TD2 : Rétro-conception de diagrammes de séquence

R.3.04 : Qualité de développement

2025/2026

Ce TD a pour objectif de mettre en pratique un processus d'analyse de code Java et de création de diagrammes de séquence qui documentent certaines interactions entre objets dans le code.

1 Exercice 1 : Rétro-conception du diagramme de séquence

Analyser le code Java fourni ci-dessous et sur Moodle pour comprendre le processus d'emprunt de livre.

```java
// User.java
public class User {
 private String name;
 public User(String name) {
  this.name = name;
 }
 public String getName() {
  return name;
 }
 public void borrowBook(Book book, Library library) {
  if (library.isBookAvailable(book)) {
   library.lendBook(book);
   System.out.println(name + " successfully borrowed
       the book: " + book.getTitle());
  } else {
   System.out.println("Sorry, " + name + ". The book
       is not available.");
  }
 }
}

// Book.java
```

```java
public class Book {
 private String title;
 public Book(String title) {
  this.title = title;
 }
 public String getTitle() {
  return title;
 }
}

// Library.java
import java.util.HashMap;
public class Library {
 private HashMap<Book, Integer> inventory = new
     HashMap<>();
 public void addBook(Book book, int quantity) {
  inventory.put(book, inventory.getOrDefault(book, 0)
      + quantity);
 }
 public boolean isBookAvailable(Book book) {
  return inventory.getOrDefault(book, 0) > 0;
 }
 public void lendBook(Book book) {
  if (isBookAvailable(book)) {
   inventory.put(book, inventory.get(book) - 1);
  }
 }
}
// Main.java
public class Main {
 public static void main(String[] args) {
  Library library = new Library();
  Book book1 = new Book("The Great Gatsby");
  Book book2 = new Book("1984");
  library.addBook(book1, 3);
  library.addBook(book2, 1);
  User user = new User("Alice");
  // Borrowing books
  user.borrowBook(book1, library);
  user.borrowBook(book2, library);
  user.borrowBook(book2, library);
 }
}
```

Créer un diagramme de séquence en utilisant PlantUML pour représenter le processus d'emprunt de livre.

2 Exercice 2 : Extension de l'application

2.1 Nouveaux besoins fonctionnels

- Gestion des réservations.
- Gestion des retards et pénalités.
- Gestion des utilisateurs premium (utilisateurs pouvant emprunter un grand nombre de livres).
- Historique des emprunts.
- Notifications automatiques.

Implémenter les nouvelles classes et méthodes en Java. Écrire un code minimaliste en respectant les bonnes pratiques de codage : nommage pertinent des classes, méthodes, ... faire en sorte que les tailles des méthodes et classes ne soient pas grandes, ...

3 Diagramme de Séquence

3.1 Scénario

Un utilisateur premium essaie d'emprunter un livre non disponible, le réserve, puis est notifié quand le livre est retourné par un autre utilisateur.

Créer le diagramme de séquence avec PlantUML pour le scénario décrit (avec deux participants de type acteur et des objets de l'application).

Ajouter la gestion des pénalités pour les retards.

4 Questions pour Approfondir

1. Comment gérer les conflits si plusieurs utilisateurs réservent le même livre ?
2. Comment optimiser la recherche des réservations et des emprunts ?
3. Comment étendre le système pour gérer des livres numériques ?