

# Padrões de Projeto

## APS

Prof. Igor Avila Pereira  
igor.pereira@riogrande.ifrs.edu.br

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)  
Campus Rio Grande  
Divisão de Computação

# Agenda

- 1 Revisão: Conceitos de O.O

## Revisão: Conceitos de O.O

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

double bascaraX1(int a, int b, int c) {
    double delta = (b*b)-4*(a)*(c);
    return -b + sqrt(delta) / 2*a;
}

double bascaraX2(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", bascaraX1(a, b, c));
    printf("X2 = %f", bascaraX2(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 solveX1() {
        double delta = (b*b)-4*(a)*(c);
        return -b + Math.sqrt(delta) / 2*a;
    }

    public double solveX2() {
        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.solveX1());
        System.out.println(bascara.solveX2());
    }
}
```

Código orientado a objetos em Java

Figura : Paradigma Estruturado x Paradigma O.O

## Revisão: Conceitos de O.O

- Orientação a objetos vai te ajudar em muito em se organizar e escrever menos
- Além de concentrar as responsabilidades nos pontos certos, flexibilizando sua aplicação, encapsulando a lógica de negócios.

## Revisão: Conceitos de O.O

Qual o resultado desse código??

```
1  class TestaReferencias {
2      public static void main(String args[]) {
3          Conta c1 = new Conta();
4          c1.deposita(100);
5
6          Conta c2 = c1; // linha importante!
7          c2.deposita(200);
8
9          System.out.println(c1.saldo);
10         System.out.println(c2.saldo);
11     }
12 }
```

Quando declaramos uma variável para associar a um objeto, na verdade, essa variável não guarda o objeto, e sim uma maneira de acessá-lo, chamada de referência.

## Revisão: Conceitos de O.O

Qual o resultado desse código??

```
1 public static void main(String args[]) {  
2     Conta c1 = new Conta();  
3     c1.dono = "Duke";  
4     c1.saldo = 227;  
5  
6     Conta c2 = new Conta();  
7     c2.dono = "Duke";  
8     c2.saldo = 227;  
9  
10    if (c1 == c2) {  
11        System.out.println("Contas iguais");  
12    }  
13 }
```

## Revisão: Conceitos de O.O

Um sistema orientado a objetos é um grande conjunto de classes que vai se comunicar, delegando responsabilidades para quem for mais apto a realizar determinada tarefa.

Dizemos que esses objetos colaboram, trocando mensagens entre si. Por isso acabamos tendo muitas classes em nosso sistema, e elas costumam ter um tamanho relativamente curto.

# Revisão: Conceitos de O.O

- **Classe**

- agrupamento de **atributos** e **comportamentos** em comum;
- A classe é a descrição de um tipo de objeto;

- **Objeto**

- instâncias das classes
- cada instância (objeto) pode ter valores exclusivos para suas variáveis

- **Visibilidade**

- **Public**: membro acessível para qualquer classe
- **Private**: membro acessível apenas para a própria classe
- **Protected**: membro acessível para as subclasses
- **Package**: membro acessível apenas dentro do pacote



## Revisão: Conceitos de O.O

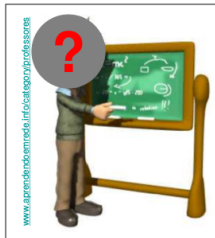
Em uma universidade existem inúmeros professores, cada um com uma matrícula SIAPE, um nome, uma área de conhecimento, uma titulação, etc..

Cada um é um **objeto**.

Porém é possível perceber que todos os professores tem o mesmo conjunto de informações relevantes ao sistema e que além disso todos desempenham as mesmas ações.

Esta observação nos leva a identificar/definir a **classe professor**.

# Revisão: Conceitos de O.O



**Classe professor**



**O professor Raimundo**



**O professor Girafales**

## Revisão: Conceitos de O.O

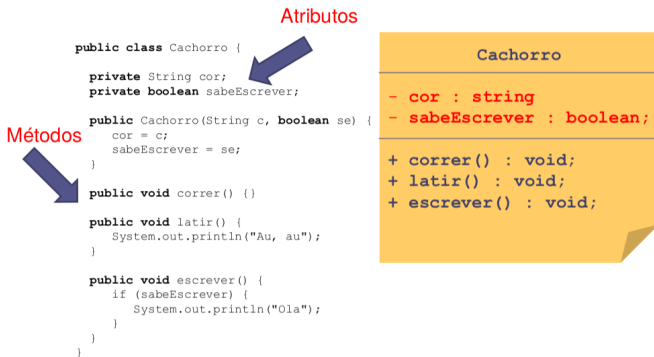


Figura : Exemplo de Classe Java

Dentro da classe, também declararemos o que cada classe faz e como isto é feito - os comportamentos que cada classe tem, isto é, o que ela faz.

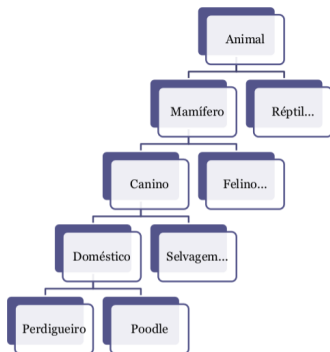
## Construtor

- Java define um construtor *default*
- A partir do momento que você definir um construtor, o **default** não é mais fornecido
- Possibilita ou obriga o usuário de uma classe a passar argumentos para o objeto durante o processo de criação do mesmo

# Revisão: Conceitos de O.O

## Herança

- uma classe herda tudo (atributos, métodos e relacionamentos) de outra;
- Classe filha (subclasse) é uma extensão da classe mãe (superclasse)
- Palavra chave *extends*



# Revisão: Conceitos de O.O

## Polimorfismo

- Quando herdamos um método, podemos alterar seu comportamento
- Podemos reescrever (sobrescrever, override) este método

**Exemplo:** Em cada tipo de veículo a operação de embarque é diferenciada



## Interfaces

- Podemos criar um **contrato** que define tudo o que uma classe deve fazer se quiser ter um determinado status;
- Uma interface pode definir uma série de métodos, mas nunca conter a implementação deles;
- Ela só expõe o que o objeto deve fazer, e não como ele faz, nem o que ele tem;
- Como ele faz vai ser definido em uma implementação dessa interface

## Revisão: Conceitos de O.O

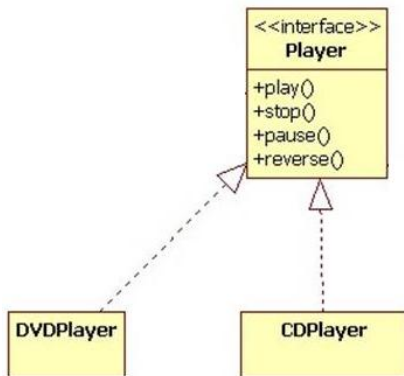


Figura : Exemplo de Interfaces



## Classe Abstrata

- Classe abstrata não pode ser instanciada
- Garante que um método seja sempre escrito pelas classes filhas;
- Caso não reescrevam, um erro de compilação ocorrerá
- Um método abstrato obriga a classe em que ele se encontra ser abstrata

## Revisão: Conceitos O.O

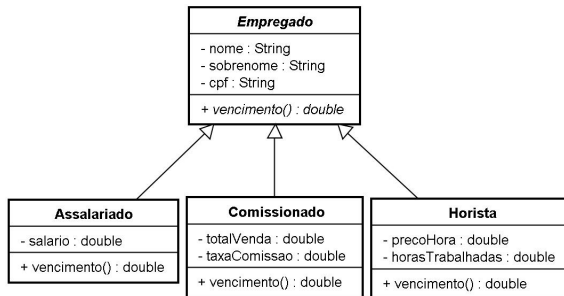


Figura : Classe Abstrata

## Revisão: Conceitos de O.O

- **Exemplo:**

- Classe Funcionário é uma classe que apenas idealiza um tipo, define apenas um esboço

```
abstract class Funcionario {  
  
    protected double salario;  
  
    public double getBonificacao() {  
        return this.salario * 1.2;  
    }  
  
    // outros atributos e métodos comuns a todos Funcionarios  
  
}
```

Imagine que você tenha sido contratado para modelar e informatizar a comanda de um restaurante;

- Que tipo de informações seriam pertinentes armazenar sobre a comanda?
- Que tipo de operações a comanda deveria oferecer a seus utilizadores?



# Padrões de Projeto

## APS

Prof. Igor Avila Pereira  
igor.pereira@riogrande.ifrs.edu.br

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)  
Campus Rio Grande  
Divisão de Computação