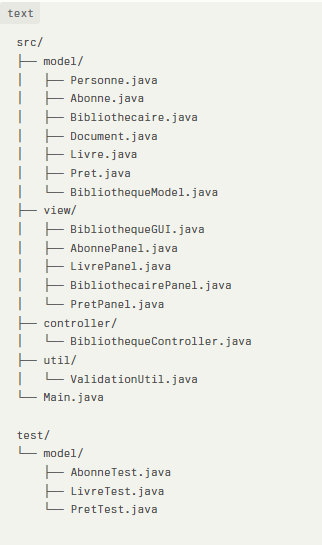
Gestion bibliothèque



Personne java

package model;

import util.ValidationUtil;

public abstract class Personne {

protected String nom;

protected String prenom;

public Personne(String nom, String prenom) {

// Validation du nom et du prénom

if (!ValidationUtil.isValidName(nom) || !ValidationUtil.isValidName(prenom)) {

throw new IllegalArgumentException("Nom ou prénom invalide");

}

this.nom = nom;

this.prenom = prenom;

}

// Getters avec copie défensive

public String getNom() {

return new String(nom);

}

public String getPrenom() {

return new String(prenom);

}

@Override

public String toString() {

return nom + " " + prenom;

}

}

Abonne.java

package model;

import util.ValidationUtil;

import java.util.Date;

import java.text.SimpleDateFormat;

public class Abonne extends Personne {

private String email;

private final Date dateInscription;

public Abonne(String nom, String prenom, String email) {

super(nom, prenom);

// Validation de l'email

if (!ValidationUtil.isValidEmail(email)) {

throw new IllegalArgumentException("Email invalide");

}

this.email = email;

this.dateInscription = new Date(); // Date courante

}

// Getter avec copie défensive

public String getEmail() {

return new String(email);

}

// Getter avec copie défensive

public Date getDateInscription() {

return new Date(dateInscription.getTime());

}

@Override

public String toString() {

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

return "Abonné: " + super.toString() + ", Email: " + email + ", Date d'inscription: " + sdf.format(dateInscription);

}

}

Bibliothecaire.java:

package model;

import util.ValidationUtil;

public class Bibliothecaire extends Personne {

private String identifiant;

public Bibliothecaire(String nom, String prenom, String identifiant) {

super(nom, prenom);

// Validation de l'identifiant

if (!ValidationUtil.isValidIdentifier(identifiant)) {

throw new IllegalArgumentException("Identifiant invalide");

}

this.identifiant = identifiant;

}

// Getter avec copie défensive

public String getIdentifiant() {

return new String(identifiant);

}

@Override

public String toString() {

return "Bibliothécaire: " + super.toString() + ", Identifiant: " + identifiant;

}

}

package model;

import util.ValidationUtil;

public class Bibliothecaire extends Personne {

private String identifiant;

public Bibliothecaire(String nom, String prenom, String identifiant) {

super(nom, prenom);

// Validation de l'identifiant

if (!ValidationUtil.isValidIdentifier(identifiant)) {

throw new IllegalArgumentException("Identifiant invalide");

}

this.identifiant = identifiant;

}

// Getter avec copie défensive

public String getIdentifiant() {

return new String(identifiant);

}

@Override

public String toString() {

return "Bibliothécaire: " + super.toString() + ", Identifiant: " + identifiant;

}

}

}

Document.java:

package model;

import util.ValidationUtil;

public abstract class Document {

protected String titre;

protected String auteur;

public Document(String titre, String auteur) {

// Validation du titre et de l'auteur

if (!ValidationUtil.isValidTitle(titre) || !ValidationUtil.isValidName(auteur)) {

throw new IllegalArgumentException("Titre ou auteur invalide");

}

this.titre = titre;

this.auteur = auteur;

}

// Getters avec copie défensive

public String getTitre() {

return new String(titre);

}

public String getAuteur() {

return new String(auteur);

}

@Override

public abstract String toString();

}

Livre.java:

package model;

public class Livre extends Document {

private int quantite;

public Livre(String titre, String auteur, int quantite) {

super(titre, auteur);

// Validation de la quantité

if (quantite < 0) {

throw new IllegalArgumentException("La quantité ne peut pas être négative");

}

this.quantite = quantite;

}

public int getQuantite() {

return quantite;

}

@Override

public String toString() {

return "Livre: " + titre + ", Auteur: " + auteur + ", Quantité: " + quantite;

}

}

Pret.java:

package model;

import java.util.Date;

import java.text.SimpleDateFormat;

import java.util.Calendar;

public class Pret {

private final Abonne abonne;

private final Document document;

private final Date dateDebut;

private final Date dateFin;

private final Bibliothecaire bibliothecaire;

public Pret(Abonne abonne, Document document, Bibliothecaire bibliothecaire) {

// Vérification des arguments non nuls

if (abonne == null || document == null || bibliothecaire == null) {

throw new IllegalArgumentException("Les arguments ne peuvent pas être nuls");

}

this.abonne = abonne;

this.document = document;

this.bibliothecaire = bibliothecaire;

this.dateDebut = new Date();

Calendar c = Calendar.getInstance();

c.setTime(this.dateDebut);

c.add(Calendar.DATE, 7);

this.dateFin = c.getTime();

}

// Getters avec copie défensive

public Abonne getAbonne() {

return new Abonne(abonne.getNom(), abonne.getPrenom(), abonne.getEmail());

}

public Document getDocument() {

if (document instanceof Livre) {

Livre livre = (Livre) document;

return new Livre(livre.getTitre(), livre.getAuteur(), livre.getQuantite());

}

// Ajouter d'autres types de documents si nécessaire

return null;

}

public Date getDateDebut() {

return new Date(dateDebut.getTime());

}

public Date getDateFin() {

return new Date(dateFin.getTime());

}

public Bibliothecaire getBibliothecaire() {

return new Bibliothecaire(bibliothecaire.getNom(), bibliothecaire.getPrenom(), bibliothecaire.getIdentifiant());

}

@Override

public String toString() {

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

return "Prêt: " + abonne.getNom() + " " + abonne.getPrenom() + " - " + document.getTitre() +

", Du: " + sdf.format(dateDebut) + " Au: " + sdf.format(dateFin) +

", Enregistré par: " + bibliothecaire.getNom();

}

}

BibliothequeModel.java:

package model;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class BibliothequeModel {

private final List<Abonne> abonnes;

private final List<Document> documents;

private final List<Pret> prets;

private final List<Bibliothecaire> bibliothecaires;

public BibliothequeModel() {

abonnes = new ArrayList<>();

documents = new ArrayList<>();

prets = new ArrayList<>();

bibliothecaires = new ArrayList<>();

}

public void ajouterAbonne(String nom, String prenom, String email) {

Abonne nouvelAbonne = new Abonne(nom, prenom, email);

abonnes.add(nouvelAbonne);

}

public void ajouterLivre(String titre, String auteur, int quantite) {

Livre nouveauLivre = new Livre(titre, auteur, quantite);

documents.add(nouveauLivre);

}

public void ajouterBibliothecaire(String nom, String prenom, String identifiant) {

Bibliothecaire nouveauBibliothecaire = new Bibliothecaire(nom, prenom, identifiant);

bibliothecaires.add(nouveauBibliothecaire);

}

public void ajouterPret(Abonne abonne, Document document, Bibliothecaire bibliothecaire) {

Pret nouveauPret = new Pret(abonne, document, bibliothecaire);

prets.add(nouveauPret);

}

// Getters avec copie défensive

public List<Abonne> getAbonnes() {

return Collections.unmodifiableList(abonnes);

}

public List<Document> getDocuments() {

return Collections.unmodifiableList(documents);

}

public List<Pret> getPrets() {

return Collections.unmodifiableList(prets);

}

public List<Bibliothecaire> getBibliothecaires() {

return Collections.unmodifiableList(bibliothecaires);

}

// Autres méthodes pour la recherche...

}

Package util:ValidationUtil.java:

package util;

import java.util.regex.Pattern;

public class ValidationUtil {

// Regex pour valider un nom (lettres, espaces, tirets et apostrophes)

private static final Pattern NAME\_PATTERN = Pattern.compile("^[\\p{L} .'-]+$");

// Regex pour valider un email

private static final Pattern EMAIL\_PATTERN = Pattern.compile("^[A-Za-z0-9+\_.-]+@(.+)$");

// Regex pour valider un identifiant (lettres et chiffres, 3 à 20 caractères)

private static final Pattern IDENTIFIER\_PATTERN = Pattern.compile("^[a-zA-Z0-9]{3,20}$");

// Regex pour valider un titre (lettres, chiffres, espaces et ponctuation courante)

private static final Pattern TITLE\_PATTERN = Pattern.compile("^[\\p{L}\\p{N}\\s\\p{Punct}]+$");

public static boolean isValidName(String name) {

return name != null && NAME\_PATTERN.matcher(name).matches();

}

public static boolean isValidEmail(String email) {

return email != null && EMAIL\_PATTERN.matcher(email).matches();

}

public static boolean isValidIdentifier(String identifier) {

return identifier != null && IDENTIFIER\_PATTERN.matcher(identifier).matches();

}

public static boolean isValidTitle(String title) {

return title != null && TITLE\_PATTERN.matcher(title).matches();

}

}

Package test:AbonneTest.java:

package test.model;

import model.Abonne;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

public class AbonneTest {

@Test

public void testCreationAbonneValide() {

assertDoesNotThrow(() -> new Abonne("Dupont", "Jean", "jean.dupont@email.com"));

}

@Test

public void testCreationAbonneNomInvalide() {

assertThrows(IllegalArgumentException.class, () -> new Abonne("Dupont123", "Jean", "jean.dupont@email.com"));

}

@Test

public void testCreationAbonnePrenomInvalide() {

assertThrows(IllegalArgumentException.class, () -> new Abonne("Dupont", "Jean123", "jean.dupont@email.com"));

}

@Test

public void testCreationAbonneEmailInvalide() {

assertThrows(IllegalArgumentException.class, () -> new Abonne("Dupont", "Jean", "jean.dupont@"));

}

}

LivreTest.java:

package test.model;

import model.Livre;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

public class LivreTest {

@Test

public void testCreationLivreValide() {

assertDoesNotThrow(() -> new Livre("Le Petit Prince", "Antoine de Saint-Exupéry", 5));

}

@Test

public void testCreationLivreTitreInvalide() {

assertThrows(IllegalArgumentException.class, () -> new Livre("", "Antoine de Saint-Exupéry", 5));

}

@Test

public void testCreationLivreAuteurInvalide() {

assertThrows(IllegalArgumentException.class, () -> new Livre("Le Petit Prince", "123", 5));

}

@Test

public void testCreationLivreQuantiteNegative() {

assertThrows(IllegalArgumentException.class, () -> new Livre("Le Petit Prince", "Antoine de Saint-Exupéry", -1));

}

}

PretTest.java:

package test.model;

import model.Abonne;

import model.Bibliothecaire;

import model.Livre;

import model.Pret;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

public class PretTest {

private Abonne abonne;

private Livre livre;

private Bibliothecaire bibliothecaire;

@BeforeEach

public void setUp() {

abonne = new Abonne("Dupont", "Jean", "jean.dupont@email.com");

livre = new Livre("Le Petit Prince", "Antoine de Saint-Exupéry", 5);

bibliothecaire = new Bibliothecaire("Martin", "Sophie", "smartin");

}

@Test

public void testCreationPretValide() {

assertDoesNotThrow(() -> new Pret(abonne, livre, bibliothecaire));

}

@Test

public void testCreationPretAbonneNull() {

assertThrows(IllegalArgumentException.class, () -> new Pret(null, livre, bibliothecaire));

}

@Test

public void testCreationPretLivreNull() {

assertThrows(IllegalArgumentException.class, () -> new Pret(abonne, null, bibliothecaire));

}

@Test

public void test

BibliothequeException.java:

package exception;

public class BibliothequeException extends Exception {

public BibliothequeException(String message) {

super(message);

}

public BibliothequeException(String message, Throwable cause) {

super(message, cause);

}

}

Maintenant, modifions quelques classes pour utiliser cette exception :BibliothequeModel.java:

package model;

import exception.BibliothequeException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class BibliothequeModel {

private final List<Abonne> abonnes;

private final List<Document> documents;

private final List<Pret> prets;

private final List<Bibliothecaire> bibliothecaires;

public BibliothequeModel() {

abonnes = new ArrayList<>();

documents = new ArrayList<>();

prets = new ArrayList<>();

bibliothecaires = new ArrayList<>();

}

public void ajouterAbonne(String nom, String prenom, String email) throws BibliothequeException {

try {

Abonne nouvelAbonne = new Abonne(nom, prenom, email);

abonnes.add(nouvelAbonne);

} catch (IllegalArgumentException e) {

throw new BibliothequeException("Erreur lors de l'ajout de l'abonné", e);

}

}

public void ajouterLivre(String titre, String auteur, int quantite) throws BibliothequeException {

try {

Livre nouveauLivre = new Livre(titre, auteur, quantite);

documents.add(nouveauLivre);

} catch (IllegalArgumentException e) {

throw new BibliothequeException("Erreur lors de l'ajout du livre", e);

}

}

public void ajouterBibliothecaire(String nom, String prenom, String identifiant) throws BibliothequeException {

try {

Bibliothecaire nouveauBibliothecaire = new Bibliothecaire(nom, prenom, identifiant);

bibliothecaires.add(nouveauBibliothecaire);

} catch (IllegalArgumentException e) {

throw new BibliothequeException("Erreur lors de l'ajout du bibliothécaire", e);

}

}

public void ajouterPret(Abonne abonne, Document document, Bibliothecaire bibliothecaire) throws BibliothequeException {

try {

Pret nouveauPret = new Pret(abonne, document, bibliothecaire);

prets.add(nouveauPret);

} catch (IllegalArgumentException e) {

throw new BibliothequeException("Erreur lors de l'ajout du prêt", e);

}

}

// Getters restent inchangés...

}

BibliothequeController.java:

package controller;

import model.\*;

import view.\*;

import exception.BibliothequeException;

public class BibliothequeController {

private BibliothequeModel model;

private BibliothequeGUI view;

public BibliothequeController(BibliothequeModel model) {

this.model = model;

this.view = new BibliothequeGUI(this);

}

public void ajouterAbonne(String nom, String prenom, String email) {

try {

model.ajouterAbonne(nom, prenom, email);

view.afficherMessage("Abonné ajouté avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de l'ajout de l'abonné: " + e.getMessage());

}

}

public void ajouterLivre(String titre, String auteur, int quantite) {

try {

model.ajouterLivre(titre, auteur, quantite);

view.afficherMessage("Livre ajouté avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de l'ajout du livre: " + e.getMessage());

}

}

public void ajouterBibliothecaire(String nom, String prenom, String identifiant) {

try {

model.ajouterBibliothecaire(nom, prenom, identifiant);

view.afficherMessage("Bibliothécaire ajouté avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de l'ajout du bibliothécaire: " + e.getMessage());

}

}

public void ajouterPret(Abonne abonne, Document document, Bibliothecaire bibliothecaire) {

try {

model.ajouterPret(abonne, document, bibliothecaire);

view.afficherMessage("Prêt ajouté avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de l'ajout du prêt: " + e.getMessage());

}

}

// Autres méthodes...

}

Maintenant, voici un diagramme de classe UML suivant les principes du DDD et du DOD :

+-------------------+ +-------------------+ +-------------------+

| Personne | | Document | | Pret |

+-------------------+ +-------------------+ +-------------------+

| - nom: String | | - titre: String | | - dateDebut: Date |

| - prenom: String | | - auteur: String | | - dateFin: Date |

+-------------------+ +-------------------+ +-------------------+

^ ^ | + effectuerPret() |

| | +-------------------+

| | |

+-------------------+ +-------------------+ |

| Abonne | | Livre | |

+-------------------+ +-------------------+ |

| - email: String | | - quantite: int | |

| - dateInscription:| +-------------------+ |

| Date | |

+-------------------+ |

^ |

| |

+-------------------+ |

| Bibliothecaire | |

+-------------------+ |

| - identifiant: | |

| String | |

+-------------------+ |

|

+-------------------+ |

| BibliothequeModel |<--------------------------------------+

+-------------------+

| - abonnes: List |

| - documents: List |

| - prets: List |

| - bibliothecaires:|

| List |

+-------------------+

| + ajouterAbonne() |

| + ajouterLivre() |

| + ajouterPret() |

| + ... |

+-------------------+

^

|

+-------------------+

|BibliothequeControl|

+-------------------+

| - model: |

| BibliothequeModel|

| - view: |

| BibliothequeGUI |

+-------------------+

| + ajouterAbonne() |

| + ajouterLivre() |

| + ajouterPret() |

| + ... |

+-------------------+

^

|

+-------------------+

| BibliothequeGUI |

+-------------------+

| + afficherMessage()|

| + afficherErreur() |

| + ... |

+-------------------+

Certainement. Je vais implémenter les méthodes pour modifier et supprimer un abonné, un livre et un prêt dans le modèle, le contrôleur et la vue. Voici les modifications à apporter :

1. Dans BibliothequeModel.java :
2. package model;
3. import exception.BibliothequeException;
4. import java.util.ArrayList;
5. import java.util.Collections;
6. import java.util.List;
7. import java.util.Date;
8. public class BibliothequeModel {
9. private final List<Abonne> abonnes;
10. private final List<Document> documents;
11. private final List<Pret> prets;
12. private final List<Bibliothecaire> bibliothecaires;
13. // ... constructeur et autres méthodes existantes ...
14. // Méthodes pour Abonné
15. public void modifierAbonne(String ancienEmail, String nouveauNom, String nouveauPrenom, String nouveauEmail) throws BibliothequeException {
16. Abonne abonne = trouverAbonneParEmail(ancienEmail);
17. if (abonne == null) {
18. throw new BibliothequeException("Abonné non trouvé");
19. }
20. try {
21. abonne = new Abonne(nouveauNom, nouveauPrenom, nouveauEmail);
22. int index = abonnes.indexOf(trouverAbonneParEmail(ancienEmail));
23. abonnes.set(index, abonne);
24. } catch (IllegalArgumentException e) {
25. throw new BibliothequeException("Erreur lors de la modification de l'abonné", e);
26. }
27. }
28. public void supprimerAbonne(String email) throws BibliothequeException {
29. Abonne abonne = trouverAbonneParEmail(email);
30. if (abonne == null) {
31. throw new BibliothequeException("Abonné non trouvé");
32. }
33. abonnes.remove(abonne);
34. }
35. private Abonne trouverAbonneParEmail(String email) {
36. return abonnes.stream()
37. .filter(a -> a.getEmail().equals(email))
38. .findFirst()
39. .orElse(null);
40. }
41. // Méthodes pour Livre
42. public void modifierLivre(String ancienTitre, String nouveauTitre, String nouveauAuteur, int nouvelleQuantite) throws BibliothequeException {
43. Livre livre = trouverLivreParTitre(ancienTitre);
44. if (livre == null) {
45. throw new BibliothequeException("Livre non trouvé");
46. }
47. try {
48. livre = new Livre(nouveauTitre, nouveauAuteur, nouvelleQuantite);
49. int index = documents.indexOf(trouverLivreParTitre(ancienTitre));
50. documents.set(index, livre);
51. } catch (IllegalArgumentException e) {
52. throw new BibliothequeException("Erreur lors de la modification du livre", e);
53. }
54. }
55. public void supprimerLivre(String titre) throws BibliothequeException {
56. Livre livre = trouverLivreParTitre(titre);
57. if (livre == null) {
58. throw new BibliothequeException("Livre non trouvé");
59. }
60. documents.remove(livre);
61. }
62. private Livre trouverLivreParTitre(String titre) {
63. return (Livre) documents.stream()
64. .filter(d -> d instanceof Livre && d.getTitre().equals(titre))
65. .findFirst()
66. .orElse(null);
67. }
68. // Méthodes pour Prêt
69. public void modifierPret(int id, Date nouvelleDateFin) throws BibliothequeException {
70. Pret pret = trouverPretParId(id);
71. if (pret == null) {
72. throw new BibliothequeException("Prêt non trouvé");
73. }
74. try {
75. pret.setDateFin(nouvelleDateFin);
76. } catch (IllegalArgumentException e) {
77. throw new BibliothequeException("Erreur lors de la modification du prêt", e);
78. }
79. }
80. public void supprimerPret(int id) throws BibliothequeException {
81. Pret pret = trouverPretParId(id);
82. if (pret == null) {
83. throw new BibliothequeException("Prêt non trouvé");
84. }
85. prets.remove(pret);
86. }
87. private Pret trouverPretParId(int id) {
88. return prets.stream()
89. .filter(p -> p.getId() == id)
90. .findFirst()
91. .orElse(null);
92. }
93. }

Dans BibliothequeController.java

package controller;

import model.\*;

import view.\*;

import exception.BibliothequeException;

import java.util.Date;

public class BibliothequeController {

private BibliothequeModel model;

private BibliothequeGUI view;

// ... constructeur et autres méthodes existantes ...

// Méthodes pour Abonné

public void modifierAbonne(String ancienEmail, String nouveauNom, String nouveauPrenom, String nouveauEmail) {

try {

model.modifierAbonne(ancienEmail, nouveauNom, nouveauPrenom, nouveauEmail);

view.afficherMessage("Abonné modifié avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de la modification de l'abonné: " + e.getMessage());

}

}

public void supprimerAbonne(String email) {

try {

model.supprimerAbonne(email);

view.afficherMessage("Abonné supprimé avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de la suppression de l'abonné: " + e.getMessage());

}

}

// Méthodes pour Livre

public void modifierLivre(String ancienTitre, String nouveauTitre, String nouveauAuteur, int nouvelleQuantite) {

try {

model.modifierLivre(ancienTitre, nouveauTitre, nouveauAuteur, nouvelleQuantite);

view.afficherMessage("Livre modifié avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de la modification du livre: " + e.getMessage());

}

}

public void supprimerLivre(String titre) {

try {

model.supprimerLivre(titre);

view.afficherMessage("Livre supprimé avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de la suppression du livre: " + e.getMessage());

}

}

// Méthodes pour Prêt

public void modifierPret(int id, Date nouvelleDateFin) {

try {

model.modifierPret(id, nouvelleDateFin);

view.afficherMessage("Prêt modifié avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de la modification du prêt: " + e.getMessage());

}

}

public void supprimerPret(int id) {

try {

model.supprimerPret(id);

view.afficherMessage("Prêt supprimé avec succès!");

} catch (BibliothequeException e) {

view.afficherErreur("Erreur lors de la suppression du prêt: " + e.getMessage());

}

}

}

1. Dans BibliothequeGUI.java, ajoutez des méthodes pour créer des formulaires de modification et de suppression :

package view;

import javax.swing.\*;

import java.awt.\*;

import controller.BibliothequeController;

import java.util.Date;

public class BibliothequeGUI extends JFrame {

private BibliothequeController controller;

// ... autres méthodes existantes ...

public void afficherFormulaireModificationAbonne(String email) {

JPanel panel = new JPanel(new GridLayout(4, 2));

JTextField nomField = new JTextField();

JTextField prenomField = new JTextField();

JTextField emailField = new JTextField();

panel.add(new JLabel("Nouveau nom:"));

panel.add(nomField);

panel.add(new JLabel("Nouveau prénom:"));

panel.add(prenomField);

panel.add(new JLabel("Nouvel email:"));

panel.add(emailField);

int result = JOptionPane.showConfirmDialog(null, panel, "Modifier Abonné",

JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

if (result == JOptionPane.OK\_OPTION) {

controller.modifierAbonne(email, nomField.getText(), prenomField.getText(), emailField.getText());

}

}

public void afficherFormulaireSuppressionAbonne() {

String email = JOptionPane.showInputDialog(this, "Entrez l'email de l'abonné à supprimer:");

if (email != null && !email.isEmpty()) {

controller.supprimerAbonne(email);

}

}

public void afficherFormulaireModificationLivre(String titre) {

JPanel panel = new JPanel(new GridLayout(4, 2));

JTextField titreField = new JTextField();

JTextField auteurField = new JTextField();

JTextField quantiteField = new JTextField();

panel.add(new JLabel("Nouveau titre:"));

panel.add(titreField);

panel.add(new JLabel("Nouvel auteur:"));

panel.add(auteurField);

panel.add(new JLabel("Nouvelle quantité:"));

panel.add(quantiteField);

int result = JOptionPane.showConfirmDialog(null, panel, "Modifier Livre",

JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

if (result == JOptionPane.OK\_OPTION) {

controller.modifierLivre(titre, titreField.getText(), auteurField.getText(),

Integer.parseInt(quantiteField.getText()));

}

}

public void afficherFormulaireSuppressionLivre() {

String titre = JOptionPane.showInputDialog(this, "Entrez le titre du livre à supprimer:");

if (titre != null && !titre.isEmpty()) {

controller.supprimerLivre(titre);

}

}

public void afficherFormulaireModificationPret(int id) {

JPanel panel = new JPanel(new GridLayout(2, 2));

JTextField dateFinField = new JTextField();

panel.add(new JLabel("Nouvelle date de fin (jj/mm/aaaa):"));

panel.add(dateFinField);

int result = JOptionPane.showConfirmDialog(null, panel, "Modifier Prêt",

JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.PLAIN\_MESSAGE);

if (result == JOptionPane.OK\_OPTION) {

// Vous devrez implémenter une méthode pour convertir la chaîne en Date

Date nouvelleDateFin = convertirEnDate(dateFinField.getText());

controller.modifierPret(id, nouvelleDateFin);

}

}

public void afficherFormulaireSuppressionPret() {

String idStr = JOptionPane.showInputDialog(this, "Entrez l'ID du prêt à supprimer:");

if (idStr != null && !idStr.isEmpty()) {

try {

int id = Integer.parseInt(idStr);

controller.supprimerPret(id);

} catch (NumberFormatException e) {

afficherErreur("ID de prêt invalide");

}

}

}

// Méthode utilitaire pour convertir une chaîne en Date

private Date convertirEnDate(String dateStr) {

// Implémentez la logique de conversion ici

// Par exemple, utilisez SimpleDateFormat

return new Date(); // Ceci est juste un placeholder

}

}

D'après les résultats de recherche, voici comment ajouter l'affichage avec JavaFX dans IntelliJ IDEA :

1. Assurez-vous d'avoir le plugin JavaFX activé dans IntelliJ IDEA.
2. Créez un nouveau projet JavaFX dans IntelliJ:
   * Allez dans File > New > Project
   * Sélectionnez JavaFX dans la liste des générateurs
   * Configurez le nom du projet, le JDK, etc.
   * Cliquez sur Create
3. IntelliJ va générer une structure de projet de base avec une classe principale et un fichier FXML.
4. Voici un exemple de code pour la classe principale (HelloApplication.java) :

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.stage.Stage;

import java.io.IOException;

public class HelloApplication extends Application {

@Override

public void start(Stage stage) throws IOException {

FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));

Scene scene = new Scene(fxmlLoader.load(), 320, 240);

stage.setTitle("Hello!");

stage.setScene(scene);

stage.show();

}

public static void main(String[] args) {

launch();

}

}

1. Et voici un exemple de fichier FXML (hello-view.fxml) :

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.layout.VBox?>

<VBox alignment="CENTER" spacing="20.0" xmlns:fx="http://javafx.com/fxml"

fx:controller="com.example.HelloController">

<padding>

<Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>

</padding>

<Label fx:id="welcomeText"/>

<Button text="Hello!" onAction="#onHelloButtonClick"/>

</VBox>

1. Vous pouvez également utiliser Scene Builder pour concevoir visuellement votre interface utilisateur

3

.

1. Pour exécuter l'application, cliquez sur le bouton Run dans IntelliJ.

N'oubliez pas de configurer correctement le SDK JavaFX dans votre projet si ce n'est pas déjà fait

5

. Vous devrez peut-être télécharger le SDK JavaFX séparément et l'ajouter à votre projet dans Project Structure > Modules > Dependencies