Centro Federal de Educação Tecnológica –
Disciplina: Programação Orientada a Objetos
Prof. Gustavo Guedes

Aluno(a): _____

Turma:_____ **Data: _____**

Códigos desnecessários e que reduzam o desempenho do sistema serão penalizados.
Utilize as boas práticas de programação, sempre que possível. Vale lembrar que você deve declarar atributos de instância como privados.
LEIA AS QUESTÕES ATÉ O FINAL ANTES DE COMECAR.

Questão 1 (7) – Desenvolva o código conforme pedido abaixo:

A1 – Um astrônomo iniciante deseja criar um sistema para buscar os Corpos Celestes que estudou e catalogou. Como não sabia Java, catalogou os Corpos Celestes em um arquivo txt, um por linha. Em seguida, pediu para você criar um sistema para buscar as estrelas catalogadas. Como você não aprendeu a ler arquivos, encontrou na internet uma classe que lê as linhas de um arquivo e retorna um conjunto com cada linha do arquivo sendo representada por uma String. Dessa maneira, quando você utiliza a classe LerArquivo e chama o método LerArquivo.retornaStrings("c:/corposCelestres.txt"), o método retorna um conjunto de Strings no formato id#nome#distancia#tipo. Note que o retorno do método é Set<String>. Nessa questão você precisa criar todas as classes necessárias para o funcionamento, menos a LerArquivo.

Escreva uma classe CorpoCeleste com 3 atributos: id (String), nome (String) e distancia (double, dado que a distância será representada em anos-luz). Crie os getters e setters apenas se precisar. Crie em CorpoCeleste APENAS UM construtor, que recebe o id como argumento.

A2 - Implemente um método em uma classe chamada Utils com a seguinte assinatura: public static boolean existe (Collection<CorpoCeleste> x, CorpoCeleste y). Escreva esse método de forma que seja verificada a existência do objeto CorpoCeleste representado por y na coleção representada por x, retorne verdadeiro se existir e falso se não existir. Considere que dois objetos CorpoCeleste são iguais se possuem o **mesmo id**. 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<CorpoCeleste> x). Esse método deve ordenar os objetos CorpoCeleste em x pela distancia (tanto faz se for em ordem crescente ou decrescente). Prepare a classe CorpoCeleste 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 uma referência a um objeto CorpoCeleste, deve sair no console o id, nome e a distancia do CorpoCeleste, ou seja, *a representação desse objeto como String*. Atente para o **idem D.**

D- Crie 2 subclasses da classe CorpoCeleste: Estrela e Planeta. Ao utilizar o System.out.println em uma referência a um objeto Estrela, deve sair no console o texto "[Estrela] " concatenado com id, nome e a distancia. No caso de Planeta, deve sair no console o texto "[Planeta] " concatenado com id, nome e a distancia. Obrigatório **reutilizar** o que foi feito no **item C.**

E - Dada a classe Utils, crie o método public Map<String, CorpoCeleste> retornaDados(Set<String> conjuntoCorpos). Utilize o ITERATOR nesse item.

Considere que o conjunto recebido como argumento (conjuntoCorpos) contém Strings no seguinte formato: id#nome#distancia#tipo. Por exemplo, considere os elementos desse conjunto como (D1586#Jupiter Quente#340#P, 553-2#Proxima Centauri#4#E, etc.). Esses valores representam id, nome, distancia e tipo do CorpoCeleste (Planeta (P) ou Estrela (E)).

Dessa maneira, implemente o método *retornaDados* de forma que seja retornado um mapa da seguinte forma: os elementos de *conjuntoCorpos* devem ser percorridos, o id de cada elemento é a chave do Mapa e os valores do mapa são objetos do tipo Planeta ou Estrela. Resumindo, você irá criar um objeto Planeta (se o último caractere da String for P) ou Estrela (se o último caractere da String for E) e adicionar ao mapa. Caso algum elemento em *conjuntoCorpos* 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 conjuntoCorpos que gerou o erro. (Isso facilita o astrônomo a saber qual corpo celeste está com problemas).

F –Crie uma classe chamada SistemaPrincipal com o método main. Em seguida, receba **do console** o ID de uma estrela que deseja verificar se está cadastrada no sistema. Essa classe deve utilizar a classe LerArquivo discutida no início da questão para ler o arquivo de corpos celestes criado pelo astrônomo. Desenvolva o resto desse item utilizando os métodos desenvolvidos nos itens anteriores para fazer o seguinte:

(f1) verifique se existe um corpo celeste com o ID inserido (utilize, obrigatoriamente, o método existe desenvolvido no item A2). Caso exista, exiba, no console, os dados do corpo celeste que o astrônomo cadastrou (id, nome e distancia), ou seja, imprima *a representação desse objeto como String;* caso tenha dúvidas como proceder, releia o item C da questão.

(f1.2) caso não exista, exiba no console a seguinte mensagem para o astrônomo.

A estrela com ID XYZ não existe. Considere que XYZ será o id inserido.

(f1.3) caso haja, em alguma string, mais de 3 caracteres # ou menos de 3 caracteres # (exemplo: D1586#Jupiter Quente), exiba no console a seguinte mensagem:

Há um problema no seu arquivo texto. O formato da String [D1586#Jupiter Quente] está incorreto.

Questão 6 (2) – Crie uma classe chamada StringUtils com um método estático chamado processa (String str); esse método deve receber uma String e retornar uma nova String toda em caixa alta (letras maiúsculas) e com as letras A, E, I, O substituídas por 4, 3,1,0 respectivamente. Dessa forma, se a String passada for “paralelismo”, o método exibirá no console a String “P4R4L3L1SM0”. Ainda nessa classe, crie um outro método capaz de inverter uma String.

Questão 3 (2) – Observe a questão abaixo. Compila? Se sim, O que sai no console?

```
public class Caneta {
	private String cor;
	private int quantidade;
	public static int x;
	public String getCor() {
		return cor;
	}
	public void setCor(String cor) {
		this.cor = cor;
```

```
	}
	public int getQuantidade() {
		return quantidade;
	}
	public void setQuantidade(int quantidade) {
		this.quantidade = quantidade;
	}
}
```

```
public class TestaCaneta {
	public static void main(String[] args) {
		Caneta c = new Caneta();
		c.setCor("rosa");
		Caneta c2 = new Caneta();
		c2.setCor("verde");
		c2.setQuantidade(3);
		Caneta c3 = new Caneta();
		c3.setCor("dourado");
		metodoCan1(c2);
		metodoCan2(c3);
		metodoCan3(c);
		c.x = 20;
		c2.x = 15;
		c3.x = 10;
		System.out.println(c.x+c2.x+c3.x);
		int i = c3.getQuantidade();
		System.out.println(i);
		System.out.println(c.getQuantidade());
		System.out.println(c2.getQuantidade());
		System.out.println(c3.getQuantidade());
		System.out.println(c.getCor());
		System.out.println(c2.getCor());
		System.out.println(c3.getCor());
		int y = 11;
		c = metodoCan4(c, y);
		System.out.println(c.getCor());
		System.out.println(c.getQuantidade());
		System.out.println(c.x);
		System.out.println(y);
	}
	public static void metodoCan1(Caneta c) {
		c.setCor("vermelho");
		c.setQuantidade(9);
		c=null;
	}
	public static void metodoCan2(Caneta c) {
		c.setQuantidade(2);
		c = new Caneta();
		c.setCor("azul");
	}
	public static void metodoCan3(Caneta c) {
		c = new Caneta();
		c.setCor("preto");
		c.setQuantidade(1);
	}
	public static Caneta metodoCan4(Caneta c, int u) {
```

```
            c=new Caneta();
            c.setCor("prata");
            u = u + 15;
            return c;
    }
}
```

BOA SORTE!