BÁO CÁO

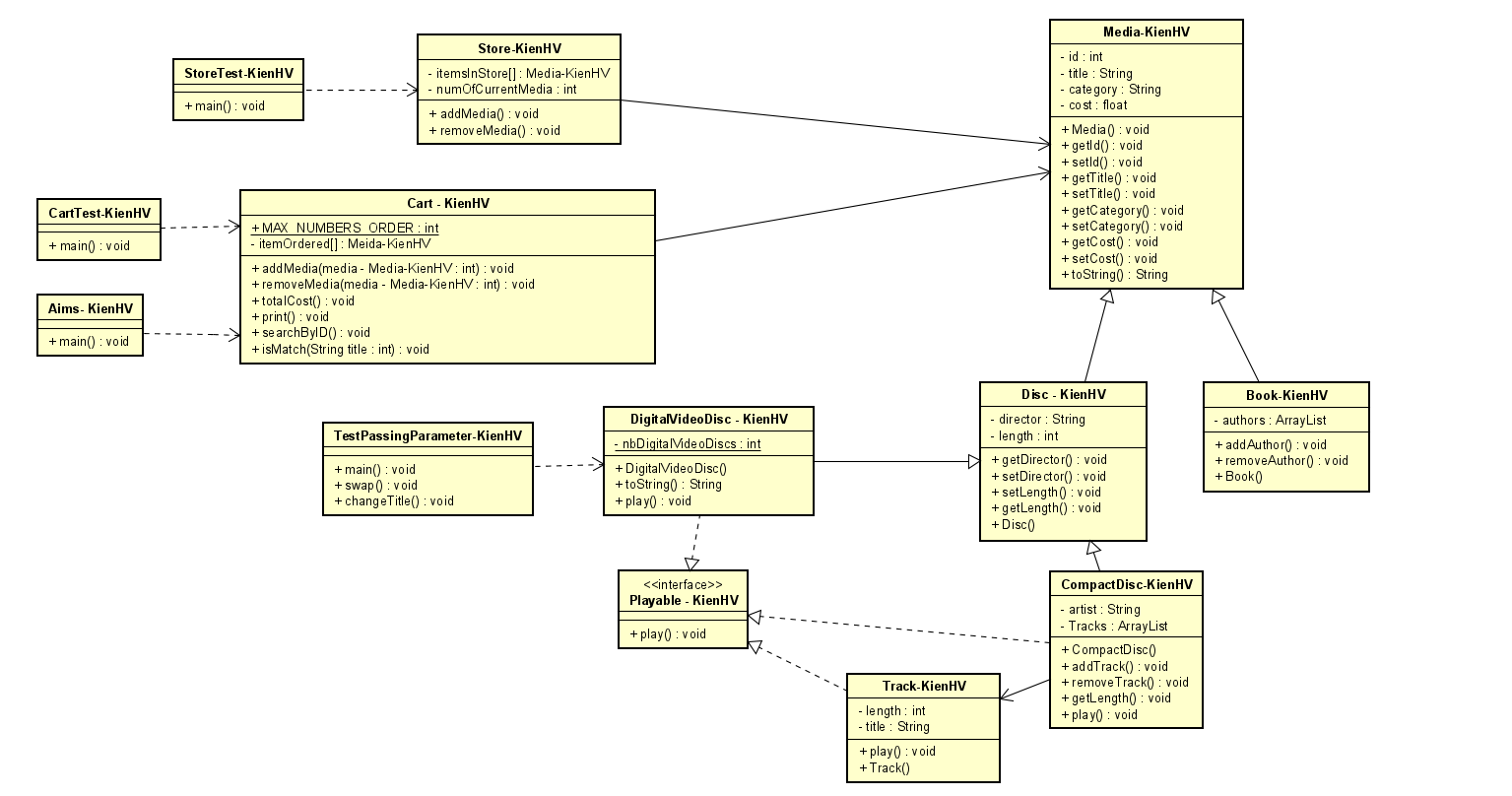
Object-Oriented Programming

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

MSSV: 20205089

**Lab 04: Inheritance and Polymorphism**

# UML Class Diagram updated



# Additional requirements of AIMS

Starting from this lab, you extend the AIMS system that you created in the previous exercises to allow customer to order 2 new types of media: books and CD.

A book’s information includes: id, title, category, cost and list of authors.

A CD’s information includes: id, title, category, artist, director, track list and price. Additionally, each track is unique in a CD with its own title and length. The length of a CD is sum of the lengths of its tracks.

When a user sees the details of a media in the store, the information displayed depends on the type of media.

* For books, the system shows their title, category, author list, the content length (i.e., the number of tokens).
* For CDs, the system displays the CD’s information (i.e. CD title, category, artist, director, CD length, and the cost for the CD) and then displays the information of all the tracks in that CD.
* For DVDs, the system displays the DVD’s information (i.e. DVD title, category, director, DVD length, and the cost for the DVD).

Additionally, the user can choose to play some media when browsing the list of media in the store or seeing the current cart. For simplicity, we establish the way the system plays a media is as follows: When a CD is played, the system displays the CD information (i.e., CD title and CD length) and plays all the tracks of the CD. To play a track, the system displays the track’s name and its length. Similarly, a DVD can also be played, i.e., the system displays the title and length of the DVD. If a DVD or track has the length 0 or less, the system must notify the user that the track, the DVD or the CD of that track cannot be played.

# Creating the **Book** class

- In the Package Explorer view, create New -> Class.

* Package: **hust.soict.dsai.aims.media**
* Name: **Book**
* Access modifier: **public**
* Superclass: **java**.**lang**.**Object**
* **public** **static** **void** **main**(**String**[] **args**): **do not check**
* Constructors from Superclass: **Check**
* All other boxes: **Do not check**

**Add fields to the Book class**

* To store the information about a **Book**, the class requires five fields: an **int** field **id,** **String** fields **title** and **category**, a **float** field **cost** and an **ArrayList** of **authors**. You will want to make these fields private, with public accessor methods for all but the **authors** field.

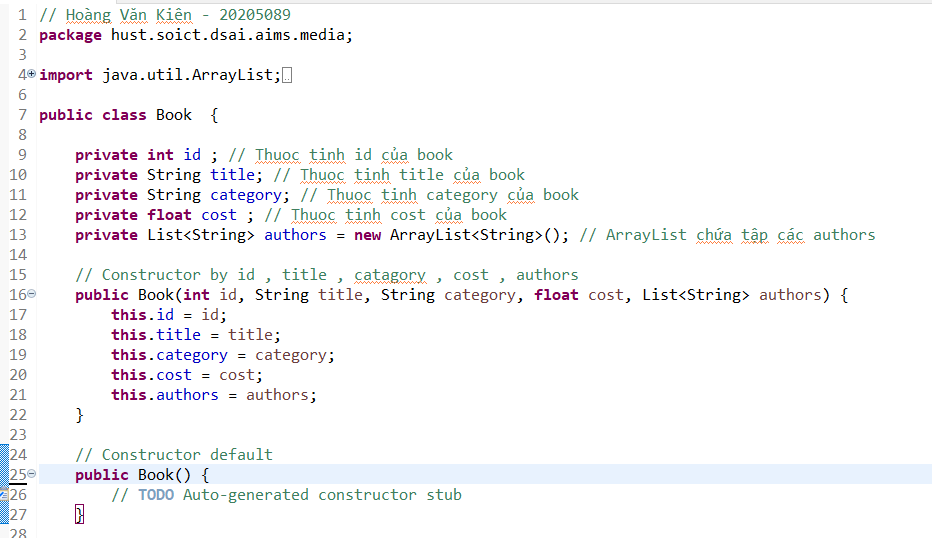
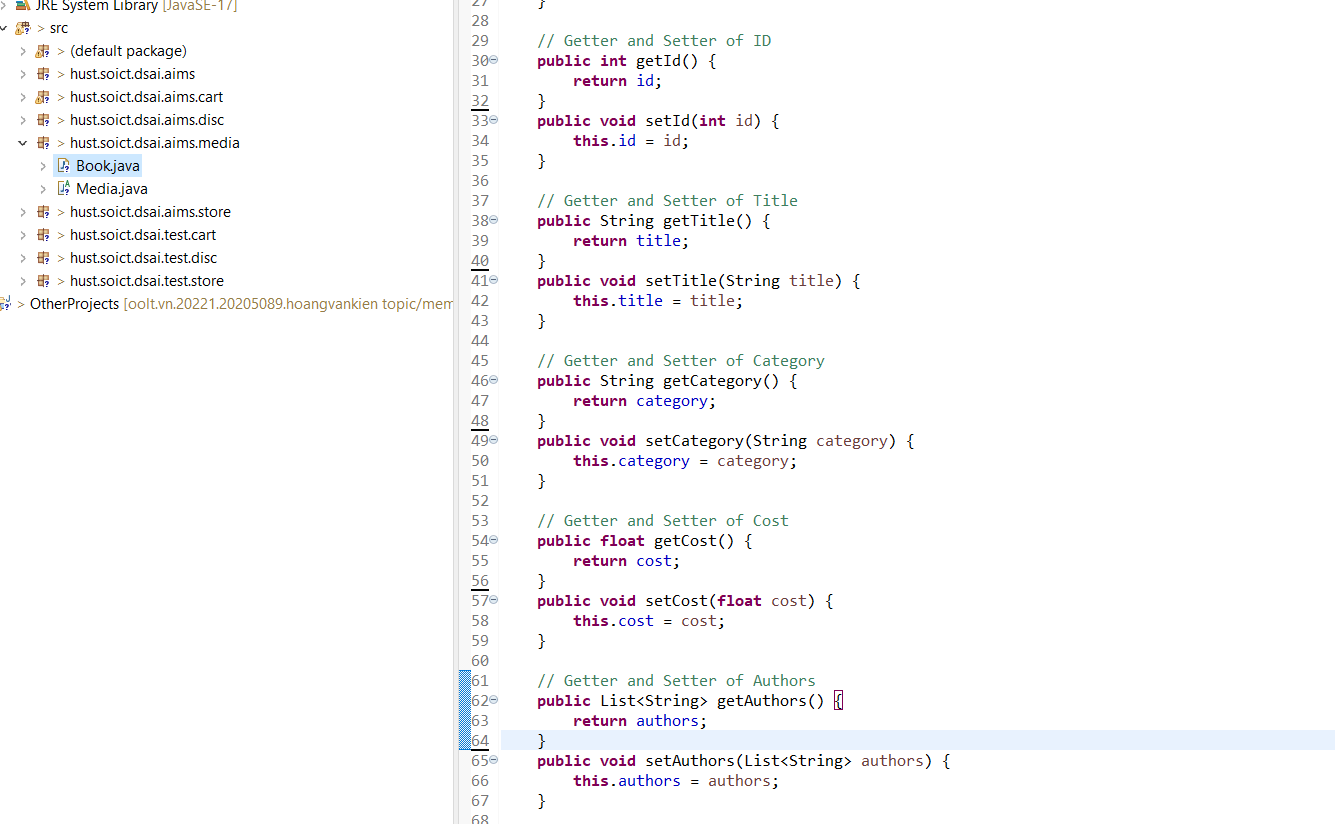
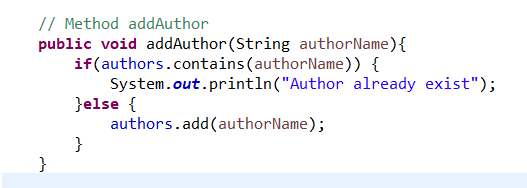


Figure 2. Adding fields to Book class

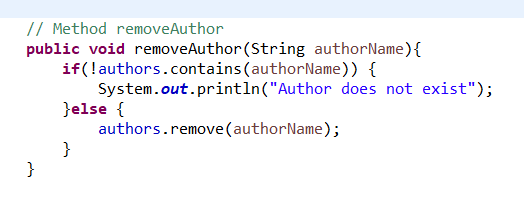
* **Generate Getter and Setter** of the Book



* Create **addAuthor**(**String** **authorName**) and **removeAuthor**(**String** **authorName**) for the **Book** class
  + The **addAuthor**(...) method should ensure that the author is not already in the **ArrayList** before adding



* + The **removeAuthor**(...) method should ensure that the author is present in the **ArrayList** before removing



# Creating the abstract **Media** class

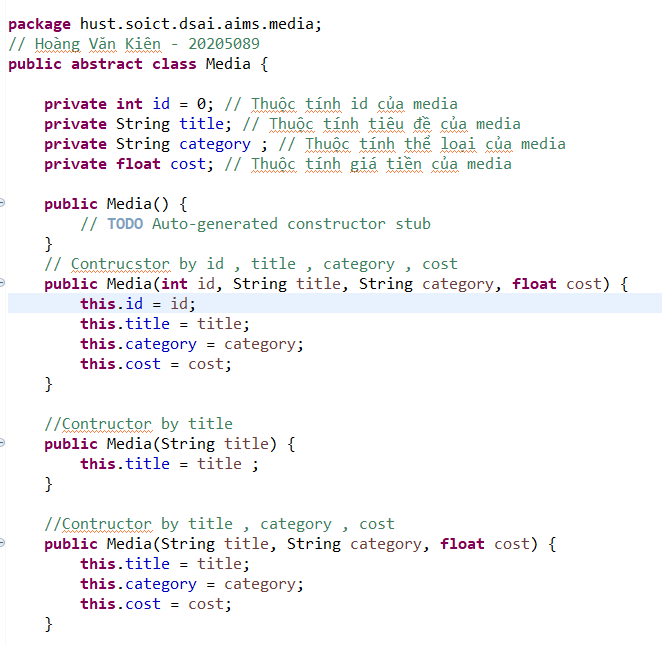
**Create the Media class in the project**

- In the **Package Explorer** view, right click to the project and select New -> Class. Adhere to the following specifications for the new class:

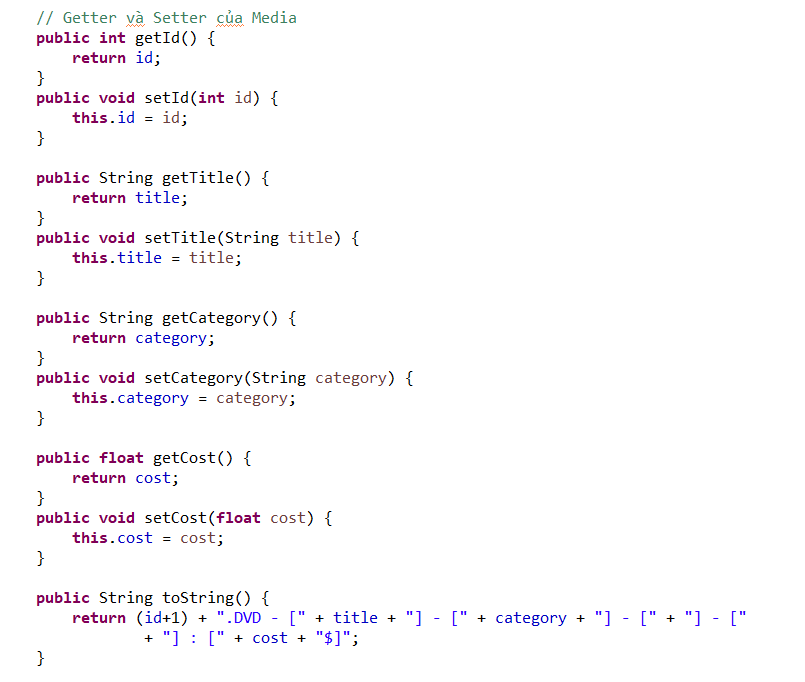
* Package: **hust.soict.dsai.aims.media**
* Name: **Media**
* Access Modifier: **public**, **abstract**
* Superclass: **java**.**lang**.**Object**
* Constructors from Superclass: Check
* **public static void main (String[] args)**: do not check
* All other boxes: Do not check

- Add fields to the **Media** class

* To store the information common to the **DigitalVideoDisc** and the **Book** classes, the **Media** class requires four private fields: **int id**, **String** **title**, **String** **category** and **float** **cost**



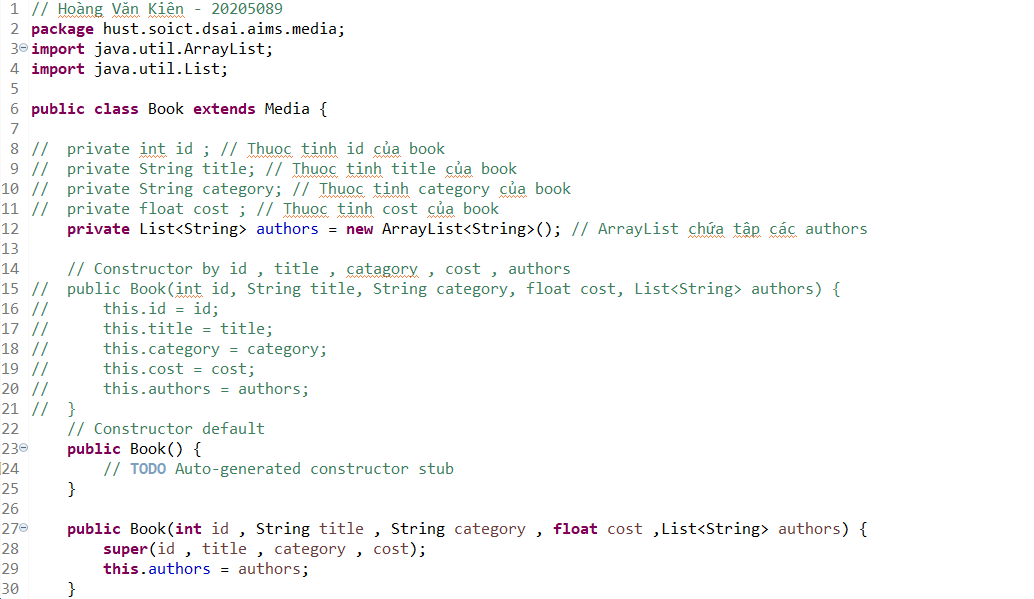
* Make public accessor methods for these fields (by using **Generate Getter and Setter** option in the **Outline** view pop-up menu)



- Remove fields and methods from **Book** and **DigitalVideoDisc** classes

* Open the Book.java in the editor
* Locate the Outline view on the right-hand side
* Select the fields id, title, category, cost and their accessors & mutators (if exist)
* Right click the selection and select Delete from the pop-up menu
* Save your changes

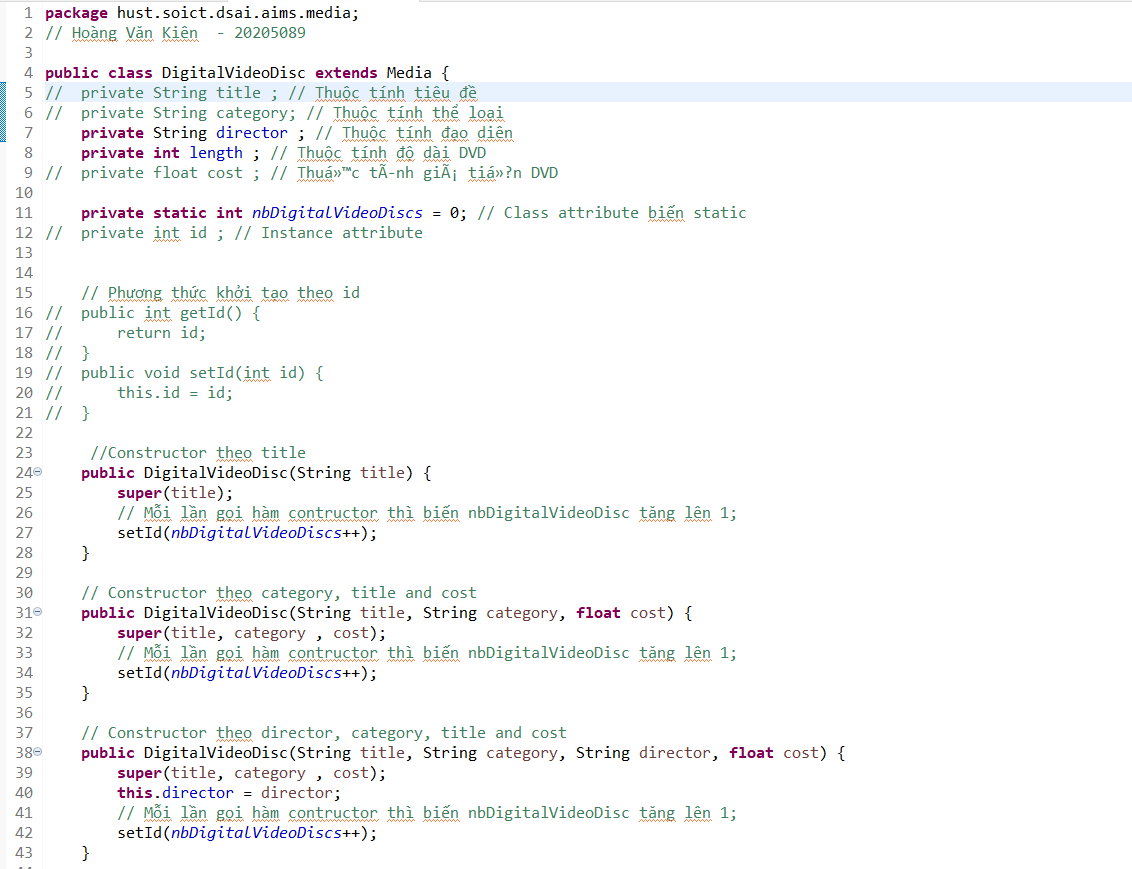
**RESULT:**

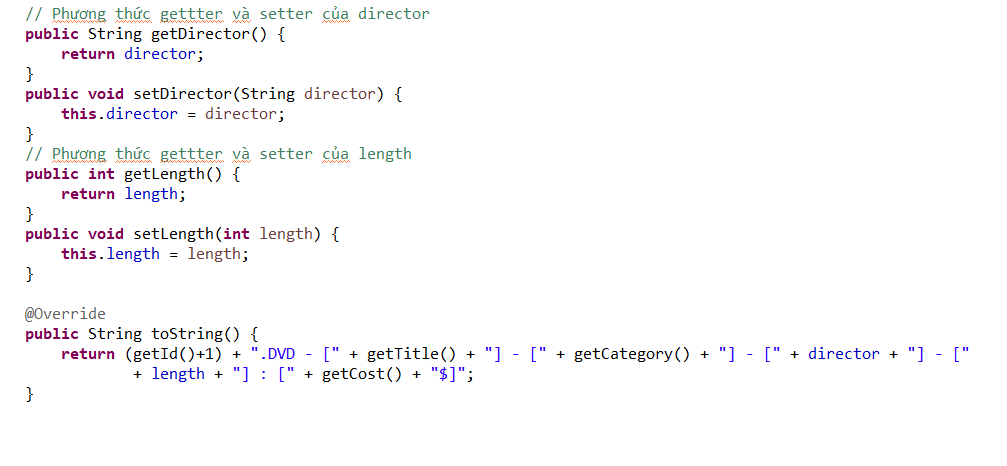




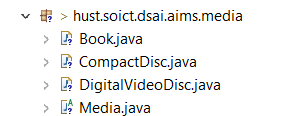
- The **DigitalVideoDisc** class and move it to the package **hust.soict.dsai.aims.media**.

**RESULT:**





Remove the package **hust.soict.dsai.aims.disc.**



* Extend the **Media** class for both **Book** and **DigitalVideoDisc**
  + **public class Book extends Media**

****

* + **public class DigitalVideoDisc extends Media**

****

# Creating the **CompactDisc** class

As with **DigitalVideoDisc** and **Book**, the **CompactDisc** class will extend **Media**, inheriting the **id, title**, **category** and **cost** fields and the associated methods.

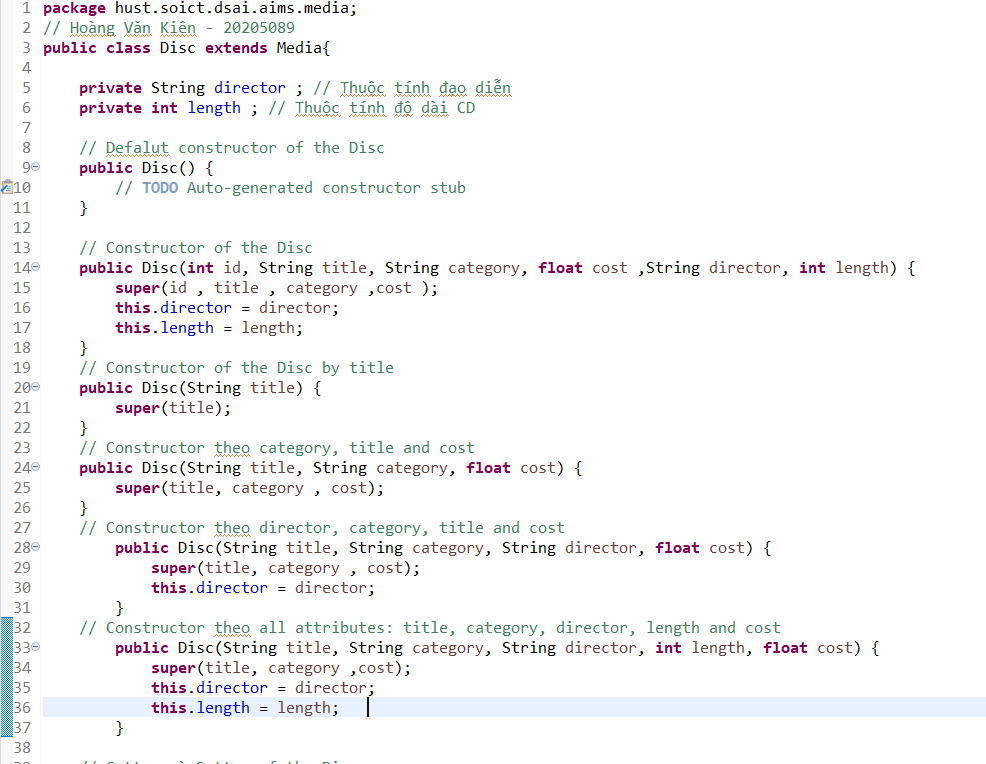
## 4.1. Create the **Disc** class extending the **Media** class

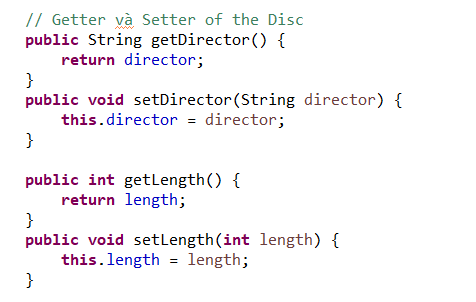
- The **Disc** class has two fields: **length** and **director**

- Create **getter** methods for these fields

- Create constructor(s) for this class. Use super() if possible.

**RESULT:**





- Make the **DigitalVideoDisc** extending the **Disc** class.



- Create the **CompactDisc** extending the **Disc** class.

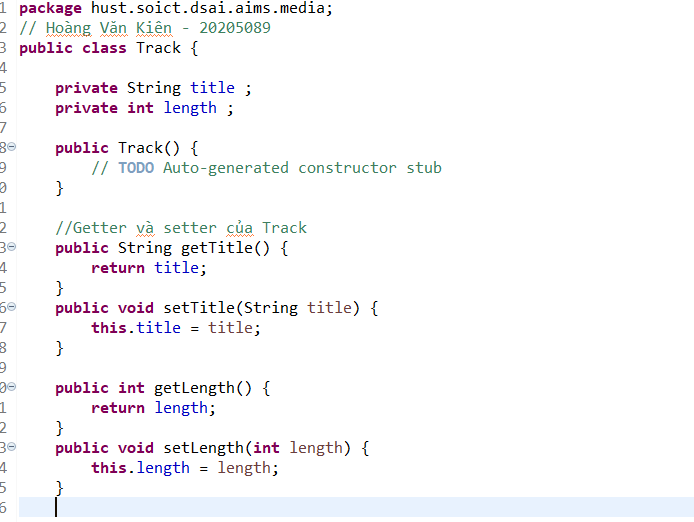


## 4.2. Create the **Track** class which models a track on a compact disc and will store information incuding the **title** and **length** of the track

- Add two fields: **String** **title** and **int** **length**

- Make these fields **private** and create their **getter** methods as **public**

- Create constructor(s) for this class.



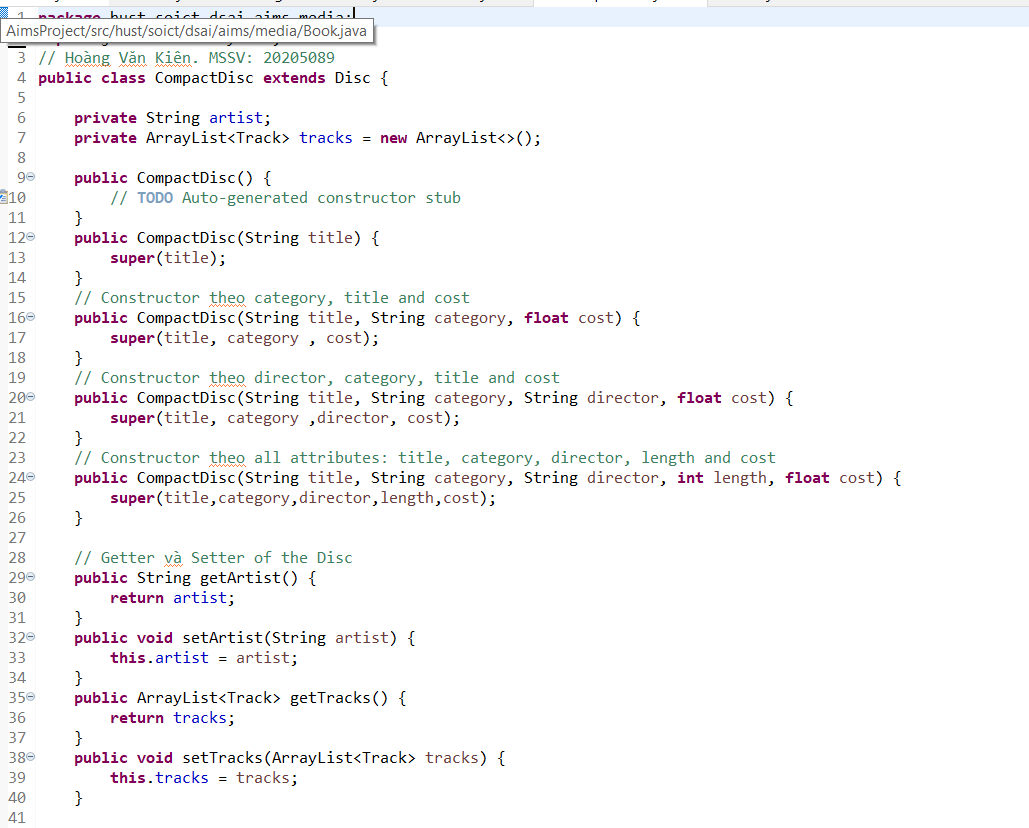
## 4.3. Open the **CompactDisc** class

- Add 2 fields to this class:

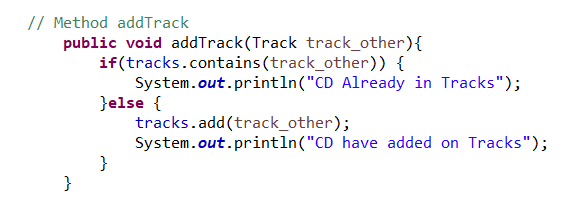
* a **String** as **artist**
* an **ArrayList** of **Track** as **tracks**

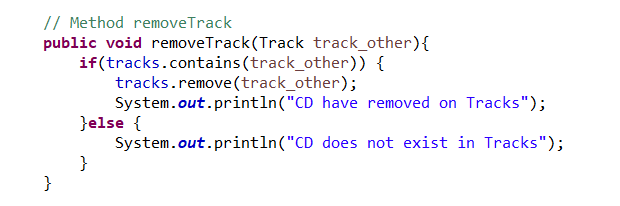
- Make all these fields as **private.** Create public **getter** method for only **artist**.

- Create constructor(s) for this class. Use super() if possible.



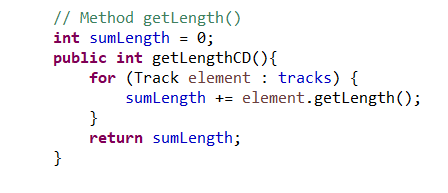
- Create methods **addTrack**() and **removeTrack**()

* The **addTrack**() method should check if the input track is already in the list of tracks and inform users
* 
* The **removeTrack**() method should check if the input track existed in the list of tracks and inform users



- Create the **getLength**() method

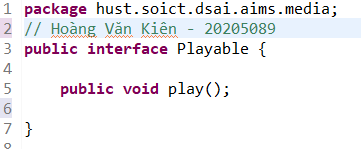
* Because each track in the CD has a length, the length of the CD should be the sum of lengths of all its tracks.



# Create the Playable interface

The **Playable** interface is created to allow classes to indicate that they implement a **play**() method.

- Create **Playable** interface, and add to it the method prototype: **public void play();**

****

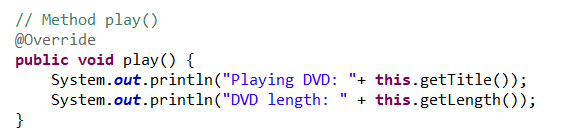
- Implement the **Playable** with **CompactDisc**, **DigitalVideoDisc** and **Track**





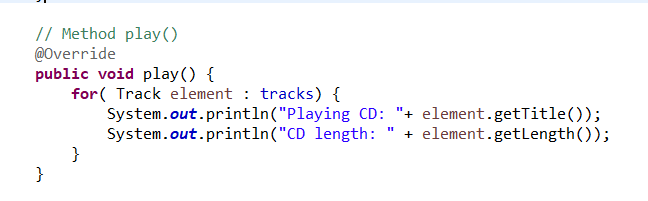


- Implement **play**() for **DigitalVideoDisc** and **Track**



- Implement **play**() for **CompactDisc**

* Since the **CompactDisc** class contains a **ArrayList** of **Tracks**, each of which can be played on its own. The **play**() method should output some information about the **CompactDisc** to console
* Loop through each track of the arraylist and call **Track's** play() method

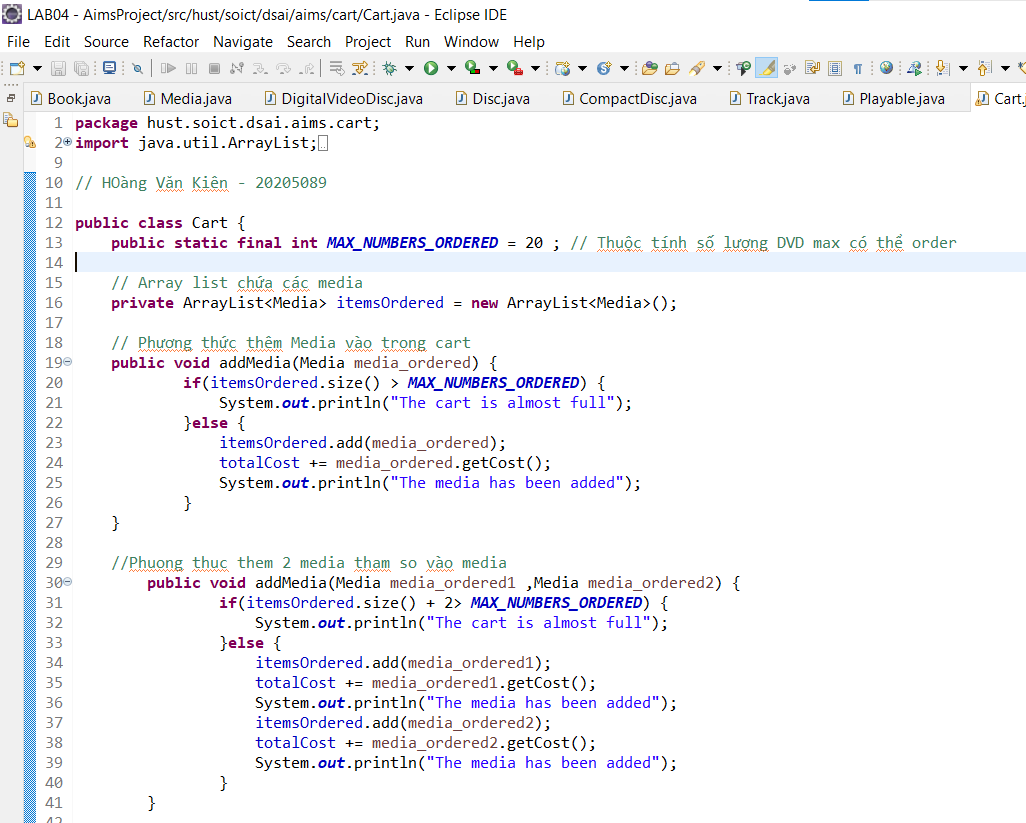


# Update the **Cart** class to work with **Media**

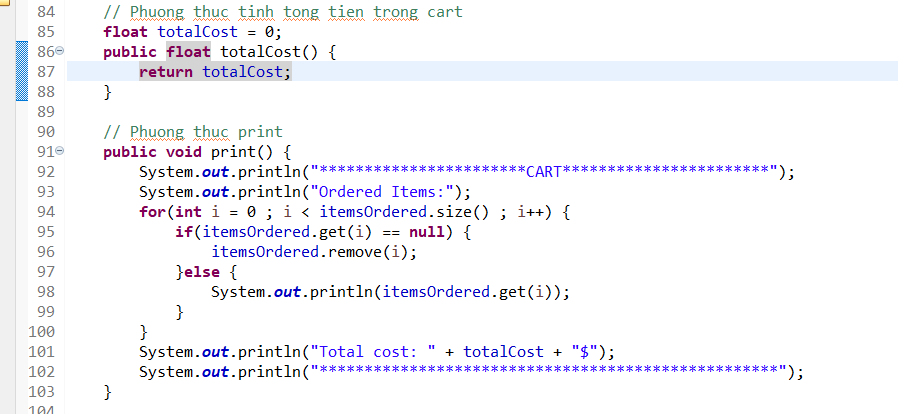
You could add more methods to add and remove **Book** and **CompactDisc**, but since **DigitalVideoDisc**, **Book** and **CompactDisc** are all subclasses of type **Media**, you can simply change **Cart** to maintain a collection of **Media** objects. Thus, you can add a **DigitalVideoDisc**, or a **Book**, or a **CompactDisc** using the same methods.

* Create **addMedia**() and **removeMedia**() to replace **addDigitalVideoDisc**() and **removeDigitalVideoDisc**()
* Update the **totalCost**() method

RESULT:





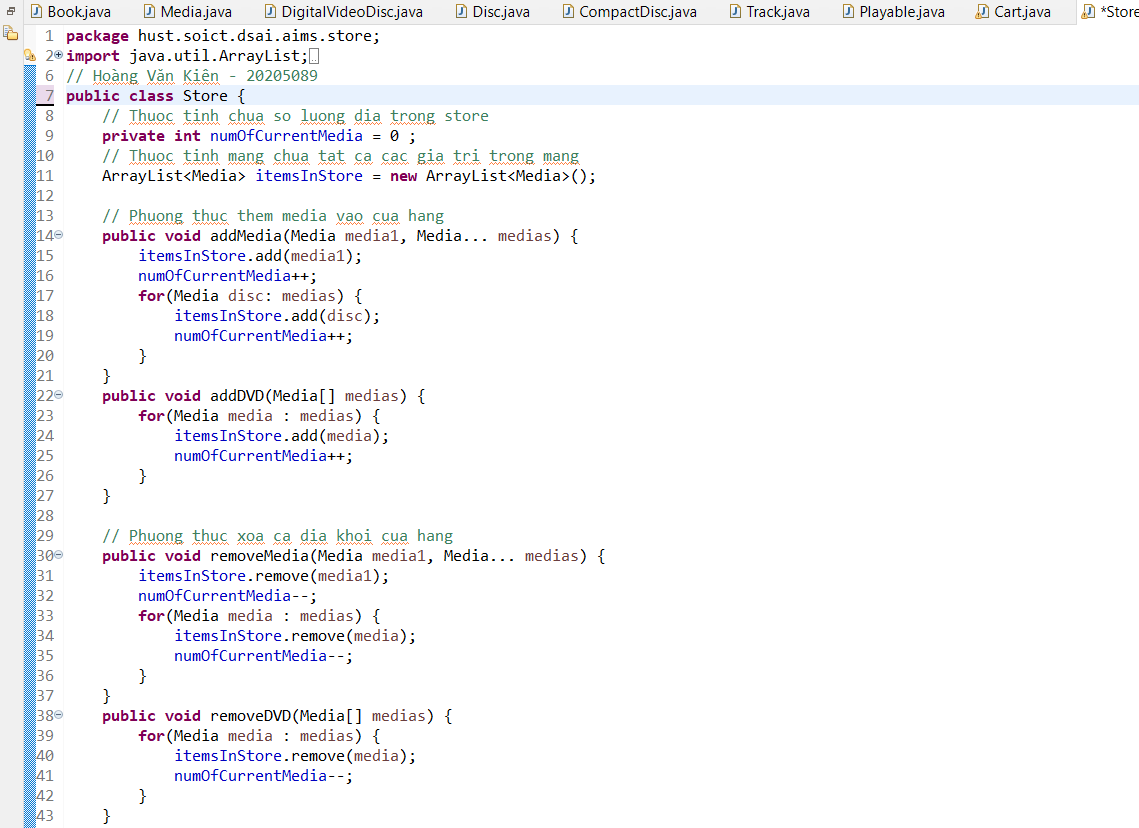




# Update the **Store** class to work with **Media**

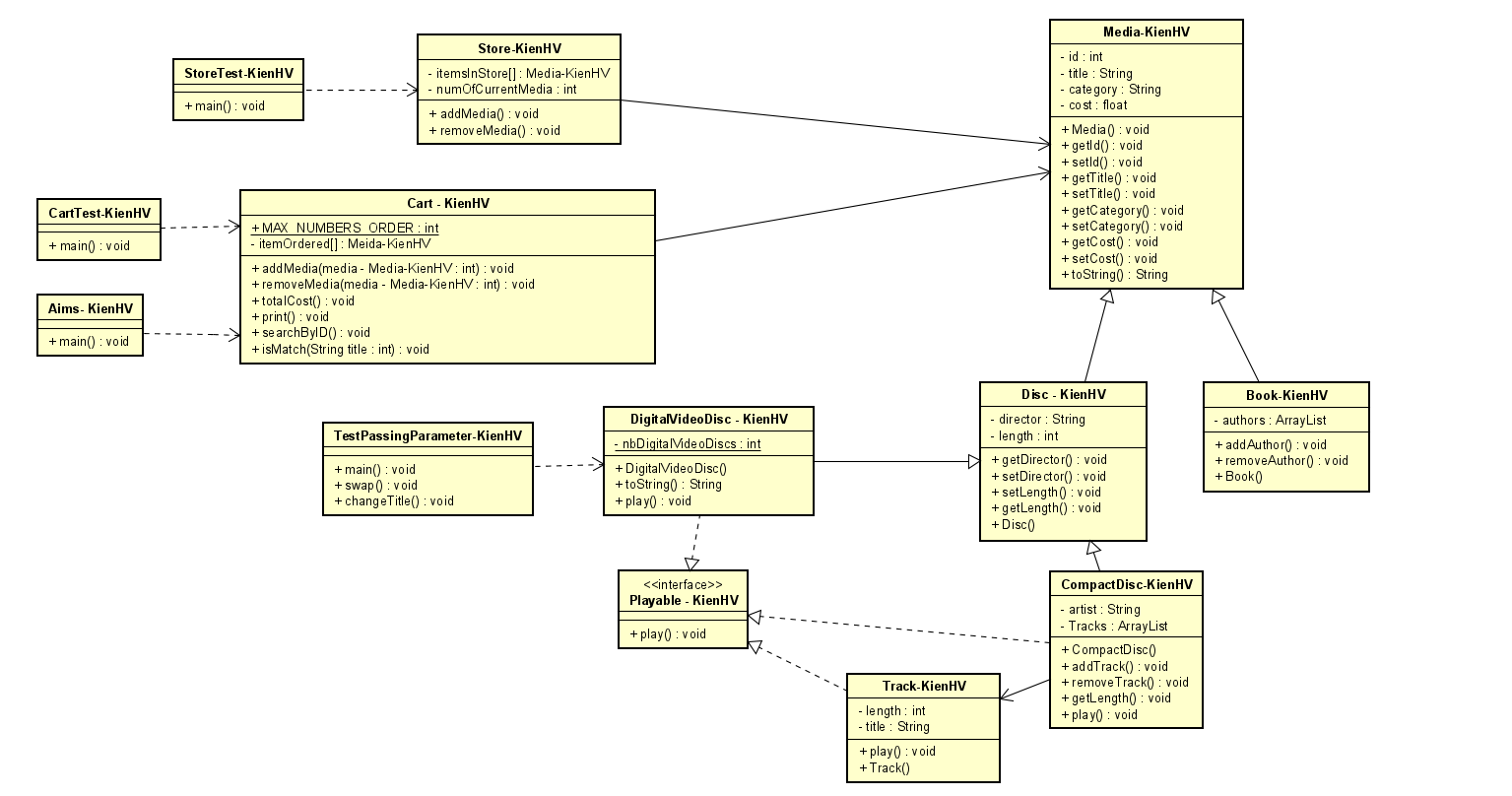
* Similar to the **Cart** class, change the **itemsInStore[]** attribute of the **Store** class to **ArrayList<Media>** type.
* Replace the **addDigitalVideoDisc**() and **removeDigitalVideoDisc**() methods with **addMedia**() and **removeMedia**()

**RESULT :**

****

# Constructors of whole classes and parent classes

- Update the UML class diagram for the **AimsProject**. Update the new .astah & .png file in the **Design** directory. We can apply Release Flow here by creating a branch, e.g., **topic/update-class-diagram/aims-project/lab04,** push the diagram and its image, and then merge with master.

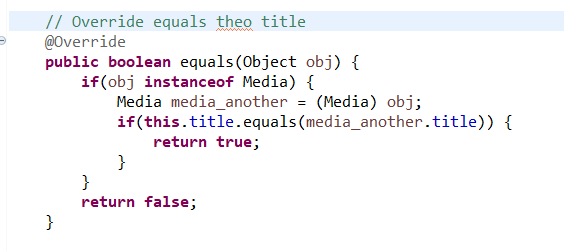


# Unique item in a list

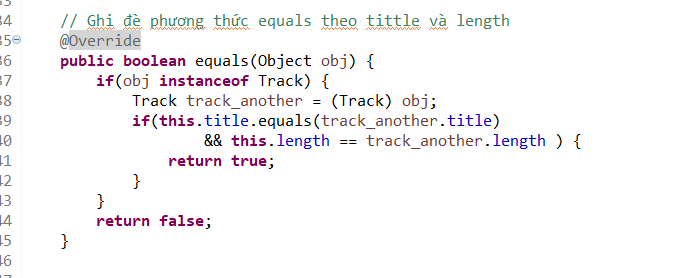
To make sure the list of media in cart or list of tracks in a CD should not contain identical objects, we can override the **equals()** method of the **Object** class

* Please override the boolean equals(Object o) of the Media and the Track class so that two objects of these classes can be considered as equal if:

+ For the Media class: the title is equal



+ For the Track class: the title and the length are equal



# Polymorphism with toString() method

- Create an ArrayList of Media, then add some media (CD, DVD or Book) into the list.

- Iterate through the list and print out the information of the media by using toString() method. Observe what happens and explain in detail.

***Code:***

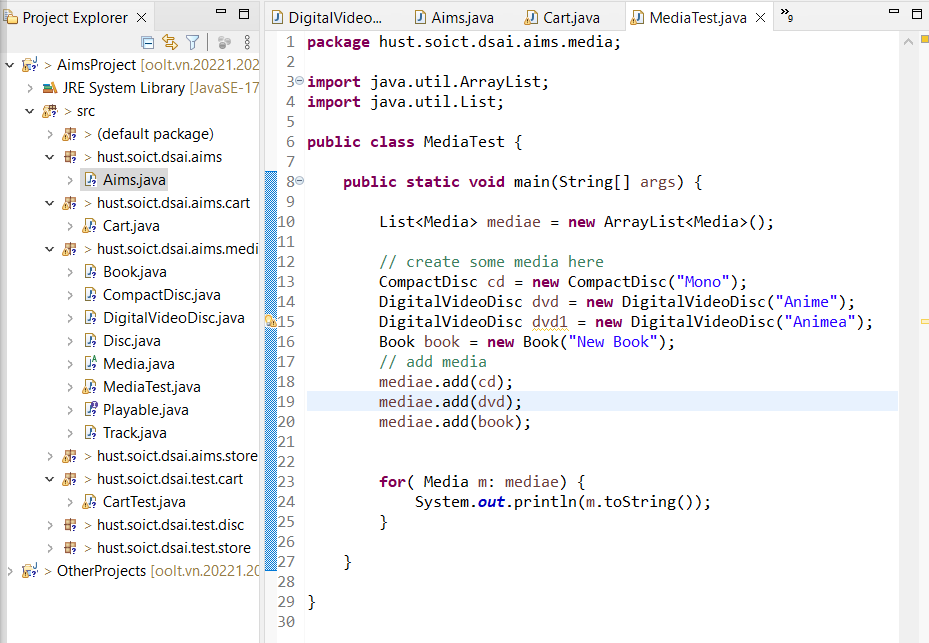
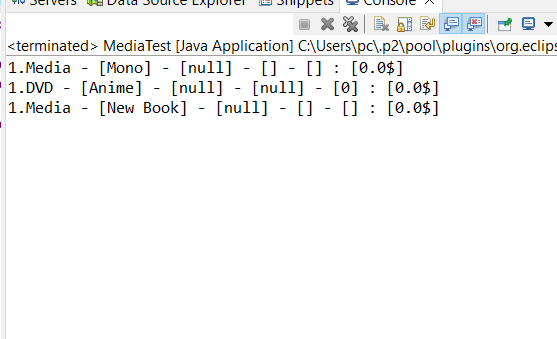
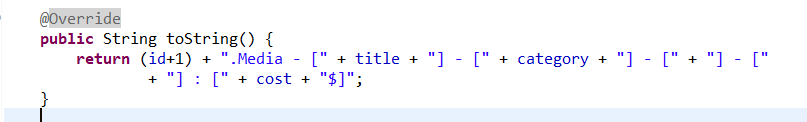
**

Figure 3. Polymorphism code

***Result:***

******

***Giải thích lí do : Vì em đã ghi đề phương thức toString trong lớp Media***

******

# Sort media in the cart

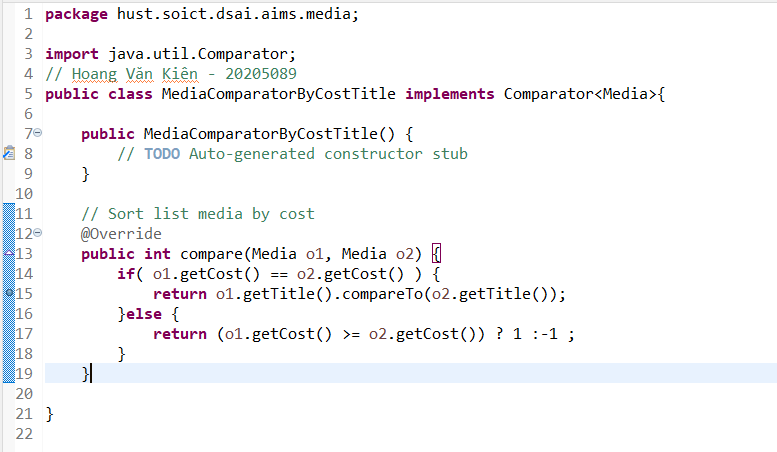
* Sort by title: the system displays all the medias in the alphabet sequence by title. In case they have the same title, the medias having the higher cost will be displayed first.

***RESULT:***

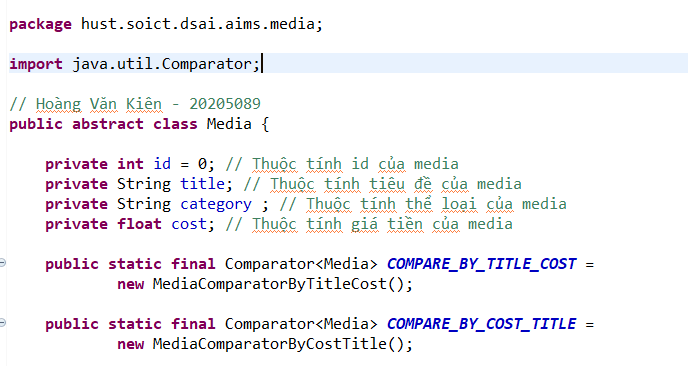


* Sort by cost: the system the system displays all the medias in decreasing cost order. In case they have the same cost, the medias will be ordered by title (alphabetical).

***RESULT:***



* Add the comparators as attributes of the Media class:

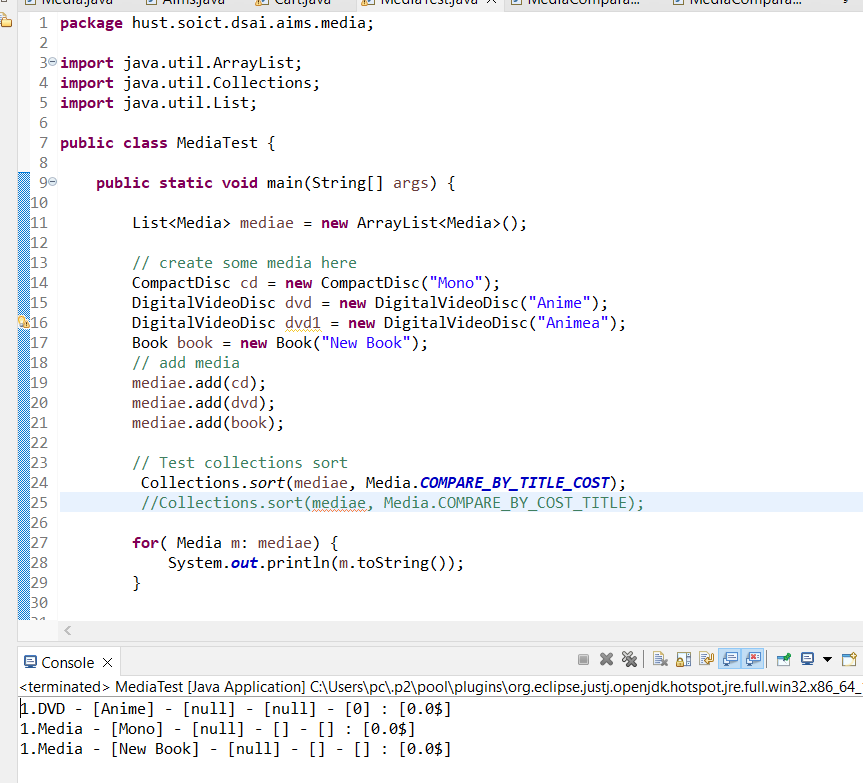


* Pass the comparator into Collections.sort: java.util.Collection.sort(collection,Media.COMPARE\_BY\_TITLE\_COST)

or

java.util.Collection.sort(collection, Media.COMPARE\_BY\_COST\_TITLE)

***TEST RESULT:***

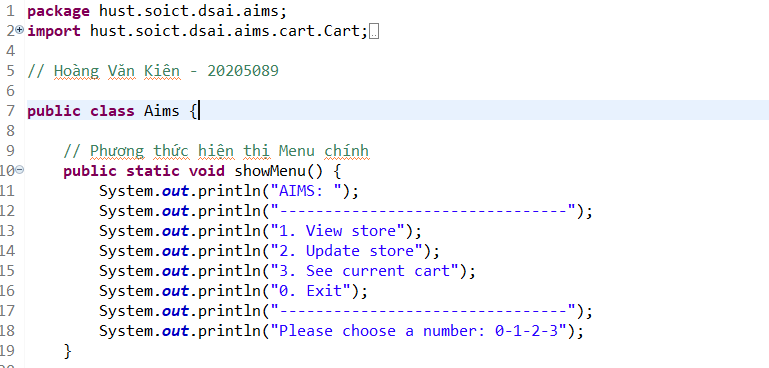


**ANSWER**:

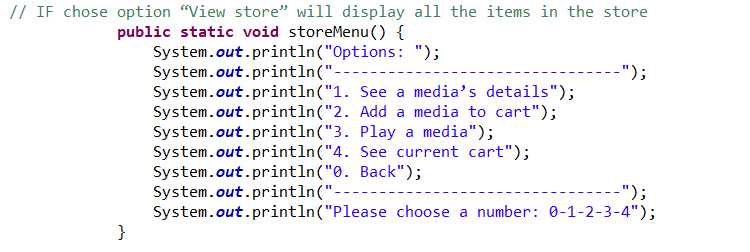


# Create a complete console application in the Aims class

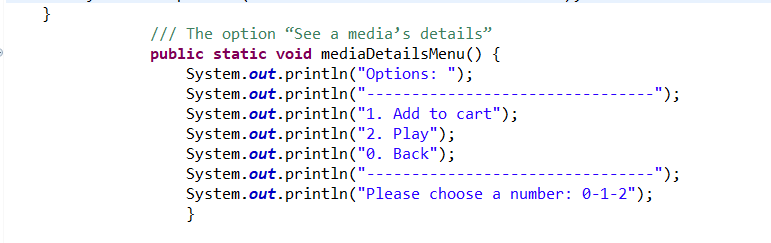
In the **main** method of **Aims**, you will now implement a complete console application, by first create an instance of the **Store** class and then, provide a list of functionalities through a menu that the user can interact with. For the home interface, you will create the main menu as following:



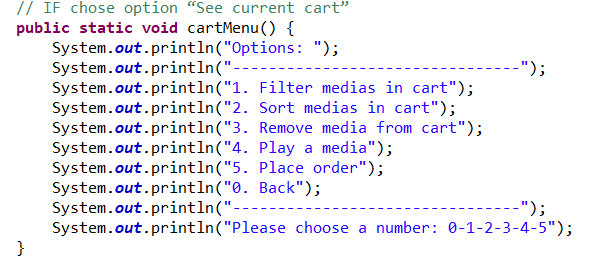
* From the main menu, if the user choses option “**View store**”, the application will display all the items in the store, and a menu as following:



The option “**See a media’s details**” will ask the user to enter the title of the media and display the information of that media. Please remember to check the validity of the title. Under the information display, the system also shows the following menu (note that the “**Play**” option is only available to CD and DVD type.



* The option “**Add a media to cart**” will ask the user to enter the title of the media that he/she sees on the screen (the list of medias in store), then add the media to cart. Please remember to check the validity of the title. After adding a DVD to cart, the system will display the number of DVDs in the current cart.
* The option “**Play a media**” will ask the same input from the user as option 2. You should again check the validity of the title.
* From the main menu, if the user choses option “**Update store**”, the application will allow the user to add a media to or remove a media from the store
* From the main menu, if the user choses option “**See current cart**”, the application will display the information of the current cart, and a menu as following:



The “**Filter medias in cart”** option should allow the user to choose one of two filtering options: by id and by title.

The “**Sort medias in cart**” option should allow the user to choose one of two sorting option: by title or by cost.

**Note**: When the user chooses option “**Place order**”, the system is supposed to move on to the Delivery Information gathering & Payment step. However, for simplicity, within the scope of this lab course, when the user chooses this option, we only need to notify the user that an order is created and empty the current cart.