E2I- 4 (option HF)
CONTRÔLE JAVA 1

Corrigé

1. Les données de type **char** sont codées en java comme :
   - ☐ un octet (code ASCII)
   - ☐ deux octets (code ASCII)
   - ☐ un octet (code unicode)
   - [X] deux octets (code unicode)
   - ☐ un ou deux octets selon l'application
   - ☐ Aucune des réponses ci-dessus n'est valable

2. Pour exécuter une application écrite en java et compilée sous Linux dans un ordinateur fonctionnant sous Windows on doit :
   - ☐ Recompiler les sources originales
   - ☐ Changer légèrement les sources et les recompiler
   - ☐ Réécrire le programme pour l'adapter au nouveau système
   - [X] Copier les binaires qui peuvent être exécutés sans changement
   - ☐ Installer un "plugin" de conversion
   - ☐ Aucune des réponses ci-dessus n'est valable

3. Pour toute classe, un constructeur par défaut (créé par le compilateur), existe dans les cas suivants :
   - ☐ Dans tous les cas
   - [X] Si il n'y a pas d'autre constructeur de défini
   - ☐ Si il y a au moins un autre constructeur de défini
   - ☐ Jamais, il faut le définir
   - ☐ Si la classe comporte une méthode main
   - ☐ Quand la classe est publique

4. On a la variable

```
String sb=new String("Bonjour");
```

   Lesquelles de ces expressions renvoient la valeur **true** ?
   - ☐ `sb="Bonjour"`
   - ☐ `sb.equals("bonjour")`
   - ☐ `sb=="Bonjour"`
   - ☐ `sb=="bonjour"`
   - [X] `sb.equalsIgnoreCase("Bonjour")`

5. Considérer la définition de la classe

```
public class Q{
  int [] tab= new int[10];

   Q(){
   for(int i=0; i<10; i++)
       tab[i]=0;
     assigneDeux(tab,3);
     System.out.println( "tab[3]="+tab[3]);
  }

   public void assigneDeux(int tab[], int i){
     tab[i]=2;
   }

   public static void main(String[] args){
     Q monApp= new Q();
   }
}
```

☐ Imprime : tab[3]=0
☒ Imprime : tab[3]=2
☐ Imprime : tab[3]=3
☐ Imprime : tab[3]=null
☐ Aucune des réponses ci-dessus n'est valable

6. On a la classe suivante :

```
  public class A extends B{
  ....
}
```

Cela signifie que (une seule réponse) :
☐ A *a un* B
☐ A *est relié à un* B
☐ A *dépends de* B
☐ A *exclue un* B
☐ A *précède un* B
☒ A *est un* B

7. Pour appeler un constructeur de la classe de base à partir de la classe dérivée on utilise le mot clé
☐ **static**
☐ **this**
☐ **extends**
☒ **super**
☐ **implements**
☐ Aucune des réponses ci-dessus n'est valable

8. La classe TroupeauDeVaches est une classe dérivée de la classe
☐ Mammifère
☐ Vache
☐ Quadrupède
☐ Mouton
☐ FermeAgricole
☒ Aucune des classes énoncées ci-dessus

9. La classe B est une classe dérivée de A et la classe C est dérivée de B. Nous avons :

```
A a= new A();
B b = new B();
C c = new C();

if(c instanceof A)
    System.out.println("Le test est vrai");
else
    System.out.println("Le test est faux");
```

imprime :
[X] Le test est vrai
[ ] Le test est faux
[ ] Donne une exception

10. Ecrire un programme qu'initialise un tableau de 100 entiers multiples de 4 et l'affiche à l'écran 10 nombres par ligne

```
public class Tab4 {

    public static void main(String[] args) {
        int tab4[] = new int[100];
        int j = 0;
        for (int i = 0; i < 100; i++) {
            tab4[i] = j;
            j += 4;
        }
        for (int i = 0; i < 100; i++) {
            System.out.print(tab4[i] + " ");
            if((i+1)%10==0)System.out.println();
        }

    }
}
```

11. Ecrire la classe Point qui décrit un point géométrique dans le cadre d'un programme de dessin. On se limitera aux dessins sur un plan. Écrivez un constructeur, les méthodes d'accès et au moins deux autres méthodes dont la méthode **toString()** ;

```
public class Point {
    private int x, y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }
```

```
    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public void decale(int dx, int dy){
        x+=dx;
        y+=dy;
    }

    public boolean estEgaleA(Point p) {
        return x == p.x && y == p.y;
    }

    public boolean estAGaucheDe(Point p){
        return x<p.x;
    }

    public String toString() {
        return "Point{" + "x=" + x + "y=" + y + '}';
    }
}
```

12. Ecrire la classe Polygone utilisant la classe Point définie ci-dessus. Écrivez, ici aussi, un constructeur, les méthodes d'accès et au moins deux autres méthodes dont la méthode **toString()** ;

```
public class Polygone {
    private Point mesPoints[];
    private int nbPoints=0;

    public Polygone() {
        mesPoints= new Point[100];
    }

    public Point[] getMesPoints() {
        return mesPoints;
    }

    public void setMesPoints(Point[] mesPoints) {
        this.mesPoints = mesPoints;
    }

    public int getNbPoints() {
        return nbPoints;
    }

    public void setNbPoints(int nbPoints) {
        this.nbPoints = nbPoints;
    }

    public boolean addPoint(Point p){
        if(nbPoints==100) return false;
        mesPoints[nbPoints]=p;
```

```
        return true;
    }
    public String toString() {
        String s="Polygone{\n";
        for(int i=0; i<nbPoints; i++){
            s+=mesPoints[i]+"\n";
        }
        s+="}\n";
        return s;
    }
}
```

E2I-3 (option M.O. et F.P)
TP Java n^{o}4 (Contrôle)
Seuls documents admis : des notes manuscrites

Attention : Créez un sous-répertoire avec votre **NOM** dans un répertoire nommé **reponse** dans le Bureau. Déposez dans ce répertoire *uniquement* les sources (fichiers .java ou xml) qui constituent vos réponses. Seulement les sources présentes dans ce repertoire seront corrigés.

Le Sac Postal

Un sac postal contient un ensemble de lettres et de colis simples. Le poids totale du sac ne peux exceder 30 kilogrammes. On veut savoir le poids total du sac, le montant total des affranchissements et on veut également imprimer les caractéristiques des lettres et colis contenues dans le sac postal.

- Une lettre est un envoi qui se caractérise par son poids en grammes ($\leq$200g) et par son adresse de destination. L'affranchissement d'une lettre se calcule grâce au tableau suivant :

Poids (grammes)	0-19	20-49	50-99	100-149	150-200
tarif (euros)	0,5	0,75	1	2	2,5

- Une lettre recommandée est une lettre dotée d'un taux de recommandation parmi trois possibles :

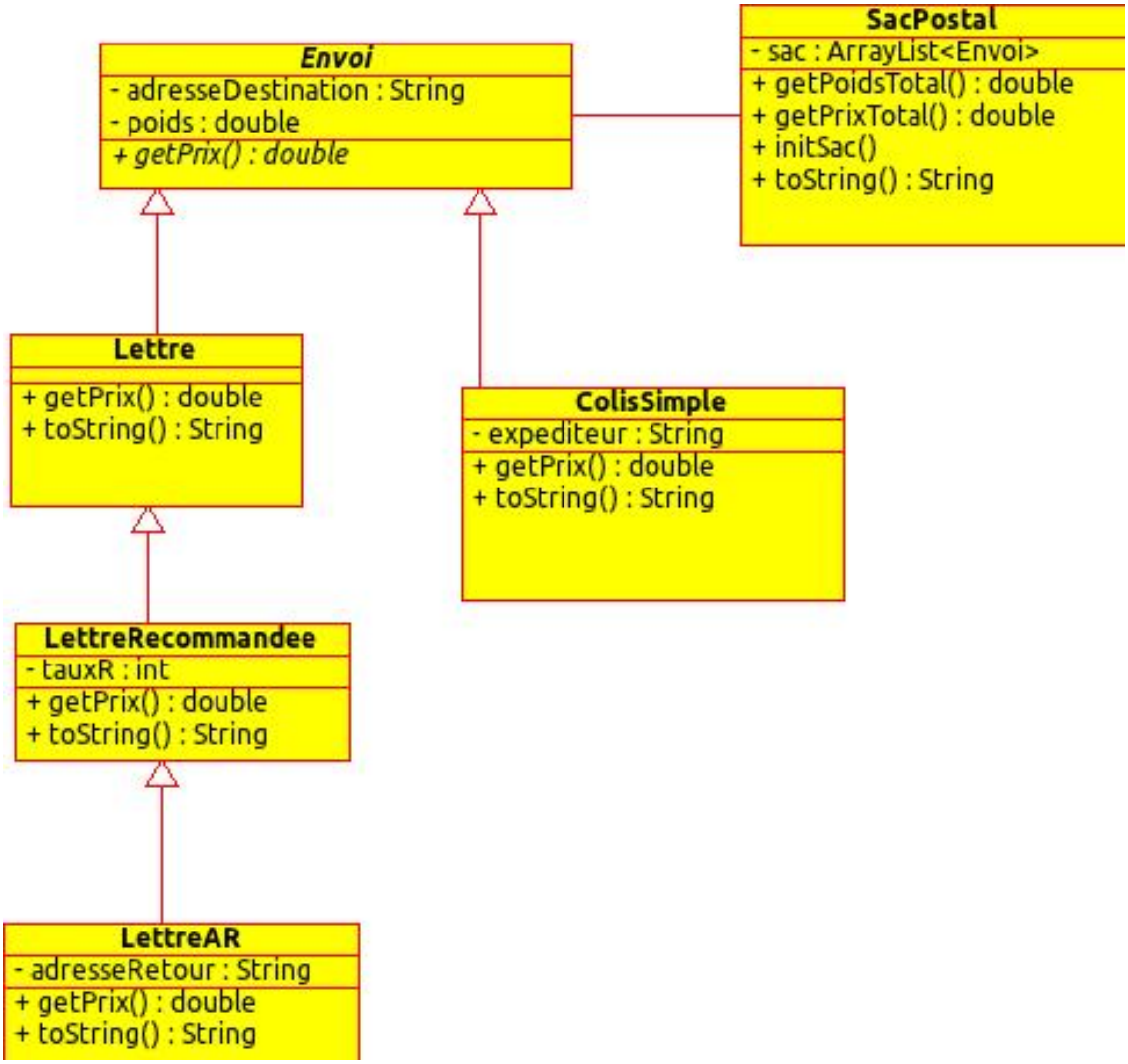
taux	R1	R2	R3
prix supplémentaire	2,7	3,2	4

- Une lettre recommandée avec accusé de réception est une lettre recommandée dotée d'une adresse de retour. L'affranchissement est celui d'une lettre recommandée majoré d'une surtaxe uniforme de 1 euro.
- Un colis simple est un envoi qui a une adresse de destination et un expediteur. Il pesse plus de 200 g et moins de 2000 g. Il ne peut pas être recommandé ni avoir d'accusé de réception (il existent d'autres types d'envoi qui ne vont pas dans notre sac postal). L'affranchissement est proportionel au poids et fixé à 10 centimes le 100 grammes avec un prix de base de 2 euros 50 pour les premiers 200g.

1. **(barème indicatif : 5 points)** Faire le diagramme UML des classes correspondantes à envoi, lettre, lettre recommandée, lettre recommandée avec accusé de réception, colis simple et sac postal. Vous pouvez aussi ajouter d'autres classes auxiliaires si vous le souhaitez. Vous pouvez le faire soit à l'aide du logiciel umbrello ou simplement sur papier libre. N'oubliez pas de le rendre à la fin ou de sauver le diagramme umbrello dans le répertoire de réponses.
2. **(barème indicatif : 3 points)** Ecrire la ou les classes pour décrire une lettre et faire un programme de test qui :
   (a) Crée une lettre
   (b) L'affiche à l'écran
   (c) Calcule le prix de son affranchissement el l'affiche à l'écran
3. **(barème indicatif : 3 points)** Ecrire les classes pour les lettres recommandées sans et avec accusé de réception.
   Completez le programme de test pour :
   (a) Créer une lettre recommandée et une lettre recommandé avec accusé de réception
   (b) Les afficher à l'écran, avec leur affranchissement.
4. **(barème indicatif : 2 points)** Ecrire la classe pour le colis simple.
   Completez le programme de test pour :
   (a) Créer un colis simple
   (b) Les afficher à l'écran, avec son affranchissement.
5. **(barème indicatif : 5 points)** Ecrire la classe SacPostal en prenant en compte qu'un sac ne peut contenir plus de 30 Kg. Écrire un programme d'application qui crée un SacPostal contenant 5 lettres de tous types, et deux colis simples, qui calcule et affiche son poids, le prix Total des affranchissements et un liste de toutes les envois que le sac contient.

6. **(barème indicatif : 2 points)** Ajouter à la classe SacPostal une méthode qui permette de trouver une lettre d'après son adresse et indiquer si elle est recommandée ou non et si elle comporte ou non un accusé de réception.
7. (Bonus) Finalement vous devez écrire une application pour gérer un sac postal. Cette application permettra (au minimum) de :
   - Ajouter une lettre au sac
   - Afficher le contenu du sac
   - Calculer le poids total du sac
   - Calculer le prix total des affranchissements des lettres du sac

Corrigé



Envoi.java

```
 1
 2 package laposte;
 3
 4 /**
 5  * Classe modélisant un envoi postal
 6  */
 7 public abstract class Envoi {
 8
 9     private String adresse;
10     private double poids;
11
12     public Envoi(String adresse, double poids) {
13         this.adresse = adresse;
14         this.poids = poids;
15     }
16
17     /** Trouve l'adresse de la lettre
18      */
19     public String getAdresse() {
20         return adresse;
21     }
22
23     /** Modifie l'adresse de la lettre
24      */
25     public void setAdresse(String adresse) {
26         this.adresse = adresse;
```

```
27      }
28
29      /** Trouve le poids de la Lettre
30       */
31      public double getPoids() {
32          return poids;
33      }
34
35      /** Donne un nouveau poids a la lettre
36       */
37      public void setPoids(double poids) {
38          this.poids = poids;
39      }
40
41      abstract double getPrix();
42
43      @Override
44      public String toString() {
45          return "adresse=" + adresse + "\n poids=" + poids + '}';
46      }
47  }
```

Lettre.java

```
 1    package laposte;
 2
 3
 4    /** class modelisant une lettre postale normale
 5     */
 6    public  class Lettre extends Envoi{
 7
 8       public Lettre( String a, double p){
 9           super(a,p);
10
11      }
12
13       /** Calcule le prix de la lettre
14       */
15       public  double getPrix(){
16         double tabPoids[]={20.,50.,100.,250.,500.,1000., 2000.};
17         double tabPrix[]={0.5,0.75,1.,2.,2.5,3.5,4.5};
18         for (int i=0; i<tabPrix.length;i++){
19            if (getPoids() < tabPoids[i])
20               return tabPrix[i];
21         }
22         return 0.;
23      };
24
25      @Override
26         public String toString(){
27           return
28              "Adresse destinataire: \n"+ getAdresse() +"\n"+"Poids: "+getPoids()+"\n";
29      }
30    }
```

LettreRecommandee.java

```
 1
2 package laposte;
3
4 public class LettreRecommandee extends Lettre {
5
6     int tauxR;
7
8     public LettreRecommandee(String a, double p, int t) {
9         super(a, p);
10        tauxR = t - 1;
11    }
12
13    public double getPrix() {
14        double tabPrixSup[] = {2.7, 3.2, 4.0};
15        return super.getPrix() + tabPrixSup[tauxR];
16    }
```

```
17
18      public String toString() {
19          return super.toString() + "Taux recommande: R"
20           + (tauxR + 1) + "\n";
21      }
22  }
```

LettreAR.java

```
 1  package laposte;
 2
 3  public class LettreAR extends LettreRecommandee {
 4
 5      private String adresseRetour;
 6
 7      public LettreAR(String a, double p, int tr, String ar) {
 8          super(a, p, tr);
 9          adresseRetour = ar;
10      }
11
12      public void setAdresseRetour(String ar) {
13          adresseRetour = ar;
14
15      }
16
17      public String getAdresseRetour() {
18          return adresseRetour;
19      }
20
21      @Override
22      public double getPrix() {
23          return super.getPrix() + 1.;
24      }
25
26      @Override
27      public String toString() {
28          return super.toString() + "Adresse retour:\n" + adresseRetour + "\n";
29      }
30  }
```

ColisSimple.java

```
 1
 2  package laposte;
 3
 4  /**
 5   *
 7   */
 8  public class ColisSimple extends Envoi{
 9       String expediteur;
10      public ColisSimple(String adresse, double poids, String ar) {
11          super(adresse, poids);
12          expediteur=ar;
13      }
14
15      @Override
16      double getPrix() {
17         return (int)((getPoids()-200.)/10)*.1+2.5;
18      }
19
20      @Override
21      public String toString() {
22          return "Colis Simple:\n" +  "Adresse destinataire: \n"+
23            getAdresse() +"\n"+"Poids: "+getPoids()+"\nexpediteur=" + expediteur+"\n" ;
24      }
25  }
```

SacPostal.java

```
 1  package laposte;
 2
 3  import java.util.ArrayList;
 4
 5  /** Class modelisant un sac postal composé
```

```java
des Lettres et de Colis Simples
 */
public class SacPostal {

    ArrayList<Envoi> sac;
    double poidsTotale;

    public SacPostal() {
        sac = new ArrayList();
    }

    /** Calcule le poids total des lettres presentes dans le sac
     * @return Le poids total des lettres du sac
     */
    public double getPoidsTotal() {
        return poidsTotale;

    }

    /** Calcule le prix total des lettres presentes dans le sac
     * @return Le prix total des lettres du sac
     */
    public double getPrixTotal() {
        double somme = 0;
        for (int i = 0; i < sac.size(); i++) {
            somme += sac.get(i).getPrix();
        }
        return somme;

    }

    /** Ajouter une Lettre au sac
     * @param l la lettre a ajouter
     * @return vrai si l'ajout a ete fait faux dans le cas contraire
     */
    public boolean addEnvoi(Envoi l) {
        if (poidsTotale+l.getPoids() <30000) {
            sac.add(l);
            return true;
        } else {
            return false;
        }
    }

    /** Initialise le sac avec 6 lettre de tous types
    cette méthode est appellée par le constructeur
     */
    public void initSac() {

        Envoi l;

        l = new Lettre("Jojo la Menace \n"
                + "1,rue Mozart\n 91678 Fresnes", 20.);
        addEnvoi(l);
        l = new LettreRecommandee("Pierre Dupont\n"
                + "5, rue du Bach\n 75031 Paris", 10., 1);
        addEnvoi(l);
        l = new Lettre("Miguel Gomez\n15, rue d'Espagne\n"
                + " 63219 Nantes", 30.);

        addEnvoi(l);
        l= new ColisSimple("Patrice Lelay 15 rue de la Convention,"+""
                + " 93300 Aubervillliers", 500, "Anonyme Boulevard Inconnu 1000 Nullpart");
        addEnvoi(l);
        l = new LettreRecommandee("Arthur Rambaud\n"
                + "21,place Jussieu\n 75231 Paris", 200., 2);
        addEnvoi(l);
        l = new LettreAR("Albert Camus\n14, rue de la Poste \n"
                + " 75005 Paris ", 1400, 2,
                "Marcel Lejuge\n11 rue des escargots\n"
                + "45098\nLa Follie sur Seine");
```

```
77          addEnvoi(l);
78          l = new Lettre("Edith Piaf\n71,rue Bethoven\n 91000 Melun", 5.);
79          addEnvoi(l);
80
81
82      }
83
84      /** Creation d'une chaine contenant les informations de
85      toutes les lettres presentes dans le sac
86       */
87      public String toString() {
88          String s = "";
89          for (int i = 0; i < sac.size(); i++) {
90              s += sac.get(i).toString() + "\n";
91          }
92
93          return s;
94      }
95
96     public  Envoi trouveAdresse(String ad){
97         for(int i=0; i< sac.size();i++)
98         {
99             Envoi e=sac.get(i);
100            if( e.getAdresse().equals(ad)){
101                return e;
102            }
103
104        }
105        return null;
106    }
107
108 }
```

LaPoste.java

```
1  package laposte;
2
3  /**
4   *
5   * @author jorge
6   */
7  public class LaPoste {
8
9      public static void main(String[] args) {
10
11         /* Deuxième point  */
12         Lettre l1 = new Lettre("Vincent Monami\n13 rue Detente, Monaco", 20.);
13         System.out.print("Lettre :" + l1);
14         System.out.println("prix : " + l1.getPrix() + "euros\n");
15
16         /* Troisième point */
17         LettreRecommandee lr = new LettreRecommandee("Administration impots\n"
18                 + "32 rue de la Dette, 75101 Paris", 120., 2);
19         System.out.print("Lettre R :" + lr);
20         System.out.println("prix : " + lr.getPrix() + "euros\n");
21
22         LettreAR lrar = new LettreAR("Syndic Copropriété\n 32 rue de la Justice,"
23                 + "21380 NUllepart", 200., 3,
24                 "21 rue des  Violettes, Fleury Merogis");
25         System.out.print("Lettre AR :" + lrar);
26         System.out.println("prix : " + lrar.getPrix() + "euros\n");
27
28         /*Quatrième point */
29
30         SacPostal sp = new SacPostal();
31         sp.initSac();
32         sp.addEnvoi(l1);
33         sp.addEnvoi(lr);
34         sp.addEnvoi(lrar);
35         sp.addEnvoi(new LettreRecommandee("Mr X, 123 rue des Radio,"
36                 + "France Inter", 300., 1));
37         sp.addEnvoi(new LettreAR("Pere Noel, rue des Plages, Laponie", 150., 3,
```

```
38                 "Kevin Nivek, 12 rue des Reves, Saint Malo"));
39
40
41         System.out.println("Poids total : " + sp.getPoidsTotal() + "g");
42         System.out.println("Prix total : " + sp.getPrixTotal() + "euros");
43         System.out.println(sp);
44         String ad = "Pere Noel";
45         Envoi l = sp.trouveAdresse(ad);
46         if (l != null) {
47             System.out.println("Trouvée :" + l);
48             if (l instanceof Lettre) {
49                 System.out.print("C'est une Lettre");
50             }
51             if (l instanceof LettreRecommandee) {
52                 System.out.print(" recommandée");
53             }
54             if (l instanceof LettreAR) {
55                 System.out.print(" avec AR");
56             }
57             if (l instanceof ColisSimple){
58                 System.out.print (" Colis Simple");
59             }
60             System.out.println();
61
62         } else {
63             System.out.println("Lettre pour :" + ad + " non trouvée!");
64         }
65
66
67
68     }
69 }
```