

BÁO CÁO THỰC HÀNH LAB 4 LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

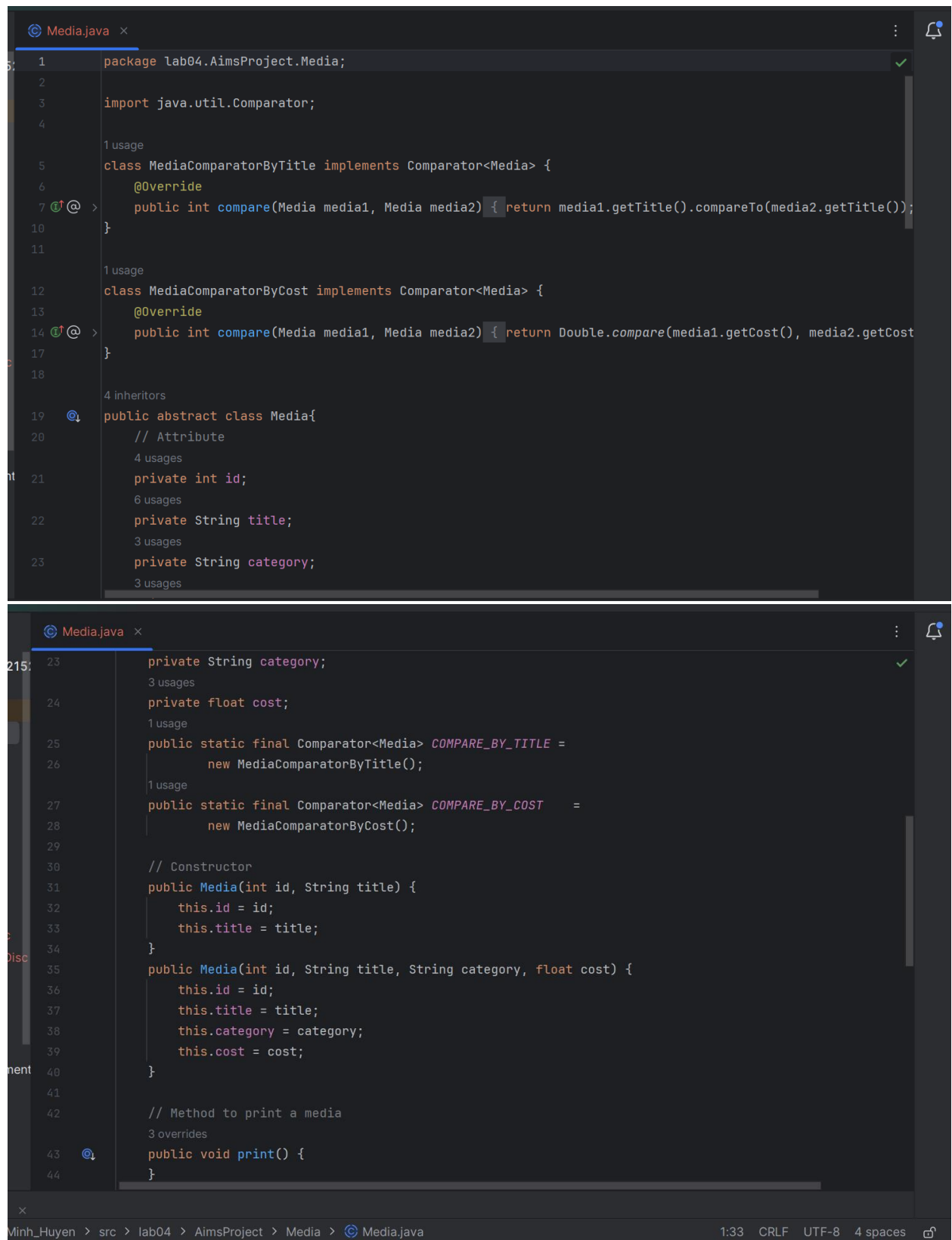
Table of Contents

1. Create the abstract Media class.....	2
2. Create the Disc class extending the Media class	3
3. Create the Playable interface.....	4
4. Create the Book class.....	5
5. Create the Track class	6
6. Create the CompactDisc class	7
7. Update the DigitalVideoDisc	9
8. Update the Cart class	10
9. Update the Store class	13
10. Update the Aims class	15
11. Demo Aims	19
12. Update the class diagram	21
13. Update the Usecase diagram	22
14. Answer the questions	22

Table of Figures

Figure 1 Media abstract class code.....	3
Figure 2 Disc class code.....	4
Figure 3 Playable interface code.....	5
Figure 4 Book class code	6
Figure 5 Track class code	7
Figure 6 CompactDisc class code	9
Figure 7 DVD class code	10
Figure 8 Cart class code	13
Figure 9 Store class code.....	15
Figure 10 Aims class code	19
Figure 11 Aims demo	21
Figure 12 Class diagram	22
Figure 13 Usecase diagram	22
Figure 14 Question code	23
Figure 15 Question code	23
Figure 16 Question code	24

1. Create the abstract Media class

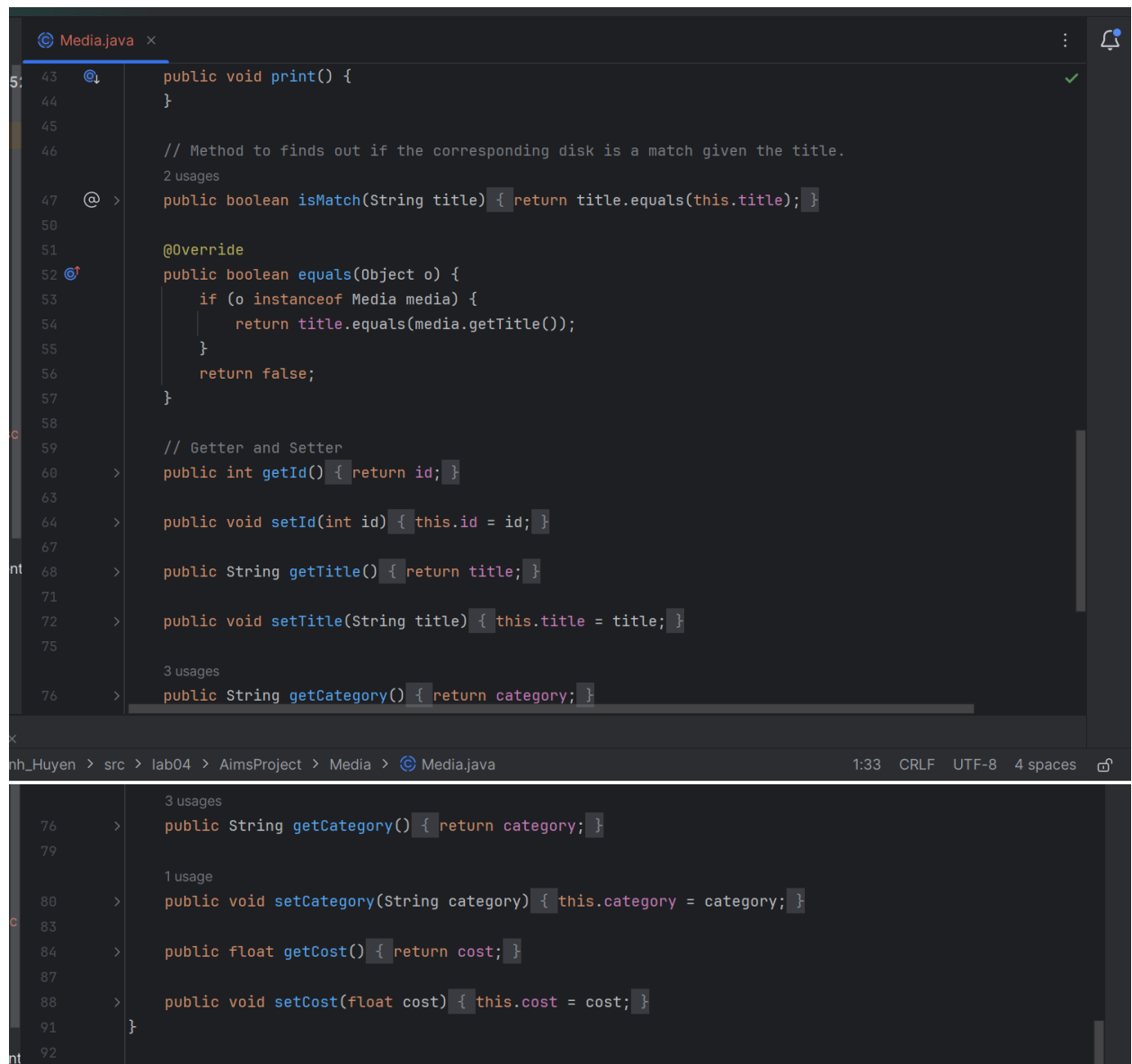


The image shows two screenshots of an IDE (IntelliJ IDEA) displaying the code for the `Media.java` file. The first screenshot shows the package declaration, imports, and two comparator classes. The second screenshot shows the abstract `Media` class with its attributes, static comparators, constructors, and a `print` method.

```

1 package lab04.AimsProject.Media;
2
3 import java.util.Comparator;
4
5 class MediaComparatorByTitle implements Comparator<Media> {
6     @Override
7     public int compare(Media media1, Media media2) { return media1.getTitle().compareTo(media2.getTitle()); }
8 }
9
10
11
12 class MediaComparatorByCost implements Comparator<Media> {
13     @Override
14     public int compare(Media media1, Media media2) { return Double.compare(media1.getCost(), media2.getCost()); }
15 }
16
17
18
19 public abstract class Media {
20     // Attribute
21     private int id;
22     private String title;
23     private String category;
24
25     // Static comparators
26     public static final Comparator<Media> COMPARE_BY_TITLE = new MediaComparatorByTitle();
27     public static final Comparator<Media> COMPARE_BY_COST = new MediaComparatorByCost();
28
29     // Constructors
30     public Media(int id, String title) {
31         this.id = id;
32         this.title = title;
33     }
34     public Media(int id, String title, String category, float cost) {
35         this.id = id;
36         this.title = title;
37         this.category = category;
38         this.cost = cost;
39     }
40
41     // Method to print a media
42     public void print() {
43     }
44 }
  
```

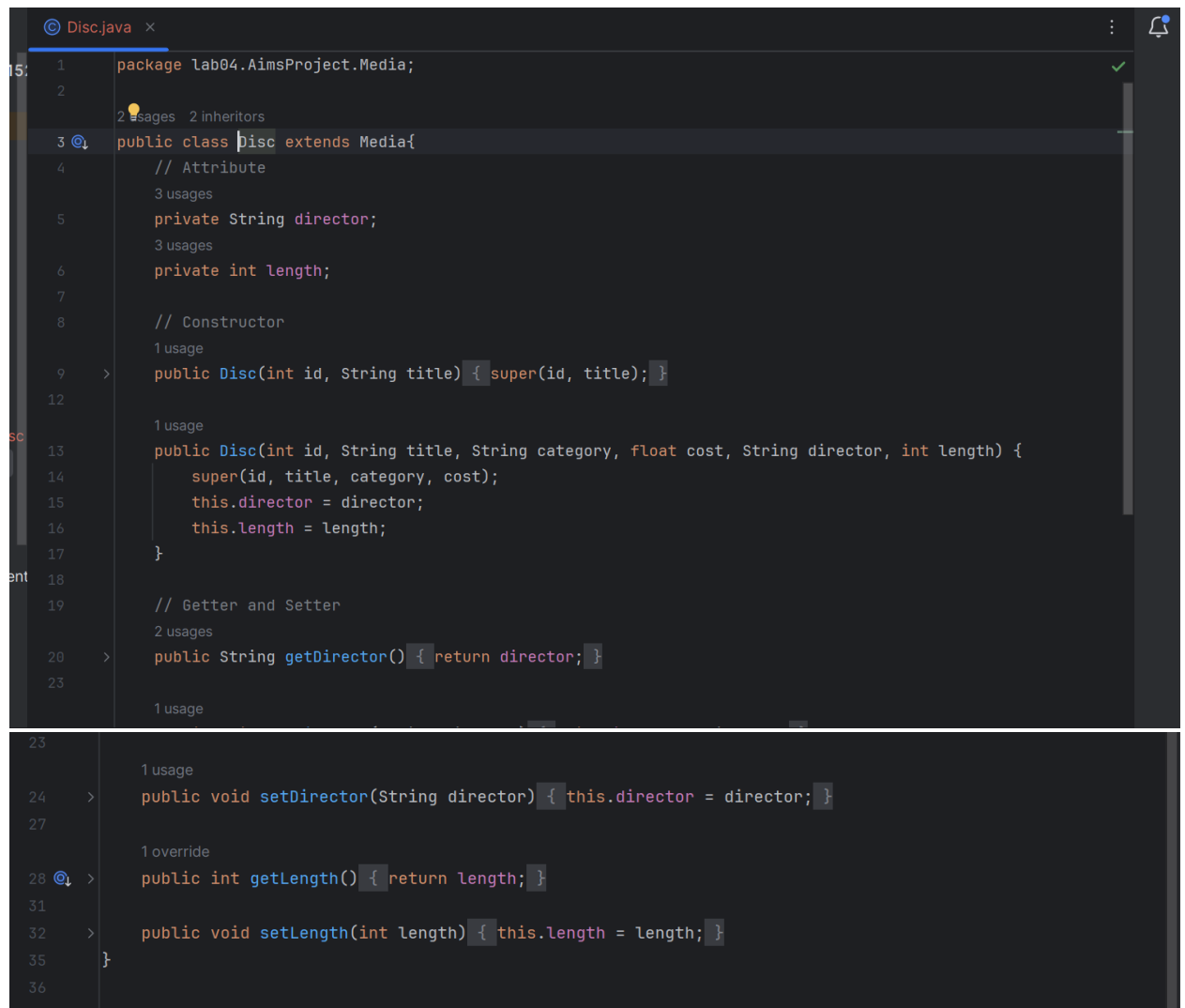
The IDE status bar at the bottom shows the file path: `Minh_Huyen > src > lab04 > AimsProject > Media > Media.java`, the time: `1:33`, the line ending: `CRLF`, the encoding: `UTF-8`, and the indentation: `4 spaces`.



```
Media.java x
43 public void print() {
44 }
45
46 // Method to finds out if the corresponding disk is a match given the title.
47 // 2 usages
48 @ > public boolean isMatch(String title) { return title.equals(this.title); }
49
50
51 @Override
52 @ public boolean equals(Object o) {
53     if (o instanceof Media media) {
54         return title.equals(media.getTitle());
55     }
56     return false;
57 }
58
59 // Getter and Setter
60 > public int getId() { return id; }
61
62
63 > public void setId(int id) { this.id = id; }
64
65
66 > public String getTitle() { return title; }
67
68 > public void setTitle(String title) { this.title = title; }
69
70
71 // 3 usages
72 > public String getCategory() { return category; }
73
74
75
76 > public String getCategory() { return category; }
77
78
79
80 // 1 usage
81 > public void setCategory(String category) { this.category = category; }
82
83
84 > public float getCost() { return cost; }
85
86
87 > public void setCost(float cost) { this.cost = cost; }
88
89
90 }
91
92
```

Figure 1 Media abstract class code

2. Create the Disc class extending the Media class



```
1 package lab04.AimsProject.Media;
2
3 public class Disc extends Media{
4     // Attribute
5     private String director;
6     private int length;
7
8     // Constructor
9     public Disc(int id, String title) { super(id, title); }
10
11     public Disc(int id, String title, String category, float cost, String director, int length) {
12         super(id, title, category, cost);
13         this.director = director;
14         this.length = length;
15     }
16
17     // Getter and Setter
18
19     public String getDirector() { return director; }
20
21     public void setDirector(String director) { this.director = director; }
22
23     public int getLength() { return length; }
24
25     public void setLength(int length) { this.length = length; }
26 }
```

Figure 2 Disc class code

3. Create the Playable interface

```

1 package lab04.AimsProject.Media;
2
3 3 usages 3 implementations
4 public interface Playable {
5     // Method to play
6     3 usages 3 implementations
7     public void play();
8 }

```

Figure 3 Playable interface code

4. Create the Book class

```

1 package lab04.AimsProject.Media;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 13 usages
7 public class Book extends Media{
8     // Attribute
9     7 usages
10    private List<String> authors = new ArrayList<>();
11
12    // Constructor
13    9 usages
14    public Book(int id, String title, String category, float cost) { super(id, title, category, cost); }
15
16    // Method to add an author
17    no usages
18    public void addAuthor(String authorName) {
19        int indexOfAuthor = authors.indexOf(authorName);
20        if (indexOfAuthor == -1) {
21            System.out.println("Author is already in the list");
22            return;
23        }
24        authors.add(authorName);
25        System.out.println("Added");
26    }
27 }

```

```

24     }
25
26     // Method to remove an author
27     no usages
28     public void removeAuthor(String authorName) {
29         int indexOfAuthor = authors.indexOf(authorName);
30         if (indexOfAuthor == -1) {
31             System.out.println("Author is absent in the list");
32             return;
33         }
34         authors.remove(indexOfAuthor);
35         System.out.println("Removed");
36     }
37
38     // Method to print a book
39     @Override
40     public void print() {
41         System.out.print(getId() + ". Book - "
42             + getTitle() + " - "
43             + getCategory() + " - ");
44         for (String author: authors) {
45             System.out.print(author + " - ");
46         }
47         System.out.println(getCost() + "$");
48     }
49
50     // Getter and Setter
51     no usages
52     public List<String> getAuthors() { return authors; }
53
54     no usages
55     public void setAuthors(List<String> authors) { this.authors = authors; }
56 }
57
58

```

Figure 4 Book class code

5. Create the Track class



```

1  package lab04.AimsProject.Media;
2
3  public class Track implements Playable{
4      // Attribute
5      private String title;
6      private int length;
7
8      // Constructor
9      public Track(String title, int length) {
10         this.title = title;
11         this.length = length;
12     }
13
14     // Method to play a track
15     public void play() {
16         System.out.println("Playing track: " + title);
17         System.out.println("Track length : " + length);
18     }
19
20     @Override
21     public boolean equals(Object o) {
22         if (o instanceof Track track) {
23             return title.equals(track.getTitle()) && length == track.getLength();
24         }
25         return false;
26     }
27
28     // Getter and Setter
29     public String getTitle() { return title; }
30
31     public void setTitle(String title) { this.title = title; }
32
33     public int getLength() { return length; }
34
35     public void setLength(int length) { this.length = length; }
36 }

```

Figure 5 Track class code

6. Create the CompactDisc class

```
1 package lab04.AimsProject.Media;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 10 usages
7 public class CompactDisc extends Disc implements Playable{
8     // Attribute
9     5 usages
10     private String artist;
11     6 usages
12     private List<Track> tracks = new ArrayList<>();
13
14     // Constructor
15     7 usages
16     public CompactDisc(int id, String title, String category, float cost,
17         String director, int length, String artist) {
18         super(id, title, category, cost, director, length);
19         this.artist = artist;
20     }
21
22     // Method to add a track
23     no usages
24     public void addTrack(Track track) {
25         int indexOfTrack = tracks.indexOf(track);
26         if (indexOfTrack == -1) {
27
28             if (indexOfTrack == -1) {
29                 System.out.println("Track is already in the list");
30                 return;
31             }
32             tracks.add(track);
33             System.out.println("Added");
34         }
35
36     // Method to remove a track
37     no usages
38     public void removeTrack(Track track) {
39         int indexOfTrack = tracks.indexOf(track);
40         if (indexOfTrack == -1) {
41             System.out.println("Track is absent in the list");
42             return;
43         }
44         tracks.remove(indexOfTrack);
45         System.out.println("Removed");
46     }
47
48     // Method to get the length of CD
49     @Override
50     public int getLength() {
51         int length = 0;
52         for (Track track: tracks) {
```



```

44         for (Track track: tracks) {
45             length += track.getLength();
46         }
47         setLength(length);
48         return length;
49     }
50
51     // Method to play a track
52     3 usages
53     public void play() {
54         System.out.println("Playing CD: " + this.getTitle());
55         System.out.println("CD artist: " + artist);
56         System.out.println("CD length: " + this.getLength());
57         for (Track track: tracks) {
58             track.play();
59         }
60
61     // Method to print a cd
62     @Override
63     public void print() {
64         System.out.println(getId() + ". CD - "
65             + getTitle() + " - "
66             + getCategory() + " - "
67             + getDirector() + " - "
68             + artist + " - "
69             + getLength() + ": "
70             + getCost() + "$");
71     }
72
73     // Getter and Setter
74     no usages
75     public String getArtist() { return artist; }
76
77     no usages
78     public void setArtist(String artist) { this.artist = artist; }
79
80 }

```

Figure 6 CompactDisc class code

7. Update the DigitalVideoDisc

```

1 package lab04.AimsProject.Media;
2
3 public class DigitalVideoDisc extends Disc implements Playable{
4     // Constructor
5     public DigitalVideoDisc(int id, String title) { super(id, title); }
6
7
8     public DigitalVideoDisc(int id, String title, String category, float cost) {
9         this(id, title);
10        this.setCategory(category);
11        this.setCost(cost);
12    }
13
14    public DigitalVideoDisc(int id, String title, String category, String director, float cost) {
15        this(id, title, category, cost);
16        this.setDirector(director);
17    }
18
19    public DigitalVideoDisc(int id, String title, String category, String director, int length, float cost) {
20        this(id, title, category, director, cost);
21        this.setLength(length);
22    }
23
24
25    // Method to print a dvd
26    @Override
27    public void print() {
28        System.out.println(getId() + ". DVD - "
29
30
31
32
33
34    }
35
36    // Method to play a dvd
37    3 usages
38    public void play() {
39        System.out.println("Playing DVD: " + this.getTitle());
40        System.out.println("DVD length: " + this.getLength());
41    }
42

```

Figure 7 DVD class code

8. Update the Cart class

```
1 package lab04.AimsProject;
2
3 import lab04.AimsProject.Media.DigitalVideoDisc;
4 import lab04.AimsProject.Media.Media;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class Cart {
9     // Attribute
10    private List<Media> itemsOrdered = new ArrayList<>();
11    private int numOfDVDs;
12
13    // Constructor
14    public Cart() { numOfDVDs = 0; }
15
16    // Method to add a new media
17    public void addMedia(Media media) {
18        if (itemsOrdered.contains(media)) {
19            System.out.println("Media is already in the list");
20            return;
21        }
22        itemsOrdered.add(media);
23        if (media.getClass() == DigitalVideoDisc.class) {
24            numOfDVDs++;
25        }
26        System.out.println("Added");
27    }
28
29    // Method to remove a media
30    public void removeMedia(Media media) {
31        // Search for media
32        int indexOfRemoved = itemsOrdered.indexOf(media);
33
34        // If not found
35        if (indexOfRemoved == -1) {
36            System.out.println("Not found");
37            return;
38        }
39
40        // Remove
41        itemsOrdered.remove(indexOfRemoved);
42        if (media.getClass() == DigitalVideoDisc.class) {
43            numOfDVDs--;
44        }
45    }
46 }
```

```

46     }
47
48     // Notify
49     System.out.println("Removed");
50 }
51
52 // Method to calculate the total cost
53 1 usage
54 public double totalCost() {
55     float cost = 0;
56     for (Media media: itemsOrdered) {
57         cost += media.getCost();
58     }
59
60     return Math.round(cost * 100.0) / 100.0;
61 }
62
63 // Method to print the list of ordered items of a cart,
64 // the price of each item, and the total price
65 4 usages
66 public void printCart() {
67     System.out.println("*****CART*****");
68
69     public void printCart() {
70         System.out.println("*****CART*****");
71         System.out.println("Ordered Items:");
72         for (Media media : itemsOrdered) {
73             media.print();
74         }
75         System.out.println("Total cost: " + totalCost());
76         System.out.println("*****");
77     }
78
79 // Method to search for media in the cart by ID and display the search results.
80 1 usage
81 public Media searchByID(int id) {
82     for (Media media: itemsOrdered) {
83         if (media.getId() == id) {
84             return media;
85         }
86     }
87     System.out.println("Not found!");
88     return null;
89 }
90
91 // Method to search for media in the cart by title.
92 3 usages
93 public Media searchByTitle(String title) {
94     for (Media media: itemsOrdered) {

```

```
85 // Method to search for media in the cart by title.
86 // 3 usages
87 public Media searchByTitle(String title) {
88     for (Media media: itemsOrdered) {
89         if (media.isMatch(title)) {
90             return media;
91         }
92     }
93     System.out.println("Not found!");
94     return null;
95 }
96
97 // Method to sort by title and print
98 // 1 usage
99 public void sortByTitle() {
100     itemsOrdered.sort(Media.COMPARE_BY_TITLE);
101     printCart();
102 }
103
104 // Method to sort by cost and print
105 // 1 usage
106 public void sortByCost() {
107     itemsOrdered.sort(Media.COMPARE_BY_COST);
108     printCart();
109 }
110
111 // Getter and Setter
112
113 // 1 usage
114 > public int getNumOfDVDs() { return numOfDVDs; }
115
116 // no usages
117 > public void setNumOfDVDs(int numOfDVDs) { this.numOfDVDs = numOfDVDs; }
118 }
```

Figure 8 Cart class code

9. Update the Store class

```
1 package lab04.AimsProject;
2
3 import lab04.AimsProject.Media.Media;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 4 usages
9 public class Store {
10     // Attribute
11     8 usages
12     private List<Media> itemsInStore = new ArrayList<>();
13
14     // Constructor
15     2 usages
16     public Store() {
17     }
18
19     // Method to add a media
20     20 usages
21     public void addMedia(Media media) {
22         if (itemsInStore.contains(media)) {
23             System.out.println("Media is already in the list");
24             return;
25         }
26         itemsInStore.add(media);
27         itemsInStore.add(media);
28         System.out.println("Added");
29     }
30
31     // Method to remove a media
32     1 usage
33     public void removeMedia(Media media) {
34         // Search for disc
35         int indexOfRemoved = itemsInStore.indexOf(media);
36
37         // If not found
38         if (indexOfRemoved == -1) {
39             System.out.println("Not found");
40             return;
41         }
42
43         // Remove
44         itemsInStore.remove(indexOfRemoved);
45
46         // Notify
47         System.out.println("Removed");
48     }
49
50     // Method to print all item in store
51     1 usage
```

```

44 // Method to print all item in store
45 1 usage
46 public void printStore() {
47     System.out.println("*****STORE*****");
48     System.out.println("Items in store:");
49     for (Media media : itemsInStore) {
50         media.print();
51     }
52     System.out.println("*****");
53 }
54
55 // Method to search for media in the store by title.
56 4 usages
57 public Media searchByTitle(String title) {
58     for (Media media: itemsInStore) {
59         if (media.isMatch(title)) {
60             return media;
61         }
62     }
63     System.out.println("Not found!");
64     return null;
65 }
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

h_Huyen > src > lab04 > AimsProject > © Store 3:38 CRLF UTF-8 4 spaces

```

64
65 // Getter and Setter
66 no usages
67 public List<Media> getItemsInStore() { return itemsInStore; }
68
69
70 no usages
71 public void setItemsInStore(List<Media> itemsInStore) { this.itemsInStore = itemsInStore; }
72
73 }
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Figure 9 Store class code

10. Update the Aims class

```
1 package lab04.AimsProject;
2
3 import lab04.AimsProject.Cart;
4 import lab04.AimsProject.Media.*;
5
6 import java.util.Scanner;
7
8 public class Aims {
9     // Create a new store
10    16 usages
11    static Store store = new Store();
12    // Create a new cart
13    13 usages
14    static Cart cart = new Cart();
15    1 usage
16    public static void showMenu() {
17        int command;
18        do {
19            Scanner scanner = new Scanner(System.in);
20            // Show store
21            System.out.println("AIMS: ");
22            System.out.println("-----");
23            System.out.println("1. View store");
24            System.out.println("2. Update store");
25            System.out.println("3. See current cart");
26            System.out.println("0. Exit");
27
28            System.out.println("0. Exit");
29            System.out.println("-----");
30            System.out.println("Please choose a number: 0-1-2-3");
31            command = scanner.nextInt();
32
33            // If chose View store
34            if (command == 1) {
35                store.printStore();
36                storeMenu();
37            }
38
39            // If chose 2
40            if (command == 2) {
41                updateStoreMenu();
42            }
43
44            // If chose 3
45            if (command == 3) {
46                cart.printCart();
47                cartMenu();
48            }
49        } while (command != 0);
50    }
51}
```



```

1 usage
47 public static void updateStoreMenu() {
48     Scanner scanner = new Scanner(System.in);
49     // Show menu
50     System.out.println("Options: ");
51     System.out.println("-----");
52     System.out.println("1. Add a media to store");
53     System.out.println("2. Remove a media from cart");
54     System.out.println("0. Back");
55     System.out.println("-----");
56     System.out.println("Please choose a number: 0-1-2");
57     int command = scanner.nextInt();
58
59     // If chose 1
60     if (command == 1) {
61         System.out.println("Added to store");
62     }
63
64     // If chose 2
65     if (command == 2) {
66         removeMediaFromStore();
67     }
68 }
69
1 usage

```

```

1 usage
70 public static void storeMenu() {
71     Scanner scanner = new Scanner(System.in);
72     // Show menu
73     System.out.println("Options: ");
74     System.out.println("-----");
75     System.out.println("1. See a media's details");
76     System.out.println("2. Add a media to cart");
77     System.out.println("3. Play a media");
78     System.out.println("4. See current cart");
79     System.out.println("0. Back");
80     System.out.println("-----");
81     System.out.println("Please choose a number: 0-1-2-3-4");
82     int command = scanner.nextInt();
83
84     // If chose 1
85     if (command == 1) {
86         Media media;
87         do {
88             System.out.println("Enter the title of the media: ");
89             scanner.nextLine();
90             String title = scanner.nextLine();
91             media = store.searchByTitle(title);
92         } while (media == null);
93         media.print();

```

```

93         media.print();
94         mediaDetailsMenu(media);
95     }
96
97     // If chose 2
98     if (command == 2) {
99         addMediaToCart();
100     }
101
102     // If chose 3
103     if (command == 3) {
104         Media media;
105         do {
106             System.out.println("Enter the title of the media: ");
107             scanner.nextLine();
108             String title = scanner.nextLine();
109             media = store.searchByTitle(title);
110         } while (media == null);
111         playAMedia(media);
112     }
113
114     // If chose 4
115     if (command == 4) {
116         cart.printCart();
117         cartMenu();
118     }
119 }
120
121 1 usage
122 public static void mediaDetailsMenu(Media media) {
123     Scanner scanner = new Scanner(System.in);
124     // Show menu
125     System.out.println("Options: ");
126     System.out.println("-----");
127     System.out.println("1. Add to cart");
128     if (!(media instanceof Book)) {
129         System.out.println("2. Play");
130     }
131     System.out.println("0. Back");
132     System.out.println("-----");
133     System.out.print("Please choose a number: 0-1");
134     if (!(media instanceof Book)) {
135         System.out.println("-2");
136     }
137     int command = scanner.nextInt();
138
139     // If chose 1
140     if (command == 1) {
141         cart.addMedia(media);

```

```
139         if (command == 1) {
140             cart.addMedia(media);
141         }
142
143         // If chose 2
144         if (command == 2) {
145             playAMedia(media);
146         }
147     }
148
149     2 usages
150     public static void cartMenu() {
151         Scanner scanner = new Scanner(System.in);
152         // Show menu
153         System.out.println("Options: ");
154         System.out.println("-----");
155         System.out.println("1. Filter medias in cart");
156         System.out.println("2. Sort medias in cart");
157         System.out.println("3. Remove media from cart");
158         System.out.println("4. Play a media");
159         System.out.println("5. Place order");
160         System.out.println("0. Back");
161         System.out.println("-----");
```

Figure 10 Aims class code

11. Demo Aims

```
AIMS:
-----
1. View store
2. Update store
3. See current cart
0. Exit
-----
Please choose a number: 0-1-2-3
1
*****STORE*****
Items in store:
1. DVD - Inception - Science Fiction - Christopher Nolan - 148: 19.99$
2. DVD - The Dark Knight - Action - Christopher Nolan - 152: 17.99$
7. DVD - Interstellar - Science Fiction - Christopher Nolan - 169: 21.99$
3. CD - Random Access Memories - Electronic - Daft Punk - Daft Punk - 0: 15.99$
4. CD - 25 - Pop - Adele - Adele - 0: 14.99$
8. CD - Lover - Pop - Taylor Swift - Taylor Swift - 0: 17.99$
5. Book - The Silent Patient - Thriller - 14.95$
6. Book - Where the Crawdads Sing - Mystery - 12.99$
9. Book - Educated - Memoir - 16.95$
10. Book - Becoming - Autobiography - 22.99$
*****
```

```
Options:
-----
1. See a media's details
2. Add a media to cart
3. Play a media
4. See current cart
0. Back
-----
Please choose a number: 0-1-2-3-4
1
Enter the title of the media:
Interstellar
7. DVD - Interstellar - Science Fiction - Christopher Nolan - 169: 21.99$
Options:
-----
1. Add to cart
2. Play
0. Back
-----
Please choose a number: 0-1-2
1
Added

AIMS:
-----
1. View store
2. Update store
3. See current cart
0. Exit
-----
Please choose a number: 0-1-2-3
0

Process finished with exit code 0
```

Figure 11 Aims demo

12. Update the class diagram

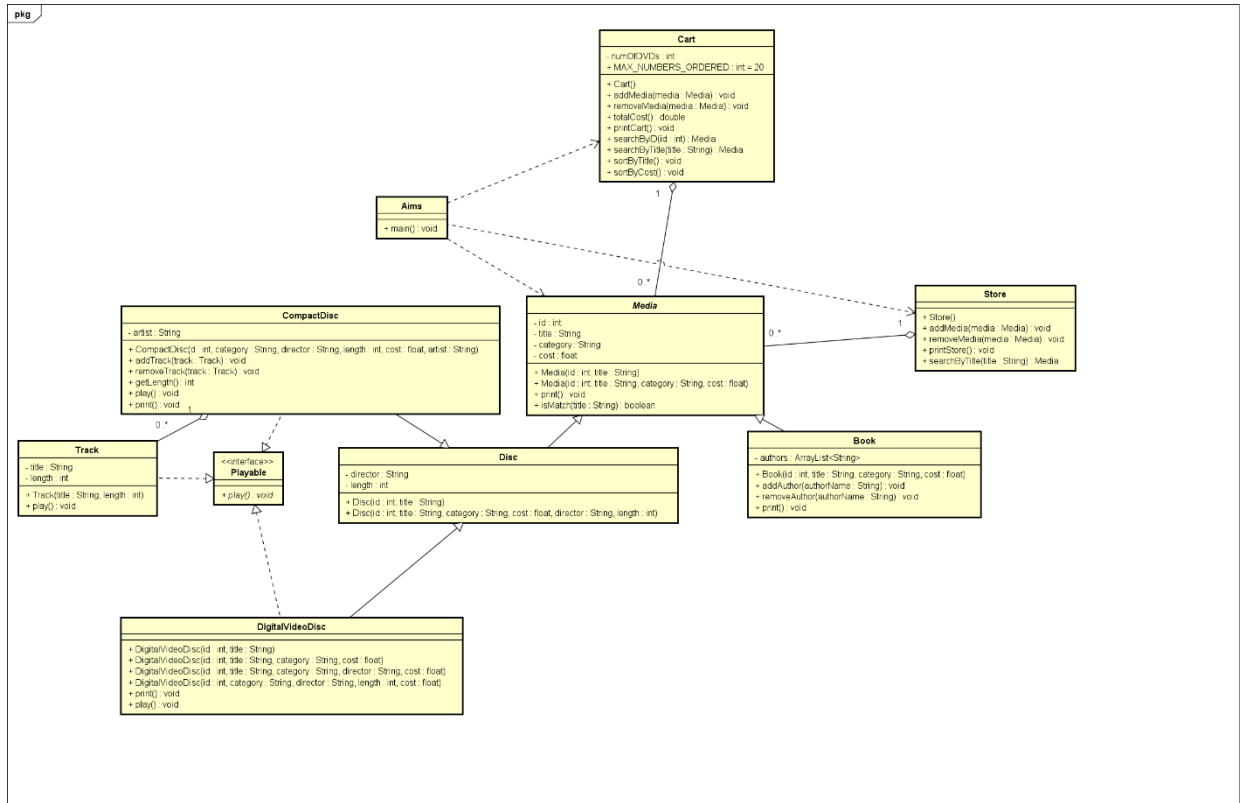


Figure 12 Class diagram

13. Update the Usecase diagram

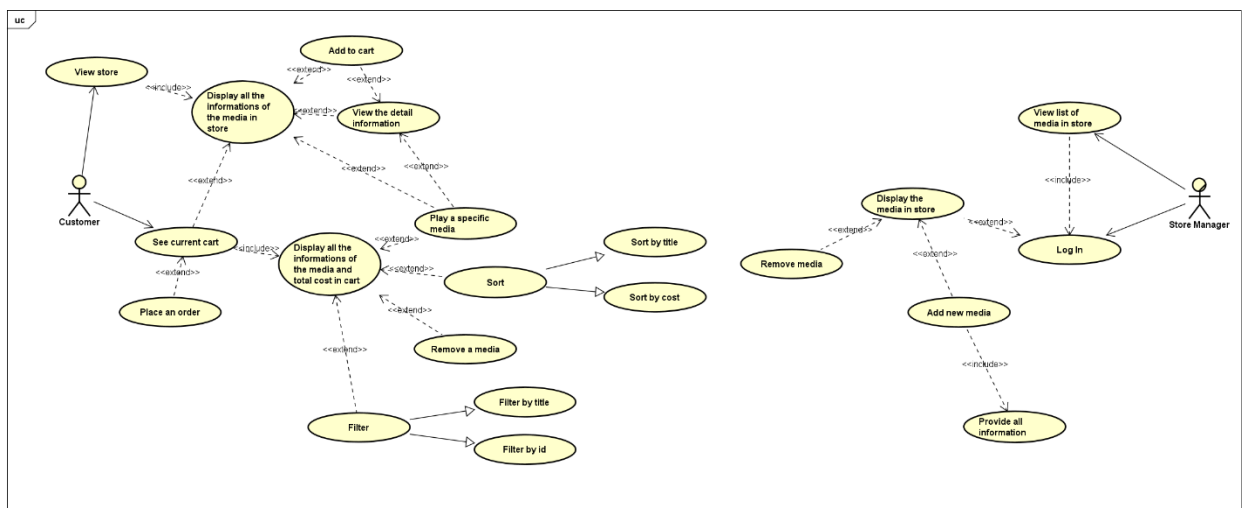


Figure 13 Usecase diagram

14. Answer the questions

- Trường hợp cần so sánh các Media với nhau bằng cách implement Comparable thay vì Comparator: Thay vì tạo class riêng cho từng Comparator, ta cần cho class Media implement interface Comparable.

- Ví dụ:

```
public abstract class Media implements Comparable<Media> {  
    new *  
    @Override  
    public int compareTo(Media otherMedia) {  
        // Compare by title  
        return this.title.compareTo(otherMedia.getTitle());  
    }  
}
```

Figure 14 Question code

- Có thể cài đặt như sau:

```
public abstract class Media implements Comparable<Media> {  
    new *  
    @Override  
    public int compareTo(Media otherMedia) {  
        // Compare by title first  
        int titleComparison = this.title.compareTo(otherMedia.getTitle());  
  
        // If titles are equal, compare by cost  
        return (titleComparison == 0) ? Float.compare(this.cost, otherMedia.getCost()) : titleComparison;  
    }  
}
```

Figure 15 Question code

- Cài đặt như sau:

```
public class DVD extends Media {  
    // Override compareTo for DVDs  
    new *  
    @Override  
    public int compareTo(Media otherMedia) {  
        if (otherMedia instanceof DVD) {  
            DVD otherDVD = (DVD) otherMedia;  
            // Compare by title first  
            int titleComparison = getTitle().compareTo(otherDVD.getTitle());  
            // If titles are equal, compare by decreasing length  
            if (titleComparison == 0) {  
                int lengthComparison = Integer.compare(otherDVD.getLength(), getLength());  
  
                // If lengths are equal, compare by cost  
                return (lengthComparison == 0) ? Float.compare(getCost(), otherDVD.getCost())  
                    : lengthComparison;  
            }  
            return titleComparison;  
        } else {  
            // For non-DVD media, use the default comparison (title then cost)  
            return super.compareTo(otherMedia);  
        }  
    }  
}
```

Figure 16 Question code