Exercício - Herança

**Instruções**

1. Descreva algumas diferenças básicas entre programação estruturada e programação orientada a objetos.

A programação estruturada baseia-se em procedimentos que executam operações em dados. O código é organizado em blocos lógicos que são chamados em sequência, a ênfase está em como fazer as etapas da execução. Os dados geralmente são globais ou passados entre funções. Na programação estruturada reutilizar código exige copiar e adaptar funções. E exemplos de linguagem de programação estruturada são o C e o Pascal.

Na programação orientada a objetos os programas são estruturados em torno de objetos, que são instâncias de classes. O código é organizado em classes que contêm atributos (dados) e métodos (funções). A ênfase está em quem faz os objetos e suas responsabilidades. Os dados são encapsulados dentro dos objetos, o que aumenta a segurança e o controle. Suporta herança, polimorfismo e reutilização de código com mais facilidade. Exemplos de linguagem em programação orientada a objetos são o Java, C++ e Python.

2. Para que um membro de uma classe-base possa ser acessado por membros da classe derivada, eles devem ser:

a) public;

b) protected;

c) private;

d) todas as anteriores.

Os membros devem ser **b)Protected**.

3. Os membros de uma classe-base podem acessar:

a) membros públicos da classe derivada;

b) membros protegidos da classe derivada;

c) membros privados da classe derivada;

d) nenhuma das anteriores.

Os membros de uma classe-base **não podem acessar nenhum membro da classe derivada**.

4. Para que serve um construtor e como ele pode ser utilizado? Implemente dois exemplos:

Um construtor serve para inicializarobjetos de uma classe quando eles são criados. Ele é um método especial chamado automaticamente no momento da criação do objeto.

a) um para ser possível fazer 'Classe("POO")'

public class Classe {

private String nome;

public Classe(String nome) {

this.nome = nome;

}

public void exibir() {

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

}

public static void main(String[] args) {

Classe obj = new Classe("POO");

obj.exibir();

}

}

b) e outro para ser possível fazer 'Classe("POO", 30)'

public class Classe {

private String nome;

private int idade;

public Classe(String nome, int idade) {

this.nome = nome;

this.idade = idade;

}

public void exibir() {

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

System.out.println("Idade: " + idade);

}

public static void main(String[] args) {

Classe obj = new Classe("POO", 30);

obj.exibir();

}

}

5. Crie uma classe calculadora. Esta classe deve ser abstrata e implementar as operações básicas (soma, subtração, divisão e multiplicação). Utilizando o conceito de herança crie uma classe chamada calculadora cientifica que implementa os seguintes cálculos: raiz quadrada e a potência. Dica utilize a classe Math do pacote java.lang.

public abstract class Calculadora {

public double somar(double a, double b) {

return a + b;

}

public double subtrair(double a, double b) {

return a - b;

}

public double multiplicar(double a, double b) {

return a \* b;

}

public double dividir(double a, double b) {

if (b == 0) {

throw new ArithmeticException("Divisão por zero não permitida.");

}

return a / b;

}

}

public class CalculadoraCientifica extends Calculadora {

public double raizQuadrada(double valor) {

if (valor < 0) {

throw new ArithmeticException("Não é possível calcular raiz quadrada de número negativo.");

}

return Math.sqrt(valor);

}

public double potencia(double base, double expoente) {

return Math.pow(base, expoente);

}

}

public class Main {

public static void main(String[] args) {

CalculadoraCientifica calc = new CalculadoraCientifica();

System.out.println("Soma: " + calc.somar(10, 5));

System.out.println("Subtração: " + calc.subtrair(10, 5));

System.out.println("Multiplicação: " + calc.multiplicar(10, 5));

System.out.println("Divisão: " + calc.dividir(10, 5));

System.out.println("Raiz quadrada de 25: " + calc.raizQuadrada(25));

System.out.println("2 elevado a 3: " + calc.potencia(2, 3));

}

}

6. Crie uma classe chamada Empresa capaz de armazenar os dados de uma empresa (Nome, Endereço, Cidade, Estado, CEP e Fone). Inclua um construtor sem argumentos e um que receba os dados como argumentos e os inicialize. Escreva duas funções, uma para fazer a interface com o usuário da entrada de dados, Get(), e outra para imprimir os dados, Print().

Use a classe Empresa como base para criar a classe Restaurante. Inclua o tipo de comida, o preço médio de um prato, duas funções construtoras, a interface de entrada de dados, Get(), e a função que imprima os dados, Print(). Construa um programa para testar a classe Restaurante.

import java.util.Scanner;

public class Empresa {

protected String nome;

protected String endereco;

protected String cidade;

protected String estado;

protected String cep;

protected String fone;

// Construtor sem argumentos

public Empresa() {

}

// Construtor com argumentos

public Empresa(String nome, String endereco, String cidade, String estado, String cep, String fone) {

this.nome = nome;

this.endereco = endereco;

this.cidade = cidade;

this.estado = estado;

this.cep = cep;

this.fone = fone;

}

public void get() {

Scanner sc = new Scanner(System.in);

System.out.print("Nome: ");

nome = sc.nextLine();

System.out.print("Endereço: ");

endereco = sc.nextLine();

System.out.print("Cidade: ");

cidade = sc.nextLine();

System.out.print("Estado: ");

estado = sc.nextLine();

System.out.print("CEP: ");

cep = sc.nextLine();

System.out.print("Fone: ");

fone = sc.nextLine();

}

public void print() {

System.out.println("---- Dados da Empresa ----");

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

System.out.println("Endereço: " + endereco);

System.out.println("Cidade: " + cidade);

System.out.println("Estado: " + estado);

System.out.println("CEP: " + cep);

System.out.println("Fone: " + fone);

}

}

public class Restaurante extends Empresa {

private String tipoComida;

private double precoMedio;

// Construtor sem argumentos

public Restaurante() {

super();

}

// Construtor com argumentos

public Restaurante(String nome, String endereco, String cidade, String estado, String cep, String fone, String tipoComida, double precoMedio) {

super(nome, endereco, cidade, estado, cep, fone);

this.tipoComida = tipoComida;

this.precoMedio = precoMedio;

}

@Override

public void get() {

super.get();

Scanner sc = new Scanner(System.in);

System.out.print("Tipo de comida: ");

tipoComida = sc.nextLine();

System.out.print("Preço médio: ");

precoMedio = sc.nextDouble();

}

@Override

public void print() {

super.print();

System.out.println("Tipo de Comida: " + tipoComida);

System.out.println("Preço Médio: R$ " + precoMedio);

}

}

public class Main {

public static void main(String[] args) {

Restaurante r = new Restaurante();

System.out.println("Insira os dados do restaurante:");

r.get();

System.out.println("\nInformações cadastradas:");

r.print();

}

}

7. Siga as instruções abaixo:

a) Implemente uma classe abstrata chamada **Forma**

Nesta classe, declare dois métodos abstratos:

Float calcularArea();

Float calcularPerimetro();

public abstract class Forma {

public abstract float calcularArea();

public abstract float calcularPerimetro();

}

b) Crie uma subclasse de Forma chamada Retângulo

A classe Retângulo deve ter dois atributos: lado e altura, ambos do tipo float.

Implemente os métodos herdados de Forma para calcular a área e o perímetro:

Área: lado \* altura

Perímetro: 2 \* (lado + altura).

public class Retangulo extends Forma {

protected float lado;

protected float altura;

public Retangulo(float lado, float altura) {

this.lado = lado;

this.altura = altura;

}

@Override

public float calcularArea() {

return lado \* altura;

}

@Override

public float calcularPerimetro() {

return 2 \* (lado + altura);

}

}

c) Crie uma subclasse de Forma chamada Circulo

A classe Circulo deve ter um atributo raio do tipo float.

Implemente os métodos herdados de Forma para calcular a área e o perímetro:

Área: Math.PI \* raio \* raio

Perímetro: 2 \* Math.PI \* raio.

public class Circulo extends Forma {

private float raio;

public Circulo(float raio) {

this.raio = raio;

}

@Override

public float calcularArea() {

return (float)(Math.PI \* raio \* raio);

}

@Override

public float calcularPerimetro() {

return (float)(2 \* Math.PI \* raio);

}

}

d) Crie uma subclasse de Retângulo chamada Quadrado

A classe Quadrado deve ser uma especialização de Retângulo, onde lado e altura têm o mesmo valor (ou seja, o lado do quadrado).

Para isso, o construtor de Quadrado deve apenas receber um valor para o lado e chamar o construtor de Retângulo com o mesmo valor para lado e altura.

public class Quadrado extends Retangulo {

public Quadrado(float lado) {

super(lado, lado);

}

}

e) Crie um programa de teste

Declare um array de Forma com 5 elementos. Nesse array, você deve armazenar instâncias de Retângulo, Circulo e Quadrado em uma ordem aleatória.

Para gerar números aleatórios, crie uma instância da classe Random (presente na biblioteca java.util).

Utilize o método nextInt(n), obtido através da classe Random, para gerar um número inteiro aleatório entre 0 e n.

Implemente um laço (loop) que percorra o array e, para cada objeto guardado, chame os métodos calcularArea e calcularPerimetro para exibir a área e o perímetro de cada forma.

import java.util.Random;

public class Main {

public static void main(String[] args) {

Forma[] formas = new Forma[5];

Random random = new Random();

for (int i = 0; i < formas.length; i++) {

int tipo = random.nextInt(3); // 0 = Retângulo, 1 = Círculo, 2 = Quadrado

switch (tipo) {

case 0:

float lado = random.nextFloat(10) + 1; // de 1 a 10

float altura = random.nextFloat(10) + 1;

formas[i] = new Retangulo(lado, altura);

System.out.println("Forma " + i + ": Retângulo");

break;

case 1:

float raio = random.nextFloat(10) + 1;

formas[i] = new Circulo(raio);

System.out.println("Forma " + i + ": Círculo");

break;

case 2:

float ladoQuadrado = random.nextFloat(10) + 1;

formas[i] = new Quadrado(ladoQuadrado);

System.out.println("Forma " + i + ": Quadrado");

break;

}

float area = formas[i].calcularArea();

float perimetro = formas[i].calcularPerimetro();

System.out.printf("Área: %.2f\n", area);

System.out.printf("Perímetro: %.2f\n\n", perimetro);

}

}

}