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.**

**Faça distinção clara entre maiúsculas e minúsculas.**

**Seja claro, formal e sucinto.**

**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 (6) – 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 no array de Biscoitos criado no item C2. Imprima o total no console.

**Questão 2 (2,0)** Considere um método de uma classe denominado **BancoDeDados.getFunc(): List** que retorne uma lista de diversos objetos do tipo String no seguinte formato: nome#idade. Ex:

Aline Ferraz#29

Arthur Maia#33

...

Essas strings representam o nome e a idade dos funcionários da empresa. Não crie o nem 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 a idade associada ao nome inserido no console e a média das idades de todos os funcionários. Como exemplo, se o nome inserido for Arthur Maia, deve sair no console:

Arthur Maia – idade: 33 – Média da idade de todos os funcionários – XXX, em que XXX representa a média da idade de todos os funcionários da lista.

**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** TesteArara {

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

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

Arara y = x;

x.setCor("Preta");

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

y.setCor("Rosa");

y = **new** Arara();

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

y.setIdade(22);

**new** Arara();

y.teste = y.teste + 15;

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

teste1(y, x);

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

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

**int** k = 9;

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

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

y.teste = y.teste + 3;

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("Azul");

y.setCor("Creme");

y = x;

x = y;

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

}

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

a = 11;

b=8;

x.setCor("Verde");

x.teste = x.teste + 7;

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

**return** x;

}

}