THỰC HÀNH LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ANSWER LAB02

Họ và tên: Phạm Công Sơn

MSSV: 20205220

6. Create the DigitalVideoDisc class and its attributes

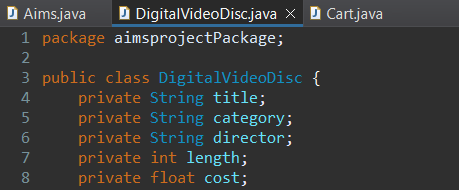


Figure 1: DigitalVideoDisc class

7. Create accessors and mutators for the class DigitalVideoDisc

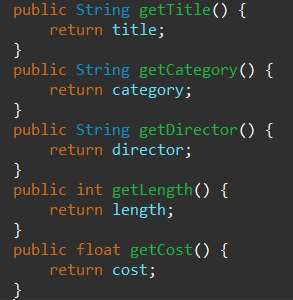


Figure 2: Generated accessors

8. Create Constructor method

**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?

**Answer:**

Yes, JAVA allows me to create a constructor method to build a DVD by title then create a constructor method to build a DVD by category. JAVA allows me to have multiple constructors with different parameter lists. This feature is called **constructor overloading**. I can create two or more constructors in a JAVA class as long as they each have different parameter lists.

For example, the task in *part 8: Create Constructor method* demonstrates creating 4 constructor methods for DigitalVideoDisc for different purposes:

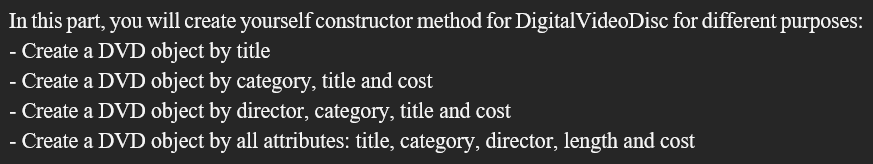


Figure 3: Task in Part 8. Create Constructor method

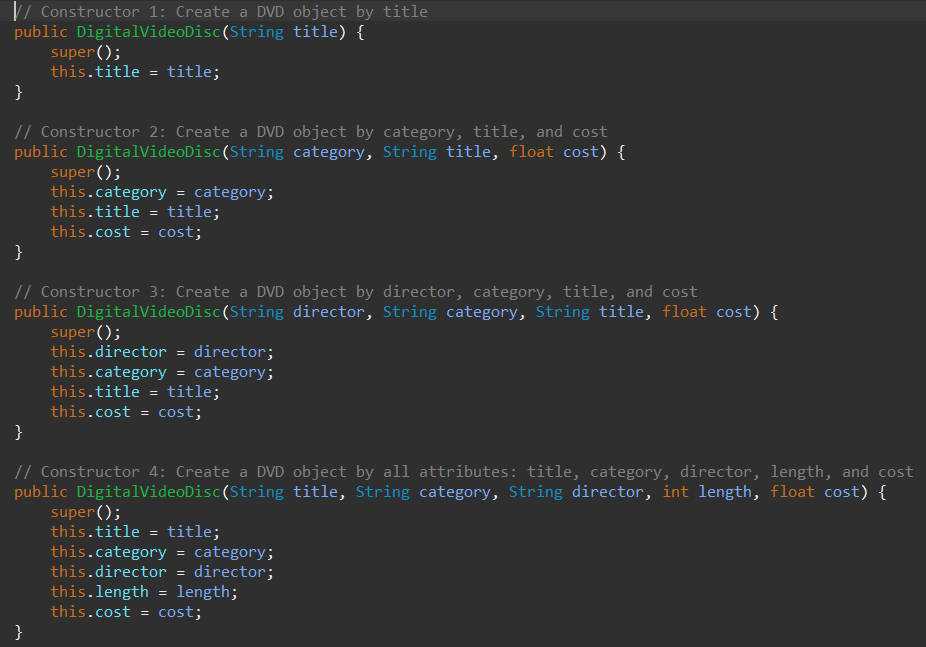


Figure 4: My work for the task

9. 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.

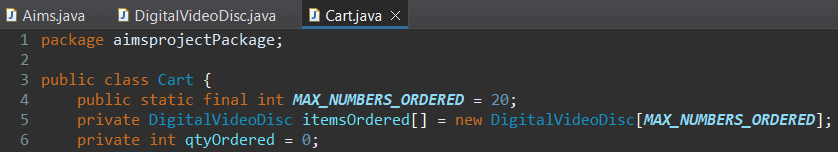


Figure 5: Cart class with Array to store list and qtyOrdered

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

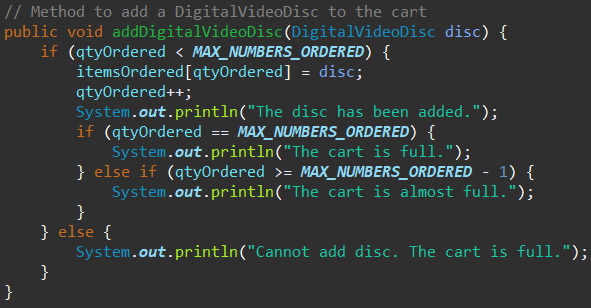


Figure 6: Method addDigitalVideoDisc

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

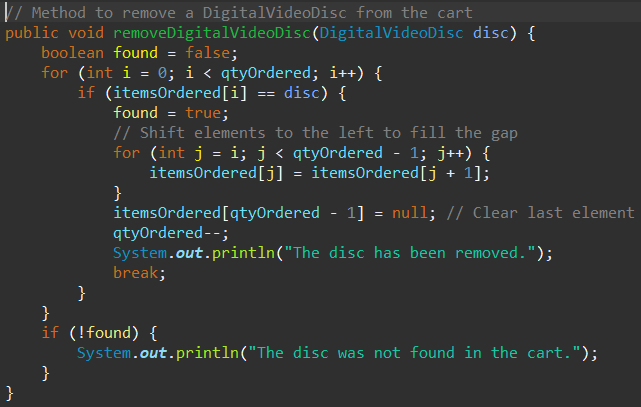


Figure 7: Method removeDigitalVideoDisc()

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.

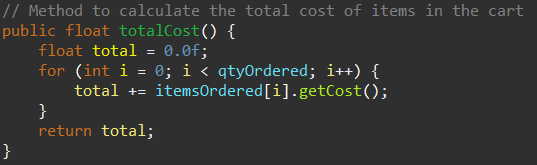


Figure 8: Method totalCost()

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

10. 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.

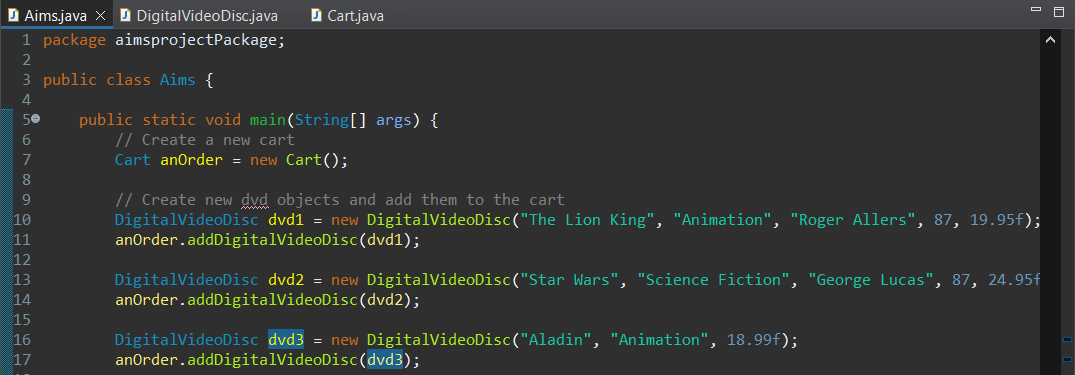


Figure 9: Aims class

The result:

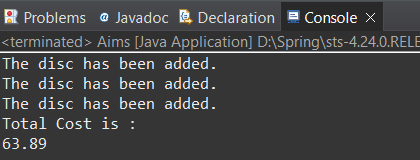


Figure 10: Result of addDigitalVideoDisc()

11. 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.

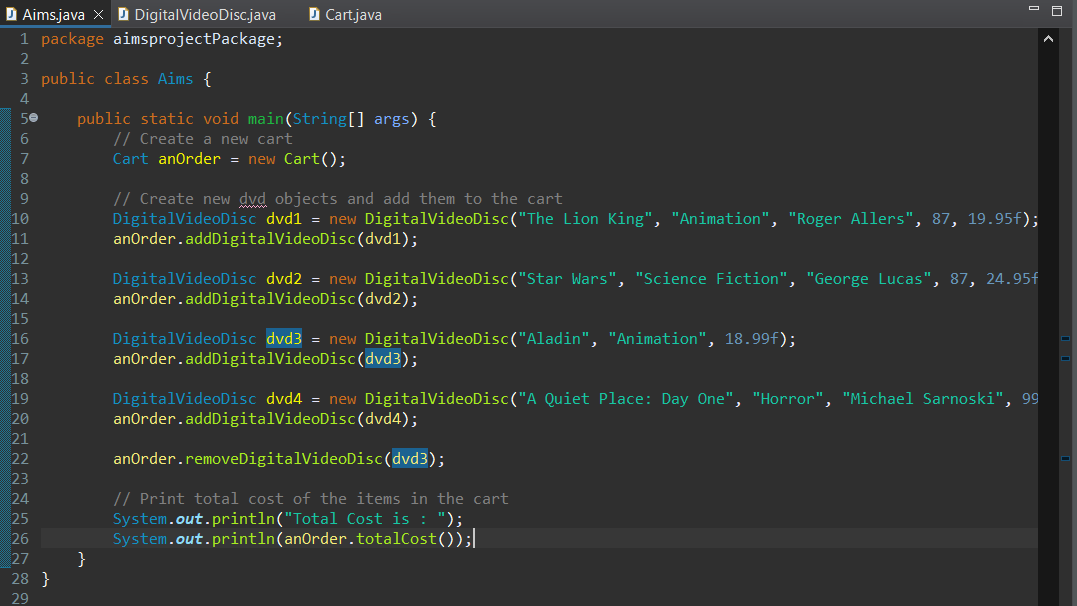


Figure 11: Added dvd4 and removeDigitalVideoDisc()

The result:

* Added dvd4 cost: 68.25f
* Remove dvd3 cost: 18.99f
* Total cost should be: 19.95+24.95+68.25=113.15

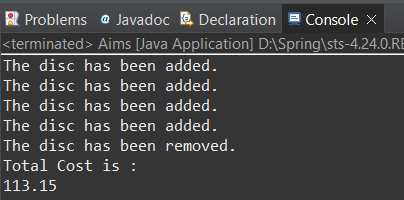


Figure 12: Result of removeDigitalVideoDisc()