

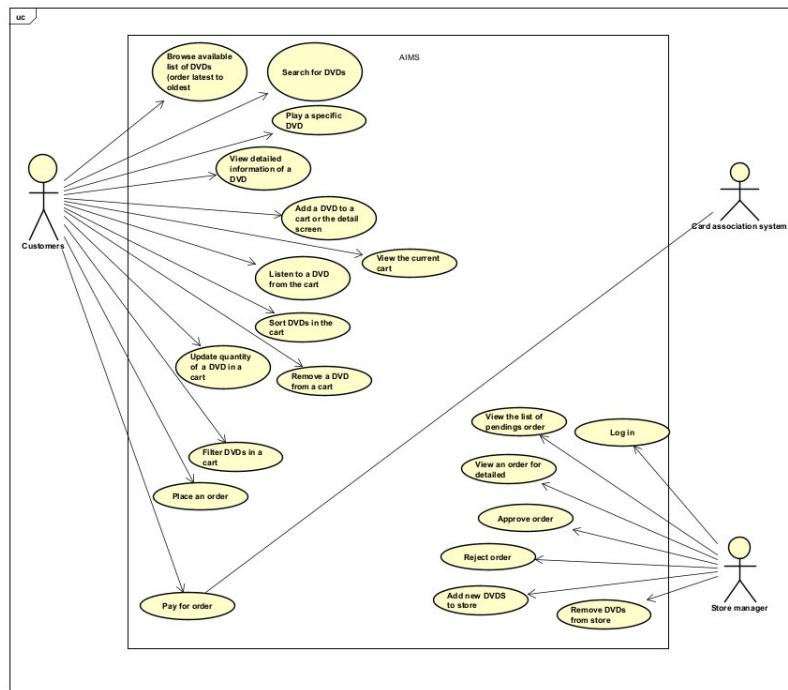
Student name: Lê Ngọc Anh Vũ

Student ID: 20236014

REPORT LAB 02

5. USE CASE DIAGRAM

Based on the problem statement in section 4, please draw the use case diagram using Astah UML for the AIMS project.

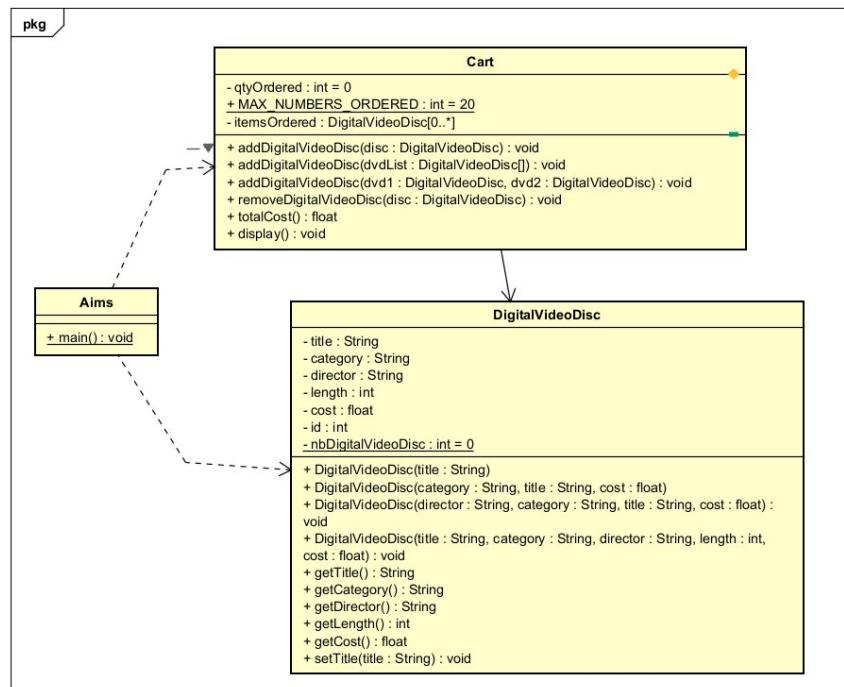


6. UML CLASS DIAGRAM FOR USE CASES RELATED TO CART MANAGEMENT

The system needs to create a new cart for the user, where it will keep information on the DVDs that the user wants to buy. The user can add, remove DVDs from the cart as well as calculate the cost. The user can add a maximum of 20 DVDs into one cart. The cart with its information and behaviors is modeled with the **Cart** class. When the user adds a DVD to the cart, the system must also create a new DVD based on the information that the user provides. This information can be displayed whenever the user decides to see it. The DVD with its information and functions is modeled with the **DigitalVideoDisc** class. Finally, the application needs an entry point for displaying to and taking input from the user (via a command-line interface), which will be the **Aims** class. A sample class diagram is illustrated in Figure 8, which includes 3 classes:

- The **Aims** class which provides a `main()` method which interacts with the rest of the system
- The **DigitalVideoDisc** class which stores the title, category, cost, director and length
- The **Cart** class to maintain an array of these **DigitalVideoDisc** objects

You have to update this class diagram following the below exercises for the final submission.



7. CREATE AIMS CLASS

```

package lab02;
public class Aims {
    public static void main(String[] args) {
    }
}

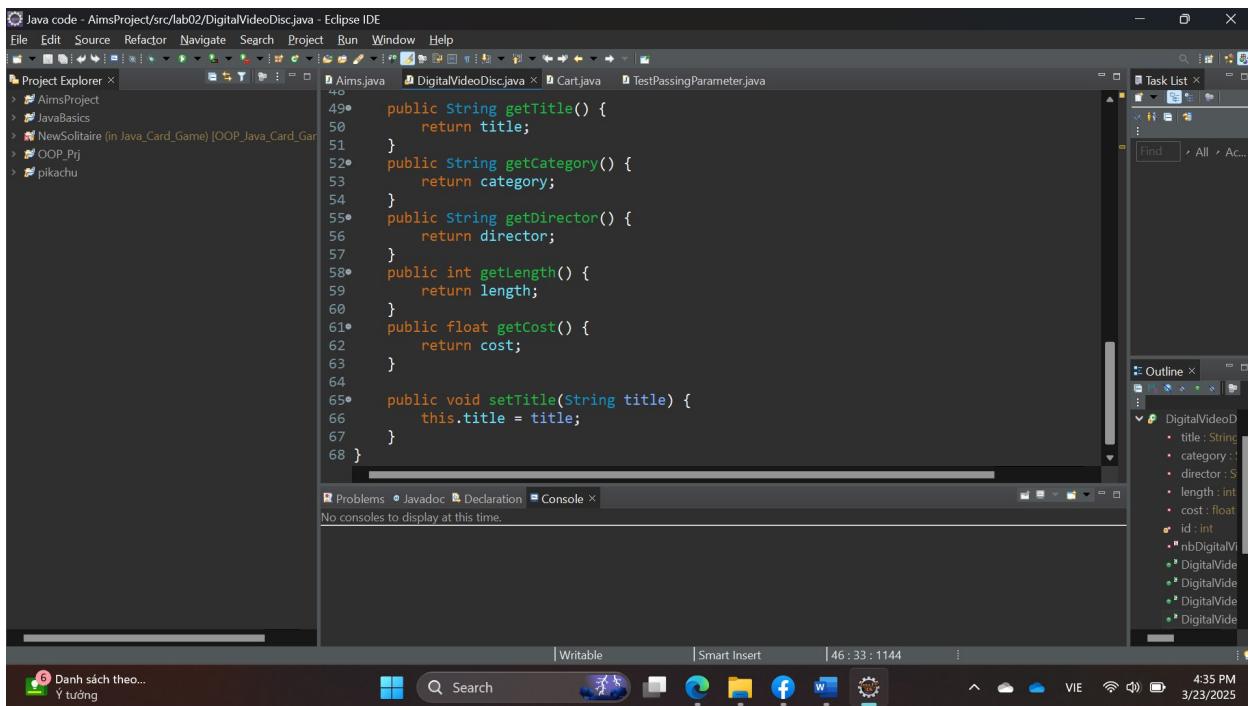
```

The screenshot shows the Eclipse IDE interface with the `Aims.java` file open in the editor. The code consists of a single package declaration and a class definition named `Aims` with a `main` method. The Project Explorer on the left shows the project structure with files like `AimsProject`, `JavaBasics`, `NewSolitaire`, `OOP_Prj`, and `pikachu`. The Outline view on the right shows the `Aims` class and its `main` method.

8. CREATE THE DIGITALVIDEODISC CLASS AND ITS ATTRIBUTES

```
Aims.java DigitalVideoDisc.java Cart.java TestPassingParameter.java
1 package lab02;
2
3 public class DigitalVideoDisc {
4     private String title;
5     private String category;
6     private String director;
7     private int length;
8     private float cost;
9     private int id;
10    private static int nbDigitalVideoDisc = 0;
11}
```

9. CREATE ACCESSORS AND MUTATORS FOR THE CLASS DIGITALVIDEODISC



10. CREATE CONSTRUCTOR METHOD

In this part, you will create yourself constructor method for DigitalVideoDisc for different purposes:

- Create a DVD object by title
- Create a DVD object by category, title and cost
- Create a DVD object by director, category, title and cost
- Create a DVD object by all attributes: title, category, director, length and cost

```

12•     public DigitalVideoDisc(String title) {
13         super();
14         this.title = title;
15         nbDigitalVideoDisc++;
16         id = nbDigitalVideoDisc;
17     }
18
19•     public DigitalVideoDisc(String category, String title, float cost) {
20         super();
21         this.title = title;
22         this.category = category;
23         this.cost = cost;
24         nbDigitalVideoDisc++;
25         id = nbDigitalVideoDisc;
26     }
27
28•     public DigitalVideoDisc(String director, String category, String title, float cost) {
29         super();
30         this.director = director;
31         this.title = title;
32         this.category = category;
33         this.cost = cost;
34         nbDigitalVideoDisc++;
35         id = nbDigitalVideoDisc;
36     }
37
38•     public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
39         super();
40         this.director = director;
41         this.title = title;
42         this.category = category;
43         this.cost = cost;
44         this.length = length;
45         nbDigitalVideoDisc++;
46         id = nbDigitalVideoDisc;
47     }

```

II. CREATE THE CART CLASS TO WORK WITH DIGITALVIDEODISC

The **Cart** class will contain a list of **DigitalVideoDisc** objects and have methods capable of modifying the list.

Add a field as an array to store a list of **DigitalVideoDisc**.

To keep track of how many DigitalVideoDiscs are in the cart, you must create a field named **qtyOrdered** in the Cart class which stores this information.

Create the method **addDigitalVideoDisc(DigitalVideoDisc disc)** to add an item to the list. You should check the current quantity to assure that the cart is not already full

- Create the method **removeDigitalVideoDisc(DigitalVideoDisc disc)** to remove the item passed by argument from the list.

Create the **totalCost()** method which loops through the values of the array and sums the costs of the individual **DigitalVideoDiscs**. This method returns the total cost of the current cart.

Note that your methods should interact with users. For example: after adding it should inform the user: "**The disc has been added**" or "**The cart is almost full**" if the cart is full.

Now you have all the classes for the application. Just practice with them in the next section.

```

package lab02;

public class Cart {
    public static final int MAX_NUMBERS_ORDERED = 20;
    private DigitalVideoDisc itemsOrdered[] = new DigitalVideoDisc[MAX_NUMBERS_ORDERED];
    private int qtyOrdered = 0;

    public void addDigitalVideoDisc(DigitalVideoDisc disc) {
        if(qtyOrdered < MAX_NUMBERS_ORDERED) {
            itemsOrdered[qtyOrdered] = disc;
            qtyOrdered++;
            System.out.println("The disc " + disc.getTitle() + " has been added");
        }
        else {
            System.out.println("The cart is almost full");
        }
    }

    public void addDigitalVideoDisc(DigitalVideoDisc[] dvdList) {
        if(dvdList.length + qtyOrdered > MAX_NUMBERS_ORDERED) {
            System.out.println("The cart is almost full, cannot add this list of dvds");
            return;
        }
        for(int i = 0; i < dvdList.length; i++) {
            itemsOrdered[qtyOrdered] = dvdList[i];
            qtyOrdered++;
        }
        System.out.println("The list of dvd has been added");
    }
}

```

```

public void removeDigitalVideoDisc(DigitalVideoDisc disc) {
    int idx;
    for(idx = 0; idx < qtyOrdered; idx++) {
        if(disc == itemsOrdered[idx]) {
            break;
        }
    }

    if(idx == qtyOrdered) {
        System.out.println("Cannot find disc " + disc.getTitle() + " in the current cart");
        return;
    }

    for(int i = idx; i < qtyOrdered - 1; i++) {
        itemsOrdered[i] = itemsOrdered[i+1];
    }
    qtyOrdered--;
    System.out.println("The disc " + disc.getTitle() + " has been removed");
}

public float totalCost() {
    float sum = 0;
    for(int i = 0; i < qtyOrdered; i++) {
        sum+=itemsOrdered[i].getCost();
    }
    return sum;
}

```

12. CREATE CARTS OF DIGITALVIDEODISCS

The **Aims** class should create a new Cart, and then create new DVDs and populate the cart with those DVDs. This will be done in the **main()** method of the Aims class. Do the following code in your main method and run the program to test.

```

package lab02;

public class Aims {
    public static void main(String[] args) {
        //Create a new cart
        Cart another = new Cart();

        //Create new dvd objects and add them to the cart
        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King",
            "Animation", "Roger Allers", 87, 19.95f);
        another.addDigitalVideoDisc(dvd1);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
            "Science Fiction", "George Lucas", 87, 24.95f);
        another.addDigitalVideoDisc(dvd2);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
            "Animation", "George Lucas", 18.99f);
        another.addDigitalVideoDisc(dvd3);

        //print total cost of the items in the cart
        System.out.println("Total Cost is: ");
        System.out.println(another.totalCost());
        System.out.print("\n");

        another.display();
    }
}

```

The result is:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with files like AimsProject, JavaBasics, NewSolitaire, OOP_Pj, and pikachu.
- Code Editor:** Displays the content of Aims.java, which contains Java code for creating a shopping cart and adding digital video discs.
- Output Console:** Shows the execution results of the program, including the addition of three discs and the total cost.
- Bottom Status Bar:** Provides system information such as temperature (26°C), battery level, and system date/time (3/23/2025, 5:45 PM).

```

Java code - AimsProject/src/lab02/Aims.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer X AimsProject JavaBasics NewSolitaire (in Java_Card_Game) [OOP_Java_Card_Gar] OOP_Pj pikachu
Aims.java X DigitalVideoDisc.java Cart.java TestPassingParameter.java Task List X
10     //Create new dvd objects and add them to the cart
11     DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King",
12         "Animation", "Roger Allers", 87, 19.95f);
13     another.addDigitalVideoDisc(dvd1);
14
15     DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
16         "Science Fiction", "George Lucas", 87, 24.95f);
17     another.addDigitalVideoDisc(dvd2);
18
19     DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
20         "Animation", "George Lucas", 18.99f);
21     another.addDigitalVideoDisc(dvd3);
22
23     //print total cost of the items in the cart
24     System.out.println("Total Cost is: ");
25     System.out.println(another.totalCost());
26
27     another.display();
28
29
Prints a String and then terminates the line. This method behaves as though it invokes print(String) and then println().
Problems Javadoc Declaration <terminated> Aims [Java Application]
The disc The Lion King has been added
The disc Star Wars has been added
The disc George Lucas has been added
Total Cost is:
63.89
Press F2 for focus

```

You are asked to write more codes for Aims and/or Cart classes to display the cart items (sequence number, title and cost for each item in one line) before the total cost.

For example:

1	The Lion King	19.95
2	Star Wars	24.95
3	Aladin	18.99
Total Cost		63.89

Source code:

```
public void display() {
    for(int i = 0; i < qtyOrdered; i++) {
        System.out.printf("%-5d%-15s%15.2f\n", i+1, itemsOrdered[i].getTitle(), itemsOrdered[i].getCost());
    }
    System.out.printf("%-5c%-15s%15.2f\n\n", ' ', "Total Cost", totalCost());
}
```

And the result is:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows projects like AimsProject, JavaBasics, NewSolitaire, OOP_Pj, and pikachu.
- Code Editor:** Displays the `Aims.java` file containing Java code for a `DigitalVideoDisc` class and a `Cart` class.
- Console Output:** Shows the execution results:
 - The disc The Lion King has been added
 - The disc Star Wars has been added
 - The disc George Lucas has been added
 - Total Cost is: 63.89
 - 1 The Lion King 19.95
 - 2 Star Wars 24.95
 - 3 George Lucas 18.99
 - Total Cost 63.89
- Bottom Status Bar:** Shows system information including temperature (26°C), search bar, and system icons.

13. REMOVING ITEMS FROM THE cart

You have to write code in your main method to test the `removeDigitalVideoDisc(DigitalVideoDisc disc)` method of the `Cart` class and check if the code is successfully run (add/remove items and display again the cart after updating).

Add this code below to test the `removeDigitalVideoDisc(DigitalVideoDisc disc)` method of the `Cart` class:

```

        //Remove disc objects
        another.removeDigitalVideoDisc(new DigitalVideoDisc("Mr.Bean", "Animation", 20.55f));
        another.removeDigitalVideoDisc(dvd2);

        another.display();
    }
}

```

And the result is:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows projects like AimsProject, JavaBasics, NewSoltaire, OOP_Pj, and pikachu.
- Code Editor:** Displays the `Aims.java` file containing Java code for managing a cart of digital video discs.
- Console:** Shows the output of the Java application. It prints a message about removing the disc "Star Wars", lists the remaining discs with their names and costs, and shows the total cost.
- Status Bar:** Shows system information including battery level (Mặt trời lặn), time (18:08), and date (3/23/2025).

```

Java code - AimsProject/src/lab02/Aims.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer X D Aims.java X D DigitalVideoDisc.java D Cart.java D TestPassingParameter.java
19     DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
20             "Animation", "George Lucas", 18.99f);
21     another.addDigitalVideoDisc(dvd3);
22
23     //print total cost of the items in the cart
24     System.out.println("Total Cost is: ");
25     System.out.println(another.totalCost());
26     System.out.print("\n");
27
28     another.display();
29
30     //Remove disc objects
31     another.removeDigitalVideoDisc(new DigitalVideoDisc("Mr.Bean", "Animation", 20.55f))
32     another.removeDigitalVideoDisc(dvd2);
33
34     another.display();
35 }
36
37
Problems Javadoc Declaration Console X
<terminated> Aims [Java Application] E:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.5.v20241023-1957\jre\bin\javaw.exe (Mar 23, 2025)
Cannot find disc Animation in the current cart
The disc Star Wars has been removed
1 The Lion King 19.95
2 George Lucas 18.99
Total Cost 38.94
Công cụ cắt ...
Đã sao chép ảnh chụp màn hình vào bảng tạm
Tự động lưu vào thư mục ảnh chụp màn hình.
Đánh dấu và Chia sẻ ...
Writable Smart Insert 35 : 6 : 1026
ENG US 5:57 PM 3/23/2025
Mặt trời lặn 18:08

```

14. WORKING WITH METHOD OVERLOADING

Method overloading allows different methods to have the **same name** but different signatures where signature can differ by **number** of input parameters or **type** of input parameter(s) or **both**.

14.1 Overloading by differing types of parameter

- Open Eclipse
- Open the JavaProject named "**AimsProject**" that you have created in the previous lab.
- Open the class `Cart.java`:** you will overload the method `addDigitalVideoDisc` you created last time.
 - The current method has one input parameter of class `DigitalVideoDisc`
 - You will create a new method that has the same name but with different types of parameters.
`addDigitalVideoDisc(DigitalVideoDisc [] dvdList)`

Source code:

```

public void addDigitalVideoDisc(DigitalVideoDisc[] dvdList) {
    if(dvdList.length + qtyOrdered > MAX_NUMBERS_ORDERED) {
        System.out.println("The cart is almost full, cannot add this list of dvds");
        return;
    }
    for(int i = 0; i < dvdList.length; i++) {
        itemsOrdered[qtyOrdered] = dvdList[i];
        qtyOrdered++;
    }
    System.out.println("The list of dvd has been added");
}

```

Test in main:

```

1 package lab02;
2
3 public class Aims {
4
5     public static void main(String[] args) {
6
7         //Create a new cart
8         Cart another = new Cart();
9
10        //Create new dvd objects and add them to the cart
11        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King",
12                                         "Animation", "Roger Allers", 87, 19.95f);
13
14        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
15                                         "Science Fiction", "George Lucas", 87, 24.95f);
16
17        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
18                                         "Animation", "George Lucas", 18.99f);
19
20        DigitalVideoDisc[] list = {dvd1, dvd2, dvd3};
21
22        another.addDigitalVideoDisc(list);
23
24        //print total cost of the items in the cart
25        System.out.println("Total Cost is: ");
26        System.out.println(another.totalCost());
27        System.out.print("\n");
28
29        another.display();
30

```

The result is:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows projects like AimsProj, Workspace, JavaBasics, NewSoltaire, OOP_Pj, and pikachu.
- Aims.java:** The code defines a class Aims with a main method that creates a Cart object and adds three DigitalVideoDisc objects to it. The output window shows the execution results.
- Output Window:** Displays the following text:


```

The list of dvd has been added
Total Cost is:
63.89

1 The Lion King      19.95
2 Star Wars         24.95
3 George Lucas     18.99
Total Cost          63.89
      
```
- System Tray:** Shows weather (25°C), battery status, and system icons.

- Try to add a method **addDigitalVideoDisc** which allows to pass an arbitrary number of arguments for dvd. Compared to an array parameter. What do you prefer in this case?

Source code:

```

public void addDigitalVideoDisc(DigitalVideoDisc ... dvdList) {
    if(dvdList.length + qtyOrdered > MAX_NUMBERS_ORDERED) {
        System.out.println("The cart is almost full, cannot add this list of dvds");
        return;
    }
    for(int i = 0; i < dvdList.length; i++) {
        itemsOrdered[qtyOrdered] = dvdList[i];
        qtyOrdered++;
    }
    System.out.println("The list of dvd has been added");
}
      
```

Test in main:

```

package lab02;

public class Aims {
    public static void main(String[] args) {
        //Create a new cart
        Cart another = new Cart();

        //Create new dvd objects and add them to the cart
        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King",
            "Animation", "Roger Allers", 87, 19.95f);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
            "Science Fiction", "George Lucas", 87, 24.95f);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
            "Animation", "George Lucas", 18.99f);

        another.addDigitalVideoDisc(dvd1, dvd2, dvd3);

        //print total cost of the items in the cart
        System.out.println("Total Cost is: ");
        System.out.println(another.totalCost());
        System.out.print("\n");

        another.display();
    }
}

```

The result is:

The screenshot shows the Eclipse IDE interface with the Aims.java file open in the editor. The code is identical to the one above. In the bottom right corner of the editor, there is a message: "The list of dvd has been added". Below this, the console output shows:

```

Total Cost is:
63.89

1 The Lion King      19.95
2 Star Wars         24.95
3 George Lucas     18.99
Total Cost          63.89

```

In situations that we just need to add a few DVDs, I prefer to use the second method.

14.2 Overloading by differing the number of parameters

- Continuing focus on the **Cart** class
- Create new method named **addDigitalVideoDisc**
 - The signature of this method has two parameters as following:
`addDigitalVideoDisc(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2)`

```
public void addDigitalVideoDisc(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {  
    if(qtyOrdered + 2 > MAX_NUMBERS_ORDERED) {  
        System.out.println("Cannot add two dvd because of full cart");  
        return;  
    }  
  
    itemsOrdered[qtyOrdered++] = dvd1;  
    System.out.println("The disc " + dvd1.getTitle() + " has been added");  
    itemsOrdered[qtyOrdered++] = dvd2;  
    System.out.println("The disc " + dvd2.getTitle() + " has been added");  
}
```

Test in main:

```
package lab02;  
  
public class Aims {  
  
    public static void main(String[] args) {  
  
        //Create a new cart  
        Cart another = new Cart();  
  
        //Create new dvd objects and add them to the cart  
        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King",  
            "Animation", "Roger Allers", 87, 19.95f);  
  
        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",  
            "Science Fiction", "George Lucas", 87, 24.95f);  
  
        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",  
            "Animation", "George Lucas", 18.99f);  
  
        another.addDigitalVideoDisc(dvd1, dvd2);  
        another.addDigitalVideoDisc(dvd3);  
  
        //print total cost of the items in the cart  
        System.out.println("Total Cost is: ");  
        System.out.println(another.totalCost());  
        System.out.print("\n");  
  
        another.display();  
    }  
}
```

The result is:

The screenshot shows the Eclipse IDE interface with the following details:

- Java code - AimsProject/src/lab02/Aims.java - Eclipse IDE**: The code defines a `DigitalVideoDisc` class and a `Cart.java` class. It creates three `DigitalVideoDisc` objects (`dvd1`, `dvd2`, `dvd3`) and adds them to a `Cart`. The `Cart` then prints the total cost and individual item details.
- Console Output**:

```
The disc The Lion King has been added
The disc Star Wars has been added
The disc George Lucas has been added
Total Cost is:
63.89

1 The Lion King      19.95
2 Star Wars        24.95
3 George Lucas     18.99
Total Cost          63.89
```
- System Information**:
 - Temperature: 25°C
 - Có năng
- Taskbar**: Shows various application icons.
- System Tray**: Shows battery status, signal strength, and system icons.

I5. PASSING PARAMETER

Now, you will practice with the `DigitalVideoDisc` class to test how JAVA passes parameters. For this exercise, you will need to temporarily add a setter for the attribute title of the `DigitalVideoDisc` class. Create a new class named `TestPassingParameter` in the current project.

The screenshot shows the Eclipse IDE interface with the following details:

- Java code - AimsProject/src/lab02/TestPassingParameter.java - Eclipse IDE**: The code contains a `TestPassingParameter` class with a `main` method and a `swap` helper method. It creates two `DigitalVideoDisc` objects and swaps their titles using the `swap` method.
- Console Output**:

```
jungle dvd title: Jungle
cinderella dvd title: Cinderella
jungle dvd title: Cinderella
```
- System Information**:
 - Temperature: 25°C
 - Có năng
- Taskbar**: Shows various application icons.
- System Tray**: Shows battery status, signal strength, and system icons.

16. CLASSIFIER MEMBER AND INSTANCE MEMBER

Now, you open the **DigitalVideoDisc** class:

Notice: You should note that this class only has instance variables: **title**, **category**, **director**, **length**, **cost**.

- We know that each DVD has a unique id assigned by the system. One simple way to manage all the ids is to give them out to new DVDs as consecutively incremented values. In order to do this, we must keep track of the number of DVDs created.
- Create a class attribute named "**nbDigitalVideoDiscs**" in the class **DigitalVideoDisc**
- Create an instance attribute named "**id**" in the class **DigitalVideoDisc**

```
private static int nbDigitalVideoDiscs = 0;
```

- Each time an instance of the **DigitalVideoDisc** class is created, the **nbDigitalVideoDiscs** should be updated. Therefore, you should update the value for this class variable inside the constructor method and assign the appropriate value for the **id**.

```
private static int nbDigitalVideoDisc = 0;

public DigitalVideoDisc(String title) {
    super();
    this.title = title;
    nbDigitalVideoDisc++;
    id = nbDigitalVideoDisc;
}

public DigitalVideoDisc(String category, String title, float cost) {
    super();
    this.title = title;
    this.category = category;
    this.cost = cost;
    nbDigitalVideoDisc++;
    id = nbDigitalVideoDisc;
}

public DigitalVideoDisc(String director, String category, String title, float cost) {
    super();
    this.director = director;
    this.title = title;
    this.category = category;
    this.cost = cost;
    nbDigitalVideoDisc++;
    id = nbDigitalVideoDisc;
}
```

```
public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
    super();
    this.director = director;
    this.title = title;
    this.category = category;
    this.cost = cost;
    this.length = length;
    nbDigitalVideoDisc++;
    id = nbDigitalVideoDisc;
}
```