Centro Federal de Educação Tecnológica –

**Disciplina: Programação Orientada a Objetos**

Prof. Gustavo Guedes

****

**Aluno(a): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Turma:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Data: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Utilize atributos de instância privados sempre que possível.**

**Códigos desnecessários e que reduzam o desempenho do sistema serão penalizados.**

**Utilize as boas práticas de programação.**

**LEIA AS QUESTÕES ATÉ O FINAL ANTES DE COMEÇAR.**

**Questão 1 (5.5) – Desenvolva o código conforme pedido abaixo:**

A - Crie a interface Comestivel com apenas um método denominado comer, que não possui argumentos e que não retorna nada.

B1 – Escreva uma classe Biscoito com 3 atributos: id (String), cor (String) e preco (int). Crie os getters e setters apenas se precisar. Crie em Biscoito APENAS UM construtor, que recebe o id como argumento. Implemente um método em uma classe chamada Utils com a seguinte assinatura: public static boolean existe (List x, Biscoito y); considere que a lista possui objetos do tipo Biscoito, mas podendo possuir outros objetos. Escreva esse método de forma que seja verificada a existência do objeto Biscoito representado por y na lista representada por x, retorne verdadeiro se existir e falso se não existir. Considere que dois objetos Biscoito são iguais se possuem o **mesmo id.** Não é permitida qualquer iteração para realizar esse item, ou seja, não use *for*, *iterator*, etc.

B2- Crie 2 subclasses da classe Biscoito: Negresco e Skiny. Implemente a interface Comestivel.

C1- Considere um método denominado **Utils2.getLista(): List** que retorne uma lista de objetos de diversos tipos, incluindo Biscoitos. Não crie nem a classe Utils2 nem o método getLista(), suponha que já existe. Crie uma classe principal que receba do console o id de um Biscoito. Em seguida, verifique, com base no código desenvolvido no item B, se o Biscoito existe na lista retornada por **Utils2.getLista().** Caso sim, exiba no console uma mensagem informando o id, cor e preço do Biscoito.

C2 – Ainda na classe principal, crie um método denominado transformaListEmArray que receba uma lista **k** de objetos (de diversos tipos (ex: Elefante, Carro, Conta), incluindo Biscoitos) e retorna um array de Biscoitos, contendo os objetos do tipo Biscoito na lista representada por **k**. O array DEVE ter o comprimento do número de Biscoitos em k. Ex: se em k existem 10 Biscoitos e 15 Carros, o array DEVE ter tamanho 10 e estar preenchido com o s 10 Biscoitos.

D – Em seguida, crie um método para calcular a soma de todos os valores dos preços no array de Biscoitos criado no item C2. Imprima o total no console.

**Questão 2 (2,5)** Considere um método de uma classe denominado **BancoDeDados.getFunc(): List** que retorne uma lista de aproximadamente 100 milhões de objetos do tipo String no seguinte formato: nome#peso#altura. Ex:

Aline Ferraz#79.7#1.77

Arthur Maia#70#1.65

...

Essas strings representam o nome, peso e altura de indivíduos cadastrados no SUS. Não crie a classe e nem o método BancoDeDados.getFunc(): List. Crie um programa para receber do console um nome. Esse programa deve utilizar o método getFunc() para imprimir no console o nome do individuo inserido no console, o IMC dele, o IMC médio (de todos os indivíduos) e a proporção de indivíduos com o IMC maior do que o do individuo inserido no console. Além disso, se o IMC do indivíduo for maior que 40, insira o texto: ALERTA.

Ex: nome inserido: Arthur da Silva

Saída: Arthur da Silva – 39.7 – 35 - 0.32

Ex2: nome inserido: Ana Maria

Saída: Ana Maria – 40.1 – 35 - 0.3 – ALERTA

\* O IMC é uma ferramenta usada para detectar casos de obesidade ou desnutrição, por exemplo, indivíduos com Obesidade grau 3 possuem IMC maior do que 40. O cálculo do IMC consiste no peso dividido pela altura ao quadrado (peso/altura\*altura).

**Questão 3 (2,0) Observe as classes abaixo e diga o que sai no console.**

**public** **class** Arara {

**private** String cor;

**public** **static** **int** *teste*;

**private** **int** idade;

**public** Arara() {

++*teste*;

}

**public** String getCor() {

**return** cor;

}

**public** **void** setCor(String cor) {

**this**.cor = cor;

}

**public** **int** getIdade() {

**return** idade;

}

**public** **void** setIdade(**int** idade) {

**this**.idade = idade;

}

}

**public** **class** TesteArara4 {

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

Arara x = **new** Arara();

Arara y = x;

x.setCor("Azul");

System.***out***.println(y.getCor());

y.setCor("Rosa");

y = **new** Arara();

System.***out***.println(y.getCor());

y.setIdade(25);

**new** Arara();

y.*teste* = y.*teste* + 13;

System.***out***.println(x.*teste*);

*teste1*(x, y);

System.***out***.println(x.getCor());

System.***out***.println(y.getCor());

**int** k = 9;

*teste2*(k, x.getIdade(), x, y);

System.***out***.println(x.*teste*);

y.*teste* = y.*teste* + 73;

System.***out***.println(x.*teste*);

System.***out***.println(k);

System.***out***.println(x.getCor());

System.***out***.println(x.getIdade());

}

**public** **static** **void** teste1(Arara x, Arara y) {

x.setCor("Dourada");

x = **new** Arara();

x.setCor("Creme");

y.setCor("Prata");

y = x;

x = y;

System.***out***.println(x.getCor());

}

**public** **static** Arara teste2(**int** a, **int** b, Arara y, Arara x) {

a = 8;

b=11;

x.setCor("Verde");

x.*teste* = x.*teste* + 17;

System.***out***.println("resultado: " + a + a);

y=x;

**return** x;

}

}