Student name: Lê Ngọc Anh Vũ

Student ID: 20236014

REPORT LAB 03

4. Update the `Cart` class and `CartTest` class

Write new methods to implement the following functions:

- Create a new method to print the list of ordered items of a cart, the price of each item, and the total price. Format the outline as below:

```
**********CART**********
Ordered Items:
1. DVD - [Title] - [category] - [Director] - [Length]: [Price] $
2. DVD - [Title] - ...
Total cost: [total cost]
**********
```

Suggestion: Write a **toString()** method for the **DigitalVideoDisc** class. What should be the return type of this method?

```java
public void print() {
    System.out.println("**********CARD**********");
    System.out.println("Ordered Items:");
    for(int i = 0; i < qtyOrdered; i++) {
        System.out.printf("%d. ", i+1);
        System.out.println(itemsOrdered[i]);
    }
    System.out.printf("Total cost: %f\n", totalCost());
    System.out.println("**********");
}
```

toString() method for the DigitalVideoDisc class:

```java
@Override
public String toString() {
    return "DVD " + getTitle() + " - " + getCategory() + " - "
            + getDirector() + " - " + getLength() + ": " + getCost() + "$";
}
```

- Search for DVDs in the cart by ID and display the search results. Make sure to notify the user if no match is found.

```java
public void searchByCartID(int id) {
    for(int i = 0; i < qtyOrdered; i++) {
        if(itemsOrdered[i].getId() == id) {
            System.out.printf("%d - ", itemsOrdered[i].getId());
            System.out.println(itemsOrdered[i]);
            return;
        }
    }
    System.out.printf("No mathch is found for this ID = %d\n", id);
}
```

- Search for DVDs in the cart by title and print the results. Make sure to notify the user if no match is found. ***Refer to the problem statement in Lab02 for the matching rule(Section 4).***

  Suggestion: write a **`boolean isMatch(String title)`** method in the **`DigitalVideoDisc`** which finds out if the corresponding disk is a match given the title.

A Boolean isMatch(String title) method in the DigitalVideoDisc class:

```
27    public boolean isMatch(String title) {
28        return getTitle().equals(title);
29    }
```

The method to search for DVDs in the cart by title and print results:

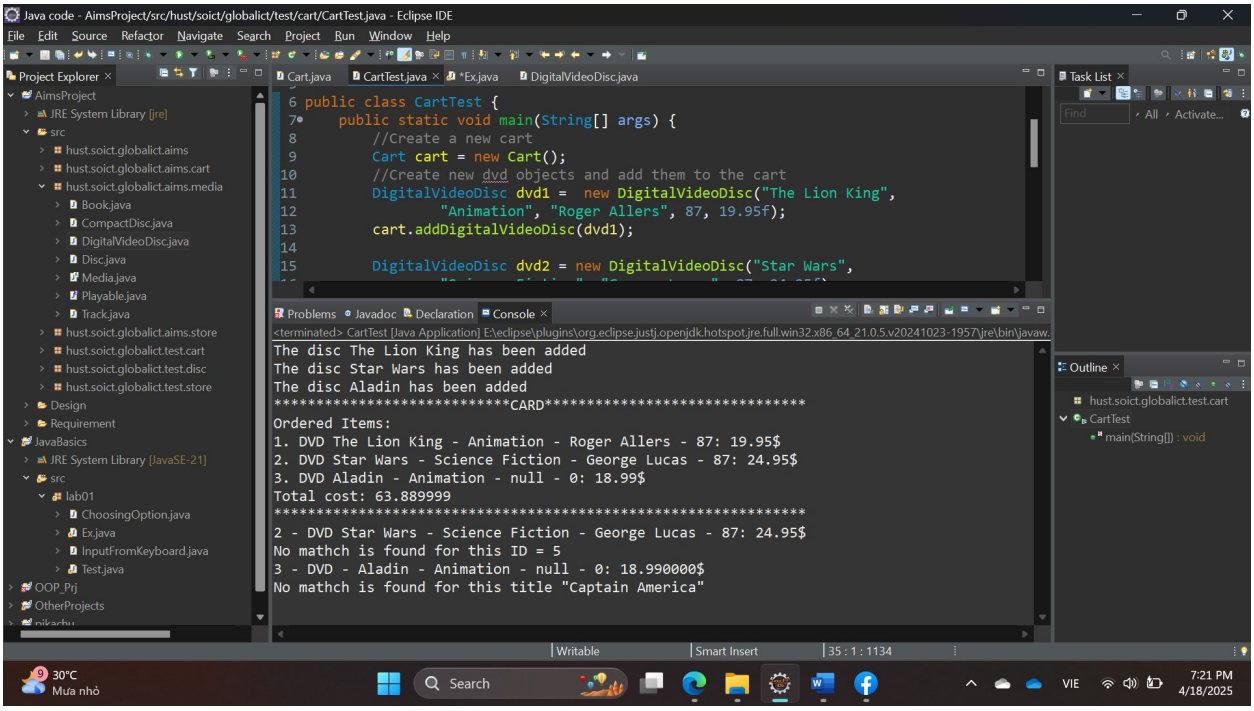
```
114    public void searchByTitle(String title) {
115        for(int i = 0; i < qtyOrdered; i++) {
116            if(itemsOrdered[i].isMatch(title)) {
117                System.out.printf("%d - DVD - %s - %s - %s - %d: %f$\n",
118                        itemsOrdered[i].getId(),
119                        itemsOrdered[i].getTitle(),
120                        itemsOrdered[i].getCategory(),
121                        itemsOrdered[i].getDirector(),
122                        itemsOrdered[i].getLength(),
123                        itemsOrdered[i].getCost());
124                return;
125            }
126        }
127        System.out.printf("No mathch is found for this title \"%s\"\n", title);
128    }
129 }
```

- In the **`CartTest`** class, write codes to test all methods you have written in this exercise. You should create sample DVDs and carts, like in this code snippet:

The source code is below:

```java
public class CartTest {
    public static void main(String[] args) {
        //Create a new cart
        Cart cart = new Cart();
        //Create new dvd objects and add them to the cart
        DigitalVideoDisc dvd1 =  new DigitalVideoDisc("The Lion King",
                "Animation", "Roger Allers", 87, 19.95f);
        cart.addDigitalVideoDisc(dvd1);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
                "Science Fiction", "George Lucas", 87, 24.95f);
        cart.addDigitalVideoDisc(dvd2);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
                "Animation", 18.99f);
        cart.addDigitalVideoDisc(dvd3);

        //Test the print method
        cart.print();
        //To-do: Test the search methods here
        // Search dvd by id that has a match
        cart.searchByCartID(2);
        // Search dvd by id that has no match
        cart.searchByCartID(5);
        // Search dvd by title that has a match
        cart.searchByTitle("Aladin");
        // Search dvd by title that has no match
        cart.searchByTitle("Captain America");
    }
```

The result:



5. Implement the **Store** class

- Create a **Store** class inside the package: **hust.soict.dsai.aims.store**, which contains one attribute **itemsInStore[]** – an array of DVDs available in the store.
- To add and remove DVDs from the store, implement two methods called **addDVD** and **removeDVD**

```java
package hust.soict.globalict.aims.store;

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

public class Store {

    public static final int MAX_NUMBERS_ORDERED = 100;
    private DigitalVideoDisc itemsInStore[] = new DigitalVideoDisc[MAX_NUMBERS_ORDERED];
    private int numDVDs = 0;

    public void addDVD(DigitalVideoDisc dvd) {
        if(numDVDs < MAX_NUMBERS_ORDERED) {
            itemsInStore[numDVDs] = dvd;
            numDVDs++;
            System.out.println("The disc " + dvd.getTitle() + " has been added in store");
        }
        else {
            System.out.println("The store is almost full");
        }
    }
```

```java
    public void removeDVD(DigitalVideoDisc dvd) {
        int idx;
        for(idx = 0; idx < numDVDs; idx++) {
            if(dvd == itemsInStore[idx]) {
                break;
            }
        }

        if(idx == numDVDs) {
            System.out.println("Cannot find disc " + dvd.getTitle() + " in the store");
            return;
        }

        for(int i = idx; i < numDVDs - 1; i++) {
            itemsInStore[i] = itemsInStore[i+1];
        }
        numDVDs--;
        System.out.println("The disc " + dvd.getTitle() + " has been removed from store");
    }
}
```

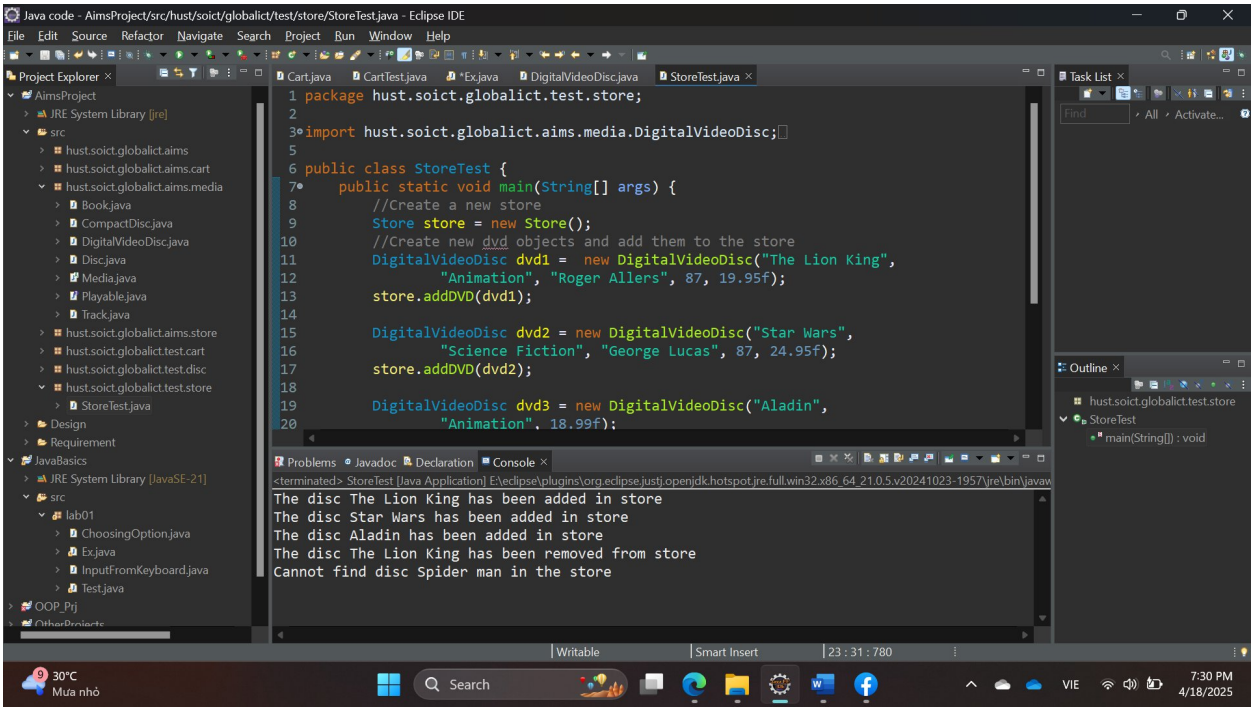
- Test these two methods in the **StoreTest** class.

As a result, you should have two classes: the **Store** class and the **StoreTest** class inside the **hust.soict.dsai.aims.store** package.

The source code is below:

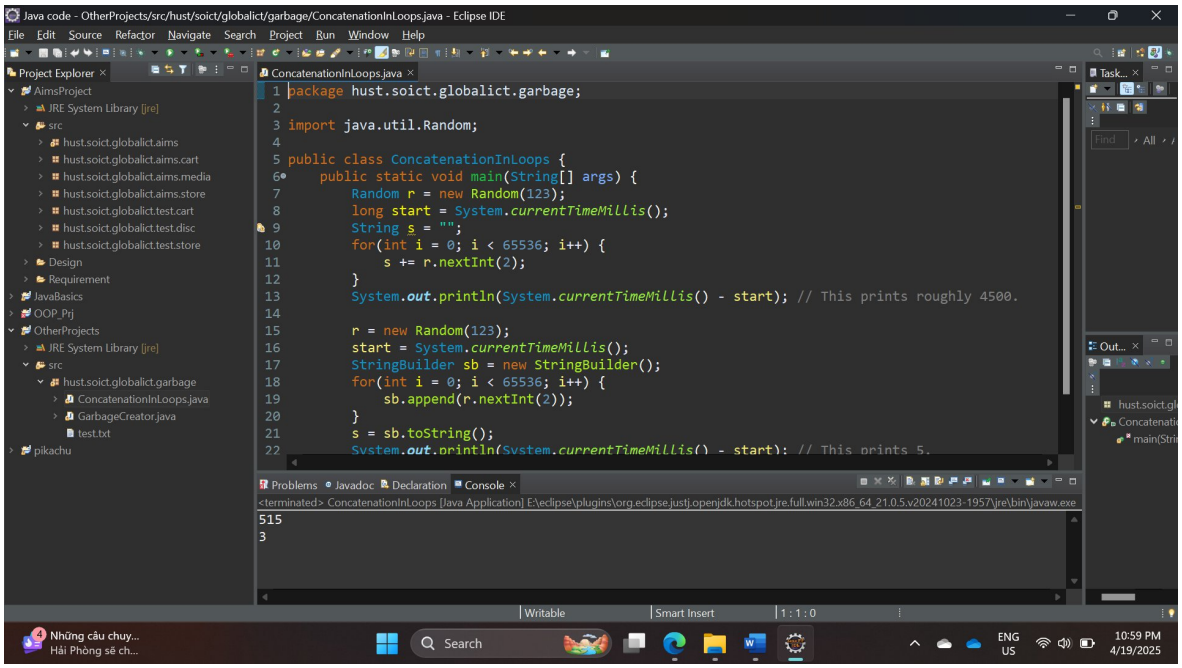
```java
package hust.soict.globalict.test.store;

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

public class StoreTest {
    public static void main(String[] args) {
        //Create a new store
        Store store = new Store();
        //Create new dvd objects and add them to the store
        DigitalVideoDisc dvd1 =  new DigitalVideoDisc("The Lion King",
                "Animation", "Roger Allers", 87, 19.95f);
        store.addDVD(dvd1);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
                "Science Fiction", "George Lucas", 87, 24.95f);
        store.addDVD(dvd2);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
                "Animation", 18.99f);
        store.addDVD(dvd3);
        //Remove dvd objects from store
        store.removeDVD(dvd1);
        //Remove dvd objects that are not available in the store
        DigitalVideoDisc dvd4 = new DigitalVideoDisc("Spider man",
                "Science Fiction", 30.07f);
        store.removeDVD(dvd4);
    }
}
```

The result is:

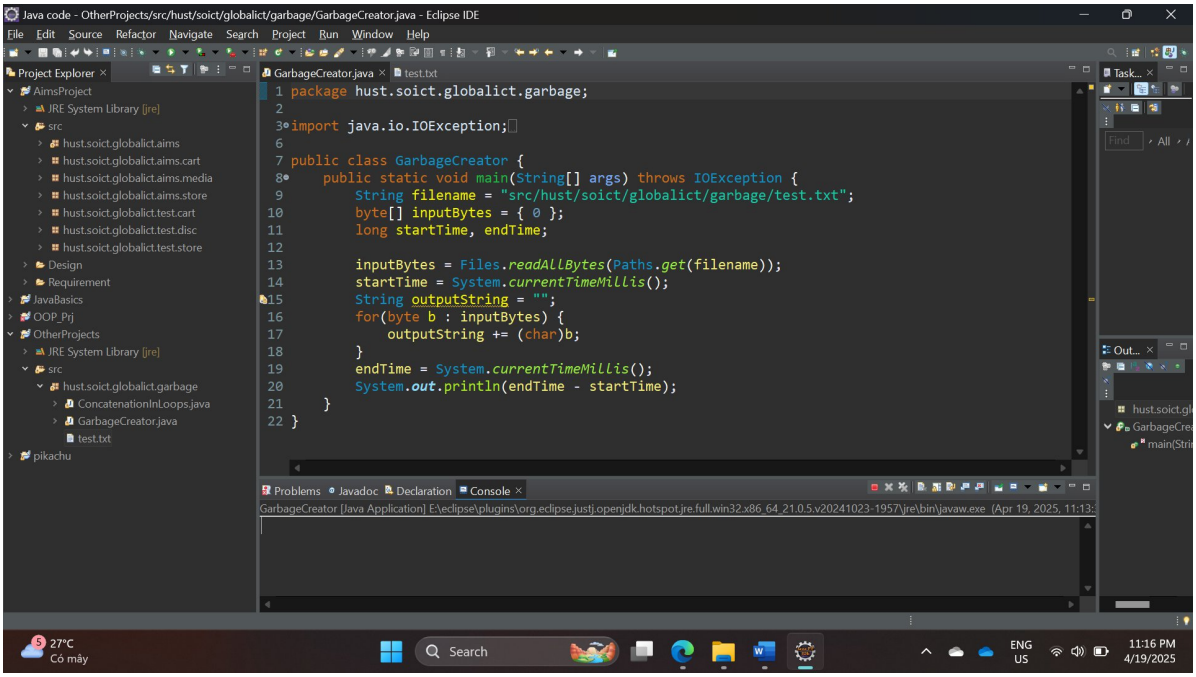


6. `String`, `StringBuilder` and `StringBuffer`

- In the **`OtherProjects`** project, create a new package **`hust.soict.dsai.garbage`**. We work with this package in this exercise.
- Create a new class **`ConcatenationInLoops`** to test the processing time to construct **`String`** using **`+`** operator, **`StringBuffer`** and **`StringBuilder`**.



- Create a new class **`GarbageCreator`**. Create "garbage" as much as possible and observe when you run a program (it should let the program hang or even stop working when there is too much "garbage"). Write another class **`NoGarbage`** to solve the problem.

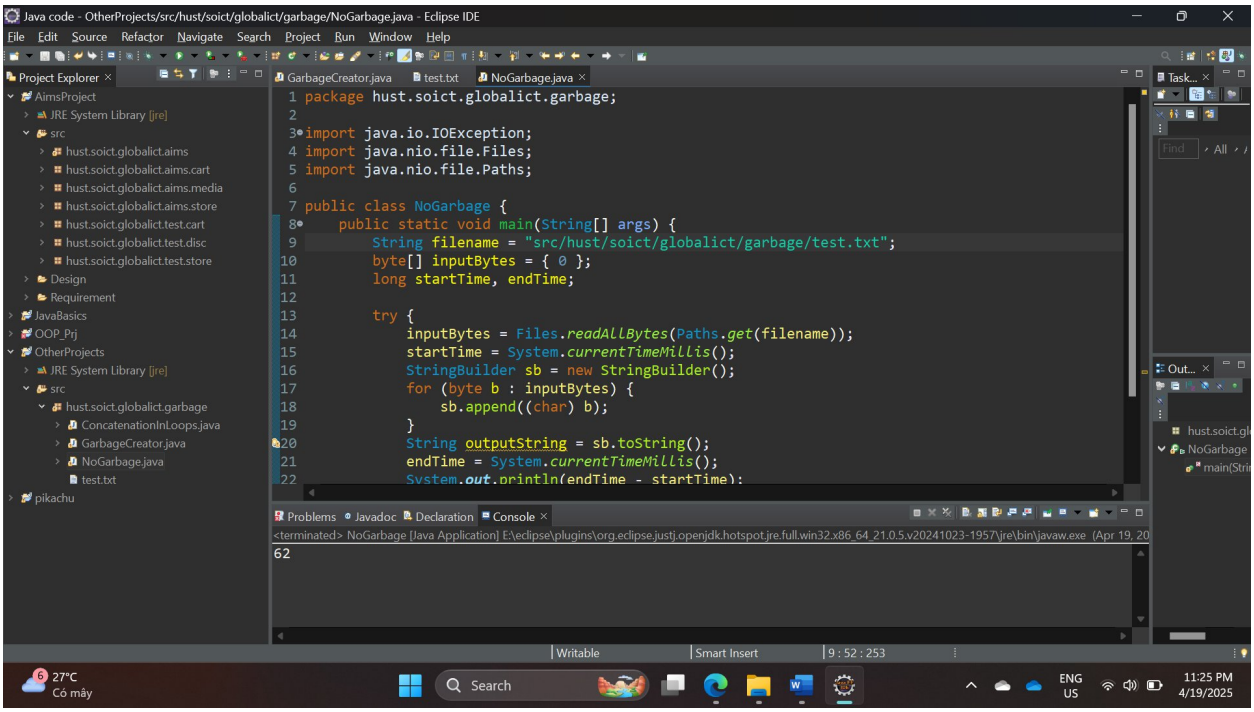


In the above picture, we can see that the program must hang out in a long time and has not printed the result yet. This is because of the large amount of “garbage”.

For NoGarbage class:

```java
package hust.soict.globalict.garbage;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class NoGarbage {
    public static void main(String[] args) {
        String filename = "src/hust/soict/globalict/garbage/test.txt";
        byte[] inputBytes = { 0 };
        long startTime, endTime;

        try {
            inputBytes = Files.readAllBytes(Paths.get(filename));
            startTime = System.currentTimeMillis();
            StringBuilder sb = new StringBuilder();
            for (byte b : inputBytes) {
                sb.append((char) b);
            }
            String outputString = sb.toString();
            endTime = System.currentTimeMillis();
            System.out.println(endTime - startTime);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

The result is:



8. Implementation of 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 which attribute will get getter or setter or both.

```java
import java.util.ArrayList;

public class Book{

    private int id;
    private String title;
    private String category;
    private float cost;
    private List<String> authors = new ArrayList<String>();

    public Book() {
        // TODO Auto-generated constructor stub
    }

    public int getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getCategory() {
        return category;
    }

    public float getCost() {
        return cost;
    }
```

- Next, 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

```
34•    public void addAuthor(String authorName) {
35         for(int i = 0; i < authors.size(); i++) {
36             if(authors.get(i).equals(authorName)) {
37                 System.out.printf("This name of author \"%s\" has already added!\n", authorName);
38                 return;
39             }
40         }
41         authors.add(authorName);
42         System.out.printf("This name of author \"%s\" has added successfully!\n", authorName);
43     }
```

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

```
45•    public void removeAuthor(String authorName) {
46         if(authors.contains(authorName)) {
47             authors.remove(authorName);
48             System.out.printf("This name of author \"%s\" has already removed!\n", authorName);
49             return;
50         }
51         System.out.printf("This name of author \"%s\" cannot be removed!\n", authorName);
52     }
```

9. Implementation of 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.

Please follow these steps to 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)

```java
package hust.soict.globalict.aims.media;

public abstract class Media {

    private int id;
    private String title;
    private String category;
    private float cost;

    public Media() {
        // TODO Auto-generated constructor stub
    }

    public int getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getCategory() {
        return category;
    }

    public float getCost() {
        return cost;
    }
```

```java
    public void setId(int id) {
        this.id = id;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public void setCost(float cost) {
        this.cost = cost;
    }

}
```

- 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**

```
public class Book extends Media
public class DigitalVideoDisc extends Media
```

Save your changes.

```
public class Book extends Media{
```

```
public class DigitalVideoDisc extends Media implements Playable{
```

10. Implementation of 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.

*10.1. 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.

```
package hust.soict.globalict.aims.media;

public class Disc extends Media{

    private int length;
    private String director;

    public Disc(String title) {
        super();
        setTitle(title);
    }

    public Disc(String title, String category, float cost) {
        super();
        setTitle(title);
        setCategory(category);
        setCost(cost);
    }

    public Disc(String title, String category, String director, float cost) {
        super();
        this.director = director;
        setTitle(title);
        setCategory(category);
        setCost(cost);
    }
```

```
    public Disc(String title, String category, String director, int length, float cost) {
        super();
        this.director = director;
        setTitle(title);
        setCategory(category);
        setCost(cost);
        this.length = length;
    }

    public int getLength() {
        return length;
    }

    public String getDirector() {
        return director;
    }

}
```

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

```java
package hust.soict.globalict.aims.media;

public class DigitalVideoDisc extends Disc implements Playable{

    public DigitalVideoDisc(String title) {
        super(title);
    }

    public DigitalVideoDisc(String title, String category, float cost) {
        super(title, category, cost);
    }

    public DigitalVideoDisc(String title, String category, String director, float cost) {
        super(title, category, director, cost);
    }

    public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
        super(title, category, director, length, cost);
    }

    @Override
    public String toString() {
        return "DVD " + getTitle() + " - " + getCategory() + " - "
                + getDirector() + " - " + getLength() + ": " + getCost() + "$";
    }

    public boolean isMatch(String title) {
        return getTitle().equals(title);
    }
```

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

```java
package hust.soict.globalict.aims.media;

import java.util.ArrayList;

public class CompactDisc extends Disc{

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

    public CompactDisc(String title, String artist) {
        super(title);
        this.artist = artist;
    }
}
```

*10.2. Create the **`Track`** class which models a track on a compact disc and will store information including 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

```java
package hust.soict.globalict.aims.media;

public class Track {

    private String title;
    private int length;

    public Track(String title, int length) {
        this.title = title;
        this.length = length;
    }

    public String getTitle() {
        return title;
    }
    public int getLength() {
        return length;
    }
}
```

10.3. *Update the* **`CompactDisc`** *class*

- Add 2 fields to this class:
  - a **`String`** as **`artist`**
  - an **`ArrayList`** of **`Track`** as **`tracks`**
- Make all these fields **`private`**. Create a public **`getter`** method for only **`artists`**.
- Create constructor(s) for this class. Use super() if possible.

```
1 package hust.soict.globalict.aims.media;
2
3 import java.util.ArrayList;
5
6 public class CompactDisc extends Disc implements Playable{
7
8     private String artist;
9     private List<Track> tracks = new ArrayList<Track>();
10
11    public CompactDisc(String title, String artist) {
12        super(title);
13        this.artist = artist;
14    }
15
16    public CompactDisc(String title, String category, float cost, String artist) {
17        super(title, category, cost);
18        this.artist = artist;
19    }
20
21    public CompactDisc(String title, String category, String director, float cost, String artist) {
22        super(title, category, director, cost);
23        this.artist = artist;
24    }
25
26    public CompactDisc(String title, String category, String director, int length, float cost, String artist) {
27        super(title, category, director, length, cost);
28        this.artist = artist;
29    }
```

```
60    public String getArtist() {
61        return artist;
62    }
```

- 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

```
31    public void addTrack(Track track) {
32        for(int i = 0; i < tracks.size(); i++) {
33            if(tracks.get(i).equals(track)) {
34                System.out.println("This track is already existed in the list!");
35                return;
36            }
37        }
38        tracks.add(track);
39        System.out.println("This track is already added!");
40    }
```

```
42•    public void removeTrack(Track track) {
43         if(tracks.contains(track)) {
44             tracks.remove(track);
45             System.out.println("This track is already removed!");
46         }
47         else {
48             System.out.println("This track does not exist in the list!");
49         }
50     }
```

- 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.
- Save your changes

```
52•    public int getLength() {
53         int sum = 0;
54         for(int i = 0; i < tracks.size(); i++) {
55             sum += tracks.get(i).getLength();
56         }
57         return sum;
58     }
```

11. Implementation of 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();**
- Save your changes

```
1 package hust.soict.globalict.aims.media;
2
3 public interface Playable {
4
5     public void play();
6 }
```

- 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

For DigitalVideoDisc class:

```java
@Override
public void play() {
    // TODO Auto-generated method stub
    if(this.getLength() <= 0) {
        System.out.println("Cannot play this DVD!");
    }
    else {
        System.out.println("Playing DVD: " + this.getTitle());
        System.out.println("DVD length: " + this.getLength());
    }
}
```

For Track class:

```java
@Override
public void play() {
    // TODO Auto-generated method stub
    if(length <= 0) {
        System.out.println("Cannot play this track!");
    }
    else{
        System.out.println("Playing track: " + title);
        System.out.println("Track length: " + length);
    }
}
```

- Implement **play**() method for **CompactDisc**
  - Since the **CompactDisc** class contains an **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

```java
@Override
public void play() {
    // TODO Auto-generated method stub
    System.out.println(getTitle() + " - " +
            getDirector() + " - " +
            getCategory() + " - " +
            artist + " - " +
            super.getLength() + ": " +
            getCost());
    for(int i = 0; i < tracks.size(); i++) {
        tracks.get(i).play();
    }
}
```

12. 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:
- Create **addMedia**() and **removeMedia**() to replace **addDigitalVideoDisc**() and **removeDigitalVideoDisc**()

```
1  package hust.soict.globalict.aims.cart;
2
3  import java.util.ArrayList;
4  import hust.soict.globalict.aims.media.DigitalVideoDisc;
5  import hust.soict.globalict.aims.media.Media;
6
7  public class Cart {
8
9      private ArrayList<Media> itemsOrdered = new ArrayList<Media>();
10
11     public void addMedia(Media media) {
20
21     public void removeMedia(Media media) {
30
31     public float totalCost() {
38
39     public void display() {
46 }
```

addMedia() method:

```
11     public void addMedia(Media media) {
12         if(itemsOrdered.contains(media)) {
13             System.out.println("This media is already in the cart!");
14         }
15         else {
16             itemsOrdered.add(media);
17             System.out.println("This media has been added in the cart");
18         }
19     }
```

removeMedia() method:

```
21     public void removeMedia(Media media) {
22         if(itemsOrdered.contains(media)) {
23             itemsOrdered.remove(media);
24             System.out.println("This media has been removed from the cart");
25         }
26         else {
27             System.out.println("This media does not exist in the cart");
28         }
29     }
```

- Update the **totalCost**() method

```
31•    public float totalCost() {
32         float sum = 0;
33         for(int i = 0; i < itemsOrdered.size(); i++) {
34             sum+=itemsOrdered.get(i).getCost();
35         }
36         return sum;
37     }
```

13. 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**()

```
1 package hust.soict.globalict.aims.store;
2
3•import java.util.ArrayList;
4 import hust.soict.globalict.aims.media.Media;
5
6 public class Store {
7
8     private ArrayList<Media> itemsInStore = new ArrayList<Media>();
```

addMedia() method:

```
10•    public void addMedia(Media media) {
11         if(itemsInStore.contains(media)) {
12             System.out.println("This media is already in the store!");
13         }
14         else {
15             itemsInStore.add(media);
16             System.out.println("This media has been added in the store");
17         }
18     }
```

removeMedia() method:

```
20•    public void removeMedia(Media media) {
21         if(itemsInStore.contains(media)) {
22             itemsInStore.remove(media);
23             System.out.println("This media has been removed from the store");
24         }
25         else {
26             System.out.println("This media does not exist in the store");
27         }
28     }
```

14. Constructors of whole classes and parent classes

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

For Media class:

```
    public Media(int id, String title, String category, float cost) {
        this.title = title;
        this.category = category;
        this.cost = cost;
    }
```

For Book class:

```
    public Book(int id, String title, String category, float cost) {
        super(id, title, category, cost);
    }
```

For Disc class:

```
    public Disc(int id, String title, String category, float cost) {
        super(id, title, category, cost);
    }

    public Disc(int id, String title, String category, String director, float cost) {
        super(id, title, category, cost);
        this.director = director;
    }

    public Disc(int id, String title, String category, String director, int length, float cost) {
        super(id, title, category, cost);
        this.director = director;
        this.length = length;
    }
```

For DigitalVideoDisc class:

```
    public DigitalVideoDisc(int id, String title, String category, float cost) {
        super(id, title, category, cost);
    }

    public DigitalVideoDisc(int id, String title, String category, String director, float cost) {
        super(id, title, category, director, cost);
    }

    public DigitalVideoDisc(int id, String title, String category, String director, int length, float cost) {
        super(id, title, category, director, length, cost);
    }
```

For CompactDisc class:

```
    public CompactDisc(int id, String title, String category, String artist, float cost) {
        super(id, title, category, cost);
        this.artist = artist;
    }

    public CompactDisc(int id, String title, String category, String artist,  String director, float cost) {
        super(id, title, category, director, cost);
        this.artist = artist;
    }

    public CompactDisc(int id, String title, String category, String artist, String director, int length, float cost) {
        super(id, title, category, director, length, cost);
        this.artist = artist;
    }
```

For Track class:

```java
    public Track(String title, int length) {
        this.title = title;
        this.length = length;
    }
```

15. 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

```java
    @Override
    public boolean equals(Object obj) {
        if(obj == null) {
            return false;
        }
        Media media = (Media)obj;
        return (media.title == null) ? false : this.title.equals(media.getTitle());
    }

```

    - For the `Track` class: the `title` and the `length` are equal

```java
    @Override
    public boolean equals(Object obj) {
        if(obj == null) {
            return false;
        }
        Track track = (Track)obj;
        if(track.title == null) {
            return false;
        }
        return this.title.equals(track.title) && this.length == track.length;
    }
```

16. Polymorphism with toString() method

This exercise gives an illustration for polymorphism at behavior level.

Recall that for each type of media, we have implemented a `toString()` method that prints out the information of the object. When calling this method, depending on the type of object, corresponding `toString()` will be performed.

toString() method in Book class:

```java
    public String toString() {
        return "Book: [ID " + getId() + "] - " + getTitle() + " - " + getCategory() + " - " + getCost() +"$";
    }

}
```

toString() method in CompactDisc class:

```
    public String toString() {
        return "CompactDisc: (ID " + getId() + ") - " + getTitle() + " - " + getCategory() + artist + " - " +
        getDirector() + " - " + getLength() + " - " + getCost() +"$";
    }
```

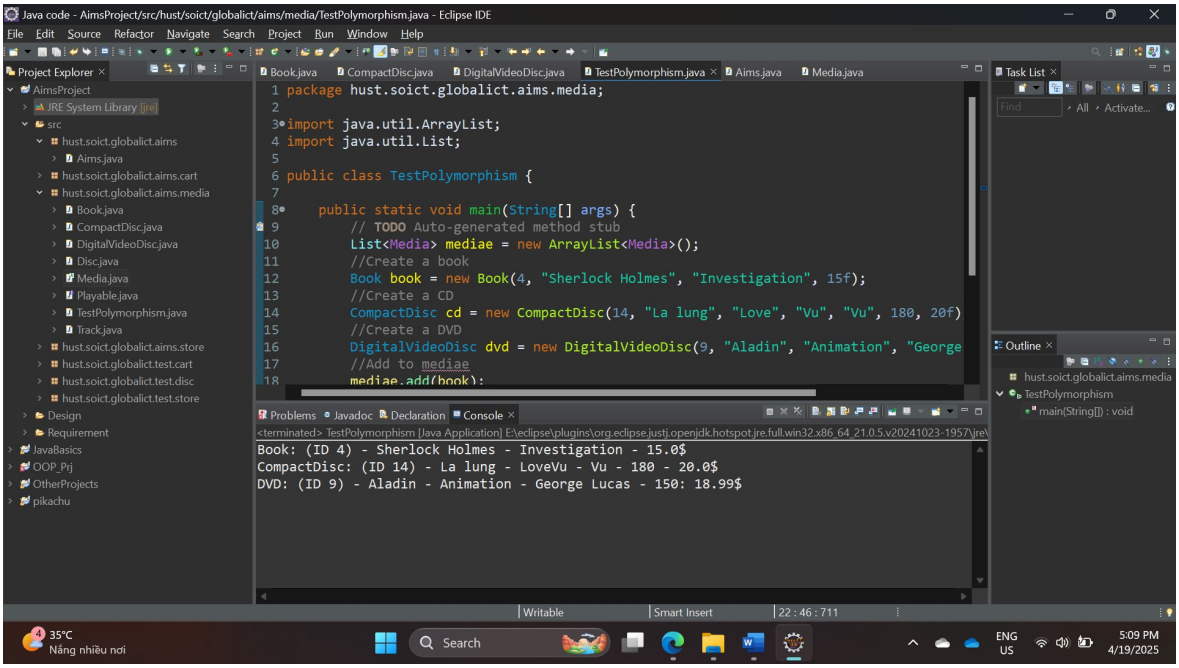
toString() method in DigitalVideoDisc class:

```
    public String toString() {
        return "DVD: (ID " + getId() + ") - " + getTitle() + " - " + getCategory() + " - "
                + getDirector() + " - " + getLength() + ": " + getCost() + "$";
    }
```

- 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.

```
package hust.soict.globalict.aims.media;

import java.util.ArrayList;
import java.util.List;

public class TestPolymorphism {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        List<Media> mediae = new ArrayList<Media>();
        //Create a book
        Book book = new Book(4, "Sherlock Holmes", "Investigation", 15f);
        //Create a CD
        CompactDisc cd = new CompactDisc(14, "La lung", "Love", "Vu", "Vu", 180, 20f);
        //Create a DVD
        DigitalVideoDisc dvd = new DigitalVideoDisc(9, "Aladin", "Animation", "George Lucas", 150, 18.99f);
        //Add to mediae
        mediae.add(book);
        mediae.add(cd);
        mediae.add(dvd);
        for(Media m:mediae) {
            System.out.println(m.toString());
        }
    }

}
```

Result:



17. 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:

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

```
public class MediaComparatorByCostTitle implements Comparator<Media>
public class MediaComparatorByTitleCost implements Comparator<Media>
```

- 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.

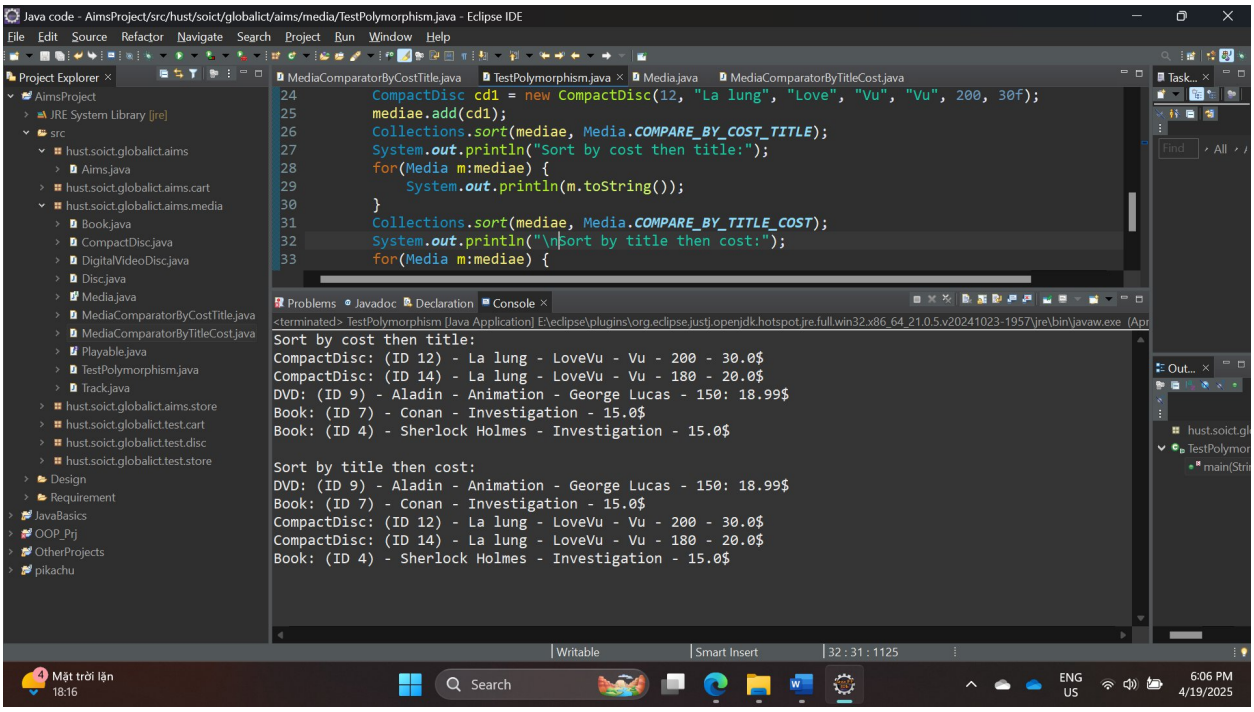
```java
package hust.soict.globalict.aims.media;

import java.util.Comparator;

public class MediaComparatorByTitleCost implements Comparator<Media> {

    @Override
    public int compare(Media o1, Media o2) {
        // TODO Auto-generated method stub
        if(o1.getTitle().equals(o2.getTitle())){
            return o1.getCost() > o2.getCost() ? -1 : 1;
        }
        return o1.getTitle().compareTo(o2.getTitle());
    }

}
```

```java
package hust.soict.globalict.aims.media;

import java.util.Comparator;

public class MediaComparatorByCostTitle implements Comparator<Media> {

    @Override
    public int compare(Media o1, Media o2) {
        // TODO Auto-generated method stub
        if(o1.getCost() < o2.getCost()) {
            return 1;
        }
        if(o1.getCost() > o2.getCost()) {
            return -1;
        }
        return o1.getTitle().compareTo(o2.getTitle());
    }

}
```

Test 2 types of sort:

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        List<Media> mediae = new ArrayList<Media>();
        //Create a book
        Book book = new Book(4, "Sherlock Holmes", "Investigation", 15f);
        //Create a CD
        CompactDisc cd = new CompactDisc(14, "La lung", "Love", "Vu", "Vu", 180, 20f);
        //Create a DVD
        DigitalVideoDisc dvd = new DigitalVideoDisc(9, "Aladin", "Animation", "George Lucas", 150, 18.99f);
        //Add to mediae
        mediae.add(book);
        mediae.add(cd);
        mediae.add(dvd);
        Book book1 = new Book(7, "Conan", "Investigation", 15f);
        mediae.add(book1);
        CompactDisc cd1 = new CompactDisc(12, "La lung", "Love", "Vu", "Vu", 200, 30f);
        mediae.add(cd1);
        Collections.sort(mediae, Media.COMPARE_BY_COST_TITLE);
        System.out.println("Sort by cost then title:");
        for(Media m:mediae) {
            System.out.println(m.toString());
        }
        Collections.sort(mediae, Media.COMPARE_BY_TITLE_COST);
        System.out.println("Sort by title then cost:");
        for(Media m:mediae) {
            System.out.println(m.toString());
        }
    }
```

Result:



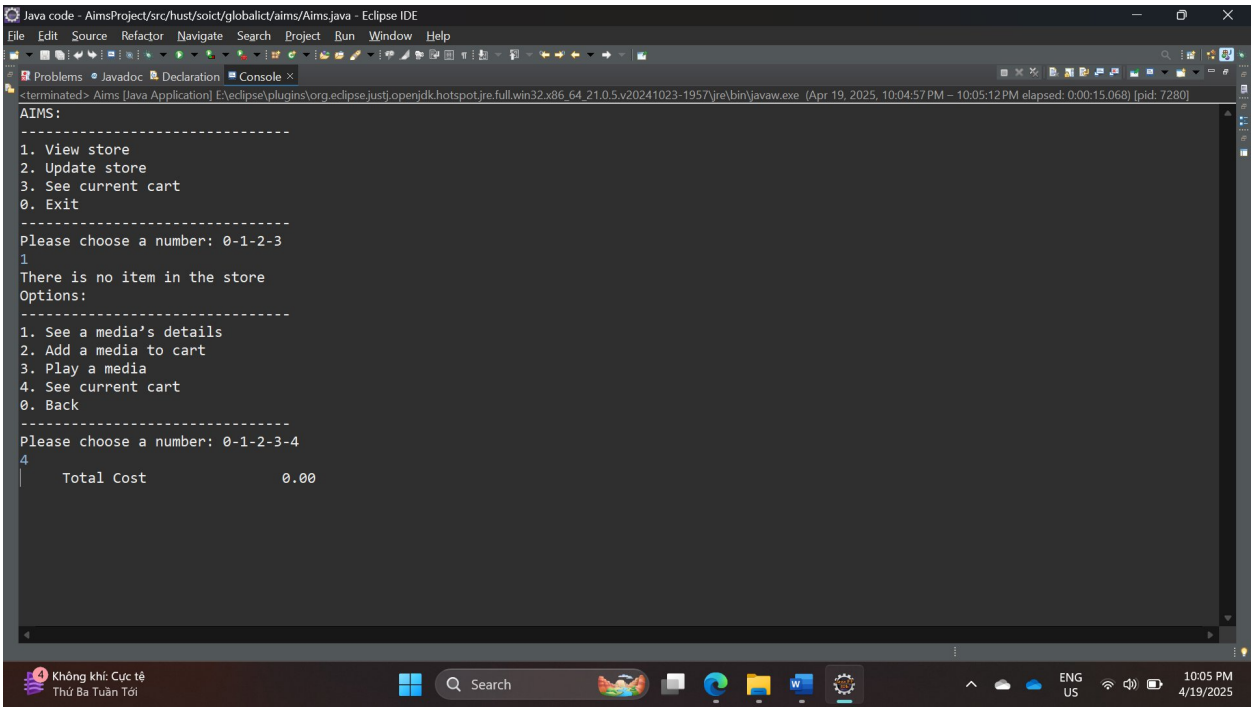
18. Create a complete console application in the Aims class

```java
package hust.soict.globalict.aims;

import java.util.Scanner;

public class Aims {

    private Store store = new Store();
    private Cart cart = new Cart();

    public static void showMenu() {
        System.out.println("AIMS: ");
        System.out.println("--------------------------------");
        System.out.println("1. View store");
        System.out.println("2. Update store");
        System.out.println("3. See current cart");
        System.out.println("0. Exit");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2-3");
    }

    public static void storeMenu() {
        System.out.println("Options: ");
        System.out.println("--------------------------------");
        System.out.println("1. See a media's details");
        System.out.println("2. Add a media to cart");
        System.out.println("3. Play a media");
        System.out.println("4. See current cart");
        System.out.println("0. Back");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2-3-4");
    }

    public static void mediaDetailsMenu() {
        System.out.println("Options: ");
        System.out.println("--------------------------------");
        System.out.println("1. Add to cart");
        System.out.println("2. Play");
        System.out.println("0. Back");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2");
    }

    public static void cartMenu() {
        System.out.println("Options: ");
        System.out.println("--------------------------------");
        System.out.println("1. Filter media in cart");
        System.out.println("2. Sort media in cart");
        System.out.println("3. Remove media from cart");
        System.out.println("4. Play a media");
        System.out.println("5. Place order");
        System.out.println("0. Back");
        System.out.println("--------------------------------");
        System.out.println("Please choose a number: 0-1-2-3-4-5");
    }

    public void chooseMainMenu() {
        showMenu();
        Scanner in = new Scanner(System.in);
        int opt = in.nextInt();
        if(opt == 1) {
            viewStore();
        }
        else if(opt == 2) {
            updateStore();
        }
        else if(opt == 3) {
            cart.display();
            cartMenu();
            seeCurrentCart();
        }
        else {
            System.exit(0);
        }
    }

    public void viewStore() {
        store.displayItems();
        storeMenu();
        Scanner in = new Scanner(System.in);
        int opt = in.nextInt();
        if(opt == 1) {
            System.out.print("Enter title of media: ");
            String title = in.nextLine();
            store.seeMediaDetail(title);
            mediaDetailsMenu();
            seeMediaDetails();
```

```
        }
        else if(opt == 2) {
            Media media = null;
            cart.addMedia(media);
        }
        else if(opt == 3){
            CompactDisc media = null;
            media.play();
        }
        else if(opt == 4) {
            cart.display();
        }
        else {
            chooseMainMenu();
        }
    }

    public void seeMediaDetails() {
        Scanner in = new Scanner(System.in);
        int opt = in.nextInt();
        DigitalVideoDisc media = null;
        if(opt == 1) {
            cart.addMedia(media);
        }
        else if(opt == 2) {
            media.play();
        }
        else {
            viewStore();
```

```
    public void updateStore() {
        Scanner in = new Scanner(System.in);
        System.out.println("Add or Remove?");
        String type = in.nextLine();
        if(type.equals("Add")) {
            Media media = null;
            store.addMedia(media);
        }
        else {
            Media media = null;
            store.addMedia(media);
        }
    }

    public void seeCurrentCart() {
        Scanner in = new Scanner(System.in);
        int opt = in.nextInt();
        if(opt == 1) {
            System.out.println("filter with id or title:");
            String type = in.nextLine();
            cart.filterCart(type);
        }
        else if(opt == 2) {
            System.out.println("sort by title or cost:");
            String type = in.nextLine();
            cart.sortMedia(type);
        }
        else if(opt == 3) {
            Media media = null;
```

```
            cart.removeMedia(media);
        }
        else if(opt == 4) {
            CompactDisc media = null;
            media.play();
        }
        else if(opt == 5) {
            System.out.println("Ordering successfully!");
            cart.clear();
        }
        else {
            chooseMainMenu();
        }
    }

    public static void main(String[] args) {
        Aims aims = new Aims();
        aims.chooseMainMenu();
    }
}
```



- Update the UML class diagram for the **`AimsProject`**.