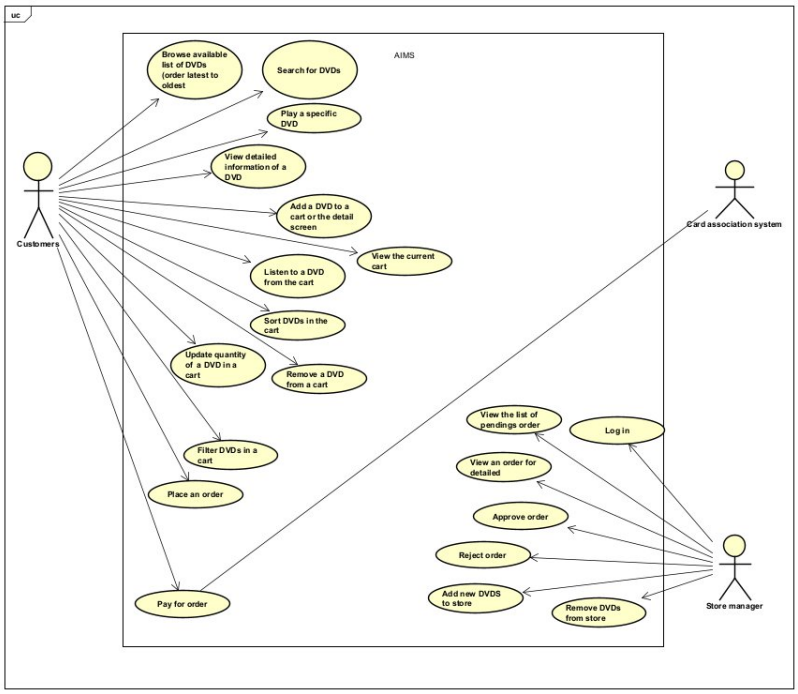
Student name: Lê Ngọc Anh Vũ

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.

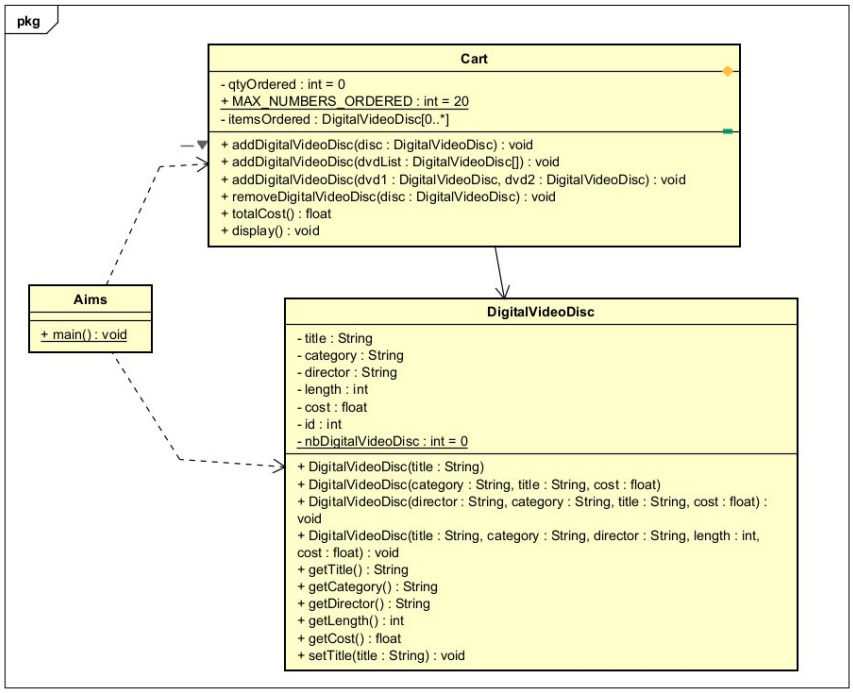


6. UML CLASS DIAGRAM FOR USE CASES RELATED TO CART MANAGEMENT

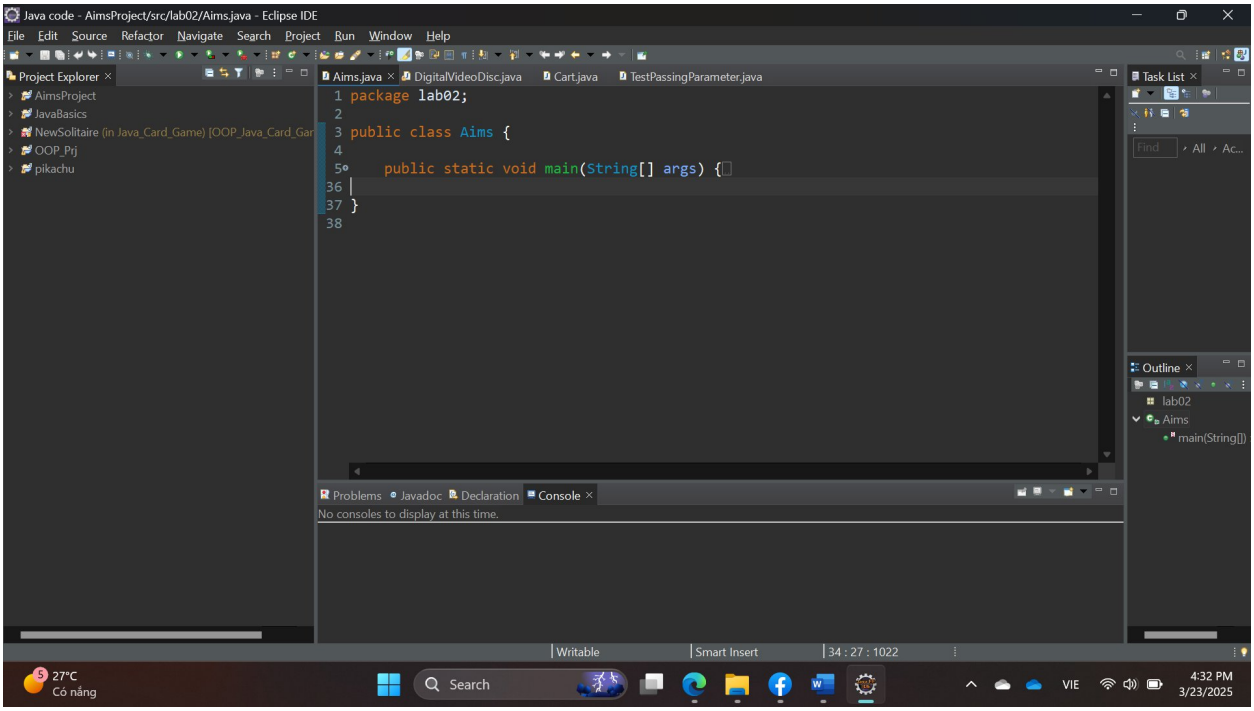
The system needs to create a new cart for the user, where it will keep information on the DVDs that the user wants to buy. The user can add, remove DVDs from the cart as well as calculate the cost. The user can add a maximum of 20 DVDs into one cart. The cart with its information and behaviors is modeled 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 DVD 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.



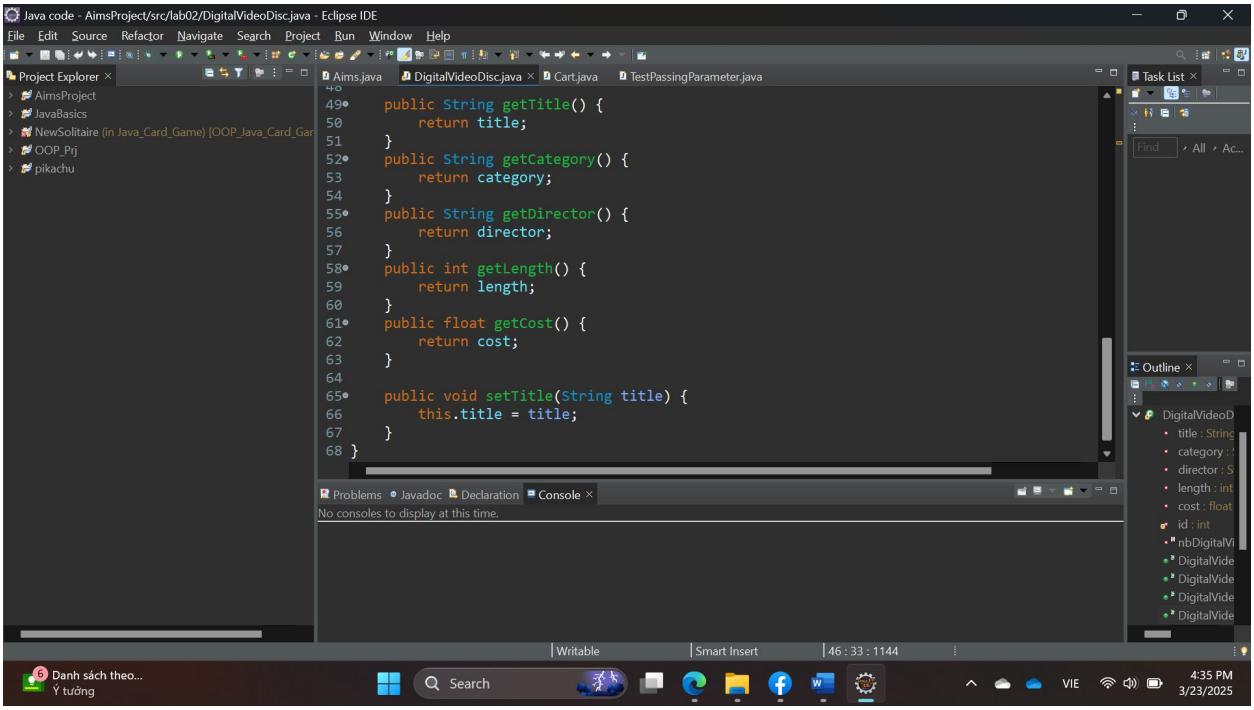
7. Create Aims class



8. Create the DigitalVideoDisc class and its attributes

```java
package lab02;

public class DigitalVideoDisc {
    private String title;
    private String category;
    private String director;
    private int length;
    private float cost;
    private int id;
    private static int nbDigitalVideoDisc = 0;
```

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

```java
    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;
    }
```

```java
    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;
    }
```

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.

```java
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");
    }
```

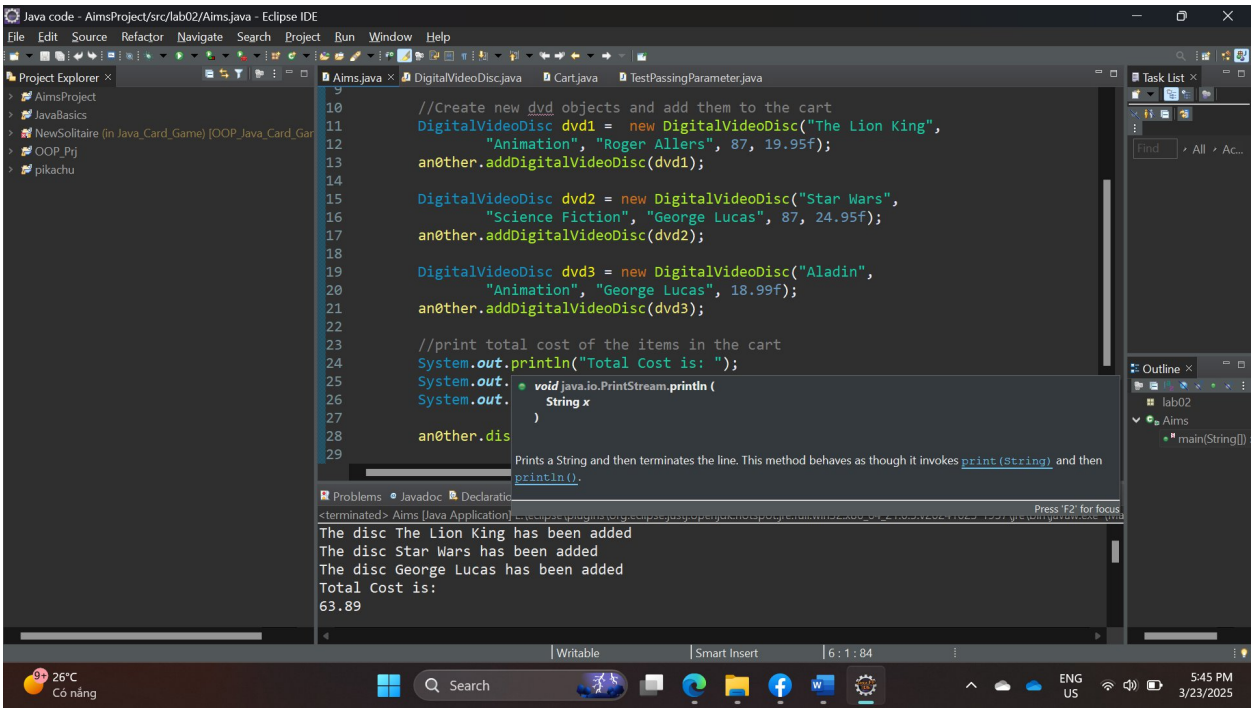
```java
public void removeDigitalVideoDisc(DigitalVideoDisc disc) {
    int idx;
    for(idx = 0; idx < qtyOrdered; idx++) {
        if(disc == itemsOrdered[idx]) {
            break;
        }
    }

    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.

```java
package lab02;

public class Aims {

    public static void main(String[] args) {

        //Create  a new cart
        Cart an0ther = 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);
        an0ther.addDigitalVideoDisc(dvd1);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
                "Science Fiction", "George Lucas", 87, 24.95f);
        an0ther.addDigitalVideoDisc(dvd2);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
                "Animation", "George Lucas", 18.99f);
        an0ther.addDigitalVideoDisc(dvd3);

        //print total cost of the items in the cart
        System.out.println("Total Cost is: ");
        System.out.println(an0ther.totalCost());
        System.out.print("\n");

        an0ther.display();
```

The result is:



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.

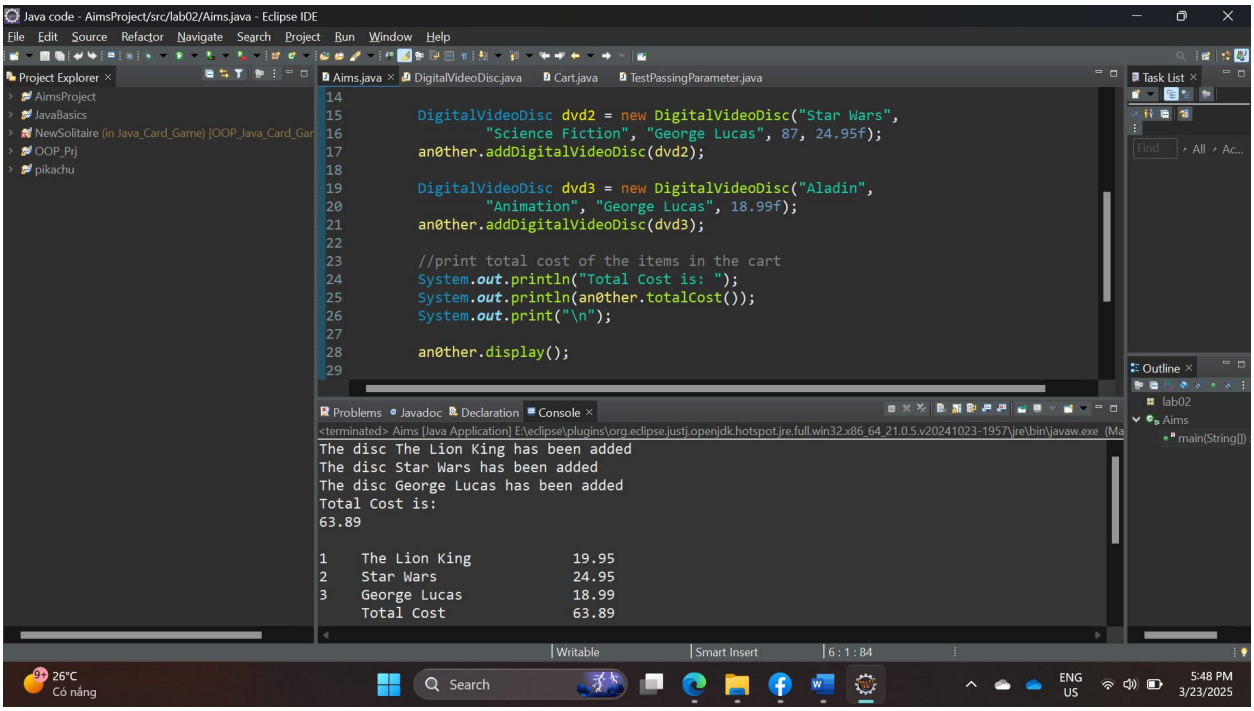
For example:

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:



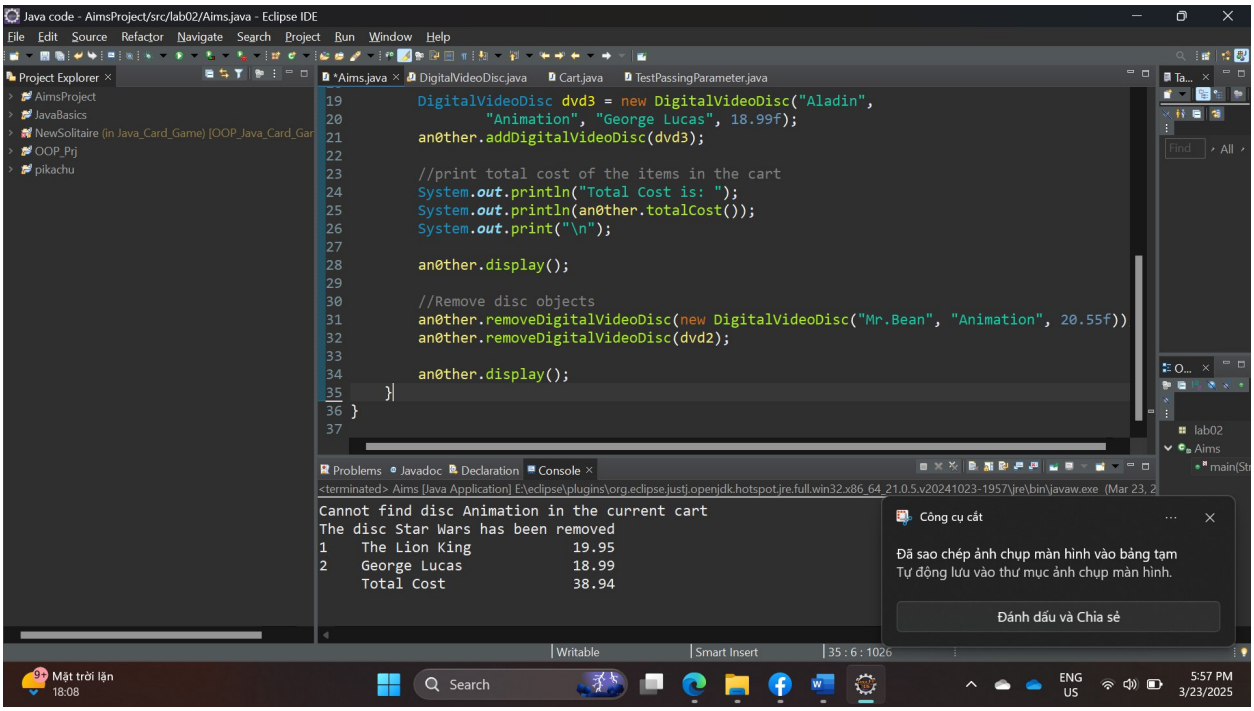
13. Removing items from the cart

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
        an0ther.removeDigitalVideoDisc(new DigitalVideoDisc("Mr.Bean", "Animation", 20.55f));
        an0ther.removeDigitalVideoDisc(dvd2);

        an0ther.display();
    }
}
```

And the result is:



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)`

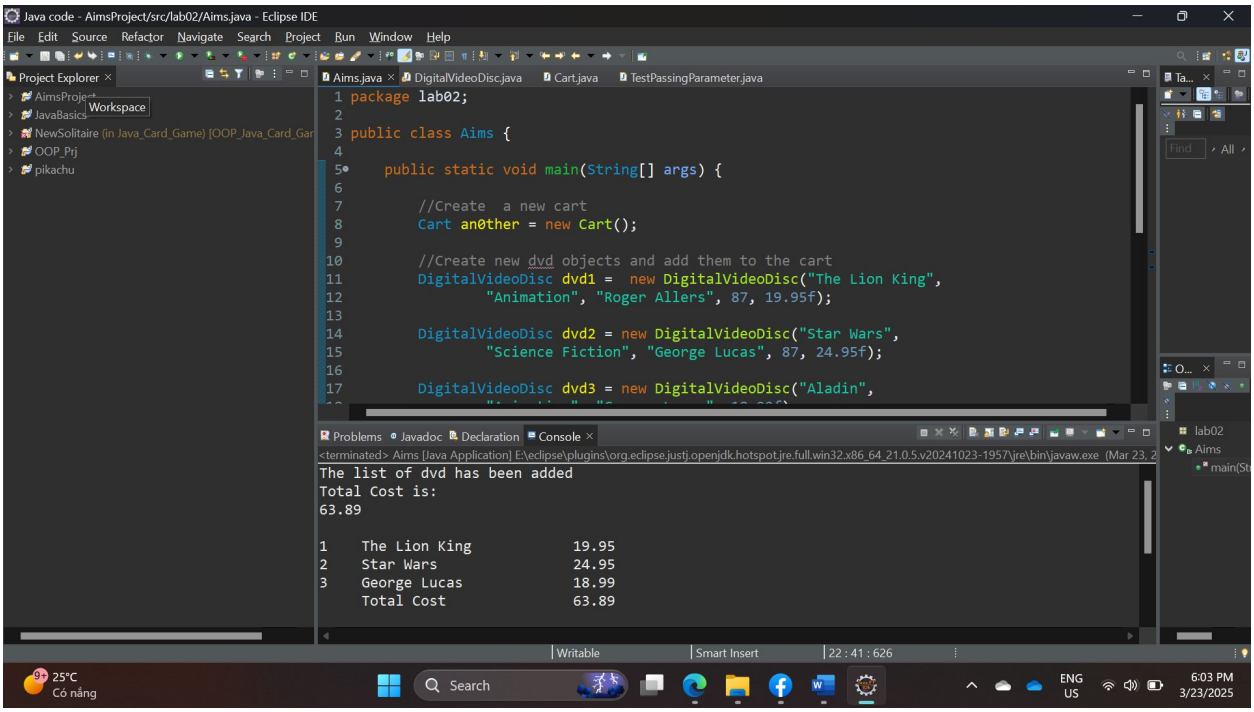
Source code:

```java
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:

```java
package lab02;

public class Aims {

    public static void main(String[] args) {

        //Create  a new cart
        Cart an0ther = 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);

        DigitalVideoDisc[] list = {dvd1, dvd2, dvd3};

        an0ther.addDigitalVideoDisc(list);

        //print total cost of the items in the cart
        System.out.println("Total Cost is: ");
        System.out.println(an0ther.totalCost());
        System.out.print("\n");

        an0ther.display();
```

The result is:



- 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?

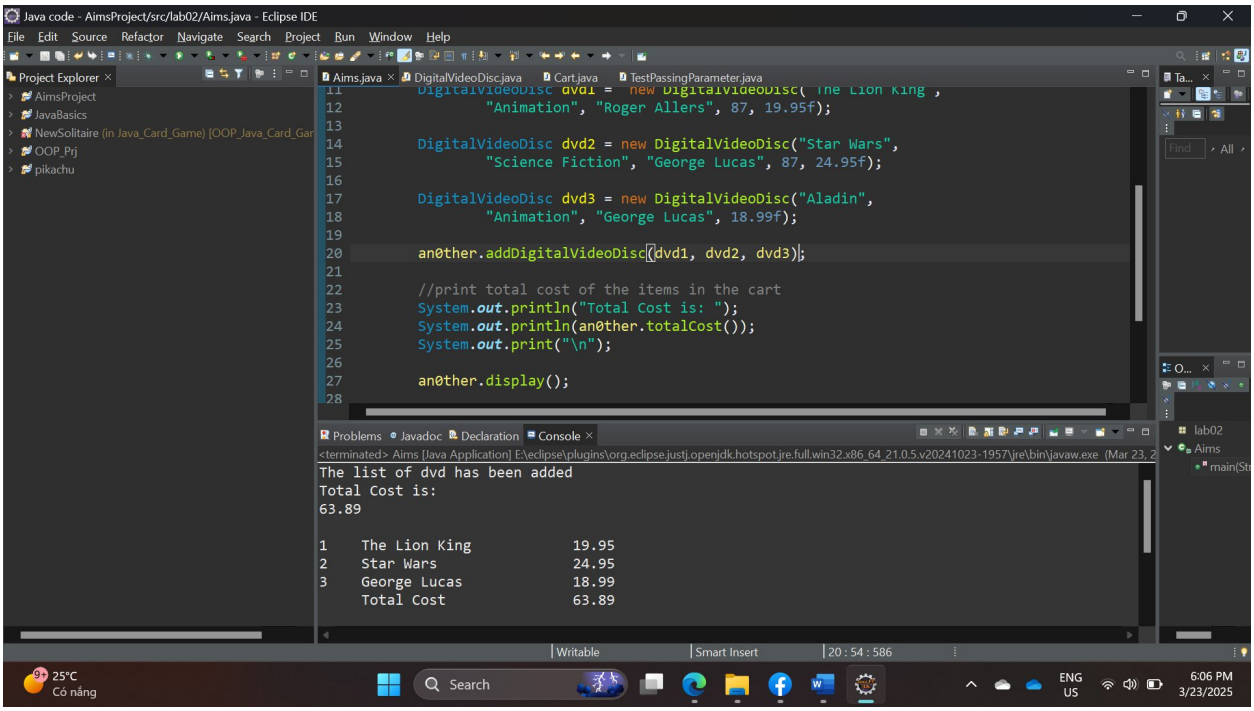
Source code:

```
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:

```java
package lab02;

public class Aims {

    public static void main(String[] args) {

        //Create  a new cart
        Cart an0ther = 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);

        an0ther.addDigitalVideoDisc(dvd1, dvd2, dvd3);

        //print total cost of the items in the cart
        System.out.println("Total Cost is: ");
        System.out.println(an0ther.totalCost());
        System.out.print("\n");

        an0ther.display();
```

The result is:



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

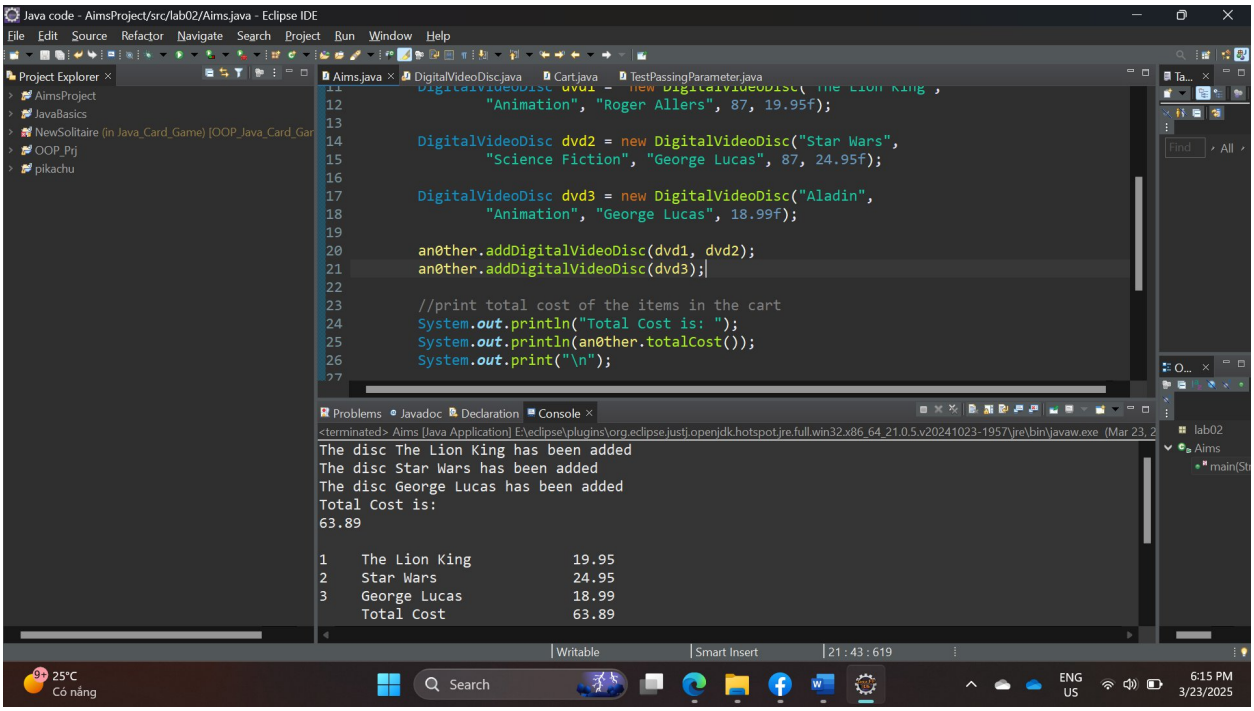
- Continuing focus on the **`Cart`** class
- Create new method named **`addDigitalVideoDisc`**
  - The signature of this method has two parameters as following:
    `addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2)`

```
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:

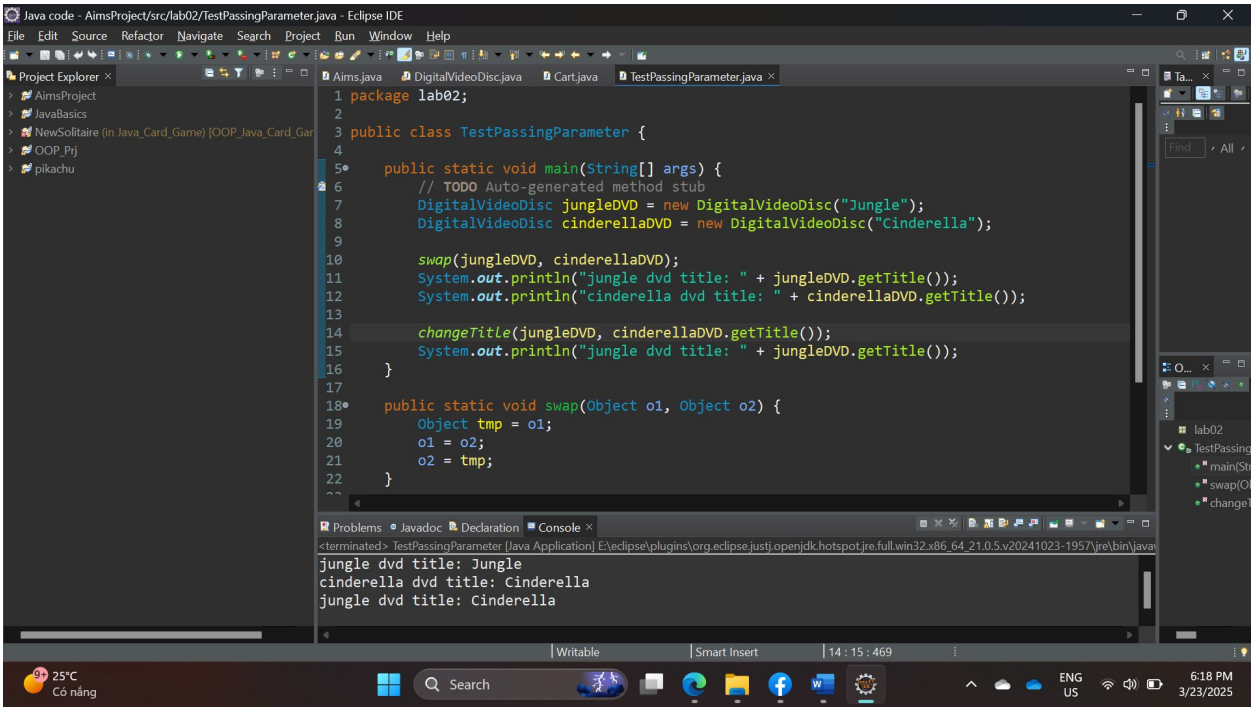
```
package lab02;

public class Aims {

    public static void main(String[] args) {

        //Create  a new cart
        Cart an0ther = 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);

        an0ther.addDigitalVideoDisc(dvd1, dvd2);
        an0ther.addDigitalVideoDisc(dvd3);

        //print total cost of the items in the cart
        System.out.println("Total Cost is: ");
        System.out.println(an0ther.totalCost());
        System.out.print("\n");

        an0ther.display();
```

The result is:



15. 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



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;
}
```