**BÁO CÁO**

Object-Oriented Programming

Student:Hoàng Van Kiên, [kien.hv205089@soict.hust.edu.vn](mailto:kien.hv205089@soict.hust.edu.vn)

MSSV: 20205089

**Lab 02: Problem Modeling and Encapsulation**

# Use case diagram

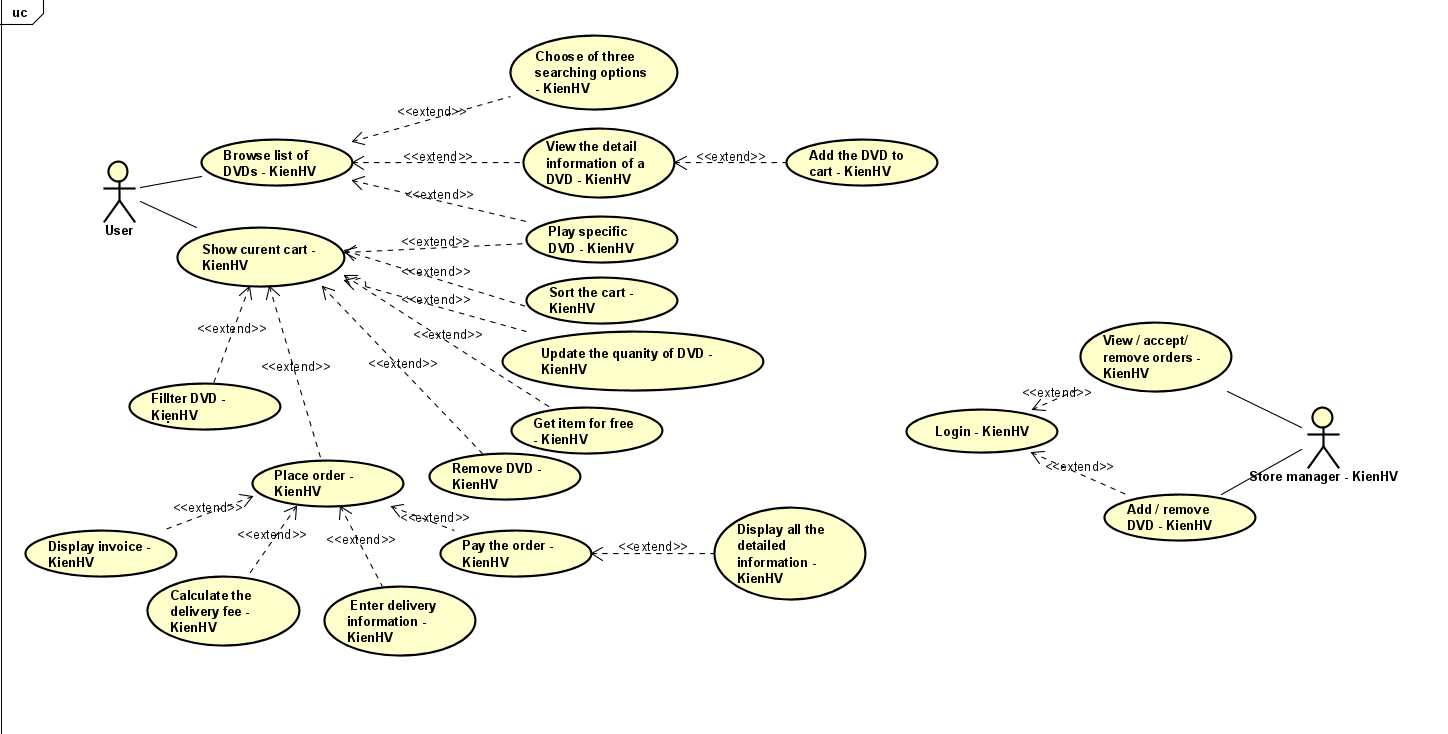


Figure 1. Use case diagram for AIMS Project.

# UML Class Diagram for use cases related to cart management

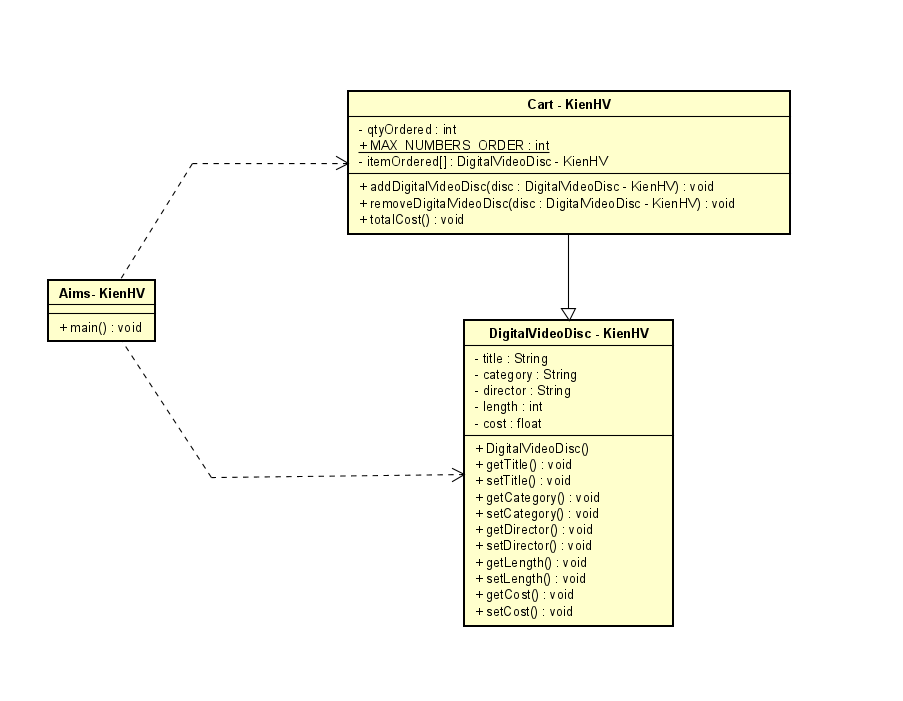


Figure 2. Class diagram for use cases related to cart management.

# Create Aims class

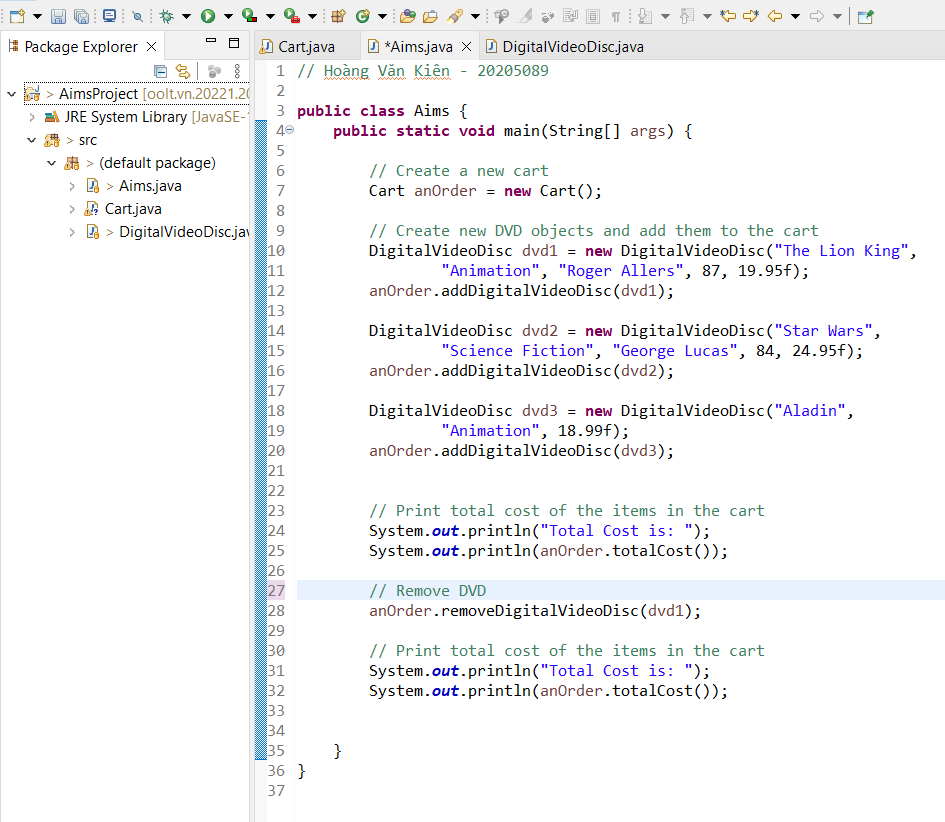
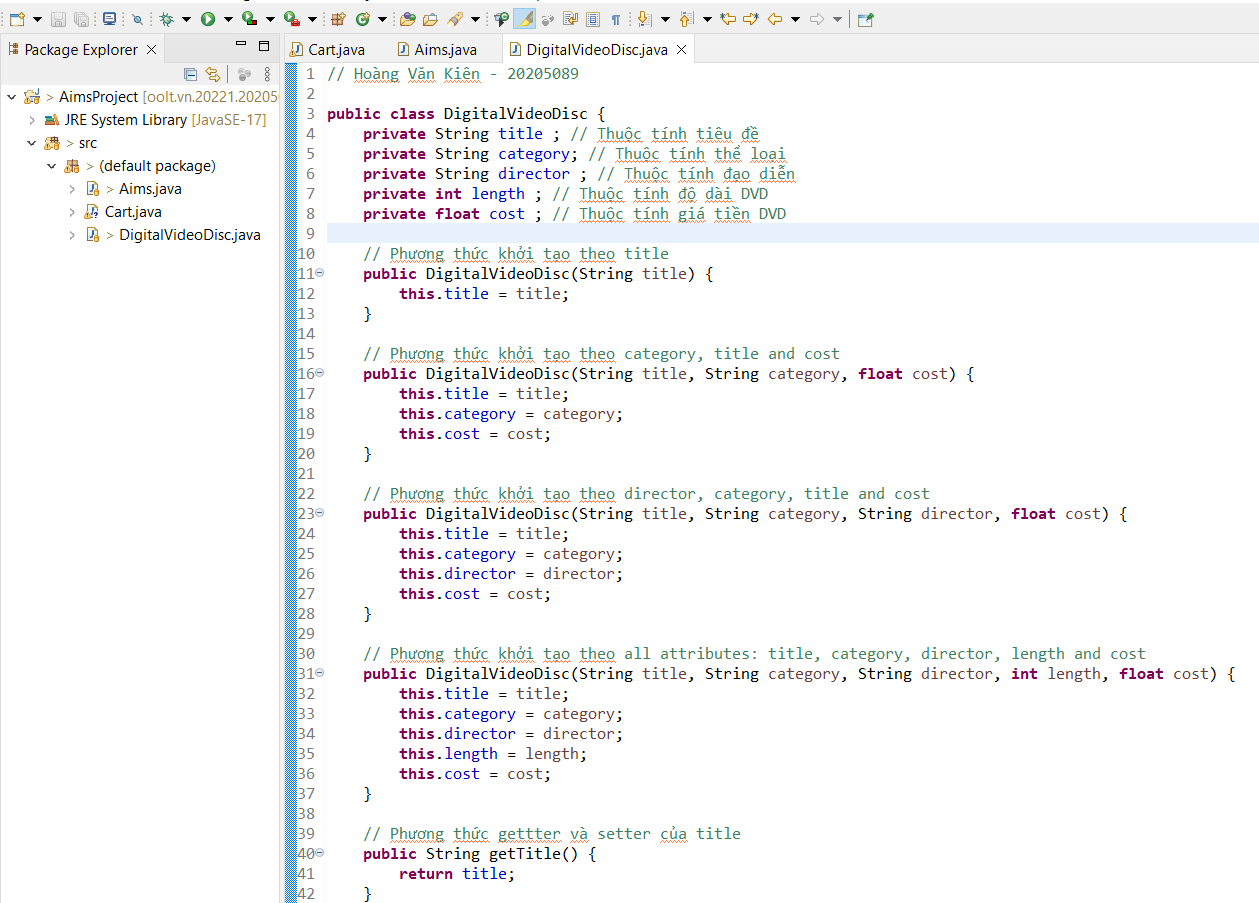


Figure 3. Aims class by Eclipse

# Create the DigitalVideoDisc class and its attributes



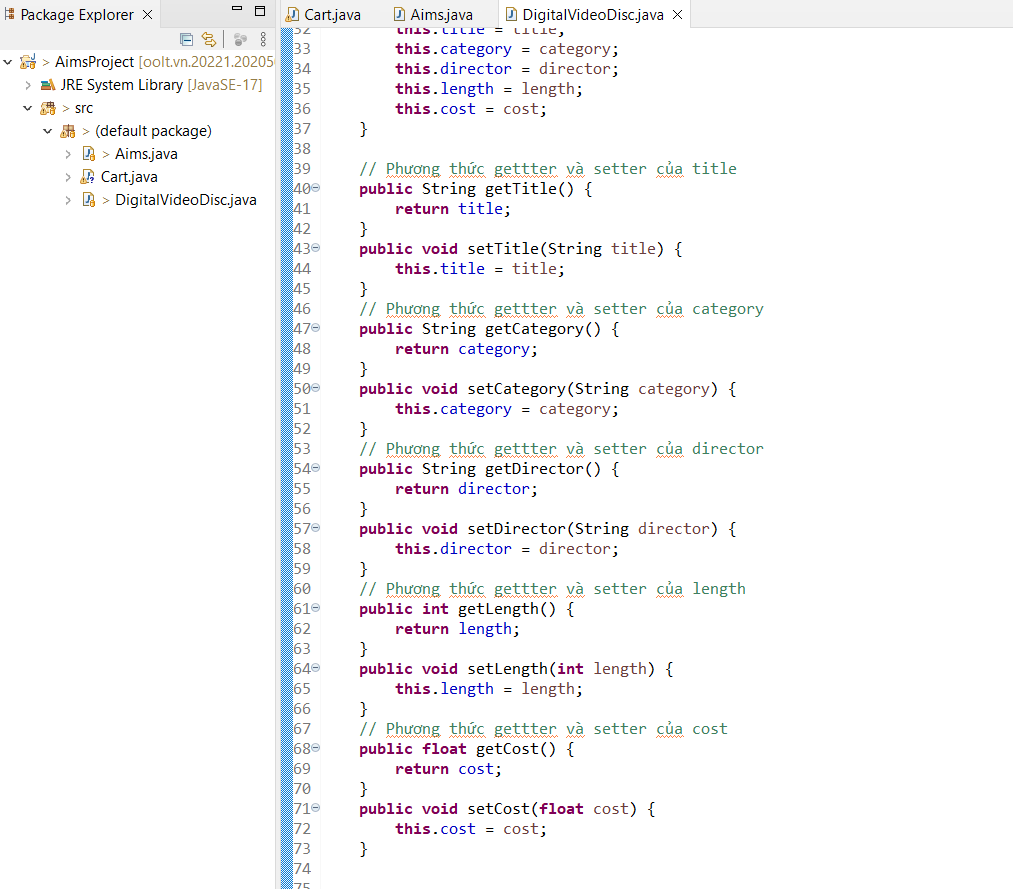


Figure 4. DigitalVideoDisc class

# Create accessors and mutators for the class DigitalVideoDisc

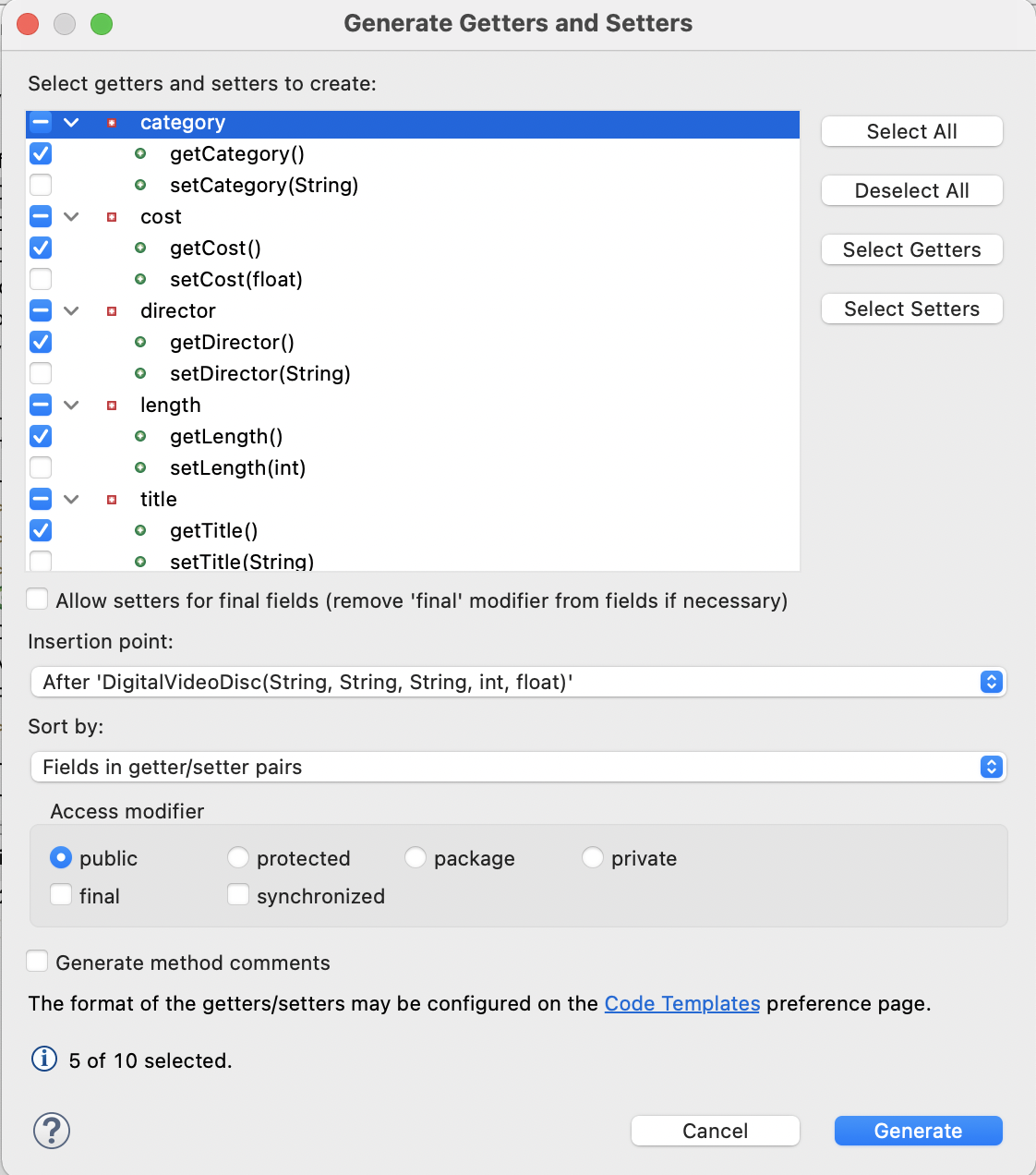
******

Figure 5. Generate getters & setters by Eclipse

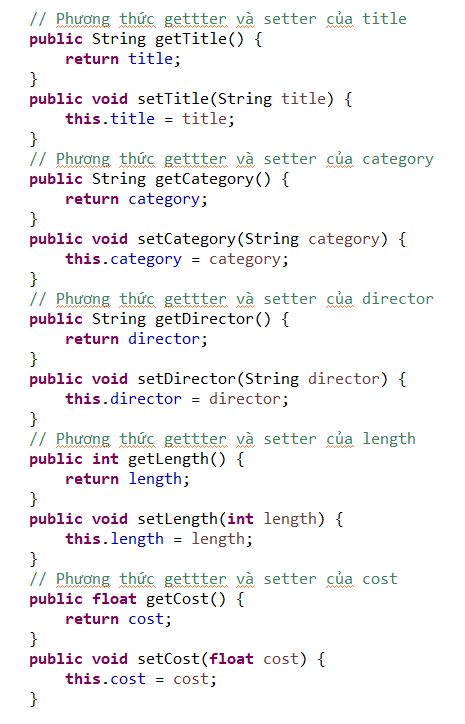
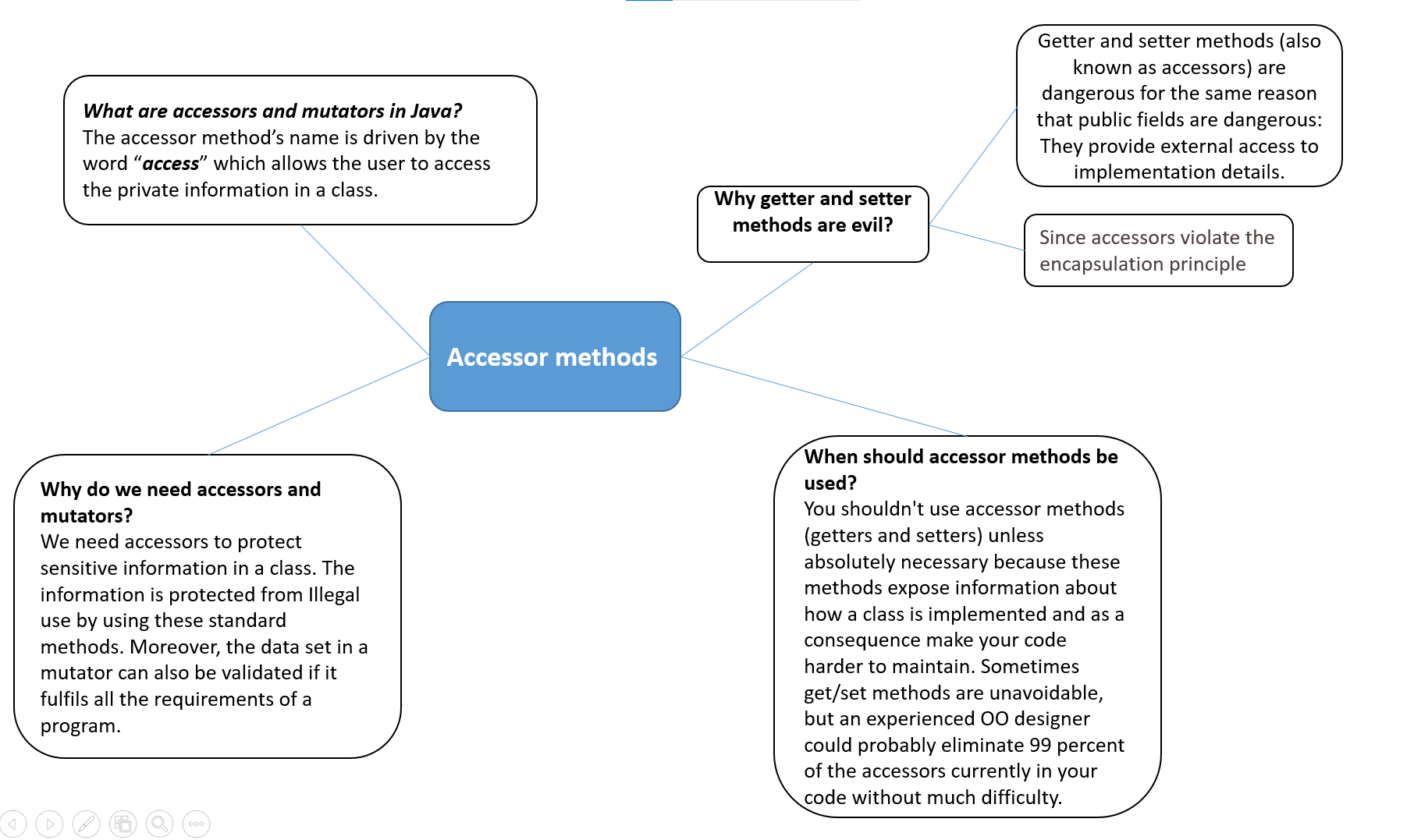
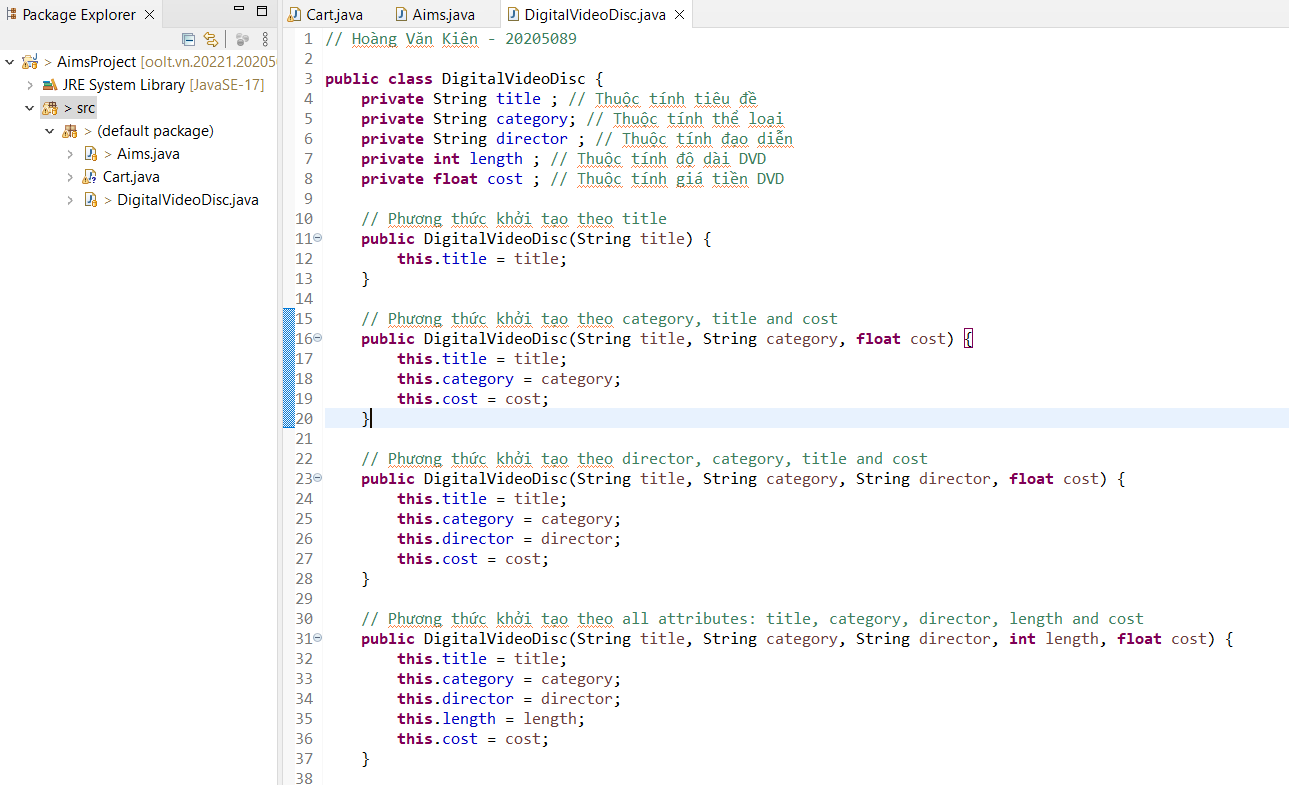


Figure 9. Generated accessors

**Reading Assignment:** When should accessor methods be used?



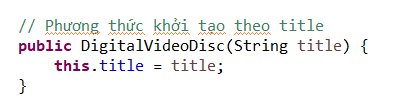
# Create Constructor method



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

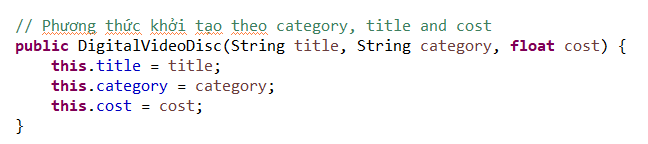
- Create a DVD constructor by title

The result is:



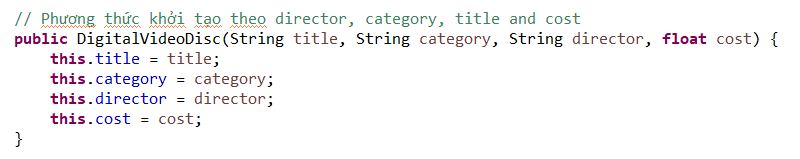
- Create a DVD constructor by category, title and cost

The result is:



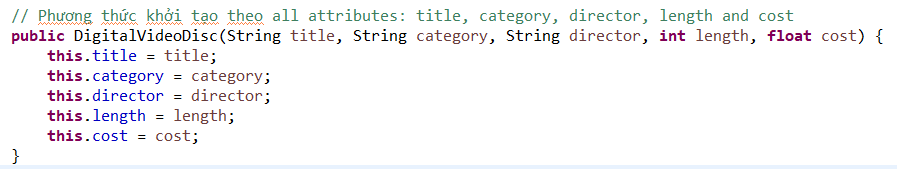
- Create a DVD constructor by director, category, title and cost

The result is:



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

The result is:



# Create the Cart class to work with DigitalVideoDisc

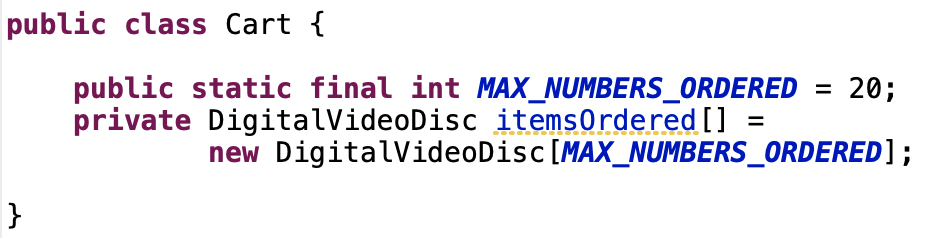
The **Cart** class will contain a list of **DigitalVideoDisc** objects and have methods capable of modifying the list.

Figure 13. Sample code of the Cart class

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.

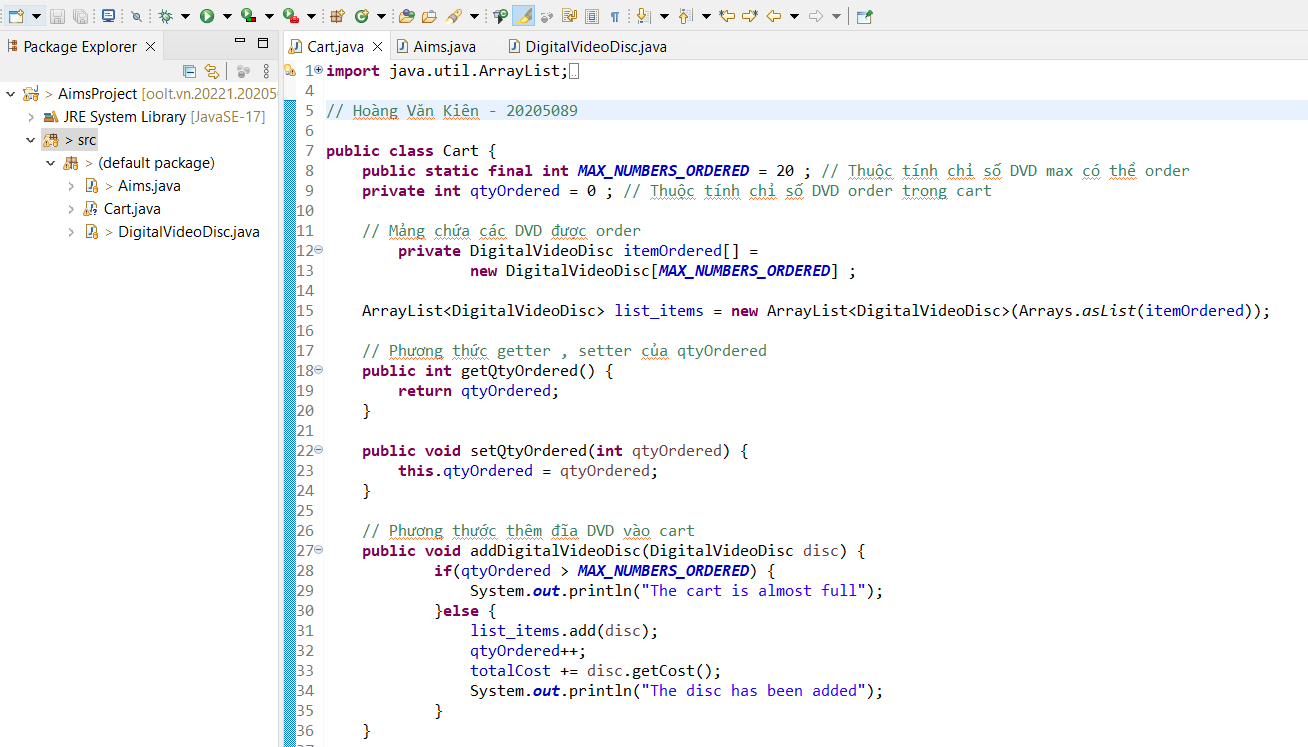
Create the method **addDigitalVideoDisc(DigitalVideoDisc disc)** to add an item to the list. You should check the current quantity to assure that the cart is not already full

* 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.



# 

# 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.

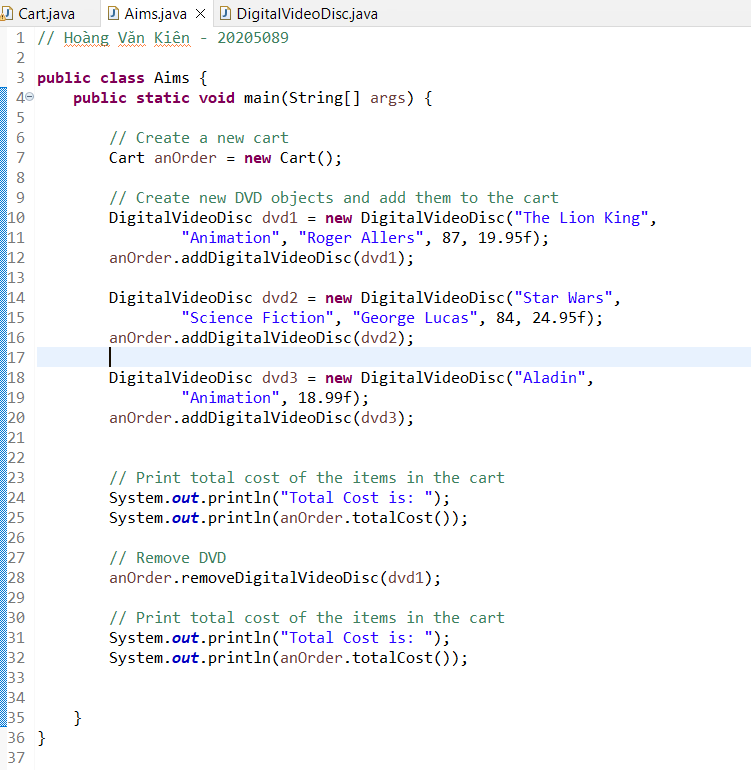
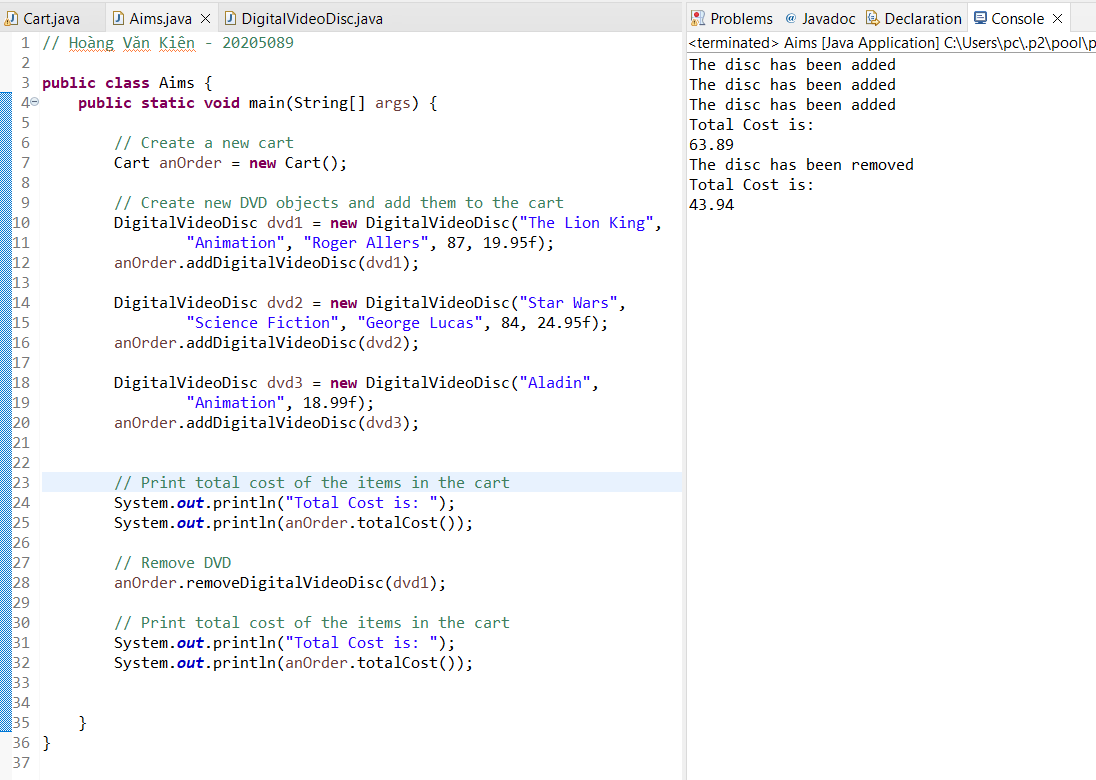


Figure 10. Code of the Aims class

**The result should be:**

****

# 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.

