Eercice1

```
class A{
private int n ;
private int p=10 ;
public A (int n)
{System.out.println ("Entree Constructeur A - valeur de l'attribut n=" + n) ;
System.out.println ("Entree Constructeur A - valeur de l'attribut p=" + p) ;
this.n = n ;
System.out.println ("Sortie Constructeur A - valeur de l'attribut n=" + n) ;
System.out.println ("Sortie Constructeur A - valeur de l'attribut p=" + p) ;
}
}
class B extends A {
private int q=25 ;
public B (int n, int pp)
{ super (n) ;
System.out.println ("Entree Constructeur A - valeur de l'attribut n=" + n) ;
System.out.println ("Entree Constructeur A - valeur de l'attribut p=" + p) ;
System.out.println ("Entree Constructeur A - valeur de l'attribut q=" + q) ;
p = pp ;
q = 2*n ;
System.out.println ("Sortie Constructeur A - valeur de l'attribut n=" + n) ;
System.out.println ("Sortie Constructeur A - valeur de l'attribut p=" + p) ;
System.out.println ("Sortie Constructeur A - valeur de l'attribut q=" + q) ;
}
}
public class Ex1{ public static void main (String args[]){ A a = new A(5) ; B b = new B(7, 3) ;}
}
```

- Erreur à la compilation: P est privé il faut changé private dans la classe A par protected. Afin que les instances de la classe B puissent acceder à p
- Erreur sémantique : n est l'argument du constructeur et non pas l'attribut n de A. Pour désigner l'attribut n il faut utiliser le mot clé this.

class B extends A { ..... }
La création d'un objet de type B se déroule en 6 étapes.

1. Allocation mémoire pour un objet de type B ; il s'agit bien de l'intégralité de la mémoire nécessaire pour un tel objet, et pas seulement pour les champs propres à B (c'est-à-dire non hérités de A).
2. Initialisation par défaut de tous les champs de B (aussi bien ceux hérités de A, que ceux propres à B) aux valeurs "nulles" habituelles.
3. Initialisation explicite, s'il y a lieu, des champs hérités de A ; éventuellement, exécution des blocs d'initialisation de A.
4. Exécution du corps du constructeur de A.
5. Initialisation explicite, s'il y a lieu, des champs propres à B ; éventuellement, exécution des blocs d'initialisation de B.
6. Exécution du corps du constructeur de B.

```
Entree Constructeur A - valeur de l'attribut n=0
Entree Constructeur A - valeur de l'attribut p=10
Sortie Constructeur A - valeur de l'attribut n=5
Sortie Constructeur A - valeur de l'attribut p=10
----------------------Fin--A---------------------
Entree Constructeur A - valeur de l'attribut n=0
Entree Constructeur A - valeur de l'attribut p=10
Sortie Constructeur A - valeur de l'attribut n=7
Sortie Constructeur A - valeur de l'attribut p=10
----------------------Fin--A---------------------
Entree Constructeur B - valeur de l'attribut n=7
Entree Constructeur B - valeur de l'attribut p=10
Entree Constructeur B - valeur de l'attribut q=25
Sortie Constructeur B - valeur de l'attribut n=7
Sortie Constructeur B - valeur de l'attribut p=3
Sortie Constructeur B - valeur de l'attribut q=14
----------------------Fin--B---------------------
```

Exercice2

```
class A {
protected int a = 5;
public A(int a) {
this.a = a;
}
public void afficherClasse() {
System.out.println("Classe A");}
public void afficherVariables() {
System.out.println("a = " + a);
} }
class B extends A {
protected int b = 6;
public B(int b) {
super(2 * b);
a = b;}
public void afficherClasse() {
super.afficherClasse();
System.out.println("Classe B");}
public void afficherVariables() {
super.afficherVariables();
System.out.println("b = " + b);}
}
class C extends B {
protected int b = 7;
protected int c = 8;
public C(int c) {
super(3 * c);
b = c;
}
public void afficherClasse() {
super.afficherClasse();
System.out.println("Classe C");
}
public void afficherVariables() {
super.afficherVariables();
System.out.println("c = " + c);
} }
class Alphabet {
public static void main(String
args[]) {
A[] as = new A[3];
as[0] = new A(1);
as[1] = new B(2);
as[2] = new C(3); for (int i = 0; i <
as.length; i++) {
as[i].afficherClasse();
System.out.println("-----");}
for (int i = 0; i < as.length; i++) {
as[i].afficherVariables();
System.out.println("-----");}
}
}
```

```
Solution
Classe A
-----
Classe A
Classe B
-----
Classe A
Classe B
Classe C
-----
a = 1
```

```
-----
a = 2
b = 6
-----
a = 9
b = 6
c = 8
----
```

Exercice 3

```
class A{
public void f(double x) { System.out.print ("A.f(double=" + x +") ") ; }
}
class B extends A {}
class C extends A{
public void f(long q) { System.out.print ("C.f(long=" + q + ") ") ; }
}
class D extends C{
public void f(int n) { System.out.print ("D.f(int=" + n + ") ") ; }
}
class F extends C{
public void f(float x) { System.out.print ("F.f(float=" + x + ") ") ; }
public void f(int n) { System.out.print ("F.f(int=" + n + ") ") ; }
}
public class Ex4{
public static void main (String arg[]){
byte bb=1 ; short p=2 ; int n=3 ; long q=4 ;
float x=5.f ; double y=6. ;
System.out.println ("** A ** ") ;
A a = new A() ; a.f(bb) ; a.f(x) ; System.out.println() ;
//A.f(double=1.0) A.f(double=5.0)

System.out.println ("** B ** ") ;
B b = new B() ; b.f(bb) ; b.f(x) ; System.out.println() ;
//A.f(double=1.0) A.f(double=5.0)

a = b ; a.f(bb) ; a.f(x) ; System.out.println() ;
//A.f(double=1.0) A.f(double=5.0)

System.out.println ("** C ** ") ;
C c = new C() ; c.f(bb) ; c.f(q) ; c.f(x) ; System.out.println() ;
//C.f(long=1) C.f(long=4) A.f(double=5.0)

a = c ; a.f(bb) ; a.f(q) ; a.f(x) ; System.out.println() ;
//A.f(double=1.0) A.f(double=4.0) A.f(double=5.0)

System.out.println ("** D ** ") ;
D d = new D() ; d.f(bb) ; d.f(q) ; d.f(y) ; System.out.println() ;
//D.f(int=1) C.f(long=4) A.f(double=6.0)

a = d ; a.f(bb) ; a.f(q) ; a.f(y) ; System.out.println() ;
//A.f(double=1.0) A.f(double=4.0) A.f(double=6.0)

System.out.println ("** F ** ") ;
F f = new F() ; f.f(bb) ; f.f(n) ; f.f(x) ; f.f(y) ; System.out.println() ;
//F.f(int=1) F.f(int=3) F.f(float=5.0) A.f(double=6.0)

a = f ; a.f(bb) ; a.f(n) ; a.f(x) ; a.f(y) ; System.out.println() ;
//A.f(double=1.0) A.f(double=3.0) A.f(double=5.0) A.f(double=6.0)

c = f ; c.f(bb) ; c.f(n) ; c.f(x) ; c.f(y) ;
//C.f(long=1) C.f(long=3) A.f(double=5.0) A.f(double=6.0)
}
}
```

L'algorithme de sélection des methodes doit trouver la methode à exécuter en fonction de l'appel. Cet algorithme procède en deux étapes :
- Etape 1 : Pendant la compilation, l'algorithme de résolution de la surcharge est appelé. pour choisir la methode avec la signature la plus adequate
- Etape 2 : Pendant l'exécution, à partir de la signature choisie, on cherche une methode qui a exactement la même signature que celle sélectionnée à l'étape 1. On exécute le premier methode rencontré, si il n'existe pas de tel methode on remonte dans heirarchie, si il en existe pas…. On remonte encore… dans la sur sur classe. Il existe obligatoirement un methode qui correspond à cette signature car sinon, il y aurait eu une erreur de compilation à étape 1.

Exercice4

```
package exercice4;

class Personne {
public void parler() {System.out.println("hum");}
}
class Adulte extends Personne{
public void parler() {System.out.println("Bonjour tout le monde!");}
}
class AdulteDistingue extends Adulte {
public void parler() {System.out.println("Mes chers amis, bonjour!");}
}
class Jeune extends Personne {
}
class Ado extends Jeune {
public void parler() {System.out.println("salut !");}
}
class Enfant extends Jeune {
}
class Bebe extends Enfant {
public void parler() {System.out.println("mamma ") ;}
}
class Nourrisson extends Bebe {
public void parler() {System.out.println("Agheu, agheu!");}
}



public class Ex4{
public static void main(String args[]) {
Personne P1 = new Personne();
Personne P2 = new AdulteDistingue();
Adulte P3 = new AdulteDistingue();
Personne P4 = new Bebe();
Jeune P5 = new Ado();
Enfant P6 = new Nourrisson();
Nourrisson P7 = new Nourrisson();
Enfant P8 = new Bebe();
P1.parler();  //hum
P2.parler();  //Mes chers amis, bonjour!
P3.parler();  //Mes chers amis, bonjour!
P4.parler();  //mamma
P5.parler();  //salut !
P6.parler();  //Agheu, agheu!
P7.parler();  //Agheu, agheu!
P8.parler();  //mamma
}
}
```

- P1 = P2; //ok
- P4 = P1; //ok
- P3 = P4; //Erreur de compilation: une Personne n'est pas un Adulte
- P3 = P1; //Erreur de compilation: une Personne n'est pas un Adulte
- P4 = P5; //ok
- P7 = P6; //Erreur de compilation: un Enfant n'est pas un nourrissant
- P7 = (Nourrisson) P4; //erreur à l'exécution : P4 référence un objet de type Bébé pas Nourrisson
- P6 = (Bebe) P4; //OK
- P3 = (AdulteDistingue) P2; //OK
- P8 = (Bebe) P5; //Erreur d'execution: Ado n'est pas un Bebe

```
Jeune[] e = new Jeune[4];
e[0] = (Jeune)P4;
e[1] = P5;
e[2] = P6;
e[3] = P7;
e[4] = P8;
```

Boucle resultats

```
e[0].parler(); //mamma
e[1].parler(); //salut !
e[2].parler(); //Agheu, agheu!
e[3].parler(); //Agheu, agheu!
e[4].parler(); //mamma
```

```
P5 = e[0];
P4 = e[1];
P7 = (Nourrisson)e[2];
P8 = (Enfant)e[3];  ou Bebe
P6 = (Enfant)e[4];  ou Bebe
```

Exercice5

```
public class Ex5 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	if(args.length==3)
	{
	 try{  Personne personne=new Personne(args[0],args[1],Integer.parseInt(args[2]));
		personne.afficher();
	    }
	    catch(Exception e){ System.out.println("le 3eme arguments n'est pas un numero"); }
	}
	else System.out.println("le nombre de valeurs entrees est different de 3");
 }
}
class Personne{
private String nom,prenom;
private int age;

 public Personne(String nom,String prenom,int age){
	this.nom=nom;
	this.prenom=prenom;
	this.age=age;
 }

 public void afficher()
  { System.out.print("je suis "+nom+" "+prenom+" est j'ai "+age+" ans");}
}
}
```

```
public class Ex5 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	  Personne[] personnes=new Personne[5];
	  personnes[0]=new Enseignant("n1","p1",40,15,"m1");
	  personnes[1]=new Enseignant("n2","p2",42,12,"m2");
	  personnes[2]=new Enseignant("n3","p3",35,18,"m3");
	  personnes[3]=new Etudiant("n4","p4",22,15,new float[]{15,12,8});
	  ((Etudiant)personnes[3]).claculMoyenne(new float[]{15,12,8});
	  personnes[4]=new Etudiant("n5","p5",23,16,new float[]{12,12,11});
	  ((Etudiant)personnes[4]).claculMoyenne(new float[]{12,12,11});


	  for(int i=0;i<personnes.length;i++)
	     {
	       if(personnes[i] instanceof Enseignant) System.out.println("Enseigant");
	       else System.out.println("Etudiant");
	       personnes[i].afficher();
	       System.out.println("-------");
	     }
	}
}
class Personne{
private String nom,prenom;
private int age;

 public Personne(String nom,String prenom,int age){
	this.nom=nom;
	this.prenom=prenom;
	this.age=age;
 }

 public void afficher()
  { System.out.println("je suis "+nom+" "+prenom+" est j'ai "+age+" ans");}
```

```
 public void afficher(boolean reduit)
  {
        if(reduit){System.out.print("je suis "+nom+" "+prenom );}
        else afficher();
  }
}

class Enseignant extends Personne{
       private int nbHeures;
       private String module;


 public Enseignant(String nom,String prenom,int age,int nbHeures, String module){
       super(nom,prenom,age);
       this.nbHeures=nbHeures;
       this.module=module;
 }

 public void afficher(){ super.afficher();System.out.println("j'ai "+nbHeures+" heures de
travaille et j'enseigne le module "+module);}

}

class Etudiant extends Personne{
       private int matricule;
       private float[] notes=new float[3];
       private float moyenne;

 public Etudiant(String nom,String prenom,int age,int matricule, float[] notes){
       super(nom,prenom,age);
       this.matricule=matricule;
       this.notes=notes;
 }

 public void afficher(){ super.afficher();System.out.println("j'ai la moyenne "+this.moyenne);}
 public float claculMoyenne(float[]notes){
this.moyenne=(notes[0]+notes[1]+notes[2])/3.0f;return(this.moyenne);}
}
```

```
import java.util.Arrays;
import java.util.Arrays;

public class Ex6 {

       public static void main(String[] args) {
              // TODO Auto-generated method stub
         Personne[] personnes=new Personne[5];
         Enseignant[] enseignants=new Enseignant[3]; Etudiant[] etudiants=new Etudiant[2];
         enseignants[0]=new Enseignant("n1","p3",40,15,new String[]{"m1","m2","m3"});
         enseignants[1]=new Enseignant("n1","p2",42,12,new String[]{"m2"});
         enseignants[2]=new Enseignant("n3","p3",35,18,new String[]{"m3"});
         etudiants[0]=new Etudiant("n4","p4",22,15,new float[]{15,12,8});
         etudiants[0].claculMoyenne(new float[]{15,12,8});
         etudiants[1]=new Etudiant("n5","p5",23,16,new float[]{12,12,15});
         etudiants[1].claculMoyenne(new float[]{12,12,15});

         personnes[0]=enseignants[0];          personnes[1]=enseignants[1];
         personnes[2]=enseignants[2];          personnes[3]=etudiants[0];
         personnes[4]= etudiants[1];

         System.out.println("enseignants");  Arrays.sort(enseignants);
         for(Enseignant ens:enseignants){ens.afficher();}
         System.out.println("etudiants");    Arrays.sort(etudiants);
         for(Etudiant etu:etudiants){etu.afficher();}
       }
}
```

```java
abstract class Personne{
private String nom,prenom;
private int age;
private static int nombrePersonne=0;

 public Personne(String nom,String prenom,int age){
       this.nom=nom;
       this.prenom=prenom;
       this.age=age;
       nombrePersonne++;
 }

 public int nombrePersonne(){
        return nombrePersonne;
 }
 abstract void afficherType();

 public void afficher()
  { System.out.println("je suis "+nom+" "+prenom+" est j'ai "+age+" ans");}

 public void afficher(boolean reduit)
  {
        if(reduit){System.out.print("je suis "+nom+" "+prenom );}
        else afficher();
  }

public String getNom() {  return nom;}

public String getPrenom() {      return prenom;}
}

class Enseignant extends Personne  implements Comparable<Enseignant>{
       private int nbHeures;
       private String[] module;


 public Enseignant(String nom,String prenom,int age,int nbHeures, String[] module){
       super(nom,prenom,age);
       this.nbHeures=nbHeures;
       this.module=module;
 }

 public void afficher(){
        String mda = "";
        for(String md:module){mda=mda+" "+md;}
        super.afficher();System.out.println("j'ai "+nbHeures+" heures de travaille et j'enseigne
le module "+mda);}

 void afficherType(){ System.out.println("Enseigant");}

@Override
public int compareTo(Enseignant o) {
       // TODO Auto-generated method stub
       int i=0;
       i=this.getNom().compareTo( o.getNom());
       if(i==0) return (this.getPrenom().compareTo(o.getPrenom()));
       return i;}
}
```

```java
class Etudiant extends Personne  implements Comparable<Etudiant>{
	private int matricule;
	private float[] notes=new float[3];
	private float moyenne;

 public Etudiant(String nom,String prenom,int age,int matricule, float[] notes){
	super(nom,prenom,age);
	this.matricule=matricule;
	this.notes=notes;
 }

 public void afficher(){ super.afficher();System.out.println("j'ai la moyenne "+this.moyenne);}

 public float claculMoyenne(float[]notes){
this.moyenne=(notes[0]+notes[1]+notes[2])/3.0f;return(this.moyenne);}

 void afficherType(){ System.out.println("Etudiant");}

@Override
public int compareTo(Etudiant o) {
	// TODO Auto-generated method stub

	float r=((moyenne)-((o.moyenne)));

	if(r==0)return 0;
	else if (r>0) return 1;
	return -1;

}
}
```

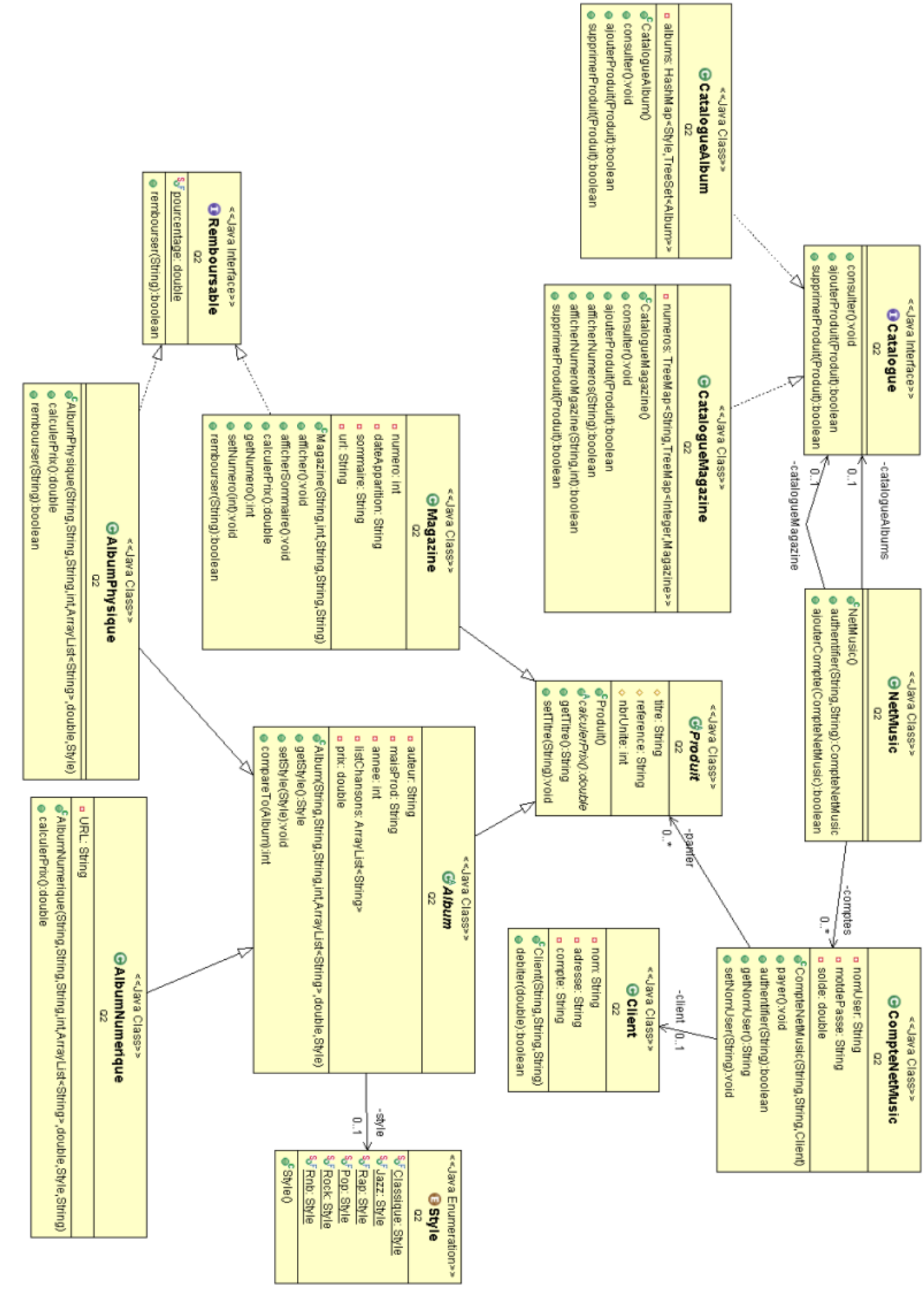
bonus

```java
package bonus;

 enum Civilite {  // dans le fichier Civilite.java

    MADAME("MME",1), MADEMOISELLE("MLLE",2), MONSIEUR("MR",3) ;

     private String abreviation ;
     private int valeur ;

     private Civilite(String abreviation,int valeur) {
         this.abreviation = abreviation ;
         this.valeur=valeur;
    }

     public String getAbreviation() {
         return  this.abreviation ;
    }
     public int getvaleur() {
         return  this.valeur ;
    }
}
public class enumeration {

        public static void main (String args[]){
               Civilite a=Civilite.MADEMOISELLE;
               System.out.println("MADEMOISELLE "+a.getvaleur());
               System.out.println("MADEMOISELLE "+a.getAbreviation());

        }
}
```

Q1


<<Java Class>>
NetMusic
Q1
nbrComptes: int
NetMusic(int,int)
authentifier(String,String):CompteNetMusic
rechecher(CompteNetMusic):CompteNetMusic
ajouterCompte(CompteNetMusic):boolean
ajouterAlbum(Album):void
consulterAlbums():void
-catalogueAlbums
0..1
-comptes
0..*
<<Java Class>>
CatalogueAlbum
Q1
nbrAlbums: int
CatalogueAlbum(int)
consulter():void
ajouter(Album):boolean
-albums
0..*
<<Java Class>>
CompteNetMusic
Q1
nomUser: String
motdePasse: String
nbr: int
solde: double
CompteNetMusic(String,String,Client)
payer():void
debiter(double):boolean
crediter(double):void
authentifier(String):boolean
getNomUser():String
setNomUser(String):void
compareTo(CompteNetMusic):int
<<Java Class>>
Album
Q1
titre: String
auteur: String
maisonProd: String
annee: int
listChansons: String[]
prix: double
Album(String,String,String,int,String[],double,Style)
calculerPrix():double
afficher():void
getStyle():Style
setStyle(Style):void
getPrix():double
setPrix(double):void
-panier
0..*
-client
0..1
<<Java Class>>
Client
Q1
nom: String
prenom: String
adresse: String
Client(String,String,String)
<<Java Enumeration>>
Style
Q1
Classique: Style
Jazz: Style
Rap: Style
Pop: Style
Rock: Style
Rnb: Style
Style()
-style
0..1
<<Java Class>>
AlbumPhysique
Q1
fraisLivraison: double
delai: int
dateAchat: String
AlbumPhysique(String,String,String,int,String[],double,Style)
calculerPrix():double
getDateAchat():String
setDateAchat(String):void
<<Java Class>>
AlbumNumerique
Q1
URL: String
reduction: double
AlbumNumerique(String,String,String,int,String[],double,Style,String)
calculerPrix():double

Q2


«Java Interface»
Catalogue
Q2
consulter():void
ajouterProduit(Produit):boolean
supprimerProduit(Produit):boolean
-catalogueAlbums
0..1
-catalogueMagazine
0..1
«Java Class»
CatalogueAlbum
Q2
albums: HashMap<Style,TreeSet<Album>>
CatalogueAlbum()
consulter():void
ajouterProduit(Produit):boolean
supprimerProduit(Produit):boolean
«Java Class»
CatalogueMagazine
Q2
numeros: TreeMap<String,TreeMap<Integer,Magazine>>
CatalogueMagazine()
consulter():void
ajouterProduit(Produit):boolean
afficherNumeros(String):boolean
afficherNumeroMagazine(String,int):boolean
supprimerProduit(Produit):boolean
«Java Class»
NetMusic
Q2
NetMusic()
authentifier(String,String):CompteNetMusic
ajouterCompte(CompteNetMusic):boolean
-comptes
0..*
«Java Class»
CompteNetMusic
Q2
nomUser: String
motdePasse: String
solde: double
CompteNetMusic(String,String,Client)
payer():void
authentifier(String):boolean
getNomUser():String
setNomUser(String):void
-client
0..1
«Java Class»
Client
Q2
nom: String
adresse: String
compte: String
Client(String,String,String)
debiter(double):boolean
-panier
0..*
«Java Class»
Produit
Q2
titre: String
reference: String
nbrUnite: int
Produit()
calculerPrix():double
getTitre():String
setTitre(String):void
«Java Interface»
Remboursable
Q2
pourcentage: double
rembourser(String):boolean
«Java Class»
Magazine
Q2
numero: int
dateApparition: String
sommaire: String
url: String
Magazine(String,int,String,String,String)
afficher():void
afficherSommaire():void
calculerPrix():double
getNumero():int
setNumero(int):void
rembourser(String):boolean
«Java Class»
AlbumPhysique
Q2
AlbumPhysique(String,String,String,int,ArrayList<String>,double,Style)
calculerPrix():double
rembourser(String):boolean
«Java Class»
Album
Q2
auteur: String
maisProd: String
annee: int
listChansons: ArrayList<String>
prix: double
Album(String,String,String,int,ArrayList<String>,double,Style)
getStyle():Style
setStyle(Style):void
compareTo(Album):int
-style
0..1
«Java Enumeration»
Style
Q2
Classique: Style
Jazz: Style
Rap: Style
Pop: Style
Rock: Style
Rnb: Style
Style()
«Java Class»
AlbumNumerique
Q2
URL: String
AlbumNumerique(String,String,String,int,ArrayList<String>,double,Style,String)
calculerPrix():double

Autre solution