

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO THỰC HÀNH
IT3103-744527-2024.1
BÀI THỰC HÀNH – LAB 2

Họ và tên SV: Đặng Hồng Minh

MSSV: 20225740

Lớp: CNTT Việt Nhật – K67

GVHD: Lê Thị Hoa

HTGD: Đặng Mạnh Cường

Hà Nội 10/2024

Mục lục

Danh mục hình ảnh	1
BÁO CÁO THỰC HÀNH LAB 2	2
1. Bài toán đặt ra	2
2. Yêu cầu hệ thống	2
2.1. Đối với Customer:	2
2.2. Đối với Store Manager:	2
3. Use Case Diagram	3
3.1. Use Case Diagram với Customer	3
3.2. Use Case Diagram với Store Manager	3
4. Class Diagram	4
5. Source Code	5
5.1. Aims Class (Aims.java)	5
5.2. DigitalVideoDisc Class (DigitalVideoDisc.java)	5
5.3. Cart Class (Cart.java)	7
6. Kết quả demo	8
7. Reading Assignment	9
8. Answer the question	10

Danh mục hình ảnh

Hình 1: Use Case Diagram với Customer	3
Hình 2: Use Case Diagram với Customer	3
Hình 3: Class Diagram	4
Hình 4: Aims class source code (Aims.java)	5
Hình 5: DigitalVideoDisc class source code (DigitalVideoDisc.java)	6
Hình 6: Cart class source code (Cart.java)	7
Hình 7: Demo: Thêm 5 DVD và kiểm tra giỏ hàng	8
Hình 8: Demo: Xóa 1 DVD 2 lần và kiểm tra giỏ hàng	8
Hình 9: Demo: Thêm 1 DVD 2 lần và kiểm tra giỏ hàng	8
Hình 10: Demo: Xóa cả 5 DVD trong giỏ hàng và kiểm tra	8
Hình 11: Mind Map trả lời Reading Assignment	10

BÁO CÁO THỰC HÀNH LAB 2

1. Bài toán đặt ra

Thiết kế hệ thống mới cho dự án AIMS (chỉ mới có DVD)

2. Yêu cầu hệ thống

2.1. Đối với Customer:

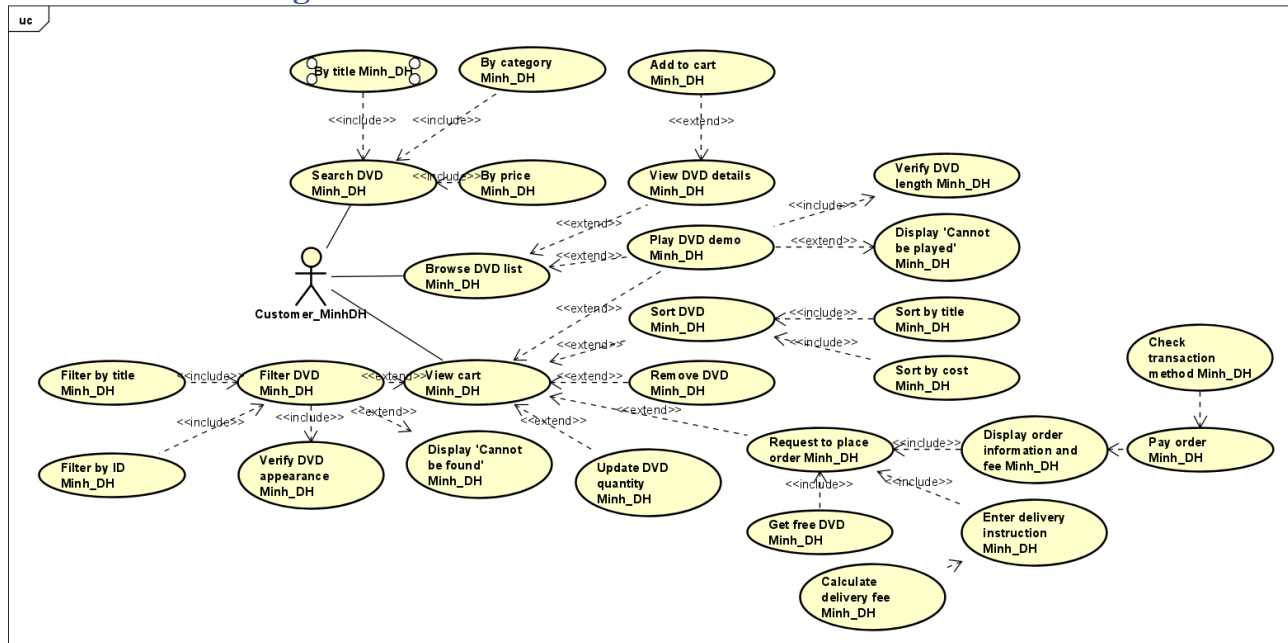
- Duyệt danh sách các đĩa DVD có sẵn trong cửa hàng, thứ tự hiển thị dựa trên ngày thêm của chúng, từ mới nhất đến cũ nhất
- Tìm kiếm các đĩa DVD theo: tiêu đề, danh mục, giá cả và chơi thử (nếu có)
- Xem thông tin chi tiết của một DVD
- Cập nhật số lượng DVD trong giỏ hàng: thêm hoặc xóa DVD trong giỏ hàng
- Xem giỏ hàng: sắp xếp thứ tự DVD theo tiêu đề hoặc giá cả, lọc DVD theo ID hoặc tiêu đề
- Đặt hàng

2.2. Đối với Store Manager:

- Đăng nhập vào hệ thống quản lý
- Thêm hoặc xóa DVD trong danh sách sản phẩm
- Xem danh sách đơn hàng đang chờ xử lý
- Xem chi tiết một đơn hàng và xử lý (chấp nhận hoặc từ chối đơn hàng)

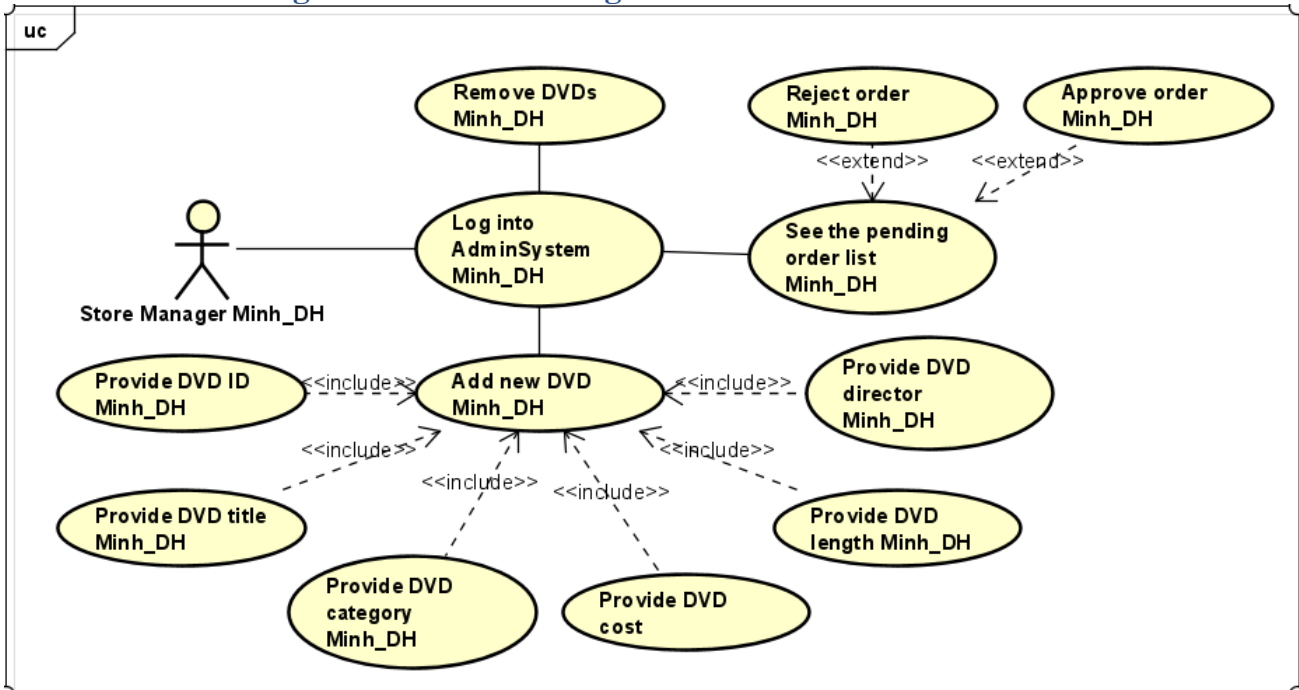
3. Use Case Diagram

3.1. Use Case Diagram với Customer



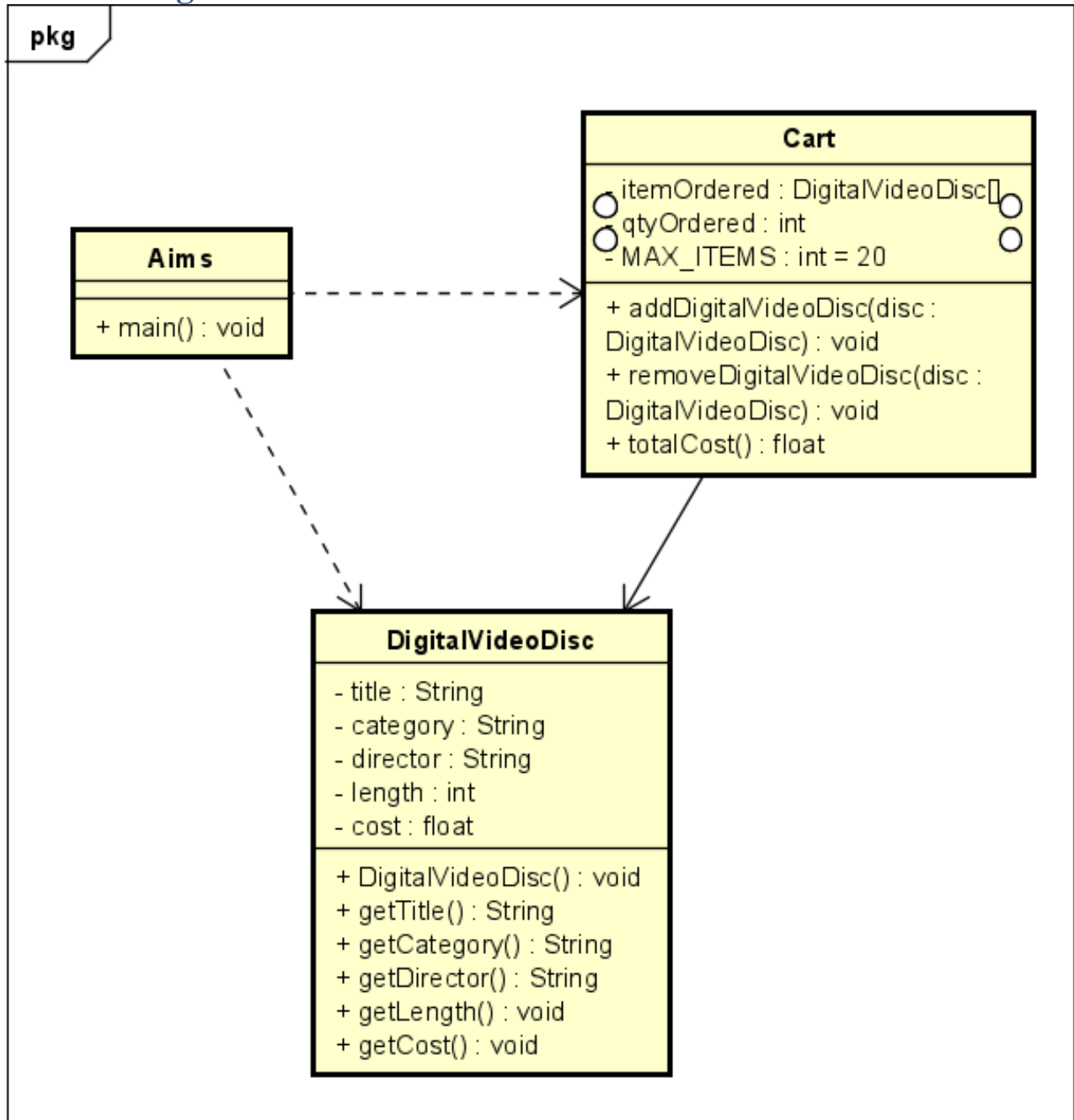
Hình 1: Use Case Diagram với Customer

3.2. Use Case Diagram với Store Manager



Hình 2: Use Case Diagram với Customer

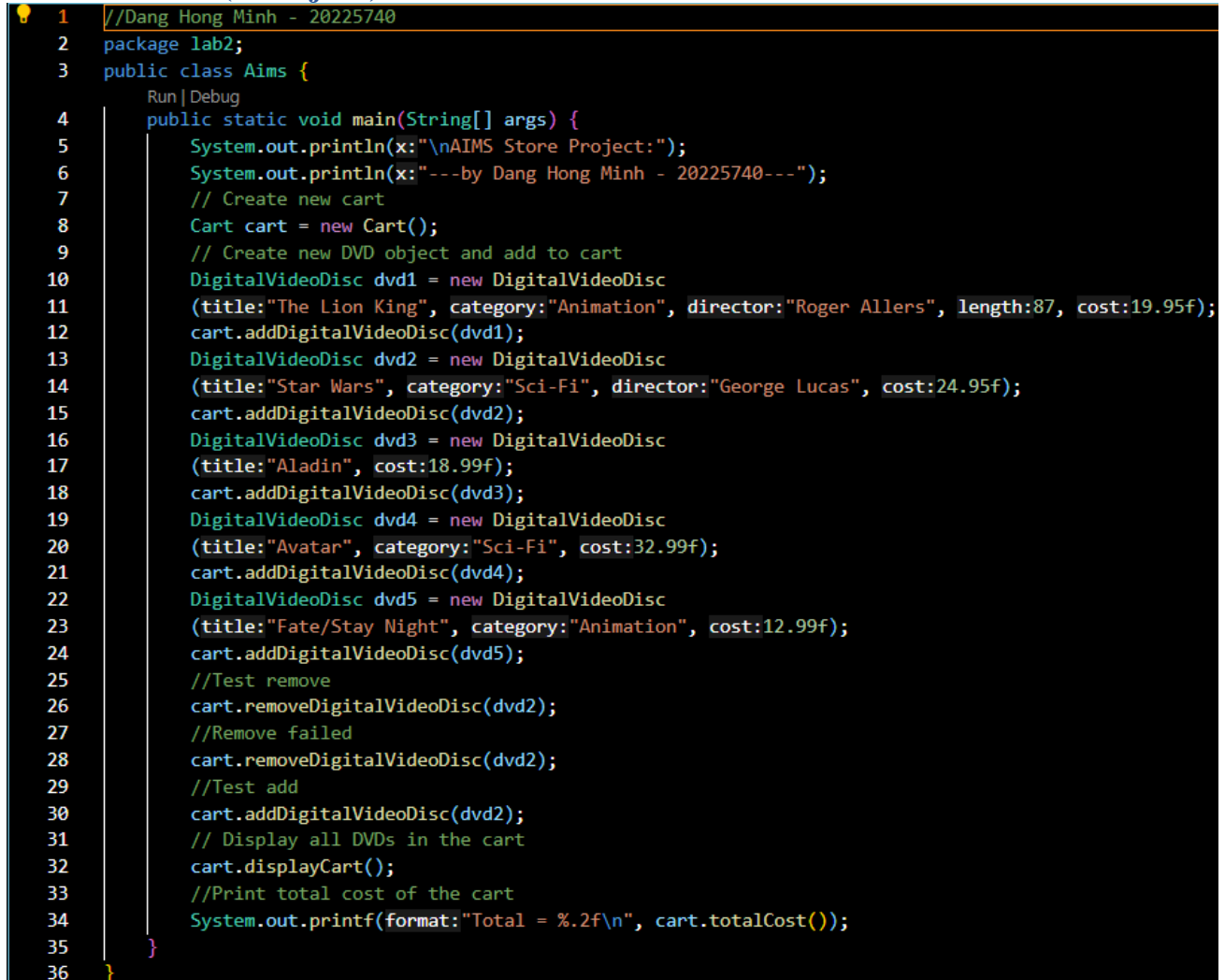
4. Class Diagram



Hình 3: Class Diagram

5. Source Code

5.1. Aims Class (Aims.java)



```
1 //Dang Hong Minh - 20225740
2 package lab2;
3 public class Aims {
4     Run | Debug
5     public static void main(String[] args) {
6         System.out.println(x:"\nAIMS Store Project:");
7         System.out.println(x:"---by Dang Hong Minh - 20225740---");
8         // Create new cart
9         Cart cart = new Cart();
10        // Create new DVD object and add to cart
11        DigitalVideoDisc dvd1 = new DigitalVideoDisc
12        (title:"The Lion King", category:"Animation", director:"Roger Allers", length:87, cost:19.95f);
13        cart.addDigitalVideoDisc(dvd1);
14        DigitalVideoDisc dvd2 = new DigitalVideoDisc
15        (title:"Star Wars", category:"Sci-Fi", director:"George Lucas", cost:24.95f);
16        cart.addDigitalVideoDisc(dvd2);
17        DigitalVideoDisc dvd3 = new DigitalVideoDisc
18        (title:"Aladin", cost:18.99f);
19        cart.addDigitalVideoDisc(dvd3);
20        DigitalVideoDisc dvd4 = new DigitalVideoDisc
21        (title:"Avatar", category:"Sci-Fi", cost:32.99f);
22        cart.addDigitalVideoDisc(dvd4);
23        DigitalVideoDisc dvd5 = new DigitalVideoDisc
24        (title:"Fate/Stay Night", category:"Animation", cost:12.99f);
25        cart.addDigitalVideoDisc(dvd5);
26        //Test remove
27        cart.removeDigitalVideoDisc(dvd2);
28        //Remove failed
29        cart.removeDigitalVideoDisc(dvd2);
30        //Test add
31        cart.addDigitalVideoDisc(dvd2);
32        // Display all DVDs in the cart
33        cart.displayCart();
34        //Print total cost of the cart
35        System.out.printf(format:"Total = %.2f\n", cart.totalCost());
36    }
```

Hình 4: Aims class source code (Aims.java)

5.2. DigitalVideoDisc Class (DigitalVideoDisc.java)

```
1 //Dang Hong Minh - 20225740
2 package lab2;
3 public class DigitalVideoDisc {
4     //Attribute
5     private String title;
6     private String category;
7     private String director;
8     private int length;
9     private float cost;
10    //Construct by title
11    public DigitalVideoDisc(String title) {
12        super();
13        this.title = title;
14    }
15    //Construct by title, cost
16    public DigitalVideoDisc(String title, float cost) {
17        super();
18        this.title = title;
19        this.cost = cost;
20    }
21    //Construct by title, category, cost
22    public DigitalVideoDisc(String title, String category, float cost) {
23        super();
24        this.title = title;
25        this.category = category;
26        this.cost = cost;
27    }
28    //Construct by title, category, director, cost
29    public DigitalVideoDisc(String title, String category, String director, float cost){
30        super();
31        this.title = title;
32        this.category = category;
33        this.director = director;
34        this.cost = cost;
35    }
36    //Construct by title, category, director, length, cost
37    public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
38        super();
39        this.title = title;
40        this.category = category;
41        this.cost = cost;
42        this.director = director;
43        this.length = length;
44    }
45    public String getTitle() {
46        return title;
47    }
48    public String getCategory() {
49        return category;
50    }
51    public float getCost() {
52        return cost;
53    }
54    public String getDirector() {
55        return director;
56    }
57    public int getLength() {
58        return length;
59    }
60 }
```

Hình 5: DigitalVideoDisc class source code (DigitalVideoDisc.java)

5.3. Cart Class (Cart.java)

```

1  //Đang Hong Minh - 20225740
2  package lab2;
3  public class Cart {
4      public static final int MAX_ITEMS = 20;
5      //Create cart
6      private DigitalVideoDisc itemOrdered[] = new DigitalVideoDisc[MAX_ITEMS];
7      //Current number of DVD
8      private int qtyOrdered = 0;
9      //Add DVD to cart
10     public void addDigitalVideoDisc(DigitalVideoDisc dvd) {
11         //If cart not full
12         if (qtyOrdered < MAX_ITEMS) {
13             itemOrdered[qtyOrdered] = dvd;
14             qtyOrdered++;
15             System.out.println(dvd.getTitle()+" has been added!");
16             if (MAX_ITEMS - qtyOrdered == 1)
17                 System.out.println(x:"Warning! The cart is almost full.");
18         } else {
19             System.out.println(x:"The cart is full! Cannot add more DVDs.");
20         }
21     }
22     public void removeDigitalVideoDisc(DigitalVideoDisc dvd) {
23         if(itemOrdered[0] == null) {
24             System.out.println(x:"ERROR! Your cart is empty!");
25             return;
26         }
27         for (int i = 0; i < qtyOrdered; i++) {
28             if (itemOrdered[i].equals(dvd)) {
29                 //Remove the disc
30                 for (int j = i; j < qtyOrdered - 1; j++)
31                     itemOrdered[j] = itemOrdered[j + 1];
32                 itemOrdered[qtyOrdered - 1] = null;
33                 qtyOrdered--;
34                 System.out.println(dvd.getTitle() + " has been removed from cart!");
35                 return;
36             }
37         }
38         //Disc not found
39         System.out.println(x:"ERROR! DVD not found in cart!");
40     }
41     public void displayCart() {
42         System.out.println(x:"\nCurrent DVDs in Cart:");
43         if (qtyOrdered == 0) {
44             System.out.println(x:"The cart is empty.");
45         } else {
46             for (int i = 0; i < qtyOrdered; i++) {
47                 DigitalVideoDisc dvd = itemOrdered[i];
48                 System.out.printf(format:"Title: %s | Cost: %.2f%n", dvd.getTitle(), dvd.getCost());
49             }
50         }
51     }
52     public float totalCost() {
53         float totalCost = 0;
54         for (int i = 0; i < qtyOrdered; i++) {
55             totalCost += itemOrdered[i].getCost();
56         }
57         return totalCost;
58     }
59 }

```

Hình 6: Cart class source code (Cart.java)

6. Kết quả demo

Các file DigitalVideoDisc.java, Cart.java không thể chạy độc lập do không có hàm main. Khi chạy chương trình cần thực thi file Aims.java

Chạy thử một vài trường hợp:

- Thêm 5 DVD và kiểm tra giỏ hàng:

```
AIMS Store Project:
---by Dang Hong Minh - 20225740---
The Lion King has been added!
Star Wars has been added!
Aladin has been added!
Avatar has been added!
Fate/Stay Night has been added!

Current DVDs in Cart:
Title: The Lion King | Cost: $19.95
Title: Star Wars | Cost: $24.95
Title: Aladin | Cost: $18.99
Title: Avatar | Cost: $32.99
Title: Fate/Stay Night | Cost: $12.99
Total = 109.87
```

Hình 7: Demo: Thêm 5 DVD và kiểm tra giỏ hàng

- Xóa 1 DVD 2 lần và kiểm tra giỏ hàng (báo lỗi)

```
AIMS Store Project:
---by Dang Hong Minh - 20225740---
The Lion King has been added!
Star Wars has been added!
Aladin has been added!
Avatar has been added!
Fate/Stay Night has been added!
Star Wars has been removed from cart!
ERROR! DVD not found in cart!

Current DVDs in Cart:
Title: The Lion King | Cost: $19.95
Title: Aladin | Cost: $18.99
Title: Avatar | Cost: $32.99
Title: Fate/Stay Night | Cost: $12.99
Total = 84.92
```

Hình 8: Demo: Xóa 1 DVD 2 lần và kiểm tra giỏ hàng

- Thêm 1 DVD 2 lần và kiểm tra giỏ hàng: - Xóa cả 5 DVD trong giỏ hàng và kiểm tra:

```
AIMS Store Project:
---by Dang Hong Minh - 20225740---
The Lion King has been added!
Star Wars has been added!
Aladin has been added!
Avatar has been added!
Fate/Stay Night has been added!
Fate/Stay Night has been added!

Current DVDs in Cart:
Title: The Lion King | Cost: $19.95
Title: Star Wars | Cost: $24.95
Title: Aladin | Cost: $18.99
Title: Avatar | Cost: $32.99
Title: Fate/Stay Night | Cost: $12.99
Title: Fate/Stay Night | Cost: $12.99
Total = 122.86
```

Hình 9: Demo: Thêm 1 DVD 2 lần và kiểm tra giỏ hàng

```
AIMS Store Project:
---by Dang Hong Minh - 20225740---
The Lion King has been added!
Star Wars has been added!
Aladin has been added!
Avatar has been added!
Fate/Stay Night has been added!
The Lion King has been removed from cart!
Star Wars has been removed from cart!
Aladin has been removed from cart!
Avatar has been removed from cart!
Fate/Stay Night has been removed from cart!

Current DVDs in Cart:
The cart is empty.
Total = 0.00
```

Hình 10: Demo: Xóa cả 5 DVD trong giỏ hàng và kiểm tra

7. Reading Assignment

- When should accessor methods be used?

(Khi nào thì nên sử dụng các phương thức truy cập (accessor methods)?)

- Vấn đề cốt lõi:

- **Vì phạm tính đóng gói:** Holub lập luận rằng các phương thức truy cập phá vỡ tính đóng gói của lớp bằng cách tiết lộ trạng thái bên trong, biến đối tượng thành nơi chứa dữ liệu thay vì là thành phần có hành vi riêng.
- **Vấn đề sử dụng quá mức:** Việc lạm dụng getter và setter có thể khiến đối tượng trở nên "nghèo nàn" (chỉ chứa dữ liệu mà không có hành vi rõ ràng).

- Những lý do hạn chế sử dụng getter và setter:

- **Tăng phụ thuộc:** Getter và setter khiến các lớp khác phụ thuộc vào chi tiết bên trong, làm cho hệ thống khó thay đổi.
- **Khuyến khích mã hóa theo thủ tục:** Getter và setter có xu hướng biến mã thành lập trình theo thủ tục thay vì lập trình hướng đối tượng.

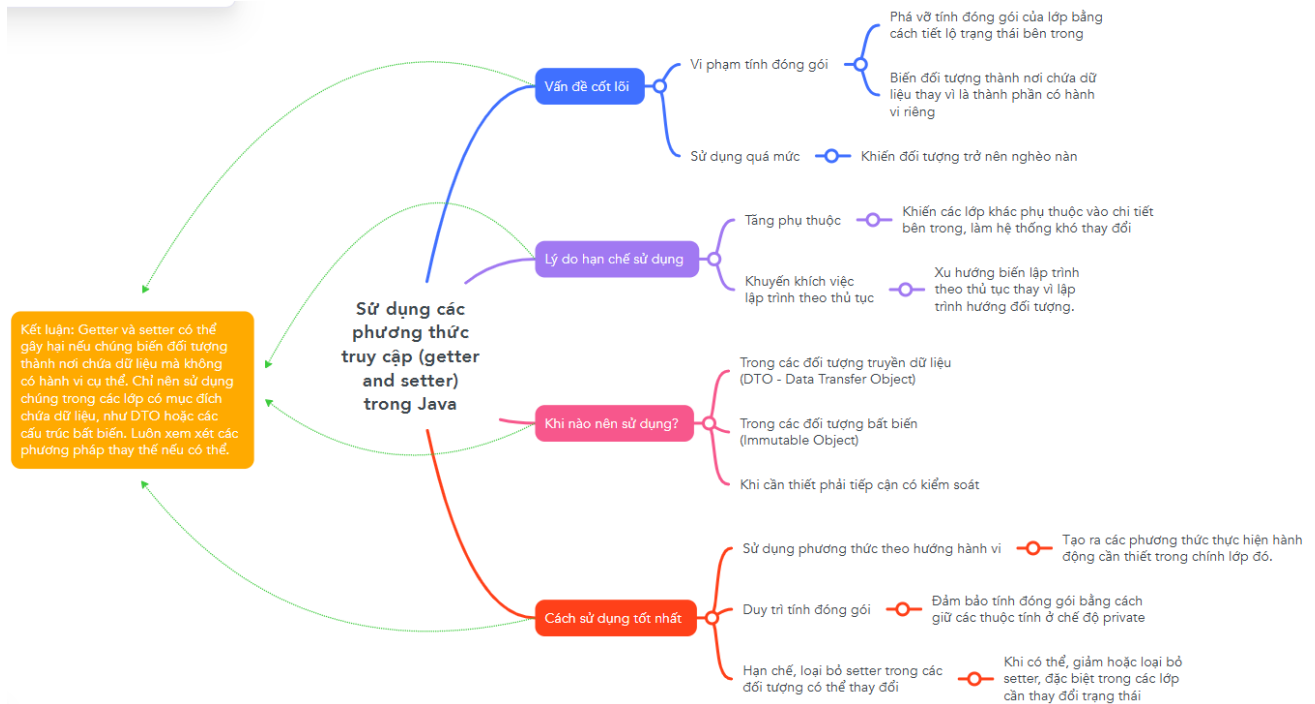
- Khi nào nên sử dụng getter và setter

- **Data Transfer Object (DTO):** Trong các đối tượng truyền dữ liệu (DTO) – nơi chỉ đơn thuần truyền dữ liệu giữa các lớp – việc sử dụng getter và setter là hợp lý.
- **Đối tượng bất biến (Immutable Objects):** Với các đối tượng bất biến, getter hữu ích vì không có setter, đảm bảo trạng thái không bị thay đổi từ bên ngoài.
- **Tiếp cận có kiểm soát:** Nếu cần thiết phải tiết lộ một phần trạng thái và phương pháp thay thế phức tạp hơn, getter có thể là lựa chọn hợp lý.

- Các sử dụng tốt nhất

- **Sử dụng phương thức theo hướng hành vi:** Thay vì tiết lộ các trường với getter và setter, tạo ra các phương thức thực hiện hành động cần thiết trong chính lớp đó.
- **Duy trì tính đóng gói:** Đảm bảo tính đóng gói bằng cách giữ các thuộc tính ở chế độ private và chỉ tiết lộ thông tin thực sự cần thiết.
- **Hạn chế hoặc loại bỏ setter trong các đối tượng có thể thay đổi:** Khi có thể, giảm hoặc loại bỏ setter, đặc biệt trong các lớp cần thay đổi trạng thái sau khi khởi tạo.

- **Kết luận:** Getter và setter có thể gây hại nếu chúng biến đối tượng thành nơi chứa dữ liệu mà không có hành vi cụ thể. Chỉ nên sử dụng chúng trong các lớp có mục đích chứa dữ liệu, như DTO hoặc các cấu trúc bất biến. Luôn xem xét các phương pháp thay thế có thể.



Hình 11: Mind Map trả lời Reading Assignment

8. Answer the question

- **If you create a constructor method to build a DVD by title then create a constructor method to build a DVD by category. Does JAVA allow you to do this?**

Nếu tạo một phương thức khởi tạo (constructor) để tạo DVD theo tiêu đề, sau đó tạo một phương thức khởi tạo khác để tạo DVD theo thể loại thì Java có cho phép làm điều này không?

- Có, Java cho phép tạo nhiều phương thức khởi tạo (constructor) trong cùng một lớp với điều kiện các constructor này phải có tham số khác nhau (số lượng, kiểu dữ liệu hoặc thứ tự của các tham số). Đây được gọi là tính năng **nạp chồng phương thức** (method overloading). Vì vậy, có thể tạo một constructor để khởi tạo DVD theo tiêu đề và một constructor khác để khởi tạo DVD theo thể loại mà không gặp xung đột trong Java, miễn là các tham số khác nhau.