INSTITUTO INFNET

LÍVIA FARIA BRAZ

TESTE DE PERFOMANCE 3

Disciplina Regular 1: Fundamentos de Desenvolvimento com Java

Professor: Bernardo Petry Prates

ITAÚNA - MG

15 de março de 2025

➢ Exercício 01: Conceito de Classe, Objeto, Campos e Métodos

A Programação Orientada a Objetos (POO) é um jeito de organizar o código pensando em objetos do mundo real. Aqui estão os conceitos básicos:

➔ **Classe:** É como um molde ou um plano para criar objetos. Define quais características (atributos) e comportamentos (métodos) os objetos vão ter.

```
public class Pirata {
```

➔ **Objeto:** É a instância da classe, ou seja, a versão real baseada no molde. Se a classe for "Pirata", um objeto seria "Luffy" ou "Zoro".

➔ **Campos (ou atributos):** São as informações que o objeto guarda, como nome e recompensa no caso dos piratas.

```
private String nome;

private double recompensa;
```

➔ **Métodos:** São as ações que o objeto pode realizar. No nosso código, um pirata pode atacar ou aumentar sua recompensa.

```
@Override

public String toString() {

    return "--- Pirata ---\nNome: " + nome + "\nRecompensa: " + recompensa + "
berries\n";

}

public void atacar(){

   System.out.println(nome + " partiu para o ataque!");

}

void aumentarRecompensa(double valor){

   recompensa += valor;

   System.out.println("Recompensa de " + nome + " aumentou para " + recompensa + "
berries.");

}
```

Criamos uma classe chamada Pirata, que define como um pirata de One Piece deve ser. Ele tem dois atributos:

nome: armazena o nome do pirata.

recompensa: guarda o valor da recompensa dele.

Temos dois construtores: um vazio (caso a gente queira criar o pirata e definir os valores depois) e outro que já recebe nome e recompensa logo na criação.

```java
public Pirata() {

}

public Pirata(String nome, double recompensa) {

   this.nome = nome;

   this.recompensa = recompensa;

}
```

Os métodos são as ações que um pirata pode fazer:

atacar(): exibe uma mensagem dizendo que o pirata partiu para o ataque.

aumentarRecompensa(double valor): adiciona um valor à recompensa atual e mostra a nova quantia.

toString(): sobrescreve o toString() padrão do Java para que, quando tentarmos imprimir um objeto com System.out.println(), ele exiba os dados formatados do jeito que queremos. Também tem os métodos get e set, que servem para acessar e modificar os atributos de um jeito controlado.

Agora, na Main, criamos dois piratas:

```java
Pirata p1 = new Pirata();

p1.setNome("Zoro");

p1.setRecompensa(32000);
```

Aqui, criamos p1 sem passar valores no construtor e depois usamos os métodos set para definir nome e recompensa.

```java
Pirata p2 = new Pirata("Luffy", 100000);

p2.atacar();

p1.aumentarRecompensa(10);
```

Já p2 (Luffy) é criado diretamente com nome e recompensa. Ele parte para o ataque e, em seguida, aumentamos a recompensa de p1 (Zoro) em 10 berries.

Por fim, usamos System.out.println(p1) e System.out.println(p2). Isso chama o método toString() da classe Pirata, que formata as informações para exibição. No retorno temos:

```
Luffy partiu para o ataque!
Recompensa de Zoro aumentou para 32010.0 berries.
---- Pirata ----
Nome: Zoro
Recompensa: 32010.0 berries

---- Pirata ----
Nome: Luffy
Recompensa: 100000.0 berries
```

➢ **Exercício 02:** Criando a Classe “Produto” (Com Contexto de Usuário)

A classe Produto é essencial para representar qualquer item que pode ser vendido ou gerenciado em um sistema. Os atributos principais são:

- nome (String): guarda o nome do produto
- preco (double): representa o valor do produto. Como o preço pode ter casas decimais (R$19,99, por exemplo), usamos double.
- quantidadeEmEstoque (int): indica quantas unidades do produto ainda estão disponíveis

Ou seja, cada atributo tem seu papel para garantir que o sistema consiga gerenciar os produtos direitinho.

```java
package com.codingloria.TP3.ex02_03_04_05_06;

public class Produto {

    String nome;

    double preco;

    int quantidadeEmEstoque;

}
```

➢ **Exercício 03:** Métodos Básicos da Classe “Produto”

```java
public double alterarPreco(double novoPreco) {

    this.preco = novoPreco;

    return this.preco;

}


public int alterarEstoque(int novaQuantidade) {

    this.quantidadeEmEstoque += novaQuantidade;

    return this.quantidadeEmEstoque;

}


public void exibirInformacoes() {

    System.out.println("Nome: " + this.nome);
```

```
    System.out.println("Preço: " + this.preco);
    System.out.println("Quantidade em estoque: " +
this.quantidadeEmEstoque);
}
```

➢ **Exercício 04:** Testando a Classe “Produto”

```
package com.codingloria.TP3.ex02_03_04_05_06;

public class Main {
    public static void main(String[] args) {
        Produto p1 = new Produto();
        p1.nome = "Camisa";
        p1.preco = 50.0;
        p1.quantidadeEmEstoque = 10;

        p1.alterarPreco(60.0);
        p1.alterarQuantidade(16);
        p1.exibirInformacoes();
    }
}
```

```
---- Produto ----
Nome: Camisa
Preço: 60.0
Quantidade em estoque: 26
```

➢ **Exercício 05:** Criando Métodos de Propriedade (Getters e Setters)

Os getters e setters trazem algumas vantagens:

- Deixam o código mais organizado e evitam acesso direto às variáveis, o que pode prevenir modificações indesejadas.
- Facilitam a validação de dados. Por exemplo, no setter de preco, poderíamos garantir que o preço nunca seja negativo.
- Melhoram a flexibilidade do código. Se um dia for necessário mudar a lógica interna de um atributo (como calcular o preço com desconto automaticamente), dá pra fazer isso dentro do getter sem impactar outras partes do código.

Ou seja, os getters e setters ajudam a manter o controle sobre os dados e evitam bagunça no código.

```
public String getNome() {
    return nome;
}
public void setNome(String nome) {
    this.nome = nome;
}
public double getPreco() {
    return preco;
}

public void setPreco(double preco) {
    this.preco = preco;
}

public int getQuantidadeEmEstoque() {
    return quantidadeEmEstoque;
}

public void setQuantidadeEmEstoque(int quantidadeEmEstoque) {
    this.quantidadeEmEstoque = quantidadeEmEstoque;
}
```

```
System.out.println("----------------");
p1.setNome("Calça");
p1.setPreco(80.0);
p1.setQuantidadeEmEstoque(20);


System.out.println("Nome: " + p1.getNome());
System.out.println("Preço: " + p1.getPreco());
System.out.println("Quantidade em estoque: " +
p1.getQuantidadeEmEstoque());
```

```
---- Produto ----
Nome: Camisa
Preço: 60.0
Quantidade em estoque: 26
----------------
Nome: Calça
Preço: 80.0
Quantidade em estoque: 20
```

➢ **Exercício 06:** Adicionando Construtores à Classe “Produto”

O construtor serve para inicializar o objeto no momento em que ele é criado. Normalmente como padrão, o Java já cria um construtor vazio pra usarmos, mas quando queremos iniciar a classe com os atributos, precisamos construí-lo, caso a gente precise deixar as duas opções (de setar depois ou na criação) podemos criar um vazio e outro com os atributos.

```
public Produto() {}
public Produto(String nome, double preco, int quantidadeEmEstoque) {
    this.nome = nome;
    this.preco = preco;
    this.quantidadeEmEstoque = quantidadeEmEstoque;
}
```

```
Produto p2 = new Produto("Tênis", 150.0, 5);
p2.exibirInformacoes();
```

```
---- Produto ----
Nome: Tênis
Preço: 150.0
Quantidade em estoque: 5
```

➢ **Exercício 07:** Modelando uma Conta Bancária

```
package com.codingloria.TP3.ex07_08_09;

public class Conta {
    String titular;
    int numero;
    String agencia;
    double saldo;
    String dataAbertura;
```

➢ **Exercício 08:** Criando Métodos

```
public Double saca(double valor) {
    if (this.saldo >= valor) {
        this.saldo -= valor;
        return valor;
    }
    return null;
}
```

```java
public void deposita(double valor) {
    this.saldo += valor;
}

public double calculaRendimento() {
    return this.saldo * 0.1;
}
```

➢ **Exercício 09:** Vamos testar nossa classe

```java
package com.codingloria.TP3.ex07_08_09;

public class TestaConta{
    public static void main(String[] args) {
        Conta c1 = new Conta();
        c1.titular = "Luan";
        c1.numero = 123;
        c1.agencia = "456";
        c1.saldo = 1000;
        c1.dataAbertura = "18/10/2022";

        System.out.println("#### Conta ####");
        System.out.println("Titular: " + c1.titular);
        System.out.println("Número: " + c1.numero);
        System.out.println("Agência: " + c1.agencia);
        System.out.println("Saldo: " + c1.saldo);
        System.out.println("Data de abertura: " + c1.dataAbertura);

        System.out.println("#### Operações ####");
        c1.saca(100);
```

```
        System.out.println("Saldo atual: " + c1.saldo);
        c1.deposita(200);
        System.out.println("Novo saldol: " + c1.saldo);
        double rendimento = c1.calculaRendimento();
        System.out.println("Rendimento: " + rendimento);
    }
}
```

```
#### Conta ####
Titular: Luan
Número: 123
Agência: 456
Saldo: 1000.0
Data de abertura: 18/10/2022
#### Operações ####
Saldo atual: 900.0
Novo saldo: 1100.0
Rendimento: 110.0
```

➢ **Exercício 10:** Definindo Classes para Formas Geométricas

O atributo raio é essencial porque ele é a base para qualquer cálculo envolvendo um círculo ou uma esfera.

- No caso do círculo, a área é calculada com π * raio². Ou seja, sem o raio, nem daria pra definir a figura corretamente.
- Já na esfera, o volume depende de 4/3 * π * raio³. Então, o raio não só define o tamanho da esfera, mas também seu volume total.

Basicamente, o raio é o que dá "proporção" à forma. Sem ele, não daria pra calcular nada nem saber o tamanho do objeto.

```java
package com.codingloria.TP3.ex10_11_12;

public class Esfera {
    double raio;
}
```

```java
package com.codingloria.TP3.ex10_11_12;

public class Circulo {
    double raio;
}
```

➢ **Exercício 11:** Criando Métodos de Cálculo

```java
double calculaVolume() {
    return (4.0 / 3.0) * Math.PI * Math.pow(raio, 3);
}
```

```java
double calculaArea() {
    return Math.PI * Math.pow(raio, 2);
}
```

➢ **Exercício 12:** Testando as Classes de Figuras

```java
package com.codingloria.TP3.ex10_11_12;

public class TestaFiguras {
    public static void main(String[] args) {
        Circulo circulo = new Circulo();
```

```
        Esfera esfera = new Esfera();


        circulo.raio = 3.0;

        esfera.raio = 5.0;


        System.out.println("Área do círculo: " + circulo.calculaArea());

        System.out.println("Volume da esfera: " +
esfera.calculaVolume());

    }

}
```