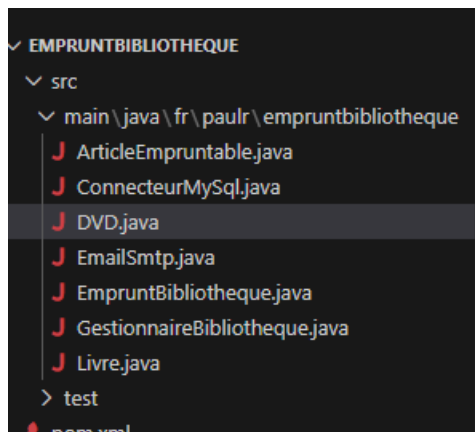


1. Importation du projet .....	1
2. Les Problème et les solutions .....	1
S – Single Responsibility Principle - SRP (Responsabilité unique) .....	2
O – Open/Closed Principle - OCP (Ouvert/Fermé) .....	3
L – Liskov Substitution Principle - LSP (Substitution de Liskov) .....	4
I – Interface Segregation Principle – ISP - (Séparation des interfaces) .....	4
D – Dependency Inversion Principle -DIP (Inversion des dépendances) .....	5

## 1. Importation du projet

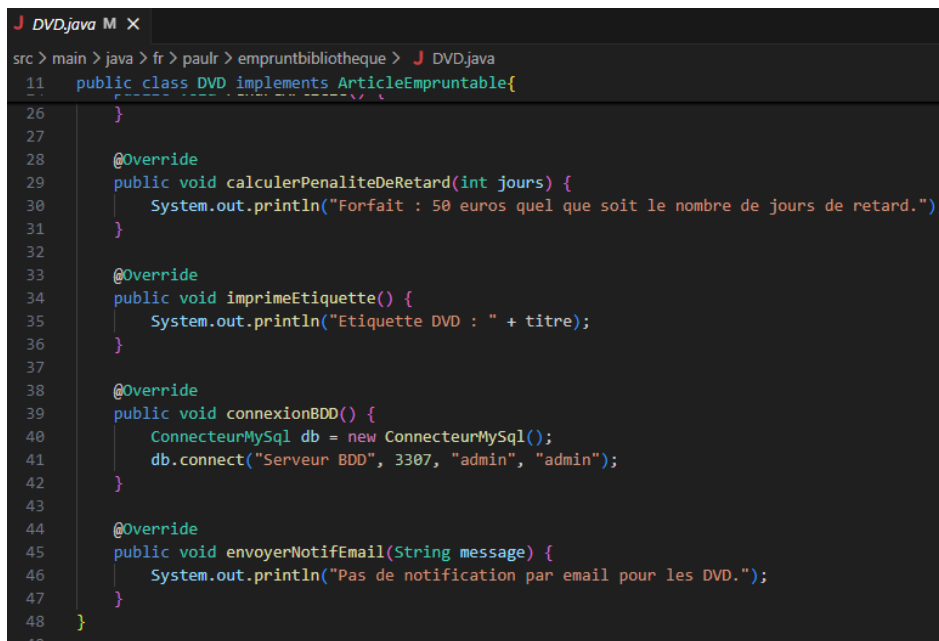
Le projet a bien été Importé.



## 2. Les Problème et les solutions

### S – Single Responsibility Principle - SRP (Responsabilité unique)

#### S.1



```
11 public class DVD implements ArticleEmprutable{
26 }
27
28 @Override
29 public void calculerPenaliteDeRetard(int jours) {
30     System.out.println("Forfait : 50 euros quel que soit le nombre de jours de retard.");
31 }
32
33 @Override
34 public void imprimeEtiquette() {
35     System.out.println("Etiquette DVD : " + titre);
36 }
37
38 @Override
39 public void connexionBDD() {
40     ConnecteurMySQL db = new ConnecteurMySQL();
41     db.connect("Serveur BDD", 3307, "admin", "admin");
42 }
43
44 @Override
45 public void envoyerNotifEmail(String message) {
46     System.out.println("Pas de notification par email pour les DVD.");
47 }
48 }
```

Problème :

Ces classes ne conviennent pas à la gestion des livres.

Solution :

De nouveaux fichiers avec des classes et des interfaces dédiés doivent être créés.

**S.2**

```
src > main > java > fr > paulr > empruntbibliotheque > Livre.java
11 public class Livre implements ArticleEmpruntable{
31     // LSP violé : comportement incohérent
32     @Override
33     public void calculerPenaliteDeRetard(int jours) {
34         if (jours < 0) {
35             throw new IllegalArgumentException("Le nombre de jours ne peut pas être négatif !");
36         }
37         System.out.println("Pénalités de retard : " + (jours * 10) + " euros");
38     }
39
40     @Override
41     public void imprimeEtiquette() {
42         System.out.println("Etiquette : " + titre);
43     }
44
45     // DIP violé : dépendance directe à une classe concrète
46     @Override
47     public void connexionBDD() {
48         ConnecteurMySQL db = new ConnecteurMySQL();
49         db.connect("localhost", 3306, "root", "root");
50     }
51
52     @Override
53     public void envoyerNotifEmail(String message) {
54         EmailSmtip sender = new EmailSmtip();
55         sender.send("user@example.com", "Message de votre bibliothèque", message);
56     }
57 }
58
```

Problème :

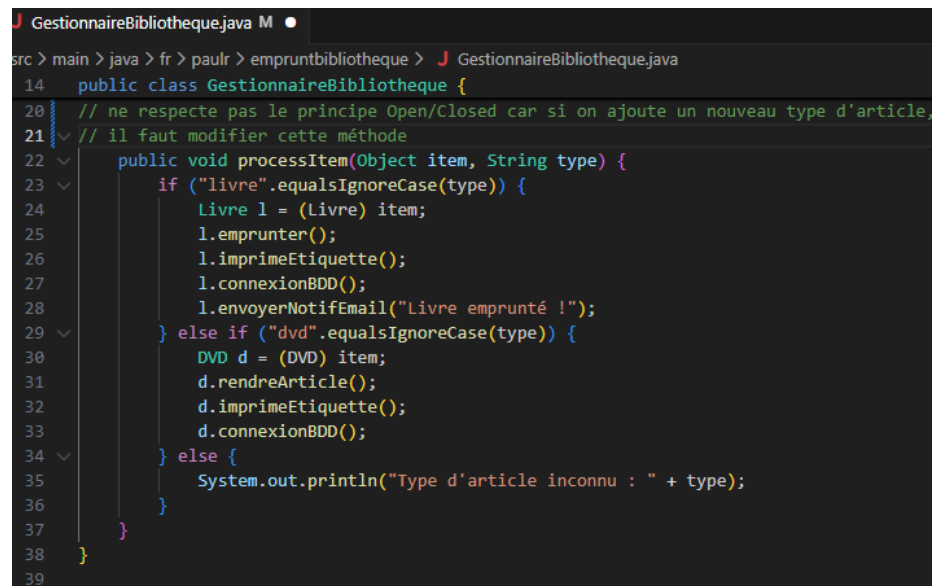
Ces classes ne conviennent pas à la gestion des DVD en plus de contrevenir à d'autres principes SOLID.

Solution :

De nouveaux fichiers avec des classes et des interfaces dédiés doivent être créés.

## O – Open/Closed Principle - OCP (Ouvert/Fermé)

### O.1



```
src > main > java > fr > paulr > empruntbibliotheque > J GestionnaireBibliotheque.java
14 public class GestionnaireBibliotheque {
20 // ne respecte pas le principe Open/Closed car si on ajoute un nouveau type d'article,
21 // il faut modifier cette méthode
22 public void processItem(Object item, String type) {
23     if ("livre".equalsIgnoreCase(type)) {
24         Livre l = (Livre) item;
25         l.emprunter();
26         l.imprimeEtiquette();
27         l.connexionBDD();
28         l.envoyerNotifEmail("Livre emprunté !");
29     } else if ("dvd".equalsIgnoreCase(type)) {
30         DVD d = (DVD) item;
31         d.rendreArticle();
32         d.imprimeEtiquette();
33         d.connexionBDD();
34     } else {
35         System.out.println("Type d'article inconnu : " + type);
36     }
37 }
38 }
39
```

Problème :

Afin de rajouter un nouveau type d'article il faudrait modifier ce code

Solution :

Création d'un nouveau fichier avec une interface objEmprunté.

## L – Liskov Substitution Principle - LSP (Substitution de Liskov)

### L.1



```
J Livre.java x
src > main > java > fr > paulr > empruntbibliotheque > J Livre.java
11 public class Livre implements ArticleEmpruntable{
12
31 // LSP violé : comportement incohérent
32 @Override
33 public void calculerPenaliteDeRetard(int jours) {
34     if (jours < 0) {
35         throw new IllegalArgumentException("Le nombre de jours ne peut pas être négatif !");
36     }
37     System.out.println("Pénalités de retard : " + (jours * 10) + " euros");
38 }
39
40 @Override
41 public void imprimeEtiquette() {
42     System.out.println("Etiquette : " + titre);
43 }
```

Problème :

La classe fille modifie le comportement de la classe mère en lançant une exception pour un cas que la classe mère acceptait.

Solution :

Rectifier le comportement en retirant l'exception ou en définir la règle dans l'abstraction pour que toutes les classes dérivées respectent les règles.

## I – Interface Segregation Principle – ISP - (Séparation des interfaces)

### I.1

```
J ArticleEmprutable.java M X
src > main > java > fr > paulr > empruntbibliotheque > J ArticleEmprutable.java
7  /**
11 | // ne respecte pas le ISP car cette interface n'est pas spécialisée.
12 | public interface ArticleEmprutable {
13 |
14 |     void emprunter();
15 |
16 |     void rendreArticle();
17 |
18 |     void calculerPenaliteDeRetard(int jours);
19 |
20 |     void imprimeEtiquette();
21 |
22 |     void connexionBDD();
23 |
24 |     void envoyerNotifEmail(String message);
25 | }
```

Problème :

L'interface n'est pas spécifique, elle est bien trop générale.

Solution :

Il faut diviser l'interface en plusieurs petites interfaces spécifique.

```
J ArticleEmprutable.java M X
src > main > java > fr > paulr > empruntbibliotheque > J ArticleEmprutable.java
27 | // Respecte le ISP en divisant l'interface en plusieurs interfaces spécifiques.
28 | public interface EmprunterArticle {
29 |     void emprunter();
30 | }
31 | public interface RendreArticle {
32 |     void rendreArticle();
33 | }
34 | public interface CalculerPenalite {
35 |     void calculerPenaliteDeRetard(int jours);
36 | }
37 | public interface ImprimerEtiquette {
38 |     void imprimeEtiquette();
39 | }
40 | public interface ConnexionBDD {
41 |     void connexionBDD();
42 | }
43 | public interface EnvoyerNotifEmail {
44 |     void envoyerNotifEmail(String message);
45 | }
```

## D – Dependency Inversion Principle -DIP (Inversion des dépendances)

### D.1

```
J DVD.java X
src > main > java > fr > paulr > empruntbibliotheque > J DVD.java
50 // Classes concrètes (DIP violé)
51 class MySQLDatabase {
52     public void connect(String host, int port, String user, String pass) {
53         System.out.println("Connexion MySQL à " + host + ":" + port);
54     }
55 }
56
57 class SmtplibEmailSender {
58     public void send(String to, String subject, String body) {
59         System.out.println("Email envoyé à " + to + ": " + subject);
60     }
61 }
62
```

Problème :

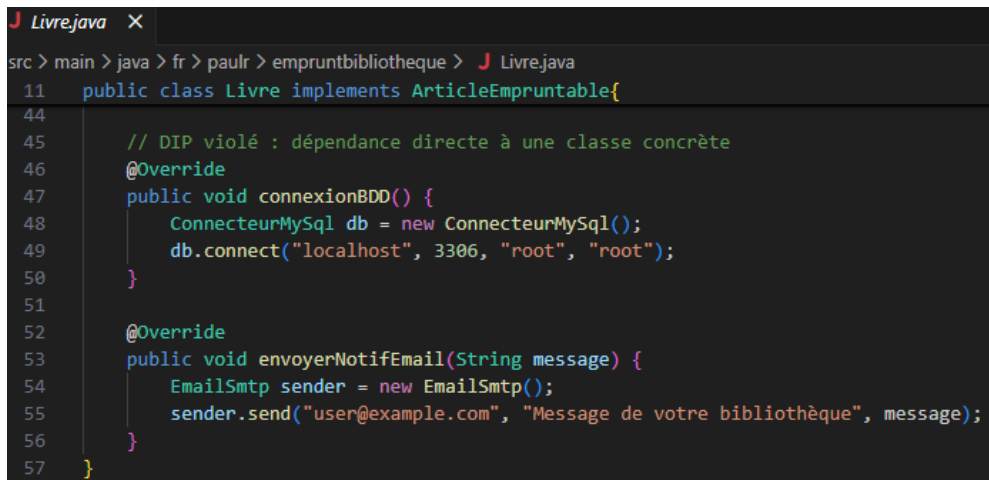
Ces classes n'ont pas leurs propres fichiers, elles sont écrites sur un fichier de bas niveau qui peut disparaître (si on propose plus de DVD).

Solution :

Il faut que ces classes aient leur propre fichier.

```
J MySQLDatabase U X J SmtplibEmailSender U
src > main > java > fr > paulr > empruntbibliotheque > J MySQLDatabase
1 public class MySQLDatabase {
2     public void connect(String host, int port, String user, String pass) {
3         System.out.println("Connexion MySQL à " + host + ":" + port);
4     }
5 }

J MySQLDatabase U J SmtplibEmailSender U X
src > main > java > fr > paulr > empruntbibliotheque > J SmtplibEmailSender
1 public class SmtplibEmailSender {
2     public void send(String to, String subject, String body) {
3         System.out.println("Email envoyé à " + to + ": " + subject);
4     }
5 }
```



```
J Livre.java X
src > main > java > fr > paulr > empruntbibliotheque > J Livre.java
11 public class Livre implements ArticleEmprutable{
44
45     // DIP violé : dépendance directe à une classe concrète
46     @Override
47     public void connexionBDD() {
48         ConnecteurMySQL db = new ConnecteurMySQL();
49         db.connect("localhost", 3306, "root", "root");
50     }
51
52     @Override
53     public void envoyerNotifEmail(String message) {
54         EmailSmt sender = new EmailSmt();
55         sender.send("user@example.com", "Message de votre bibliothèque", message);
56     }
57 }
```

Problème :

Solution :