

Lycée Technique Ibn Sina

Kénitra

Brevet de Technicien Supérieur

Développement des systèmes d'information



Mini Projet

Sujet :

``Gestion de bibliothèque``

Réalisé par :

- AIT ABDIELMOMIN HAJAR

Professeur :

- Mr. ALHIANE Hamid

I. Objectif du projet :

Ce projet a pour objectif de conception d'une application **java** permettant d'aider à la gestion de la bibliothèque et plus précisément, aider à la gestion des locataires et à la gestion des livres, en utilisant des **AGL** que nous étudié.

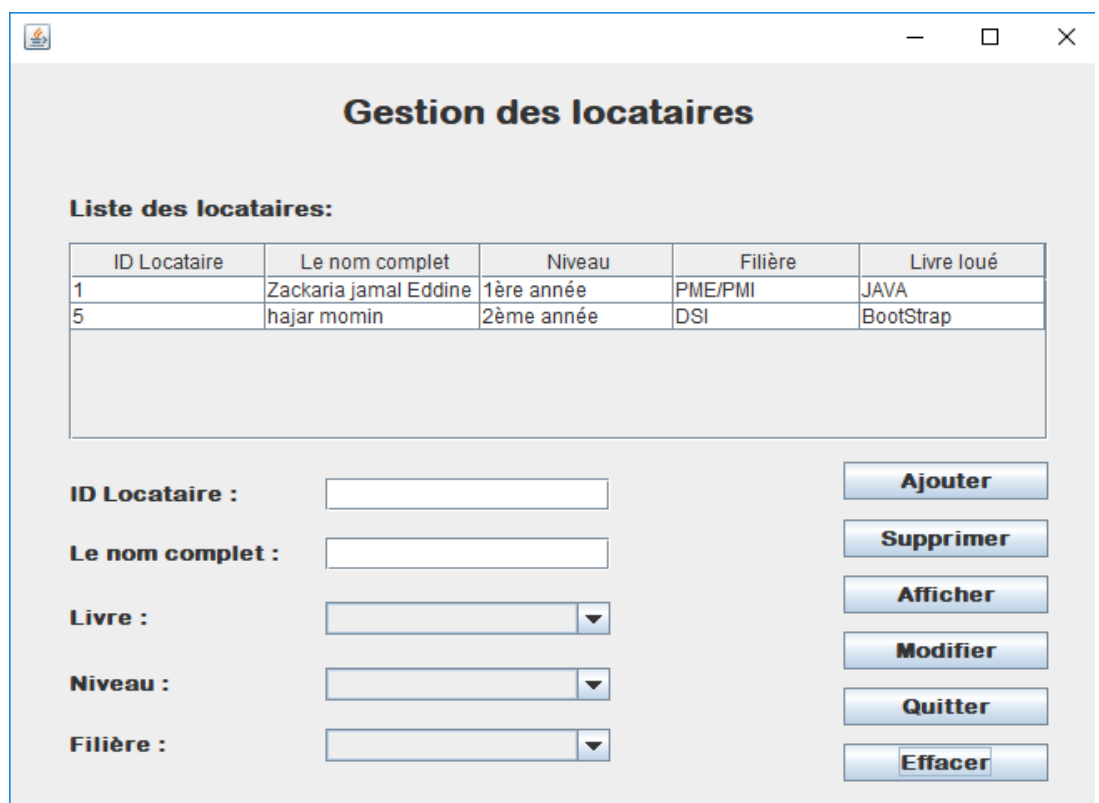
➤ Voici les interfaces de l'application :



The screenshot shows a window titled "Gestion des livres". It contains a table with the following data:

ID Livre	Titre	Genre	Auteur	Quantité
1	SQLServer	DB	Bouaabid	30
2	JAVA	Programmation	Bouaabid	23
4	BootStrap	Design	EL Hiaine	20

Below the table, there are input fields for "Numéro de livre :", "Titre :", "Genre :", "L'auteur :", and "Quantité :". The "Quantité" field has a value of "1" and a small up/down arrow. To the right of these fields are buttons: "Ajouter", "Supprimer", "Modifier", "Afficher", "Quitter", and "Effacer". At the bottom left, there is a button labeled "Gestion des locataires".



The screenshot shows a window titled "Gestion des locataires". It contains a table with the following data:

ID Locataire	Le nom complet	Niveau	Filière	Livre loué
1	Zackaria jamal Eddine	1ère année	PME/PMI	JAVA
5	hajar momin	2ème année	DSI	BootStrap

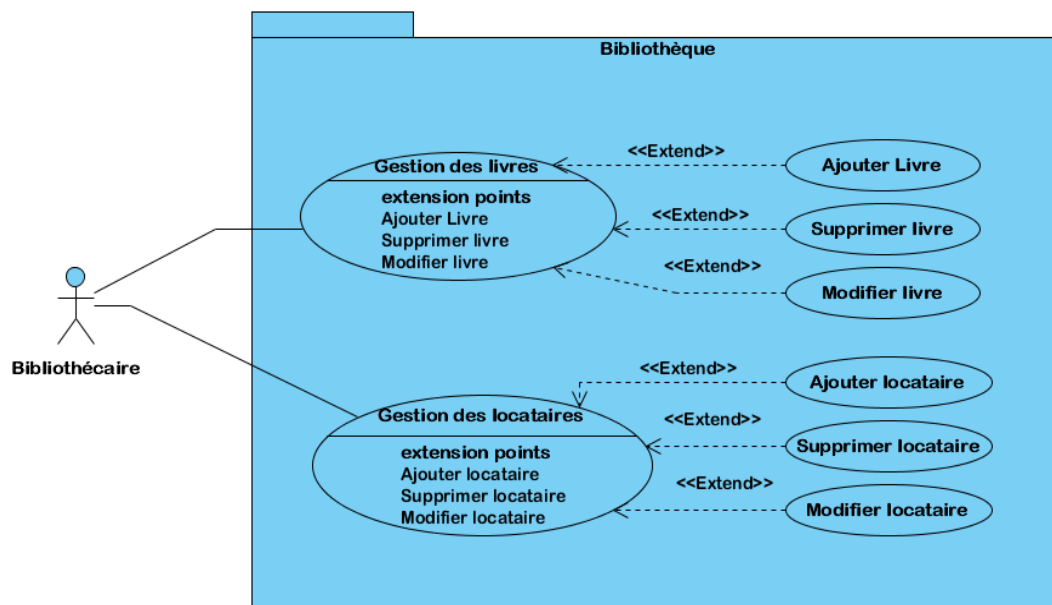
Below the table, there are input fields for "ID Locataire :", "Le nom complet :", "Livre :", "Niveau :", and "Filière :". The "Livre", "Niveau", and "Filière" fields are dropdown menus. To the right of these fields are buttons: "Ajouter", "Supprimer", "Afficher", "Modifier", "Quitter", and "Effacer".

II. Description du fonctionnement de l'application :

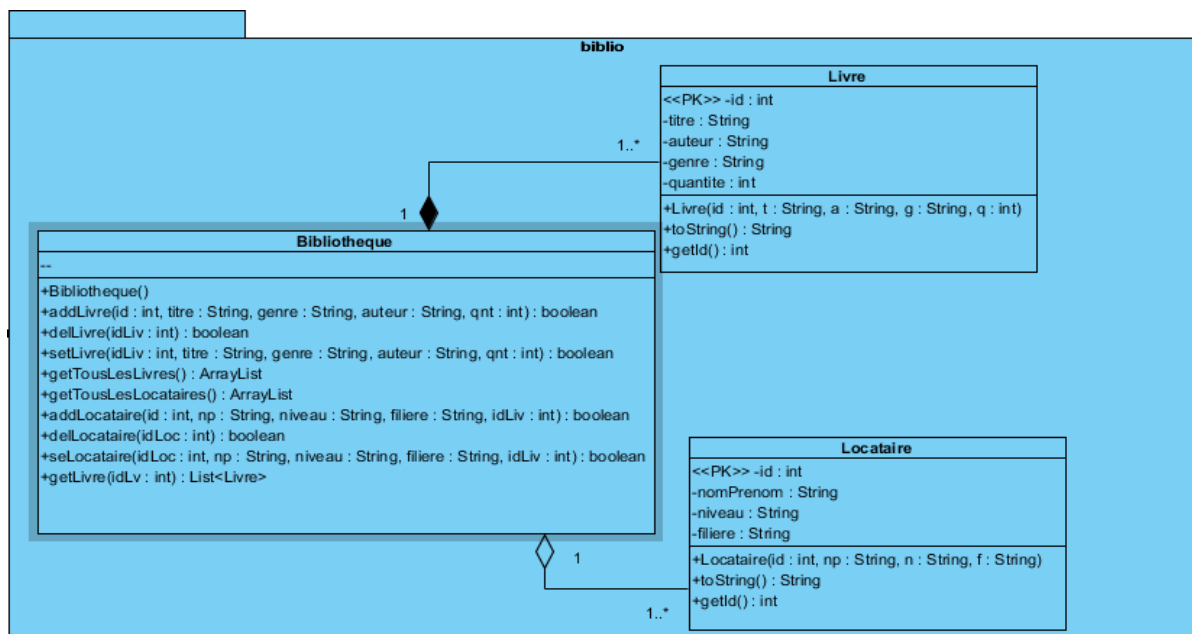
- Les boutons d'ajout des deux interfaces sont responsables d'ajout des nouveaux livres et nouvelles locataires avec vérification du formulaire.
- Les boutons de la suppression des deux interfaces sont responsables de la suppression d'un livre/locataire sélectionné dans la grille.
- Les boutons d'affichage permettent d'afficher les données d'un livre/locataire dans le formulaire selon le numéro ou id de livre/locataire saisie par le bibliothécaire dans la 1^{ère} zone de texte, pour faire des changements sur les données affichées sur le formulaire et la mise à jour des données sera faite à l'aide de bouton Modifier.
- Les boutons de modifications permettent d'effectuer une mise à jour des données.

III. Phase d'analyse :

1. Diagramme de cas d'utilisation (Use Case)



2. Diagramme de classe (Class Diagram)





3. Entity Relationship Diagram (ERD) :

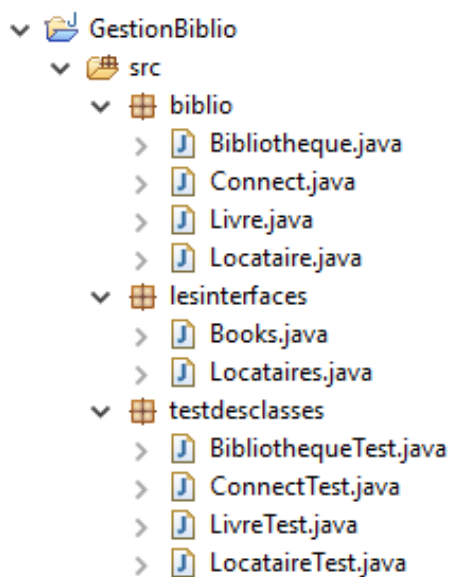


IV. Estimation de charge (Méthode COCOMO) :

1. Modèle de base :

Type de projet	Charge en M/h	TDEV en M
Organique	$\text{Charge} = 2,4 * (\text{KLOC})^{1,05}$	$\text{TDEV} = 2,5 * (\text{Charge})^{0,38}$
Semi-détaché	$\text{Charge} = 3 * (\text{KLOC})^{1,12}$	$\text{TDEV} = 2,5 * (\text{Charge})^{0,35}$
Imbriqué	$\text{Charge} = 3,6 * (\text{KLOC})^{1,20}$	$\text{TDEV} = 2,5 * (\text{Charge})^{0,32}$

1) Modèle COCOMO de base



LOC = 1466 lignes

LOC < 50 000 lignes

Donc c'est un Projet Organique.

Charge = $2,4 * (\text{KLOC})^{1,05}$

KLOC = LOC / 1000

KLOC = 1466 / 1000 = 1,466

Charge = $2,4 * (1,466)^{1,05}$

Charge = 3,6 M/h

TDEV = $2,5 * (\text{Charge})^{0,38}$

TDEV = 4,067 M

Taille moyenne d'équipe = Charge / TDEV = 3,6 M/h / 4,067 M = 0,9 h

Taille moyenne d'équipe = 1 h

✓ Très petit projet : **Charge** < 6 M/h

2. Modèle détaillé :

+ La méthode de répartition proportionnelle :

Etape	Ratio
Etude préalable	10% de la charge totale
Etude détaillé	20 à 30% de la charge totale
Etude technique	5 à 15% de la charge "réalisation"
Réalisation	2 fois la charge "étude détaillé"
Mise en œuvre	30 à 40% de la charge "réalisation"

Charge = 3,6 M/h

Etude préalable = **Charge** * 10% = 3,6 * 10% = 0,36 M/h

Etude détaillé = **Charge** * 20% = 3,6 * 20% = 0,72 M/h

Réalisation = 2 * Etude détaillé = 2 * 0,72 = 1,44 M/h

Etude technique = Réalisation * 15% = 1,44 * 15% = 0,216 M/h

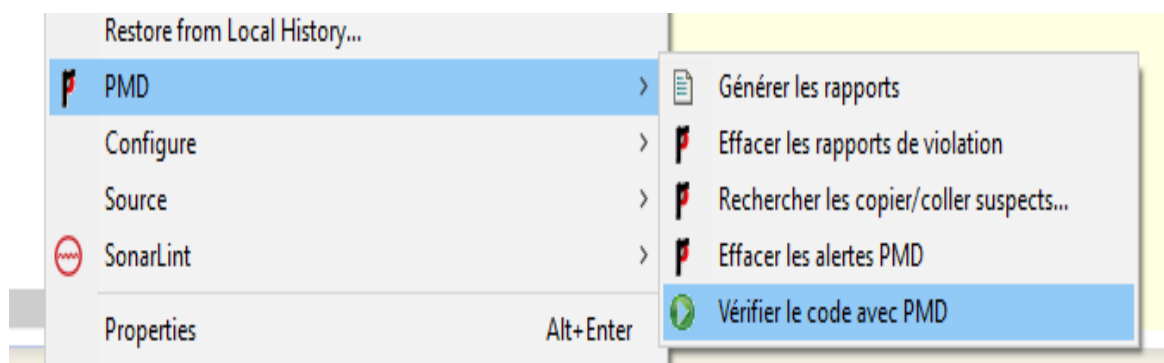
Mise en œuvre = Réalisation * 40% = 1,44 * 40% = 0,576 M/h

V. Qualité du code :

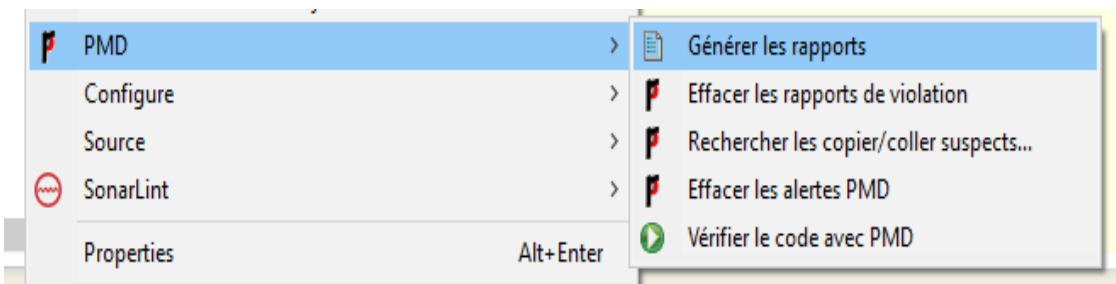
1. PMD :

PMD est un framework qui permet d'analyser le code source **Java**. Il contient un certain nombre de règles qui assurent la qualité de code : le code inutile, les imbrications trop complexes... Il permet d'obtenir le résultat par le biais d'un rapport.

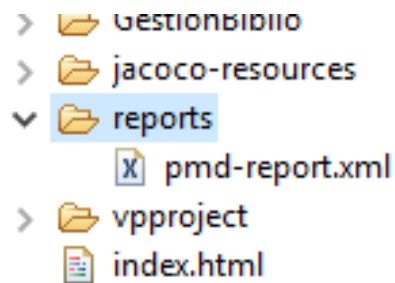
➤ Vérification du code par PMD se fait comme suit :



➤ Génération du rapport PMD :



Le rapport se trouve à l'intérieur d'un dossier nommé **reports** créé par PMD.

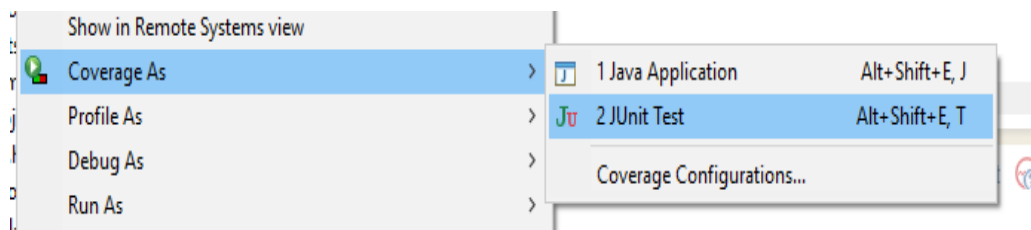


2. Eclemma :

Eclemma est un outil libre d'assurance de code Java pour Eclipse, c'est un plugin qui est combiné à JUnit.

Les tests unitaires une fois lancés, l'outil va espionner le code appelé et va effectuer un marquage du code selon un code couleur.

➤ Analyse du code :



➤ Couverture de méthodes :

Element	Coverage	Covered Instructi...	Missed Instructions	Total Instructions
▼ GestionBiblio		1 184	2 387	3 571
▼ src		1 184	2 387	3 571
▼ testdesclasses		559	11	570
> ConnectTest.java		50	0	50
> LivreTest.java		166	0	166
> LocataireTest.java		139	0	139
> BibliothequeTest.java		204	11	215
▼ biblio		625	91	716
> Livre.java		78	0	78
> Locataire.java		78	0	78
> Connect.java		58	10	68
> Bibliotheque.java		411	81	492
▼ lesinterfaces		0	2 285	2 285
> Books.java		0	1 126	1 126
> Locataires.java		0	1 159	1 159

3. SonarLint :

SonarLint est un plugin permettant de réaliser les analyses de code **SonarQube** au fil des développements, directement dans Eclipse.

➤ Lancement d'une analyse SonarLint :

