Nom :

Prénom :

DS POO 2019-2020

1ère session

Tous documents autorisés, durée : 2h

L'objectif du sujet est de réaliser un programme permettant de consulter les documents disponibles d'une médiathèque.

Le sujet est simplifié, on se contentera de proposer des fonctionnalités de consultation et de tri de documents.

Le sujet comporte plus de points que nécessaire pour obtenir la note maximale. La moyenne se situera autour de 40 points.

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).

➢ ***Les questions sont en gras italique comme ceci précédées d'une flèche.***

Le barème est sur 100 ; il est indicatif

1 Cahier des charges :

Une médiathèque propose trois types de documents :

- des livres ;
- des CDRom ;
- des Oeuvre d'art.

Tous ces documents sont **consultables** sur place.

En **synthèse**, nous préciserons également si le document est **empruntable** ou non.

1.1 Code

Le code est organisé en 3 packages : model, ui et test.

Le package test contient une seule classe de test.

Le code est volontairement peu commenté.

Pour simplifier, on considère que tout document dispose d'un numéro ISBN unique (normalement, cela concerne uniquement les livres). Un numéro ISBN identifie une **référence** de document (même titre, même auteur). **Une même référence peut se retrouver dans plusieurs documents, qui seront alors différenciés par un numéro d'exemplaire.**

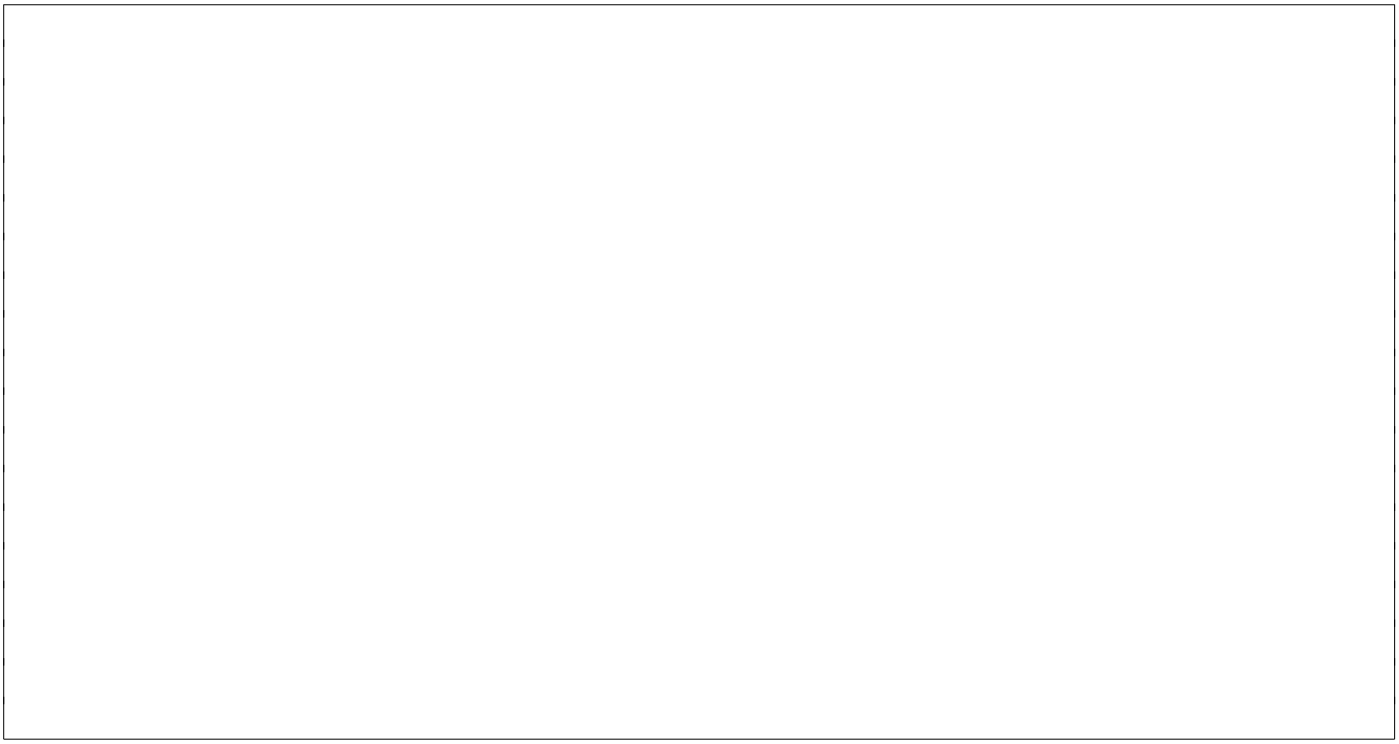
2 Le modèle de base (34 points)

- Prenez connaissance des fragments de code suivants et de la trace d'exécution du programme de test.

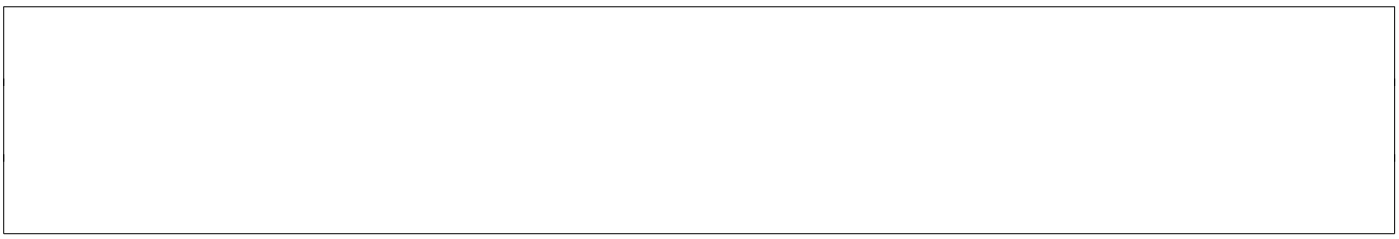
2.1 Code du modèle et du Launcher

```
1  package model.documents;
2  public interface DocumentConsultable extends Comparable<DocumentConsultable>{
3      public boolean consulter();
4      public int getISBN();
5      public String getTitre();
6      public String getAuteur();
7      public void setNumExemplaire(int num);
8  }
9
10 package model.documents;
11 public interface DocumentEmpruntable                    DocumentConsultable {
12     public boolean emprunter(int duree);
13     public boolean restituer(int duree);
14     public boolean declarerPerte();
15 }
16
17 package model.documents;
18 public enum Genre {
19     POLICIER, DOCUMENTAIRE, BANDE_DESSINEE, ROMAN
20 }
21
22 package model.documents;
23 public                    class AbstractDocument                                    {
24
25     private int numISBN;
26     private int numExemplaire;
27     private String titre;
28     private String auteur;
29     private boolean disponible;
30
31     public AbstractDocument(int numISBN, String titre, String auteur) {
                                                                                       
                                                                                       
                                                                                       
                                                                                       
32     }
33
34
35     public AbstractDocument(int numISBN, String titre) {
36         (numISBN, titre, "inconnu");
37     }
38
39     private void setNum(int numISBN) {
40       this.numISBN=0;
41       if(numISBN>0 && numISBN>100) {
42             this.numISBN = numISBN;
43       }
44     }
```

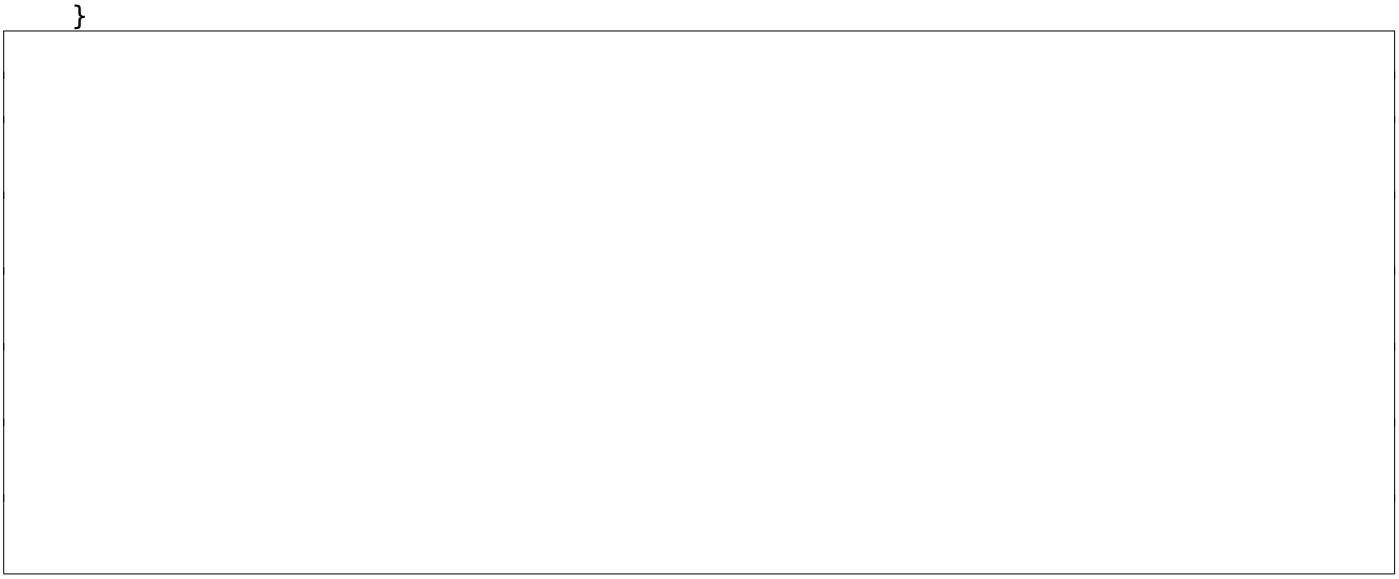
```
46      @Override
47      public boolean consulter() {
48        boolean retour = false;
49        if(isDisponible()) {
50              disponible=false;
51              retour= true;
52        }
53        return retour;
54      }
55
56      private boolean isDisponible() {
57        return disponible;
58      }
59
60      @Override
61      public int getISBN() {
62        return numISBN;
63      }
64
65      public String getTitre() {
66        return titre;
67      }
68
69      public String getAuteur() {
70        return auteur;
71      }
72
73      public void setNumExemplaire(int num) {
74        numExemplaire = num;
75      }
76
77      @Override
78      public int compareTo(DocumentConsultable o) {
79        return o.getISBN()-getISBN();
80      }
81
82      @Override
83      public int hashCode() {
84        final int prime = 31;
85        int result = 1;
86        result = prime * result + numExemplaire;
87        result = prime * result + numISBN;
88        return result;
89      }
90
91      @Override
92      public boolean equals(Object obj) {
93        if (this == obj)
94              return true;
95        if (obj == null)
96              return false;
97        if (getClass() != obj.getClass())
98              return false;
99        AbstractDocument other = (AbstractDocument) obj;
100       if (numExemplaire != other.numExemplaire)
101             return false;
102       if (numISBN != other.numISBN)
103             return false;
104       return true;
105     }
106
```



```
}

package model.documents;

public                  class Livre                                    {

    private Genre genre;
    public Livre(int numISBN, String titre, String auteur, Genre genre) {
```

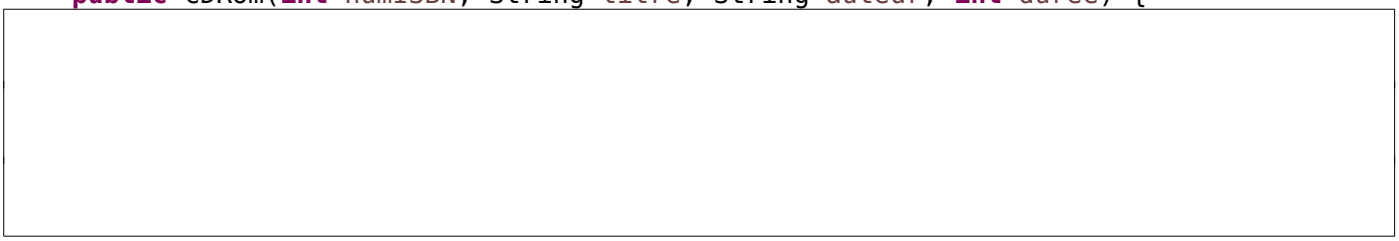


```
    }
    public Genre getGenre() {
      return genre;
    }
```

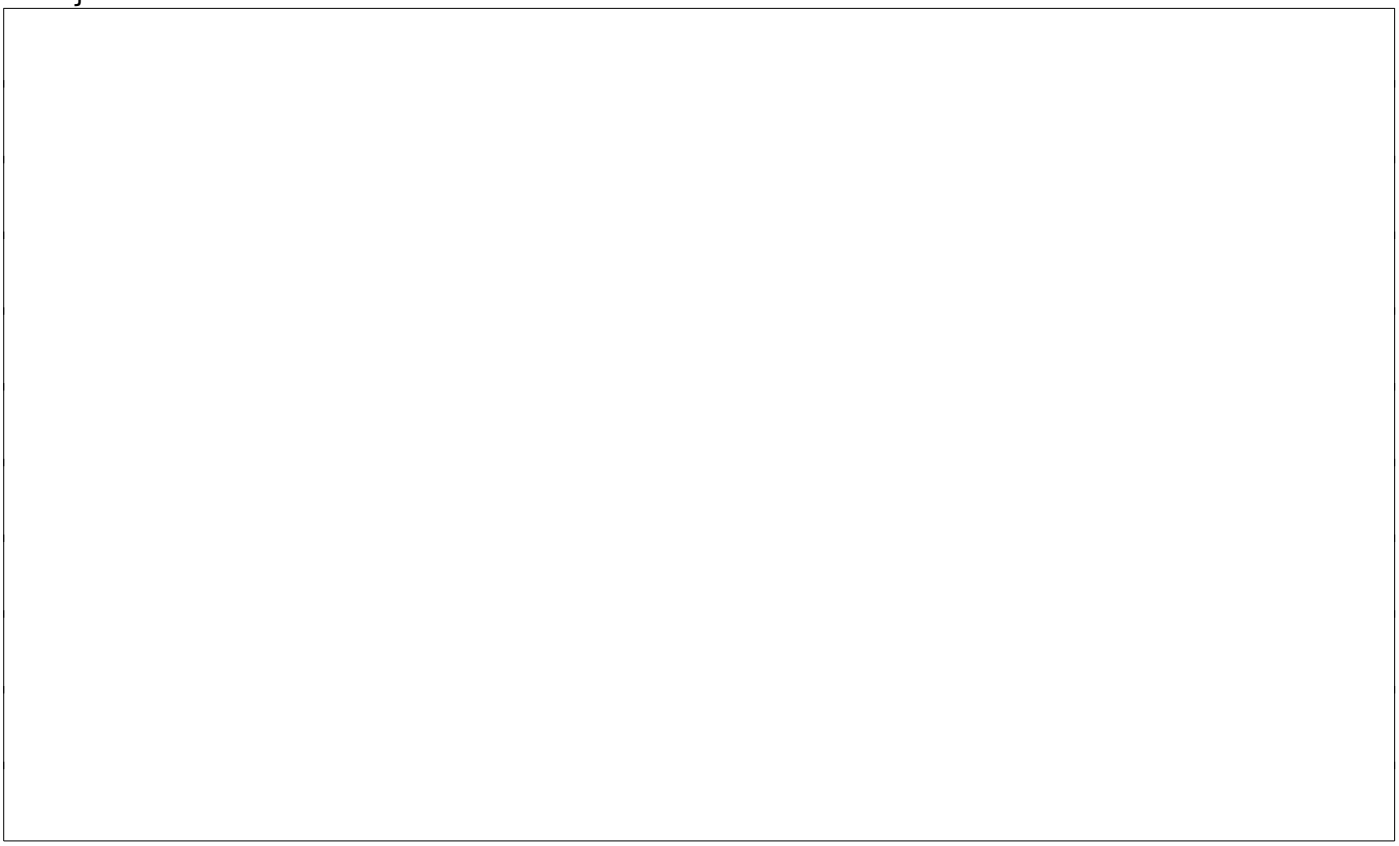


```
}
```

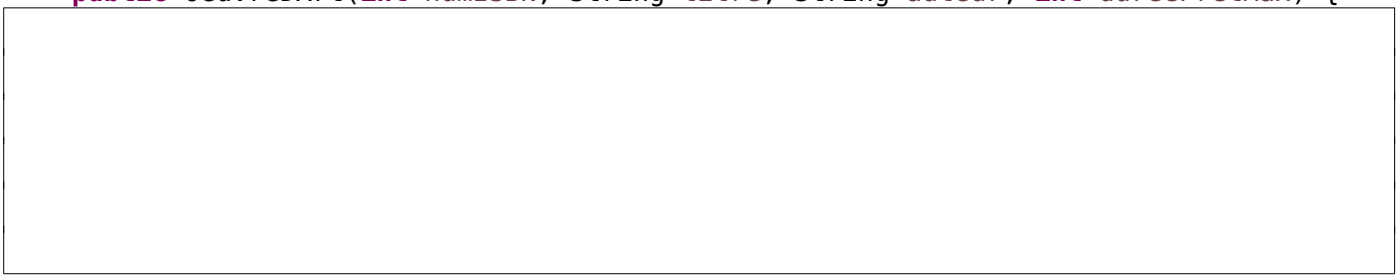
```java
package model.documents;
public class CDRom {

    private int duree;

    public CDRom(int numISBN, String titre, String auteur, int duree) {
```



```java
    }
```

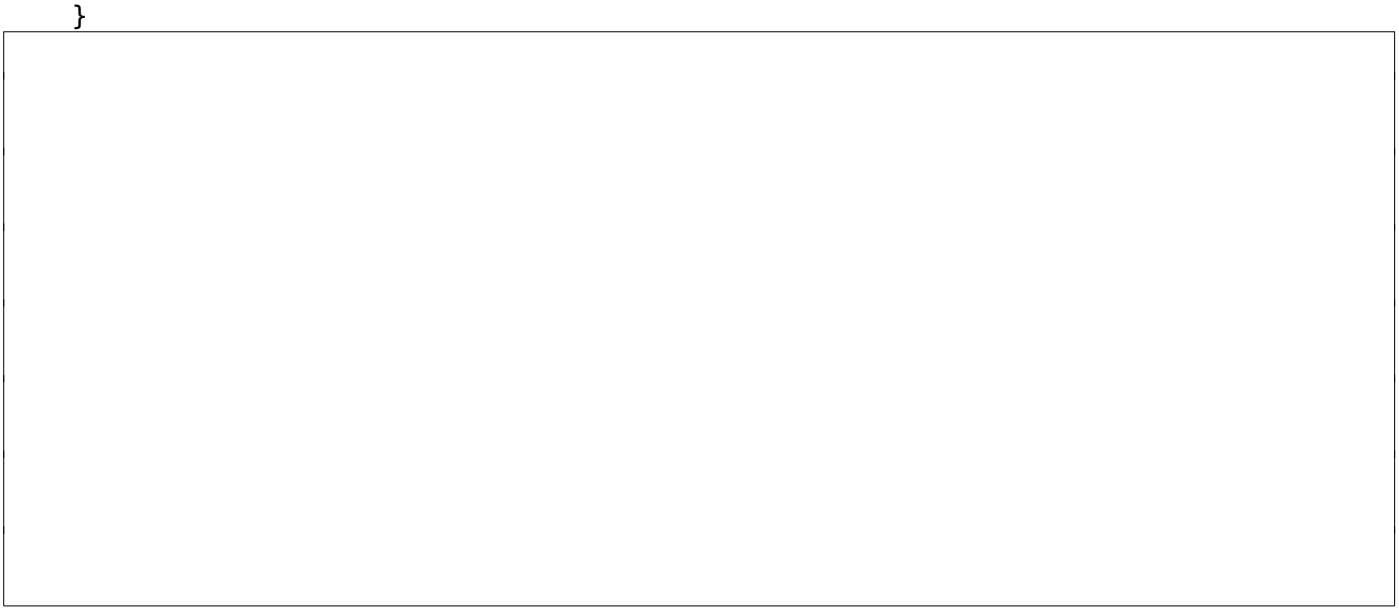


```java
}

package model.documents;
import model.Mediatheque;

public class OeuvreDArt extends AbstractDocument {

    private int dureePretMax;

    public OeuvreDArt(int numISBN, String titre, String auteur, int dureePretMax) {
```

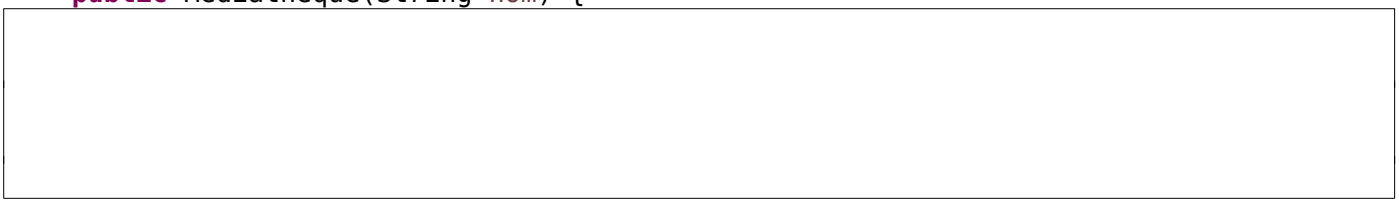


```java
}
```

```
138      public void setDureePretMax(int dureePretMax) {
139        if(dureePretMax>=0 && dureePretMax<Mediatheque.PRET_MAX) {
140              this.dureePretMax = dureePretMax;
141        }
142        else {
143              this.dureePretMax=0;
144        }
145
146      }
147
148      public int getDureePretMax() {
149        return dureePretMax;
150      }
```

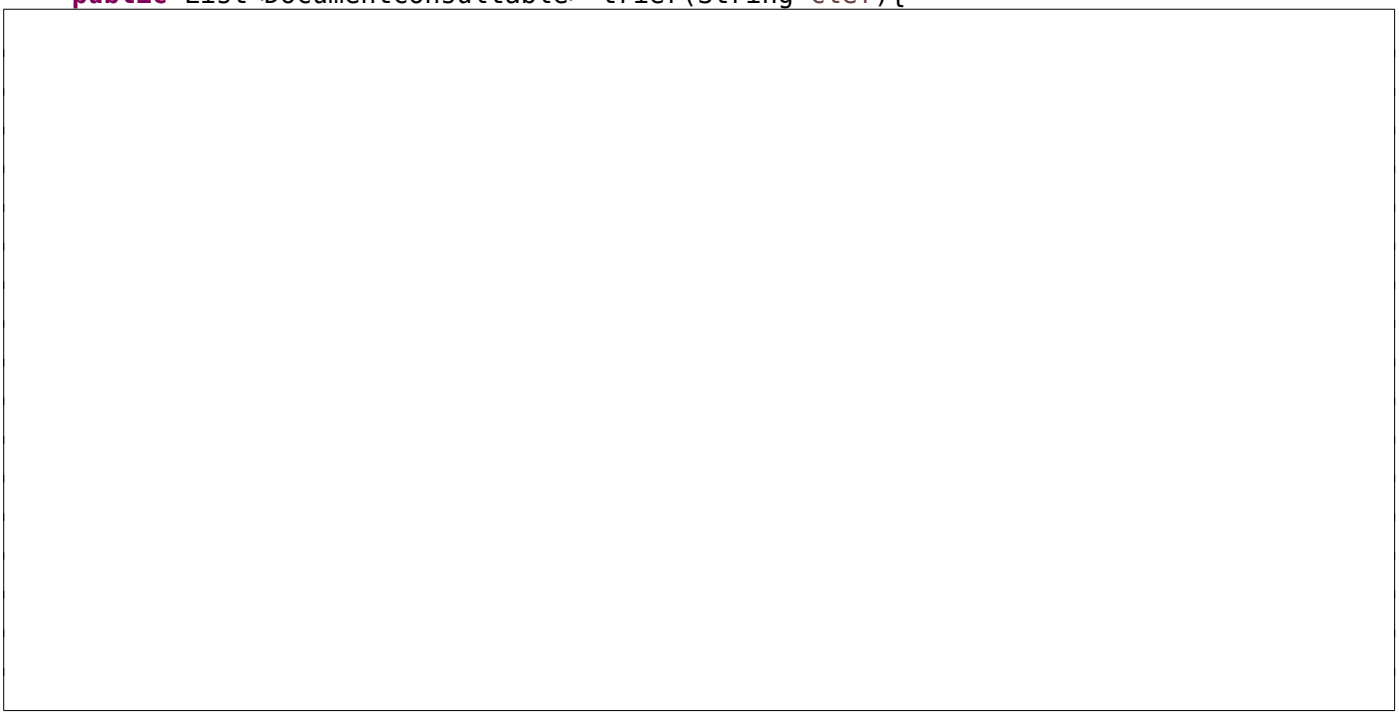


```
151  }
152
153  package model;
154
155  import …….....
156
157  import model.documents.DocumentConsultable;
158
159  public class Mediatheque {
160
161      public static final int PRET_MAX = 90;
162      private Set<DocumentConsultable> collection;
163      private String nom;
164      private Map<String,Comparator<DocumentConsultable>> comparateurs;
165
166      public Mediatheque(String nom, Set<DocumentConsultable> collection) {
167        this.collection = collection;
168        this.nom = nom;
169
170        comparateurs = new HashMap<String,Comparator<DocumentConsultable>>();
171        comparateurs.put("TITRE", new ComparateurTitre());
172        comparateurs.put("AUTEUR", new ComparateurAuteur());
173      }
174
175      public Mediatheque(String nom) {
```



```
176       }
177
178       private void addComparateur(String type, Comparator<DocumentConsultable>
179   comparateur) {
180         comparateurs.put(type,comparateur);
181       }
182
183
184       public String toString() {
185         String ret = "Médiathèque "+nom;
186         return ret;
187       }
188
189       public Set<DocumentConsultable> documentsParTitre(){
190         Set<DocumentConsultable> ret =
191         ret.addAll(collection);
192         return ret;
193       }
194
195       public Set<DocumentConsultable> documentsParAuteur(){
196         Set<DocumentConsultable> ret =
197         ret.addAll(collection);
198         return ret;
199       }
200
201       public List<DocumentConsultable> documentsParTitre2(){
202         List<DocumentConsultable> ret =                    ;
203         ret.addAll(collection);
204
205         return ret;
206       }
207
208       public List<DocumentConsultable> documentsParAuteur2(){
209         List<DocumentConsultable> ret =                    ;
210         ret.addAll(collection);
211
212
213         return ret;
214       }
```

```
215      public Map<String, Comparator<DocumentConsultable>> getComparators(){
216        return comparateurs;
217      }
218
219      public List<DocumentConsultable> trier(String clef){
```



```
220      }
221
222  }
223
224  package launcher;
225  import java.util.Collection;
226  import java.util.HashSet;
227  import java.util.List;
228  import java.util.Set;
229  import ui.Fenetre;
230  import model.Mediatheque;
231  import model.documents.CDRom;
232  import model.documents.DocumentConsultable;
233  import model.documents.Genre;
234  import model.documents.Livre;
235  import model.documents.OeuvreDArt;
236
237  public class TestModel {
238
239      public static void main(String[] args) {
240        Set<DocumentConsultable> documents = createDocumentsCollection();
241
242        Mediatheque mediatheque1 = new Mediatheque("Réné Char", documents);
243        System.out.println(mediatheque1);
244
245        System.out.println("Par titre");
246        Set<DocumentConsultable> docTitre = mediatheque1.documentsParTitre();
247        afficheCollection(docTitre);
248
249        System.out.println("Par auteur:");
250        Set<DocumentConsultable> docAuteur = mediatheque1.documentsParAuteur();
251        afficheCollection(docAuteur);
252
253        System.out.println("Par titre 2ème technique");
254        List<DocumentConsultable> docTitre2 = mediatheque1.documentsParTitre2();
```

```
255         afficheCollection(docTitre2);
256 
257         System.out.println("Par auteur 2ème technique");
258         List<DocumentConsultable> docAuteur2 = mediatheque1.documentsParAuteur2();
259         afficheCollection(docAuteur2);
260 
261         new Fenetre("Recherche de documents", mediatheque1);
262       }
263 
264       private static void afficheCollection(Collection<DocumentConsultable> col) {
265         for(DocumentConsultable doc: col) {
266               System.out.println("\t* "+doc);
267         }
268 
269       }
270 
271       public static Set<DocumentConsultable> createDocumentsCollection() {
272         Set<DocumentConsultable> ret = new HashSet<DocumentConsultable>();
273 
274         ret.add(new Livre(1000, "Au bonheur des dames", "Emile Zola", Genre.ROMAN));
275         ret.add(new Livre(2000, "Les passagers du vent, tome 1", "Bourgeons",
276 Genre.BANDE_DESSINEE));
277         ret.add(new Livre(3000, "Le crime de l'orient express", "Agatha Christie",
278 Genre.POLICIER));
279         ret.add(new Livre(3500, "ABC contre Poirot", "Agatha Christie",
280 Genre.POLICIER));
281         ret.add(new Livre(4000, "La terre vue du ciel", "Yann Arthus Bertrand",
282 Genre.DOCUMENTAIRE));
283 
284         DocumentConsultable livre = new Livre(1000, "Au bonheur des dames", "Emile
285 Zola", Genre.ROMAN);
286         livre.setNumExemplaire(2);
287         ret.add(livre);
288 
289         ret.add(new CDRom(16000, "Le lac des cygnes", "Piotr Tchaïkowsky", 145));
290         ret.add(new CDRom(18000, "La terre vue du ciel", "Yann Arthus Bertrand", 96));
291 
292         ret.add(new OeuvreDArt(28000, "La terre vue du ciel", "Yann Arthus Bertrand",
293 60));
294         ret.add(new OeuvreDArt(29000, "Fac simile Aurore, J'accuse", "Emile Zola", 80));
295 
296         return ret;
297       }
298 
299 }
300 
```

2.2 Sortie console

```
Médiathèque Réné Char
Nous gérons 10 références
     - Livre (Les passagers du vent, tome 1, auteur: Bourgeons, num: 2000, genre:
BANDE_DESSINEE)
     - Livre (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 4000, genre:
DOCUMENTAIRE)
     - CD-ROM (Le lac des cygnes, auteur: Piotr Tchaïkowsky, num: 16000, durée 145
     - CD-ROM (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 18000, durée 96
     - Oeuvre d'art (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 28000, durée
maximale de prêt 60
     - Livre (Au bonheur des dames, auteur: Emile Zola, num: 1000, genre: ROMAN)
     - Livre (Au bonheur des dames, auteur: Emile Zola, num: 1000, genre: ROMAN)
     - Livre (Le crime de l'orient express, auteur: Agatha Christie, num: 3000, genre: POLICIER)
     - Oeuvre d'art (Fac simile Aurore, J'accuse, auteur: Emile Zola, num: 29000, durée maximale
de prêt 80
     - Livre (ABC contre Poirot, auteur: Agatha Christie, num: 3500, genre: POLICIER)
Par titre
     * Livre (ABC contre Poirot, auteur: Agatha Christie, num: 3500, genre: POLICIER)
     * Livre (Au bonheur des dames, auteur: Emile Zola, num: 1000, genre: ROMAN)
     * Oeuvre d'art (Fac simile Aurore, J'accuse, auteur: Emile Zola, num: 29000, durée maximale
de prêt 80
     * Livre (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 4000, genre:
DOCUMENTAIRE)
     * Livre (Le crime de l'orient express, auteur: Agatha Christie, num: 3000, genre: POLICIER)
     * CD-ROM (Le lac des cygnes, auteur: Piotr Tchaïkowsky, num: 16000, durée 145
     * Livre (Les passagers du vent, tome 1, auteur: Bourgeons, num: 2000, genre:
BANDE_DESSINEE)
Par auteur:
     * Livre (Le crime de l'orient express, auteur: Agatha Christie, num: 3000, genre: POLICIER)
     * Livre (Les passagers du vent, tome 1, auteur: Bourgeons, num: 2000, genre:
BANDE_DESSINEE)
     * Livre (Au bonheur des dames, auteur: Emile Zola, num: 1000, genre: ROMAN)
     * CD-ROM (Le lac des cygnes, auteur: Piotr Tchaïkowsky, num: 16000, durée 145
     * Livre (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 4000, genre:
DOCUMENTAIRE)
Par titre 2ème technique
     * Livre (ABC contre Poirot, auteur: Agatha Christie, num: 3500, genre: POLICIER)
     * Livre (Au bonheur des dames, auteur: Emile Zola, num: 1000, genre: ROMAN)
     * Livre (Au bonheur des dames, auteur: Emile Zola, num: 1000, genre: ROMAN)
     * Oeuvre d'art (Fac simile Aurore, J'accuse, auteur: Emile Zola, num: 29000, durée maximale
de prêt 80
     * Livre (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 4000, genre:
DOCUMENTAIRE)
     * CD-ROM (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 18000, durée 96
     * Oeuvre d'art (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 28000, durée
maximale de prêt 60
     * Livre (Le crime de l'orient express, auteur: Agatha Christie, num: 3000, genre: POLICIER)
     * CD-ROM (Le lac des cygnes, auteur: Piotr Tchaïkowsky, num: 16000, durée 145
     * Livre (Les passagers du vent, tome 1, auteur: Bourgeons, num: 2000, genre:
BANDE_DESSINEE)
Par auteur 2ème technique
     * Livre (Le crime de l'orient express, auteur: Agatha Christie, num: 3000, genre: POLICIER)
     * Livre (ABC contre Poirot, auteur: Agatha Christie, num: 3500, genre: POLICIER)
     * Livre (Les passagers du vent, tome 1, auteur: Bourgeons, num: 2000, genre:
BANDE_DESSINEE)
     * Livre (Au bonheur des dames, auteur: Emile Zola, num: 1000, genre: ROMAN)
     * Livre (Au bonheur des dames, auteur: Emile Zola, num: 1000, genre: ROMAN)
     * Oeuvre d'art (Fac simile Aurore, J'accuse, auteur: Emile Zola, num: 29000, durée maximale
de prêt 80
     * CD-ROM (Le lac des cygnes, auteur: Piotr Tchaïkowsky, num: 16000, durée 145
     * Livre (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 4000, genre:
DOCUMENTAIRE)
     * CD-ROM (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 18000, durée 96
* Oeuvre d'art (La terre vue du ciel, auteur: Yann Arthus Bertrand, num: 28000, durée maximale
de prêt 60
```

- ***Faites un schéma représentant les classes et les interfaces du package modèle. Précisez les liens de composition (bonus si vous détaillez agrégation et composition) d'héritage et d'implémentation. (4 points) (*)***

- ***Complétez dans le code les encadrés permettant de coder la hiérarchie des documents (4 points) (*)***

- ***Une interface peut-elle hériter d'une autre interface ? Quelle(s) conséquence(s) cela implique quand une classe veut implémenter l'interface fille ? (2 points) (**)***

- ***Codez les constructeurs des classes concrètes de document (6 points) (*)***

  Pensez bien à appliquer les bonnes pratiques de maintenabilité.

- ***Ajoutez le code permettant d'obtenir un affichage de document tel que produit dans la sortie console (6 points) (*/**)***

- ***Complétez le deuxième constructeur de Médiathèque (2 points)(**)***

  Pensez bien à TOUT ce que doit faire un constructeur...

- ***Étudiez la méthode de test `createDocumentsCollection()` ligne 271. Quelle collection concrète est utilisée en retour ? Expliquez comment les éléments de cette collection sont organisés et ordonnés (4 points) (**)***

➢ ***Expliquer les quatre techniques permettant de déterminer si deux documents sont identiques, et ce qui est utilisé (méthode, opérateur,...) dans chacune de ces techniques (6 points) (*)***

3 Tri des documents dans mediatheque (38 points)

On se propose d'ajouter des fonctionnalités dans la classe `Mediatheque` afin de retourner des collections triées de documents.

On envisage pour le moment deux types de tris :

- par nom d'auteur
- par nom de document (titre)

3.1 Justifications préliminaires (10 points)

On dispose de deux classes implémentant ***Comparator*** pour les comparaisons dont le code est donné juste après.

```
366 package model;
367
368 import java.util.Comparator;
369 import model.documents.DocumentConsultable;
370
371 public class ComparateurAuteur implements Comparator<DocumentConsultable> {
372
```

```
373     public String toString() {
374       return "AUTEUR";
375     }
376
377 }
378
```

```
379 package model;
380
381 import java.util.Comparator;
382 import model.documents.DocumentConsultable;
383
384 public class ComparateurTitre implements Comparator<DocumentConsultable> {
```

```
385     public String toString() {
386       return "TITRE";
387     }
388
389 }
390
```

- ***Sachant que la classe String implémente l'interface Comparable, complétez les deux comparateurs avec la méthode manquante (4 points) (*/**)***

- ***Quelles abstractions de collection sont utilisées dans les attributs de Mediatheque ? (2 points). (*)***

- ***Quelles collections concrètes sont utilisées dans les attributs de Mediatheque ? Faites attention avant de répondre ! (2 points) (*)***

- ***Justifiez ces choix avec un bref argumentaire (2 points) (*)***

3.2 Première tentative (8 points)

On réalise deux méthodes dans une première approche en se basant sur une collection de type Set.

- La méthode `documentsParTitre()`
- La méthode `documentsParAuteur()`

➢ ***Complétez ces deux méthodes en utilisant les Comparator instanciés dans la Map comparateurs de la classe Mediatheque, et en instanciant un type concret pertinent pour la valeur de retour. (4 points) (**)***

➢ ***Critiquez cette première solution. Vous pouvez vous baser sur les résultats produits par la sortie console. (4 points) (**)***

3.3 Deuxième tentative (10 points)

Depuis java 8, l'interface List contient la méthode suivante :

`void`	`sort(Comparator<? super E> c)` Sorts this list according to the order induced by the specified `Comparator`.

On réalise deux autres méthodes dans une seconde approche en se basant sur une collection de type List.

- La méthode `documentsParTitre2()`
- La méthode `documentsParAuteur2()`

➢ ***Complétez ces deux méthodes en utilisant les Comparator instanciés dans la Map comparateurs de la classe Mediatheque, et en instanciant un type concret pertinent pour la valeur de retour. (6 points) (**/***)***

➢ ***Expliquez pourquoi cette solution résout le problème identifié dans la première solution (4 points) (**)***

3.4 Troisième itération (10 points)

➢ ***Quel principe de POO la solution 2 ne respecte-t-elle pas ? Pensez à ce qui se passerait si vous deviez coder un nouveau comportement de tri (2 points) (**)***

➢ ***Codez la méthode trier afin de corriger ce problème, sans réutiliser les méthodes documentsParTitre... ou DocumentParAuteur.... (8 points) (***)***

4 Corrections et conception (28 points)

4.1 Améliorations (8 points)

➢ ***Quel problème pose la méthode `getComparators` ligne 215 ? Sachant que l'interface Map n'hérite pas de Cloneable, mais qu'elle dispose de la méthode putAll() décrite ainsi :***

> void **putAll**(Map<? extends K,? extends V> m)
>
> Copies all of the mappings from the specified map to this map (optional operation). The effect of this call is equivalent to that of calling `put(k, v)` on this map once for each mapping from key `k` to value `v` in the specified map. The behavior of this operation is undefined if the specified map is modified while the operation is in progress.
>
> Parameters:
>
> `m` - mappings to be stored in this map

Proposez une solution permettant de corriger ce problème (4 points) (**)

➢ ***Que pensez-vous de la méthode `addComparateur` ligne 178 ? Quel usage devrait-il en être fait et pourquoi ? Est-ce le cas ? (4 points) (*)***

En vous basant sur l'existant, sans détailler le contenu du code mais en précisant dans le détail où se situent les modifications et en quoi elles consistent, prenez en considération les fonctionnalités suivantes :

- seuls les livres et les œuvres d'art sont empruntables (**4 points**) (*)
- savoir si un document est référencé mais n'est pas disponible (**4 points**) (**)
- obtenir la liste de tous les documents empruntés qui sont en retard (**4 points**). (**/***)

Il y a plusieurs solutions possibles.

Vous vous efforcerez de respecter les bons principes de la POO dans vos réponses en expliquant pourquoi. **La qualité de l'argumentaire est notée sur 8 points.**