**Programação Orientada a Objetos**

1. **(1,25) Considerando os conceitos básicos de Orientação a Objetos, explique o significado dos conceitos abaixo:**
   1. **Encapsulamento:** Encapsulamento vem de encapsular, que em programação orientada a objetos significa separar o programa em partes, o mais isolado possível. A ideia é tornar o software mais flexível, fácil de modificar e de criar novas implementações. O Encapsulamento serve para controlar o acesso aos atributos e métodos de uma classe de uma forma eficiente de proteger os dados manipulados dentro da classe, além de determinar onde esta classe poderá ser manipulada.
   2. **Herança:** É um princípio de [orientação a objetos](https://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos), que permite **que as classes compartilhem seus atributos, métodos e outros membros da classe entre si. Para a ligação entre as classes, a herança adota um relacionamento esquematizado hierarquicamente**.
   3. **Polimorfismo:** É um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas. Esse mecanismo por meio do qual selecionamos as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução.
   4. **Atributos**: São as características de um objeto, essas características também são conhecidas como variáveis, utilizando o exemplo dos cães, temos alguns atributos, tais como: cor, peso, altura e nome.

**Métodos:** São as ações que os objetos podem exercer quando solicitados, onde podem interagir e se comunicarem com outros objetos, utilizando o exemplo dos cães, temos alguns exemplos: latir, correr, pular.

5. **Atributos Estático:** É um estado que vive na classe. Por conta disso, sempre que você cria um novo objeto dessa classe, eles possuem o mesmo valor para o atributo. Geralmente é usado para constantes.

**Métodos Estáticos:** Método estático também é um método que vive na classe. Então é um método que não pode usar nenhum atributo do objeto e geralmente é usado para fazer operações que não dependem de nenhum estado.

1. **(0,75) Diferencia os mecanismos de sobrecarga e sobrescrita, demonstrando cenário de uso de cada um desses.**

**Sobrecarga** é você ter dois ou mais métodos com o mesmo nome, mas variando os tipos dos parâmetros que esse método recebe, ou overload de métodos em Java a sobrecarga.

**Exemplo:**

Class Email{

public void enviarEmail(){

System.out.println(“Enviar email”);

}

Public void enviarEmail(String assunto){

System.out.println(“Enviar email,assunto: “+assunto);

}

}

**Sobrescrita** é você sobrescrever o comportamento de um método, ou seja, o ato de sobrescrever um método ou propriedade que significa dar uma nova forma ao mesmo a uma nova versão. Exemplo: Dado o cenário que tenho uma classe base móvel, e uma classe derivada Cama. Sendo assim, quando você instanciar um objeto do tipo Cama, o método Preço será executado de acordo com a operação que estiver dentro do método Cama. Preço, pois ele está sobrescrevendo o método Preço da classe base (Móvel). Em suma, podemos reescrever nas classes filhas (derivadas) métodos criados inicialmente na classe pai (base).

class Movel { public virtual void Preco(){ //corpo do método } }

class Cama : Movel { public override void Preco(){ //corpo do método } }

**Outro Exemplo:**

class Animal {

public void comer(){

System.out.println(“Animal,comer”);

}

}

Class Leao extends Animal{

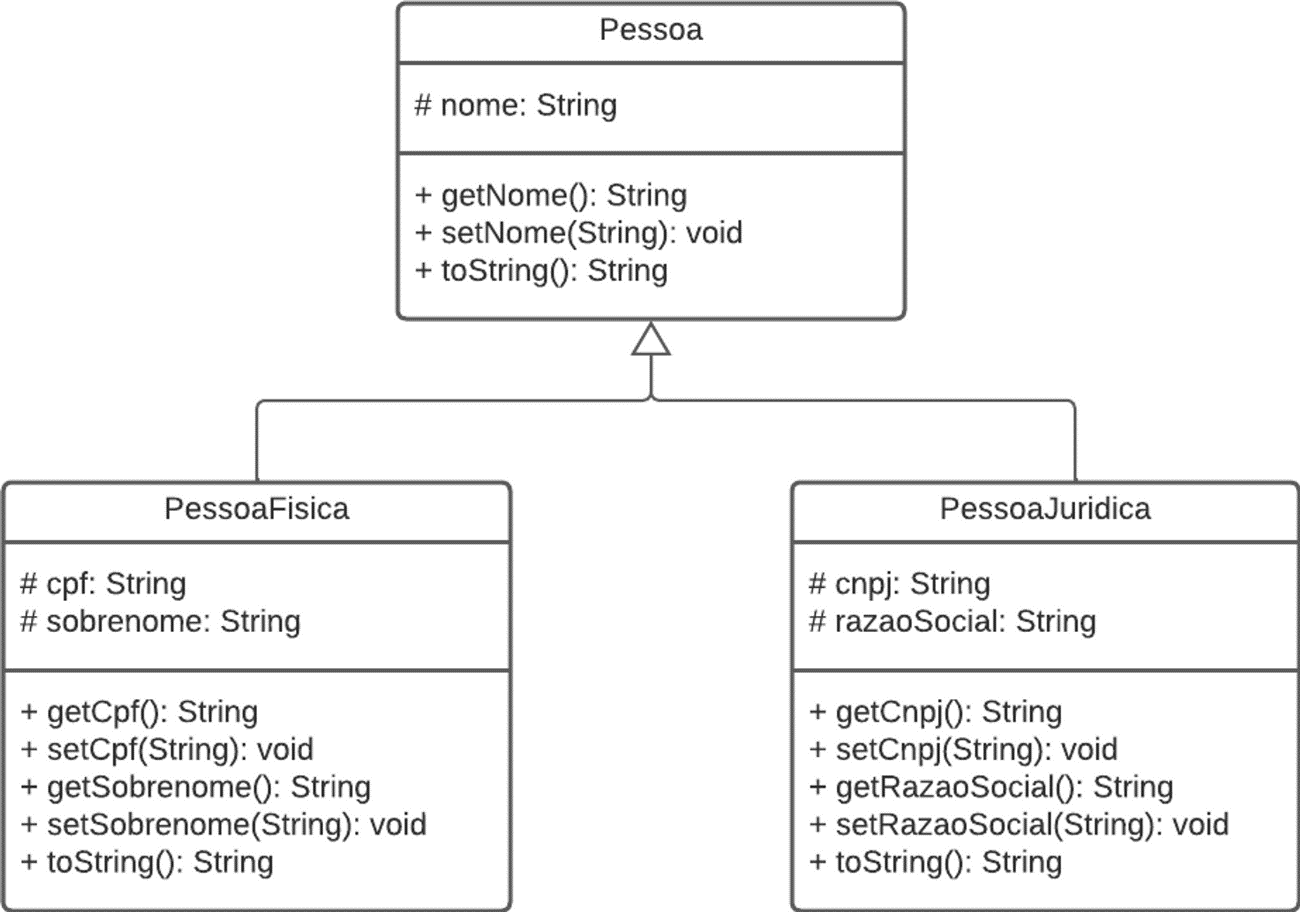
Public void comer(){

System.out.println(“Leão, comer”);

}

}

1. **(1,5) A partir do diagrama abaixo, implemente a estrutura de classes.**



public class Pessoa {  
  
 protected String nome;  
  
 public String getNome() {  
 return nome;  
 }  
 public void setNome(String nome) {  
 this.nome = nome;  
 }  
  
 @Override  
 public String toString() {  
 return "Pessoa{" +  
 "nome='" + nome + '\'' +  
 '}';

}

}

public class PessoaFisica extends Pessoa {

protected String cpf;

protected String sobrenome;

public String getCpf() {

return cpf;

}

public void setCpf(String cpf) {

this.cpf = cpf;

}

public String getSobrenome() {

return sobrenome;

}

public void setSobrenome(String sobrenome) {

this.sobrenome = sobrenome;

}

@Override

public String toString() {

return "PessoaFisica{" +

"cpf='" + cpf + '\'' +

", sobrenome='" + sobrenome + '\'' +

'}';

}

}

public class PessoaJuridica extends Pessoa {

protected String cnpj;

protected String razaoSocial;

public String getCnpj() {

return cnpj;

}

public void setCnpj(String cnpj) {

this.cnpj = cnpj;

}

public String getRazaoSocial() {

return razaoSocial;

}

public void setRazaoSocial(String razaoSocial) {

this.razaoSocial = razaoSocial;

}

@Override

public String toString() {

return "PessoaJuridica{" +

"cnpj='" + cnpj + '\'' +

", razaoSocial='" + razaoSocial + '\'' +

'}';

}

}

1. (1,5) Modifique a o projeto banco (abordado em sala de aula) para que o cliente usado seja um tipo PessoaFisica ou PessoaJuridica.

public class Principal {

public static void main(String[] args) {

PessoaFisica jeova = new PessoaFisica("Jeová", "Tavares", 37);

Conta c = new Conta(jeova);

c.creditar(2000.0);

c.debitar(100);

PessoaFisica samuel = new PessoaFisica("Samuel", "Rodrigues", 37);

ContaEspecial c2 = new ContaEspecial(samuel, 200);

c2.debitar(500);

ContaPoupanca c3 = new ContaPoupanca

(new PessoaFisica("Agamenon", "Quinderé", 25), 10000);

c3.rendeJuros();

System.out.println(c);

System.out.println(c2);

System.out.println(c3);

PessoaJuridica empresaX = new PessoaJuridica(12345, "razaoSocial1");

Conta cj = new Conta(empresaX);

cj.creditar(2000.0);

cj.debitar(100);

System.out.println(cj);

PessoaJuridica empresaY = new PessoaJuridica(123, "razaoSocial2");

ContaEspecial cj2 = new ContaEspecial(empresaY, 200);

c2.debitar(500);

System.out.println(cj2);

ContaPoupanca cj3 = new ContaPoupanca

(new PessoaJuridica(1234, "razaoSocial3"), 10000);

c3.rendeJuros();

System.out.println(cj3);

}

}

public class Conta {

protected String numero;

protected PessoaFisica clienteFisico;

protected PessoaJuridica clienteJuridico;

protected double saldo;

private int index;

private String estado;

public Conta() {

int n = (int) (Math.random() \* 10000 + 1);

String numConta = "c" + n;

this.numero = numConta;

}

public Conta(PessoaFisica clienteFisico) {

this();

this.index = 1;

this.clienteFisico = clienteFisico;

}

public Conta(PessoaJuridica clienteJuridico) {

this();

this.index = 2;

this.clienteJuridico = clienteJuridico;

}

private String mostraFisico() {

String saida = "--------------";

saida += "\nNome: " + this.clienteFisico.toString();

saida += "\nNumero: " + this.numero;

saida = saida + "\nSaldo :" + this.saldo;

saida += "\n-------------";

return saida;

}

private String mostraJuridico() {

String saida = "------------------------------";

saida += "\ncnpj: " + this.clienteJuridico.getCnpj();

saida += "\nrazaoSocial: " + this.clienteJuridico.getRazaoSocial();

saida += "\nNumero: " + this.numero;

saida = saida + "\nSaldo :" + this.saldo;

saida += "\n----------------------------------";

return saida;

}

@Override

public String toString() {

if (this.index == 1) {

this.estado = "\n" + mostraFisico();

} else if (this.index == 2) {

this.estado = "\n" + mostraJuridico();

}

return estado;

}

public void creditar(double valor) {

if (valor >= 0) {

this.saldo += valor;

}

}

public void debitar(double valor) {

if (valor <= this.saldo) {

this.saldo = this.saldo - valor;

}

}

}

public class ContaPoupanca extends Conta {

private static double taxa = 0.1;

public ContaPoupanca(PessoaFisica clienteFisico, double saldo) {

super(clienteFisico);

creditar(saldo);

}

public ContaPoupanca(PessoaJuridica clienteJuridico, double saldo) {

super(clienteJuridico);

creditar(saldo);

}

public void rendeJuros() {

creditar(taxa\*this.saldo);

}

}

public class ContaEspecial extends Conta {

private double limite;

public ContaEspecial(PessoaFisica clienteFisico) {

super(clienteFisico);

this.limite = 500;

this.numero += "S";

}

public ContaEspecial(PessoaFisica clienteFisico, double saldo) {

this(clienteFisico);

this.saldo = saldo;

}

public ContaEspecial(PessoaJuridica clienteJuridico) {

super(clienteJuridico);

this.limite = 500;

this.numero += "S";

}

public ContaEspecial(PessoaJuridica clienteJuridico, double saldo) {

this(clienteJuridico);

this.saldo = saldo;

}

@Override

public void debitar(double valor) {

double saldoTotal = saldo + limite;

if(valor <= saldoTotal) {

this.saldo = this.saldo - valor;

}

}

public class PessoaFisica extends Pessoa {

protected String sobrenome;

protected int idade;

public PessoaFisica(String nome, String sobrenome, int idade) {

super(nome);

this.sobrenome = sobrenome;

this.idade = idade;

}

public String fullName() {

return super.nome + " " + sobrenome;

}

@Override

public String toString() {

return fullName()+ ", com "+ idade+ " anos.";

}

}

public class PessoaJuridica extends Pessoa {

protected int cnpj;

protected String razaoSocial;

public PessoaJuridica(int cnpj, String razaoSocial) {

this.cnpj = cnpj;

this.razaoSocial = razaoSocial;

}

public int getCnpj() {

return cnpj;

}

public String getRazaoSocial() {

return razaoSocial;

}

}

public class Pessoa {

protected String nome;

public Pessoa() {

}

public Pessoa(String nome) {

this.nome = nome;

}

}