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

BÁO CÁO THỰC HÀNH LAB 02

Học phần: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mã học phần: IT3103
Mã lớp: 744530

Giảng viên hướng dẫn: GV Lê Thị Hoa
Sinh viên thực hiện: Tạ Hồng Phúc
MSSV: 20225906

Hà Nội, tháng 10 năm 2024

Mục lục

Table of Contents

Table of Figures

1. Bài toán đặt ra

- Thiết kế hệ thống mới cho dự án AIMS (An Internet Media Store) và hiện tại chỉ có 1 dạng phương tiện truyền thông là Digital Video Disc (DVD).

2. Yêu cầu hệ thống

2.1. Đối với Customer:

- Khách hàng có thể xem danh sách DVD trong cửa hàng, sắp xếp theo ngày thêm từ mới nhất đến cũ nhất. Khi tìm kiếm DVD để thêm vào giỏ hàng, khách hàng có thể chọn ba cách tìm kiếm: theo tiêu đề, thể loại, hoặc giá. Kết quả tìm kiếm sẽ hiển thị danh sách DVD phù hợp (theo thứ tự mới nhất). Khách hàng có thể phát thử một phần DVD; nếu DVD có độ dài bằng 0 hoặc ít hơn, hệ thống sẽ thông báo không thể phát.
- Các phương thức tìm kiếm DVD:

    a. Theo tiêu đề: Người dùng cần cùng cấm chuỗi từ khóa, hệ thống sẽ trả lại DVD có tiêu đề chứa từ khóa không phân biệt kí tự viết hoa, viết thường.
    b. Theo danh mục: Người dùng cung cấp tên danh mục, hệ thống sẽ trả lại DVD có tiêu đề trùng khớp không phân biệt kí tự viết hoa, viết thường.
    c. Theo giá: Người dùng có thể chỉ cung cấp giá tối đa hoặc cả giá tối thiểu theo nhu cầu.

- Trong giỏ hàng, khách hàng có thể xem chi tiết, thêm, xóa, hoặc cập nhật số lượng DVD. Việc thêm DVD vào giỏ hàng có thể thực hiện từ danh sách DVD hoặc màn hình chi tiết. Khách hàng có thể lọc DVD theo ID hoặc tiêu đề, sắp xếp theo tiêu đề hoặc giá, và được chọn một sản phẩm ngẫu nhiên miễn phí. Họ có thể nghe thử DVD trước khi đặt hàng, xem thông tin chi tiết và tổng chi phí.
- Khi đặt hàng, khách hàng không cần đăng nhập, chỉ cần nhập thông tin giao hàng. Phí giao hàng sẽ tính dựa trên khối lượng và vị trí giao. Hóa đơn sẽ bao gồm danh sách DVD, tổng giá trước và sau VAT, và phí giao hàng. Khách hàng thanh toán qua thẻ tín dụng, sau đó nhận thông tin giao dịch và trạng thái đơn hàng qua email.

2.2. Đối với Store Manager:

- Quản lý cần đăng nhập để quản lý, xem và duyệt đơn hàng, thêm hoặc xóa DVD khỏi cửa hàng, với đầy đủ thông tin như ID, tiêu đề, thể loại, đạo diễn, độ dài, và giá.

3. Use Case Diagram

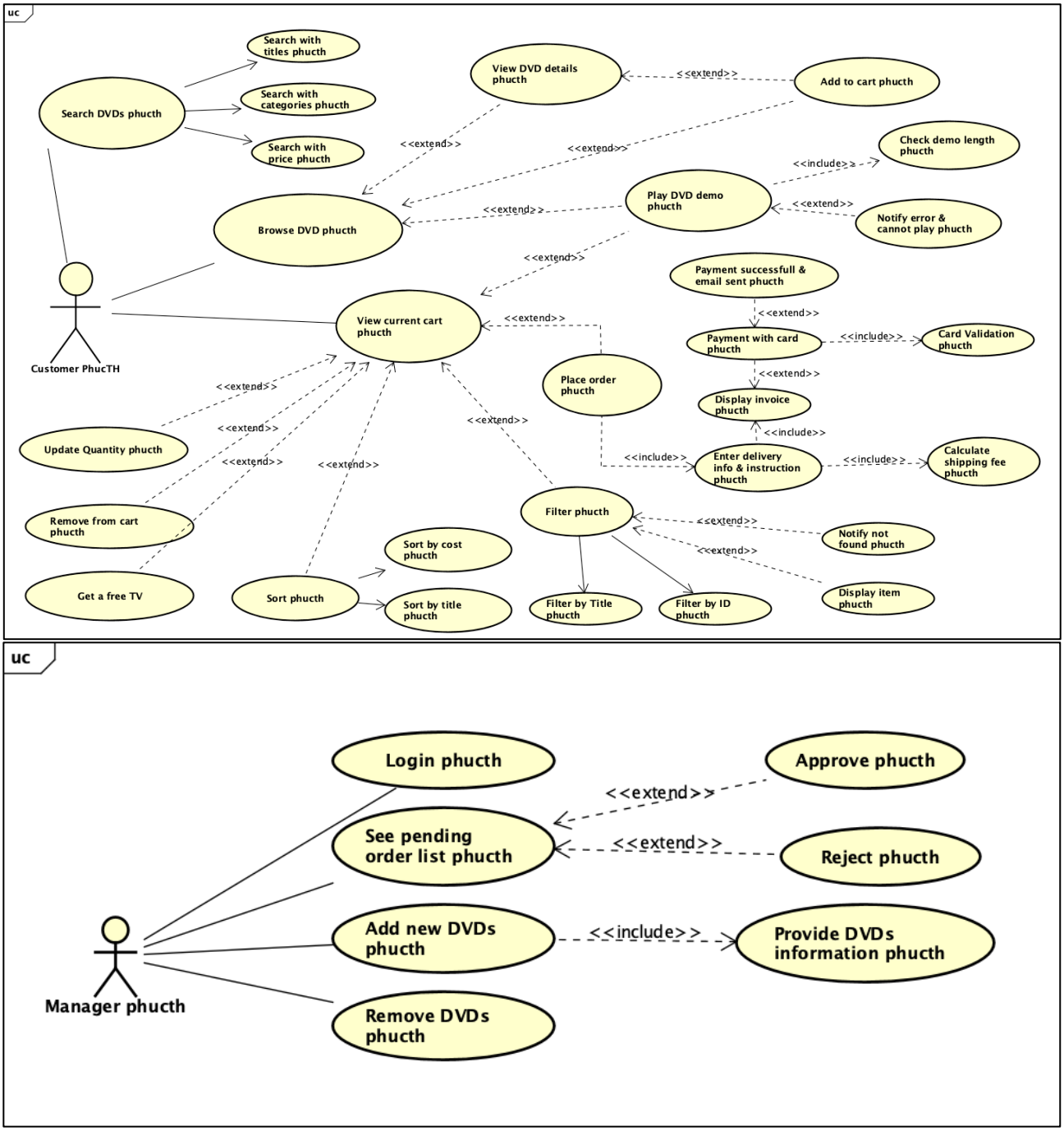

Figure 1: Use Case Diagram

Chú thích:
<<extend>> : useCase mở rộng, không bắt buộc
<<include>>: useCase bắt buộc phải có

4. Class Diagram

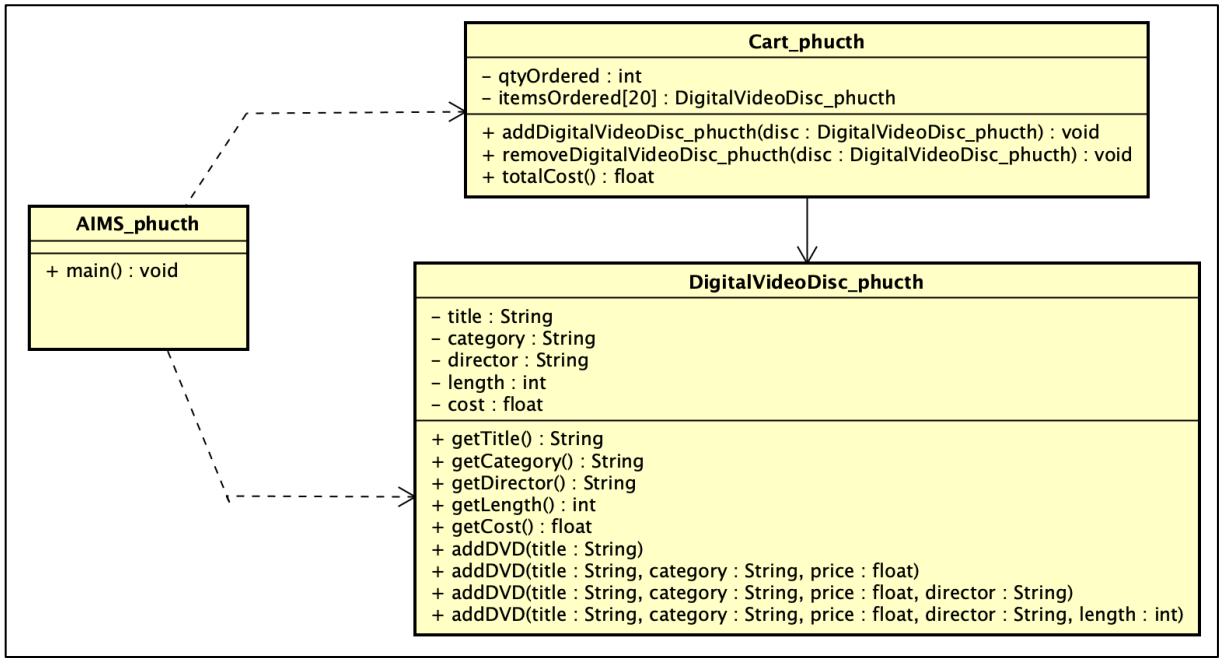

Figure 2: Class Diagram

5. Source Code

5.1. Aims Class

*Aims_phucth.java | DigitalVideoDisc_phucth.java | *Cart_phucth.java

```java
package lab02;

public class Aims_phucth {

    public static void main(String[] args) {
        //create new cart
        Cart_phucth anOrder = new Cart_phucth();

        //create new DVD and add to cart
        DigitalVideoDisc_phucth dvd1 =
                new DigitalVideoDisc_phucth("Mắt biếc", "Tình cảm", "Victor Vũ", 120, 20.5f);
        anOrder.addDigitalVideoDisc_phucth(dvd1);
        DigitalVideoDisc_phucth dvd2 =
                new DigitalVideoDisc_phucth("Bố già", "Hà", "Thánh Trần", 87, 24.15f);
        anOrder.addDigitalVideoDisc_phucth(dvd2);
        DigitalVideoDisc_phucth dvd3 =
                new DigitalVideoDisc_phucth("Despicable Me 3", "Animation", 15.35f);
        anOrder.addDigitalVideoDisc_phucth(dvd3);

        //remove a DVD
        anOrder.removeDigitalVideoDisc_phucth(dvd3);

        //print total price of the items in the cart
        System.out.println("Total cost is: " + anOrder.totalCost() + "$");
    }
}
```

Figure 3: Aims class

5.2. DigitalVideoDisc Class

```java
package lab02;

public class DigitalVideoDisc_phucth {
    private String title;
    private String category;
    private String director;
    private int length;
    private float cost;

    public String getTitle() {
        return title;
    }
    public String getCategory() {
        return category;
    }
    public String getDirector() {
        return director;
    }
    public int getLength() {
        return length;
    }
    public float getCost() {
        return cost;
    }
```

Figure 4: DigitalVideoDisc Class 1

```java
    //Create DVD by Title
    public DigitalVideoDisc_phucth(String title) {
        super();
        this.title = title;
    }
    //Create DVD by Title, Category, Cost
    public DigitalVideoDisc_phucth(String title, String category, float cost) {
        super();
        this.title = title;
        this.category = category;
        this.cost = cost;
    }
    //Create DVD by Title, Category, Director, Cost
    public DigitalVideoDisc_phucth(String title, String category, String director, float cost) {
        super();
        this.title = title;
        this.category = category;
        this.director = director;
        this.cost = cost;
    }
    //Create DVD by Title, Category, Director, Length, Cost
    public DigitalVideoDisc_phucth(String title, String category, String director, int length, float cost) {
        super();
        this.title = title;
        this.category = category;
        this.director = director;
        this.length = length;
        this.cost = cost;
    }
}
```

Figure 5: DigitalVideoDisc Class 2

5.3. Cart Class

```java
package lab02;

public class Cart_phucth {

    public static final int MAX_NUMBERS_ORDERED = 20;
    private DigitalVideoDisc_phucth itemsOrdered[]=
            new DigitalVideoDisc_phucth[MAX_NUMBERS_ORDERED];

    private int qtyOrdered = 0;

    public void addDigitalVideoDisc_phucth(DigitalVideoDisc_phucth disc) {
        if(qtyOrdered < 20) {
            itemsOrdered[qtyOrdered] = disc;
            qtyOrdered++;
            if(qtyOrdered > 15)System.out.println("Your cart is almost full!");
            else System.out.println('"' + itemsOrdered[qtyOrdered-1].getTitle() + '"'+ " has been added successfully");
        } else {
            System.out.println("Your cart is full, please create new cart or remove other DVD!");
        }
    }

    public void removeDigitalVideoDisc_phucth(DigitalVideoDisc_phucth disc) {
        boolean found = false;
        String rmvTitle = "";
        for (int i = 0; i < itemsOrdered.length; i++) {
            if (itemsOrdered[i] != null && itemsOrdered[i].equals(disc)) {
                rmvTitle = itemsOrdered[i].getTitle();
                itemsOrdered[i] = null;
                for (int j = i; j < itemsOrdered.length - 1; j++) {
                    itemsOrdered[j] = itemsOrdered[j + 1];
                }
                itemsOrdered[itemsOrdered.length - 1] = null;
                found = true;
                break;
            }
        }qtyOrdered--;

        if (found) {
                System.out.println('"' + rmvTitle + '"' + " was removed successfully.");
                if (itemsOrdered[0] == null) System.out.println("Cart is empty!.");
        } else {
            if (itemsOrdered[0] == null) System.out.println("Cart is empty!");
            else System.out.println("DVD not found.");

        }

    }
```

Figure 6: Cart Class 1

```java
    public float totalCost() {
        float totalCost = 0.0f;

        for (DigitalVideoDisc_phucth disc : itemsOrdered) {
            if (disc != null) {
                totalCost += disc.getCost();
            }
        }
        return totalCost;
    }
}
```

Figure 7: Cart Class 2

6. Kết quả demo

- Kiểm tra 3 chức năng addDigitalVideoDisc, removeDigitalVideoDisc và totalCost :

*Aims_phucth.java | DigitalVideoDisc_phucth.java | Cart_phucth.java

```
1  package lab02;
2
3  public class Aims_phucth {
4
5      public static void main(String[] args) {
6          //create new cart
7          Cart_phucth anOrder = new Cart_phucth();
8
9          //create new DVD and add to cart
10         DigitalVideoDisc_phucth dvd1 =
11                 new DigitalVideoDisc_phucth("Mắt biếc", "Tình cảm", "Victor Vũ", 120, 20.5f);
12         anOrder.addDigitalVideoDisc_phucth(dvd1);
13         DigitalVideoDisc_phucth dvd2 =
14                 new DigitalVideoDisc_phucth("Bố già", "Hà", "Thánh Trần", 87, 24.15f);
15         anOrder.addDigitalVideoDisc_phucth(dvd2);
16         DigitalVideoDisc_phucth dvd3 =
17                 new DigitalVideoDisc_phucth("Despicable Me 3", "Animation", 15.35f);
18         anOrder.addDigitalVideoDisc_phucth(dvd3);
19
20         //print total price of the items in the cart
21         System.out.println("Total cost is: " + anOrder.totalCost() + "$\n");
22
23         //remove DVDs
24         anOrder.removeDigitalVideoDisc_phucth(dvd3);
25         anOrder.removeDigitalVideoDisc_phucth(dvd2);
26         anOrder.removeDigitalVideoDisc_phucth(dvd1);
27
28         //print total price of the items in the cart
29         System.out.println("Total cost is: " + anOrder.totalCost() + "$");
30     }
31 }
```

Figure 8: Code Demo

- Kết quả:

```
"Mắt biếc" has been added successfully
"Bố già" has been added successfully
"Despicable Me 3" has been added successfully
Total cost is: 60.0$

"Despicable Me 3" was removed successfully.
"Bố già" was removed successfully.
"Mắt biếc" was removed successfully.
Cart is empty!.
Total cost is: 0.0$
```

Figure 9: Result

- Nhận xét:
  - Chức năng add thực hiện đúng.
  - Sau khi xóa DVD, chương trình có thể kiểm tra giỏ hàng rỗng.
  - Chức năng tính tiền đúng với thực tế.

7. Reading Assignment

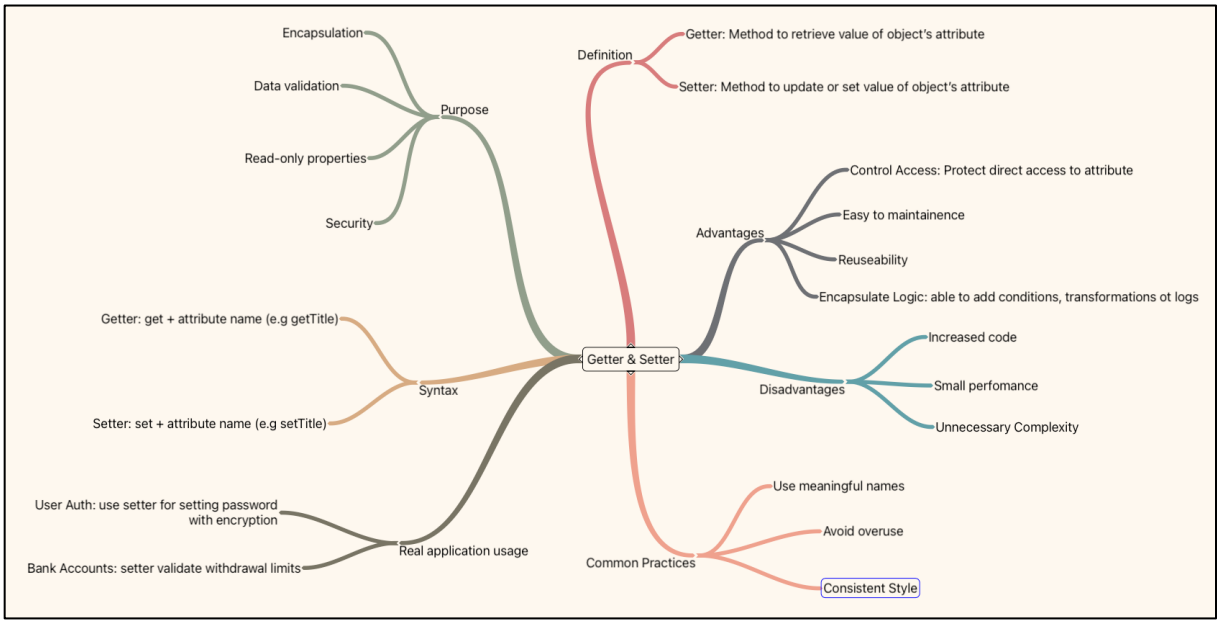

Figure 10: Getter and Setter Methods

- **Returning an object as an interface:** It's acceptable for a method to return an object through an interface, as it hides internal details, isolating external code from implementation changes. This type of method isn't a traditional "getter" since it doesn't directly expose a field.
- **Procedural boundary layers in OO systems:** Object-oriented systems often interact with procedural subsystems (like OS or databases) through boundary layers. These layers need accessor methods for flexibility, as seen in Java's JDBC, which uses accessors to support various databases.
- **Uncertain future usage:** When future use of a class is unpredictable, accessor methods can provide flexibility for unforeseen interactions, though it's best to avoid unnecessary flexibility.

8. Answer the question

- Java fully allows users to create multiple constructors for the same class, as long as they have different parameter types so that Java can distinguish between the methods. This is also known as Constructor Overloading.