esprit Se former autrement HONORIS UNITED UNIVERSITIES	# *EXAMEN* Semestre : 2 Session : Principale
ÉTUDIANT(e) Nom et Prénom : ……... Classe: …….....	Code (administration) :

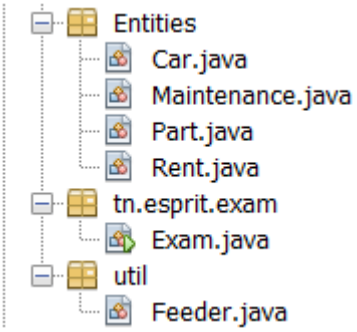
✂ ----------

Module : Programmation Orientée Objet Java et Application
Enseignant(s) : Équipe Java
Classe(s) : 3B, 3IA
Documents autorisés : NON Nombre de pages : XXXXXX
Calculatrice autorisée : NON Internet autorisée : NON
Date : 02/06/2022 Heure . 10:30 Durée : 1:30h

Pendant un entretien d'embauche, le responsable technique vous donne un programme à terminer avec quelques instructions, respectez les pour espérer être pris dans son équipe.

Le programme en question est pour une agence de location de voitures, il comporte plusieurs classes et plusieurs paquets comme suit:

- Un package “Entities” qui contient toutes les classes entités:
  - Car.java (représente une voiture)
  - Maintenance.java (représente une maintenance)
  - Part.java (représente une piece de rechange)
  - Rent.java (représente une location de voiture)
- Un package “tn.esprit.exam” qui contient la classe principale:
  - Exam.java (contient la méthode main)
- Un package “util” qui contient une classe de données
  - Feeder.java (contient une méthode qui retourne la liste de voitures



Vous êtes demandé de suivre les instructions (TODO) pour terminer les bouts de code qui manquent

⚠Attention, Veuillez bien lire les instructions avant de répondre !⚠

⚠Attention, on suppose que tous les getters et setters sont déjà implémentés(sauf si on vous le demande) !⚠

⚠Attention, on suppose que tous les equals et toString sont déjà implémentés (sauf si on vous le demande) !⚠

NE RIEN ECRIRE ICI

✂-

Maintenance.java

```
public class Maintenance {

  private String description;
  private float laborCost;
  private double maintenanceDate;

  public Maintenance(String description, float laborCost, double maintenanceDate) {
    this.description = description;
    this.laborCost = laborCost;
    this.maintenanceDate = maintenanceDate;
  }
  ...
```

Part.java

```
public class Part {

  private String name;
  private float unitPrice;

  public Part(String name, float unitPrice) {
    this.name = name;
    this.unitPrice = unitPrice;
  }
  ...
```

Feeder.java

```
public class Feeder {

  public static List < Car > getOldCarsDatas() {
    Car c1 = new Car("Mercedes", "S class", 84000);
    ...

    ArrayList < Car > cars = new ArrayList < > ();
    cars.add(c1);
    cars.add(c2);
    ...
    return cars;
  }

}
```

NE RIEN ECRIRE ICI

✂----------

Car.java

TODO 1 :créer un constructeur pour initialiser les attributs || utiliser Lambda Expression pour les maintenances et les locations

```
public class Car {

  private String brand;
  private String model;
  private float price;

  Map < Maintenance, List < Part >> maintenances; // Toujours triés par maintenanceDate (voir TODO 1)
  Set < Rent > rents; //Toujours triés par rentDate (voir TODO 1)

    //TODO 1
  public Car(String brand, String model, float price) {
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
  }
```

TODO 2 : ajouter maintenance à la voiture

```
  //TODO 2
  public void addMaintenance(Maintenance maintenance) {
    ..........
    ..........
  }
```

NE RIEN ECRIRE ICI

✂----------

TODO 3 : ajouter une pièce à la maintenance || ATTENTION la maintenance peut ne pas exister

```
//TODO 3
public void addPartToMaintenance(Maintenance maintenance, Part part) {
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
}
```

TODO 4 : redéfinir la méthode toString pour afficher tous les attributs d'une voiture

```
    //TODO 4
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
```

TODO 5 : la méthode retourne tous les revenus de location d'une voiture (parcours ou Stream : 2 opérations)

```
    //TODO 5
    public double getAllRentsCosts() {
    ..........
    ..........
    ..........
    ..........
    ..........
    }
```

TODO 6 : La méthode renvoie tous les frais de la voiture (entretiens et pièces) (Stream :

NE RIEN ECRIRE ICI

✂----------

2 opérations)

```
//TODO 6
public double getAllCarCharge() {
  return this.maintenances.entrySet().stream()
    .mapToDouble((entry) -> {
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    })
    ..........
    ..........
}
```

Rent.java

TODO 7.1 : redéfinir la méthode equals en se basant sur le “rentDate” et le “cost”

TODO 7.2 : faire en sorte que la méthode “setCost” déclenche une exception de type

NE RIEN ECRIRE ICI

✂----------

"NegatifCostException"(préalablement définie) si le "cost" en paramètre est négatif

```
public class Rent {

  private double rentDate;
  private float cost;

  public Rent(double rentDate, float cost) {
    this.rentDate = rentDate;
    this.cost = cost;
  }
  //TODO 7.1
  @Override
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........

  //TODO 7.2
  public void setCost(float cost) ..................................... {
    ..........
    ..........
    ..........
    ..........
    ..........
    ..........
  }
  ...
```

NE RIEN ECRIRE ICI

✂----------

Exam.java

TODO 8 : retourner la valeur totale de la flotte de voitures (Stream : 2 opérations)

```
public class Exam {

  public static void main(String[] args) {

      //TODO 8
      Double totalValue = Feeder.getOldCarsDatas().stream()
        ..........
        ..........
        ..........
        ..........
```

TODO 9 : retourne un arrayList de voitures dont la valeur est supérieure à 21000 (Stream : 2 opérations)

```
      //TODO 9
      List < Car > carTreeSet = Feeder.getOldCarsDatas().stream()
        ..........
        ..........
        ..........
        ..........
```

TODO 10 : afficher la marque (en MAJUSCULES) de la première voiture qui présente un déficit entre le coût de la location(les gains) et les frais d'entretien(maintenance et pièces). (ASTUCE : UTILISEZ LE TODO 5 ET LE TODO 6) (Stream : 3 opérations)

```
      //TODO 10:
      Feeder.getOldCarsDatas().stream()
        ..........
        ..........
        ..........
        ..........
```

Bon travail