



Projet AppServ JAVA

***Baptiste BERTRAND / Eymen H'MIDA / Ilyes HASSAINE /
Bilal MAHDJOUBI***

Groupe 207

Sommaire

Le Projet	3
Les différents Services	3
• Service réservation :	3
• Service emprunt :	3
• Service retour :	3
Diagramme d'Architecture	4
Les design-patterns utilisés	5
Pattern State	5
Pattern Singleton	6
Concurrence	7
Test	8
Conclusion	9

Le Projet

Ce projet a pour but de créer une application permettant de gérer plusieurs services pour des documents, comme des DVD par exemple. L'application est constamment active, on peut donc effectuer les services de l'application à n'importe quel moment.

Cette application est utilisable par tous, l'utilisateur est référencé par son nom et sa date de naissance. Un numéro unique est attribué automatiquement à cet utilisateur (ce qui représente sa clé primaire dans un SGBD).

La médiathèque possède une liste de documents référencé par son titre et un indicateur « adulte » permettant de décider si le document doit être réservé ou non par une personne de plus de 16 ans. Les documents ont eux aussi chacun un numéro unique.

Les différents Services

Chaque service est relié à un serveur possédant un numéro de port unique pour chacun. Pour chaque service le numéro d'abonné et celui du DVD sont demandés.

- **Service réservation :**

Ce service permet de réserver un document à distance pour ensuite pouvoir l'emprunter en toute sécurité. La réservation dure 2 heures, l'abonné a donc 2 heures pour aller emprunter le document. La réservation s'effectue sur un ordinateur à n'importe quel endroit. Ce service s'effectue sur le port 3000.

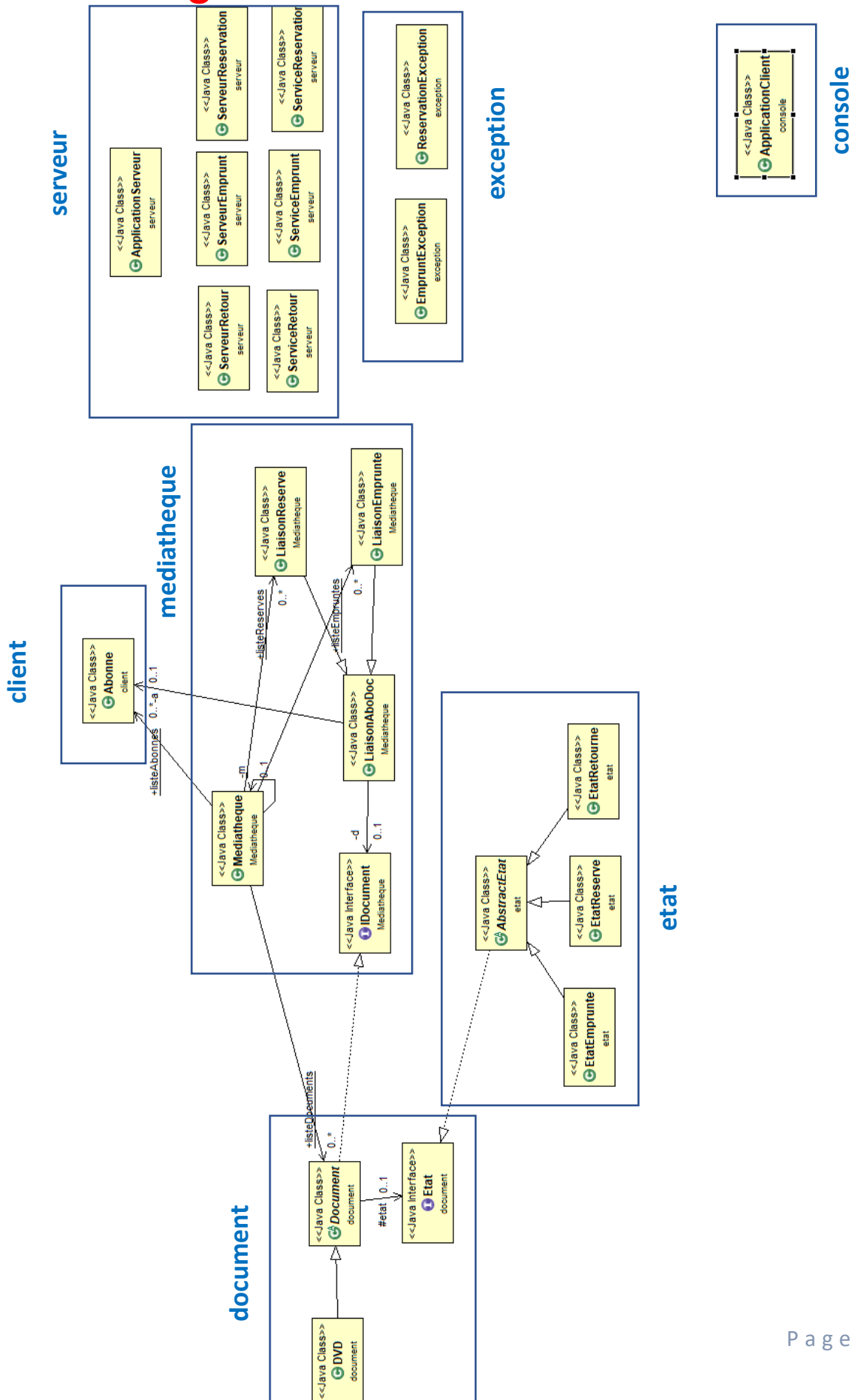
- **Service emprunt :**

Ce service permet d'emprunter un document pour le ramener chez soi pendant une certaine période. Pour emprunter un document, celui-ci doit-être réservé ou bien disponible. L'emprunt s'effectue sur place à la médiathèque. Ce service s'effectue sur le port 4000.

- **Service retour :**

Ce service permet de retourner un document emprunté. Le retour s'effectue sur place à la médiathèque. Ce service s'effectue sur le port 5000.

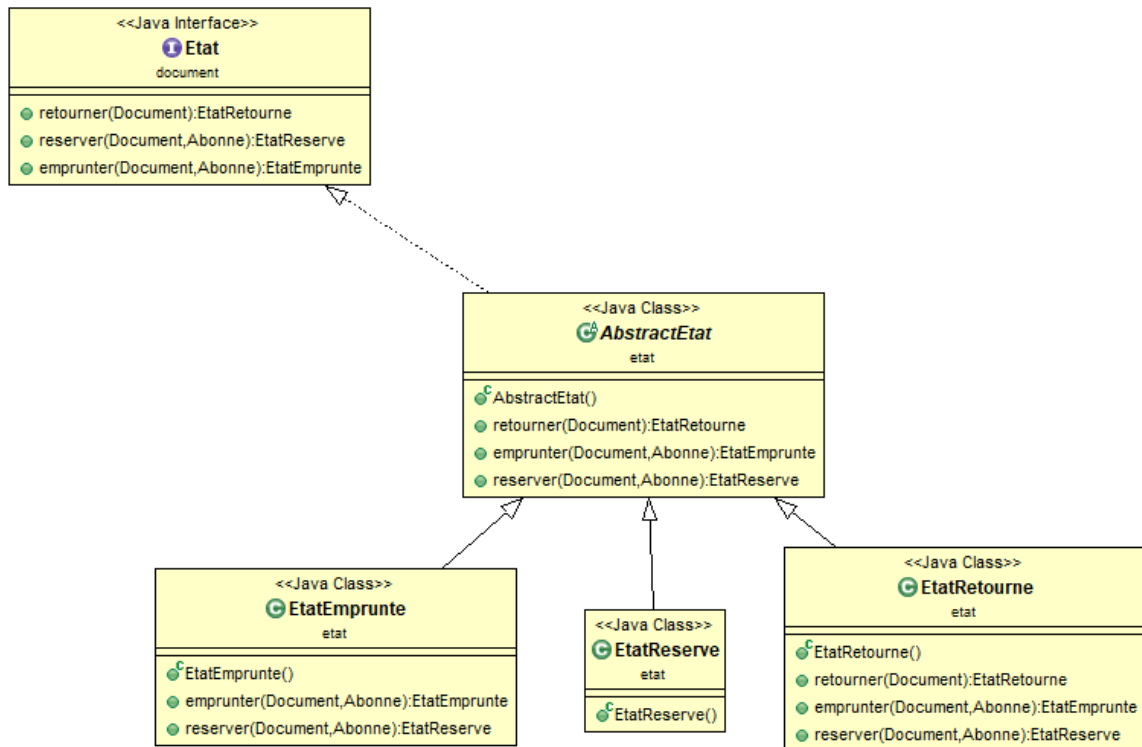
Diagramme d'Architecture



Les design-patterns utilisés

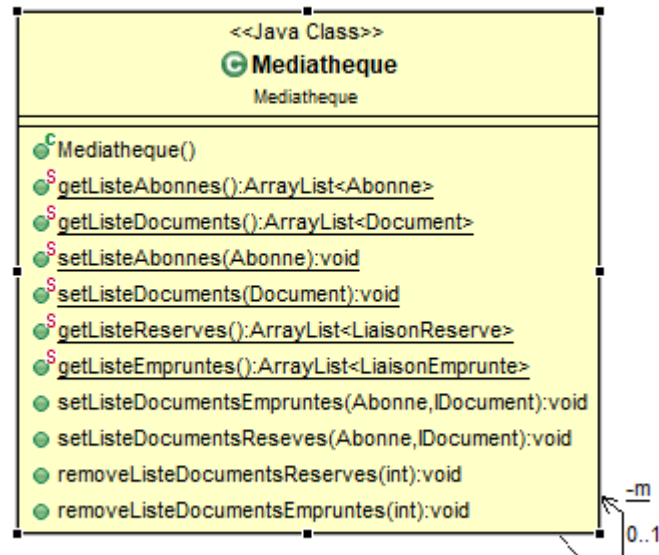
Pattern State

Package état :



Pattern Singleton

Package médiathèque :



```

public final class Mediatheque {
    private static Mediatheque m = new Mediatheque();
    public static ArrayList<Abonne> listeAbonnes = new ArrayList<>();
    public static ArrayList<Document> listeDocuments = new ArrayList<>();
    public static ArrayList<LiaisonReserve> listeReserves = new ArrayList<>();
    public static ArrayList<LiaisonEmprunte> listeEmpruntes = new ArrayList<>();
}
    
```

Concurrence

Chaque serveur (Retour, Reservation, Emprunt) sont lancés via des thread lors du lancement de l'application Serveur. De ce fait, plusieurs clients peuvent se connecter en même temps aux serveurs et les serveurs s'occuperont d'eux en même temps (principe de concurrence).

```
new Thread(new ServeurReservation(PORT_SERVEUR_RESERVATION)).start();  
new Thread(new ServeurEmprunt(PORT_SERVEUR_EMPRUNT)).start();  
new Thread(new ServeurRetour(PORT_SERVEUR_RETOUR)).start();
```

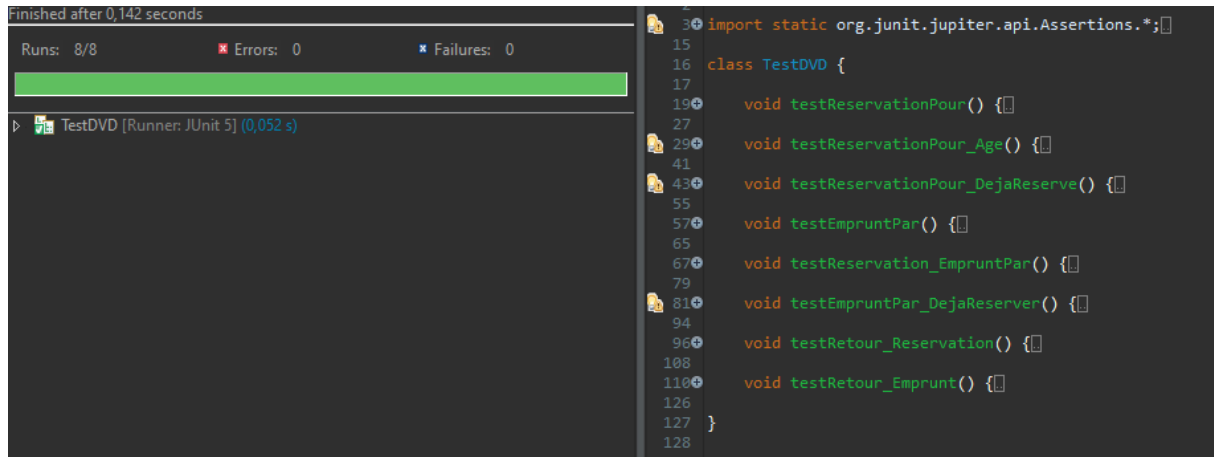
Cependant, afin d'éviter les erreurs de concurrence, le contenu de chaque méthode « Retour », « emprunterPar » et « reservationPour » est encadré par un synchronized.

Exemple :

```
@Override  
public void retour() {  
    synchronized(this) {  
        this.etat = etat.retourner(this);  
    }  
}
```

Test

Nous avons effectué une batterie de test JUnit, chacun des tests retourne un résultat positif. Ceci prouve la bonne fonctionnalité de notre application.

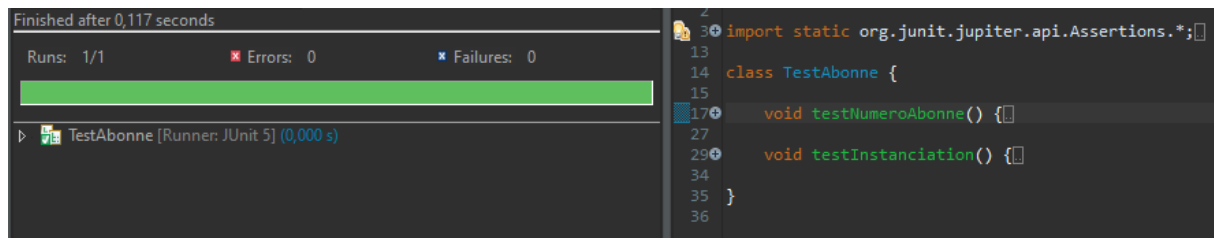


Finished after 0,142 seconds

Runs: 8/8 Errors: 0 Failures: 0

TestDVD [Runner: JUnit 5] (0,052 s)

```
30 import static org.junit.jupiter.api.Assertions.*;
15
16 class TestDVD {
17
19     void testReservationPour() {}
27
29     void testReservationPour_Age() {}
41
43     void testReservationPour_DejaReserve() {}
55
57     void testEmpruntPar() {}
65
67     void testReservation_EmpruntPar() {}
79
81     void testEmpruntPar_DejaReserver() {}
94
96     void testRetour_Reservation() {}
108
110     void testRetour_Emprunt() {}
126
127 }
128
```



Finished after 0,117 seconds

Runs: 1/1 Errors: 0 Failures: 0

TestAbonne [Runner: JUnit 5] (0,000 s)

```
30 import static org.junit.jupiter.api.Assertions.*;
13
14 class TestAbonne {
15
17     void testNumeroAbonne() {}
27
29     void testInstanciation() {}
34
35 }
36
```


Conclusion

Nous avons réussi à créer une application fonctionnelle permettant de pouvoir gérer les réservations et emprunts d'une médiathèque grâce à des serveurs. Nous sommes fiers d'être arrivé à l'aboutissement du projet et d'avoir produit une application complète. Ce projet nous a permis de mieux comprendre et mieux maîtriser la programmation concurrentielle.

Chaque membre du groupe a participé à la production de l'application, les membres ayant plus de connaissances et de maîtrise ont pu expliquer les problèmes que certains ont rencontrés. Le travail du groupe a été bénéfique pour chacun d'entre nous.

Dû au peu de temps que nous avons pour ce projet nous n'avons pas pu réaliser les certifications BretteSoft. Nous pensons qu'avec plus de temps disponible, nous aurions pu réaliser ces certifications.

Malgré les conditions difficiles ces temps-ci, nous nous sommes bien organisés pour nous répartir les tâches. Nous avons pu communiquer et échanger du code grâce à Discord et Google Drive notamment.