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 você deve declarar atributos de instância como privados.**

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

**Questão 1 (3,5) – Observe a classe abaixo e a saída no console. Escreva todas as classes para que o programa compile e rode sem problemas. A saída deve ser a saída apresentada no console. Algumas observações:**

1)As classes não devem ter métodos set.

2)Você deve mudar o comportamento de suas classes para que ao imprimir uma coleção de veículos, apareçam as características dos veículos no console, ao invés de xxx@12312.

3)O uso de más práticas de POO resultarão em perda de pontos ou até mesmo em anulação da questão.

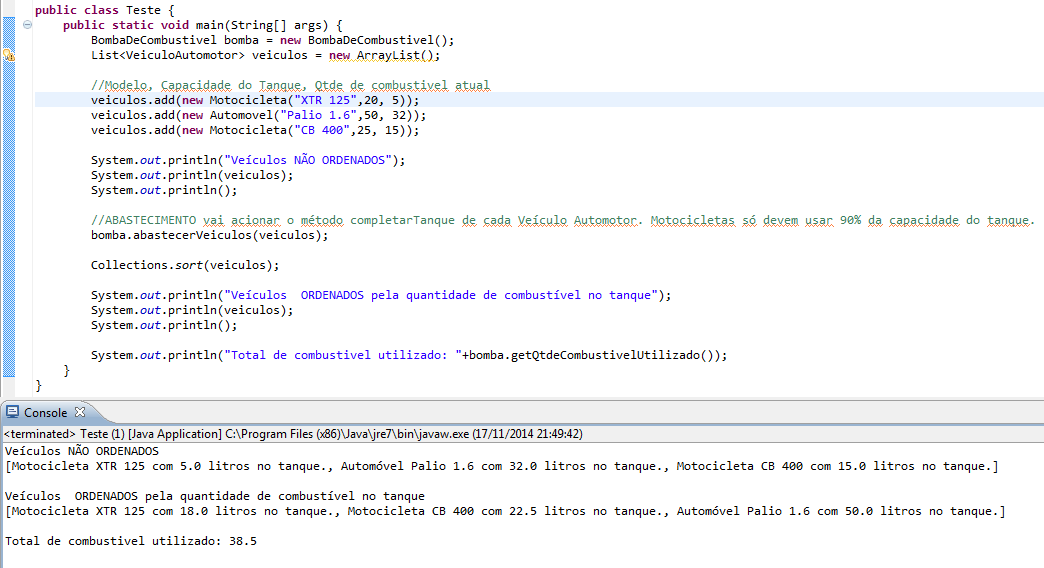
4)Observe os comentários antes de começar a escrever suas classes.

5)O método abastecerVeículos deve chamar o método completarTanque de cada veículo distinto. O tanque das Motocicletas deve ser utilizado em apenas 90% de sua capacidade.

6) Considere o segundo parâmetro do construtor como a capacidade máxima do veículo. O terceiro é a quantidade de combustível inicial.

7) Cuide para que nossa coleção possa ser ordenada pela quantidade de combustível no tanque.

8) Faça com que a classe VeiculoAutomotor não seja instanciável.



**Questão 2 (1) – Dado o método public int contaCaractere (String x, char y), implemente este método de forma que seja retornado quantas vezes ocorre o caractere passado para variável y na String x.**

**Questão 2 (4,0) – Desenvolva o código conforme pedido abaixo:**

A – Escreva uma classe Pessoa com 3 atributos: cpf, nome e idade (defina os tipos). Crie os getters e setters apenas se precisar. Implemente um método em uma classe chamada Utils com a seguinte assinatura: public 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 e a mesma idade**. Não é permitida qualquer iteração para realizar esse item, ou seja, não use *for*, *iterator*, etc.

B - Dada a classe Utils, crie o método public Map<String, Pessoa> retornaDados(Set<String> conjuntoPessoas) throws FormatoIncorretoException.

Considere que o conjunto recebido como argumento (conjuntoPessoas) contém Strings no seguinte formato: CPF#nome#idade. Por exemplo, considere os elementos desse conjunto como (080949343-23#Arthur Novaes Moura#34, 310949321-44#Erika da Silva Souza#30, etc.).

Dessa maneira, implemente o método *retornaDados* de forma que seja retornado um mapa da seguinte forma: os elementos de *conjuntoPessoas* devem ser percorridos e o CPF de cada elemento é a chave do Mapa e os valores do mapa são objetos do tipo PessoaResumindo, você irá criar um objeto Pessoa representando cada elemento em *conjuntoPessoas* e adicionar ao mapa. Caso algum elemento em *conjuntoPessoas* possua mais de dois caracteres # ou menos de dois 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.

**Questão 3 (1,5) – Observe a questão abaixo. Compila? Se não, qual o erro? Se sim, O que sai no console?**

**public** **class** Caneta {

**private** String cor;

**private** **int** quantidade;

**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** Principal {

**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);

**int** i = c3.getQuantidade();

*qtd*(i++);

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());

}

**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** **void** qtd(**int** i) {

++i;

i++;

}

}

**BOA SORTE!**