UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

INFO0502 - Introduction à la programmation répartie

TP1 : Retour sur Java
Gestion d'une Médiathèque

Despoullains Romain

2024-2025

Table des matières

1 Introduction

Dans ce TP, nous allons mettre en place la gestion d'une médiathèque en Java. La première partie consistera en l'écriture des différentes classes liées aux médias, tandis que la seconde portera sur la gestion de la médiathèque elle-même.

2 Les Classes Liées aux Médias

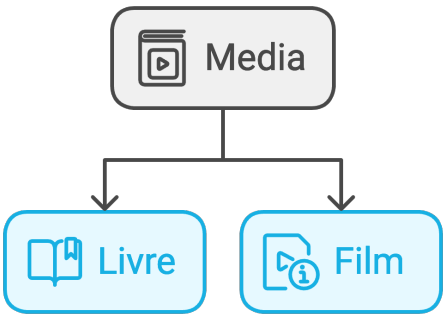

FIGURE 1 – Schema de l'hèritage

Nous allons mettre en place un système d'héritage où la classe `Media` sera la classe mère, et les classes `Livre` et `Film` seront les classes filles.

2.1 La Classe `Media`

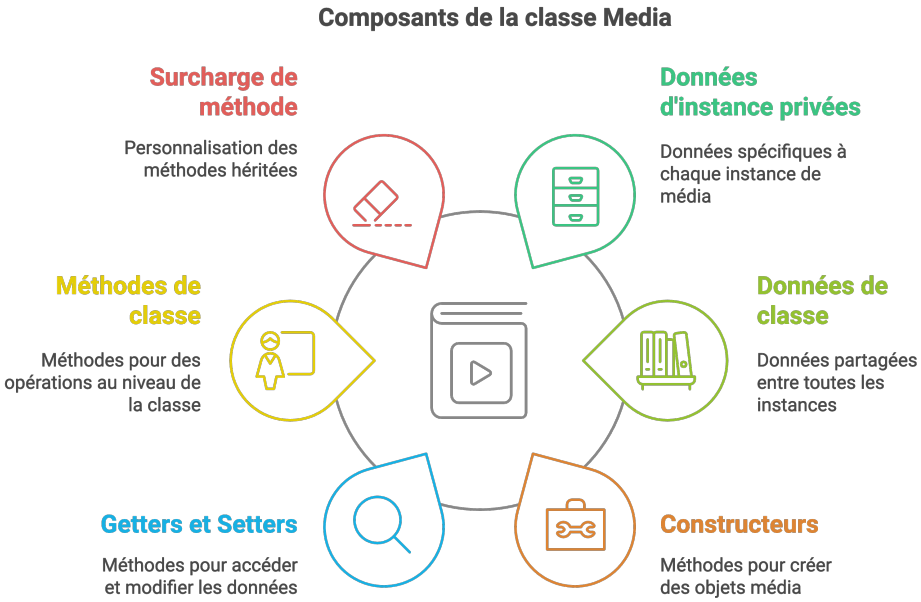

FIGURE 2 – Schema de classe de la classe `Media`.

La classe `Media` est constituée de :
- **Données d'instance privées** :
  - `titre` : le titre du média (`String`)
  - `cote` : la cote du média (`StringBuffer`)
  - `note` : la note attribuée au média (`int`)
- **Donnée de classe privée** :
  - `nom` : le nom de la médiathèque (`String`)
- **Méthodes** :
  - Trois constructeurs : par défaut, par initialisation et par copie
  - Les *getters* et *setters* associés aux données d'instance
  - Deux méthodes de classe :
    - Une permettant de modifier `nom`
    - Une permettant de récupérer `nom`
  - Le masquage des méthodes `clone`, `equals` et `toString`

2.2 Question 1

Donnez la structure mémoire d'un objet de type `Media`, ainsi que les différents accès possibles aux différents éléments de cet objet ou de la classe elle-même.

Réponse

Un objet de type `Media` contient :
- **Variables d'instance (privées)** :
  - `titre` : `String`
  - `cote` : `StringBuffer`
  - `note` : `int`
- **Variable de classe (privée)** :
  - `nom` : `String`

Accès aux éléments :
- Les variables d'instance privées ne sont accessibles qu'à l'intérieur de la classe `Media` elle-même.
- Les accès aux variables d'instance se font via les méthodes *getters* et *setters*.
- Les méthodes de classe pour accéder à `nom` sont statiques et permettent de modifier ou récupérer la variable de classe `nom`.

2.3 Question 2

Écrivez la classe `Media`.

Réponse

Voici l'implémentation de la classe `Media` :

```
1 /*
2 TP 1: Retour sur Java
3 Universit  de Reims Champagne Ardennes
4 Despoullains Romain
5 */
6
```

```java
public class Media {

    // Variable de classe (priv e)
    private static String nom;

    // Variables d'instance (priv es)
    private String titre;          // Titre du m dia
    private StringBuffer cote;     // Cote du m dia
    private int note;              // Note attribu e au m dia

    /**
     * Constructeur par d faut.
     * Initialise les variables d'instance avec des valeurs par d faut.
     */
    public Media() {
        this.titre = "";
        this.cote = new StringBuffer();
        this.note = 0;
    }

    /**
     * Constructeur par initialisation.
     * @param titre Le titre du m dia.
     * @param cote La cote du m dia.
     * @param note La note attribu e au m dia.
     */
    public Media(String titre, String cote, int note) {
        this.titre = titre;
        this.cote = new StringBuffer(cote);
        this.note = note;
    }

    /**
     * Constructeur par copie.
     * @param m L'objet Media    copier.
     */
    public Media(Media m) {
        this.titre = m.titre;
        this.cote = new StringBuffer(m.cote);
        this.note = m.note;
    }

    // Getters et setters pour les variables d'instance

    /**
     * Obtient le titre du m dia.
     * @return Le titre.
     */
    public String getTitre() {
        return titre;
    }

    /**
     * D finit le titre du m dia.
     * @param titre Le titre    d finir.
     */
    public void setTitre(String titre) {
        this.titre = titre;
```

```
65      }
66
67      /**
68       * Obtient une copie de la cote du m dia.
69       * Retourne un nouveau StringBuffer pour  viter  la modification
             externe.
70       * @return Une copie de la cote.
71       */
72      public StringBuffer getCote() {
73          return new StringBuffer(this.cote);
74      }
75
76      /**
77       * D finit la cote du m dia.
78       * @param cote La cote    d finir.
79       */
80      public void setCote(String cote) {
81          this.cote = new StringBuffer(cote);
82      }
83
84      /**
85       * Obtient la note du m dia.
86       * @return La note.
87       */
88      public int getNote() {
89          return note;
90      }
91
92      /**
93       * D finit la note du m dia.
94       * @param note La note    d finir.
95       */
96      public void setNote(int note) {
97          this.note = note;
98      }
99
100     // M thodes de classe pour obtenir et modifier 'nom'
101
102     /**
103      * Obtient le nom de la m diath que.
104      * @return Le nom de la m diath que.
105      */
106     public static String getNom() {
107         return nom;
108     }
109
110     /**
111      * D finit le nom de la m diath que.
112      * @param nom Le nom    d finir.
113      */
114     public static void setNom(String nom) {
115         Media.nom = nom;
116     }
117
118     // M thodes red finies
119
120     /**
121      * Retourne une repr sentation sous forme de cha ne du m dia.
```

```
122      * @return Une cha ne contenant le titre, la cote et la note.
123      */
124     @Override
125     public String toString() {
126         return "Media [titre=" + titre + ", cote=" + cote + ", note=" +
                note + "]";
127     }
128 
129     /**
130      * V rifie si ce m dia est  gal  un autre objet.
131      * @param obj L'objet  comparer.
132      * @return Vrai si  gal , faux sinon.
133      */
134     @Override
135     public boolean equals(Object obj) {
136         if (this == obj)
137             return true;
138         if (obj instanceof Media) {
139             Media other = (Media) obj;
140             return this.titre.equals(other.titre) && this.cote.toString
                    ().equals(other.cote.toString())
141                     && this.note == other.note;
142         }
143         return false;
144     }
145 
146     /**
147      * Cr e et retourne une copie de ce m dia.
148      * @return Un clone de ce m dia.
149      */
150     @Override
151     protected Object clone() {
152         return new Media(this);
153     }
154 }
```

Listing 1 – Classe `Media`

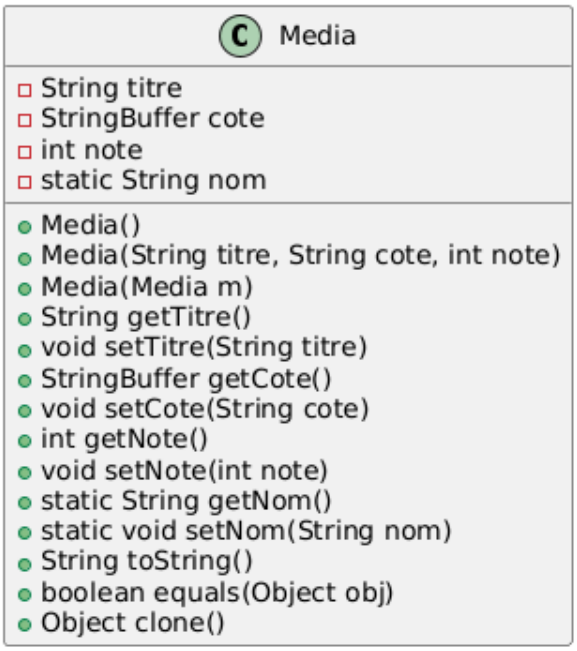

FIGURE 3 – Diagramme de classe de la classe `Media`.

Explication : Ce diagramme illustre la classe `Media` avec ses attributs privés, ses constructeurs, ses méthodes publiques, et les méthodes de classe pour gérer le nom de la médiathèque.

2.4 Question 3

Considérons le getter de l'objet `StringBuffer cote`. Si ce getter a été écrit comme suit :

```
1 public StringBuffer getCote() {
2     return this.cote;
3 }
```

Que donnera le code suivant :

```
1 Media m1 = new Media("Kill Bill vol 1","2BXX5",5);
2 System.out.println(m1);
3 m1.getCote().reverse();
4 System.out.println(m1);
```

Comment écrire le getter afin d'éviter cela ?

Réponse

Avec le getter tel qu'il est écrit, le code ci-dessus donnera :

```
Media [titre=Kill Bill vol 1, cote=2BXX5, note=5]
Media [titre=Kill Bill vol 1, cote=5XXB2, note=5]
```

Cela se produit parce que `StringBuffer` est mutable et que le getter retourne une référence directe à l'objet interne `cote`. En appelant `reverse()`, nous modifions l'état interne de `cote` dans l'objet `Media`.

Pour éviter cela, nous devons retourner une copie de `cote` dans le getter :

```java
public StringBuffer getCote() {
    return new StringBuffer(this.cote);
}
```

Ainsi, toute modification sur l'objet retourné par le getter n'affectera pas l'état interne de `cote` dans l'objet `Media`.

2.5 Question 4

Donnez un programme d'exemple de la classe `Livre`.

Réponse

Voici un exemple de programme utilisant la classe `Livre` :

```java
/*
TP 1: Retour sur Java
Universit  de Reims Champagne Ardennes
Despoullains Romain
*/

public class TestLivre {
    public static void main(String[] args) {
        Livre livre1 = new Livre("Le Petit Prince", "L001", 5, "Antoine
            de Saint-Exup ry", "978-0156013987");
        System.out.println(livre1);

        // Modification de la note
        livre1.setNote(4);
        System.out.println("Apr s modification de la note : " + livre1.
            getNote());
    }
}
```

Listing 2 – Programme `TestLivre`

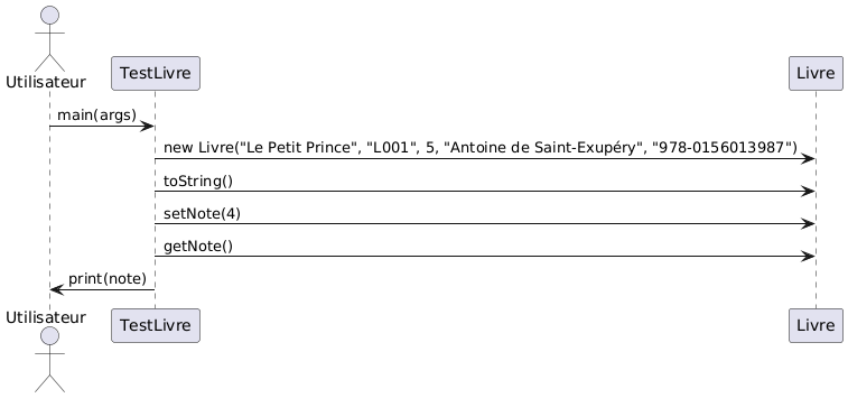

FIGURE 4 – Diagramme de séquence pour `TestLivre.java`.

Explication : Ce diagramme de séquence montre les interactions entre l'utilisateur, le programme de test `TestLivre`, et la classe `Livre` lors de la création et la modification d'un objet `Livre`.

2.6 Question 5

Donnez toutes les déclarations possibles avec les classes `Media` et `Livre`.

Réponse

Voici quelques déclarations possibles :
— `Media m1 = new Media();`
— `Media m2 = new Media("Titre", "Cote", 5);`
— `Livre l1 = new Livre();`
— `Livre l2 = new Livre("Titre", "Cote", 5, "Auteur", "ISBN");`
— `Media m3 = new Livre("Titre", "Cote", 5, "Auteur", "ISBN");`
— `Livre l3 = (Livre) m3;` (après vérification avec `instanceof`)

2.7 Question 6

Rappelez les règles de résolution des méthodes en Java.

Réponse

En Java, les règles de résolution des méthodes sont les suivantes :
— **Liage dynamique** : La méthode appelée est celle de l'objet réel, même si la référence est de type parent.
— **Redéfinition** : Une classe fille peut redéfinir une méthode de sa classe mère avec la même signature.
— **Surcharge** : Plusieurs méthodes peuvent avoir le même nom mais avec des paramètres différents dans la même classe.
— **Visibilité** : Les méthodes privées ne sont pas héritées et ne peuvent pas être redéfinies.
— **Méthodes statiques** : Elles sont liées à la classe et non à l'instance, la résolution se fait à la compilation.

2.8 Question 7

Parmi les instructions suivantes, lesquelles sont valides.

1. `Media m1 = new Livre("Le livre de la jungle","D...",5,"R. K..","208..");`
2. `System.out.println(m1);`
3. `m1.setCote("SD77DS");`
4. `System.out.println(m1.getAuteur());`
5. `System.out.println(((Livre)m1).getAuteur());`

Réponse

1. **Valide** : Création d'un objet `Livre` avec une référence de type `Media`.
2. **Valide** : Appel de `toString()` sur `m1`.
3. **Valide** : `setCote()` est une méthode de `Media`.
4. **Invalide** : `getAuteur()` n'est pas une méthode de `Media`.
5. **Valide** : Après transtypage (*casting*) en `Livre`, on peut appeler `getAuteur()`.

2.9 Question 8

La classe `Film` hérite de la classe `Media`.

Rappel : les variables `String titre` et `StringBuffer cote` ne seront plus `private` mais `protected`.

La classe `Film` hérite donc de la classe `Media`, et l'augmente de :

- **Données d'instance (privées)** :
  - `realisateur` : le réalisateur (`String`)
  - `annee` : l'année (`int`)
- **Méthodes** :
  - Trois constructeurs : par défaut, par initialisation et par copie
  - Les *getters* et *setters* associés aux données d'instance
  - Le masquage des méthodes `clone`, `equals` et `toString`

Écrivez la classe `Film`.

Réponse

Voici l'implémentation de la classe `Film` :

```
/*
TP 1: Retour sur Java
Universit  de Reims Champagne Ardennes
Despoullains Romain
*/

public class Film extends Media {

    // Variables d'instance priv es
    private String realisateur;   // R alisateur du film
    private int annee;            // Ann e de sortie du film

    /**
     * Constructeur par d faut.
     */
    public Film() {
        super();
        this.realisateur = "";
        this.annee = 0;
    }

    /**
     * Constructeur par initialisation.
     * @param titre Le titre du film.
     * @param cote La cote du film.
     * @param note La note attribu e au film.
     * @param realisateur Le r alisateur du film.
     * @param annee L'ann e de sortie du film.
     */
    public Film(String titre, String cote, int note, String realisateur,
        int annee) {
        super(titre, cote, note);
        this.realisateur = realisateur;
        this.annee = annee;
    }

    /**
```

```
37      * Constructeur par copie.
38      * @param f L'objet Film    copier.
39      */
40     public Film(Film f) {
41         super(f);
42         this.realisateur = f.realisateur;
43         this.annee = f.annee;
44     }
45 
46     // Getters et setters
47 
48     /**
49      * Obtient le r alisateur du film.
50      * @return Le r alisateur.
51      */
52     public String getRealisateur() {
53         return realisateur;
54     }
55 
56     /**
57      * D finit le r alisateur du film.
58      * @param realisateur Le r alisateur    d finir.
59      */
60     public void setRealisateur(String realisateur) {
61         this.realisateur = realisateur;
62     }
63 
64     /**
65      * Obtient l'ann e de sortie du film.
66      * @return L'ann e de sortie.
67      */
68     public int getAnnee() {
69         return annee;
70     }
71 
72     /**
73      * D finit l'ann e de sortie du film.
74      * @param annee L'ann e    d finir.
75      */
76     public void setAnnee(int annee) {
77         this.annee = annee;
78     }
79 
80     // M thodes red finies
81 
82     /**
83      * Retourne une repr sentation sous forme de cha ne du film.
84      * @return Une cha ne contenant les informations du film.
85      */
86     @Override
87     public String toString() {
88         return "Film [titre=" + getTitre() + ", cote=" + getCote() + ",
               note=" + getNote() +
89                 ", realisateur=" + realisateur + ", annee=" + annee + "]"
                   ;
90     }
91 
92     /**
```

```java
     * V rifie si ce film est  gal    un autre objet.
     * @param obj L'objet    comparer.
     * @return Vrai si  gal , faux sinon.
     */
    @Override
    public boolean equals(Object obj) {
        if (!super.equals(obj))
            return false;
        if (obj instanceof Film) {
            Film other = (Film) obj;
            return this.realisateur.equals(other.realisateur) &&
                   this.annee == other.annee;
        }
        return false;
    }

    /**
     * Cr e et retourne une copie de ce film.
     * @return Un clone de ce film.
     */
    @Override
    protected Object clone() {
        return new Film(this);
    }
}
```

Listing 3 – Classe `Film`

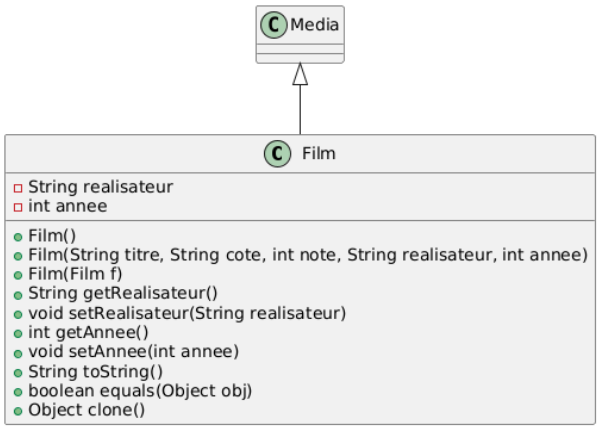

FIGURE 5 – Diagramme de classe de la classe `Film`.

Explication : Ce diagramme illustre que `Film` hérite de `Media` et ajoute des attributs spécifiques tels que `realisateur` et `annee`, ainsi que leurs méthodes d'accès et de modification.

2.10 Question 9

Écrivez un programme d'exemple exploitant la classe `Film`.

Réponse

Voici un exemple de programme utilisant la classe `Film` :

```java
/*
TP 1: Retour sur Java
Universit  de Reims Champagne Ardennes
Despoullains Romain
*/

public class TestFilm {
    public static void main(String[] args) {
        Film film1 = new Film("Inception", "F001", 5, "Christopher Nolan
            ", 2010);
        System.out.println(film1);

        // Modification de l'ann e
        film1.setAnnee(2011);
        System.out.println("Apr s modification de l'ann e : " + film1.
            getAnnee());
    }
}
```

Listing 4 – Programme `TestFilm`

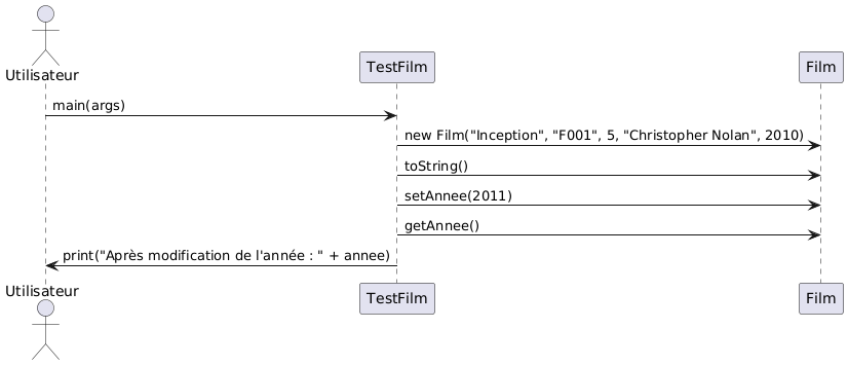

FIGURE 6 – Diagramme de séquence pour `TestFilm.java`.

Explication : Ce diagramme de séquence illustre les interactions entre l'utilisateur, le programme de test `TestFilm`, et la classe `Film` lors de la création et la modification d'un objet `Film`.

2.11 Question 10

Une médiathèque est un ensemble de médias (`Livre` ou `Film`).
La classe `Mediatheque` est constituée de :

- **Données** :
  - `proprietaire` : le nom du propriétaire (`String`)
  - `medias` : un objet `Vector<Media>`
- **Méthodes** :
  - Deux constructeurs : par défaut et par copie
  - La méthode `void add(Media)`
  - La méthode `toString`

Écrivez la classe `Mediatheque`.

Réponse

Voici l'implémentation de la classe `Mediatheque` :

```
/*
TP 1: Retour sur Java
Universit  de Reims Champagne Ardennes
Despoullains Romain
*/

import java.util.Vector;

public class Mediatheque {

    // Variables d'instance priv es
    private String proprietaire;        // Nom du propri taire
    private Vector<Media> medias;       // Collection de m dias

    /**
     * Constructeur par d faut.
     */
    public Mediatheque() {
        this.proprietaire = "";
        this.medias = new Vector<Media>();
    }

    /**
     * Constructeur par copie.
     * @param m La m diath que    copier.
     */
    public Mediatheque(Mediatheque m) {
        this.proprietaire = m.proprietaire;
        this.medias = new Vector<Media>(m.medias);
    }

    // Getters et setters
    public String getProprietaire() {
        return proprietaire;
    }

    public void setProprietaire(String proprietaire) {
        this.proprietaire = proprietaire;
    }

    // M thode pour ajouter un m dia
    public void add(Media media) {
        medias.add(media);
    }

    // Red finition de toString
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("M diath que de ").append(proprietaire).append(" :\n
            ");
        for (Media media : medias) {
            sb.append(media.toString()).append("\n");
        }
        return sb.toString();
```

```
55      }
56  }
```

Listing 5 – Classe `Mediatheque`

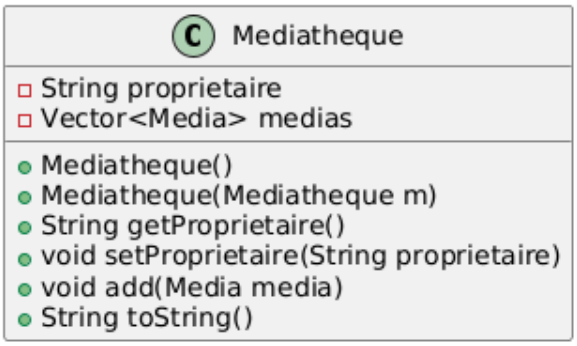

FIGURE 7 – Diagramme de classe de la classe `Mediatheque`.

Explication : Ce diagramme représente la classe `Mediatheque` qui contient un propriétaire et une collection de médias (`Vector<Media>`). Il inclut des méthodes pour gérer ces attributs et ajouter des médias à la collection.

2.12 Question 11

Donnez un exemple d'utilisation de la classe `Mediatheque`.

Réponse

Voici un exemple de programme utilisant la classe `Mediatheque` :

```
1  /*
2  TP 1: Retour sur Java
3  Universit  de Reims Champagne Ardennes
4  Despoullains Romain
5  */
6
7  public class TestMediatheque {
8      public static void main(String[] args) {
9          Mediatheque mediatheque = new Mediatheque();
10         mediatheque.setProprietaire("Jean Dupont");
11
12         Livre livre = new Livre("1984", "L002", 4, "George Orwell", "
               978-0451524935");
13         Film film = new Film("The Matrix", "F002", 5, "The Wachowskis",
               1999);
14
15         mediatheque.add(livre);
16         mediatheque.add(film);
17
18         System.out.println(mediatheque);
19     }
20 }
```

Listing 6 – Programme `TestMediatheque`

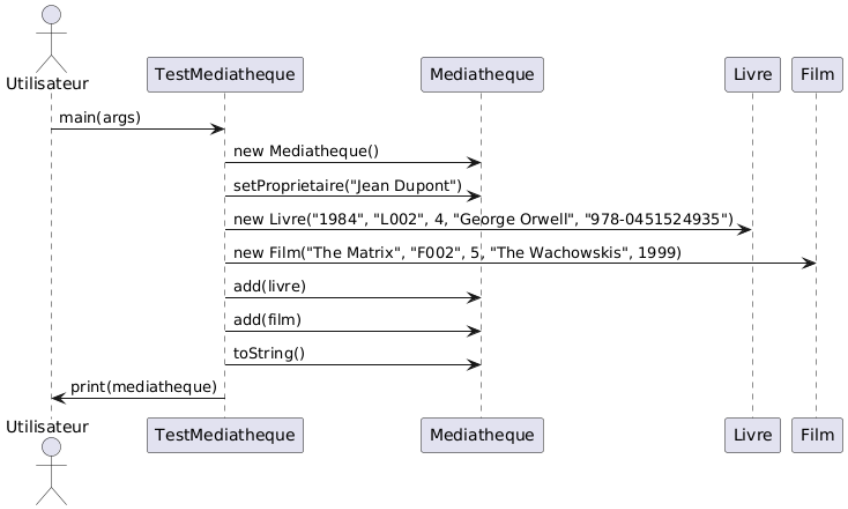

FIGURE 8 – Diagramme de séquence pour `TestMediatheque.java`.

Explication : Ce diagramme de séquence représente les interactions entre l'utilisateur, le programme de test `TestMediatheque`, et la classe `Mediatheque` lors de l'ajout de médias et l'affichage de la médiathèque.

3 Conclusion

Ce TP était simple et m'a permis de revoir les bases de Java. J'ai pu utiliser les concepts de la programmation orientée objet tels que l'héritage et l'encapsulation pour structurer efficacement mon projet. Et j'ai hâte des prochains TP.

4 Annexes

4.1 Code Complet des Classes

Classe `Media`

```java
/*
TP 1: Retour sur Java
Universit  de Reims Champagne Ardennes
Despoullains Romain
*/

package TP1;

public class Media {

    // Variable de classe (priv e)
    private static String nom;

    // Variables d'instance (priv es)
    private String titre;          // Titre du m dia
    private StringBuffer cote;     // Cote du m dia
    private int note;              // Note attribu e au m dia

    /**
     * Constructeur par d faut.
     * Initialise les variables d'instance avec des valeurs par d faut.
     */
    public Media() {
        this.titre = "";
        this.cote = new StringBuffer();
        this.note = 0;
    }

    /**
     * Constructeur par initialisation.
     * @param titre Le titre du m dia.
     * @param cote La cote du m dia.
     * @param note La note attribu e au m dia.
     */
    public Media(String titre, String cote, int note) {
        this.titre = titre;
        this.cote = new StringBuffer(cote);
        this.note = note;
    }

    /**
     * Constructeur par copie.
     * @param m L'objet Media    copier.
     */
    public Media(Media m) {
        this.titre = m.titre;
        this.cote = new StringBuffer(m.cote);
        this.note = m.note;
    }

    // Getters et setters pour les variables d'instance

```

```
53      /**
54       * Obtient le titre du m dia.
55       * @return Le titre.
56       */
57      public String getTitre() {
58          return titre;
59      }
60
61      /**
62       * D finit le titre du m dia.
63       * @param titre Le titre    d finir.
64       */
65      public void setTitre(String titre) {
66          this.titre = titre;
67      }
68
69      /**
70       * Obtient une copie de la cote du m dia.
71       * Retourne un nouveau StringBuffer pour  viter  la modification
             externe.
72       * @return Une copie de la cote.
73       */
74      public StringBuffer getCote() {
75          return new StringBuffer(this.cote);
76      }
77
78      /**
79       * D finit la cote du m dia.
80       * @param cote La cote    d finir.
81       */
82      public void setCote(String cote) {
83          this.cote = new StringBuffer(cote);
84      }
85
86      /**
87       * Obtient la note du m dia.
88       * @return La note.
89       */
90      public int getNote() {
91          return note;
92      }
93
94      /**
95       * D finit la note du m dia.
96       * @param note La note    d finir.
97       */
98      public void setNote(int note) {
99          this.note = note;
100     }
101
102     // M thodes de classe pour obtenir et modifier 'nom'
103
104     /**
105      * Obtient le nom de la m diath que.
106      * @return Le nom de la m diath que.
107      */
108     public static String getNom() {
109         return nom;
```

```
110     }
111
112     /**
113      * D finit le nom de la m diath que.
114      * @param nom Le nom    d finir.
115      */
116     public static void setNom(String nom) {
117         Media.nom = nom;
118     }
119
120     // M thodes red finies
121
122     /**
123      * Retourne une repr sentation sous forme de cha ne du m dia.
124      * @return Une cha ne contenant le titre, la cote et la note.
125      */
126     @Override
127     public String toString() {
128         return "Media [titre=" + titre + ", cote=" + cote + ", note=" +
                 note + "]";
129     }
130
131     /**
132      * V rifie si ce m dia est  gal     un autre objet.
133      * @param obj L'objet    comparer.
134      * @return Vrai si  gal , faux sinon.
135      */
136     @Override
137     public boolean equals(Object obj) {
138         if (this == obj)
139             return true;
140         if (obj instanceof Media) {
141             Media other = (Media) obj;
142             return this.titre.equals(other.titre) && this.cote.toString
                    ().equals(other.cote.toString())
143                     && this.note == other.note;
144         }
145         return false;
146     }
147
148     /**
149      * Cr e et retourne une copie de ce m dia.
150      * @return Un clone de ce m dia.
151      */
152     @Override
153     protected Object clone() {
154         return new Media(this);
155     }
156 }
```

Listing 7 – Classe `Media`

Classe Livre

```
1 /*
2 TP 1: Retour sur Java
3 Universit  de Reims Champagne Ardennes
```

```java
Despoullains Romain
*/

package TP1;

public class Livre extends Media {

    // Variables d'instance priv es
    private String auteur;   // Auteur du livre
    private String isbn;     // ISBN du livre

    /**
     * Constructeur par d faut.
     */
    public Livre() {
        super();
        this.auteur = "";
        this.isbn = "";
    }

    /**
     * Constructeur par initialisation.
     * @param titre Le titre du livre.
     * @param cote La cote du livre.
     * @param note La note attribu e au livre.
     * @param auteur L'auteur du livre.
     * @param isbn Le num ro ISBN du livre.
     */
    public Livre(String titre, String cote, int note, String auteur,
        String isbn) {
        super(titre, cote, note);
        this.auteur = auteur;
        this.isbn = isbn;
    }

    /**
     * Constructeur par copie.
     * @param l L'objet Livre    copier.
     */
    public Livre(Livre l) {
        super(l);
        this.auteur = l.auteur;
        this.isbn = l.isbn;
    }

    // Getters et setters pour les variables d'instance

    /**
     * Obtient l'auteur du livre.
     * @return L'auteur.
     */
    public String getAuteur() {
        return auteur;
    }

    /**
     * D finit l'auteur du livre.
     * @param auteur L'auteur    d finir.
```

```
61          */
62         public void setAuteur(String auteur) {
63             this.auteur = auteur;
64         }
65 
66         /**
67          * Obtient le num ro ISBN du livre.
68          * @return Le num ro ISBN.
69          */
70         public String getIsbn() {
71             return isbn;
72         }
73 
74         /**
75          * D finit le num ro ISBN du livre.
76          * @param isbn Le num ro ISBN    d finir.
77          */
78         public void setIsbn(String isbn) {
79             this.isbn = isbn;
80         }
81 
82         // M thodes red finies
83 
84         /**
85          * Retourne une repr sentation sous forme de cha ne du livre.
86          * @return Une cha ne contenant les informations du livre.
87          */
88         @Override
89         public String toString() {
90             return "Livre [titre=" + getTitre() + ", cote=" + getCote() + ",
                   note=" + getNote() +
91                     ", auteur=" + auteur + ", isbn=" + isbn + "]";
92         }
93 
94         /**
95          * V rifie si ce livre est  gal     un autre objet.
96          * @param obj L'objet    comparer.
97          * @return Vrai si  gal , faux sinon.
98          */
99         @Override
100        public boolean equals(Object obj) {
101            if (!super.equals(obj))
102                return false;
103            if (obj instanceof Livre) {
104                Livre other = (Livre) obj;
105                return this.auteur.equals(other.auteur) && this.isbn.equals(
                       other.isbn);
106            }
107            return false;
108        }
109 
110        /**
111         * Cr e et retourne une copie de ce livre.
112         * @return Un clone de ce livre.
113         */
114        @Override
115        protected Object clone() {
116            return new Livre(this);
```

```
117     }
118 }
```

Listing 8 – Classe `Livre`

Classe `Film`

```
1  /*
2  TP 1: Retour sur Java
3  Universit  de Reims Champagne Ardennes
4  Despoullains Romain
5  */
6
7  package TP1;
8
9  public class Film extends Media {
10
11     // Variables d'instance priv es
12     private String realisateur;   // R alisateur du film
13     private int annee;            // Ann e de sortie du film
14
15     /**
16      * Constructeur par d faut.
17      */
18     public Film() {
19         super();
20         this.realisateur = "";
21         this.annee = 0;
22     }
23
24     /**
25      * Constructeur par initialisation.
26      * @param titre Le titre du film.
27      * @param cote La cote du film.
28      * @param note La note attribu e au film.
29      * @param realisateur Le r alisateur du film.
30      * @param annee L'ann e de sortie du film.
31      */
32     public Film(String titre, String cote, int note, String realisateur,
            int annee) {
33         super(titre, cote, note);
34         this.realisateur = realisateur;
35         this.annee = annee;
36     }
37
38     /**
39      * Constructeur par copie.
40      * @param f L'objet Film    copier.
41      */
42     public Film(Film f) {
43         super(f);
44         this.realisateur = f.realisateur;
45         this.annee = f.annee;
46     }
47
48     // Getters et setters pour les variables d'instance
49
```

```
50      /**
51       * Obtient le r alisateur du film.
52       * @return Le r alisateur.
53       */
54      public String getRealisateur() {
55          return realisateur;
56      }
57
58      /**
59       * D finit le r alisateur du film.
60       * @param realisateur Le r alisateur    d finir.
61       */
62      public void setRealisateur(String realisateur) {
63          this.realisateur = realisateur;
64      }
65
66      /**
67       * Obtient l'ann e de sortie du film.
68       * @return L'ann e de sortie.
69       */
70      public int getAnnee() {
71          return annee;
72      }
73
74      /**
75       * D finit l'ann e de sortie du film.
76       * @param annee L'ann e    d finir.
77       */
78      public void setAnnee(int annee) {
79          this.annee = annee;
80      }
81
82      // M thodes red finies
83
84      /**
85       * Retourne une repr sentation sous forme de cha ne du film.
86       * @return Une cha ne contenant les informations du film.
87       */
88      @Override
89      public String toString() {
90          return "Film [titre=" + getTitre() + ", cote=" + getCote() + ",
                note=" + getNote() +
91                  ", realisateur=" + realisateur + ", annee=" + annee + "]"
                    ;
92      }
93
94      /**
95       * V rifie si ce film est  gal    un autre objet.
96       * @param obj L'objet    comparer.
97       * @return Vrai si  gal , faux sinon.
98       */
99      @Override
100     public boolean equals(Object obj) {
101         if (!super.equals(obj))
102             return false;
103         if (obj instanceof Film) {
104             Film other = (Film) obj;
```

```
105             return this.realisateur.equals(other.realisateur) && this.
                    annee == other.annee;
106         }
107         return false;
108     }
109 
110     /**
111      * Cr e et retourne une copie de ce film.
112      * @return Un clone de ce film.
113      */
114     @Override
115     protected Object clone() {
116         return new Film(this);
117     }
118 }
```

Listing 9 – Classe `Film`

Classe `Mediatheque`

```
1  /*
2  TP 1: Retour sur Java
3  Universit  de Reims Champagne Ardennes
4  Despoullains Romain
5  */
6 
7  package TP1;
8 
9  import java.util.Vector;
10 
11 public class Mediatheque {
12 
13     // Variables d'instance priv es
14     private String proprietaire;        // Nom du propri taire
15     private Vector<Media> medias;       // Collection de m dias
16 
17     /**
18      * Constructeur par d faut.
19      */
20     public Mediatheque() {
21         this.proprietaire = "";
22         this.medias = new Vector<Media>();
23     }
24 
25     /**
26      * Constructeur par copie.
27      * @param m La m diath que    copier.
28      */
29     public Mediatheque(Mediatheque m) {
30         this.proprietaire = m.proprietaire;
31         this.medias = new Vector<Media>(m.medias);
32     }
33 
34     /**
35      * Obtient le nom du propri taire.
36      * @return Le nom du propri taire.
37      */
```

```
38      public String getProprietaire() {
39          return proprietaire;
40      }
41
42      /**
43       * D finit le nom du propri taire.
44       * @param proprietaire Le nom    d finir.
45       */
46      public void setProprietaire(String proprietaire) {
47          this.proprietaire = proprietaire;
48      }
49
50      /**
51       * Ajoute un m dia    la m diath que.
52       * @param media Le m dia    ajouter.
53       */
54      public void add(Media media) {
55          medias.add(media);
56      }
57
58      /**
59       * Retourne une repr sentation sous forme de cha ne de la
             m diath que.
60       * @return Une cha ne repr sentant la m diath que.
61       */
62      @Override
63      public String toString() {
64          StringBuilder sb = new StringBuilder();
65          sb.append("M diath que de ").append(proprietaire).append(" :\n
                ");
66          for (Media media : medias) {
67              sb.append(media.toString()).append("\n");
68          }
69          return sb.toString();
70      }
71  }
```

Listing 10 – Classe `Mediatheque`

Programmes de Test

```
1   /*
2   TP 1: Retour sur Java
3   Universit  de Reims Champagne Ardennes
4   Despoullains Romain
5   */
6
7   package TP1;
8
9   public class TestLivre {
10      public static void main(String[] args) {
11          Livre livre1 = new Livre("Le Petit Prince", "L001", 5, "Antoine
                de Saint-Exup ry", "978-0156013987");
12          System.out.println(livre1);
13
14          // Test des m thodes
15          livre1.setNote(4);
```

```
16          System.out.println("Apr s modification de la note : " + livre1.
                getNote());
17      }
18  }
```

Listing 11 – Programme `TestLivre`

```
1   /*
2   TP 1: Retour sur Java
3   Universit  de Reims Champagne Ardennes
4   Despoullains Romain
5   */
6
7   package TP1;
8
9   public class TestFilm {
10      public static void main(String[] args) {
11          Film film1 = new Film("Inception", "F001", 5, "Christopher Nolan
                ", 2010);
12          System.out.println(film1);
13
14          // Test des m thodes
15          film1.setAnnee(2011);
16          System.out.println("Apr s modification de l'ann e : " + film1.
                getAnnee());
17      }
18  }
```

Listing 12 – Programme `TestFilm`

```
1   /*
2   TP 1: Retour sur Java
3   Universit  de Reims Champagne Ardennes
4   Despoullains Romain
5   */
6
7   package TP1;
8
9   public class TestMediatheque {
10      public static void main(String[] args) {
11          Mediatheque mediatheque = new Mediatheque();
12          mediatheque.setProprietaire("Jean Dupont");
13
14          Livre livre = new Livre("1984", "L002", 4, "George Orwell", "
                978-0451524935");
15          Film film = new Film("The Matrix", "F002", 5, "The Wachowskis",
                1999);
16
17          mediatheque.add(livre);
18          mediatheque.add(film);
19
20          System.out.println(mediatheque);
21      }
22  }
```

Listing 13 – Programme `TestMediatheque`