

 Ecole Supérieure Privée d'Ingénierie et de Technologies		Année Universitaire : 2016-2017 Examen Java	
Module : Conception orientée objet et programmation Java		Documents autorisés : Non	
Enseignants : M Nader RAHMAN , M Sofien GHARBI Mlle Sana BEN FADHEL		Nombre de pages : 4	
Date : 26/10/2016	Heure : 11H15	Durée : 1h30	
Classes : 4info B		Session : PRINCIPALE	

GESTION SERVICE DES MINES

L'agence tunisienne du transport terrestre en partenariat avec l'école Esprit souhaite développer une application qui permet de gérer la liste des contrôleurs de la visite technique et l'ensemble des véhicules attribués à ce dernier qui ont effectués une visite technique. Pour réaliser notre application, nous vous demandons de compléter les classes présentées ci-dessous.

Les parties à compléter sont numérotées de 1 à 18.

Vous devez mettre dans vos copies le numéro correspondant à chaque code ajouté //TODO XX :

Travail demandé :

En précisant le numéro de chaque partie à ajouter :

- 1) Complétez la classe **Vehicule** avec les méthodes **equals**, **hashCode** et **toString**, deux véhicules sont égaux en cas d'égalité de leurs id et marque. // **TODO 1 // 2 points**
- 2) Un Véhicule est peut être soit **Voiture** ou **Camion**. Une voiture est caractérisée par l'année de mise en circulation et un camion est caractérisé par son prix.
- 3) Implémenter la classe **VehiculeException**. // **TODO 2 // 1 point**
- 4) Complétez la classe **ListeVisite** regroupant un ensemble de Vehicule dans une **List**. Complétez les méthodes de cette classe. // **TODO 3 -> TODO 12 // 9.5 points**
- 5) Complétez la classe **ServiceDesMines** qui regroupe pour chaque **Contrôleur**(voir annexe) l'ensemble de ses **Vehicules** dans une **map**. // **TODO 13 -> TODO 18 // 7.5 points**

Remarque 1 : On suppose que la classe **Contrôleur** contient déjà les méthodes **equals**, **hashCode** et **toString**. Deux contrôleurs sont égaux en cas d'égalité de leur id et leur nom.

Remarque 2 : Toutes les classes sont présentées dans l'ANNEXE ci-dessous.

ANNEXE :

<pre>public class Vehicule { private int idVehicule; private String marque; private boolean visited; public Vehicule(int idVehicule, String marque, boolean visited) { this.idVehicule = idVehicule; this.marque = marque; this.visited = visited; } public Vehicule() { } public int getIdVehicule() { return idVehicule; } public void setIdVehicule(int idVehicule) { this.idVehicule = idVehicule; } public String getMarque() { return marque; } public void setMarque(String Marque) { this.marque = marque; } public boolean isVisited() { return visited; } public void setVisited(boolean visited) { this.visited = visited; } // TODO 1 - à compléter // /*equals, hashCode et toString */ (1 point) (0,5 point) (0,5 point) }</pre>	<pre>public class Voiture extends Vehicule { private int annee; public Voiture(int annee, int idVehicule, String marque, boolean visited) { super(idVehicule, marque, visited); this.annee = annee; } public int getAnnee() { return annee; } public void setAnnee(int annee) { this.annee = annee; } }</pre>
	<pre>public class Camion extends Vehicule { private float prix; public Camion(float prix, int idVehicule, String marque, boolean visited) { super(idVehicule, marque, visited); this.prix = prix; } public float getPrix() {return prix; } public void setPrix(float prix) { this.prix = prix; } }</pre>
	<pre>public class Controleur { private int id; private String nom; public Controleur(int id, String nom) { this.id = id; this.nom = nom; } public int getId() {return id;} public void setId(int id) { this.id = id; } public String getNom() {return nom;} public void setNom(String nom) { this.nom = nom; } }</pre>

```

import java.sql.*;
import java.util.*;

public class ListeVisite {
    List<Vehicule> lst;

    public ListeVisite() {
        // TODO 3 - à compléter //(0,5 point)
    }

    public boolean ajouterVehicule(Vehicule v) throws VehiculeException {
        // TODO 4 - à compléter //(1,5 point)
        Cette méthode permet de vérifier si le véhicule a effectué une visite
        technique avant de l'ajouter dans la liste, le test sera fait en utilisant
        l'attribut visited de la classe véhicule
    }

    public boolean supprimerVehicule(Vehicule v) throws VehiculeException {
        // TODO 5 - à compléter //(1 point)

    }

    public void afficherVehicules() {
        // TODO 6 - à compléter //(1,5 points)

    }

    public void trierVehiculeParMarque() {
        // TODO 7 - à compléter en utilisant en expression Lambda //(2 points)

    }

    public int add(Vehicule v) // TODO 8 - à compléter //(0,5 point) {
        Cette méthode stocke les véhicules dans une base de données nommée « visite »
        ayant comme login « root » et pwd « root » et la table « t_vehicule » ayant
        comme colonne l'id auto incrément, la marque varchar(20) et visited boolean.

        String req = "insert into t_vehicule (marque,visited) values // TODO 9 - à compléter //(0,5
        point) ;

        // Connexion à la base - TODO 10 - à compléter //(1 points)
        // préparation de la requête - TODO 11 - à compléter //(0,5 point)
        // Exécution de la requête - TODO 12 - à compléter //(0,5 point)
    }
}

```

```

import java.util.*;

public class ServiceDesMines {
    Map<Controleur, ListeVisite> map ;
    String nom;

    public ServiceDesMines(String nom) {
        // TODO 13 - à compléter // (0,5 point)
    }

    public void ajouterVisite(Controleur in, Vehicule v) throws VehiculeException{
        // TODO 14 - à compléter // (1,5 point)
    }

    public void afficherLesControleursEtListeVehicules() {
        // TODO 15 - à compléter // (1 point)
    }

    public float calculRecette() {
        // TODO 16 - à compléter // (1,5 points)
        Retourne la recette totale qui sera calculée comme suit :
        Si le véhicule est une voiture, le prix de la visite est 22.800 Dt
        Si le véhicule est un camion, le prix de la visite est 40 Dt
    }

    public int isNew() {
        // TODO 17 - à compléter // (1,5 point)
        Retourne le nombre de véhicule de type voiture qui ont effectués une
        visite dont le nombre d'année de circulation par rapport à l'année 2016 est
        inférieur à 10.
    }

    public void nbVehiculesParControleur(Contrôleur C) {
        // TODO 18 - à compléter // (1,5 point)
        Afficher le nombre de voiture ainsi que le nombre de camion qui ont passé une
        visite technique pour un contrôleur donné
    }
}

```