

FIA/P GRADUAÇÃO



DOMAIN DRIVEN DESIGN

Prof. Me. Thiago T. I. Yamamoto

#10 – POLIMORFISMO



TRAJETÓRIA



- ✓ Orientação a Objetos
- ✓ Introdução ao Java
- ✓ IDE e Tipos de Dados
- ✓ Classes, atributos e métodos
- ✓ Encapsulamento
- ✓ Construtores
- ✓ Conversões e Tomada de decisões
- ✓ Manipulação de Strings
- ✓ Herança
- ✓ Polimorfismo

#10 - AGENDA

- Polimorfismo
- Sobrecarga de métodos
- Sobrescrita de métodos



- A palavra **polimorfismo** significa:
 - Qualidade ou estado de ser capaz de **assumir diferentes formas**, (dicionário Houaiss);
- No contexto da OO, **polimorfismo** significa ter **múltiplos comportamentos**;
- A capacidade **polimórfica** decorre diretamente do mecanismo de **herança**;
- Uma operação polimórfica resulta em **diferentes ações** dependendo do **objeto** que está sendo **referenciado**;

- Um **recurso** usual em **programação OO** é o uso de **sobrecarga de métodos**;
- **Sobrecarregar** um método significa **prover mais de uma versão de um mesmo método**;
- As **versões devem**, necessariamente, **conter parâmetros diferentes**, seja no tipo ou no número desses parâmetros (o tipo de retorno é indiferente);

Ninja
~ mover(x : double) : void ~ mover(x : double, y : double) : double ~ mover(x : double, y : double, velocidade : int) : void ~ mover(destino : String) : int ~ mover(inimigoMaisProximo : Inimigo) : Inimigo

**Métodos com
mesmo nome, mas
com parâmetros
diferentes**

```
public class Ninja {
```

```
    void mover(double x) {  
        //..  
    }
```

```
    double mover(double x, double y) {  
        //..  
    }
```

```
    void mover(double x, double y, int velocidade) {  
        //..  
    }
```

```
    int mover(String destino) {  
        //..  
    }
```

```
    Inimigo mover(Inimigo inimigoMaisProximo) {  
        //..  
    }
```

```
}
```

- Em **Java**, a palavra reservada **this** significa a **referência ao próprio objeto**;
- Através da palavra reservada **this** é possível acessar **atributos, métodos e construtores** do objeto da classe em questão;
- Na maioria das vezes a palavra **this** é utilizada em **duas situações**:
 - Quando houver **duas variáveis com o mesmo nome** numa mesma classe, **uma pertencendo à classe e outra pertencendo a algum dos métodos** da classe. Nesse caso, apenas esse método específico requer o uso do **this**, se quiser fazer **referência ao campo da classe**;
 - Quando uma classe passar uma **referência de si própria** a um método.


```
public class Carro {
```

```
    String modelo;
```

```
    float motor;
```

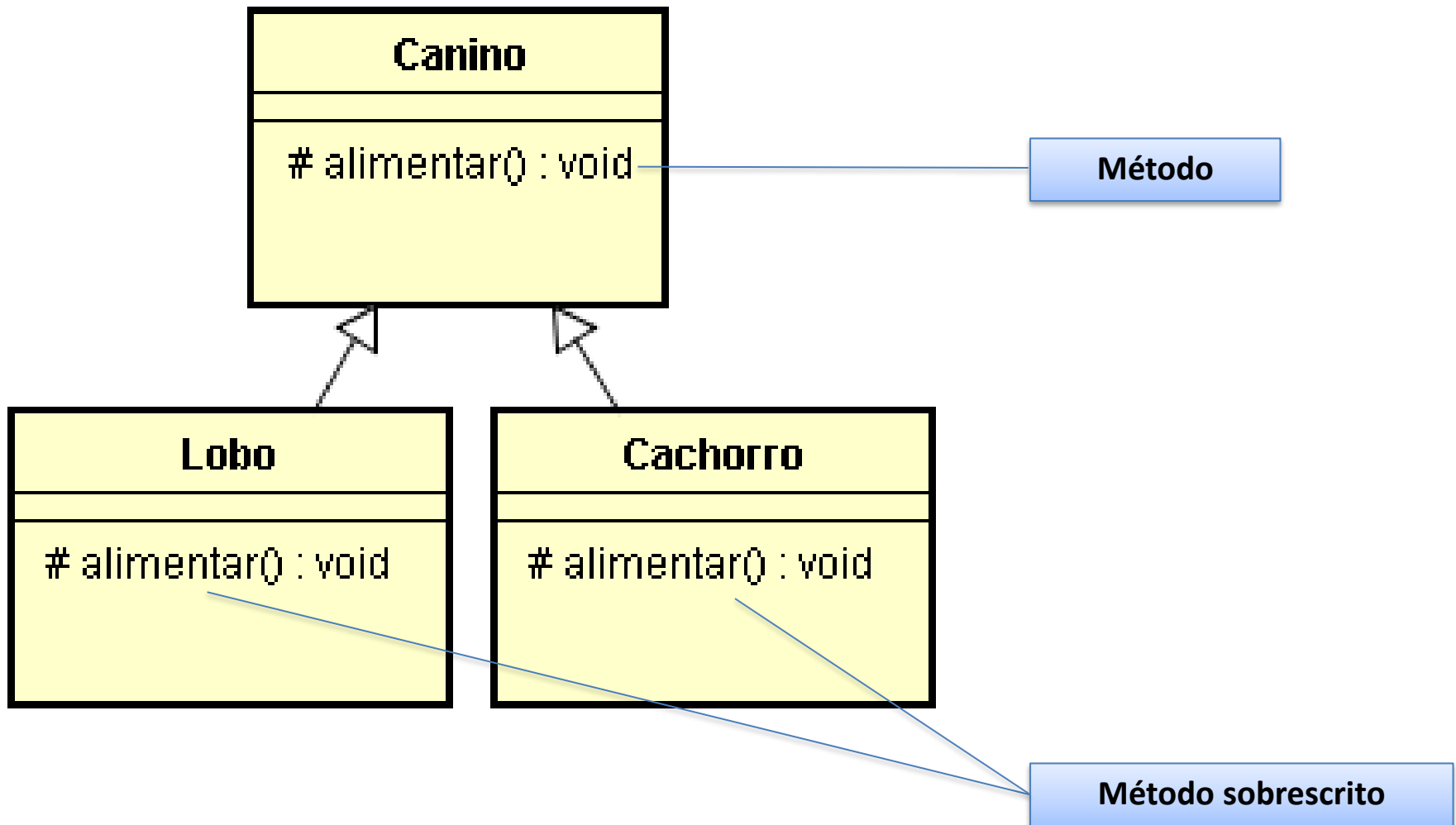
```
    void setModelo(String modelo) {
```

```
        this.modelo = modelo;
```

```
    }
```

```
}
```

- **Sobrescrita**, também conhecida como **sobreposição**, é a **implementação de métodos em subclasses** de tal forma que anule o comportamento que ele apresentava em sua **superclasse** ou **apenas acrescente novas instruções**;
- Ocorre quando o **método herdado apresenta o mesmo nome** em relação ao **método** que está sendo codificado na **subclasse**;
- **A sobrescrita de métodos é realizada quando:**
 - Um método da subclasse realize sua tarefa diferente daquela da superclasse;
 - Desejamos acrescentar novas instruções à implementação de um método da subclasse;



- Neste exemplo, a classe **Diretor** sobrescreveu o método **getSalario()** da classe **Funcionario**:

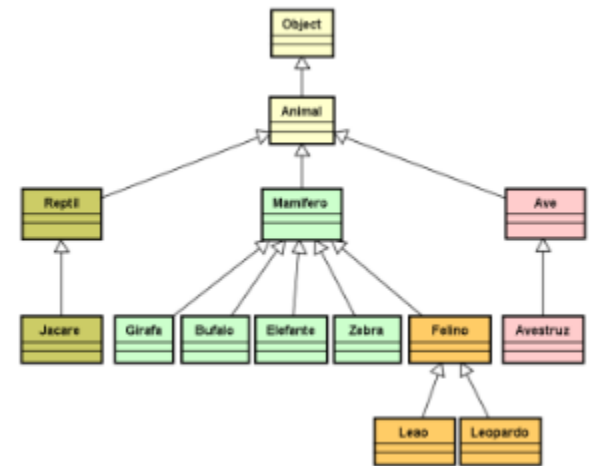
```
public class Funcionario{  
  
    private double salario;  
  
    public double getSalario(){  
        return salario;  
    }  
  
}  
  
public class Diretor extends Funcionario{  
  
    public double getSalario(){  
        return super.getSalario() + 5050.00;  
    }  
  
}
```

- Verifica se um **objeto** é uma **instância da classe** testada;
 - Retorna **true** se o objeto à **esquerda do operador** é do **tipo(classe) especificado** à direita do operador;

```
Leao leao = new Leao();
```

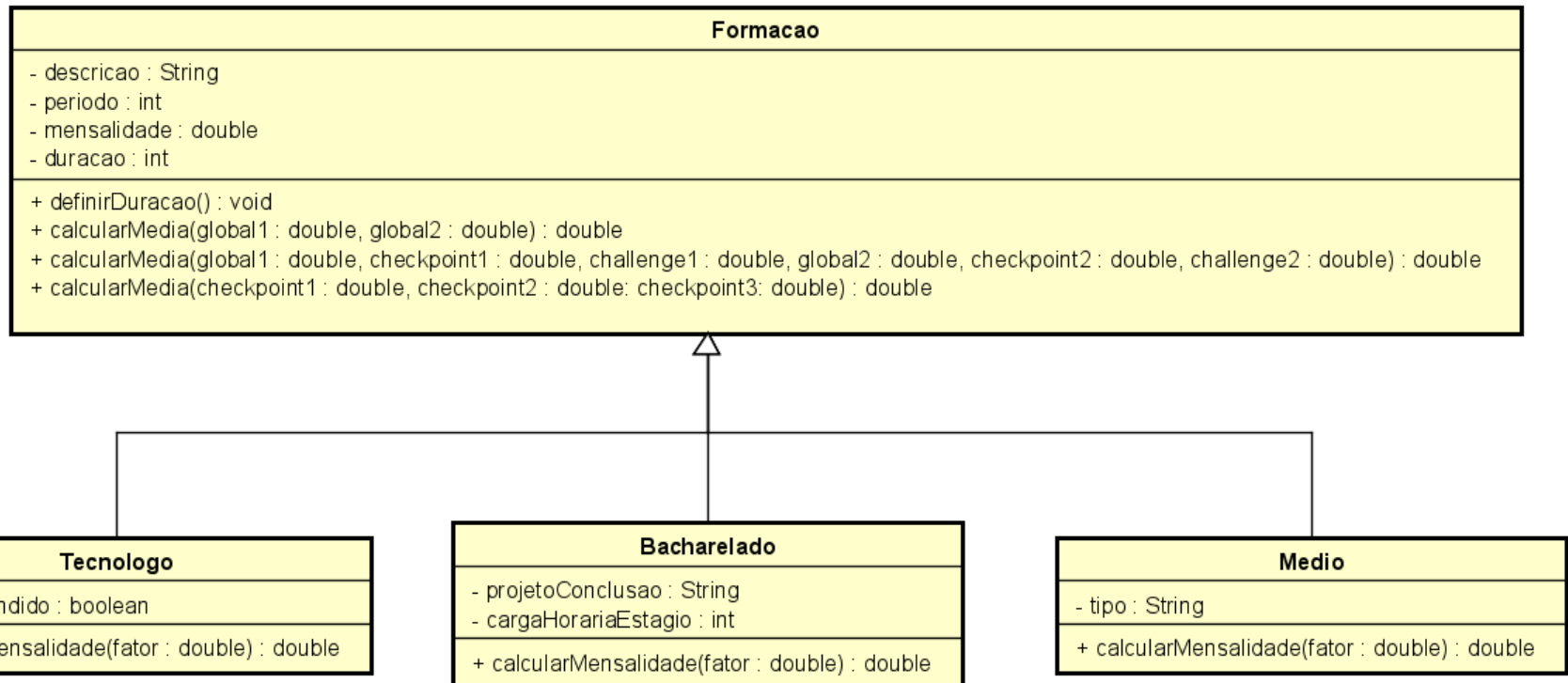
```
Animal ani = leao;
```

```
if(ani instanceof Felino){  
    System.out.println("ani eh do tipo Felino");  
} else {  
    System.out.println("ani NAO eh do tipo Felino");  
}
```



Saída para o Console: ani eh do tipo Felino

- Crie as classes de acordo com o diagrama;
- Acrescente os getters e setters e os construtores;



PRÁTICA 1

Implemente o método **toString()** para todas as classes.

Para programar o método **exibirMedia()**, leve em consideração:

- Global 1 (40%) e Global 2 (60%)
- Checkpoint 1, 2 e 3, será a média aritmética das duas maiores notas;
- Cálculo normal com todas as notas;

Para programar os métodos **calcularMensalidade()**, leve em consideração:

- A duração representa a quantidade de meses do curso.
- Para atribuir à mensalidade, será necessário aplicar as seguintes fórmulas:

Médio => duracao * fator * 500

Tecnologo => duracao * fator * 600

Bacharelado => (duracao * fator * 600) + (cargaHorariaEstagio * 12)

O método **definirDuracao()** terá que definir o atributo “duracao”, seguindo a seguinte regra:

- Se o objeto instanciado for Medio deverá ser atribuído: 36
- Se for Tecnologo deverá ser atribuído: 24
- Se for Bacharelado deverá atribuir 60 se possuir na descrição a palavra “ENGENHARIA”, caso contrário deverá atribuir 48.



PRÁTICA 1



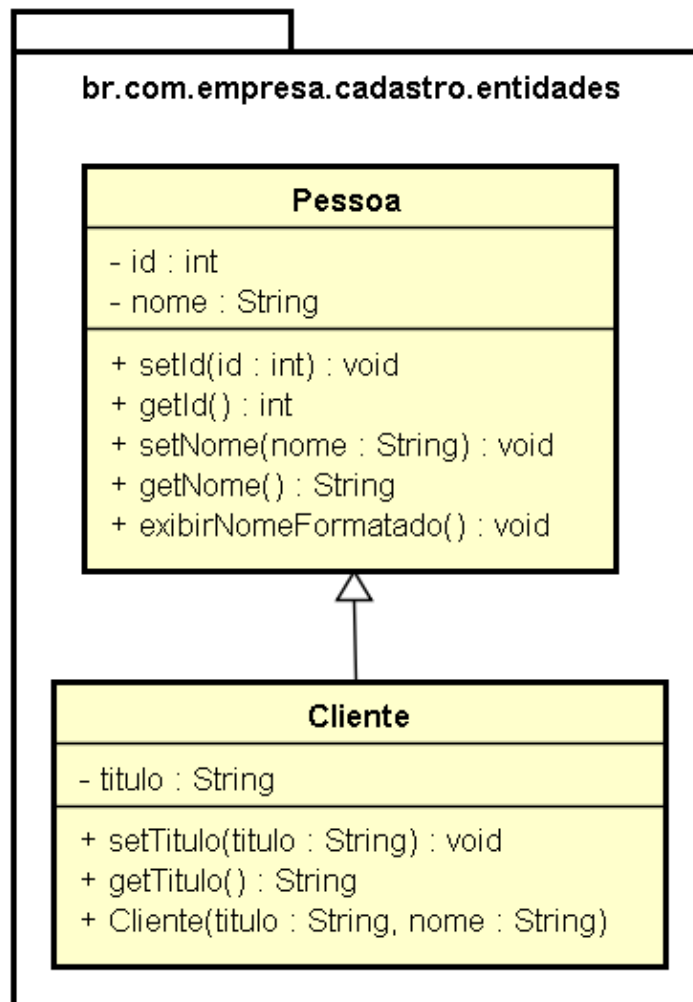
Crie a classe de TesteFormacao

Programa para que seja realizada a pergunta ao usuário sobre qual formação deseja cadastrar. Então preencha o objeto devidamente (via construtor) e ao término utilize o método **toString()** a fim de verificar se a duração está sendo definida corretamente.

Problema: caso deixe vazio, a descrição e instancie um objeto Bacharelado, será gerado um erro. Altere o código para que este erro seja sanado.

PRÁTICA 2

- Implemente o seguinte diagrama de classe.



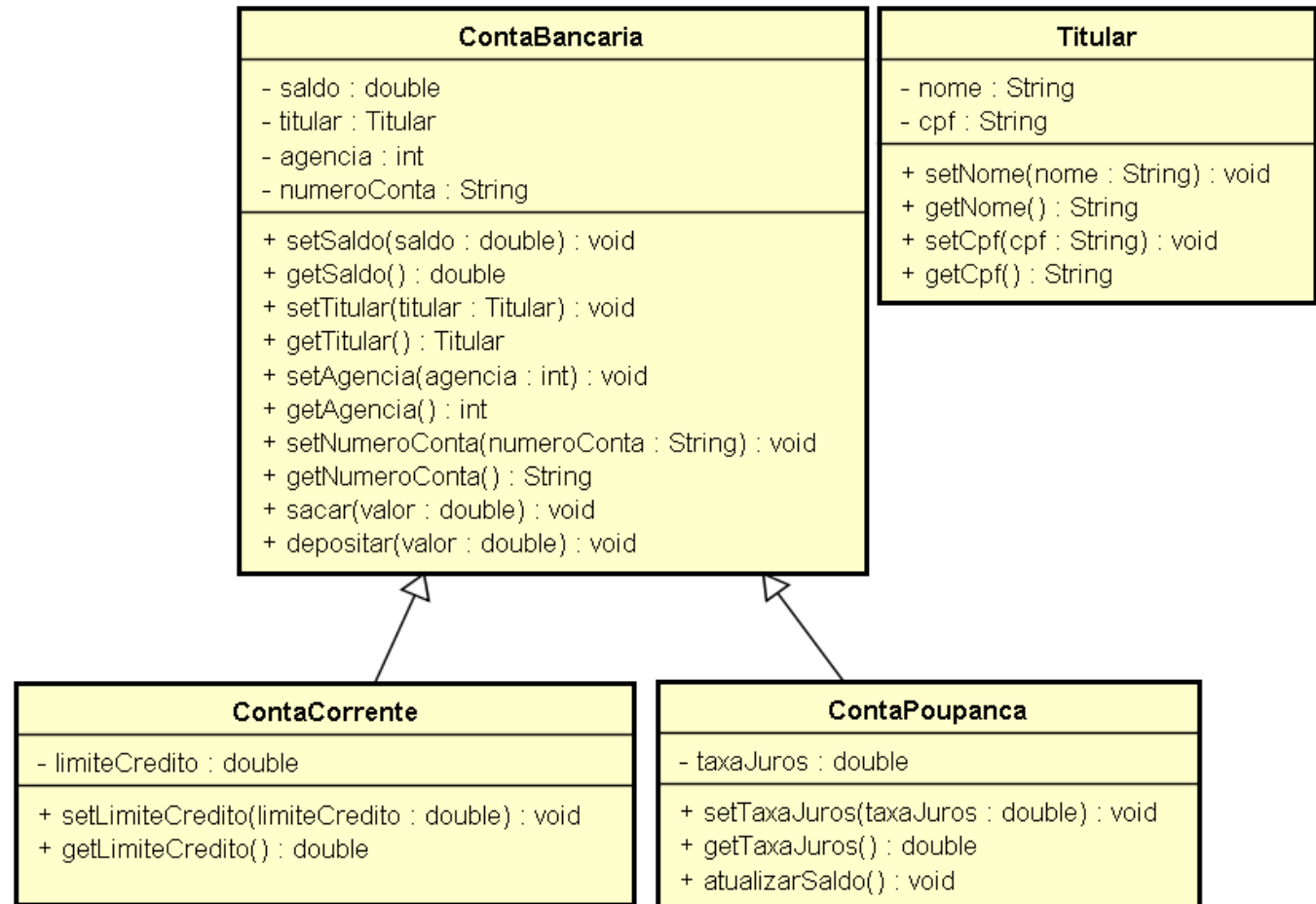
PRÁTICA 2



- Faça as seguintes alterações nas classes criadas anteriormente:
 - Implemente o método **exibirNomeFormatado** da classe **Pessoa** de forma que ele exiba o atributo **nome** em maiúsculo
 - Sobrescreva o método **exibirNomeFormatado** na classe **Cliente** de forma que ele exiba o **título** e o **nome** do cliente, em maiúsculo e no seguinte formato: **<TITULO> - <NOME>**
 - Adicione um construtor na classe **Pessoa** de forma que o atributo **nome** possa ser inicializado por ele. Fique livre para fazer todas as alterações nas classes do projeto que sejam necessárias para a compilação das classes.
- Escreva um programa para cadastrar um Cliente. Instancie o objeto do tipo **Cliente** e preencha os atributos deste objeto com dados vindos do usuário e depois imprima o nome do cliente cadastrado através do método **exibirNomeFormatado**.
- *Nota: Utilize as classes criadas no exercício anterior.*

PRÁTICA 3

- Implemente o seguinte diagrama de classe.



PRÁTICA 3



- Faça as seguintes alterações no exercício anterior:
 - O método **depositar** da classe **ContaBancaria** deve adicionar ao atributo **saldo** o valor passado como parâmetro.
 - O método **sacar** da classe **ContaBancaria** deve subtrair do atributo **saldo** o valor passado como parâmetro.
 - O método **atualizarSaldo** da classe **ContaPoupanca** deve atualizar o valor do **saldo** através do seguinte cálculo:
 - $\text{saldo} + (\text{saldo} * (\text{taxa de juros} / 100))$
- Faça um programa para testar as classes **ContaCorrente** e **ContaPoupanca** e invoque os métodos **sacar** e **depositar** das instâncias destas classes.
- Crie o método **exibirSaldo()**, que deve existir na “super classe” e na subclasse **ContaCorrente** onde o método deverá levar em consideração o limite.

Copyright © 2020 - 2025
Prof. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

“É costume de um tolo, quando erra, queixar-se dos outros. É costume de um sábio queixar-se de si mesmo”. (Sócrates)