BÁO CÁO THỰC HÀNH LAP 4  
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

[**3. Creating the Book class** 1](#_Toc154780414)

[**4. Creating the abstract Media class** 3](#_Toc154780415)

[**5. Creating the CompactDisc class** 5](#_Toc154780416)

[5.1. Create the Disc class extending the Media class 6](#_Toc154780417)

[5.2. Create the Track class which models a track on a compact disc and will store information incuding the title and length of the track 7](#_Toc154780418)

[5.3. Open the CompactDisc class 8](#_Toc154780419)

[**6. Create the Playable interface** 10](#_Toc154780420)

[**7. Update the Cart class to work with Media** 11](#_Toc154780421)

[**8. Update the Store class to work with Media** 14](#_Toc154780422)

[**9. Constructors of whole classes and parent classes** 15](#_Toc154780423)

[**12. Sort media in the cart** 16](#_Toc154780424)

# **3. Creating the Book class**

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

● Package: hust.soict.dsai.aims.media

● Name: Book

● Access modifier: public

● Superclass: java.lang.Object

● public static void main(String[] args): do not check

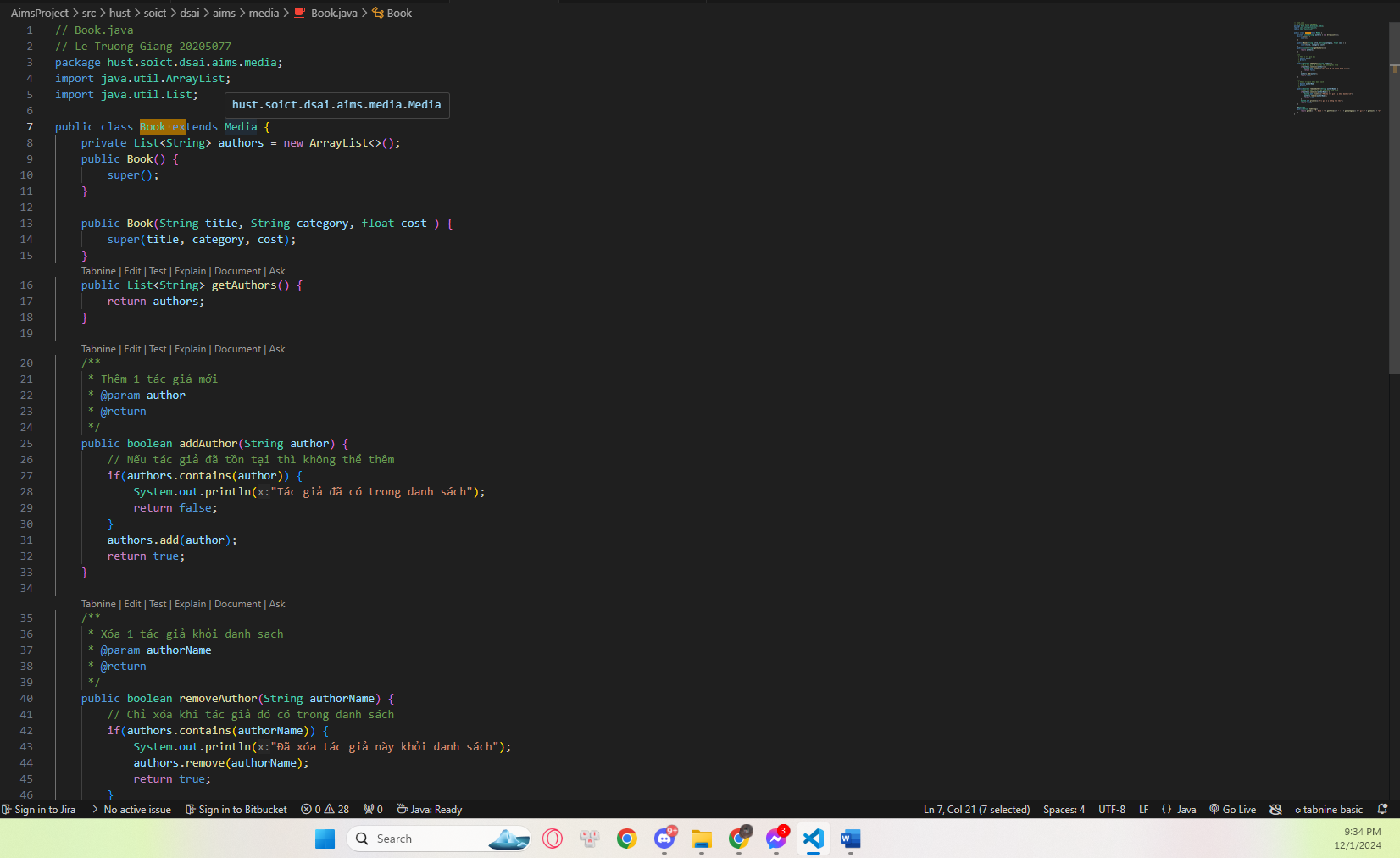
● Constructors from Superclass: Check

● 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.

Instead of typing the accessor methods for these fields, you may use the Generate Getter and Setter option in the Outline view pop-up menu (i.e., Right Click -> Source -> Generate Getters and Setters...). Note that in reality, not all attributes need to have getter and setter. We only create this when necessary. Getter and setter generator of Eclipse also let you decide with attribute will get getter or setter or both.



● Next, create addAuthor(String authorName) and removeAuthor(String authorName) for the Book class

o The addAuthor(...) method should ensure that the author is not already in the ArrayList before adding

A computer screen with text and images

Description automatically generated

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

A computer screen with text

Description automatically generated

o Reference to some useful methods of the ArrayList class

# **4. Creating the abstract Media class**

At this point, the DigitalVideoDisc and the Book classes have some fields in common namely id, title, category and cost. Here is a good opportunity to create a common superclass between the two, to eliminate the duplication of code. This process is known as refactoring. You will create an abstract class called Media which contains these fields and their associated get and set methods.

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

● You will want to 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

- Do similarly for the DigitalVideoDisc class and move it to the package hust.soict.dsai.aims.media. Remove the package hust.soict.dsai.aims.disc.

● After doing that you will see a lot of errors because of the missing fields

● Extend the Media class for both Book and DigitalVideoDisc

o public class Book extends Media

o public class DigitalVideoDisc extends Media

● Save your changes.

// Media.java

// Le Truong Giang 20205077

package hust.soict.dsai.aims.media;

import java.util.Comparator;

import java.util.Objects;

public abstract class Media {

    private static int autoIncrement = 1;

    private int id;

    private String title;

    private String category;

    private float cost;

    public Media() {

        this.id = autoIncrement;

        incrementId();

    }

    public Media(String title, String category, float cost) {

        this.id = autoIncrement;

        this.title = title;

        this.category = category;

        this.cost = cost;

        incrementId();

    }

    public static final Comparator<Media> COMPARATOR\_BY\_COST\_TITLE = new MediaComparatorByCostTitle();

    public static final Comparator<Media> COMPARATOR\_BY\_TITLE\_COST = new MediaComparatorByTitleCost();

    private void incrementId() {

        autoIncrement++;

    }

    public int getId() {

        return id;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getCategory() {

        return category;

    }

    public void setCategory(String category) {

        this.category = category;

    }

    public float getCost() {

        return cost;

    }

    public void setCost(float cost) {

        this.cost = cost;

    }

    /\*\*

     \* So sánh 2 media.Bằng nhau nếu title giong nhau

     \*

     \* @param o

     \* @return

     \*/

    @Override

    public boolean equals(Object o) {

        if (this == o)

            return true;

        if (o == null || getClass() != o.getClass())

            return false;

        Media media = (Media) o;

        return Objects.equals(title, media.title);

    }

    @Override

    public int hashCode() {

        return Objects.hash(id, title, category, cost);

    }

    /\*\*

     \* So khớp title với instance hiện tại

     \*

     \* @param title

     \* @return

     \*/

    public boolean isMatch(String title) {

        return getTitle().equals(title);

    }

    @Override

    public String toString() {

        return id + ". Media - " + title + " - " + category + " - " + cost;

    }

}

# **5. 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.

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

- The Disc class has two fields: length and director

A computer screen with a black background

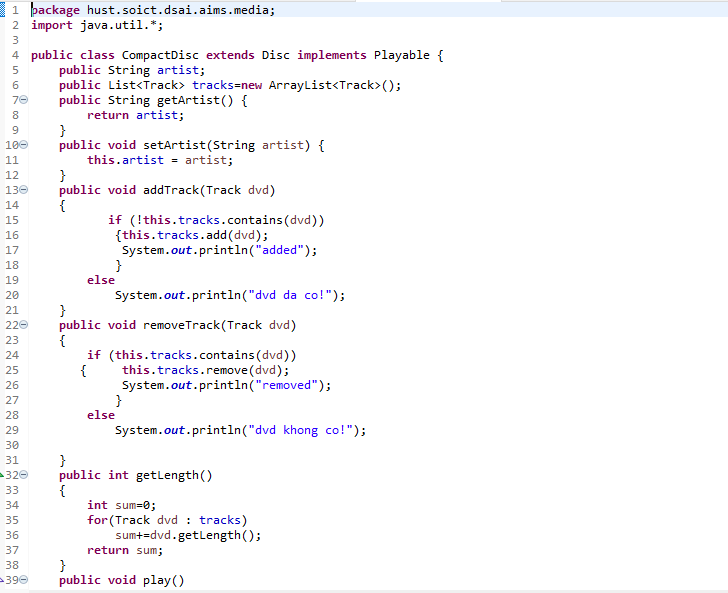
Description automatically generated

- Create getter methods for these fields

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

- Make the DigitalVideoDisc extending the Disc class. Make changes if need be.

- Create the CompactDisc extending the Disc class. Save your changes.



// CompactDisc.java

// Le Truong Giang 20205077

package hust.soict.dsai.aims.media;

import hust.soict.dsai.aims.media.model.Track;

import java.util.ArrayList;

import java.util.List;

public class CompactDisc extends Disc implements Playable {

    private String artist;

    private List<Track> tracks = new ArrayList<Track>();

    public CompactDisc() {

        super();

    }

    public CompactDisc(String artist) {

        super();

        this.artist = artist;

    }

    public CompactDisc(String artist, String title) {

        super();

        this.artist = artist;

        this.setTitle(title);

    }

    @Override

    public void play() {

        for (int i = 0; i < tracks.size(); i++) {

            tracks.get(i).play();

        }

    }

    /\*\*

     \* Thêm một track mới vào danh sách nếu không tồn tại

     \*

     \* @param track

     \* @return

     \*/

    public boolean addTrack(Track track) {

        if (tracks.contains(track)) {

            return false;

        }

        tracks.add(new Track(this.getTitle(), this.getLength()));

        return true;

    }

    /\*\*

     \* Xóa 1 track khỏi danh sách tracks nếu nó tồn tại

     \*

     \* @param track

     \* @return

     \*/

    public boolean removeTrack(Track track) {

        if (tracks.contains(track)) {

            tracks.remove(track);

            return true;

        }

        return false;

    }

    /\*\*

     \* Tính tổng thời gian của tất cả các track trong CD

     \*

     \* @return

     \*/

    public int getLength() {

        int result = 0;

        for (int i = 0; i < tracks.size(); i++) {

            result += tracks.get(i).getLength();

        }

        return result;

    }

    @Override

    public String toString() {

        return getId() + ". CD - " + getTitle() + " - tác giả: " + artist + " - độ dài: " + getLength();

    }

}

## 5.2. 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.

- Save your changes

// Track.java

// Le Truong Giang 20205077

package hust.soict.dsai.aims.media.model;

import hust.soict.dsai.aims.media.Playable;

import java.util.Objects;

public class Track implements Playable {

    private String title;

    private int length;

    public Track() {

    }

    public Track(String title, int length) {

        this.title = title;

        this.length = length;

    }

    @Override

    public void play() {

        System.out.println("Playing Track: " + this.getTitle());

        System.out.println("Track length: " + this.getLength());

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public int getLength() {

        return length;

    }

    public void setLength(int length) {

        this.length = length;

    }

    @Override

    public boolean equals(Object o) {

        if (this == o)

            return true;

        if (o == null || getClass() != o.getClass())

            return false;

        Track track = (Track) o;

        return length == track.length && Objects.equals(title, track.title);

    }

    @Override

    public int hashCode() {

        return Objects.hash(title, length);

    }

}

## 5.3. Open the CompactDisc class

- Add 2 fields to this class:

● a String as artist

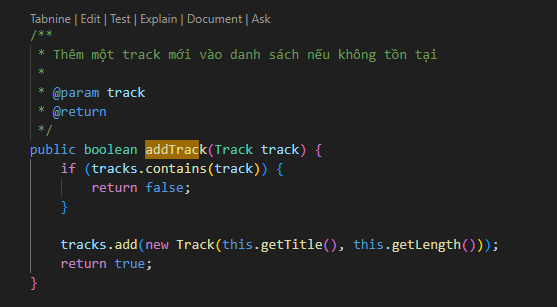
● an ArrayList of Track as tracks

- 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

A screen shot of a computer program

Description automatically generated

- Create the getLength() method

A computer screen with text and numbers

Description automatically generated

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

- Save your changes

# **6. Create the Playable interface**

The Playable interface is created to allow classes to indicate that they implement a play() method. You can apply Release Flow here by creating a topic branch for implementing Playable interface.

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

A screen shot of a computer program

Description automatically generated

- Save your changes

- Implement the Playable with CompactDisc, DigitalVideoDisc and Track

● For each of these classes CompactDisc and DigitalVideoDisc, edit the class description to include the keywords implements Playable, after the keyword extends Disc

● For the Track class, insert the keywords implements Playable after the keywords public class Track

- Implement play() for DigitalVideoDisc and Track

● Add the method play() to these two classes

● In the DigitalVideoDisc, simply print to screen:

public void play() {

System.out.println("Playing DVD: " + this.getTitle());

System.out.println("DVD length: " + this.getLength());

}

● Similar additions with the Track class

- 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

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

You must now update the Cart class to accept not only DigitalVideoDisc but also Book and CompactDisc. Currently, the Cart class has methods:

● addDigitalVideoDisc()

● removeDigitalVideoDisc().

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.

- Remove the itemsOrdered array, as well as its add and remove methods.

a. From the Package Explorer view, expand the project

b. Double-click Cart.java to open it in the editor

c. In the Outline view, select the itemsOrdered array and the methods addDigitalVideoDisc() and removeDigitalVideoDisc() and hit the Delete key

d. Click Yes when prompted to confirm the deletion

e. Recreate the itemsOrdered field, this time as an object ArrayList instead of an array.

- The qtyOrdered field is no longer needed since it was used to track the number of DigitalVideoDiscs in the itemsOrdered array, so remove it and its accessor and mutator (if exist).

- Add the itemsOrdered to the Cart class

f. To create this field, type the following code in the Cart class, in place of the itemsOrdered array declaration that you deleted:

private ArrayList<Media> itemsOrdered = new ArrayList<Media>();

- Note that you should import the java.util.ArrayList in the Cart class

g. A quicker way to achieve the same affect is to use the Organize Imports feature within Eclipse

h. Right-click anywhere in the editor for the Cart class and select Source -> Organize Imports (Or Ctrl+Shift+O). This will insert the appropriate import statements in your code.

i. Save your class

- Create addMedia() and removeMedia() to replace addDigitalVideoDisc() and removeDigitalVideoDisc()

- Update the totalCost() method

package hust.soict.dsai.aims.cart;// hust.soict.dsai.aims.cart.Cart.java

// Le Truong Giang 20205077

import hust.soict.dsai.aims.media.DigitalVideoDisc;

import hust.soict.dsai.aims.media.Media;

import java.util.ArrayList;

import java.util.Arrays;

public class Cart {

    public static final int MAX\_NUMBERS\_ORDERD = 20;

    private ArrayList<Media> itemsOrdered = new ArrayList<Media>();

    /\*\*

     \* Thêm nhiều media vào giỏ hàng

     \*

     \* @param medias

     \* @return

     \*/

    public boolean addMedia(Media... medias) {

        if (itemsOrdered.size() + medias.length >= MAX\_NUMBERS\_ORDERD) {

            System.out.println(

                    "Bạn chỉ được phép thêm tối đa: " + (MAX\_NUMBERS\_ORDERD - itemsOrdered.size()) + " vào giỏ hàng");

            return false;

        }

        for (Media media : medias) {

            if (!itemsOrdered.contains(media)) {

                itemsOrdered.add(media);

            } else {

                System.out.println("Sản phẩm " + media.getTitle() + " đã tồn tại trong giỏ hàng");

            }

        }

        return true;

    }

    /\*\*

     \* Tìm dvd bằng id

     \*

     \* @param id

     \*/

    public int searchProduct(int id) {

        int index = -1;

        for (int i = 0; i < itemsOrdered.size(); i++) {

            Media item = itemsOrdered.get(i);

            if (item.getId() == id) {

                return i;

            }

        }

        return index;

    }

    /\*\*

     \* Tìm dvd bằng title

     \*

     \* @param title

     \*/

    public int searchProduct(String title) {

        int index = -1;

        for (int i = 0; i < itemsOrdered.size(); i++) {

            Media item = itemsOrdered.get(i);

            if (item.isMatch(title)) {

                return i;

            }

        }

        return index;

    }

    /\*\*

     \* Xóa 1 media nào đó khỏi giỏ hàng nếu sản phẩm đó tồn tại

     \*

     \* @param id

     \* @return

     \*/

    public void removeMedia(int id) {

        itemsOrdered.remove(id);

    }

    /\*\*

     \* Tính tổng số tiền giỏ hàng

     \*

     \* @return float

     \*/

    public float totalCost() {

        float cartTotal = 0f;

        for (int i = 0; i < itemsOrdered.size(); i++) {

            cartTotal += itemsOrdered.get(i).getCost();

        }

        return cartTotal;

    }

    /\*\*

     \* In các DVD có trong cart

     \*

     \* @param

     \* @return void

     \*/

    public void printCart() {

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CART\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        for (int i = 0; i < itemsOrdered.size(); i++) {

            Media item = itemsOrdered.get(i);

            System.out.println(item.toString());

        }

        System.out.printf("Total cost: %f\n", totalCost());

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

    }

    /\*\*

     \* Lấy danh sách sản phẩm trong cart

     \*

     \* @return

     \*/

    public ArrayList<Media> getItemsOrdered() {

        return itemsOrdered;

    }

    /\*\*

     \* Lấy số sản phẩm có trong giỏ hàng hiện tại

     \*

     \* @return

     \*/

    public int getCountItem() {

        return itemsOrdered.size();

    }

}

CartController.java:  
// CartController.java

// Le Truong Giang 20205077

package hust.soict.dsai.aims.cart.controller;

import hust.soict.dsai.aims.cart.Cart;

import hust.soict.dsai.aims.data.InitData;

import hust.soict.dsai.aims.media.DigitalVideoDisc;

import hust.soict.dsai.aims.media.Media;

import java.util.Collections;

import java.util.Scanner;

public class CartController {

    private static Cart myCart = InitData.myCart;

    /\*\*

     \* In các sản phẩm có trong Cart

     \*/

    public static void printCart() {

        myCart.printCart();

    }

    /\*\*

     \* Lọc sản phẩm theo ID hoặc tiêu đề

     \*/

    public static void filterItem() {

        while (true) {

            System.out.println("Lọc theo: ");

            System.out.println("0. Thoát chế độ lọc");

            System.out.println("1. ID");

            System.out.println("2. Tiêu đề");

            System.out.println("Nhập lựa chọn của bạn: 1-2");

            Scanner sc = new Scanner(System.in);

            int type = sc.nextInt();

            int index = -1;

            // Lọc theo ID

            if (type == 1) {

                System.out.print("Nhập ID cần tìm: ");

                int id = sc.nextInt();

                // Tìm index của sản phẩm theo ID

                index = myCart.searchProduct(id);

                if (index == -1) {

                    System.out.println("Không tìm thấy sản phẩm này trong giỏ hàng của bạn");

                    return;

                }

            } else if (type == 2) {

                // Lọc theo tiêu đề

                System.out.print("Nhập title cần tìm: ");

                sc.nextLine();

                String title = sc.nextLine();

                System.out.println(title);

                // Tìm index của sản phẩm theo tiêu đề

                index = myCart.searchProduct(title);

            } else if (type == 0) {

                break;

            } else {

                System.out.println("Lựa chọn của bạn không hợp lệ.Thử lại sau");

                return;

            }

            if (index == -1) {

                System.out.println("Không tìm thấy sản phẩm này trong giỏ hàng của bạn");

                return;

            }

            System.out.println("Thông tin sản phẩm: ");

            // Nếu tìm thấy thì in ra

            System.out.println(myCart.getItemsOrdered().get(index).toString());

        }

    }

    /\*\*

     \* Sắp xếp sản phẩm theo tiêu đề hoặc ID

     \*/

    public static void sortItem() {

        System.out.println("Chọn phương án sắp xếp");

        System.out.println("1. Theo tiêu đề");

        System.out.println("2. Theo ID");

        Scanner sc = new Scanner(System.in);

        int type = sc.nextInt();

        if (type == 1) {

            Collections.sort(myCart.getItemsOrdered(), Media.COMPARATOR\_BY\_COST\_TITLE);

        } else if (type == 2) {

            Collections.sort(myCart.getItemsOrdered(), Media.COMPARATOR\_BY\_TITLE\_COST);

        } else {

            System.out.println("Lựa chọn không hợp lệ");

            return;

        }

        System.out.println("Sắp xếp thành công");

    }

    /\*\*

     \* Xóa một sản phẩm khỏi giỏ hàng bởi tiêu đề

     \*/

    public static void removeItem() {

        System.out.println("Nhập tiêu đề sản phẩm cần xóa");

        Scanner sc = new Scanner(System.in);

        String title = sc.nextLine();

        // Tìm index của sản phẩm cần xóa theo tiêu đề

        int index = myCart.searchProduct(title);

        if (index == -1) {

            System.out.println("Sản phẩm này không tồn tại trong giỏ hàng của bạn");

            return;

        }

        myCart.removeMedia(index);

        System.out.println("Đã xóa sản phẩm " + title + " khỏi giỏ hàng của bạn");

    }

    /\*\*

     \* Play sản phẩm

     \*/

    public static void playItem() {

        System.out.println("Nhập tiêu đề sản phẩm cần play");

        Scanner sc = new Scanner(System.in);

        String title = sc.nextLine();

        int index = myCart.searchProduct(title);

        if (index == -1) {

            System.out.println("Sản phẩm này không tồn tại trong giỏ hàng của bạn");

            return;

        }

        if (myCart.getItemsOrdered().get(index) instanceof DigitalVideoDisc) {

            DigitalVideoDisc item = (DigitalVideoDisc) myCart.getItemsOrdered().get(index);

            item.play();

        } else {

            System.out.println("Sản phẩm này không phải là DVD, không hỗ trợ play");

        }

    }

    /\*\*

     \* Order, thanh toán

     \*/

    public static void order() {

        // Xóa giỏ hàng

        myCart.getItemsOrdered().removeAll(myCart.getItemsOrdered());

        System.out.println("Tạo đơn hàng thành công !");

    }

}

CartMenuController.java:  
package hust.soict.dsai.aims.cart.controller;

import hust.soict.dsai.aims.store.controller.StoreController;

import java.util.Scanner;

public class CartMenuController {

    public static int handleMenuCart() {

        int back = 0;

        while(true) {

            back = 0;

            cartMenu();

            System.out.print("Vui lòng chọn: ");

            Scanner sc= new Scanner(System.in);

            int chosen = sc.nextInt();

            switch (chosen) {

                case 0:

                    back = 1;

                    break;

                case 1:

                    CartController.filterItem();

                    break;

                case 2:

                    CartController.sortItem();

                    break;

                case 3:

                    CartController.removeItem();

                    break;

                case 4:

                    CartController.playItem();

                    break;

                case 5:

                    CartController.order();

                    break;

            }

            if(back == 1)

                return back;

        }

    }

    public static void cartMenu() {

        System.out.println("Lựa chọn: ");

        System.out.println("--------------------------------");

        System.out.println("1. Lọc sản phẩm trong giỏ hàng");

        System.out.println("2. Sắp xếp danh sách sản phẩm trong giỏ hàng");

        System.out.println("3. Xóa media khỏi cart");

        System.out.println("4. Play media");

        System.out.println("5. Thanh toán");

        System.out.println("0. Quay lại");

        System.out.println("--------------------------------");

        System.out.println("Vui lòng lựa chọn: 0-1-2-3-4-5");

    }

}

# **8. 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()

// Store.java

// Le Truong Giang 20205077

package hust.soict.dsai.aims.store;

import hust.soict.dsai.aims.media.DigitalVideoDisc;

import hust.soict.dsai.aims.media.Media;

import java.util.ArrayList;

public class Store {

    private ArrayList<Media> itemsInStore = new ArrayList<Media>();

    /\*\*

     \* Thêm 1 hay nhiều media vào cửa hàng

     \*

     \* @param args

     \*/

    public void addMedia(Media... args) {

        for (Media media : args) {

            // System.out.printf("Thêm thành công sản phẩm %s với id: %d\n",

            // media.getTitle(), media.getId());

            itemsInStore.add(media);

        }

    }

    /\*\*

     \* Xóa Media bằng index

     \*

     \* @param id

     \*/

    public void removeMedia(int id) {

        itemsInStore.remove(id);

    }

    /\*\*

     \* Xóa sản phẩm theo title

     \*

     \* @param title

     \*/

    public void removeMedia(String title) {

        boolean isRemoved = false;

        for (int i = 0; i < itemsInStore.size(); i++) {

            // Nếu tìm thấy sản phẩm thì xóa

            Media item = itemsInStore.get(i);

            if (item.isMatch(title)) {

                itemsInStore.remove(i);

                isRemoved = true;

                System.out.printf("Đã xóa sản phẩm có title: %s\n", title);

                return;

            }

        }

        if (isRemoved == false) {

            System.out.printf("Không sản phẩm nào có Title: %s\n", title);

        }

    }

    /\*\*

     \* Kiểm tra xem sản phẩm đã tồn tại hay chưa, trả về index của sản phẩm đó

     \*

     \* @param title

     \* @return

     \*/

    public int isExist(String title) {

        int result = -1;

        for (int i = 0; i < itemsInStore.size(); ++i) {

            if (itemsInStore.get(i).getTitle().equals(title)) {

                result = i;

                break;

            }

        }

        return result;

    }

    /\*\*

     \* In danh sách sản phẩm

     \*/

    public void printStore() {

        for (int i = 0; i < itemsInStore.size(); i++) {

            System.out.println(itemsInStore.get(i).toString());

        }

    }

    /\*\*

     \* Lấy một sản phẩm nào đó trong cart

     \*/

    public Media getItemByIndex(int index) {

        return itemsInStore.get(index);

    }

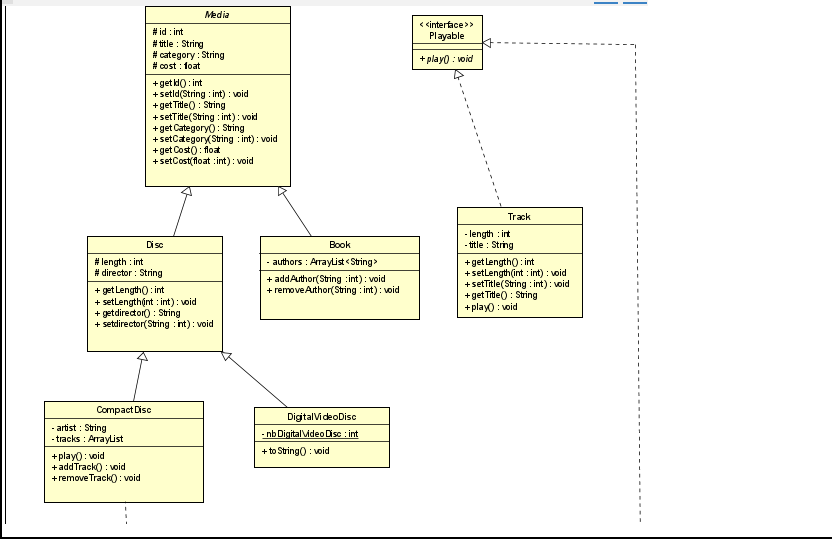
}

# **9. 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.

- Which classes are aggregates of other classes? Checking all constructors of whole classes if they initialize for their parts?

- Write constructors for parent and child classes. Remove redundant setter methods if any



# **12. Sort media in the cart**

As mentioned before, when seeing the current cart, the user can sort the items in the cart by title or by cost:

o 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.

o 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).

Here, we can use Comparator to allow multiple sorting ways of Media:

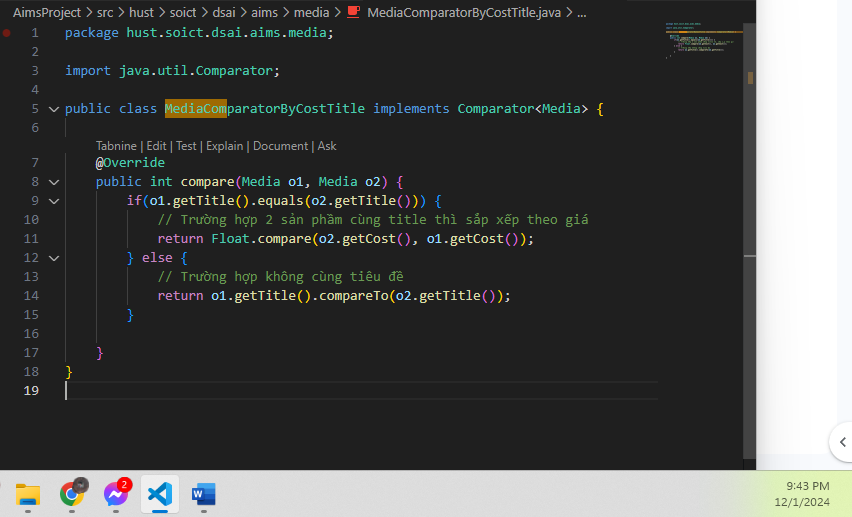
Note: The Comparator interface is a comparison function, which impose a total ordering on some

collection of objects. Comparators can be passed to a sort method (such as Collections.sort())

to allow precise control over the sort order.

Please open the Java docs to see the information of this interface.

- Create two classes of comparators, one for each type of ordering

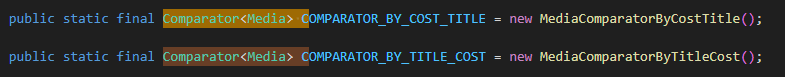
A screen shot of a computer screen

Description automatically generated

- Implement the compare()method of each comparator class to reflect the ordering that we want, either

by title then cost, or by cost then title. You may utilize the method

Comparator.thenComparing()to sort using multiple fields.

- 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)

**Question:** Alternatively, to compare items in the cart, instead of using Comparator, we can use the Comparable interface and override the compareTo()method. You can refer to the Java docs to see the information of this interface.

Suppose we are taking this Comparable interface approach.

- What class should implement the Comparable interface?

Answer: Lớp Media nên implement giao diện Comparable. Điều này giúp định nghĩa cách so sánh các đối tượng Media dựa trên tiêu chí mặc định (ví dụ: so sánh theo title, hoặc bất kỳ thuộc tính nào cần thiết).

- In those classes, how should you implement the compareTo()method be to reflect the ordering that we want?

Answer: Trong lớp Media, phương thức compareTo() nên được triển khai như sau để phản ánh cách sắp xếp theo tiêu chí yêu cầu (ví dụ: so sánh theo title trước, nếu bằng nhau thì tiếp tục so sánh theo cost):

- Can we have two ordering rules of the item (by title then cost and by cost then title) if we use this Comparable interface approach?

Answer:

Không, giao diện Comparable chỉ cho phép định nghĩa một quy tắc so sánh duy nhất. Nếu cần hai quy tắc sắp xếp khác nhau .Chúng ta cần sử dụng giao diện Comparator để định nghĩa các bộ so sánh riêng biệt.

- Suppose the DVDs has a different ordering rule from the other media types, that is by title, then decreasing length, then cost. How would you modify your code to allow this?

Answer: Nếu DVD có quy tắc sắp xếp khác biệt, bạn có thể tạo một lớp con của Media (ví dụ: DVD) và override phương thức compareTo() trong lớp này để phản ánh quy tắc sắp xếp đặc thù cho DVD