Université de Carthage
**Ecole Supérieure de la Statistique et de l'Analyse de l'Information**

**Enseignante : Mme Aïcha El Golli**

**Niveau d'études : 2ème année**

**Année Universitaire 2023/2024**

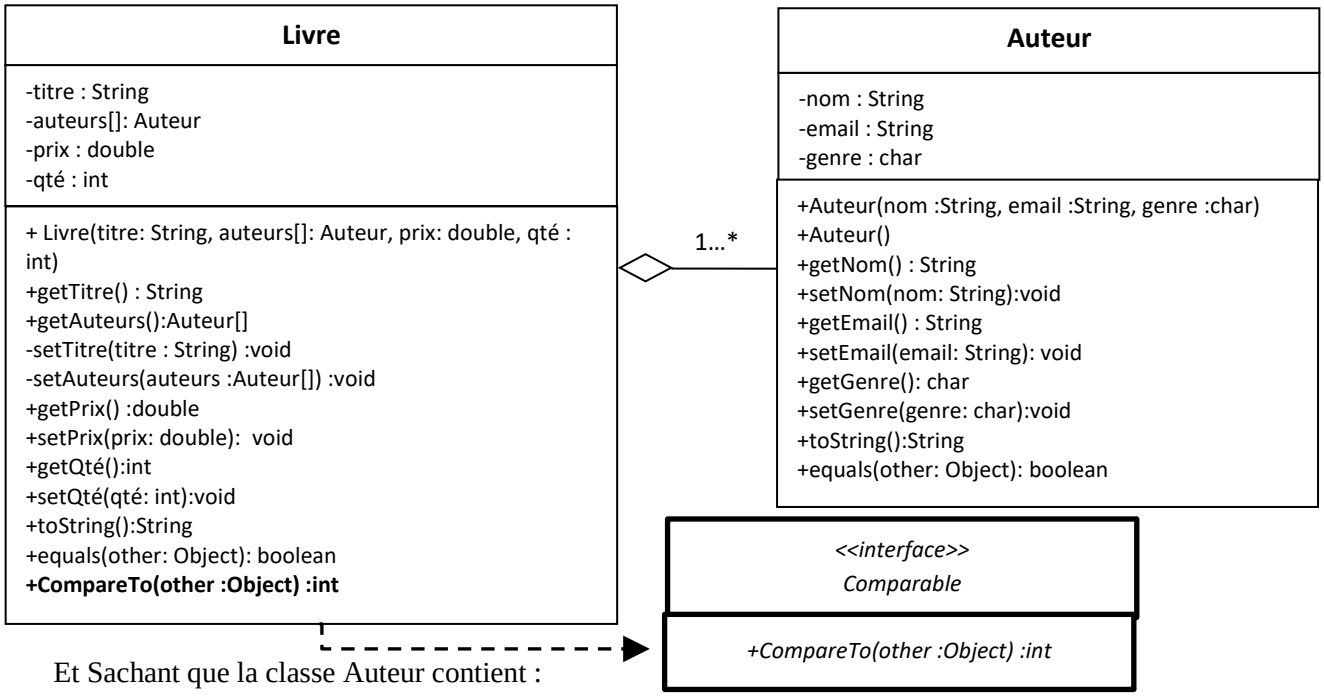
**Examen Session Principale**
**Module :** POO -Java niveau I

**Durée de l'épreuve :** 1h30

**Remarques** : *Les Documents, calculatrices, **téléphone portable sont interdits**. Veuillez rendre une copie propre et claire. Si la syntaxe d'une instruction est fausse alors la note est 0. La qualité de l'écriture et de la présentation sera prise en compte dans la note finale.*

## Exercice 1 (14 Points)

Soit le schémas UML suivant présentant une association d'agrégation entre les classes Livre et Auteur :




Et Sachant que la classe Auteur contient :

- Trois attributs d'instance privés à savoir le nom (String), email (String) et genre (caractère soit 'm' soit 'f' ; Un constructeur complet qui initialise le nom, l'email et le genre avec les valeurs données. Un constructeur par défaut qui initialise le nom et l'email à une chaine vide et le genre à 'm'. Des getters/setters publiques, le setter du genre n'acceptant que les caractères 'm' et 'f' ;
- Une redéfinition de la méthode toString() qui retourne une représentation textuelle d'un auteur sous la forme « Auteur [nom= ?, email= ?, genre= ?] » et une redéfinition de la méthode equals().

**1) Ecrire que la classe Livre contenant (11 points) :**

- Quatre variables d'instance privées : titre (String), auteurs (de type un tableau d'Auteur en supposant qu'un livre a un à plusieurs auteurs), le prix du livre (double) et la quantité (int) ;
- Des getters pour les attributs titre et auteurs, les setters sont privés pour ces attributs car ils ne peuvent être utilisés que par le constructeur et après les attributs ne peuvent plus être changés ; le titre ne doit pas être null et dans ce cas il sera initialisé à une chaine vide et le tableau des auteurs doit aussi être non nuls. Ecrire le setter des auteurs en **émettant une exception adaptée** en cas de null.
- Un constructeur complet ;
- Des getters/setters pour le prix et la quantité (qui doivent être positifs);
- Une redéfinition pour la méthode toString() qui retourne la chaine suivante : "Livre[titre=?, Auteur [nom=?, email=?, genre=?], Auteur[nom=?, email=?, genre=?], …, prix=?, qté=?]". Il faudra appeler la méthode toString() de la classe Auteur.
- Une redéfinition pour la méthode equals() ; deux livres sont les mêmes si ils ont les mêmes auteurs et le même titre.
- La classe Livre doit aussi implémenter l'interface Comparable suivante :

```
public interface Comparable
{int compareTo ( Object other ) ; }
```

l'unique méthode compareTo, renvoie un nombre entier qui représente le résultat de la comparaison. La mise en œuvre de cette méthode doit renvoyer un nombre entier basé sur la différence des quantités des livres:

- 0 si cette instance du livre et other ont la même quantité (la différence)
- > 0 si cette instance du livre a une quantité supérieure à celle de other
- < 0 si cette instance du livre a une quantité inférieure à celle de other

```
public class Livre implements Comparable { //(0,5pt)
              // Variables
              private String titre;//(0,5pt)
              private Auteur[] auteurs; //(0,5pt)
              private int qté;//(0,5pt)
              private double prix ;//(0,5pt)
              // Constructeurs
public Livre(String titre,Auteur[]auteurs, double prix, int qté) {//(1 pt)
               setAuteurs(auteurs) ;
                setTitre(titre);
                setPrix(prix);
                setQté(qté);    }
             // Accesseur
             public Auteur[] getAuteur() {//(0,5pt)
                return auteurs;  }

              public String getTitre() {//(0,5pt)
                return titre;   }

              public int getQté() {//(0,5pt)
                return qté;     }

              public double getPrix() {//(0,5pt)
                return prix;  }
              // Modificateur

              public void setQté(int n) {//(0,5pt)
                if (n > 0) qté = n;
          else System.out.println("Erreur : Quantité négative !") ;}
              private void setTitre(String titre) {//(0,5pt)
                  if( titre== null)
                        titre= » »;
                        else
                              this.titre=titre; }


            private void setAuteurs(Auteur[] auteurs) {//(1pt)
                  if( auteurs== null)
                        throw new IllegalArgumentException("La variable auteur
ne doit pas être null ");
                        else
                              this.auteurs = auteurs;
            }

            public void setPrix(double prix) {//(0,5pt)
               if (prix >= 0.0)
                  this.prix = prix;
```

```
                else
                   System.out.println("Erreur : prix négatif !");}

                public String toString() {//(1pt)
                 String s=   "Livre [titre=" + titre;// + ", ";
                for(Auteur a: auteurs) {
                      s+=", ";
                     s+=a ;}
                   s+= ", prix=" + prix + ", qté=" + qté+"]";
                   return s;
                }
                //Livre[nom=?,Auteur[nom=?,email=?,genre=?],prix=?,qté=?].

            @Override
            public boolean equals(Object o) {//(1pt)
if (o==this)return true;
if (!(o instanceof Livre)) return false;
                        Livre l= (Livre)o;
                        if(l.auteurs.length!=auteurs.length) return false;
                        for(int i=0; i<auteurs.length; i++) {
                              if(!l.auteurs[i].equals(auteurs[i])) return
false;
                        }
                        return(titre.equals(l.titre));
            }
            public int compareTo ( Object o ) {//(1pt)
                  Livre l =(Livre)o;
                  return (qté-l.qté);
            }

}
```

**2) Donnez l'affichage du code suivant (3 points) :**

```
Auteur a1= new Auteur("Delonney","delonney@gmail.com", 'm');
Auteur a2= new Auteur("Herby","Herby@gmail.com", 'm');
Comparable c1=null,  c2=null;
try {
            Object o=new Livre("Exercices en java", new Auteur[]{a1 ,a2}, 27.650,35);
            System.out.println(o);
            c1=(Comparable) o;
            c2=(Comparable) a1;
    }
catch(ClassCastException e) {
            System.out.println("Erreur de class cast");}
catch(Exception e1) {
            System.out.println("Exception");}
finally {
    if(c1==null)
            System.out.println("attention c1 est null");
    if(c2==null)
            System.out.println("attention c2 est null");      }
```

```
Livre [titre=Exercices en java, Auteur [nom=Delonney,
email=delonney@gmail.com, genre=m], Auteur [nom=Herby,
email=Herby@gmail.com, genre=m], prix=27.65, qté=35]
Erreur de class cast
attention c2 est null
```

**Questions de réflexion (6 points)**

1) Ayant la déclaration suivante : String[] t= new String[]{"hello", null, ""} ; que retournent chacune des instructions suivantes :
   a) t[1].equals(t[2]) ? levée d'exception
   b) t[2].equals(t[1]) ? false
   c) t[0].charAt(0) ? h
2) Soit une interface Java I, et deux classes C1 et C2 qui l'implémentent. Indiquer si chaque instruction est correcte ou non :
   a) I x = new I() ; non correcte
   b) I[] z = {new C1(), new C2()} ; correcte
3) Ayant une classe Livre définie et ayant la déclaration suivante : Livre t[]=new Livre[4] ; que retourne l'instruction : t[0].getTitre() ; ? levée d'exception
4) Que signifie le mot clef "static" associé à un attribut ?
   (a) Que la valeur de cet attribut est constante
   (b) Que cet attribut sera toujours passé par valeur
   (c) Que cet attribut a une valeur unique pour toutes les instances de la classe
5) Pour le mot clef "abstract", quelles assertions sont fausses ?
   (a) Une classe abstraite ne peut être instanciée
   (b) Une méthode abstraite n'a pas d'implémentation
   (c) Une classe abstraite n'a pas forcément de classe fille
   (d) Une classe abstraite doit contenir au moins une méthode abstraite
6) Pour la classe ArrayList, quelles assertions sont justes ?
   (a) la classe ArrayList implémente les tableaux dynamiques
   (b) la classe ArrayList implémente les listes chaînées
   (c) la classe ArrayList appartient au package util de java
   (d) sa méthode public boolean add(E e), permet l'ajout d'une référence qui pointe le null

***Bon travail***