2I002 - Éléments de programmation par objets avec Java

Examen du 19 décembre 2014 - Durée : 2 heures

Tous documents interdits, Téléphones portables éteints

Le barème sur 60 points est donné à titre indicatif :
Partie A : Questions de cours (20 points)
Partie B : Problème (40 points)

Partie A : Questions de cours (20 points)

Question 1 (3 points) : Soient les classes suivantes d'un jeu d'échecs :

```
1 public abstract class Piece {
2         public abstract void afficher();
3 }
4 public class Tour extends Piece {
5         public void afficher() { System.out.println("Je suis une tour"); }
6 }
7 public class Cavalier extends Piece {
8         public void afficher() { System.out.println("Je suis un cavalier"); }
9 }
```

On suppose que l'on se trouve dans la méthode `main` d'une classe `TestEchiquier`.

- Déclarez la variable `echiquier` de type tableau de `Piece` à deux dimensions de 8 cases sur 8 cases.
- Ajoutez une tour dans la première colonne de l'échiquier, le numéro de la ligne étant choisi aléatoirement (rappel : `Math.random()` retourne un nombre entre 0 et 1 (non-compris)).
- Ajoutez un cavalier sur la première ligne, deuxième colonne de l'échiquier.
- Faites s'afficher chacune des pièces présentes sur l'échiquier (on suppose que d'autres pièces peuvent se trouver sur l'échiquier).

```
1 Piece [][] echiquier=new Piece [8][8];
2 echiquier[(int)(Math.random()*echiquier.length)][0]=new Tour();
3 echiquier[0][1]=new Cavalier();
4 for(int i=0;i<echiquier.length;i++) {
5         for(int j=0;j<echiquier[i].length;j++) {
6                 if (echiquier[i][j]!=null) {
7                         echiquier[i][j].afficher();
8                 }
9         }
10 }
```

Question 2 (4 points) : Soient les classes suivantes :

```
1 public class Point {
2         private int x,y;
3         public Point(int x,int y) { this.x=x;this.y=y; }
4         public Point somme(Point p) { return new Point(x+p.x,y+p.y); }
5         public String toString() { return "("+x+","+y+")"; }
6 }
7 public class TestInstanciation {
8         public static void main(String [] args) {
9                 Point p1=new Point(1,2);
10                Point p2=new Point(5,7);
11                Point p3=p1;
```

```
12                  p2=p1;
13                  Point p4=p3.somme(p2);
14                  System.out.println(p4);          }  }
```

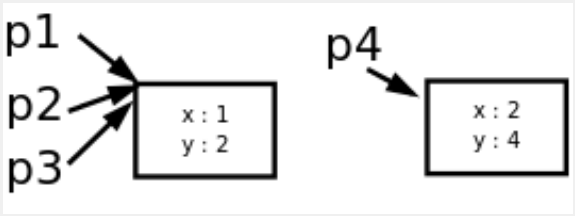
Q2.1 : Combien y-a-t-il de variables de la classe `Point` déclarées dans la méthode `main` ?

Il y a 4 variables : `p1, p2, p3, p4`

Q2.2 : Combien y-a-t-il d'instances de la classe `Point` créées ? Donner le numéro des lignes où sont créées chacune de ces instances.

Il y a 3 objets créés aux lignes 9, 10 et 4 (appelée à la ligne 13)

Q2.3 : Faire un schéma des objets dans la mémoire à la ligne 14.



Remarque : à la ligne 14, le deuxième objet créé (5,7) n'existe plus, car plus aucune variable ne le référence, le ramasse-miette (*garbage collector*) a libéré la mémoire pour cet objet.

Q2.4 : Qu'affiche ce programme ?

Il affiche la somme du point référencé par `p1` avec lui-même, c'est-à-dire il affiche `(2,4)`.

Question 3 (4 points) : Soient les classes suivantes :

```
1 public class Appareil {
2     public String toString() { return "Appareil"; }
3 }
4 public class Television extends Appareil {
5     public String toString() { return "Television"; }
6 }
7 public class TeleTNT extends Television { }
8 public class AppareilInformatique extends Appareil { }
9 public class Ordinateur extends AppareilInformatique { }
10 public class Test {
11     public static void main(String [] args) {
12         Ordinateur ordi=new Ordinateur();System.out.println(ordi.toString());
13         TeleTNT tvTNT=new TeleTNT();System.out.println(tvTNT.toString());
14
15         Appareil a1=new Ordinateur();System.out.println(a1.toString());
16         Appareil a2=new TeleTNT();System.out.println(a2.toString());
17
18         AppareilInformatique ai1=new TeleTNT();
19         AppareilInformatique ai2=new Ordinateur();
20         AppareilInformatique ai3=new Appareil();
```

```
21
22          AppareilInformatique ai4=ordi;
23          AppareilInformatique ai5=(AppareilInformatique)ordi;
24          AppareilInformatique ai6=a1;
25          AppareilInformatique ai7=(AppareilInformatique)a1;
26
27          Appareil ap1=new TeleTNT();
28          Appareil ap2=new Ordinateur();
29          Television tv1=(Television)ap1;
30          Television tv2=(Television)ap2;                                    } }
```

Q3.1 : (a) Qu'affiche les lignes 12 et 13 ? Expliquez comment est déterminée la méthode `toString()` qui est appelée. (b) Quelle est la différence avec l'affichage des lignes 15 et 16 ? Expliquez.

(a) ligne 12 affiche `Appareil`, ligne 13 affiche `Television`
Explication : s'il existe une méthode `toString()` dans la classe de l'objet crée, c'est cette méthode qui est appelée, sinon la méthode appelée est la méthode `toString()` de la classe mère, sinon on remonte dans la hiérarchie d'héritage jusqu'à ce qu'une méthode `toString()` soit trouvée.

(b) La ligne 15 affiche la même chose que la ligne 12 (resp. la ligne 16 affiche la même chose que la ligne 13), car les objets crées sont de même type, même si le handle n'est pas le même. Or la méthode `toString()` appelée est celle du type de l'objet réel (l'objet qui a été créé par le `new`), et ne dépend pas du type du handle.

Q3.2 : Parmi les lignes 18 à 20, lesquelles sont correctes, lesquelles sont fausses. Expliquez chaque erreur.

Ligne 18 fausse : une télévision TNT n'est pas un appareil informatique
Ligne 19 correcte
Ligne 20 fausse : un appareil n'est pas un appareil informatique

Q3.3 : Parmi les lignes 22 à 25, lesquelles ne compilent pas ? Expliquez chaque erreur.

1 erreur : Echec à la compilation de la ligne 24, car un appareil n'est pas un appareil informatique
Les lignes 22 et 23 compilent, car un ordinateur est un appareil informatique.
La ligne 25 compile grâce au cast, pas d'erreur à l'exécution car `a1` référence un objet de type réel `Ordinateur` (voir ligne 15) qui est un appareil informatique.

Q3.4 : Parmi les lignes 29 et 30, lesquelles ne compilent pas ? lesquelles provoquent une erreur lors de l'exécution du programme ? Expliquez chaque erreur.

Les deux lignes compilent. La ligne 30 provoque une erreur à l'exécution car l'objet référencé par `ap2` est de type réel `Ordinateur` (voir ligne 28). Or un `Ordinateur` n'est pas par héritage une `Television`. La référence `tv2` de type `Television` ne peut donc pas être utilisée pour atteindre un objet de type réel `Ordinateur`.

Question 4 (3 points) : Soient les classes `Date`, `RendezVous` et `Test` suivantes :

```
1 public class Date {
2       private int jour, mois, annee;
3       public Date(int jour, int mois, int annee) {
4             this.jour=jour; this.mois=mois; this.annee=annee;
5       }
6       public String toString() {return jour+"/"+mois+"/"+annee;}
7 }
8 public class RendezVous {
9       private Date d;
10      private String description;
11      public RendezVous(Date d, String desc) {
12            this.d=d; description=desc;
13      }
14      public String toString() { return d.toString()+" : "+description; }
15 }
16 public class Test {
17      public static void main(String [] args) {
18            RendezVous rv=new RendezVous(new Date(19,12,2014),"Cours Java");
19      } }
```

Dans un agenda, on a parfois le même rendez-vous qui revient régulièrement. Par exemple, le cours de Java est tous les jeudis. On propose de gérer ce genre de problème en utilisant des copies (clonage ou constructeur par copie) de rendez-vous. Pour simplifier, on veut créer une copie d'un rendez-vous qui soit décalé d'un jour (on ne gèrera pas le problème de dépassemenent du nombre de jours du mois (si `jour=31` alors `jour+1=32` est accepté)).
On souhaite créer une copie de l'objet de type `RendezVous` référencé par `rv` (ligne 18) (la date de la copie du rendez-vous sera décalée d'un jour), donnez précisément toutes les méthodes et instructions qu'il faut ajouter, y compris les instructions nécessaires dans la méthode `main`.

Solution 1 : constructeur par copie

```
1 // Dans la classe Date
2 public Date(Date d) {
3     this.jour=d.jour+1;  this.mois=d.mois; this.annee=d.annee;
4 }
5 // Dans la classe RendezVous
6 public RendezVous(RendezVous rv) {
7     this(new Date(rv.d),rv.description);
8 }
9 // Dans le main
10 RendezVous copie=new RendezVous(rv);
```

Solution 2 : clonage

```
1 // Dans la classe Date
2 public Date clone() {
3     return new Date(jour+1,mois,annee);
4 }
5 // Dans la classe RendezVous
6 public RendezVous clone() {
7     return new RendezVous(d.clone(),description);
8 }
9 // Dans le main
10 RendezVous copie=rv.clone();
```

Question 5 (4 points) : Une adresse IP est un numéro d'identification qui est attribué à chaque appareil relié à un réseau. On considère un réseau privé où tous les ordinateurs ont une adresse IP de la forme : `192.168.0.X` où `X` est un nombre entre 0 et 255. On veut écrire un programme qui vérifie qu'une adresse IP est valide. Dans notre cas (et pour simplifier), une adresse IP est valide si le quatrième nombre de l'adresse est compris 0 et 255, invalide sinon. Si l'adresse n'est pas valide, alors on veut que le programme lève l'exception : `BadIPException`. Voici ci-après le début du programme sans le traitement des exceptions.

```
1 public class AdresseIP {
2     private int [] tab=new int [4];
3     public AdresseIP(int x) { tab[0]=192;tab[1]=168;tab[2]=0;tab[3]=x; }
4     public void verif() // a ecrire
5     public String toString(){return tab[0]+"."+tab[1]+"."+tab[2]+"."+tab[3];}
6 }
7 public class TestAdresseIP {
8     public static void main(String [] args) {
9         AdresseIP ad=new AdresseIP(Integer.parseInt(args[0]));
10        ad.verif();
11        System.out.println("L'adresse IP "+ad+" est valide");          } }
```

Q5.1 : Ecrire une classe `BadIPException` qui possède un constructeur qui a un seul paramètre de type `AdresseIP`. Par exemple, pour l'adresse IP invalide suivante : 192.168.0.350, le message retourné par l'exception doit être : `Adresse IP invalide : 192.168.0.350`.

```
1 public class BadIPException extends Exception {
2         public BadIPException(AdresseIP ad) {
3                 super("Adresse IP non valide : "+ad);
4         }
5 }
```

Q5.2 : Ecrire la méthode `verif()` pour qu'elle lève l'exception `BadIPException` quand l'adresse est invalide. La gestion de cette exception sera déléguée à la méthode appellante.

```
1 public void verif() throws BadIPException {
2         if ( tab[3] < 0 || tab[3] > 255 ) {
3                 throw new BadIPException(this);
4         }
5 }
```

Q5.3 : Ré-écrire sur votre feuille la méthode `main` pour qu'elle puisse capturer l'exception `BadIPException` et gérer correctement l'affichage (ce programme doit afficher soit que l'adresse IP est valide soit qu'elle n'est pas valide, mais pas les deux ensembles).

```
1 try {
2         ad.verif();
3         System.out.println("L'adresse IP "+ad+" est valide");
```

```
4 } catch(BadIPException bipe) {
5         System.out.println(bipe);
6 }
```

Question 6 (2 points) : A quoi sert la classe (a) `File` ? (b) `FileInputStream` ?

(a) La classe `File` permet de gérer les fichiers :
- test d'existence
- distinction fichier/répertoire
- copie/effacement...

(b) D'après le nom de la classe : `InputStream`=>lecture d'octets, `File`=> dans un fichier.
La classe `FileInputStream` permet de lire des octets dans un fichier.

Partie B : Un problème de recyclage (40 points)

Remarque 1 : Les informations données ne sont pas le reflet de la réalité. Ceci est un examen d'informatique, et non pas un examen sur le recyclage.
Remarque 2 : Les variables seront déclarées privées et les méthodes publiques (sauf si cela est demandé explicitement).

On souhaite créer un programme qui calcule le poids des déchets recyclés et non-recyclés par un centre de recyclage. Les déchets papiers et les déchets plastiques sont des déchets recyclables. Les déchets recyclables sont des déchets. Une ville possède des camions poubelles et est associée à un centre de recyclage. Les camions poubelles ramassent tous les déchets (recyclables ou non) et les amènent au centre de recyclage qui s'occupe ensuite de les trier pour les recycler si possible.

Un déchet recyclable ne peut pas toujours être recyclé par le centre de recyclage. Par exemple, un papier s'il est souillé ne peut plus être recyclé ; autre exemple, certains types de plastiques ne sont pas encore recyclés par le centre de recyclage. Soit τ le taux de recyclage d'un déchet recyclable et p le poids du déchet, alors seulement une partie du poids du déchet peut être recyclé :
– un papier aura un poids recyclé de $\tau \times p$ si le papier n'est pas souillé, de 0 sinon,
– un plastique aura un poids recyclé de $\tau \times p$ si le plastique peut être recyclé, de 0 sinon.

Voici un exemple d'affichage du programme :

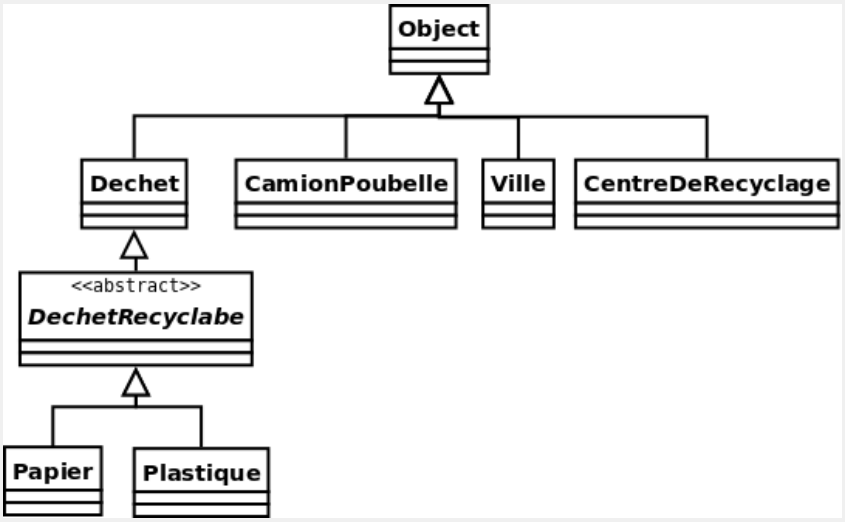
```
Affichage de 3 exemples de déchets :
- Dechet de poids 0.64 kg
- Plastique de type PET de poids 2.12 kg, de taux technique de recyclage 52.0%,
     de poids recycle 1.10 kg et de poids non recycle 1.02 kg
- Papier souille de poids 0.68 kg, de taux technique de recyclage 63.0%,
     de poids recycle 0.0 kg et de poids non recycle 0.68 kg

Ville nbCamions=2 maxDechets=2500
     Camion CP10001 (maxDechets=1000)
     Camion CP10002 (maxDechets=1500)
Centre de recyclage : poidsRecycle=0 kg poidsNonRecycle=0 kg
Ramassage des dechets
CamionPleinException: le camion CP10001 est plein
Tri des dechets du camion Camion CP10001 (maxDechets=1000) contient
      actuellement 1000 déchets pour un poids total de 2008 kg
```

```
CamionPleinException: le camion CP10002 est plein
Tri des dechets du camion Camion CP10002 (maxDechets=1500) contient
      actuellement 1500 déchets pour un poids total de 2928 kg
Centre de recyclage : poidsRecycle=748 kg poidsNonRecycle=4188 kg
```

On a bien le poids total des déchets : 2008+2928 égal à la somme du poids des déchets recyclés et des déchets non-recyclés : 748+4188.

Question 1 (2 points) : Donnez le schéma de la hiérarchie d'héritage Java des classes : `Dechet`, `DechetRecyclable`, `Papier`, `Plastique`, `CamionPoubelle`, `Ville`, `CentreDeRecyclage` et `Object`.



Question 2 (4 points) : Un déchet a un poids. Écrire la classe `Dechet` qui contient au moins les variables et méthodes suivantes :

- `poids` : le poids du déchet (`double`),
- `nbDechets` : le nombre de déchets créés depuis le début du programme,
- un constructeur qui prend en paramètre le poids du déchet,
- un constructeur sans paramètre qui initialise aléatoirement le poids du déchet entre 0 et 5 (non-compris). Ce constructeur **doit** appeler le constructeur à 1 paramètre. Aide : utiliser `Math.random()` qui retourne un nombre entre 0 et 1 (non-compris).
- l'accesseur de la variable `poids`,
- l'accesseur de la variable `nbDechets`,
- une méthode `toString()`. Par exemple, pour un déchet de 0.64 kg, cette méthode retourne `"Dechet de poids 0.64 kg"`.

Remarque : cette classe n'est pas abstraite.

```
1 public class Dechet {
2         private static int nbDechets=0;
3         private double poids;
4         public Dechet(double poids) {
5                 this.poids=poids;
6                 nbDechets++;
```

```
7         }
8         public Dechet() {
9                 this(Math.random()*5); // entre 0 et 5 non compris
10        }
11        public double getPoids() {
12                return poids;
13        }
14        private static int getNbDechets() {
15                return nbDechets;
16        }
17        public String toString(){
18                return "Dechet de poids "+poids+" kg";
19        }   }
```

Question 3 (1 points) : Donner une instruction qui permet d'afficher le nombre de déchets créés depuis le début du programme (en supposant qu'on se trouve dans le `main` de la classe `TestRecyclage`).

```
1 System.out.println("Nombre de dechets crees : "+Dechet.getNbDechets());
```

Question 4 (6 points) : Un déchet recyclable est un déchet qui possède un taux de recyclage. Écrire la classe `DechetRecyclable` qui contient au moins les variables et méthodes suivantes :

- `tauxRecyclage` : le taux de recyclage du déchet (`double`). On demande que cette variable d'instance soit déclarée protégée et qu'une fois qu'elle a été initialisée on ne puisse plus modifier sa valeur.
- un constructeur qui prend en paramètre le taux de recyclage du déchet, et initialise aléatoirement le poids du déchet entre 0 et 3 (non-compris).
- une méthode **abstraite** `getPoidsRecycle()` qui retourne le poids correspondant à la partie du déchet qui a pu être recyclée,
- une méthode non-abstraite `getPoidsNonRecycle()` qui retourne la différence entre le poids du déchet et le poids recyclé du déchet,
- une méthode `toString()`. Par exemple, pour un déchet dont le poids est de 2.12 kg, et le taux de recyclage de 52%, cette méthode doit retourner : `"de poids 2.12 kg, de taux technique de recyclage 52.0%, de poids recycle 1.10 kg et de poids non recycle 1.02 kg"`.

La méthode `getPoidsRecycle()` est abstraite, la classe `DechetRecyclable` doit donc être déclarée abstraite.

```
1 public abstract class DechetRecyclable extends Dechet {
2         protected final double tauxRecyclage; // protected final
3
4         public DechetRecyclable(double tauxRecyclage) {
5                 super(Math.random()*3); // entre 0 et 3 non compris
6                 this.tauxRecyclage=tauxRecyclage;
7         }
8         public abstract double getPoidsRecycle(); // abstract
9
```

```
10          public double getPoidsNonRecycle() {
11                  return getPoids()-getPoidsRecycle();
12          }
13          public String toString() {
14                  return "de poids "+getPoids()+" kg, de taux technique de
                       recyclage "+tauxRecyclage+"%, de poids recycle "+
                       getPoidsRecycle()+" kg et de poids non recycle "+
                       getPoidsNonRecycle()+" kg";
15          }
16 }
```

Question 5 (5 points) : Écrire la classe `Papier` qui contient au moins :
- `souille` : une variable de type `boolean`,
- un constructeur sans paramètre qui initialise le taux de recyclage du déchet à 63%. Le papier est souillé dans 30% des cas,
- une méthode `toString()`. Par exemple, pour un papier souillé de poids 0.68 kg, cette méthode doit retourner : `"Papier souille de poids 0.68 kg, de taux technique de recyclage 63.0%, de poids recycle 0.0 kg et de poids non recycle 0.68 kg"`.

```
1 public class Papier extends DechetRecyclable {
2          private boolean souille;
3
4          public Papier() {
5                  super(63);
6                  souille=Math.random()<0.30; // 30% des cas
7          }
8          public double getPoidsRecycle() {
9                  if (souille) {
10                         return 0;
11                 } else {
12                         return tauxRecyclage/100*getPoids();
13                 }
14         }
15         public String toString() {
16                 String s="";
17                 if (souille) {
18                         s+=" souille";
19                 }
20                 return "Papier"+s+" "+super.toString();
21         }
22 }
```

Question 6 (4 points) : Soit la classe `InfoPlastique` qui donne certaines informations sur les déchets plastiques. Elle indique notamment que les types de plastique qui sont pour l'instant recyclés sont le "PET", le "PEHD", et le "PP", les autres plastiques ne sont pas recyclés.

```
1 public class InfoPlastique {
```

```
2     private final static String[] typesPlastiques={"PET","PEHD","PP","Autres"};
3     public static String getTypePlastique() {
4         return typesPlastiques[(int)(Math.random()*typesPlastiques.length)];
5     }
6     public static boolean estRecycle(String typePlastique) {
7         return ( ! typePlastique.equals("Autres") ) ; // tous sauf Autres
8     }   }
```

Écrire la classe `Plastique` qui contient au moins les variables et méthodes suivantes :
- `type` : une variable de type `String` indiquant le type du plastique,
- un constructeur sans paramètre qui initialise le taux de recyclage du déchet à 52% et qui initialise le type du plastique en le choisissant aléatoirement à l'aide de la méthode `getTypePlastique()` de la classe `InfoPlastique`.

```
1 public class Plastique extends DechetRecyclable {
2         private String type;
3         public Plastique() {
4                 super(52);
5                 type=InfoPlastique.getTypePlastique();
6         }
7         public double getPoidsRecycle() {
8                 if (InfoPlastique.estRecycle(type)) {
9                         return tauxRecyclage/100*getPoids();
10                } else {
11                        return 0;
12                }
13        }
14 }
```

Question 7 (2 points) : Dans la classe `CamionPoubelle` (question suivante), il y a une méthode `ajouterDechet(Dechet d)` qui ajoute des déchets dans le camion poubelle. On veut que si le camion est plein, cette méthode lève l'exception `CamionPleinException`, cette exception sera ensuite capturée dans une méthode de la classe `Ville`. Ecrire la classe `CamionPleinException` avec un constructeur qui prend en paramètre un message.

```
1 public class CamionPleinException extends Exception {
2         public CamionPleinException(String msg) { super(msg); }
3 }
```

Question 8 (7 points) : Un camion poubelle a un identifiant. Il contient un nombre d'emplacements maximal pour stocker les déchets. A tout moment, il contient un certain nombre de déchets. Écrire la classe `CamionPoubelle` qui contient au moins les variables et méthodes suivantes :
- `cpt` : une variable de type "compteur" initialisée à 10000,
- `idCP` : un identifiant de type `String`. Le premier camion aura l'identifiant CP10001, le deuxième CP10002, le troisième CP10003 et ainsi de suite...
- `tab` : un tableau de `Dechet`,
- `nbDechets` : le nombre de déchets présents dans le camion,

- un constructeur prenant en paramètre le nombre maximal de déchets que peut contenir le camion,
- une méthode `ajouterDechet (Dechet d)` qui ajoute le déchet `d` dans le tableau et qui lève l'exception `CamionPleinException` si il n'y a plus de place dans le camion. Par exemple, pour le camion CP10001, le message de l'exception sera : `le camion CP10001 est plein`.
- une méthode dont la signature est : `public Dechet [] vider()` dont l'objectif est de vider le camion poubelle de ses déchets. Cette méthode doit retourner la référence du tableau de déchets à vider. Pour cela, cette méthode crée d'abord une nouvelle variable appelée `sauvegarde` qui référence le tableau de déchets à vider, puis crée un nouveau tableau de déchets de même taille et l'affecte à la variable `tab`, puis remet le nombre de déchets du camion à zéro, et enfin retourne la sauvegarde.

```
1 public class CamionPoubelle {
2     private static int cpt=10000;
3     private String idCP; // identifiant camion poubelle
4     private Dechet [] tab;
5     private int nbDechets; // nb dechets actuellement dans la poubelle
6     public CamionPoubelle(int max) {
7         tab=new Dechet[max];
8         cpt++;
9         idCP="CP"+cpt;
10        nbDechets=0;
11    }
12    public void ajouterDechet (Dechet d) throws CamionPleinException {
13        if ( nbDechets >= tab.length ) {
14           throw new CamionPleinException("le camion "+idCP+" est plein");
15        } else {
16           tab[nbDechets]=d;
17           nbDechets++;
18        }
19    }
20    public Dechet [] vider() {
21        Dechet [] sauvegarde=tab;
22        tab=new Dechet[tab.length];
23        nbDechets=0;
24        return sauvegarde;
25    }
26 }
```

Question 9 (4 points) : Dans notre modélisation, un centre de recyclage trie tous les déchets. Écrire la classe `CentreRecyclage` qui contient au moins les variables et méthodes suivantes :

- `poidsRecycle` : la somme des poids des déchets recyclés par le centre de recyclage,
- `poidsNonRecycle` : la somme des poids des déchets non-recyclés par le centre de recyclage,
- un constructeur sans paramètre qui initialise les deux variables à zéro,
- une méthode `trier` qui prend en paramètre un camion poubelle. Cette méthode crée une variable `tabDechets` de type tableau de `Dechet` qui est initialisée avec le retour de la méthode `vider` de la classe `CamionPoubelle`. Puis pour chaque déchet (non `null`) :
  - si le déchet est un déchet recyclable alors on ajoute le poids recyclé de ce déchet à la somme des poids des déchets recyclés, et le poids non-recyclé de ce déchet à la somme des poids des déchets non-recyclés,
  - sinon on ajoute le poids du déchet à la somme des poids des déchets non-recyclés,

– une méthode `toString()`. Par exemple, pour un centre de recyclage qui a recyclé 748kg de déchets, et qui n'a pas recyclé 4188kg de déchets, cette méthode retourne : `Centre de recyclage : poidsRecycle=748 kg poidsNonRecycle=4188 kg`. On veut que les valeurs affichées des poids soient des entiers. Pour cela, utiliser la méthode statique `format` de la classe `String` qui, par exemple, si on veut 2 chiffres après la virgule, s'utilise ainsi : `double nombre=3.14159; String chaine=String.format("pi=%.2f",nombre);` La variable `chaine` vaut `"pi=3.14"`.

```
1 public class CentreRecyclage {
2     private double poidsRecycle, poidsNonRecycle;
3
4     public CentreRecyclage() { poidsRecycle=0; poidsNonRecycle=0; }
5     public void trier(CamionPoubelle cp) {
6         System.out.println("Tri des dechets du camion "+cp);
7         Dechet [] tabDechets=cp.vider();
8         for(int i=0;i<tabDechets.length;i++) {
9             if ( tabDechets[i] != null ) {
10                if (tabDechets[i] instanceof DechetRecyclable) {
11                    DechetRecyclable dr=(DechetRecyclable)tabDechets[i];
12                    poidsRecycle+=dr.getPoidsRecycle();
13                    poidsNonRecycle+=dr.getPoidsNonRecycle();
14                } else {
15                    poidsNonRecycle+=tabDechets[i].getPoids();
16                }
17            }
18        }
19    }
20    public String toString() {
21        return String.format("Centre de recyclage : poidsRecycle=%.0f kg
              poidsNonRecycle=%.0f kg",poidsRecycle,poidsNonRecycle);
22    }
23 }
```

Question 10 (4 points) : Une ville contient une liste de camions poubelles et un centre de recyclage. Écrire la classe `Ville` qui contient au moins les variables et méthodes suivantes :
– `listeCP` : une liste au sens `ArrayList` de camions poubelles (voir la documentation en annexe)
– `cr` : un centre de recyclage,
– un constructeur qui prend en paramètre le centre de recyclage dont dépend la ville,
– une méthode `ajouterCamionPoubelle` qui ajoute un camion poubelle dans la liste,
– une méthode dont la signature est : `public Dechet creerDechet()` qui retourne un `Dechet`. Cette méthode a (aléatoirement) 25% de chance de retourner un papier, 25% un plastique et 50% un déchet (non-recyclable).
– une méthode `remplirCamionPoubelle` qui prend en paramètre un camion poubelle et qui ajoute des déchets dans le camion tant que le camion n'est pas plein,
– une méthode `ramasserEtTrierPoubelles` sans paramètre qui, pour chaque camion poubelle, le remplit de déchets, puis trie les déchets (voir l'affichage après `Ramassage des déchets`).

```
1 import java.util.ArrayList;
```

```
2 public class Ville {
3     private ArrayList<CamionPoubelle> listeCP=new ArrayList<CamionPoubelle>();
4     private CentreRecyclage cr;
5
6     public Ville(CentreRecyclage cr) {
7         this.cr=cr;
8     }
9     public void ajouterCamionPoubelle(CamionPoubelle cp){
10        listeCP.add(cp);
11    }
12    public Dechet creerDechet() {
13        double alea=Math.random();
14        if (alea<0.5) {
15            return new Dechet();
16        } else if(alea<0.75) {
17            return new Papier();
18        } else {
19            return new Plastique();
20        }
21    }
22    public void remplirCamionPoubelle(CamionPoubelle cp) {
23        try {
24            while(true) { cp.ajouterDechet(creerDechet()); }
25        } catch(CamionPleinException cpe) {
26            System.out.println(cpe.toString());
27        }
28    }
29    public void ramasserEtTrierPoubelles() {
30        System.out.println("Ramassage des dechets");
31        for(CamionPoubelle cp : listeCP) {
32            remplirCamionPoubelle(cp);
33            cr.trier(cp);
34        }
35    }
36 }
```

Question 11 (1 points) : On suppose que l'on est dans la méthode `main` de la classe `TestRecyclage`. Donner les instructions pour créer un centre de recyclage, une ville avec deux camions, puis ramasser et trier les poubelles de cette ville.

```
1 Ville v=new Ville(new CentreRecyclage());
2 CamionPoubelle cp1=new CamionPoubelle(1000); v.ajouterCamionPoubelle(cp1);
3 CamionPoubelle cp2=new CamionPoubelle(1500); v.ajouterCamionPoubelle(cp2);
4 v.ramasserEtTrierPoubelles();
```

Annexe : L'utilisation de la classe `ArrayList` nécessite de préciser le type `E` des objets qui sont dans la liste. Pour cela, on indique le type des objets entre `<...>`. Voici un extrait de cette classe :

- `ArrayList<E> ()` Crée une liste vide ; les objets insérés devront être de classe E
- `boolean add(E e)` Ajoute l'objet `e` en fin de la liste
- `E get(int index)` Retourne l'objet à la position `index` dans la liste
- `int size()` Retourne le nombre d'objets contenu dans la liste