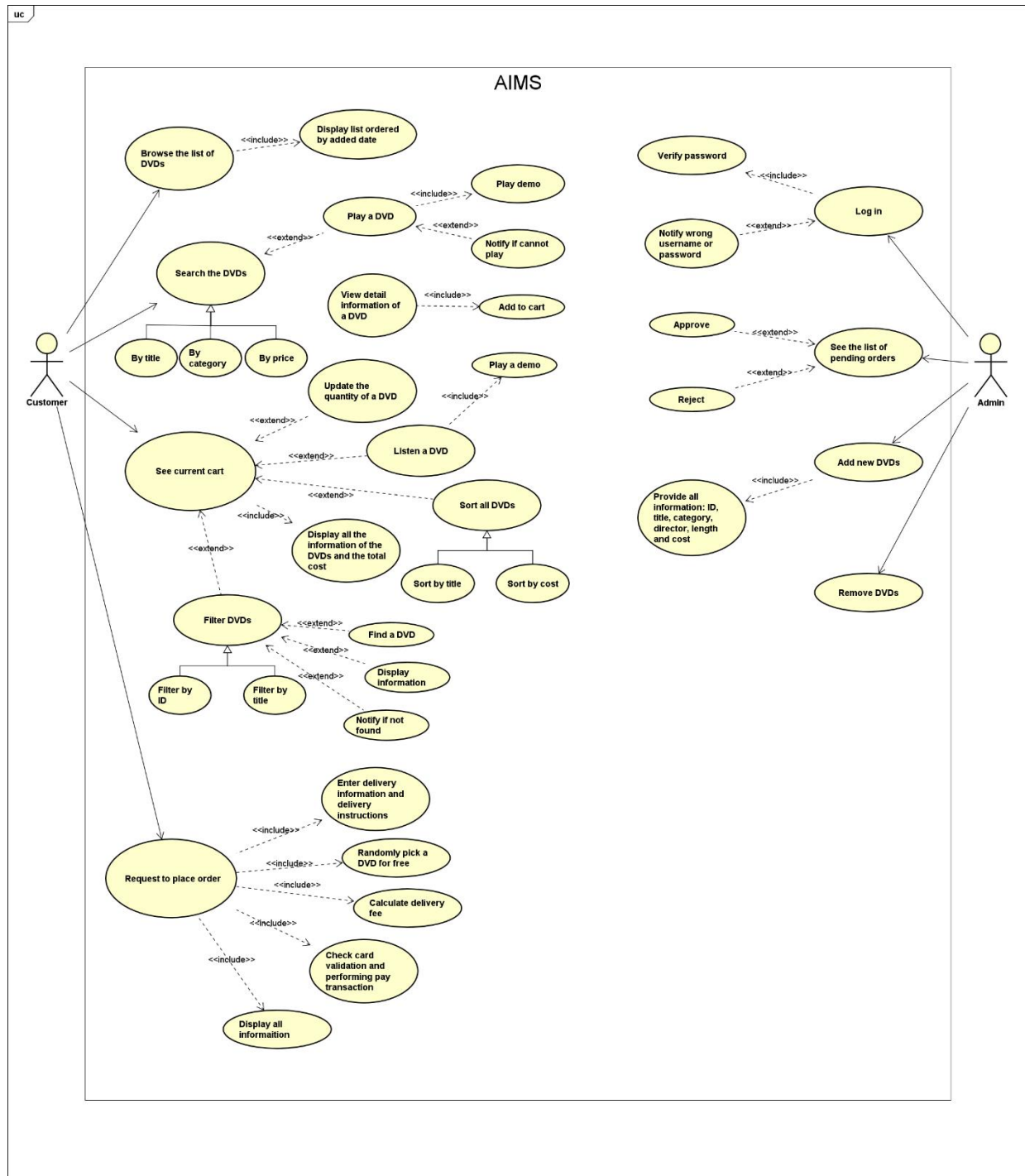


Lab02 Report

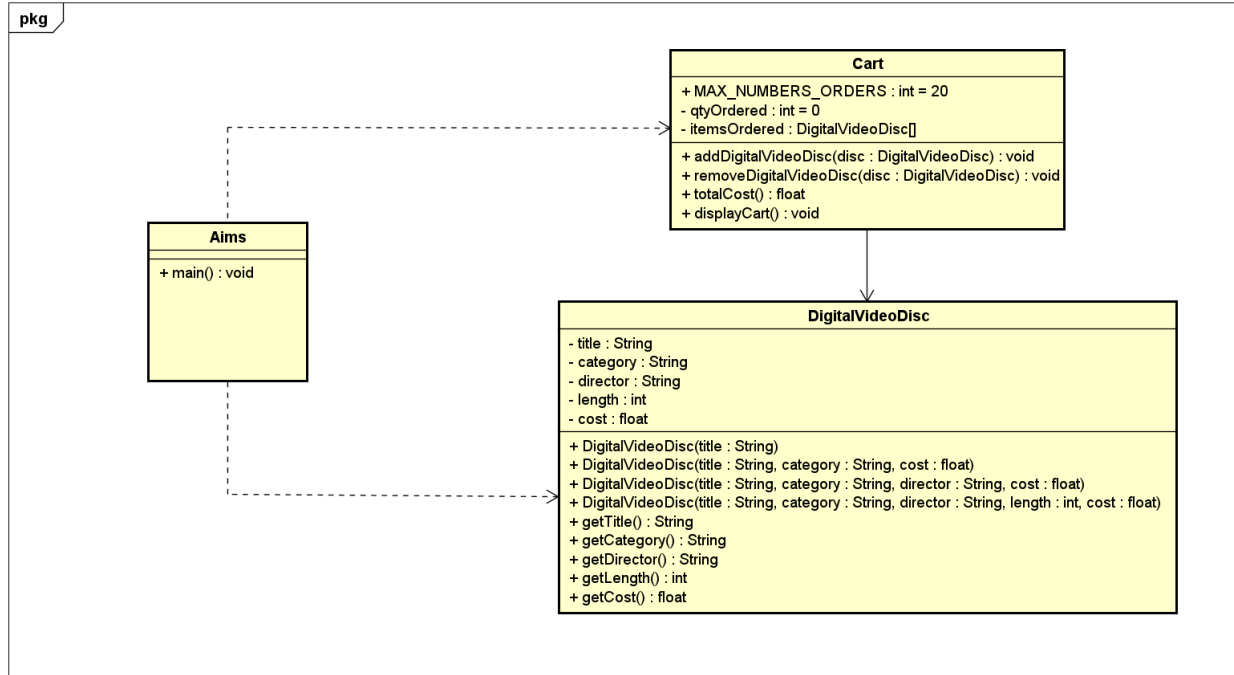
Name: Lương Minh Hiếu

ID: 20230083

1. Use-case diagram



2. Class diagram



3. Source code

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

The screenshot shows the source code of the `DigitalVideoDisc` class in an IDE. The code defines the class with private attributes for title, category, director, length, and cost. It includes four constructors: a no-argument constructor, a constructor with title, category, and cost, a constructor with title, category, director, and cost, and a full constructor with all attributes. Additionally, there are five getter methods corresponding to each attribute. The IDE interface includes a file explorer on the left, a terminal at the bottom, and a status bar at the very bottom.

This screenshot shows the Visual Studio Code editor with the `DigitalVideoDisc.java` file open. The Explorer sidebar on the left shows the project structure, including `AimsProject`, `src`, and various class files. The main editor area displays the following Java code:

```
17 this.title = title;
18 this.category = category;
19 this.director = director;
20 this.cost = cost;
21 }
22 public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
23     this.title = title;
24     this.category = category;
25     this.director = director;
26     this.length = length;
27     this.cost = cost;
28 }
29
30 public String getTitle() {
31     return title;
32 }
33 public String getCategory() {
34     return category;
35 }
36 public String getDirector() {
37     return director;
38 }
39 public int getLength() {
40     return length;
41 }
42 public float getCost() {
43     return cost;
44 }
45 }
46
```

The status bar at the bottom indicates the cursor is at line 46, column 1, with 4 spaces and UTF-8 encoding. The system tray shows the time as 2:35 PM on 10/27/2024.

This screenshot shows the Visual Studio Code editor with the `Cart.java` file open. The Explorer sidebar on the left shows the project structure, including `AimsProject`, `src`, and various class files. The main editor area displays the following Java code:

```
1 public class Cart {
2     public static final int MAX_NUMBERS_ORDERS = 20;
3     private DigitalVideoDisc itemsOrdered[] = new DigitalVideoDisc[MAX_NUMBERS_ORDERS];
4     private int qtyOrdered = 0;
5
6     public void addDigitalVideoDisc(DigitalVideoDisc disc) {
7         if (qtyOrdered < MAX_NUMBERS_ORDERS) {
8             itemsOrdered[qtyOrdered] = disc;
9             qtyOrdered++;
10            System.out.println(x:"The disc has been added.");
11        } else {
12            System.out.println(x:"The cart is almost full.");
13        }
14    }
15
16    public void removeDigitalVideoDisc(DigitalVideoDisc disc) {
17        boolean found = false;
18        for (int i = 0; i < qtyOrdered; i++) {
19            if (itemsOrdered[i].equals(disc)) {
20                found = true;
21                for (int j = i; j < qtyOrdered - 1; j++) {
22                    itemsOrdered[j] = itemsOrdered[j + 1];
23                }
24                itemsOrdered[qtyOrdered - 1] = null; // Clear the last spot
25                qtyOrdered--;
26                System.out.println(x:"The disc has been removed.");
27                break;
28            }
29        }
30        if (!found) {
31            System.out.println(x:"The disc was not found.");
32        }
33    }
34
35    public float totalCost() {
36        float result = 0;
37        for (int i = 0; i < qtyOrdered; i++) {
38            result += itemsOrdered[i].getCost();
39        }
40        return result;
41    }
42 }
```

The status bar at the bottom indicates the cursor is at line 54, column 1, with 4 spaces and UTF-8 encoding. The system tray shows the time as 2:36 PM on 10/27/2024.

```
22 itemsOrdered[j] = itemsOrdered[j + 1];
23 }
24 itemsOrdered[qtyOrdered - 1] = null; // Clear the last spot
25 qtyOrdered--;
26 System.out.println(x: "The disc has been removed.");
27 break;
28 }
29 }
30 if (!found) {
31 System.out.println(x: "The disc was not found.");
32 }
33 }
34
35 public float totalCost() {
36 float result = 0;
37 for (int i = 0; i < qtyOrdered; i++) {
38 result += itemsOrdered[i].getCost();
39 }
40 return result;
41 }
42
43 public void displayCart() {
44 for (int i = 0; i < qtyOrdered; i++) {
45 System.out.println("Disk " + (i+1) + ":");
46 System.out.println("Title: " + itemsOrdered[i].getTitle());
47 System.out.println("Category: " + itemsOrdered[i].getCategory());
48 System.out.println("Director: " + itemsOrdered[i].getDirector());
49 System.out.println("Length: " + itemsOrdered[i].getLength());
50 System.out.println("Cost: " + itemsOrdered[i].getCost());
51 }
52 }
53 }
54 }
```

```
1 public class Aims {
2 Run | Debug
3 public static void main(String[] args) {
4 Cart anOrder = new Cart();
5
6 DigitalVideoDisc disc1 = new DigitalVideoDisc(title: "The Lion King",
7 category: "Animation", director: "Roger Allers", length: 87, cost: 19.95f);
8 anOrder.addDigitalVideoDisc(disc1);
9
10 DigitalVideoDisc disc2 = new DigitalVideoDisc(title: "Star Wars",
11 category: "Science Fiction", director: "George Lucas", length: 87, cost: 24.95f);
12 anOrder.addDigitalVideoDisc(disc2);
13
14 DigitalVideoDisc disc3 = new DigitalVideoDisc(title: "Aladin", category: "Animation", cost: 18.99f);
15 anOrder.addDigitalVideoDisc(disc3);
16
17 anOrder.displayCart();
18 System.out.println("Total Cost is: " + anOrder.totalCost());
19 anOrder.removeDigitalVideoDisc(disc2);
20 System.out.println("Total Cost is: " + anOrder.totalCost());
21 }
22 }
```

4. Reading Assignment

Question: When should accessor methods be used?

Answer: In my opinion, accessor methods shouldn't be used unless absolutely necessary because these methods violate the encapsulation principle and one basic principle of OO systems is data abstraction. Using these methods would make your code harder to maintain. We can use a method to return an object in terms of an interface that the object implement because that interface isolates you from changing the implementing class. We can focus on the action not the information, don't ask the information you need to do the work; ask the object that has the information to do the work for you.

5. Answering questions

- 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?

My answer is no. This will violate the method of overloading principle since these constructor have the same name, the same number of parameters and the same type of parameters.