

BÁO CÁO THỰC HÀNH LAP 4 LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mục lục nội dung

1. Creating the abstract Media class.....	3
2. Create the Disc class extending the Media class	7
3. Creating the Playable interface.....	8
4. Creating the Book class	8
5. Creating the Track class	10
6. Creating the CompactDisc class	12
7. Update the DigitalVideoDisc class	15
8. Update the Cart class.....	17
9. Update the Store class	22
10. Update the Aims class.....	25
11. Demo Aims.....	38
12. Update the class diagram	40
13. Update the usecase diagram	40
14. Answer the questions	41

Mục lục hình ảnh

Figure 1 Media abstract class code.....	3
Figure 2 Media abstract class code.....	4
Figure 3 Media abstract class code.....	5
Figure 4 Media abstract class code.....	6
Figure 5 Disc class code.....	7
Figure 6 Disc class code.....	7
Figure 7 Playable interface code	8
Figure 8 Book class code	8
Figure 9 Book class code	9
Figure 10 Book class code	9
Figure 11 Track class code	10
Figure 12 Track class code	11
Figure 13 CompactDisc class code	12
Figure 14 CompactDisc class code	13
Figure 15 CompactDisc class code	14
Figure 16 CompactDisc class code	15
Figure 17 DVD class code	15
Figure 18 DVD class code	16

Figure 19 Cart class code	17
Figure 20 Cart class code	18
Figure 21 Cart class code	19
Figure 22 Cart class code	20
Figure 23 Cart class code	21
Figure 24 Store class code.....	22
Figure 25 Store class code.....	23
Figure 26 Store class code.....	24
Figure 27 Aims class code	25
Figure 28 Aims class code	26
Figure 29 Aims class code	27
Figure 30 Aims class code	28
Figure 31 Aims class code	29
Figure 32 Aims class code	30
Figure 33 Aims class code	31
Figure 34 Aims class code	32
Figure 35 Aims class code	33
Figure 36 Aims class code	34
Figure 37 Aims class code	35
Figure 38 Aims class code	36
Figure 39 Aims class code	37
Figure 40 Aims class code	37
Figure 41 Aims class code	38
Figure 42 Aims demo	38
Figure 43 Aims demo	39
Figure 44 Aims demo	39
Figure 45 Class diagram	40
Figure 46 Usecase diagram	40
Figure 47 Question code	41
Figure 48 Question code	41
Figure 49 Question code	42

1. Creating the abstract Media class

```
1 package lab04.AimsProject.Media;
2
3 import java.util.Comparator;
4
5 1 usage  🧑 manhnguyen41
6 class MediaComparatorByTitle implements Comparator<Media> {
7     🧑 manhnguyen41
8     @Override
9     public int compare(Media media1, Media media2) {
10         return media1.getTitle().compareTo(media2.getTitle());
11     }
12 }
13
14 1 usage  🧑 manhnguyen41
15 class MediaComparatorByCost implements Comparator<Media> {
16     🧑 manhnguyen41
17     @Override
18     public int compare(Media media1, Media media2) {
19         return Double.compare(media1.getCost(), media2.getCost());
20     }
21 }
22
23 4 inheritors  🧑 manhnguyen41
24 public abstract class Media{
25     // Attribute
```

Figure 1 Media abstract class code

```
21     private int id;  
    6 usages  
22     private String title;  
    3 usages  
23     private String category;  
    3 usages  
24     private float cost;  
    1 usage  
25     public static final Comparator<Media> COMPARE_BY_TITLE =  
26         new MediaComparatorByTitle();  
    1 usage  
27     public static final Comparator<Media> COMPARE_BY_COST =  
28         new MediaComparatorByCost();  
29  
30     // Constructor  
    manhnguyen41  
31     public Media(int id, String title) {  
32         this.id = id;  
33         this.title = title;  
34     }  
    manhnguyen41  
35     public Media(int id, String title, String category, float cost) {  
36         this.id = id;  
37         this.title = title;  
38         this.category = category;
```

Figure 2 Media abstract class code

```
39         this.cost = cost;
40     }
41
42     // Method to print a media
43     3 overrides  manhnguyen41
44     public void print() {
45     }
46
47     // Method to finds out if the corresponding disk is a match given the title.
48     2 usages  manhnguyen41
49     public boolean isMatch(String title) { return title.equals(this.title); }
50
51     manhnguyen41
52     @Override
53     public boolean equals(Object o) {
54         if (o instanceof Media media) {
55             return title.equals(media.getTitle());
56         }
57         return false;
58     }
59
60     // Getter and Setter
61     manhnguyen41
62     public int getId() { return id; }
63
```

Figure 3 Media abstract class code

```
67 > public void setId(int id) { this.id = id; }  
68 >   
71 > public String getTitle() { return title; }  
72 >   
75 > public void setTitle(String title) { this.title = title; }  
76 >   
79 > public String getCategory() { return category; }  
80 >   
83 > public void setCategory(String category) { this.category = category; }  
84 >   
87 > public float getCost() { return cost; }  
88 >   
91 > public void setCost(float cost) { this.cost = cost; }  
92 > }
```

Figure 4 Media abstract class code

2. Create the Disc class extending the Media class

```

1  package lab04.AimsProject.Media;
2
3  2 usages  2 inheritors  manhnguyen41
4  public class Disc extends Media{
5      // Attribute
6      3 usages
7      private String director;
8      3 usages
9      private int length;
10
11     // Constructor
12     1 usage  manhnguyen41
13     public Disc(int id, String title) { super(id, title); }
14
15     1 usage  manhnguyen41
16     public Disc(int id, String title, String category, float cost, String director, int length) {
17         super(id, title, category, cost);
18         this.director = director;
19         this.length = length;
20     }
21
22     // Getter and Setter
23     2 usages  manhnguyen41
24     public String getDirector() { return director; }
25
26
27

```

Figure 5 Disc class code

```

24  > public void setDirector(String director) { this.director = director; }
25
26
27
28  1 override  manhnguyen41
29  public int getLength() { return length; }
30
31
32  manhnguyen41
33  public void setLength(int length) { this.length = length; }
34
35  }

```

Figure 6 Disc class code

3. Creating the Playable interface

```

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

```

Figure 7 Playable interface code

4. Creating the Book class

```

1 package lab04.AimsProject.Media;
2
3 > import ...
4
5 13 usages 1 manhnguyen41
6 public class Book extends Media{
7     // Attribute
8     7 usages
9     private List<String> authors = new ArrayList<>();
10
11 // Constructor
12 9 usages 1 manhnguyen41
13 > public Book(int id, String title, String category, float cost) { super(id, title, category, cost); }
14
15 // Method to add an author
16 no usages 1 manhnguyen41
17 public void addAuthor(String authorName) {
18     int indexOfAuthor = authors.indexOf(authorName);
19     if (indexOfAuthor == -1) {
20         System.out.println("Author is already in the list");
21         return;
22     }
23     authors.add(authorName);
24     System.out.println("Added");
25 }

```

Figure 8 Book class code


```

26      // Method to remove an author
      no usages  🧑 manhnghuyen41
27      public void removeAuthor(String authorName) {
28          int indexOfAuthor = authors.indexOf(authorName);
29          if (indexOfAuthor == -1) {
30              System.out.println("Author is absent in the list");
31              return;
32          }
33          authors.remove(indexOfAuthor);
34          System.out.println("Removed");
35      }
36
37      // Method to print a book
      🧑 manhnghuyen41
38      @Override
39      public void print() {
40          System.out.print(getId() + ". Book - "
41              + getTitle() + " - "
42              + getCategory() + " - ");
43          for (String author: authors) {
44              System.out.print(author + " - ");
45          }
46          System.out.println(getCost() + "$");
47      }
48

```

Figure 9 Book class code

```

49      // Getter and Setter
      no usages  🧑 manhnghuyen41
50      > public List<String> getAuthors() { return authors; }
53
      no usages  🧑 manhnghuyen41
54      > public void setAuthors(List<String> authors) { this.authors = authors; }
57  }

```

Figure 10 Book class code

5. Creating 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     }
```

Figure 11 Track class code

```
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      manhnguyen41
30      > public String getTitle() { return title; }
31
32      manhnguyen41
33      > public void setTitle(String title) { this.title = title; }
34
35      manhnguyen41
36      > public int getLength() { return length; }
37
38      manhnguyen41
39      > public void setLength(int length) { this.length = length; }
40
41      }
42
43
44
45
```

Figure 12 Track class code

6. Creating the CompactDisc class

```
1 package lab04.AimsProject.Media;
2
3 > import ...
4
5 10 usages  ⓘ manhnguyen41
6 public class CompactDisc extends Disc implements Playable{
7     // Attribute
8     5 usages
9     private String artist;
10    6 usages
11    private List<Track> tracks = new ArrayList<>();
12
13    // Constructor
14    7 usages  ⓘ manhnguyen41
15    public CompactDisc(int id, String title, String category, float cost,
16                        String director, int length, String artist) {
17        super(id, title, category, cost, director, length);
18        this.artist = artist;
19    }
20
21    // Method to add a track
22    no usages  ⓘ manhnguyen41
23    public void addTrack(Track track) {
24        int indexOfTrack = tracks.indexOf(track);
25        if (indexOfTrack == -1) {
```

Figure 13 CompactDisc class code

```
22         System.out.println("Track is already in the list");
23         return;
24     }
25     tracks.add(track);
26     System.out.println("Added");
27 }
28
29 // Method to remove a track
no usages  🧑 manhnghuyen41
30 public void removeTrack(Track track) {
31     int indexOfTrack = tracks.indexOf(track);
32     if (indexOfTrack == -1) {
33         System.out.println("Track is absent in the list");
34         return;
35     }
36     tracks.remove(indexOfTrack);
37     System.out.println("Removed");
38 }
39
40 // Method to get the length of CD
🧑 manhnghuyen41
41 @Override
42 public int getLength() {
43     int length = 0;
44     for (Track track: tracks) {
```

Figure 14 CompactDisc class code

```

45         length += track.getLength();
46     }
47     setLength(length);
48     return length;
49 }
50
51 // Method to play a track
52 3 usages  🧑 manhnghuyen41
52 ④↑ public void play() {
53     System.out.println("Playing CD: " + this.getTitle());
54     System.out.println("CD artist: " + artist);
55     System.out.println("CD length: " + this.getLength());
56     for (Track track: tracks) {
57         track.play();
58     }
59 }
60
61 // Method to print a cd
62 ④ manhnghuyen41
62 @Override
63 ④↑ public void print() {
64     System.out.println(getId() + ". CD - "
65         + getTitle() + " - "
66         + getCategory() + " - "
67         + getDirector() + " - "

```

Figure 15 CompactDisc class code

```

68         + artist + " - "
69         + getLength() + ": "
70         + getCost() + "$");
71     }
72
73     // Getter and Setter
74     no usages  manhnguyen41
75     > public String getArtist() { return artist; }
76
77     no usages  manhnguyen41
78     > public void setArtist(String artist) { this.artist = artist; }
79
80 }
81
82

```

Figure 16 CompactDisc class code

7. Update the DigitalVideoDisc class

```

1  package lab04.AimsProject.Media;
2
3  manhnguyen41
4  public class DigitalVideoDisc extends Disc implements Playable{
5      // Constructor
6      manhnguyen41
7      > public DigitalVideoDisc(int id, String title) { super(id, title); }
8
9      manhnguyen41
10     public DigitalVideoDisc(int id, String title, String category, float cost) {
11         this(id, title);
12         this.setCategory(category);
13         this.setCost(cost);
14     }
15
16     manhnguyen41
17     public DigitalVideoDisc(int id, String title, String category, String director, float cost) {
18         this(id, title, category, cost);
19         this.setDirector(director);
20     }
21
22     manhnguyen41
23     public DigitalVideoDisc(int id, String title, String category, String director, int length, float cost) {
24         this(id, title, category, director, cost);
25         this.setLength(length);
26     }
27
28 }

```

Figure 17 DVD class code

```
23     }
24
25     // Method to print a dvd
26     @Override
27     public void print() {
28         System.out.println(getId() + ". DVD - "
29             + getTitle() + " - "
30             + getCategory() + " - "
31             + getDirector() + " - "
32             + getLength() + ": "
33             + getCost() + "$");
34     }
35
36     // Method to play a dvd
37     public void play() {
38         System.out.println("Playing DVD: " + this.getTitle());
39         System.out.println("DVD length: " + this.getLength());
40     }
41 }
42
```

Figure 18 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  4 usages  🧑 manhnghuyen41
9  public class Cart {
10     // Attribute
11     10 usages
12     private List<Media> itemsOrdered = new ArrayList<>();
13     5 usages
14     private int numOfDVDs;
15
16     // Constructor
17     2 usages  🧑 manhnghuyen41
18     > public Cart() { numOfDVDs = 0; }
19
20     // Method to add a new media
21     2 usages  🧑 manhnghuyen41
22     public void addMedia(Media media) {
23         if (itemsOrdered.contains(media)) {
24             System.out.println("Media is already in the list");
25         }
26         return;
27     }
28 }
```

Figure 19 Cart class code

```
23     }
24     itemsOrdered.add(media);
25     if (media.getClass() == DigitalVideoDisc.class) {
26         numOfDVDs++;
27     }
28     System.out.println("Added");
29 }
30
31 // Method to remove a media
32 1 usage  🧑 manhnghuyen41
33 public void removeMedia(Media media) {
34     // Search for media
35     int indexOfRemoved = itemsOrdered.indexOf(media);
36
37     // If not found
38     if (indexOfRemoved == -1) {
39         System.out.println("Not found");
40         return;
41     }
42
43     // Remove
44     itemsOrdered.remove(indexOfRemoved);
45     if (media.getClass() == DigitalVideoDisc.class) {
46         numOfDVDs--;
```

Figure 20 Cart class code

```
48         // Notify
49         System.out.println("Removed");
50     }
51
52     // Method to calculate the total cost
53     1 usage  🧑 manhnghuyen41
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  🧑 manhnghuyen41
66     public void printCart() {
67         System.out.println("*****CART*****");
68         System.out.println("Ordered Items:");
69         for (Media media : itemsOrdered) {
70             media.print();
71         }
72         System.out.println("Total cost: " + totalCost());
73     }
```

Figure 21 Cart class code

```
71         System.out.println("*****");
72     }
73
74     // Method to search for media in the cart by ID and display the search results.
75     1 usage  🧑 manhnguyen41
76     public Media searchByID(int id) {
77         for (Media media: itemsOrdered) {
78             if (media.getId() == id) {
79                 return media;
80             }
81         }
82         System.out.println("Not found!");
83         return null;
84     }
85
86     // Method to search for media in the cart by title.
87     3 usages  🧑 manhnguyen41
88     public Media searchByTitle(String title) {
89         for (Media media: itemsOrdered) {
90             if (media.isMatch(title)) {
91                 return media;
92             }
93         }
94         System.out.println("Not found!");
95         return null;
96     }
```

Figure 22 Cart class code

```
94     }
95
96     // Method to sort by title and print
1 usage  👤 manhnnguyen41
97     ✓ public void sortByTitle() {
98         itemsOrdered.sort(Media.COMPARE_BY_TITLE);
99         printCart();
100     }
101
102     // Method to sort by cost and print
1 usage  👤 manhnnguyen41
103     ✓ public void sortByCost() {
104         itemsOrdered.sort(Media.COMPARE_BY_COST);
105         printCart();
106     }
107
108     // Getter and Setter
109
1 usage  👤 manhnnguyen41
110     > public int getNumOfDVDs() { return numOfDVDs; }
113
114     no usages  👤 manhnnguyen41
114     > public void setNumOfDVDs(int numOfDVDs) { this.numOfDVDs = numOfDVDs; }
117 }
```

Figure 23 Cart class code

9. Update the Store class

```
1  package lab04.AimsProject;
2
3  > import ...
7
4 usages  👤 manhnguyen41
8  public class Store {
9      // Attribute
10     8 usages
11     private List<Media> itemsInStore = new ArrayList<>();
12
13     // Constructor
14     2 usages  👤 manhnguyen41
15     public Store() {
16
17     // Method to add a media
18     20 usages  👤 manhnguyen41
19     public void addMedia(Media media) {
20         if (itemsInStore.contains(media)) {
21             System.out.println("Media is already in the list");
22             return;
23         }
24         itemsInStore.add(media);
25         System.out.println("Added");
26     }
```

Figure 24 Store class code

```
26 // Method to remove a media
   1 usage  🧑 manhnnguyen41
27 public void removeMedia(Media media) {
28     // Search for disc
29     int indexOfRemoved = itemsInStore.indexOf(media);
30
31     // If not found
32     if (indexOfRemoved == -1) {
33         System.out.println("Not found");
34         return;
35     }
36
37     // Remove
38     itemsInStore.remove(indexOfRemoved);
39
40     // Notify
41     System.out.println("Removed");
42 }
43
44 // Method to print all item in store
   1 usage  🧑 manhnnguyen41
45 public void printStore() {
46     System.out.println("*****STORE*****");
47     System.out.println("Items in store:");
48     for (Media media : itemsInStore) {
```

Figure 25 Store class code

```
49         media.print();
50     }
51     System.out.println("*****");
52 }
53
54 // Method to search for media in the store by title.
55 4 usages  📄 manhnguyen41
56 public Media searchByTitle(String title) {
57     for (Media media: itemsInStore) {
58         if (media.isMatch(title)) {
59             return media;
60         }
61     }
62     System.out.println("Not found!");
63     return null;
64 }
65
66 // Getter and Setter
67 no usages  📄 manhnguyen41
68 > public List<Media> getItemsInStore() { return itemsInStore; }
69
70 no usages  📄 manhnguyen41
71 > public void setItemsInStore(List<Media> itemsInStore) { this.itemsInStore = itemsInStore; }
72
73 }
```

Figure 26 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    static Store store = new Store();
11    // Create a new cart
12    static Cart cart = new Cart();
13    public static void showMenu() {
14        int command;
15        do {
16            Scanner scanner = new Scanner(System.in);
17            // Show store
18            System.out.println("AIMS: ");
19            System.out.println("-----");
20            System.out.println("1. View store");
21            System.out.println("2. Update store");
```

Figure 27 Aims class code

```
22      System.out.println("3. See current cart");
23      System.out.println("0. Exit");
24      System.out.println("-----");
25      System.out.println("Please choose a number: 0-1-2-3");
26      command = scanner.nextInt();
27
28      // If chose View store
29      if (command == 1) {
30          |   store.printStore();
31          |   storeMenu();
32          |
33      }
34
35      // If chose 2
36      if (command == 2) {
37          |   updateStoreMenu();
38          |
39      }
40
41      // If chose 3
42      if (command == 3) {
43          |   cart.printCart();
44          |   cartMenu();
45          |
46      }
47      } while (command != 0);
48  }
```

Figure 28 Aims class code

```
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      }
```

Figure 29 Aims class code

```
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();
94             mediaDetailsMenu(media);
95         }
```

Figure 30 Aims class code

```
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 }
```

Figure 31 Aims class code

```
121  ✓    public static void mediaDetailsMenu(Media media) {  
122      Scanner scanner = new Scanner(System.in);  
123      // Show menu  
124      System.out.println("Options: ");  
125      System.out.println("-----");  
126      System.out.println("1. Add to cart");  
127      ✓    if (!(media instanceof Book)) {  
128          System.out.println("2. Play");  
129      }  
130      System.out.println("0. Back");  
131      System.out.println("-----");  
132      System.out.print("Please choose a number: 0-1");  
133      ✓    if (!(media instanceof Book)) {  
134          System.out.println("-2");  
135      }  
136      int command = scanner.nextInt();  
137  
138      // If chose 1  
139      ✓    if (command == 1) {  
140          cart.addMedia(media);  
141      }  
142  
143      // If chose 2  
144      ✓    if (command == 2) {  
145          playAMedia(media);
```

Figure 32 Aims class code

```
146     }
147 }
148
2 usages  manhnguyen41
149 ✓ public static void cartMenu() {
150     Scanner scanner = new Scanner(System.in);
151     // Show menu
152     System.out.println("Options: ");
153     System.out.println("-----");
154     System.out.println("1. Filter medias in cart");
155     System.out.println("2. Sort medias in cart");
156     System.out.println("3. Remove media from cart");
157     System.out.println("4. Play a media");
158     System.out.println("5. Place order");
159     System.out.println("0. Back");
160     System.out.println("-----");
161     System.out.println("Please choose a number: 0-1-2-3-4-5");
162     int command = scanner.nextInt();
163
164     // If chose 1
165     ✓ if (command == 1) {
166         filterCartMenu();
167     }
168
169     // If chose 2
```

Figure 33 Aims class code

```
170         if (command == 2) {
171             sortCartMenu();
172         }
173
174         // If chose 3
175         if (command == 3) {
176             removeMediaFromCart();
177         }
178
179         // If chose 4
180         if (command == 4) {
181             Media media;
182             do {
183                 System.out.println("Enter the title of the media: ");
184                 String title = scanner.nextLine();
185                 media = cart.searchByTitle(title);
186             } while (media == null);
187             playAMedia(media);
188         }
189
190         // If chose 5
191         if (command == 5) {
192             System.out.println("Order is created");
193             cart = new Cart();
194         }
```

Figure 34 Aims class code


```
195     }
196
197     1 usage  🧑 manhnghuyen41
198     public static void filterCartMenu() {
199         Scanner scanner = new Scanner(System.in);
200         // Show menu
201         System.out.println("Options: ");
202         System.out.println("-----");
203         System.out.println("1. By id");
204         System.out.println("2. By title");
205         System.out.println("0. Back");
206         System.out.println("-----");
207         System.out.println("Please choose a number: 0-1-2");
208         int command = scanner.nextInt();
209
210         // If chose 1
211         if (command == 1) {
212             System.out.println("Enter the id: ");
213             int id = scanner.nextInt();
214             Media media = cart.searchByID(id);
215             if (media != null) {
216                 media.print();
217             }
218         }
219     }
```

Figure 35 Aims class code

```
219 // If chose 2
220 if (command == 2) {
221     System.out.println("Enter the title: ");
222     scanner.nextLine();
223     String title = scanner.nextLine();
224     Media media = cart.searchByTitle(title);
225     if (media != null) {
226         media.print();
227     }
228 }
229 }
230
231 1 usage  🧑 manhnghuyen41
232 public static void sortCartMenu() {
233     Scanner scanner = new Scanner(System.in);
234     // Show menu
235     System.out.println("Options: ");
236     System.out.println("-----");
237     System.out.println("1. By title");
238     System.out.println("2. By cost");
239     System.out.println("0. Back");
240     System.out.println("-----");
241     System.out.println("Please choose a number: 0-1-2");
242     int command = scanner.nextInt();
```

Figure 36 Aims class code

```
243         // If chose 1
244         if (command == 1) {
245             cart.sortByTitle();
246         }
247
248         // If chose 2
249         if (command == 2) {
250             cart.sortByCost();
251         }
252     }
253
254     1 usage  🧑 manhnghuyen41
255     public static void removeMediaFromCart() {
256         Scanner scanner = new Scanner(System.in);
257         Media media;
258         do {
259             System.out.println("Enter the title of the media: ");
260             String title = scanner.nextLine();
261             media = cart.searchByTitle(title);
262         } while (media == null);
263         cart.removeMedia(media);
264     }
265
266     1 usage  🧑 manhnghuyen41
267     public static void removeMediaFromStore() {
```

Figure 37 Aims class code

```
266 Scanner scanner = new Scanner(System.in);
267 Media media;
268 do {
269     System.out.println("Enter the title of the media: ");
270     String title = scanner.nextLine();
271     media = store.searchByTitle(title);
272 } while (media == null);
273 store.removeMedia(media);
274 }
275
3 usages  🧑 manhnghuyen41
276 public static void playAMedia(Media media) {
277     if (media instanceof DigitalVideoDisc dvd) {
278         dvd.play();
279     } else if (media instanceof CompactDisc cd) {
280         cd.play();
281     }
282 }
283
1 usage  🧑 manhnghuyen41
284 public static void addMediaToCart() {
285     Scanner scanner = new Scanner(System.in);
286     Media media;
287     do {
288         System.out.println("Enter the title of the media: ");
```

Figure 38 Aims class code

```

289         String title = scanner.nextLine();
290         media = store.searchByTitle(title);
291     } while (media == null);
292     cart.addMedia(media);
293     if (media instanceof DigitalVideoDisc) {
294         System.out.println("Number of DVDs in the current cart: " + cart.getNumOfDVDs());
295     }
296 }
297
298 public static void main(String[] args) {
299     // Create new media and add them to the store
300     // Adding DVDs
301     Media dvd1 = new lab04.AimsProject.Media.DigitalVideoDisc( id: 1, title: "Inception",
302         category: "Science Fiction", director: "Christopher Nolan", length: 148, cost: 19.99f);
303     store.addMedia(dvd1);
304
305     Media dvd2 = new lab04.AimsProject.Media.DigitalVideoDisc( id: 2, title: "The Dark Knight",
306         category: "Action", director: "Christopher Nolan", length: 152, cost: 17.99f);
307     store.addMedia(dvd2);
308
309     Media dvd3 = new DigitalVideoDisc( id: 7, title: "Interstellar",
310         category: "Science Fiction", director: "Christopher Nolan", length: 169, cost: 21.99f);
311     store.addMedia(dvd3);

```

Figure 39 Aims class code

```

313     // Adding CDs
314     Media cd1 = new CompactDisc( id: 3, title: "Random Access Memories",
315         category: "Electronic", cost: 15.99f, director: "Daft Punk", length: 13, artist: "Daft Punk");
316     store.addMedia(cd1);
317
318     Media cd2 = new CompactDisc( id: 4, title: "25",
319         category: "Pop", cost: 14.99f, director: "Adele", length: 11, artist: "Adele");
320     store.addMedia(cd2);
321
322     Media cd3 = new CompactDisc( id: 8, title: "Lover",
323         category: "Pop", cost: 17.99f, director: "Taylor Swift", length: 18, artist: "Taylor Swift");
324     store.addMedia(cd3);
325
326     // Adding Books
327     Media book1 = new Book( id: 5, title: "The Silent Patient",
328         category: "Thriller", cost: 14.95f);
329     store.addMedia(book1);
330
331     Media book2 = new Book( id: 6, title: "Where the Crawdads Sing",
332         category: "Mystery", cost: 12.99f);
333     store.addMedia(book2);
334
335     Media book3 = new Book( id: 9, title: "Educated",
336         category: "Memoir", cost: 16.95f);

```

Figure 40 Aims class code

```

337         store.addMedia(book3);
338
339         Media book4 = new Book( id: 10, title: "Becoming",
340                                category: "Autobiography", cost: 22.99f);
341         store.addMedia(book4);
342         showMenu();
343     }
344 }

```

Figure 41 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$
*****

```

Figure 42 Aims demo

```
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
```

Figure 43 Aims demo

```
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 44 Aims demo

14. Answer the questions

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

-Ví dụ như sau:

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

Figure 47 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 48 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 49 Question code