**Instituto Federal de Educação, Ciência e Tecnologia do Ceará**

**Campus Maracanaú**

**Coordenadoria de Computação**

**Curso de Bacharelado em Ciência da Computação**

**Disciplina: Programação Orientada a Objetos**

**Professor: Igor Rafael Silva Valente**

# ATIVIDADE

**Assunto:**

Reuso de classes.

**Orientações:**

A atividade deve ser executada individualmente e entregue através do ambiente *Google Classroom*.

**Regras de criação dos programas:**

Crie um novo projeto Java denominado **AtividadeReusoDeClasses**. As classes devem possuir os nomes informados no texto. Ao final, o projeto deve ser exportado para um arquivo em formato ZIP.

**Nome completo: Francisco Aldenor Silva Neto**

1. Existem duas formas básicas para realizar o reuso de classes: composição e herança. Explique o significado e cite um exemplo para cada uma.

**Composição** : temos a composição quando uma classe A compõe uma classe B, ou seja temos uma instancia da classe A declarado como um objeto na classe B, a expressão de composição também é bastante conhecida como o **“tem um”** , ou seja, a classe A tem uma instancia(objeto ) da classe B.

Exemplo: carro e pneu, a classe carro **tem um** pneu.

**Herança** : como o próprio nome propõe se trata de quando uma classe A herda os atributos e métodos de uma classe B evitando assim uma repetição de códigos, a expressão de herança também é bastante conhecida pela expressão **“é um”** .

Exemplo: pessoa e professor, todo professor é uma pessoa, logo os atributos e métodos de uma pessoa também se aplica ao professor.

1. Explique a diferença entre sobrecarga e sobrescrita de métodos. Crie um código-fonte na linguagem Java demonstrando a diferença entre os dois conceitos.

**Sobrecarga** :é quando a mesma classe tem dois ou mais métodos com o mesmo nome variando somente em seus argumentos podendo ter variações de tipos, ordem e quantidade.

**Sobrescrita** : é quando uma classe “Filha” tem um mesmo método da classe “Pai” idêntico em sua assinatura fazendo assim que o método criado na classe filha seja o que será realizado quando for chamado.

1. Analise o código-fonte do programa Java a seguir e informe quais mensagens serão impressas de acordo com a sequência de execução do programa. **OBS: não execute o código-fonte antes de ter a sua resposta, aproveite para treinar o entendimento dos conceitos.**

**public** **class** ClientePF **extends** PessoaFisica {

**private** **int** codCliente;

**public** ClientePF() {

**this**(999);

System.***out***.println("ClientePF()");

}

**public** ClientePF(**int** codCliente) {

**super**();

System.***out***.println("ClientePF("+codCliente+")");

**this**.codCliente = codCliente;

**super**.show();

}

**public** **void** show() {

System.***out***.println("Show ClientePF");

}

**public** **static** **void** main(String[] args) {

**new** ClientePF();

}

}

**class** PessoaFisica **extends** Pessoa {

**private** **int** cpf;

**public** PessoaFisica() {

**super**();

System.***out***.println("PessoaFisica() com CPF "+cpf);

show();

}

**public** **void** show() {

System.***out***.println("Show PessoaFisica");

}

}

**class** Pessoa {

**private** String nome;

**public** Pessoa() {

**this**.nome = "João";

System.***out***.println("Pessoa()");

}

}

**Resposta :**

Pessoa()

PessoaFisica() com CPF 0

Show PessoaFisica

ClientePF(0)

Show PessoaFisica

ClientePF()

1. Identifique e explique o(s) erro(s) da classe a seguir. **OBS: não execute o código-fonte antes de ter a sua resposta, aproveite para treinar o entendimento dos conceitos.**

**public** **class** DemoConstrutor {

**private** **int** a, b;

**public** DemoConstrutor() {

System.***out***.println("Sem argumentos...");

DemoConstrutor(0,0);

}

**public** DemoConstrutor(**int** xa, **int** xb) {

System.***out***.println("Com argumentos...");

a = xa; b = xb;

}

}

**Resposta:**

* No construtor vazio quando se faz necessário chamar um construtor com argumentos ele deve ficar na 1ª linha do construtor vazio.
* Ainda na mesma linha do item 1 não se faz necessário chamar o construtor com parâmetro pelo nome dele já que se está dentro de um construtor vazio, basta que se escreva o comando “this” passando os argumentos desejados . Ex. : this(0, 0);
* No construtor com parâmetro, quando se atribui os argumentos recebidos pelo construtor nas variáveis recomenda-se por boa pratica de programação que se use o “this.(nome da variável)”

1. Escreva a classe ObjetoGeometrico que representa um objeto geométrico em duas dimensões. Essa classe deve ter um construtor para inicializar o objeto e métodos para mostrar seus dados, calcular e retornar sua área e perímetro. Usando essa classe como base, escreva as classes herdeiras Circulo (contendo duas coordenadas para o centro e um raio), Retangulo (contendo dois valores para os lados) e Triangulo (contendo três valores para os lados), que sobrescrevem os métodos em ObjetoGeometrico. Dicas: A área de um círculo pode ser calculada com Math.PI\*r\*r, em que r é o raio do círculo. O perímetro de um círculo é dado por 2\*Math.PI\*r. A área do retângulo é dada por b\*h, onde b é um dos lados e h é o outro lado. Seu perímetro é dado por 2\*b+2\*h. A área de um triângulo é dada por Math.sqrt(s\*(s-a)\*(s-b)\*(s-c)), onde Math.sqrt é a função que calcula a raiz quadrada, a, b e c são os lados do triângulo, e s é a metade do perímetro do triângulo. O perímetro do triângulo é calculado como (a+b+c).

Boa sorte!

Prof. Igor.