Centro Federal de Educação Tecnológica –

**Disciplina: Programação Orientada a Objetos**

Prof. Gustavo Guedes

****

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

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

**Seja claro, formal e sucinto.**

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

**Utilize as boas práticas de programação.**

**Utilize as boas práticas de programação, sempre que possível. Vale lembrar que, quando possível, declarar atributos de instância como privados são uma boa prática de programação.**

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

**Questão 1 (6,0) – Desenvolva o código conforme pedido abaixo:**

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

B – Crie um método em Utils com a seguinte assinatura public static void ordena (List<Pessoa> x). Esse método deve ordenar os objetos pessoas em x por ordem alfabética. Prepare a classe Pessoa para que isso ocorra corretamente. Não é permitida qualquer iteração para realizar esse item, ou seja, não use *for*, *iterator*, etc.

C – Ao utilizar o System.out.println em um objeto Pessoa, deve sair no console o cpf, nome e idade da pessoa.

D- Crie 2 subclasses da classe Pessoa: PessoaFisica e PessoaJuridica.

E - Dada a classe Utils, crie o método public Map<String, Pessoa> retornaDados(Set conjuntoPessoas) throws FormatoIncorretoException. (UTILIZE o ITERATOR nessa questão)

Considere que o conjunto recebido como argumento (conjuntoPessoas) contém Strings no seguinte formato: CPF#nome#idade#tipo. Por exemplo, considere os elementos desse conjunto como (080949343-23#Arthur Novaes Moura#34#J, 310949321-44#Erika da Silva Souza#30#F, etc.). Esses valores representam CPF, nome, idade e tipo de pessoa (Fisica ou Juridica).

Dessa maneira, implemente o método *retornaDados* de forma que seja retornado um mapa da seguinte forma: os elementos de *conjuntoPessoas* devem ser percorridos, o CPF de cada elemento é a chave do Mapa e os valores do mapa são objetos do tipo PessoaFisica ou PessoaJuridica. Resumindo, você irá criar um objeto PessoaFisica (se o último caractere da string for F) ou PessoaJuridica representando cada elemento em *conjuntoPessoas* (se o último caractere da string for J) e adicionar ao mapa. Caso algum elemento em *conjuntoPessoas* possua mais de três caracteres # ou menos de três caracteres #, lance a exceção checked FormatoIncorretoException. O formato de saída da exceção deve ser:FormatoIncorretoException: O formato da String [XXX] esta incorreto. [XXX] representa a String em conjuntoPessoas que gerou o erro. Crie a classe FormatoIncorretoException.

**Questão 2 (1.5)** – Crie uma classe Principal. Crie um conjunto no qual você irá adicionar Strings. Receba do console 1000 Strings. O usuário deve inserir Strings no seguinte formato: CPF#nome#idade#tipo. Insira essas Strings no conjunto criado. Após a leitura das 1000 Strings, invoque o método da questão 1 E (*retornaDados*). Em seguida, receba UMA String do console que representa um CPF. Verifique se existe um objeto com esse CPF no mapa retornado pelo método retornaDados.

**Questão 3 (1)**  – Crie um método estático em uma classe denominada Utils2 chamado quantidadePessoasJuridicasMaior21(Map<String, Pessoa> **m**) que retorne a quantidade de objetos PessoaJuridica com idade maior que 21 no mapa referenciado por **m**.

**Questão 4 (1,5) – Observe a questão abaixo. Compila? Se não, qual o erro? Se sim, O que sai no console?**

**public** **class** TesteArvore {

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

Arvore v1 = **new** Arvore();

System.***out***.println(v1.getNome());

v1.setNome("Acacia");

Arvore v2 = **new** Arvore();

v2.setNome("Ipe");

*teste1*(v1, v2);

System.***out***.println(v1.getNome());

System.***out***.println(v2.getNome());

v1.a = 10;

**int** x = v1.a;

v1.a=5;

v2.a = v1.a;

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

*teste2*(v1.a, x, v1);

System.***out***.println(v1.getNome());

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

System.***out***.println(v1.b);

System.***out***.println(v1.a);

System.***out***.println(v2.a);

System.***out***.println(v2.b);

*teste3*(v2);

System.***out***.println(v2.getNome());

System.***out***.println(v2.a);

}

**public** **static** **void** teste1(Arvore v2, Arvore v1) {

v1.setNome("Eucalipto");

v1 = v2;

v1.setNome("Abacateiro");

}

**public** **static** **void** teste2(**int** a, **int** x, Arvore arv) {

a = 30;

x=11;

arv.*b* = 15;

arv.setNome("Mogno");

arv = **new** Arvore();

}

**public** **static** Arvore teste3(Arvore v2) {

v2 = **new** Arvore();

v2.setNome("Macieira");

**return** v2;

}

}

**public** **class** Arvore {

**private** String nome;

**public** **int** a;

**public** **static** **int** *b*;

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

}