Phân tích và đánh giá chức năng duyệt sản phẩm

1. Coupling

Đánh giá:

- Data Coupling: Các đối tượng giữa các lớp (Book, CD, DVD, LP, Media) chia sẻ nhiều thuộc tính và phương thức, đặc biệt là thông qua kế thừa và truy vấn repository. Tuy nhiên, các lớp này phụ thuộc mạnh vào tầng repository (MediaRepository).

Giải pháp:

1. Sử dụng Design Pattern như Strategy: Tách logic xử lý của các loại Media (Book, CD, ...) thành các chiến lược riêng biệt.
2. Giảm phụ thuộc tầng Repository: Sử dụng các interface/abstraction để đảm bảo tính trừu tượng, ví dụ như Repository cho từng loại sản phẩm riêng (BookRepository, CDRepository).

2. Cohesion

Đánh giá:

- Low Cohesion trong MediaRepository: Repository đang kiêm nhiệm nhiều vai trò (truy vấn Book, CD, DVD). Điều này làm giảm khả năng tái sử dụng.
- Class-level Cohesion: Mức độ kết dính trong các lớp Media và các subclass của nó như Book, CD, ... là tốt, vì chúng tập trung vào một loại phương tiện cụ thể.
- Functional Cohesion: Các phương thức trong lớp xử lý logic sản phẩm (MediaRepositoryImpl) khá quy trình hóa thay vì đảm bảo nguyên tắc đơn nhiệm.

Giải pháp: Chia tầng Repository thành nhiều lớp tương ứng với từng loại Media (tăng độ kết dính và phân tách trách nhiệm).

3. SOLID Principles

a. Single Responsibility Principle (SRP)

- Vi phạm: Lớp MediaRepositoryImpl đang xử lý logic truy vấn của tất cả loại sản phẩm.
- Giải pháp: Chia MediaRepositoryImpl thành các cụm nhỏ hơn (BookRepositoryImpl, CDRepositoryImpl).

b. Open/Closed Principle (OCP)

- Tuân thủ một phần: Có thể mở rộng để thêm loại sản phẩm mới, nhưng phải chỉnh sửa tầng MediaRepository.
- Giải pháp: Sử dụng Factory Pattern để tạo các repository dựa trên loại Media.

c. Liskov Substitution Principle (LSP)

- Tuân thủ: Các lớp con Book, CD, DVD kế thừa đúng chức năng từ lớp cha Media.

d. Interface Segregation Principle (ISP)

- Không áp dụng do hệ thống thiếu các Interface riêng cho từng loại xử lý.

e. Dependency Inversion Principle (DIP)

- Vi phạm một phần: Repository trực tiếp sử dụng EntityManager thay vì trích abstraction.
- Giải pháp: Inject EntityManager thông qua Constructor hoặc Factory để đảm bảo DIP.

4. Cải tiến thực tế

1. Refactor HomeScreenController với Factory Pattern

Mục tiêu là thay thế logic hiện tại trong setmediaOnMouseClick bằng cách sử dụng **Factory Pattern** để tạo các ScreenController phù hợp theo loại Media.

a. Hiện tại

HomeScreenController.java

```
1 usage  20215459-phuongvv
private void setmediaOnMouseClick(ImageView imageView, Media media) {
    if (media.getCategory().equals("Book")) {
        imageView.setOnMouseClicked(event -> new BookScreenController(currentScene, PathConfig.BOOK_PATH, media.getId(), cartController));
    } else if (media.getCategory().equals("CD")) {
        imageView.setOnMouseClicked(event -> new CDScreenController(currentScene, PathConfig.CD_PATH, media.getId(), cartController));
    } else {
        imageView.setOnMouseClicked(event -> new DVDScreenController(currentScene, PathConfig.DVD_PATH, media.getId(), cartController));
    }
}
```

b. Cải tiến

Tạo HomeScreenControllerFactory

```java
1 usage
public class ScreenControllerFactory {
    1 usage
    public static BaseScreenController getController(Scene currentScene, String mediaType, int mediaId, CartController cartController) {
        return switch (mediaType.toUpperCase()) {
            case "BOOK" -> new BookScreenController(currentScene, PathConfig.BOOK_PATH, mediaId, cartController);
            case "CD" -> new CDScreenController(currentScene, PathConfig.CD_PATH, mediaId, cartController);
            case "DVD" -> new DVDScreenController(currentScene, PathConfig.DVD_PATH, mediaId, cartController);
            default -> throw new IllegalArgumentException("Unsupported media type: " + mediaType);
        };
    }
}
```

HomeScreenController.java

```java
1 usage  20215459-phuongvv *
private void setmediaOnMouseClick(ImageView imageView, Media media) {
    imageView.setOnMouseClicked(event -> ScreenControllerFactory.getController(
            currentScene,
            media.getCategory(),
            media.getId(),
            cartController
    ));
}
```

Thay vì sử dụng if-else, gọi ScreenControllerFactory để tạo Controller phù hợp.

Ưu điểm

1. **Loại bỏ if-else:** Sử dụng Factory Pattern để giảm phức tạp và tăng khả năng mở rộng.
2. **Dễ dàng mở rộng loại Media mới:**
    - Ví dụ, thêm LP:
      case "LP" -> new LPScreenController(currentScene, PathConfig.LP_PATH, mediaId, cartController);
3. **Tách biệt trách nhiệm:**
    - HomeScreenController chỉ quan tâm việc xử lý sự kiện click.
    - Logic khởi tạo các Controller được quản lý bởi ScreenControllerFactory.

1. Tách MediaRepository thành các lớp tương ứng từng loại sản phẩm

Chia tầng Repository thành các lớp tương ứng từng loại Media (BookRepository, CDRepository, ...) để tăng độ kết dính và phân tách trách nhiệm.

a. Hiện tại

MediaRepository đang đảm nhận nhiều vai trò

```java
package com.example.aims.repository.media;

import ...

13 usages  1 implementation  20215459-phuongvv
public interface MediaRepository {
    1 usage  1 implementation  20215459-phuongvv
    MediaResponse getMediaListByFilter(String query, String priceFilter, int pageCount);
    1 usage  1 implementation  20215459-phuongvv
    Book getBookById(int id);
    1 usage  1 implementation  20215459-phuongvv
    DVD getDVDById(int id);
    1 usage  1 implementation  20215459-phuongvv
    CD getCDById(int id);
}
```

b. Cải tiến

Tách MediaRepository thành các repository cụ thể:

```java
package com.example.aims.repository.media;

import com.example.aims.entity.media.Book;

2 usages  1 implementation
public interface BookRepository {    You, 3 minute
    1 implementation
    Book getById(int id);
}
```

```java
package com.example.aims.repository.media.impl;

import com.example.aims.entity.media.Book;
import com.example.aims.repository.AIMSDB;
import com.example.aims.repository.media.BookRepository;

import javax.persistence.EntityManager;

public class BookRepositoryImpl implements BookRepository {
    @Override
    public Book getById(int id) {
        EntityManager em = AIMSDB.getEntityManager();
        return em.find(Book.class, id);
    }
}
```

Tương tự đối với CDRepository, DVDRepository, LPRepository

b. Factory Pattern tạo repository phù hợp

MediaRepositoryFactory

```java
public class MediaRepositoryFactory {
    public static Object getRepository(String mediaType) {
        return switch (mediaType) {
            case "BOOK" -> new BookRepositoryImpl();
            case "CD" -> new CDRepositoryImpl();
            case "DVD" -> new DVDRepositoryImpl();
            case "LP" -> new LPRepositoryImpl();
            default -> throw new IllegalArgumentException("Unsupported media type.");
        };
    }
}
```

Áp dụng vào BookScreenController

Ban đầu

```
1 usage   20215459-phuongvv
private void init() {
    mediaRepository = new MediaRepositoryImpl();
}

 20215459-phuongvv
@Override
public <T> void initData(T... data) {
    // init repo
    init();
    this.cartController = (CartController) data[1];
    Book book = mediaRepository.getBookById((Integer) data[0]);
```

Sau khi cải tiến

```
 20215459-phuongvv *
@Override
public <T> void initData(T... data) {
    this.cartController = (CartController) data[1];
    BookRepository bookRepository = (BookRepository) MediaRepositoryFactory.getRepository( mediaType: "BOOK");
    Book book = bookRepository.getById((Integer) data[0]);
```

Tương tự với CDController, DVDController, LPController

Ưu điểm

- **Giảm sự phụ thuộc:** Controller không còn sử dụng trực tiếp MediaRepositoryImpl, thay vào đó sử dụng Factory.
- **Dễ mở rộng:** Nếu thêm các loại sản phẩm khác như DVD, CD, … chỉ cần cập nhật MediaRepositoryFactory.
- **Tách biệt rõ ràng các trách nhiệm:** BookRepository chỉ quản lý logic liên quan đến Book.

4. Dependency Injection (DIP)

Hiện tại tầng repository và logic phụ thuộc mạnh vào EntityManager. Thay vào đó, DI giúp tăng tính linh hoạt.

Ban đầu:

```
2 usages
public class BookRepositoryImpl implements BookRepository {
    @Override
    public Book getById(int id) {
        EntityManager em = AIMSDB.getEntityManager();
        return em.find(Book.class, id);
    }
}   You, 52 minutes ago • Uncommitted changes
```

Cải tiến:

```
3 usages
public class BookRepositoryImpl implements BookRepository {
    2 usages
    private final EntityManager em;

    1 usage
    public BookRepositoryImpl(EntityManager em) {
        this.em = em;
    }

    @Override
    public Book getById(int id) {
        return em.find(Book.class, id);
    }
}   You, 25 minutes ago • Uncommitted changes
```

```
1 usage   new *
public static BookRepository getBookRepository() {
    EntityManager em = AIMSDB.getEntityManager();
    return new BookRepositoryImpl(em);
}
```

```
6 usages
public class MediaRepositoryFactory {
    3 usages
    public static Object getRepository(String mediaType) {
        return switch (mediaType) {
            case "BOOK" -> AppConfig.getBookRepository();
            case "CD" -> AppConfig.getCDRepository();
            case "DVD" -> AppConfig.getDVDRepository();
            case "LP" -> AppConfig.getLPRepository();
            default -> throw new IllegalArgumentException("Unsupported media type.");
        };
    }
}   You, A minute ago • Uncommitted changes
```