Contents

[**Lab 04: Inheritance and Polymorphism** 2](#_Toc151780313)

[Use case diagram 2](#_Toc151780314)

[Class diagram 2](#_Toc151780315)

[3. Creating the Book class 3](#_Toc151780316)

[4. Creating the abstract Media class 8](#_Toc151780317)

[5. Creating the CompactDisc class 11](#_Toc151780318)

[5.1. Create the Disc class extending the Media class 11](#_Toc151780319)

[5.2. Create the Track class: 12](#_Toc151780320)

[5.3. Open the CompactDisc class and save your changes. 13](#_Toc151780321)

[6. Create the Playable interface 14](#_Toc151780322)

[7. Update the Cart class to work with Media 16](#_Toc151780323)

[8. Cập nhật phương thức totalCost(): 18](#_Toc151780324)

[9. Constructors of whole classes and parent classes 19](#_Toc151780325)

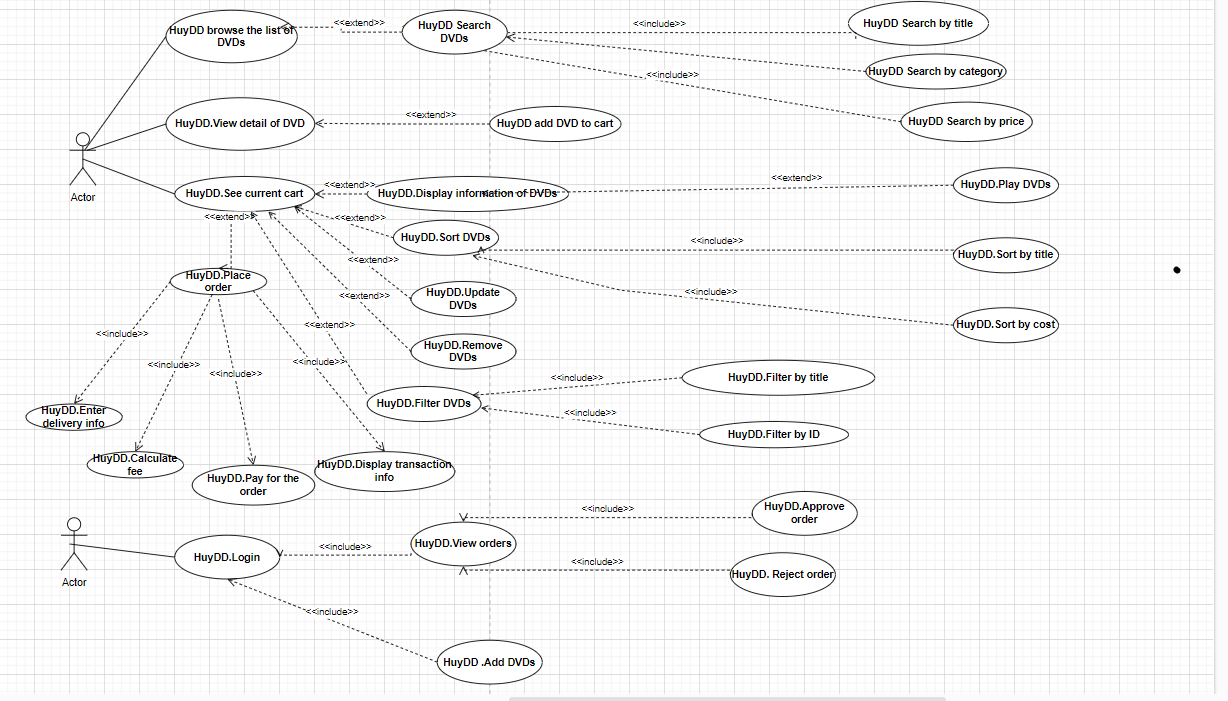
[10. Unique item in a list 21](#_Toc151780326)

[11. Polymorphism with toString() method 22](#_Toc151780327)

[13. Create a complete console application in the Aims class 24](#_Toc151780328)

# **Lab 04: Inheritance and Polymorphism**

# Use case diagram



**Mối quan hệ:**

- Include: quan hệ bắt buộc phải có giữa các use case với nhau.

- Extend: quan hệ mở rộng giữa các use case với nhau.

# Class diagram

A screenshot of a computer

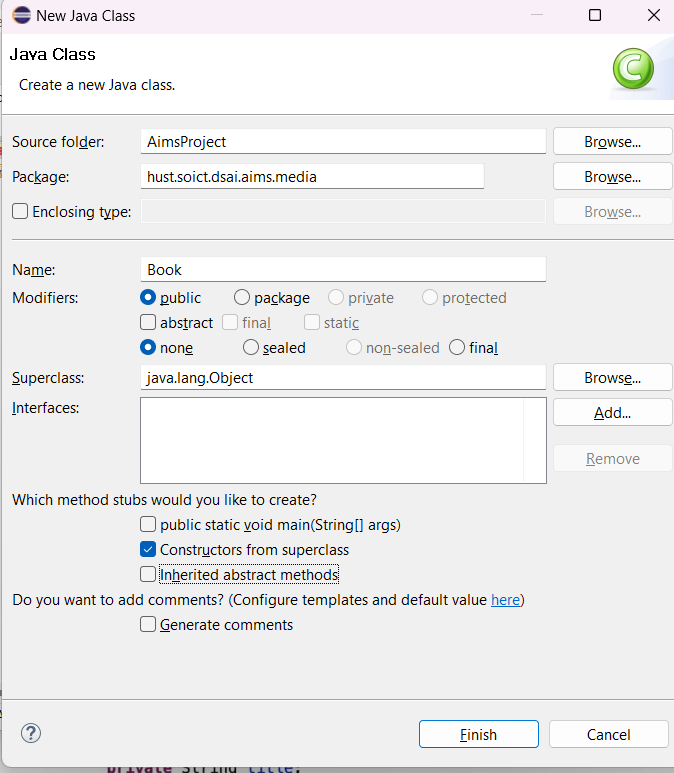
Description automatically generated

**Mối quan hệ:**

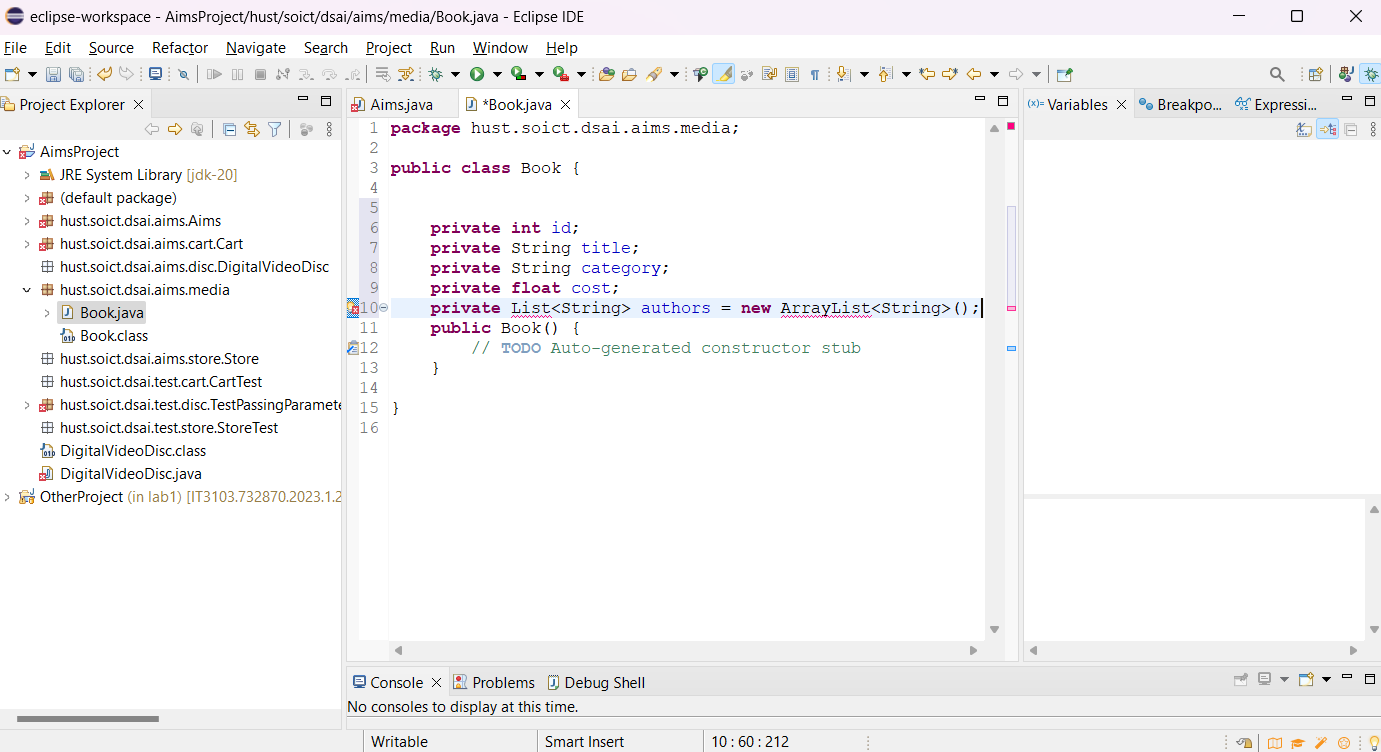
Quan hệ kế thừa : Đối tượng cụ thể sẽ kế thừa các thuộc tính và phương thức của đối tượng tổng quát .

Quan hệ phụ thuộc : quan hệ giữa 2 phần tử trong mô hình mà thay đổi ở phần tử này (phần tử độc lập) có thể gây ra thay đổi ở phần tử kia (phần tử phụ thuộc).

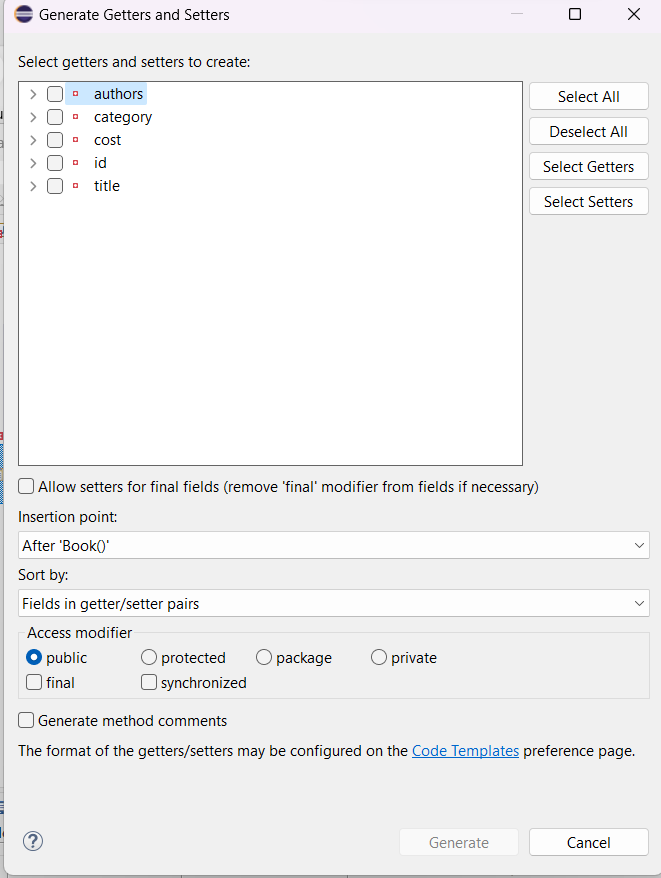
# 3. Creating the Book class

1. 

Thêm các trường vào lớp Book



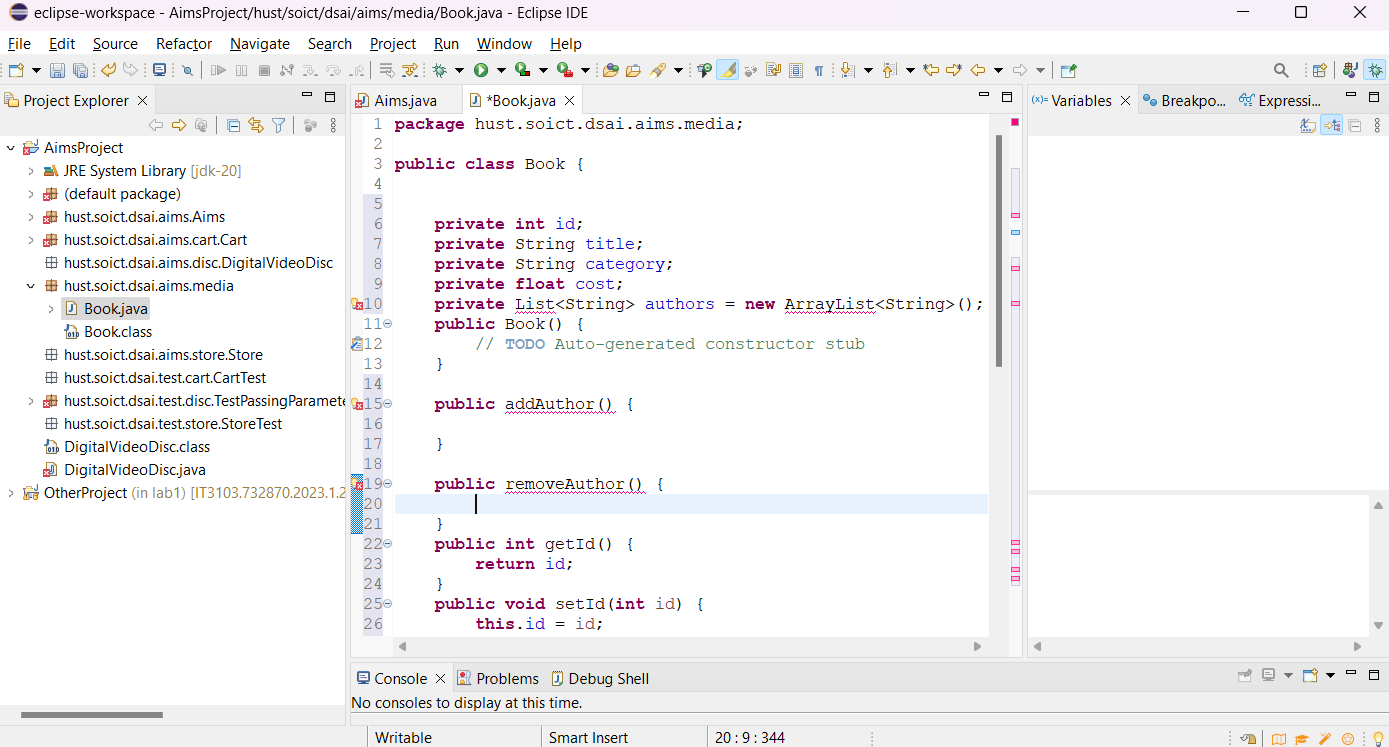
Generate Getter and Setter option in the Outline view pop-up menu



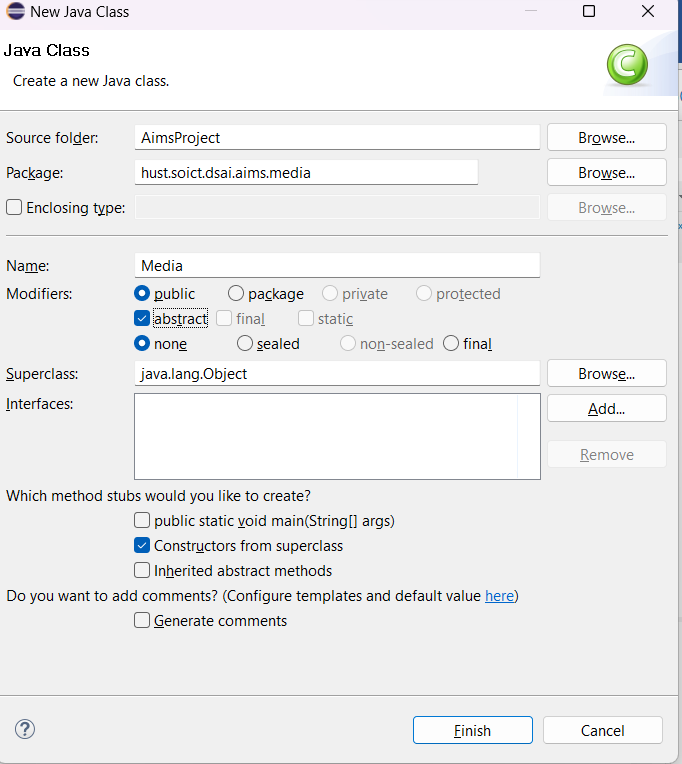
Tiếp theo, tạo addAuthor(String AuthorName) và RemoveAuthor(String AuthorName) cho lớp Book

Phương thức addAuthor(...) phải đảm bảo rằng tác giả chưa có trong ArrayList trước khi thêm

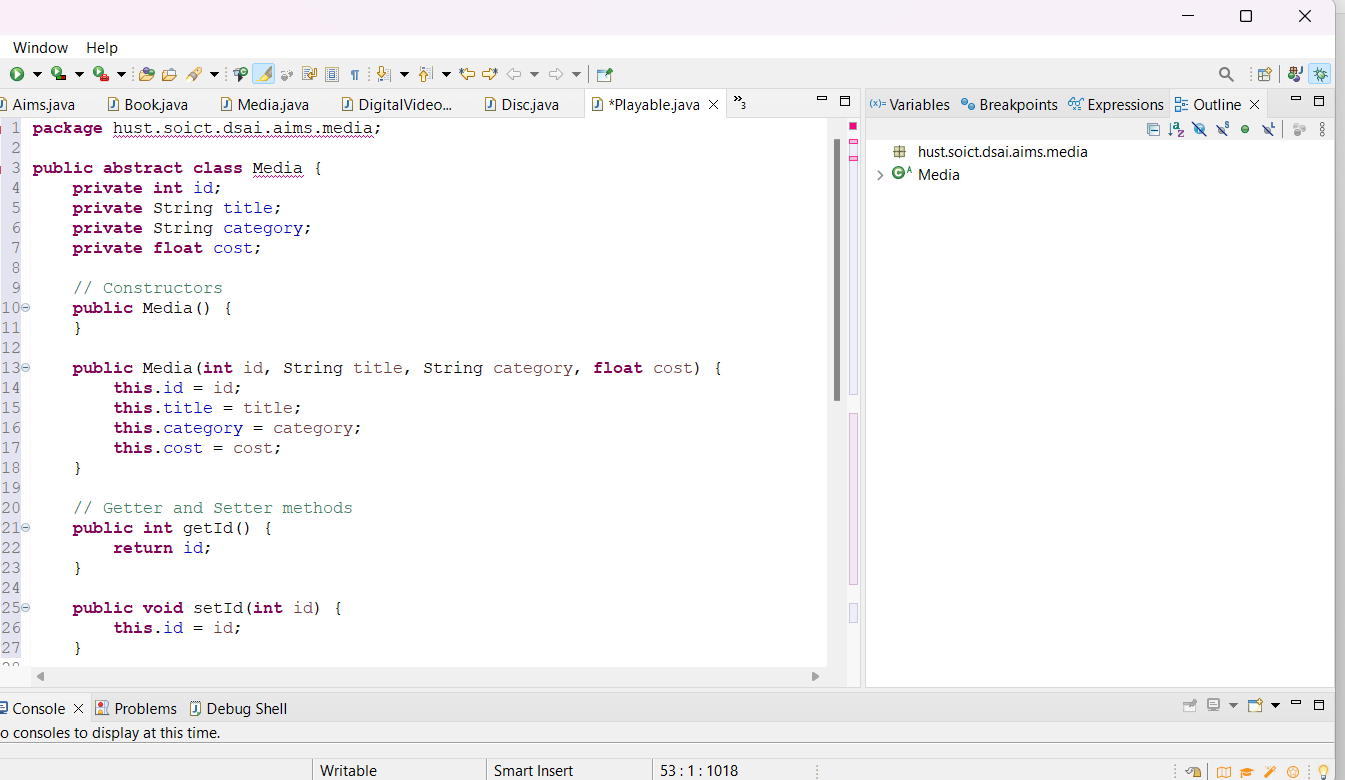
Phương thức RemoveAuthor(...) phải đảm bảo rằng tác giả có mặt trong ArrayList trước khi xóa

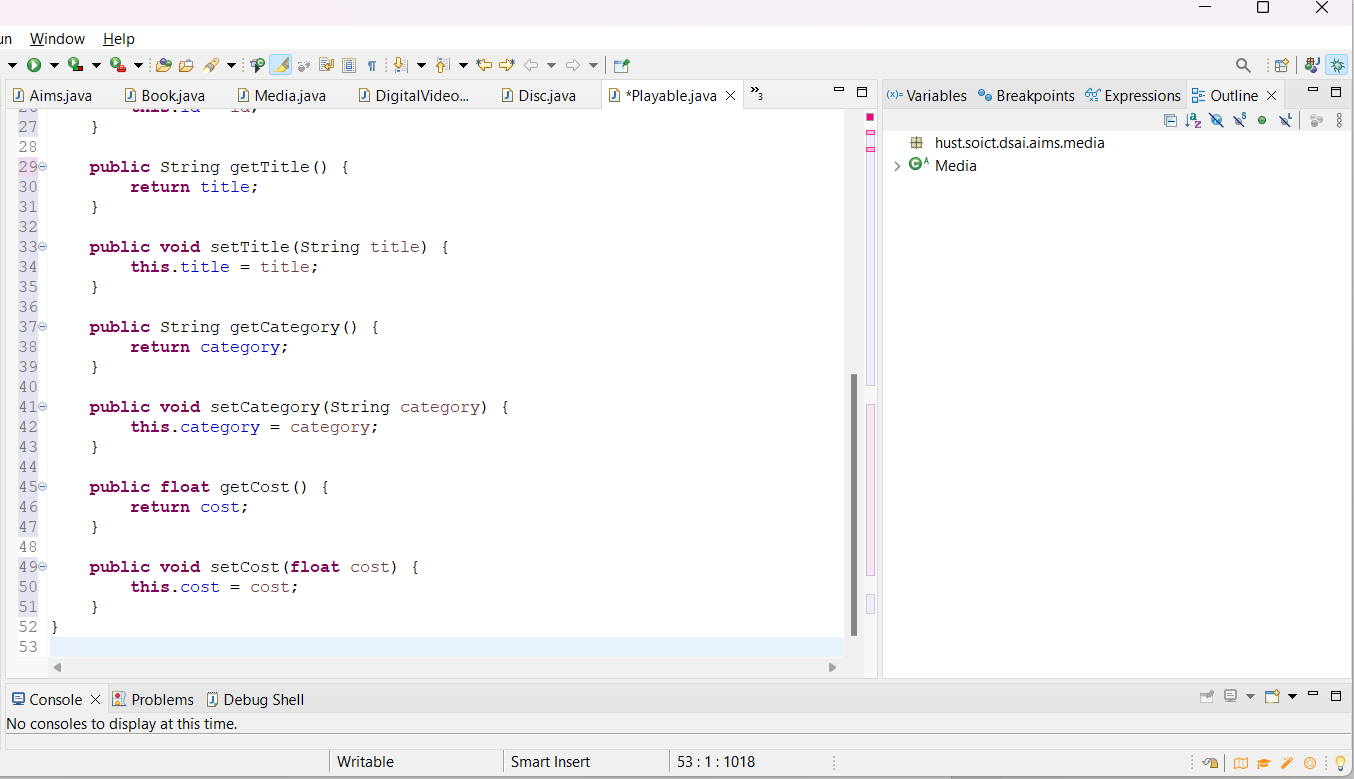


# 4. Creating the abstract Media class

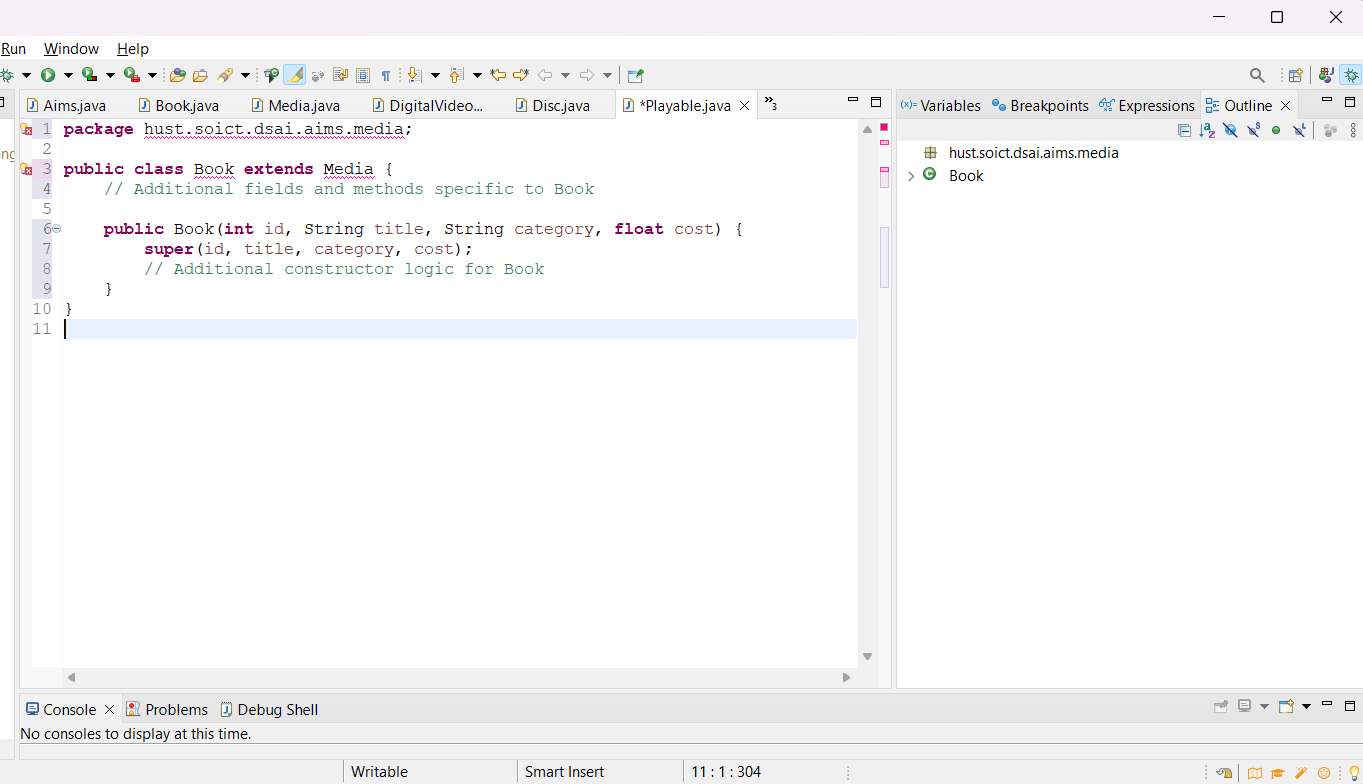


Thêm các trường vào lớp Media

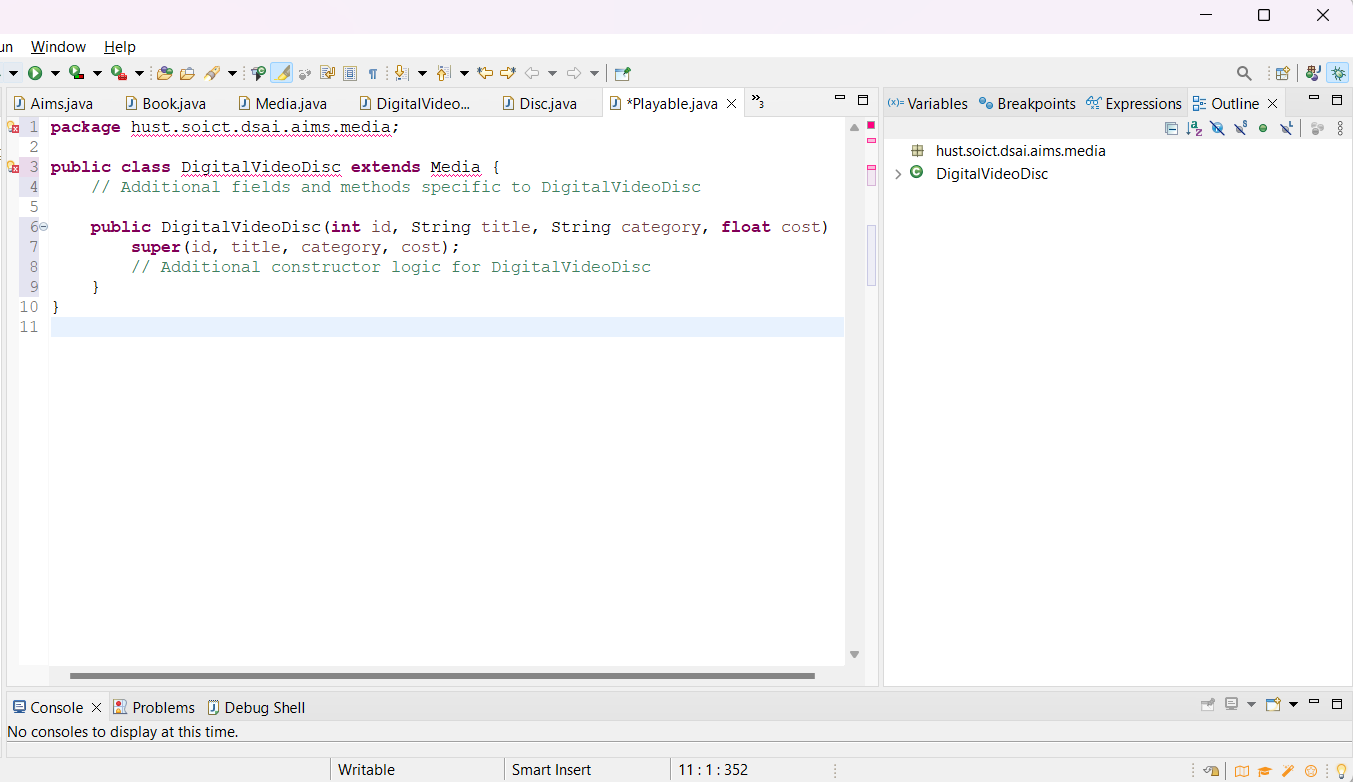




2. Update the Book class:

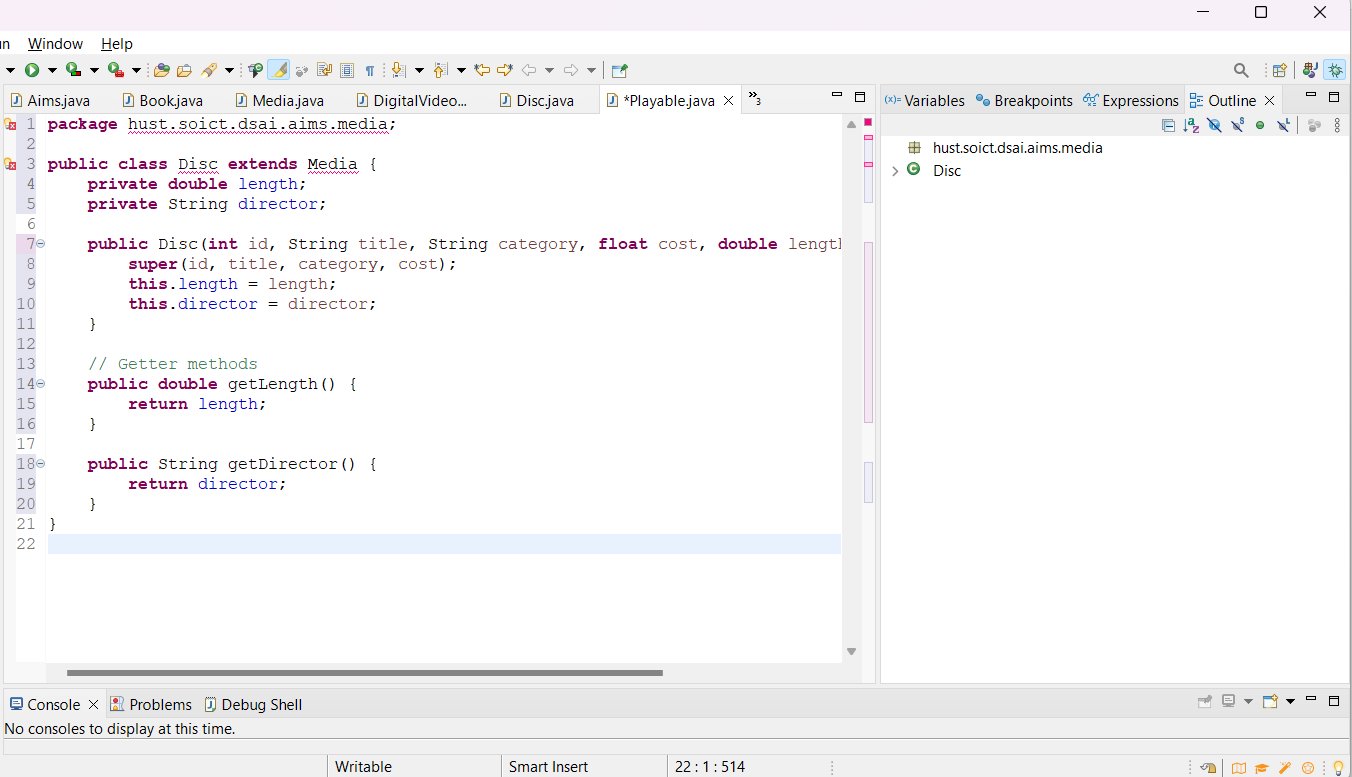
****

3. Update the DigitalVideoDisc class:

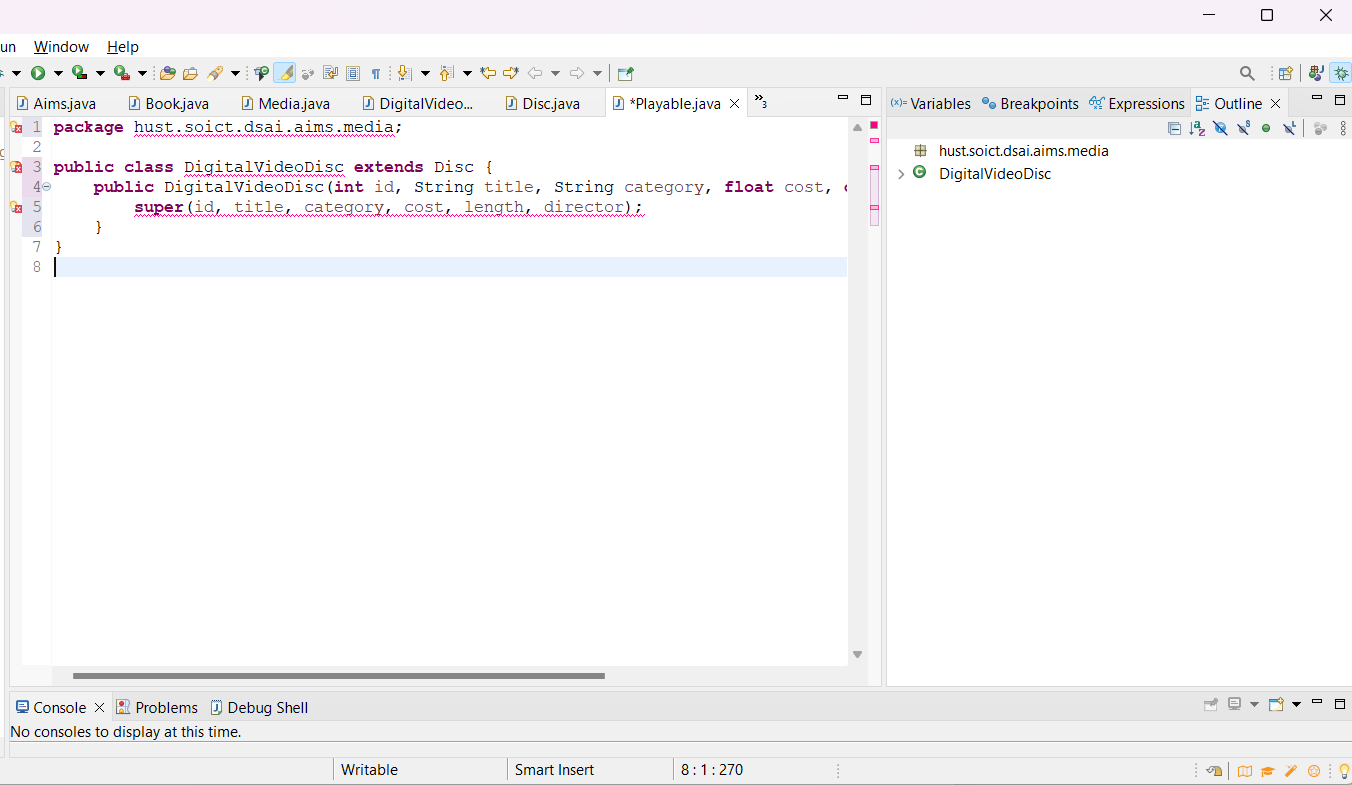


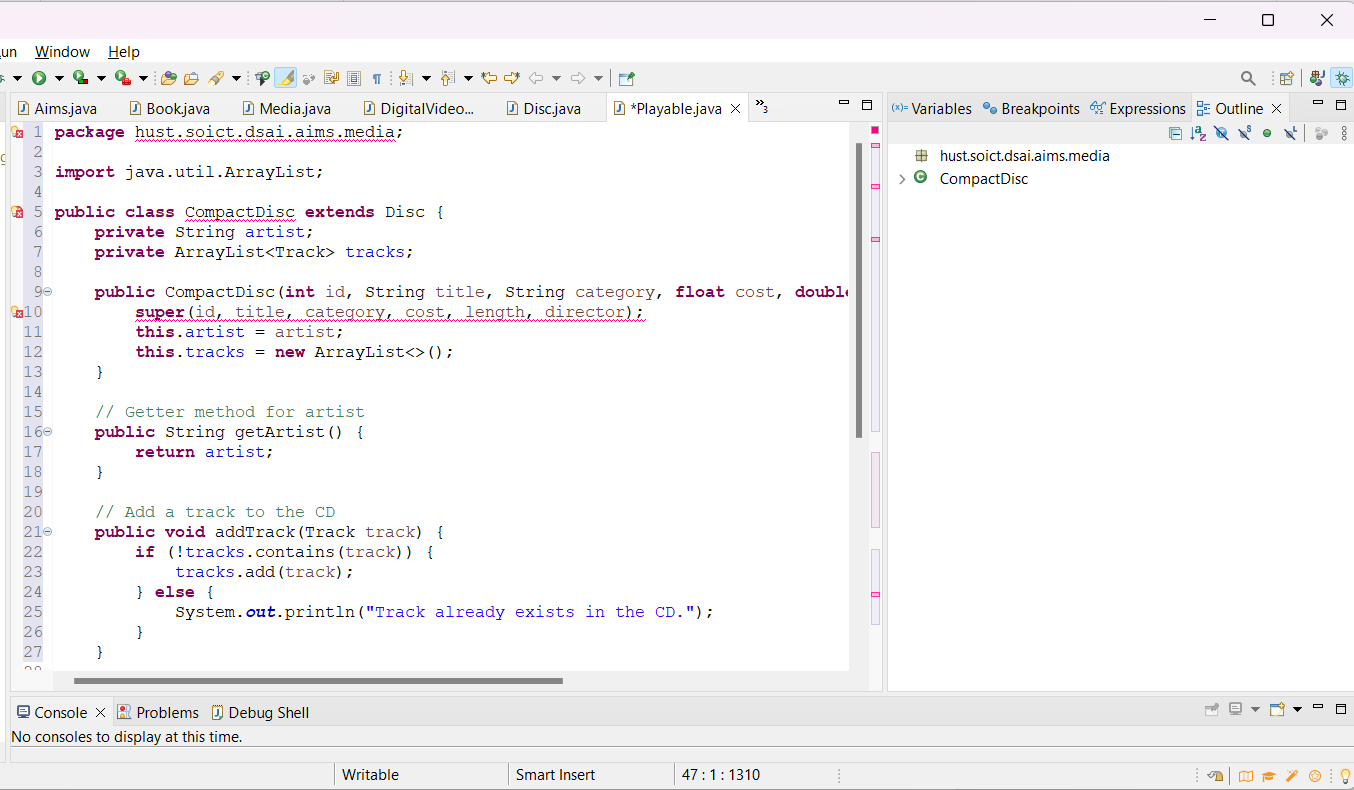
# 5. Creating the CompactDisc class

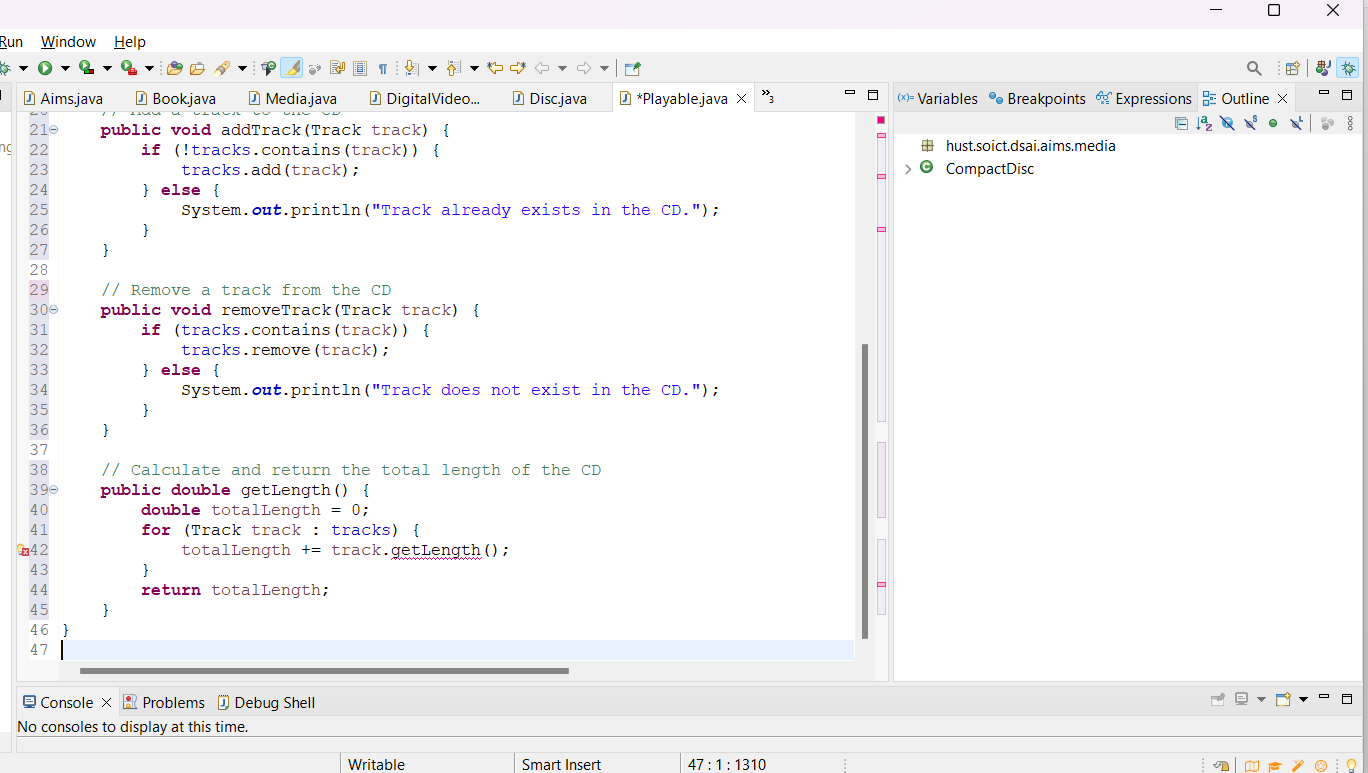
## 5.1. Create the Disc class extending the Media class



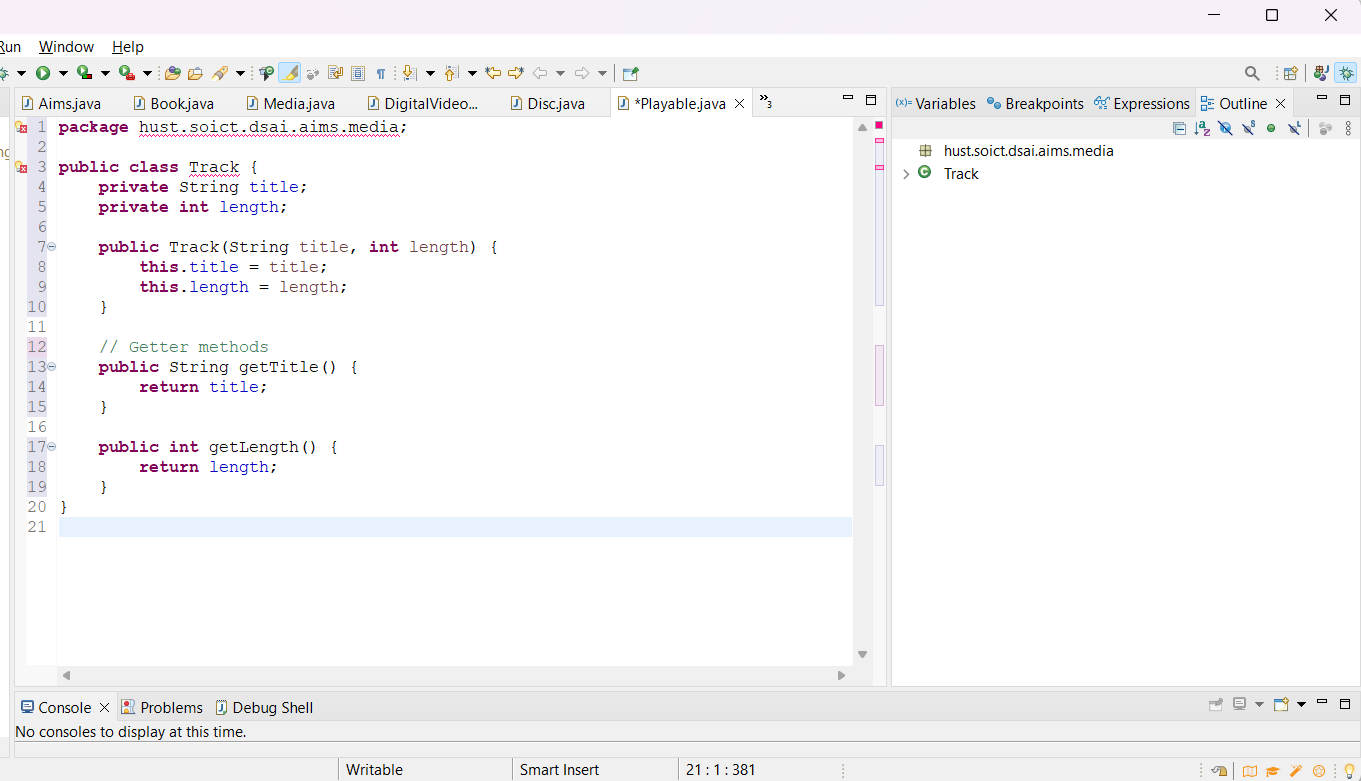
Tạo DigitalVideoDisc mở rộng lớp Disc:



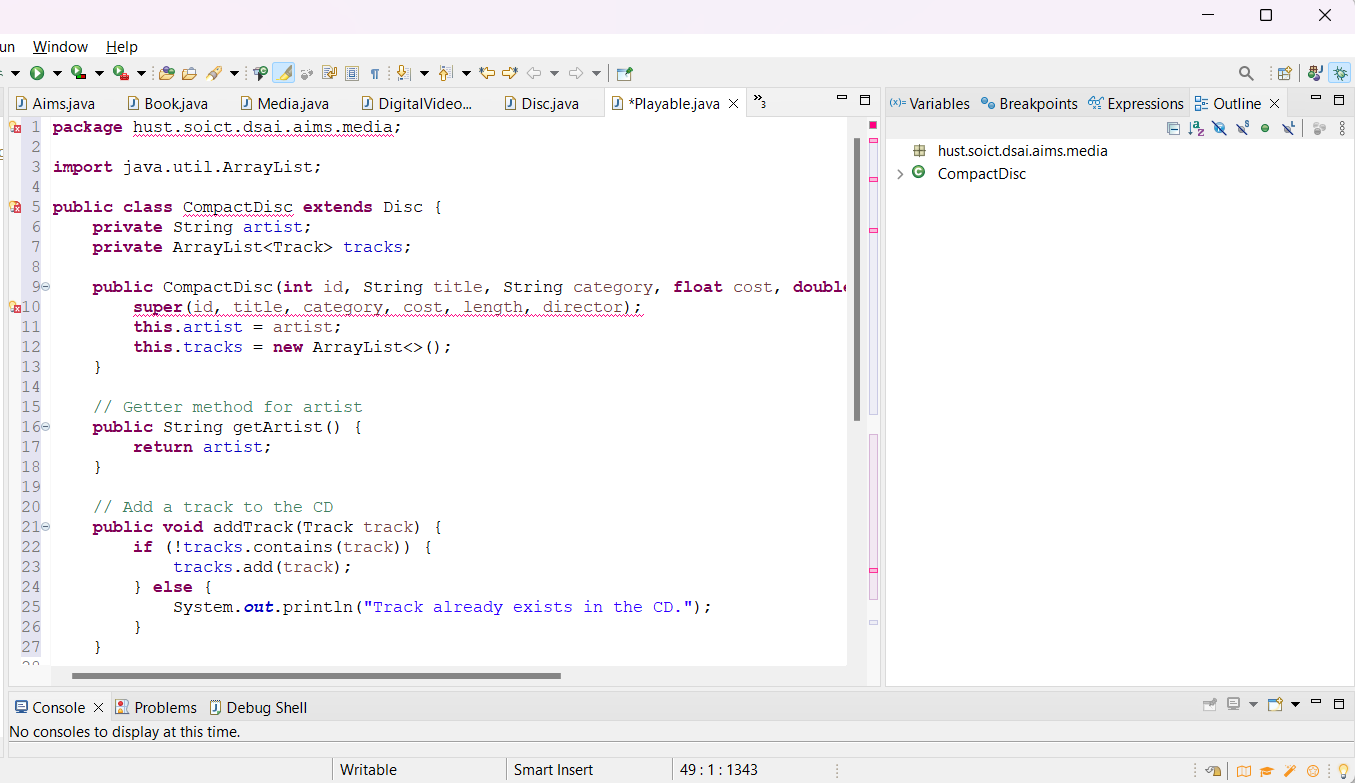
Tạo CompactDisc mở rộng lớp Disc:

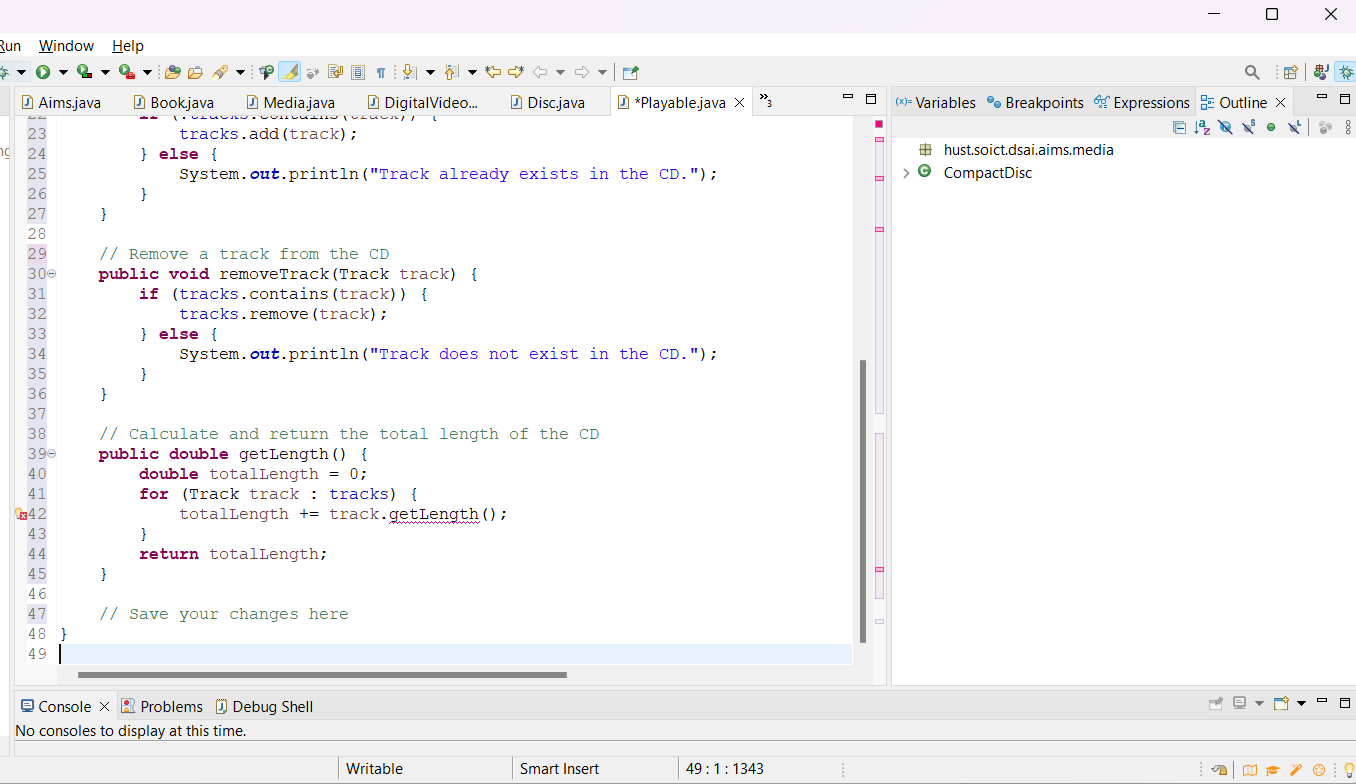


## 5.2. Create the Track class:



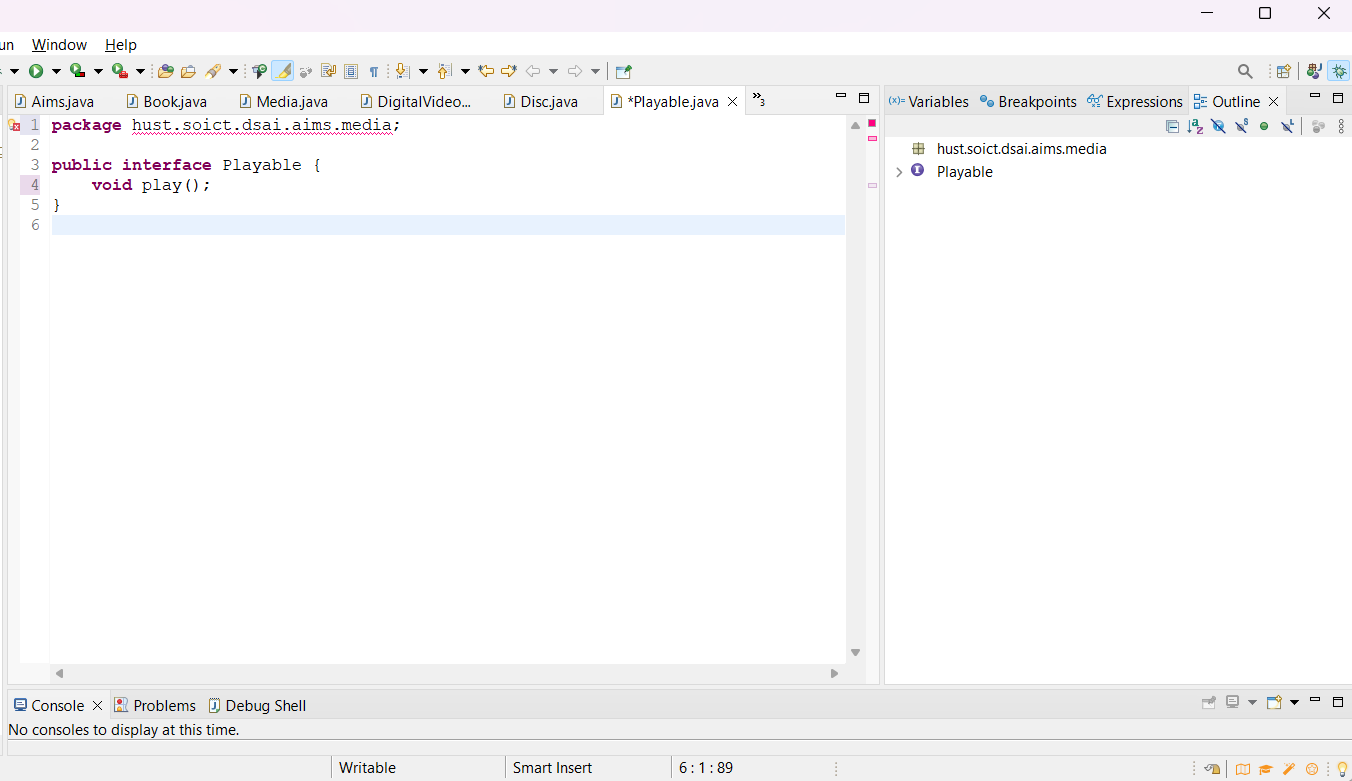
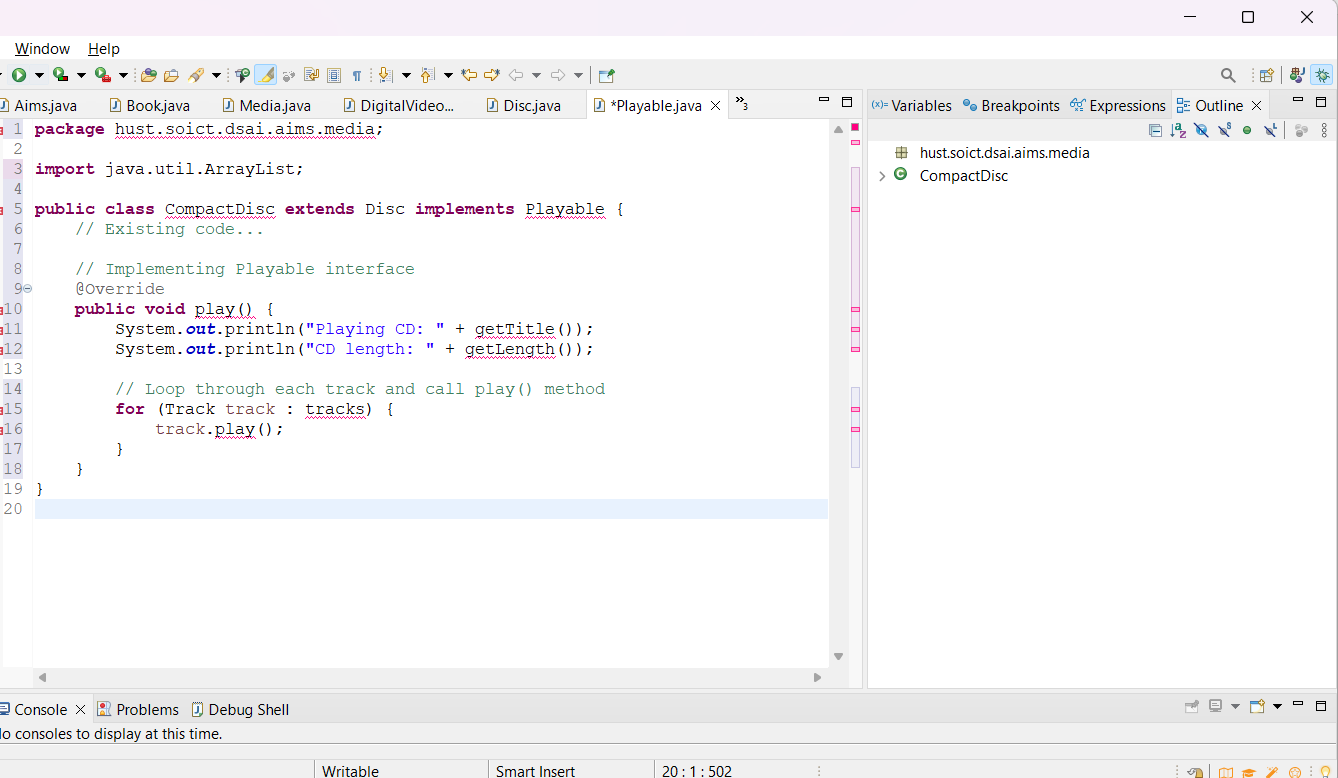
## 5.3. Open the CompactDisc class and save your changes.

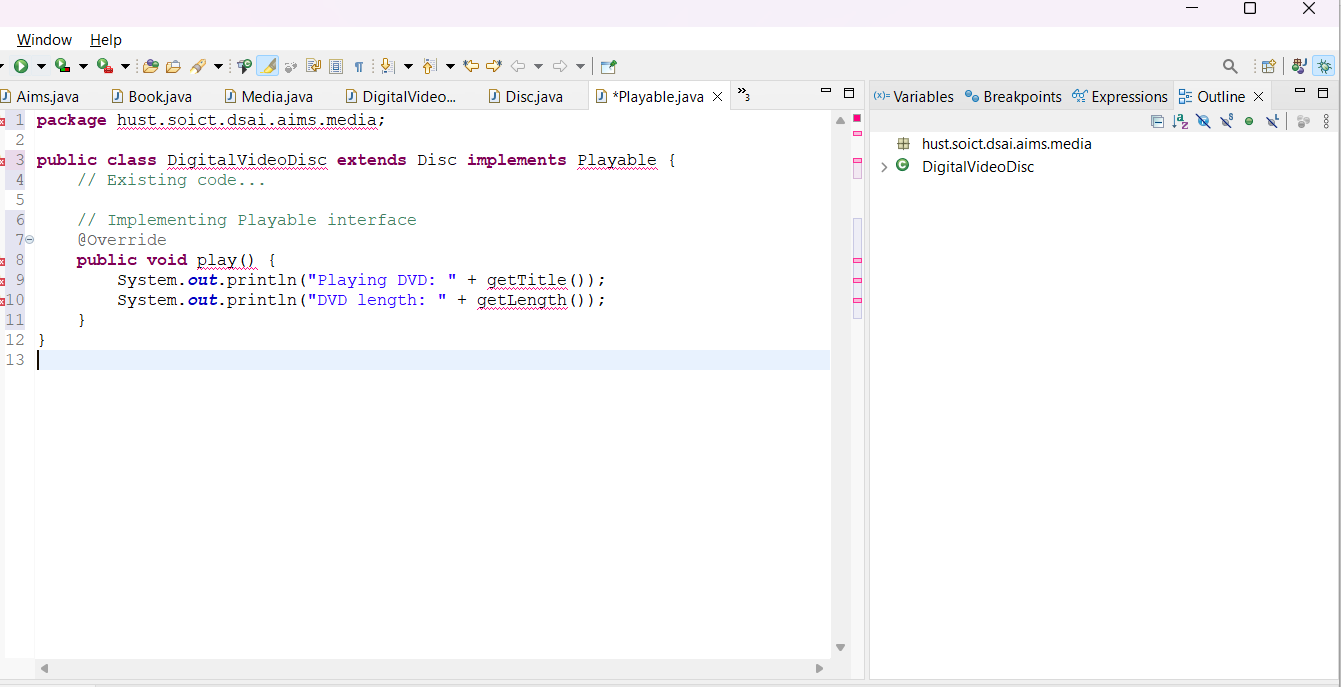


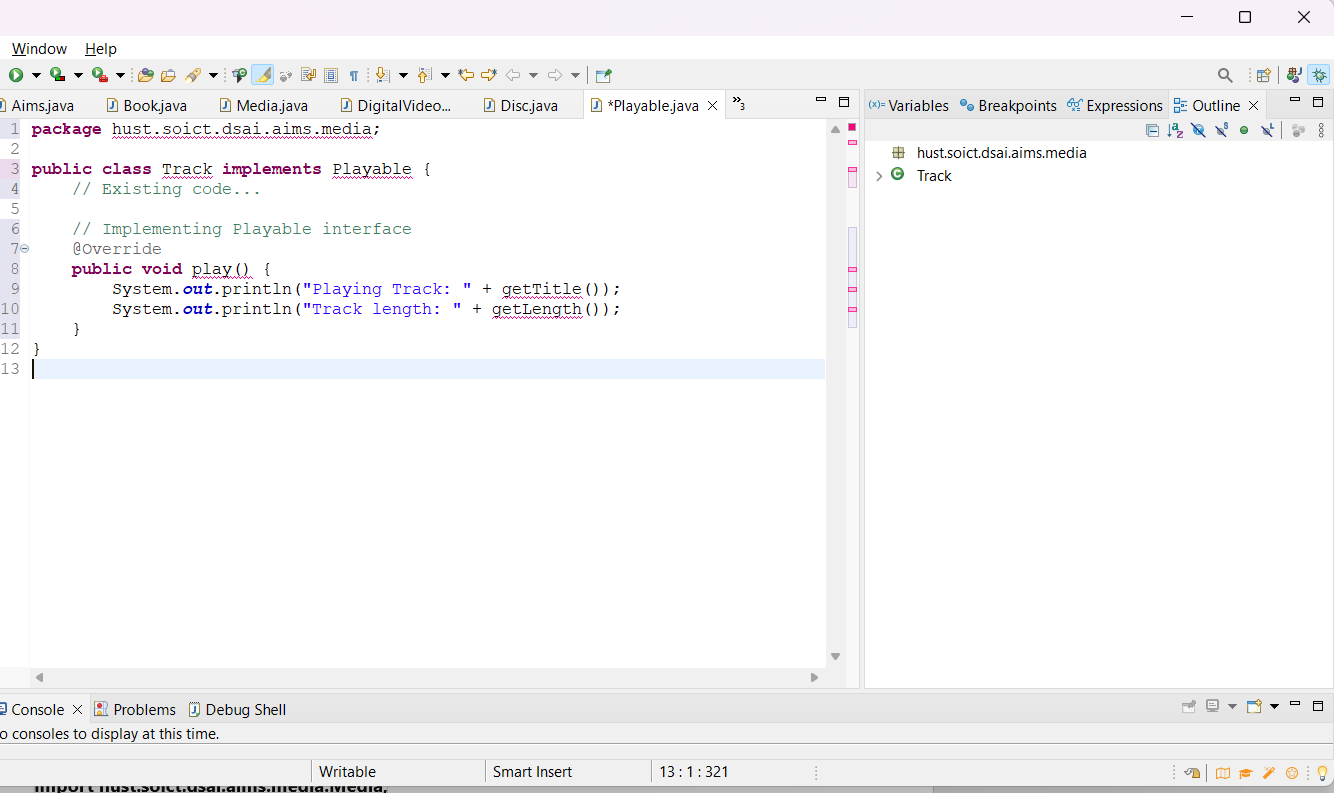


# 6. Create the Playable interface

1. Tạo giao diện Playable:

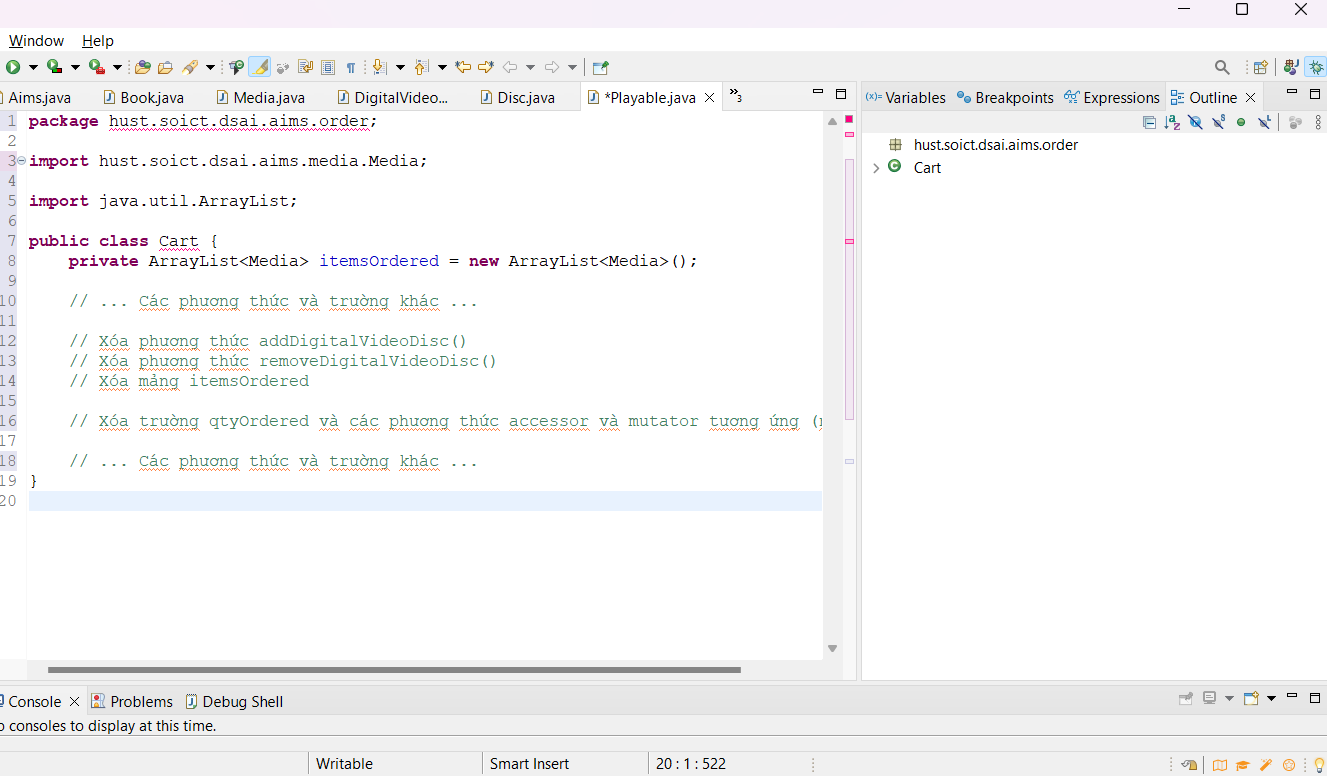
****  
2. Triển khai có thể phát được trong CompactDisc, DigitalVideoDisc và Track:**** Trong lớp DigitalVideoDisc:



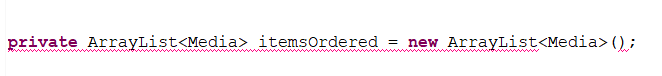
Trong lớp Track

# 7. Update the Cart class to work with Media

1. Mở tệp Cart.java và xóa mảng itemsOrdered cùng với các phương thức thêm/xóa:

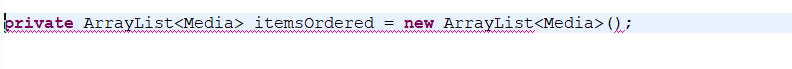
****

2. Tạo lại trường itemsOrdered dưới dạng ArrayList:

****

3. Xóa trường qtyOrdered và các phương thức accessor và mutator (nếu có):

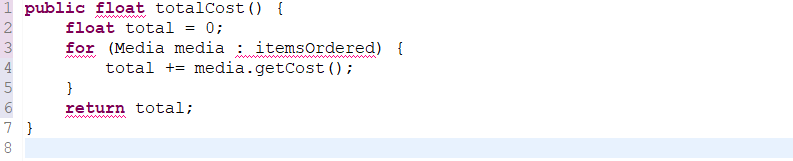
4. Thêm trường itemsOrdered vào class Cart:

****5. Sắp xếp Imports:

6. Lưu lại class của bạn.

7. Tạo phương thức addMedia() và removeMedia():

# 8. Cập nhật phương thức totalCost():

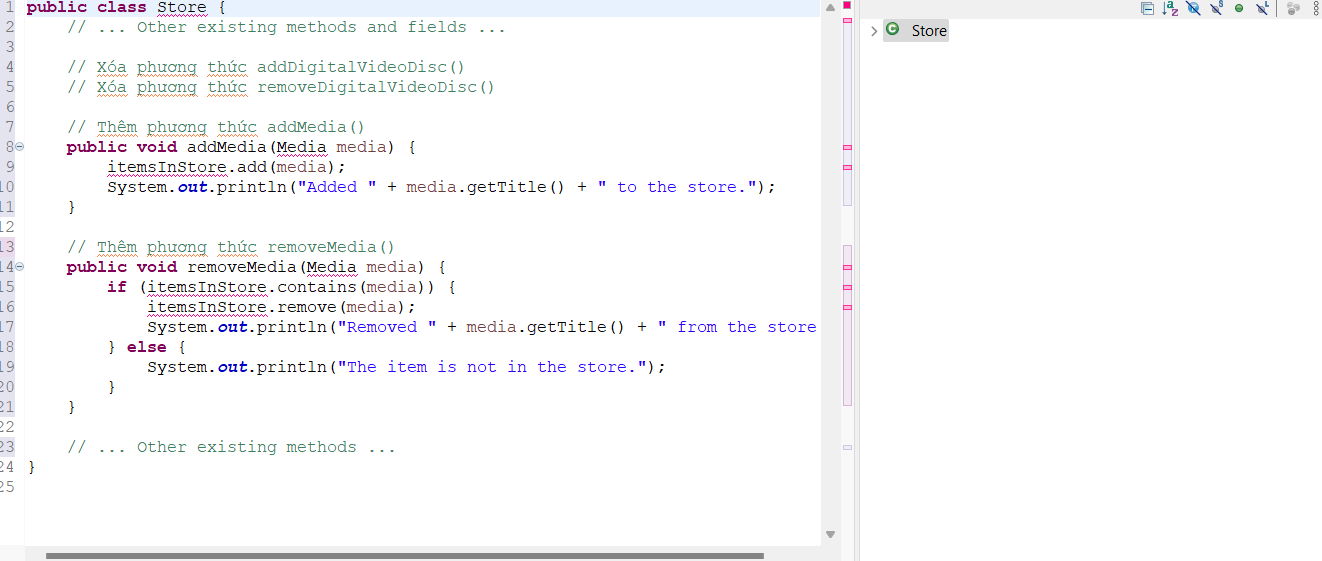
****

1. Cập nhật lớp Store để hoạt động với Media

Open Store.java and change the itemsInStore[] attribute to ArrayList<Media>:

****

2. Thay thế addDigitalVideoDisc() và RemoveDigitalVideoDisc() bằng addMedia() và RemoveMedia()

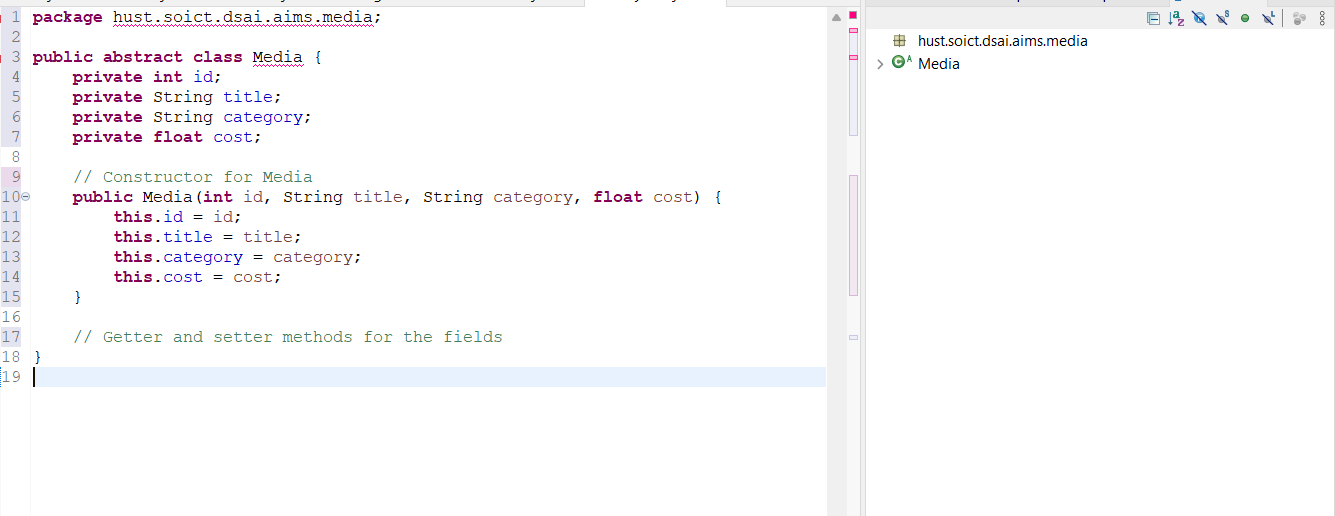
****

# 9. Constructors of whole classes and parent classes

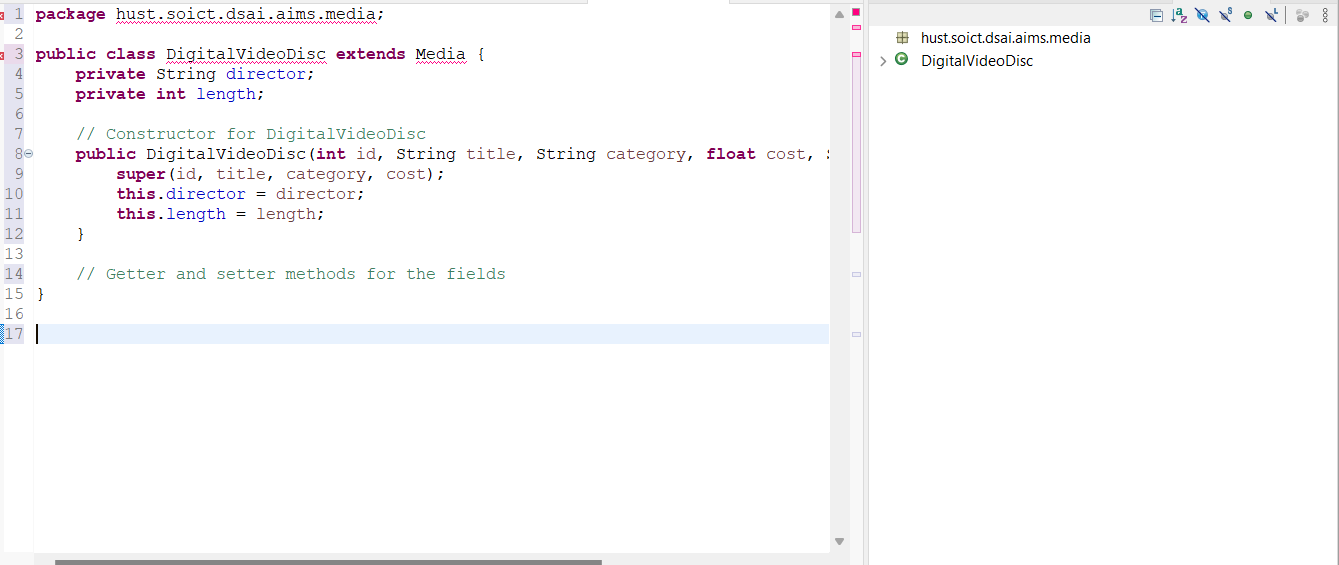
3. Write constructors for parent and child classes:

Ví dụ, giả sử bạn có một lớp cha Media và các lớp con DigitalVideoDisc và Book.

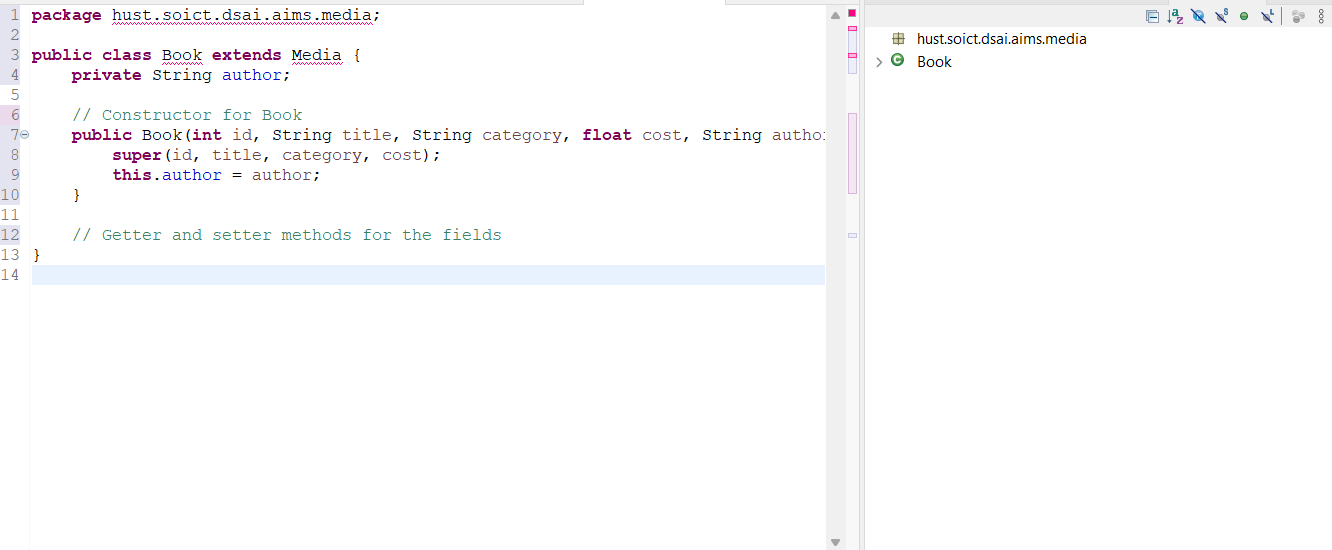
Lớp cha Media:

****

**Lớp con DigitalVideoDisc:**

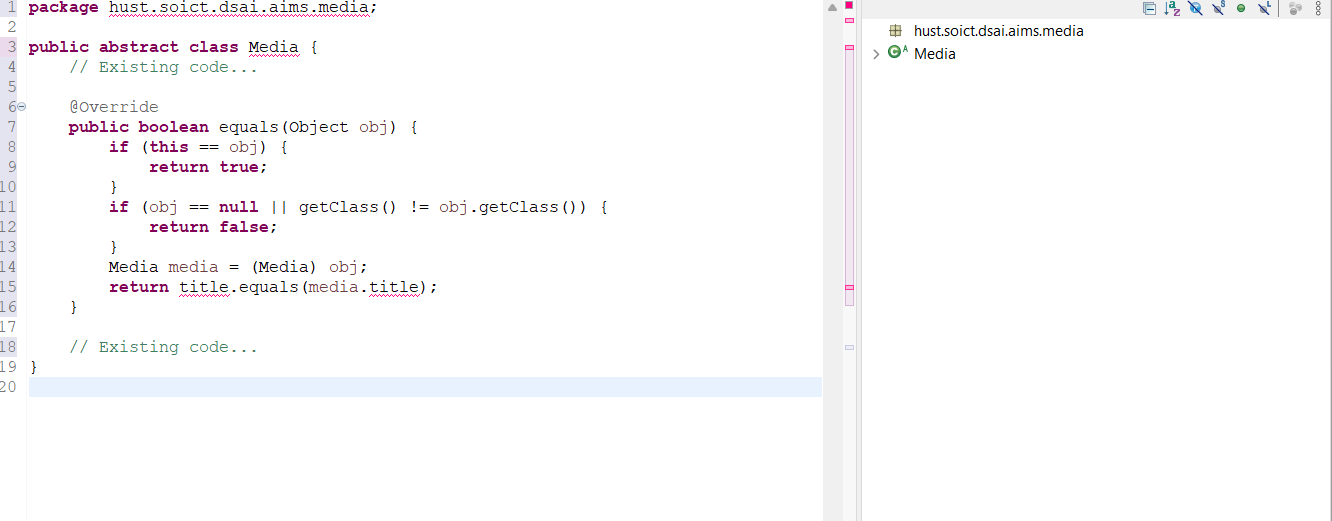
****

**Lớp con Book:**

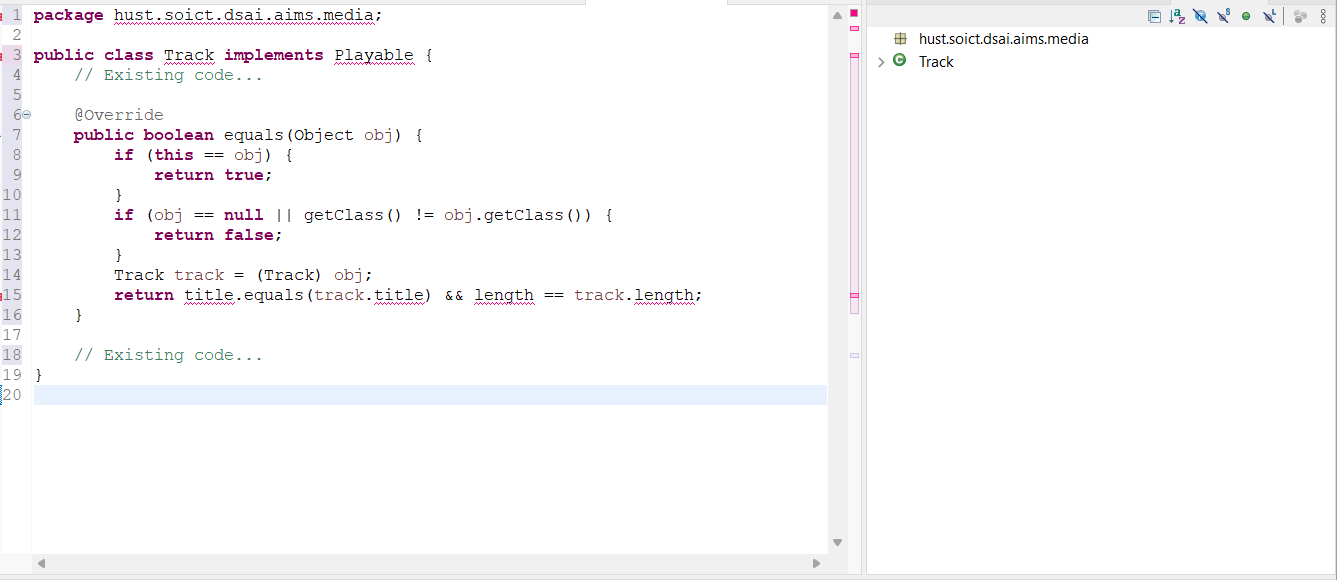
****

# 10. Unique item in a list

**Ghi đè phương thức equal() trong lớp Media**

****

**Ghi đè phương thức equal() trong lớp Track:**

****

**Khi ghi đè phương thức equal():**

-Kiểm tra xem đối tượng được so sánh có phải là cùng một thể hiện hay không, trả về true.

-Kiểm tra xem đối tượng được so sánh có null hay không cùng lớp, trả về false.

-Truyền đối tượng được so sánh vào loại lớp.

-So sánh các thuộc tính liên quan dựa trên các điều kiện được chỉ định.

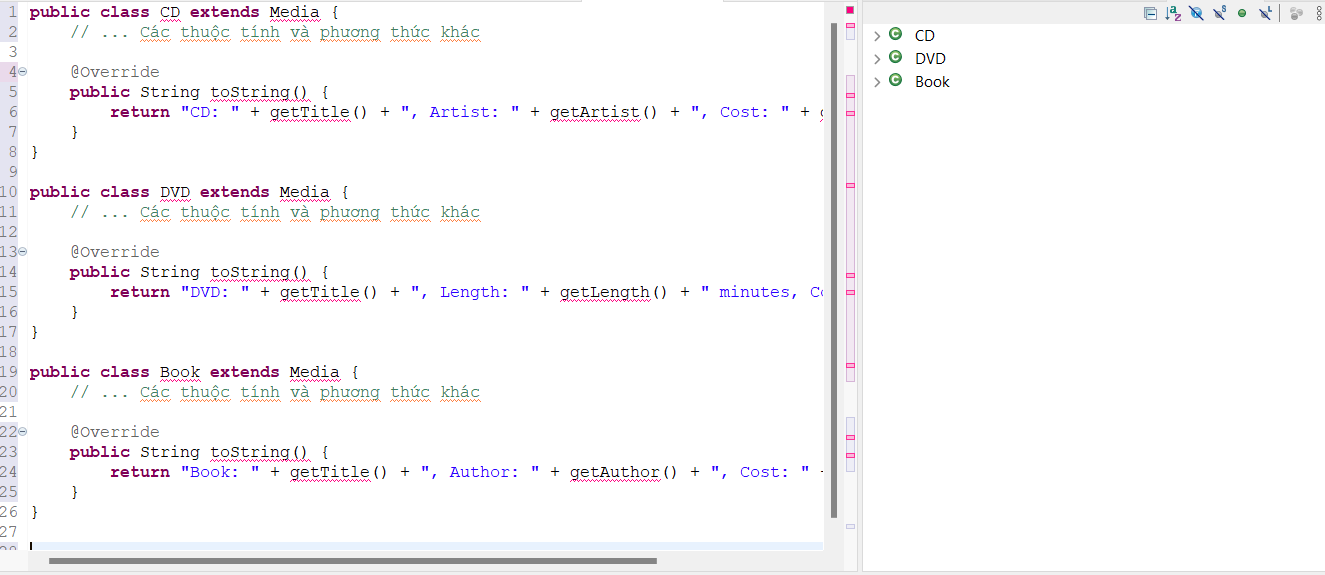
-Nếu đối tượng truyền không phải là phiên bản của Media hoặc Track, thì phương thức Equals() sẽ trả về false.

# 11. Polymorphism with toString() method

Đối với từng loại phương tiện, triển khai phương thức toString() để in ra thông tin của đối tượng. Khi gọi phương thức này, tùy thuộc vào loại đối tượng, toString() tương ứng sẽ được thực hiện.

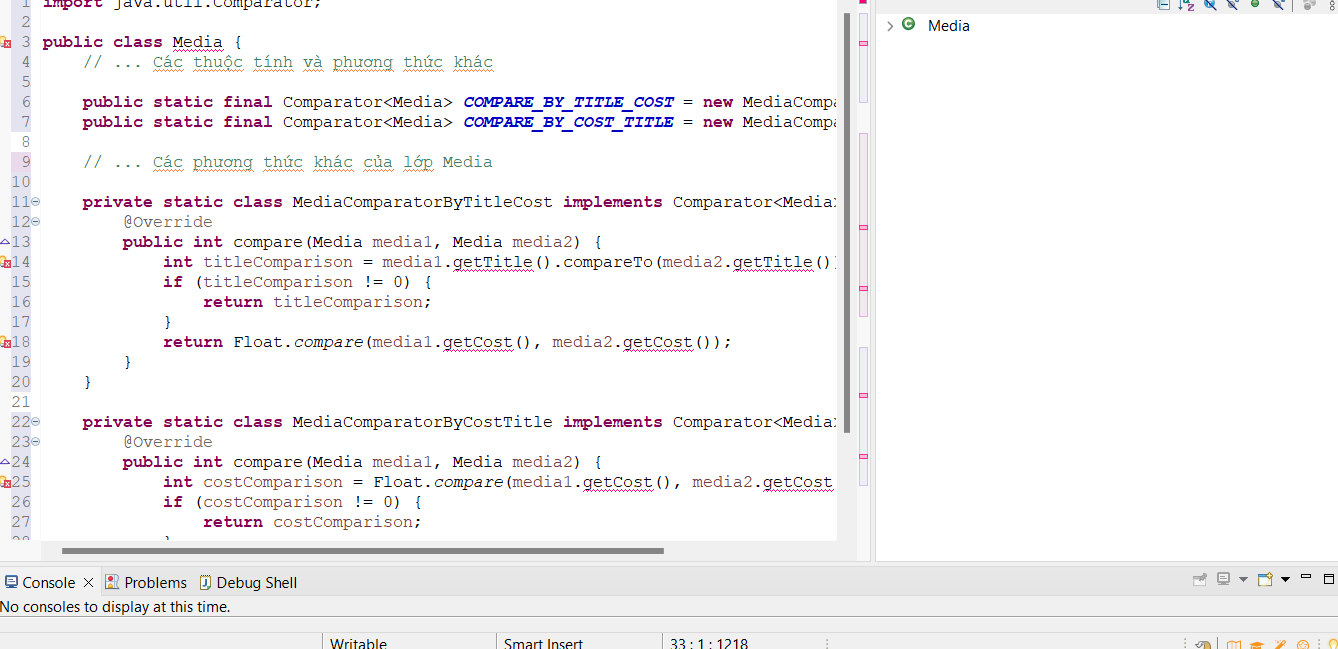
- Tạo một ArrayList of Media, sau đó thêm một số media (CD, DVD hoặc Book) vào danh sách.

- Lặp lại danh sách và in ra thông tin của media bằng phương thức toString(). Quan sát

12. Sort media in the cart

Sắp xếp theo tiêu đề: hệ thống hiển thị tất cả các phương tiện theo thứ tự bảng chữ cái theo tiêu đề. Trong trường hợp có cùng tiêu đề thì phương tiện có giá cao hơn sẽ được hiển thị trước.

Sắp xếp theo chi phí: hệ thống hiển thị tất cả các phương tiện theo thứ tự chi phí giảm dần. Trong trường hợp có cùng mức giá thì các phương tiện sẽ được sắp xếp theo tiêu đề (theo bảng chữ cái).



## **Câu hỏi:**

1. *Lớp nào nên triển khai giao diện Comparable?*

Lớp muốn so sánh các đối tượng của nó nên triển khai giao diện Comparable.

1. *Làm thế nào để triển khai phương thức compareTo() để phản ánh thứ tự mà chúng ta muốn?*

Phương thức compareTo() nên trả về một số nguyên âm, zero hoặc số nguyên dương, tùy thuộc vào việc đối tượng hiện tại có nhỏ hơn, bằng hoặc lớn hơn đối tượng đã xác định. Để phản ánh thứ tự mong muốn:

Đối với thứ tự theo tiêu đề rồi giá: So sánh tiêu đề, và nếu chúng bằng nhau, so sánh giá.

Đối với thứ tự theo giá rồi tiêu đề: So sánh giá, và nếu chúng bằng nhau, so sánh tiêu đề.

Đối với quy tắc đặc biệt của DVD (tiêu đề, giảm chiều dài, giá): So sánh tiêu đề, sau đó so sánh chiều dài theo thứ tự ngược lại, và cuối cùng, so sánh giá.

1. *Có thể có hai quy tắc sắp xếp cho mục (theo tiêu đề rồi giá và theo giá rồi tiêu đề) nếu sử dụng phương thức compareTo() của giao diện Comparable?*

Không, một lớp chỉ có thể có một quy tắc sắp xếp tự nhiên. Nếu muốn có các quy tắc sắp xếp khác nhau, có thể sử dụng các trình so sánh cho những quy tắc cụ thể đó.

1. *Giả sử DVDs có một quy tắc sắp xếp khác biệt so với các loại phương tiện khác (tiêu đề, giảm chiều dài, giá). Làm thế nào sẽ sửa đổi mã của mình để cho phép điều này?*

Có thể tạo một trình so sánh tùy chỉnh, ví dụ như DvdComparator, triển khai giao diện Comparator, và sử dụng nó đặc biệt cho DVDs. Trong trình so sánh này, xác định quy tắc sắp xếp cho DVDs (tiêu đề, giảm chiều dài theo thứ tự ngược lại và giá). Sau đó, sử dụng trình so sánh này khi sắp xếp các bộ sưu tập chứa DVDs. Đối với các loại phương tiện khác, có thể sử dụng sắp xếp tự nhiên mặc định hoặc một trình so sánh khác nếu cần.

# 13. Create a complete console application in the Aims class

