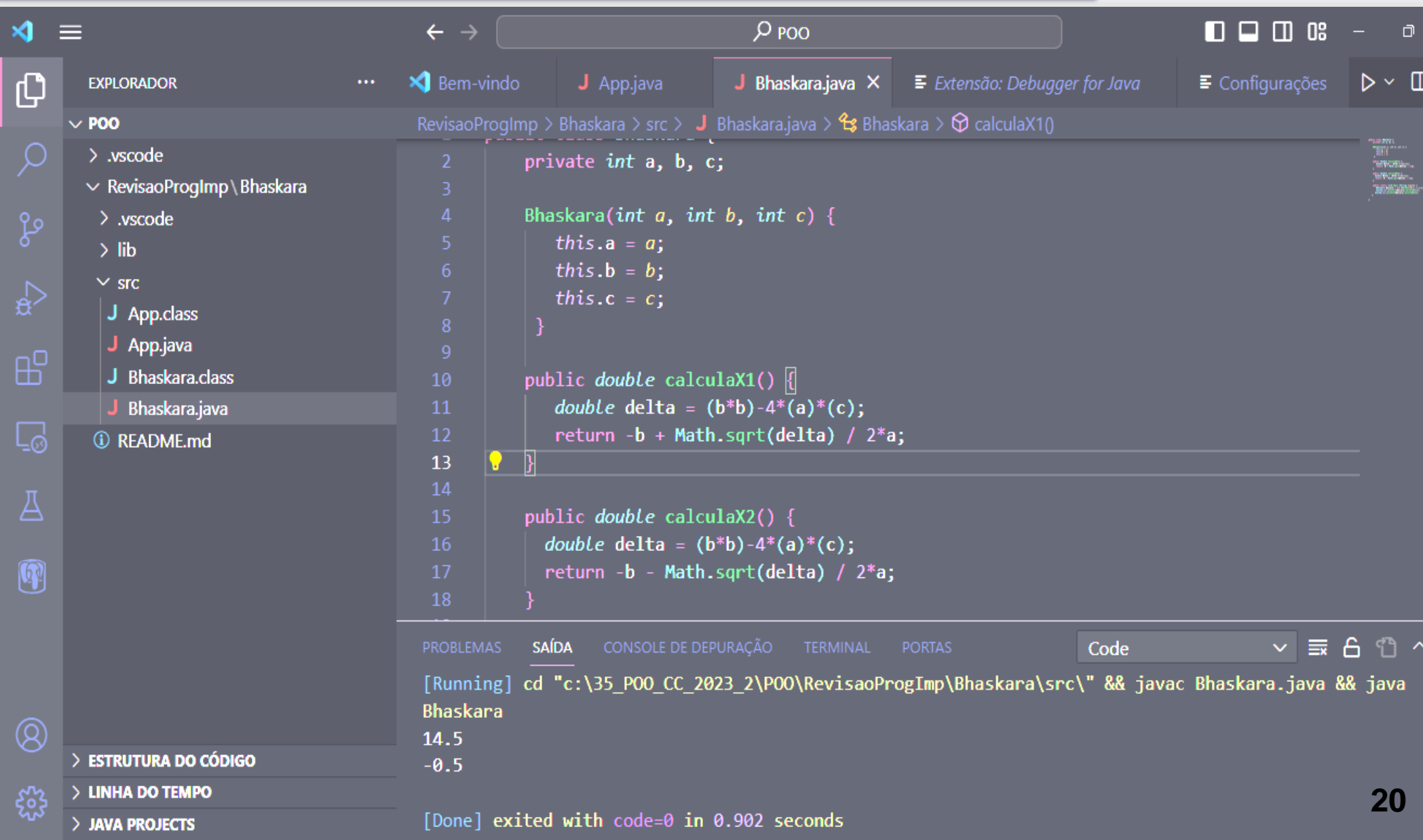
    public double calculaX2() {
        double delta = (b*b)-4*(a)*(c);
        return -b - Math.sqrt(delta) / 2*a;
    }

    public static void main (String args[]) {
        Bhaskara bascara = new Bhaskara(3,-7,2);
        System.out.println(bascara.calculaX1());
        System.out.println(bascara.calculaX2());
    }
}
```

Código orientado a objetos em Java

Programação Orientada a Objetos

Executando no VS Code



The screenshot displays the Visual Studio Code interface with the following components:

- EXPLORADOR (Explorer):** Shows the project structure. The 'src' folder is expanded, showing 'App.class', 'App.java', 'Bhaskara.class', 'Bhaskara.java', and 'README.md'. 'Bhaskara.java' is selected.
- EDITOR:** Displays the code for 'Bhaskara.java'. The code defines a class 'Bhaskara' with private attributes 'a', 'b', and 'c', a constructor, and two methods: 'calculaX1()' and 'calculaX2()'. A lightbulb icon indicates a suggestion for closing the curly brace on line 13.
- SAÍDA (Output):** Shows the execution results. The command executed is `cd "c:\35_P00_CC_2023_2\P00\RevisaoProgImp\Bhaskara\src\" && javac Bhaskara.java && java Bhaskara`. The output shows the values 14.5 and -0.5.
- STATUS BAR:** Indicates the current state of the editor and the active extension, 'Extensão: Debugger for Java'.

```
2 private int a, b, c;
3
4 Bhaskara(int a, int b, int c) {
5     this.a = a;
6     this.b = b;
7     this.c = c;
8 }
9
10 public double calculaX1() {
11     double delta = (b*b)-4*(a)*(c);
12     return -b + Math.sqrt(delta) / 2*a;
13 }
14
15 public double calculaX2() {
16     double delta = (b*b)-4*(a)*(c);
17     return -b - Math.sqrt(delta) / 2*a;
18 }
```

[Running] cd "c:\35_P00_CC_2023_2\P00\RevisaoProgImp\Bhaskara\src\" && javac Bhaskara.java && java Bhaskara

14.5
-0.5

[Done] exited with code=0 in 0.902 seconds

Programação Orientada a Objetos

```
import java.util.Scanner;

public class CalculoIMC {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Entre o nome da pessoa:");
        String nome = sc.nextLine();
        System.out.println("Entre a idade:");
        int idade = Integer.parseInt(sc.nextLine());
        System.out.println("Entre a altura:");
        double altura = Double.parseDouble(sc.nextLine());
        System.out.println("Entre a peso:");
        double peso = Double.parseDouble(sc.nextLine());

        double imc = peso / (altura*altura);

        System.out.println("Nome: " + nome + " - Idade: " + idade + " -
        Altura: " + altura + " - Peso: " + peso + " - IMC: " + imc);
    }
}
```

Código estruturado em Java

```
public class PessoaPessoaIdeal {
    String nome;
    int idade;
    double peso;
    double altura;

    Pessoa(String nome, int idade, double peso, double altura){
        this.nome = nome;
        this.idade = idade;
        this.peso = peso;
        this.altura = altura;
    }

    public double IMC() {
        return peso/(altura * altura);
    }

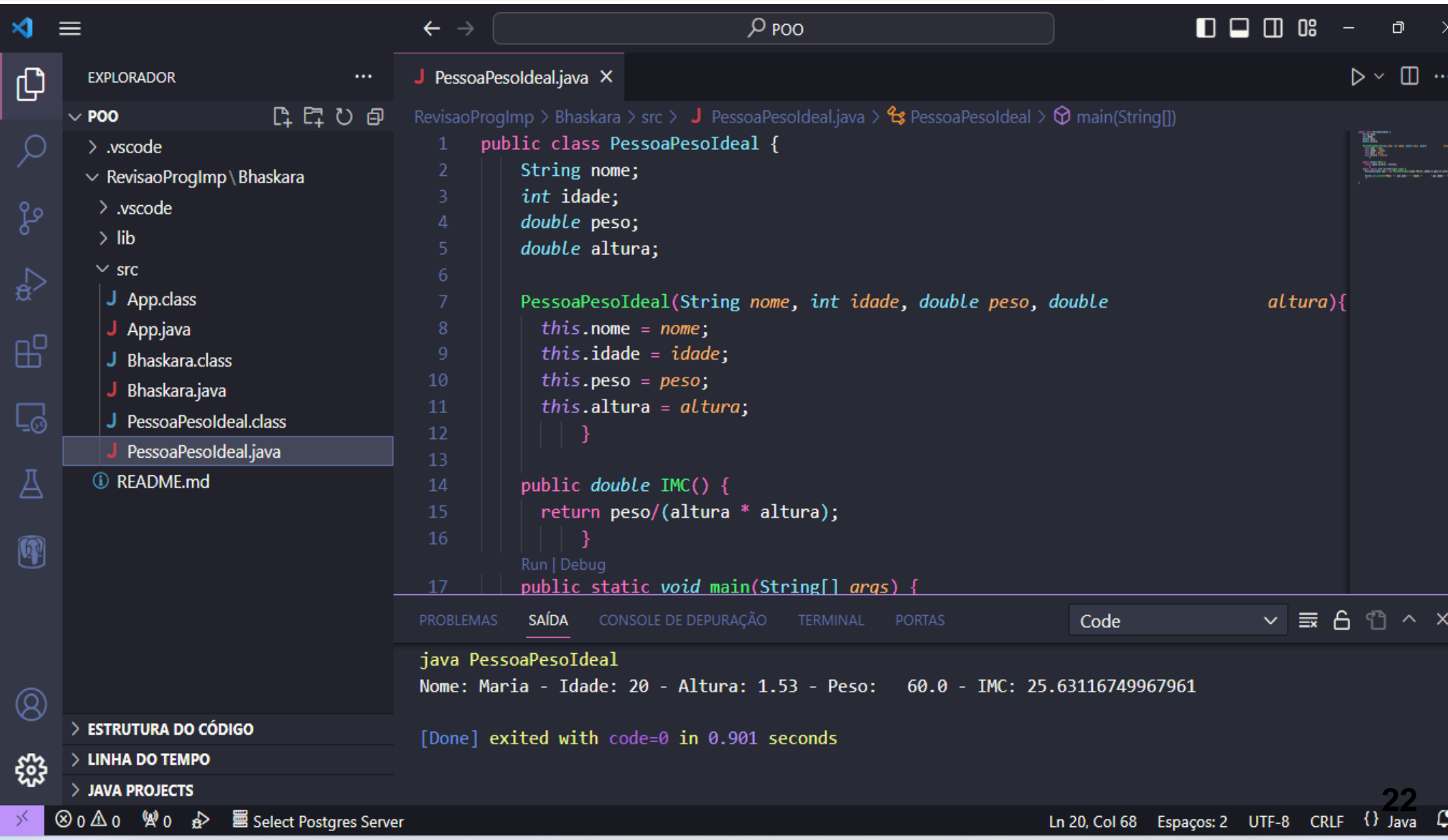
    public static void main(String[] args) {
        Pessoa pes = new Pessoa("Maria",20,60,1.53);

        System.out.println("Nome: " + pes.nome + " - Idade: " + pes.idade + " - Altura: " +
        pes.altura + " - Peso: " + pes.peso + " - IMC: " + pes.IMC());
    }
}
```

Código orientado a objetos em Java

Programação Orientada a Objetos

Executando no VS Code



Programação Orientada a Objetos

Orientação a objetos é necessária?

• Em muitas situações ...

• Imagine uma aplicação “mundo real” repleta de janelas que apresentam as mesmas funcionalidades e se compõem de uma infinidade de outros controles gráficos que se repetem ao longo da aplicação (botões, caixas de texto, janelas de diálogo...);

• Imagine simular o movimento de espermatozoides a procura de um óvulo utilizando a programação estruturada normal ... De forma que cada célula (dentro as milhões) tenha atributos diferentes (velocidade, tamanho, agilidade...)



Programação Orientada a Objetos

Vantagens da Orientação a Objetos:

- Ajuda na organização e resolve problemas da programação procedural/estruturada
- Minimiza o código - escreve-se menos
- Em alguns casos facilita o entendimento
- Concentra as responsabilidades nos pontos certos
- Diminui a responsabilidade dos programadores

Programação Orientada a Objetos

Aplicação

–Resolução de Problemas

–Termos semelhantes que serão muito utilizados :

- .Problema
- .Domínio
- .Contexto
- .Cenário
- .Situação

Programação Orientada a Objetos

Exemplo:

Cenário: inscrição de um aluno em uma disciplina

- Como seria no paradigma estrutural?
- Como seria no paradigma orientado a objetos?

Programação Orientada a Objetos

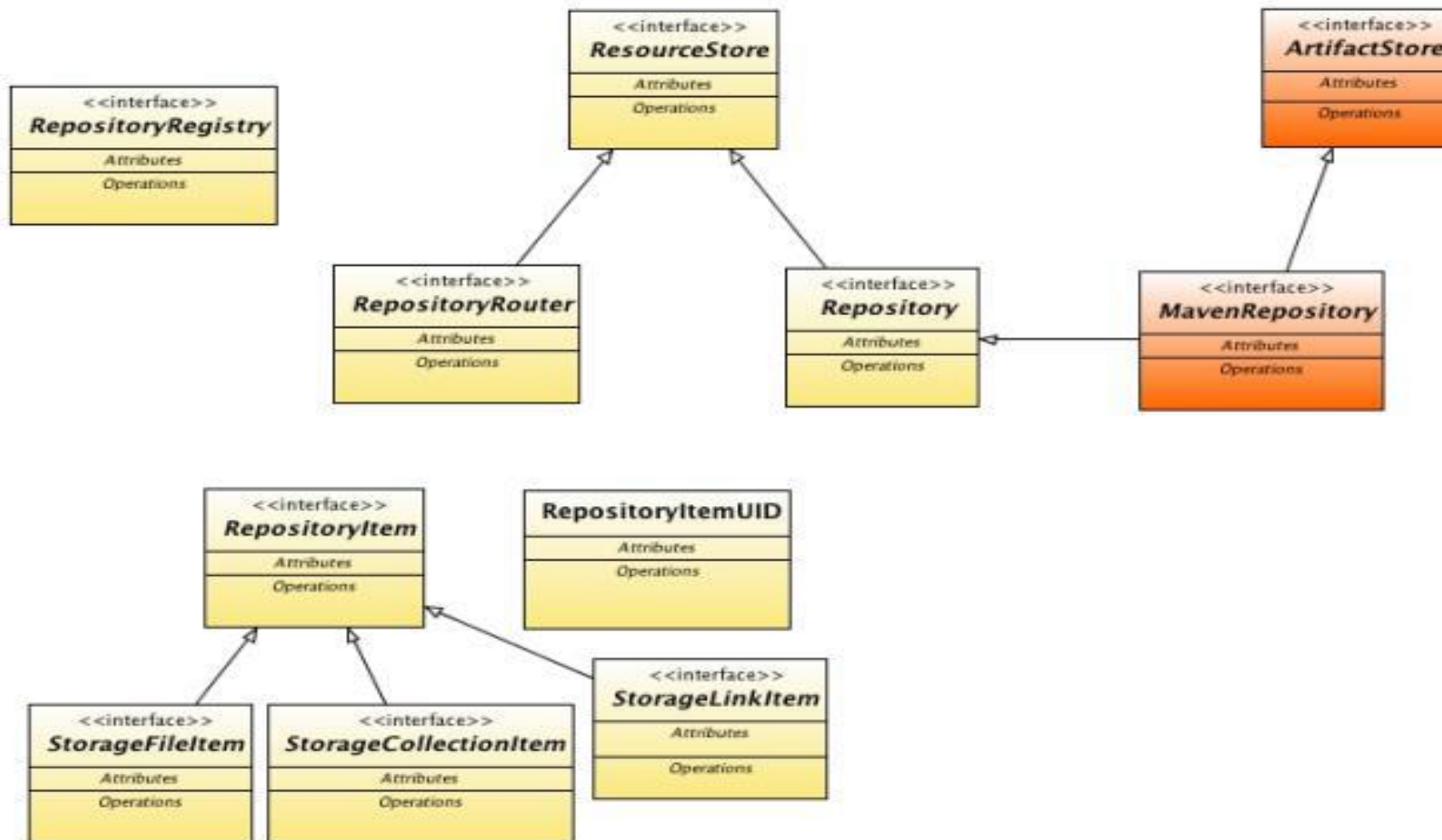
Exemplo:

Cenário: Saque de uma conta corrente de um cliente

- Como seria no paradigma estrutural?
- Como seria no paradigma orientado a objetos?

Programação Orientada a Objetos

POO - Conceitos Fundamentais



Programação Orientada a Objetos

Uma linguagem orientada a objetos deve oferecer:

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

De acordo com...

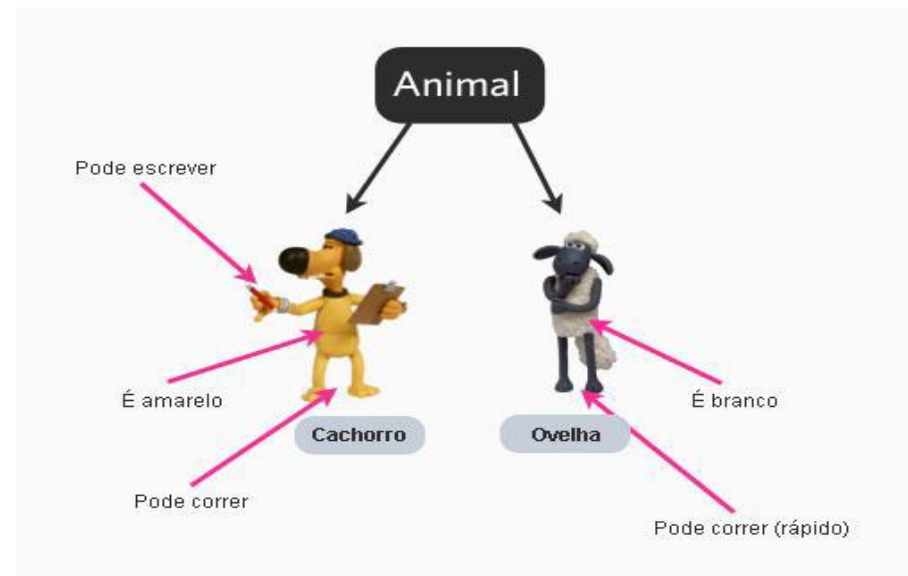
**Grady Booch: Coming of Age in an Object-Oriented World.
IEEE Software 11(6): 33-41 (1994).**

Programação Orientada a Objetos

ABSTRAÇÃO

CLASSES X OBJETOS

- Habilidade de se concentrar nos aspectos essenciais de um **contexto** qualquer, ignorando características menos importantes ou acidentais;
- Imaginar quais são os **objetos** e o que eles vão realizar no sistema



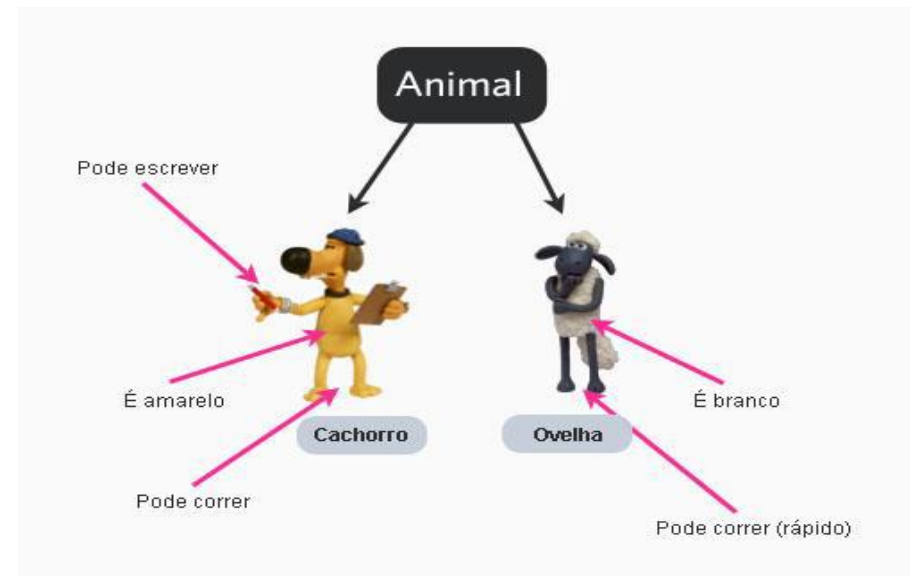
Programação Orientada a Objetos

ABSTRAÇÃO

CLASSES X OBJETOS

• Na POO, uma **classe** é uma abstração de entidades existentes no **domínio** do sistema de software;

• Os **objetos** são instâncias das classes e cada um tem sua identidade, propriedades e comportamentos.

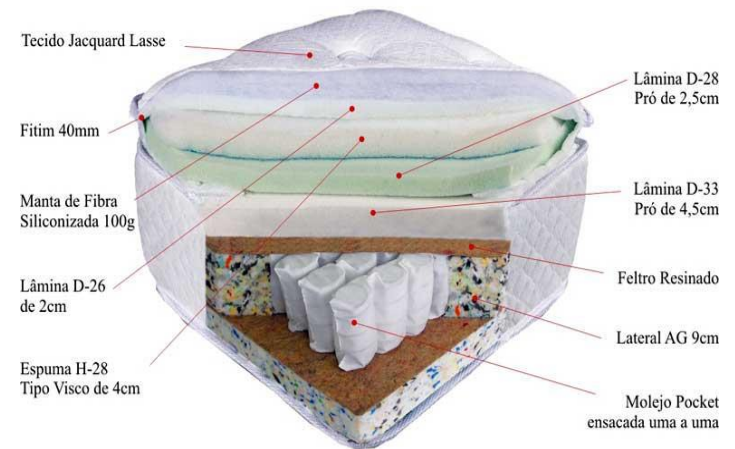


Programação Orientada a Objetos

ENCAPSULAMENTO

Ocultação de Informações

- Adicionam **segurança** à aplicação impedindo o acesso direto ao estado de um objeto;
- Consiste na separação dos aspectos internos e externos de um objeto;
- Permite ignorar os detalhes de implementação (de como as coisas funcionam internamente) permitindo ao desenvolvedor idealizar seu trabalho em um nível mais alto de abstração;

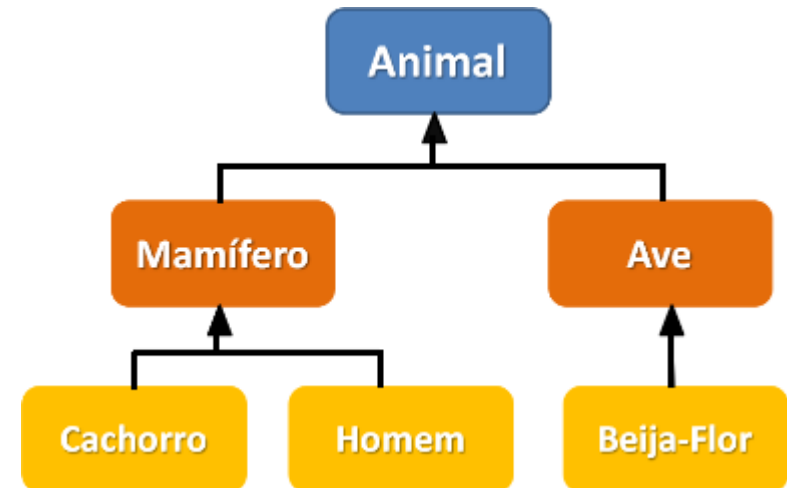


Programação Orientada a Objetos

HERANÇA

Reutilização de Código

- É o mecanismo de reaproveitamento e **reutilização de código**;
- Permite que elementos mais específicos incorporem a estrutura e o comportamento de elementos mais genéricos;

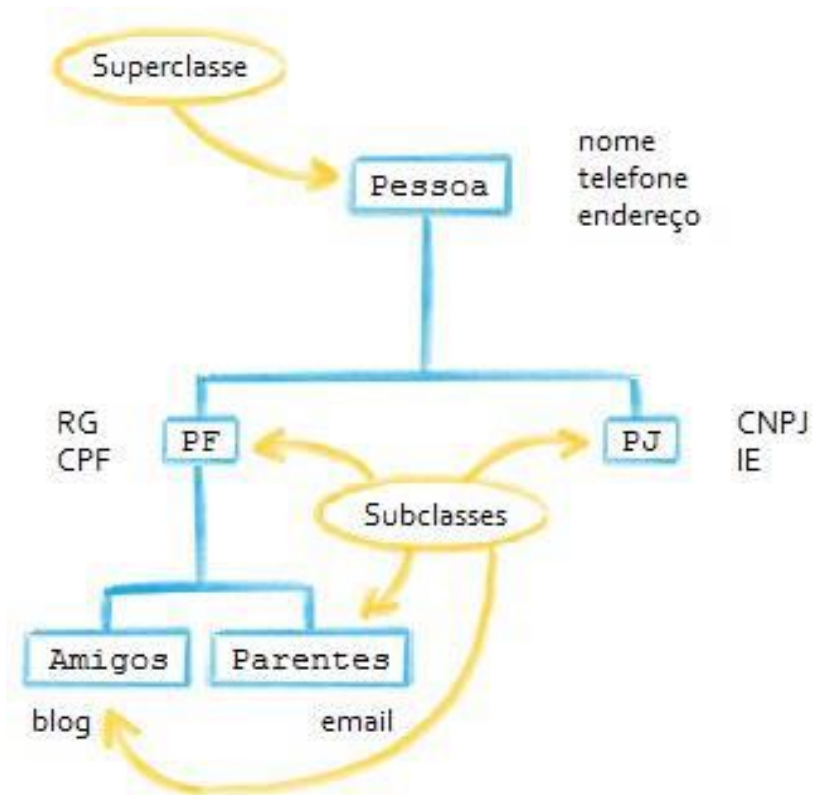


Programação Orientada a Objetos

HERANÇA

Reutilização de Código

• Podem ser diretas e indiretas e as estruturas são formadas para que algumas **Superclasses** agrupem características e comportamentos mais genéricos que possam ser reaproveitados por **Subclasses** mais específicas



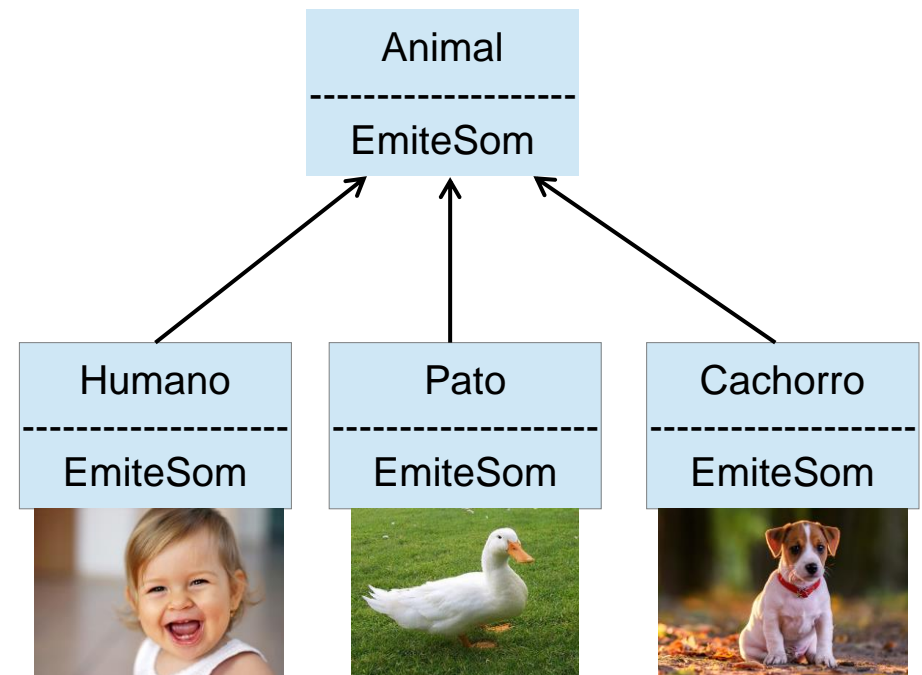
Programação Orientada a Objetos

POLIMORFISMO

Várias Formas

• Permite a um mesmo objeto se manifestar de diferentes formas;

• Implementações diferentes para uma mesma operação.





Abstração em POO

Abstração

- Habilidade de se concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais;

Abstração

- Na Orientação a Objetos, uma **classe** é uma abstração de entidades existentes no domínio do sistema de software;
 - Recordando, a **orientação a objetos** é um modelo de análise, projeto e programação de sistemas de software baseado na **composição** e **interação** entre diversas unidades de software chamadas de objetos, criados a partir das classes.

Classificando em diferentes domínios

Domínio: Sala de Aula

O que podemos identificar neste ambiente?

- Alunos;
- Carteiras;
- Quadros;
- Professores;
- Pessoas;



Classificando em diferentes domínios

Domínio: Sala de Aula

Todos os Alunos são iguais?

Não, mas..

Cada aluno é um objeto diferente do cenário, mas todos são *considerados/classificados* como **Alunos**. Logo, **Aluno** é uma **Classe**.

Características de um Aluno (atributos):

- Nome
- Cor do cabelo
- Idade
- Sexo
- Altura
- Peso



Classificando em diferentes domínios

Domínio: Roupas e Acessórios

O que podemos identificar neste caso?

- Calças;
- Casacos;
- Camisas;
- Saias;
- Bolsas;
- Sapatos;
- Vestidos



Classificando em diferentes domínios

Domínio: Roupas e Acessórios

Todas as Camisas são iguais?

Não, mas..

Cada camisa é um objeto diferente do cenário, mas todas são *consideradas/classificadas* como **Camisa**.

Características de uma Camisa (atributos):

- Cor;
- Tamanho;
- Modelo;
- Marca



Exemplos de Classe

Domínio: Exemplo de uma
Classe **Pessoa**

Exemplos de Características /
Atributos:

- Nome;
- Idade;
- Sexo;
- Nacionalidade;
- Raça;
- etc...

Exemplos de Comportamentos /
Métodos:

- Respirar;
- Dormir;
- Andar;
- Comer;
- etc...



Objeto

Objeto é uma *instância* da **classe**;

É uma representação real da classe que ocupa espaço na memória e consome recursos do computador;

A classe é a descrição de um *tipo de objeto*;

Uma classe pode ser considerada uma “fábrica” para criação de objetos;

Atributos & Métodos

"A classe é uma coleção de dados(atributos) e operações(métodos) que manipulam tais dados."

Atributos

- São os dados (simples ou compostos) que caracterizam os objetos daquela classe;
- São armazenados em variáveis;
- Constituem o *estado* do objeto.

Métodos

- São as operações (procedimentos ou funções) que manipulam os dados;
- Através deles, realiza-se a *troca de mensagens* com o objeto.

Exemplos de Classe

Domínio: Exemplo de uma Classe

Cachorro

Exemplos de Características /
Atributos:

- ...

Exemplos de Comportamentos /
Métodos:

- ...



Exemplos de Classe

**Domínio: Exemplo de uma
Classe Moto**

Exemplos de Características
/ Atributos:

- ...

Exemplos de
Comportamentos / Métodos:

- ...





Diagrama UML de Classes

Diagrama de Classes UML

- O padrão UML - Linguagem de Modelagem Unificada foi criado para modelar diferentes aspectos de um sistema.
- Dentre seus diagramas, o **diagrama de classes** é o mais usado por engenheiros de software para documentar arquiteturas de software, porque descrevem o que deve estar presente no sistema a ser modelado.

Codificando uma classe a partir de um diagrama

Código fonte:
Pessoa.java

```
public class Pessoa {  
    String nome;  
    int idade;  
    String cpf;  
    char sexo;  
    //  
    public void andar(int metros){  
        // ...  
    }  
    public void comer(String comida){  
        // ...  
    }  
    public void acordar(){  
        // ...  
    }  
    public String falar(){  
        // ...  
        return "";  
    }  
}
```

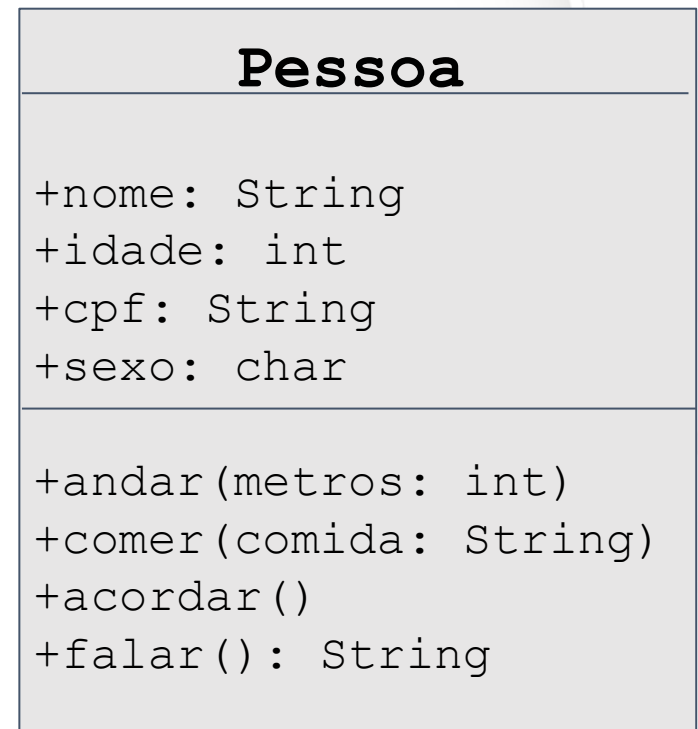


Diagrama de Classes UML

Lista de Atributos

O nome antecede o tipo, separado por ":"

Lista de Métodos

O nome antecede a lista de parâmetros, delimitada por (). Parâmetros e retorno também possuem nome e tipo, separados por ":". Retorno void pode ser ignorado.

Nome da Classe

