Nom :

Prénom :

DS POO 2017-2018

2ème session

L'objectif du sujet est de réaliser un programme permettant à une agence de transport de gérer ses colis.

Il est indispensable de lire le sujet dans son ensemble avant de commencer à répondre.

Le sujet comporte plus de points que nécessaire pour obtenir la note maximale.

Des bonus sont attribués lorsque vous traitez correctement une partie complète.

Le code partiel de ce programme est donné en fin de sujet, il vous appartient de le compléter en fonction des questions. Les questions sont indépendantes mais il est préférable de les traiter dans l'ordre.

Les étoiles (*, **, ***) indiquent le niveau de difficulté de la question (facile, moyen, difficile).

1 Cahier des charges :

Une agence de transport dispose de moyens de transport permettant d'acheminer des colis.

Un moyen de transport est capable de transporter un ou plusieurs types de colis.

Ces colis ont différentes caractéristiques, qui affectent leur mode de transport, les délais de livraison et le coût.

Dans ce sujet, nous traiterons uniquement 3 caractéristiques possibles, mais il peut y en avoir d'autres :

- colis nécessitant un transport frigorifique ;
- colis nécessitant un transport urgent ;
- colis transportés en mode économique.

Un colis peut combiner plusieurs caractéristiques : frigorifique ET urgent, urgent ET économique, etc.

Par défaut, un colis est transporté en mode économique (pour simplifier).

Il est possible qu'une agence ne dispose pas de moyen de transport pour exépédier un colis (dans la classe Test, l'agence n'a pas de transport frigorifique.

2 Analyse des solutions possibles (16 points)

Une première solution consisterait à faire une hiérarchie de Colis (ColisUrgent, ColisUrgentEtFroid, etc.).

2.1 Pourquoi cette solution n'est-elle pas souhaitable ? (2 pt) *

Une autre solution consisterait à utiliser des attributs booléens dans la classe Colis, indiquant si un mode de transport doit être utilisé :

```
1  public class Colis
2  {
3      private boolean froid;
4      private boolean urgent ;
5      private boolean eco;....
```

2.2 Pourquoi cette solution n'est-elle pas souhaitable non plus ? (2 pt) *

La solution retenue (cf.code donné) consiste à associer aux modes de transport et aux colis des comportements (ComportementColis).

2.3 Dessinez l'organisation des classes et des interfaces du code donné. Distinguez les abstractions (classes abstraites, interfaces) des classes concrêtes (2 pt) *

2.4 Quelle est l'utilité de l'organisation en packages et sous-packages ? (1pt) *

2.5 A quoi sert ComportementConcret ? Pourquoi implémente-t-elle Comparable(3pt) **

2.6 Pourquoi utilise-t-on un Set comme attribut `comportementsPossibles` dans Agence ? (1pt) **

2.7 Pourquoi déclare-t-on des List et des Set et non pas des TreeSet et des ArrayList ? (2pt) **

2.8 Quel peut être l'intérêt que Colis implémente ComportementColis, alors que cette classe contient déjà un attribut listant les comportements possible pour le colis (3pt) ***

3 Codage du Modèle (26 points)

Les espaces laissés vides sont à compléter. Leur taille devrait suffire pour contenir votre code.

Parfois, seule une petite partie de cet espace est nécessaire.

Il faut respecter les import déclarés (pas d'autre import nécessaires).

Le modèle contient les classes qui modélisent le problème sans gérer la partie IHM.

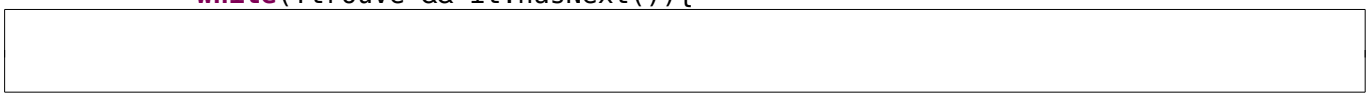
LISEZ BIEN LES COMMENTAIRES QUI DONNENT DES INDICES SUR CE QUI DOIT ÊTRE CODÉS ! Certaines partie du code sont très faciles, d'autre beaucoup plus dures !

3.1 Classe Agence (8 points) */**

```
6  package ds4eti2018_2S.modele;
7
8  import java.util.HashMap;
9  import java.util.Iterator;
10 import java.util.List;
11 import java.util.ArrayList;
12 import java.util.Map;
13 import java.util.Set;
14 import java.util.TreeSet;
15
16 import ds4eti2018_2S.modele.comportements.ComportementColis;
17 import ds4eti2018_2S.modele.transport.MoyenTransport;
18
19 public class Agence {
20     private String nom;
21     private String ville;
22     private Set<ComportementColis> comportementsPossibles = new
23 TreeSet<ComportementColis>();
24     private List<MoyenTransport> transports;
25     private List<Colis> lesColis = new ArrayList<Colis>();
26
27     public Agence(String nom, String adresse, List<MoyenTransport> transports){
28       this.nom=nom;
29       this.ville=adresse;
30       this.transports=transports;
31       if(transports!=null){majComportements();}
32     }
33 //constructeur ne donnant pas de liste de moyen de transport
34     public Agence(String nom, String adresse){
                                                                                
                                                                                
                                                                                
                                                                                
35     }
36     private void majComportements() {
37 //ajouter les comportements des transport à la liste des comportements possibles
38       for(MoyenTransport m:transports){
                                                                                
                                                                                
                                                                                
39       }
40     }
41
42     public boolean addColis(Colis c){
43       boolean ok=true;
44       Iterator<ComportementColis> it = c.getCaracteristiques().iterator();
```

```
      while(ok && it.hasNext()){
//vérifier que l'agence a bien les compétences pour prendre en charge le colis
```

```
      }
      if(ok){lesColis.add(c);}
      return ok;
    }

    public void affecteChargement(){
      for(Colis c:lesColis){
            boolean trouve=false;
            Iterator<MoyenTransport> it = transports.iterator();
            while(!trouve && it.hasNext()){
```



```
            }
      }
    }

    public double calculPrix(int km){
      double prix=0;
      for(Colis c:lesColis){prix+=c.getPrix(km);}
      return prix;
    }

    public String toString(){
      String cmp ="";
      for(ComportementColis cc:comportementsPossibles){
            cmp+="\t"+cc+"\n";
      }
      return "Agence "+nom+ " située à "+ville+"\n"+cmp;
    }
}
```

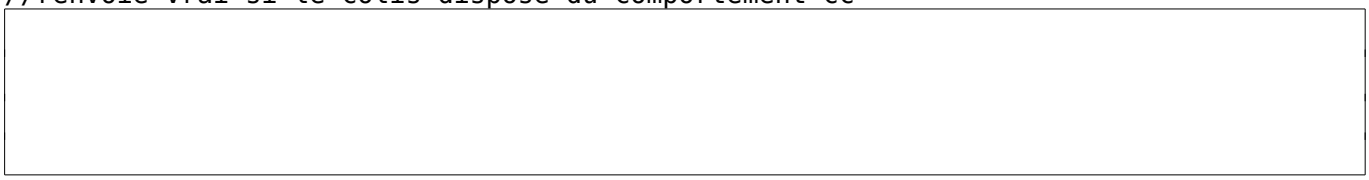
3.2 Classe Colis (8 points) */**/***

```
package ds4eti2018_2S.modele;

import java.util.ArrayList;
import java.util.List;
import ds4eti2018_2S.modele.comportements.ComportementColis;

public class Colis implements ComportementColis{

//un attribut à trouver pour gérer l'incrémentation automatique des numéros de colis
```

```
    private List<ComportementColis> caracteristiques = new
ArrayList<ComportementColis>();
    private int numero;

```

```
//le constructeur ne prends pas de numéro : à chaque création de Colis, le numéro
est obtenu en ajoutant 1 au numéro du colis précédemment créé
    public Colis(List<ComportementColis> caracteristiques){
```

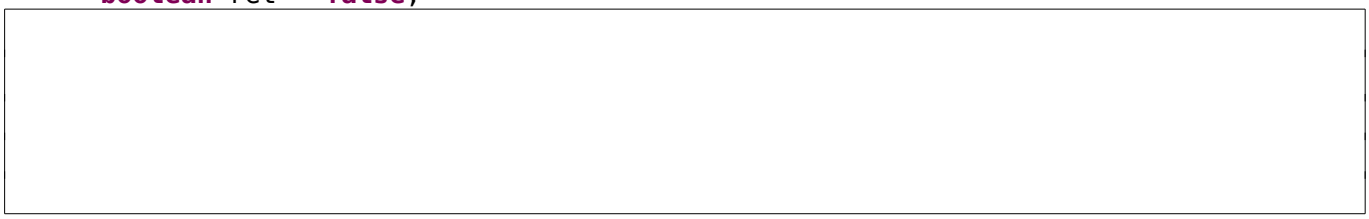
```
    }
    public boolean hasComportement(ComportementColis cc) {
//renvoie vrai si le colis dispose du comportement cc
```



```
    }

    public List<ComportementColis> getCaracteristiques() {
      return caracteristiques;
    }

    public String toString(){
      String info="";
      for(ComportementColis cc:caracteristiques){info+=cc+",";}
      return "Colis n°"+numero+" ("+info+")";
    }

    @Override
    public double getPrix(int km) {
      double ret=0;
      for(ComportementColis c:caracteristiques)
      {
            ret+=c.getPrix(km);
      }
      return ret*km;
    }

    @Override
    public int getDelai() {
      int ret=0;
      for(ComportementColis c:caracteristiques)
      {
            if(ret>0 && c.getDelai()<ret){ret = c.getDelai();}
            else{ret = c.getDelai();}
      }
      return ret;
    }

}
```

3.3 MoyenTransport (3 points) **

```
131 package ds4eti2018_2S.modele.transport;
132
133 import java.util.ArrayList;
134 import java.util.List;
135
136 import ds4eti2018_2S.modele.Colis;
137 import ds4eti2018_2S.modele.comportements.ComportementColis;
138
139 public class MoyenTransport {
140     private List<ComportementColis> comportementsGeres = new
141 ArrayList<ComportementColis>();
142     private String immatriculation;
143     private List<Colis> chargement = new ArrayList<Colis>();
144
145     public MoyenTransport(String immatriculation, List<ComportementColis> cmp){
146       this.immatriculation = immatriculation;
147       comportementsGeres = cmp;
148     }
149
150     //ajoute le colis uniquement si le moyen de transport a bien TOUS les
151 comportements permettant de gérer le colis.
152     public boolean addColis(Colis c){
153       boolean ret = false;
```

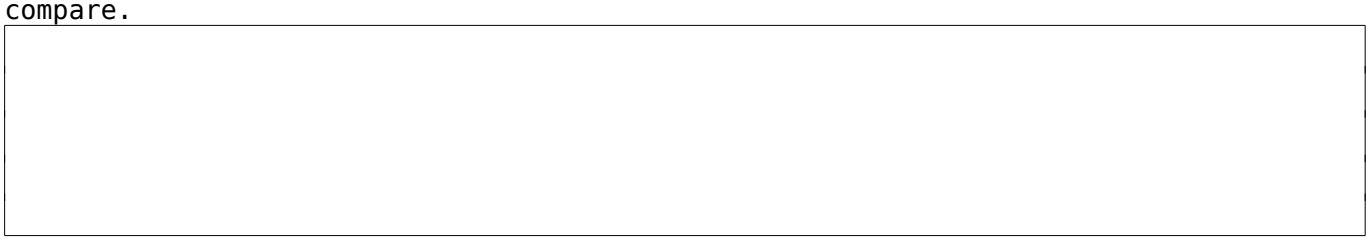


```
154       return ret;
155     }
156
157     public List<ComportementColis> getComportementsGeres(){return
158 comportementsGeres;}
159
160     public String toString(){
161       String ret = "Chargement du véhicule "+immatriculation;
162       for(Colis c:chargement){
163             ret+="\n\t"+c;
164       }
165       return ret;
166     }
167 }
168
```

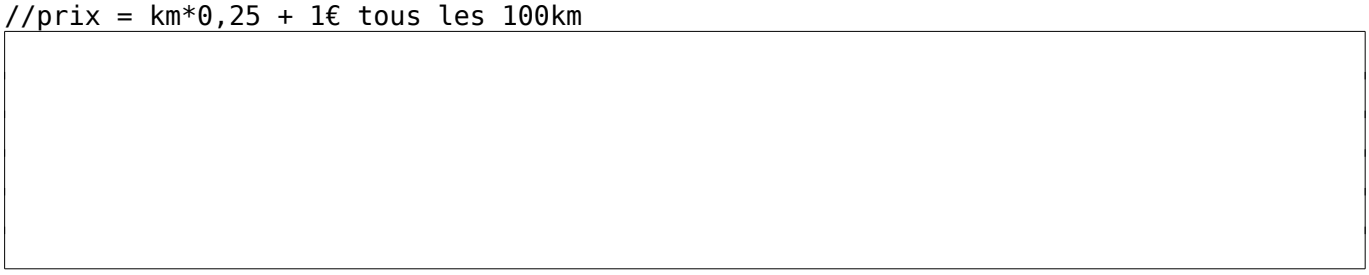
3.4 Comportements (3 points) *

```
169 package ds4eti2018_2S.modele.comportements;
170
171 public interface ComportementColis {
172     public double getPrix(int km);
173     public int getDelai();
174
175 }
176
177 package ds4eti2018_2S.modele.comportements;
178
179 public abstract class ComportementConcret implements ComportementColis,
180 Comparable<ComportementColis> {
181
182     public boolean equals(Object o){
```

```
      return this.getClass().equals(o.getClass());
    }
    public int hashcode(){
      return this.getClass().getSimpleName().hashCode();
    }

    public int compareTo(ComportementColis c){
//les comportements sont comparés en fonction de la comparaison du nom de leur
classe. Deux instances de comportement ayant la même classe sont égaux lorsqu'on les
compare.
```

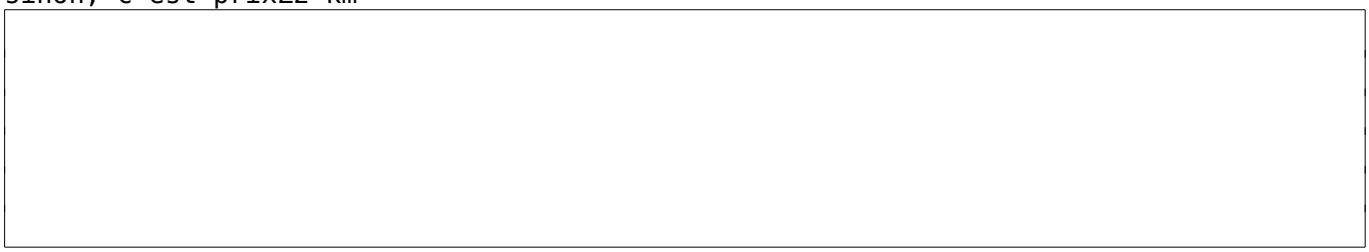


```
    }

    public String toString(){return this.getClass().getSimpleName();}

}

package ds4eti2018_2S.modele.comportements;

public class CmpEco extends ComportementConcret {

    @Override
    public double getPrix(int km) {
      return km*0.1;
    }

    @Override
    public int getDelai() {
      return 3;
    }

}

package ds4eti2018_2S.modele.comportements;

public class CmpFroid extends ComportementConcret {

    @Override
    public double getPrix(int km) {
//prix = km*0,25 + 1€ tous les 100km
```



```
    }

    @Override
    public int getDelai() {
      return 2;
    }

}
```

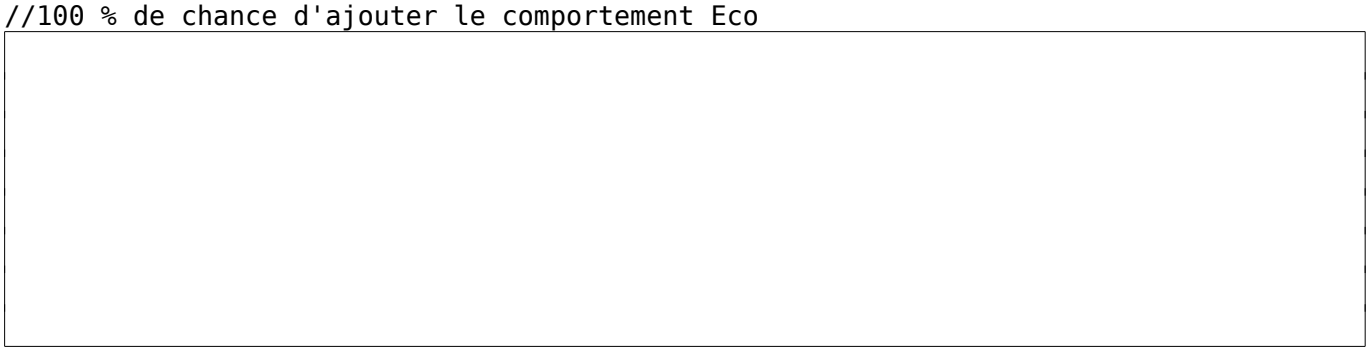
```
package ds4eti2018_2S.modele.comportements;

public class CmpUrgent extends ComportementConcret {

    private static final double prixZ1 = 0.5;
    private static final double prixZ2 = 1;
    private static final int zone1 = 30;

    @Override
    public double getPrix(int km) {
      double prix = 0;
    //le prix dépend de la zone : si km est inférieur à zone1, le prix = prixZ1*km.
sinon, c'est prixZ2*km
```



```
    }

    @Override
    public int getDelai() {
      return 1;
    }

}
```

3.5 Classe de Test (4 points) **

```
package ds4eti2018_2S.modele;
import java.util.ArrayList;
import java.util.List;

import ds4eti2018_2S.modele.comportements.CmpEco;
import ds4eti2018_2S.modele.comportements.CmpFroid;
import ds4eti2018_2S.modele.comportements.CmpUrgent;
import ds4eti2018_2S.modele.comportements.ComportementColis;
import ds4eti2018_2S.modele.transport.MoyenTransport;

public class Test {

    public static List<Colis> fabriqueColis(int nb){
      List<Colis> liste = new ArrayList<Colis>();
      for (int i=0;i<nb;i++){
//50 % de chance d'ajouter un comportement Urgent
//50 % de chance AUSSI d'ajouter un comportement Froid
//100 % de chance d'ajouter le comportement Eco
```



```
      }
      return liste;}
```

```
    public static void main(String[] args) {
      ComportementColis c1 = new CmpUrgent();
      ComportementColis c2 = new CmpFroid();
      ComportementColis c3 = new CmpEco();

      List<ComportementColis> lT1=new ArrayList<ComportementColis>();
      List<ComportementColis> lT2=new ArrayList<ComportementColis>();
      List<ComportementColis> lT3=new ArrayList<ComportementColis>();

      lT1.add(c1);
      lT1.add(c3);
      lT2.add(c3);
      lT3.add(c1);
      lT3.add(c3);
      List<MoyenTransport> transports = new ArrayList<MoyenTransport>();
      transports.add(new MoyenTransport("1234FZ21",lT1));
      transports.add(new MoyenTransport("4578AZ12",lT2));
      transports.add(new MoyenTransport("BF214",lT3));

      Agence a = new Agence("Mon Agence","Lyon", transports);
      System.out.println(a);

      List<Colis> l = fabriqueColis(20);
      for(Colis c:l){
            if(a.addColis(c)){System.out.println("Colis "+c +" ajouté");}
            else{System.out.println("Colis "+ c + "non pris en charge");}
      }

      a.affecteChargement();
      System.out.println("Prix du transport pour 25km: "+a.calculPrix(25));
      System.out.println("Prix du transport pour 185km: "+a.calculPrix(185));

    }

}
```

3.6 Exemple de Sortie de Test

```
Agence Mon Agence située à Lyon
    CmpEco
    CmpUrgent

Colis Colis n°1 (CmpUrgent,CmpFroid,CmpEco,)non pris en charge
Colis Colis n°2 (CmpFroid,CmpEco,)non pris en charge
Colis Colis n°3 (CmpUrgent,CmpEco,) ajouté
Colis Colis n°4 (CmpUrgent,CmpFroid,CmpEco,)non pris en charge
Colis Colis n°5 (CmpEco,) ajouté
Colis Colis n°6 (CmpFroid,CmpEco,)non pris en charge
Colis Colis n°7 (CmpUrgent,CmpFroid,CmpEco,)non pris en charge
Colis Colis n°8 (CmpFroid,CmpEco,)non pris en charge
Colis Colis n°9 (CmpUrgent,CmpEco,) ajouté
Colis Colis n°10 (CmpUrgent,CmpFroid,CmpEco,)non pris en charge
Colis Colis n°11 (CmpUrgent,CmpEco,) ajouté
Colis Colis n°12 (CmpEco,) ajouté
Colis Colis n°13 (CmpFroid,CmpEco,)non pris en charge
Colis Colis n°14 (CmpUrgent,CmpEco,) ajouté
Colis Colis n°15 (CmpFroid,CmpEco,)non pris en charge
Colis Colis n°16 (CmpUrgent,CmpEco,) ajouté
Colis Colis n°17 (CmpEco,) ajouté
Colis Colis n°18 (CmpUrgent,CmpFroid,CmpEco,)non pris en charge
Colis Colis n°19 (CmpUrgent,CmpFroid,CmpEco,)non pris en charge
Colis Colis n°20 (CmpUrgent,CmpFroid,CmpEco,)non pris en charge
Prix du transport pour 25km: 2062.5
Prix du transport pour 185km: 198505.0
```

4 Interface graphique (15 points)

4.1 (6 points) Proposez une maquette (dessin) d'interface graphique SIMPLE permettant :

1) d'ajouter un colis et ses caractéristiques
2) d'affecter les colis à des transports valables
3) d'afficher la liste des colis répartis dans les transports

Vous indiquerez le nom des composants swing que vous utilisez, ainsi que le fonctionnement des gestionnaires de positionnement.

4.2 Écrivez les parties de code impliquant des Listener (ajout, implémentation) (9 points)

5 Évolutions (8 points)

5.1 Comment prendre en compte la notion de capacité maximale pour un moyen de transport (par exemple, pas plus de 12 colis pour M1, 8 pour M2, etc.) (4 points)

5.2 On souhaite ajouter une fonctionnalité permettant de dresser le liste des colis qui n'ont pas été affecté par l'agence à des transports, faute de compétence pour ce type de colis. Expliquez comment prendre en compte cette fonctionnalité (4 points)