Parte 2 - 12.5 valores

Considere as seguintes definições de classes de uma aplicação que tem como objectivo gerir uma frota de veículos de aluguer. A aplicação da *UberUM* possui a informação dos veículos nela registados e em cada veículo dever-se-á guardar a informação respeitante às viagens que efectuou. O preço das viagens depende do tipo de veículo que é alocado a cada uma das viagens, bem como da distância a percorrer entre a origem e o destino.

Considere os seguintes excertos de código:

```
public abstract class Veiculo implements Comparable<Veiculo>, Serializable {
    private String matricula;
    private String marca;
    private String modelo;
    private double precokm;  // preço base por km
    ...

    public abstract float custoViagem(float distancia);

}

public class VeiculoLuxo extends Veiculo implements Comparable<VeiculoLuxo>  {
    private float taxaLuxo;
    ...

    public float custoViagem(float distancia) {
        return distancia * getPrecoKm() * (1.1 + this.taxaLuxo);
    }
    ...
}

public class Autocarro extends Veiculo implements Comparable<Autocarro> {
    private int lotacao; //lotação máxima do autocarro
    ...

    public float custoViagem(float distancia) {
        if (this.lotacao > 10)
            return distancia * getPrecoKm() * 0.5/this.lotacao;
        else
            return distancia * 0.75 * this.lotacao;
    }
    ...
}

public class Viagem implements Serializable {
  private String origem;  // local de origem da viagem
  private String destino;  // local de destino
  private float distância;
  private float custo; // valor do custo da viagem
  ...
  public Viagem(String origem, String destino, float distancia, float custo) {...}
  ...
}
```

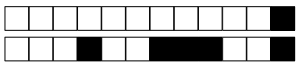
Responda às perguntas seguintes:

Questão 1

Efectue a declaração das variáveis de instância de `UberUM` e complete a declaração das variáveis de instância de `Veículo`. Justifique brevemente a escolha que faz dos tipos de dados das variáveis de instância que declara.

Codifique também o construtor parametrizado `public UberUM(Set<Map.Entry<String,Veiculo>> info)`, assumindo que estamos numa estratégia de composição.

☐0 ☐.1 ☐.2 ☐.3 ☐.4 ☐.5 ☐.6 ☐.7 ☐.8 ☐.9 ☐1



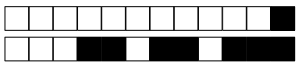
Questão 2 Desenhe o Diagrama de Classes, considerando que a estratégia utilizada é de composição. Seja o mais exaustivo possível na descrição das relações existentes entre as diversas classes.

☐0 ☐.1 ☐.2 ☐.3 ☐.4 ☐.5 ☐.6 ☐.7 ☐.8 ☐.9 ☐1

Questão 3 Caso não tenha feito a questão 6 considere que a classe `UberUM` possui internamente como estrutura de dados um Map de matrícula para Veículo e responda às questões seguintes assumindo esta definição de estado.

Codifique o método `public Viagem viagemMaisAntiga(String matricula) throws...`, da classe `UberUM`, que devolve a primeira viagem efectuada pelo veículo de matrícula fornecida. Note que deverá prever o facto de o veículo existir e dever possuir viagens efectuadas.

☐0 ☐.1 ☐.2 ☐.3 ☐.4 ☐.5 ☐.6 ☐.7 ☐.8 ☐.9 ☐1



Questão 4 Codifique o método `public void adicionaViagem(String matricula, String origem, String destino, float distancia) throws...`, da classe `UberUM`, que guarda mais uma viagem associada ao veículo identificado no parâmetro. Note que será necessário calcular o valor da viagem.

☐0 ☐.1 ☐.2 ☐.3 ☐.4 ☐.5 ☐.6 ☐.7 ☐.8 ☐.9 ☐1

Questão 5 Considere que se pretende obter um Map que associa a cada marca a lista dos veículos dessa marca ordenados por ordem crescente do valor cobrado nas viagens e como segundo factor pela ordem alfabética decrescente da designação do modelo (`public Map<String,List<Veiculo» porMarcaPorValor()`). Codifique todas as declarações necessárias.

☐0 ☐.1 ☐.2 ☐.3 ☐.4 ☐.5 ☐.6 ☐.7 ☐.8 ☐.9 ☐1

Questão 6

Considere que temos uma implementação de uma nova classe `TukTuk`, com a seguinte declaração:

```
public class TukTuk  {
    private int lotacao; // numero de lugares
    private String condutor; //nome do condutor
    private Viagem[] viagensEfectuadas;
    private int numViagens; //número viagens efectuadas

        ...
/**
 * método que adiciona uma nova viagem ao TukTuk.
 * Está devidamente implementado.
 */
public void adicionaViagem(Viagem v) throws NaoAceitaMaisViagensException {
    if (this.numViagens == this.viagensEfectuadas.length)
       throw new NaoAceitaMaisViagensException();
    else
       this.viagensEfectuadas[this.numViagens++] = v.clone();
}

...
}
```

Não dispomos do código desta classe, pelo que não podemos alterá-la, mas queremos ter na empresa de veículos os TukTuk (apesar de terem um número limitado de viagens). Diga como é que podemos compatibilizar esta classe com o resto das classes existentes, sabendo que cada vez que tentarmos adicionar uma viagem a um TukTuk que já não aceita mais viagens o objecto `Viagem` deve ser gravado num ficheiro de texto com o nome igual à concatenação da origem com o destino.

0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1

Para o seu exame, use preferencialmente documentos compilados do auto-multiple-choice.