

# Lab04: Inheritance and Polymorphism

## Section 1: Import the existing project into the workspace of Eclipse

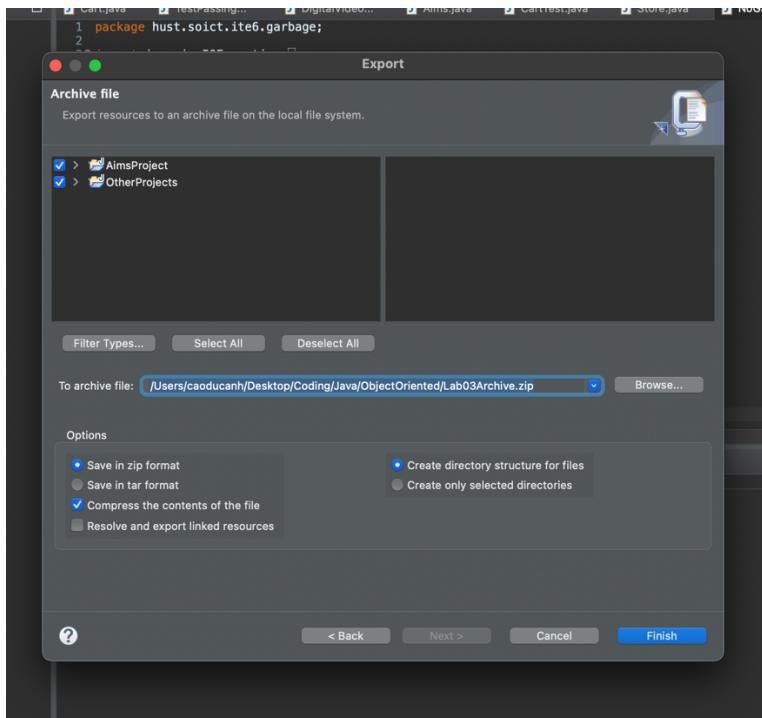


Figure 0-1 Archive the previous lab's projects to a compressed file

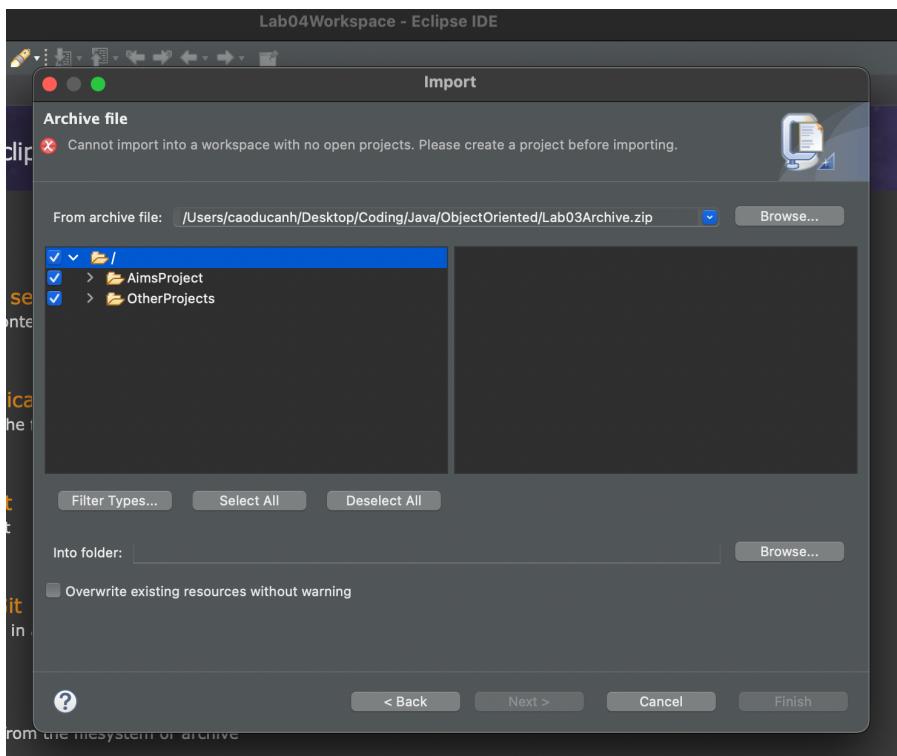
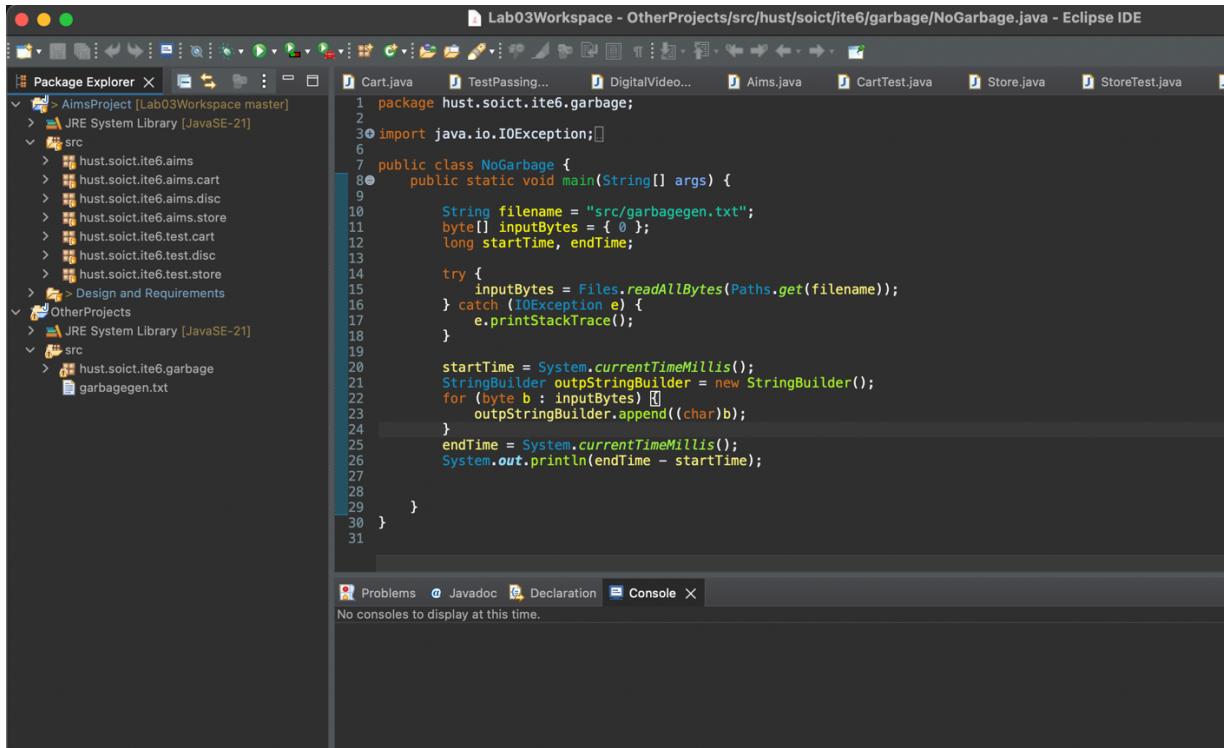


Figure 0-2 Import the archive to the new workspace



```

1 package hust.soict.ite6.garbage;
2
3 import java.io.IOException;
4
5 public class NoGarbage {
6     public static void main(String[] args) {
7         String filename = "src/garbagegen.txt";
8         byte[] inputBytes = { 0 };
9         long startTime, endTime;
10
11         try {
12             inputBytes = Files.readAllBytes(Paths.get(filename));
13         } catch (IOException e) {
14             e.printStackTrace();
15         }
16
17         startTime = System.currentTimeMillis();
18         StringBuilder outStringBuilder = new StringBuilder();
19         for (byte b : inputBytes) {
20             outStringBuilder.append((char)b);
21         }
22         endTime = System.currentTimeMillis();
23         System.out.println(endTime - startTime);
24
25     }
26
27 }
28
29 }
30
31

```

Figure 0-3 Imported project

## Section 2: Additional requirements of AIMS

No tasks were given in this section

## Section 3: Creating the Book class

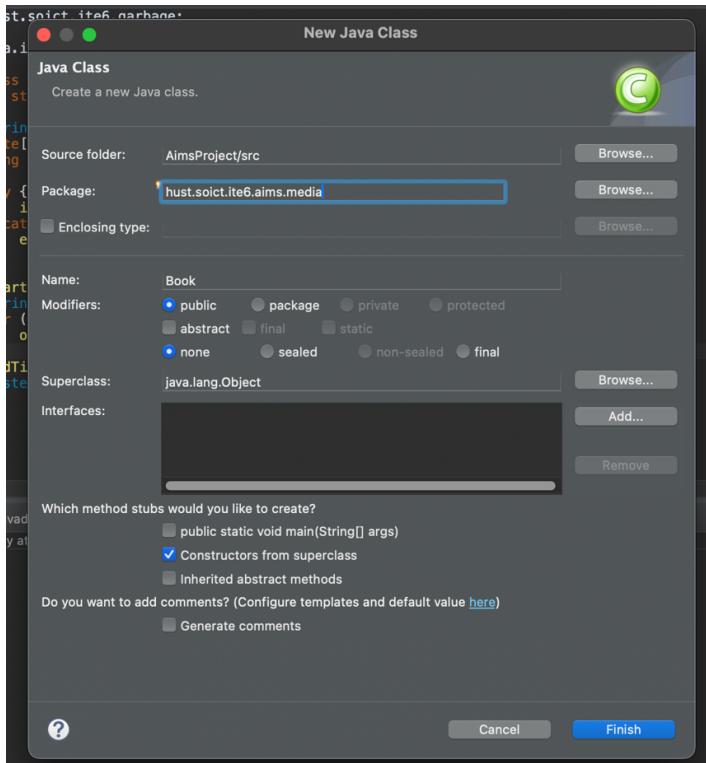
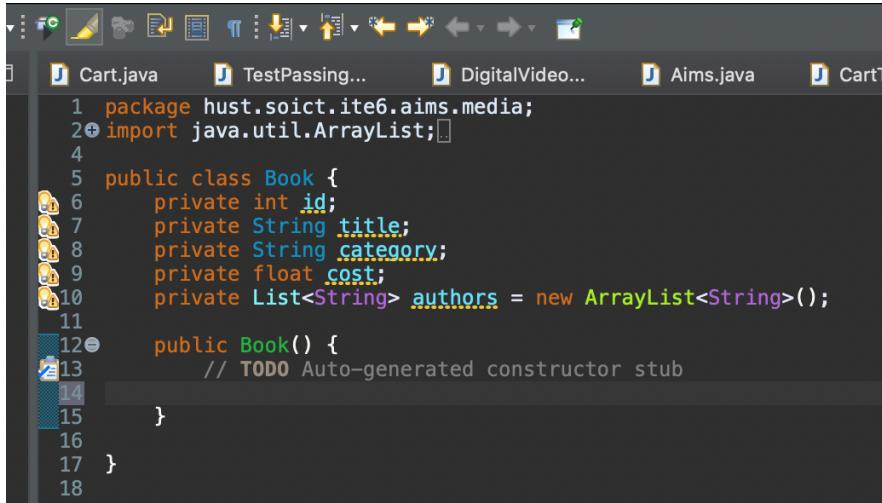


Figure 0-4 Creating the new 'Book' class

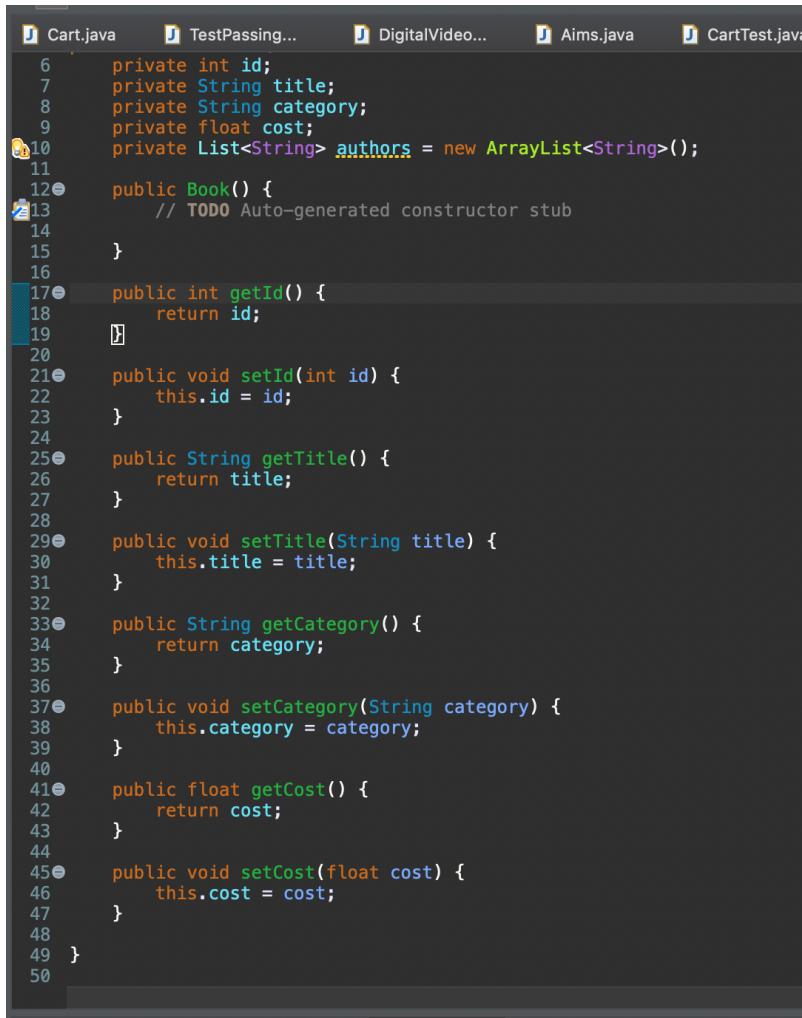


```

1 package hust.soict.ite6.aims.media;
2 import java.util.ArrayList;
3
4
5 public class Book {
6     private int id;
7     private String title;
8     private String category;
9     private float cost;
10    private List<String> authors = new ArrayList<String>();
11
12    public Book() {
13        // TODO Auto-generated constructor stub
14    }
15
16
17 }
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

Figure 0-5 Creating private attributes for 'Book' class

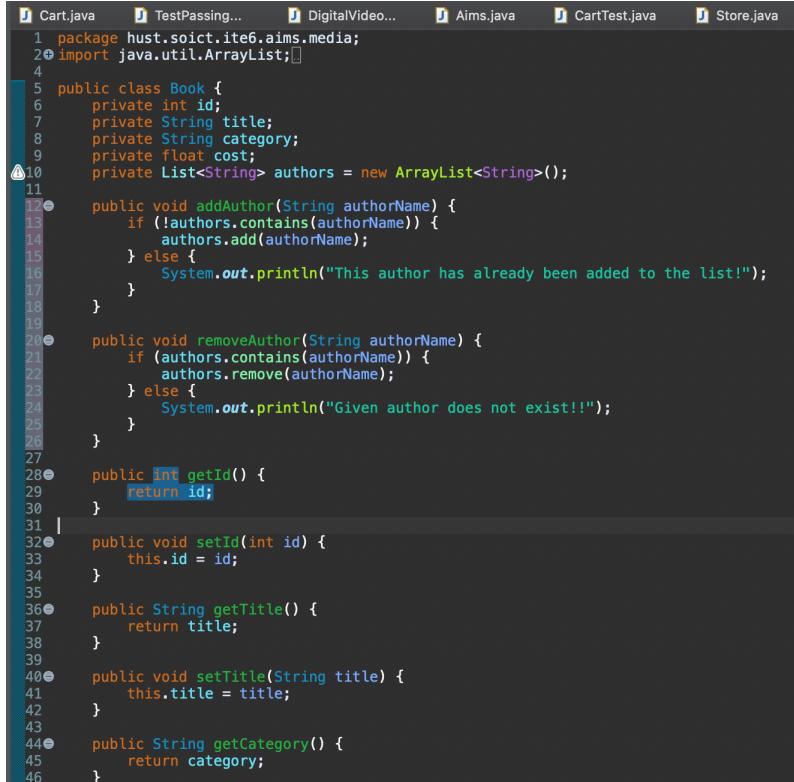


```

6     private int id;
7     private String title;
8     private String category;
9     private float cost;
10    private List<String> authors = new ArrayList<String>();
11
12    public Book() {
13        // TODO Auto-generated constructor stub
14    }
15
16
17    public int getId() {
18        return id;
19    }
20
21    public void setId(int id) {
22        this.id = id;
23    }
24
25    public String getTitle() {
26        return title;
27    }
28
29    public void setTitle(String title) {
30        this.title = title;
31    }
32
33    public String getCategory() {
34        return category;
35    }
36
37    public void setCategory(String category) {
38        this.category = category;
39    }
40
41    public float getCost() {
42        return cost;
43    }
44
45    public void setCost(float cost) {
46        this.cost = cost;
47    }
48
49 }
50

```

Figure 0-6 Generate public accessors for private fields



```

1 package hust.soict.ite6.aims.media;
2 import java.util.ArrayList;
3
4
5 public class Book {
6     private int id;
7     private String title;
8     private String category;
9     private float cost;
10    private List<String> authors = new ArrayList<String>();
11
12    public void addAuthor(String authorName) {
13        if (!authors.contains(authorName)) {
14            authors.add(authorName);
15        } else {
16            System.out.println("This author has already been added to the list!");
17        }
18    }
19
20    public void removeAuthor(String authorName) {
21        if (authors.contains(authorName)) {
22            authors.remove(authorName);
23        } else {
24            System.out.println("Given author does not exist!!!");
25        }
26    }
27
28    public int getId() {
29        return id;
30    }
31
32    public void setId(int id) {
33        this.id = id;
34    }
35
36    public String getTitle() {
37        return title;
38    }
39
40    public void setTitle(String title) {
41        this.title = title;
42    }
43
44    public String getCategory() {
45        return category;
46    }
}

```

Figure 0-7 Methods to add and remove authors from list

## Section 4: Creating the abstract 'Media' class

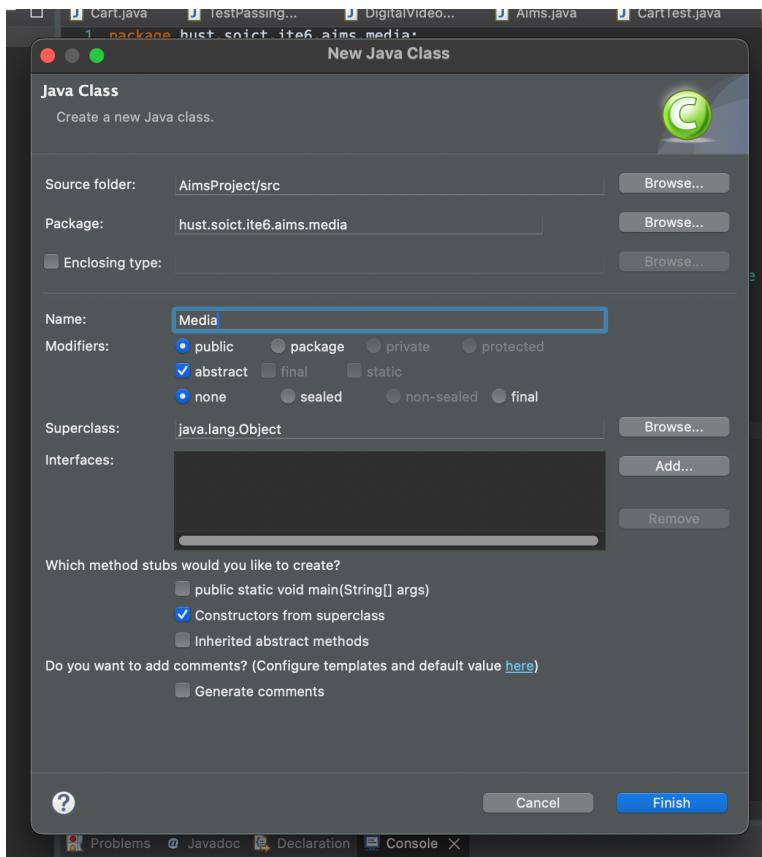
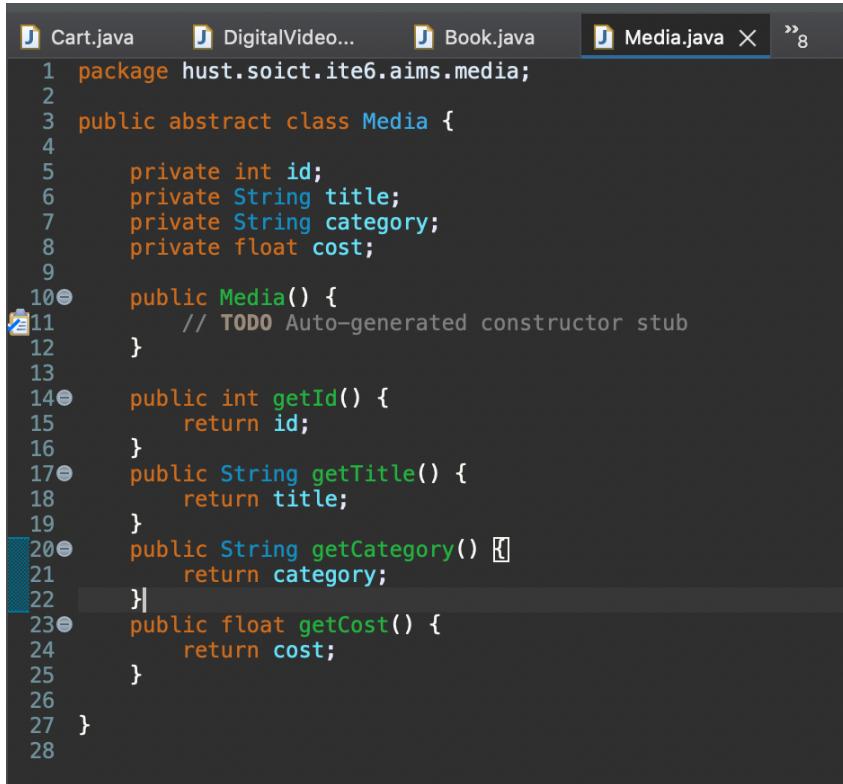


Figure 0-8 Creating new abstract class 'Media'

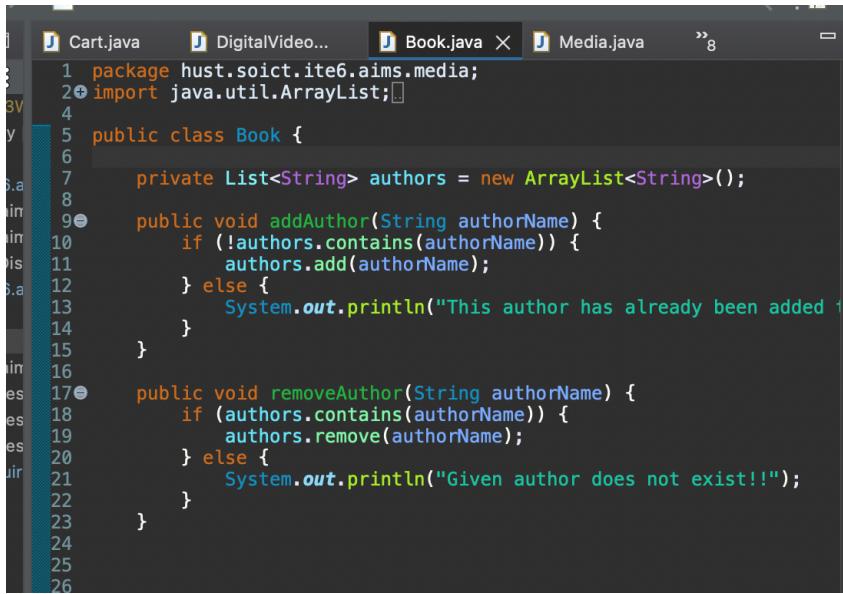


```

1 package hust.soict.ite6.aims.media;
2
3 public abstract class Media {
4
5     private int id;
6     private String title;
7     private String category;
8     private float cost;
9
10    public Media() {
11        // TODO Auto-generated constructor stub
12    }
13
14    public int getId() {
15        return id;
16    }
17    public String getTitle() {
18        return title;
19    }
20    public String getCategory() {
21        return category;
22    }
23    public float getCost() {
24        return cost;
25    }
26
27 }
28

```

Figure 0-9 Creating private attributes and generating public accessors

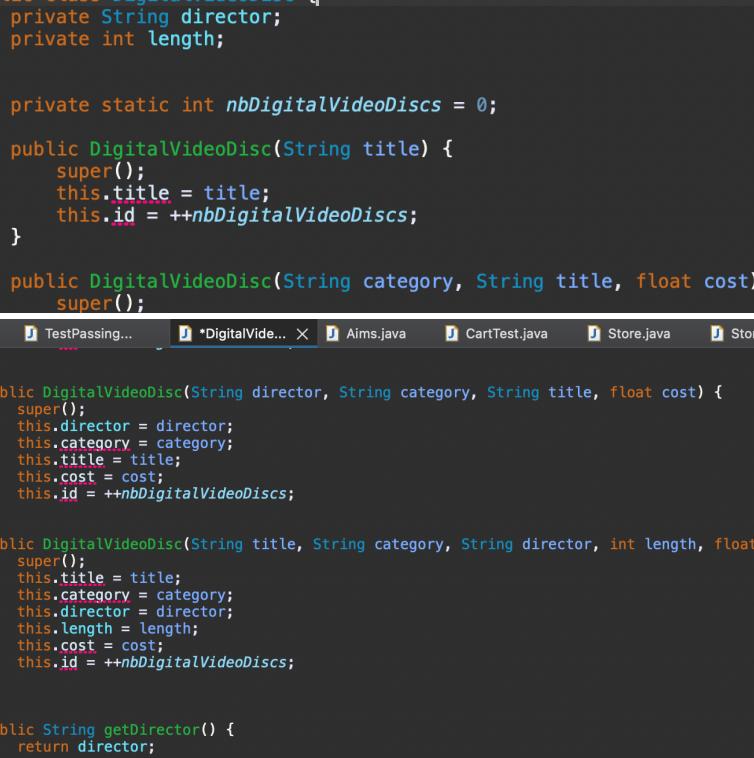


```

1 package hust.soict.ite6.aims.media;
2
3 import java.util.ArrayList;
4
5 public class Book {
6
7     private List<String> authors = new ArrayList<String>();
8
9     public void addAuthor(String authorName) {
10        if (!authors.contains(authorName)) {
11            authors.add(authorName);
12        } else {
13            System.out.println("This author has already been added!");
14        }
15    }
16
17    public void removeAuthor(String authorName) {
18        if (authors.contains(authorName)) {
19            authors.remove(authorName);
20        } else {
21            System.out.println("Given author does not exist!!!");
22        }
23    }
24
25
26

```

Figure 0-10 Deleting mutual attributes and their accessors in the 'Book' class



The screenshot shows a Java code editor with the following code for the `DigitalVideoDisc` class:

```
1 package hust.soict.ite6.aims.disc;
2
3 public class DigitalVideoDisc {
4     private String director;
5     private int length;
6
7
8     private static int nbDigitalVideoDiscs = 0;
9
10    public DigitalVideoDisc(String title) {
11        super();
12        this.title = title;
13        this.id = ++nbDigitalVideoDiscs;
14    }
15
16    public DigitalVideoDisc(String category, String title, float cost) {
17        super();
18    }
19
20    public DigitalVideoDisc(String director, String category, String title, float cost) {
21        super();
22        this.director = director;
23        this.category = category;
24        this.title = title;
25        this.cost = cost;
26        this.id = ++nbDigitalVideoDiscs;
27    }
28
29    public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
30        super();
31        this.title = title;
32        this.category = category;
33        this.director = director;
34        this.length = length;
35        this.cost = cost;
36        this.id = ++nbDigitalVideoDiscs;
37    }
38
39    public String getDirector() {
40        return director;
41    }
42
43    public int getLength() {
44        return length;
45    }
46
47    @Override
```

The code contains several errors, indicated by red markers and underlines:

- Line 12: `this.title = title;` (red underline)
- Line 13: `this.id = ++nbDigitalVideoDiscs;` (red underline)
- Line 25: `this.title = title;` (red underline)
- Line 26: `this.cost = cost;` (red underline)
- Line 35: `this.director = director;` (red underline)
- Line 36: `this.length = length;` (red underline)
- Line 39: `this.cost = cost;` (red underline)
- Line 40: `return director;` (red underline)
- Line 47: `public int getLength() {` (red underline)
- Line 48:  `return length;` (red underline)

Figure 0-11 Similarly, delete mutual attributes and their public accessors in 'DigitalVideoDisc' class

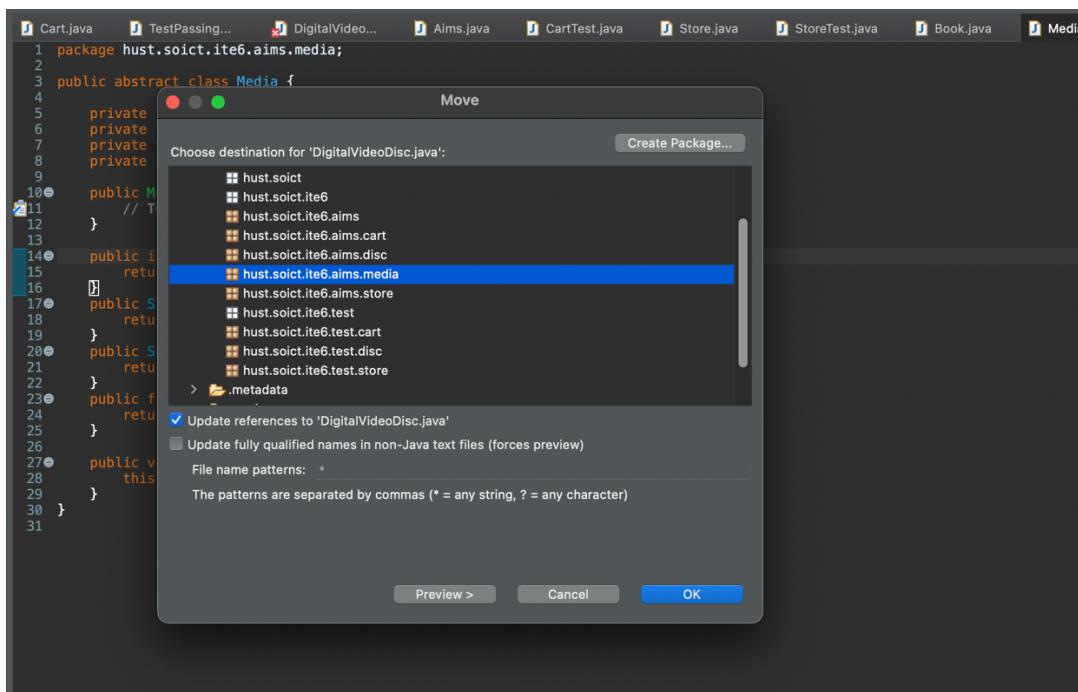
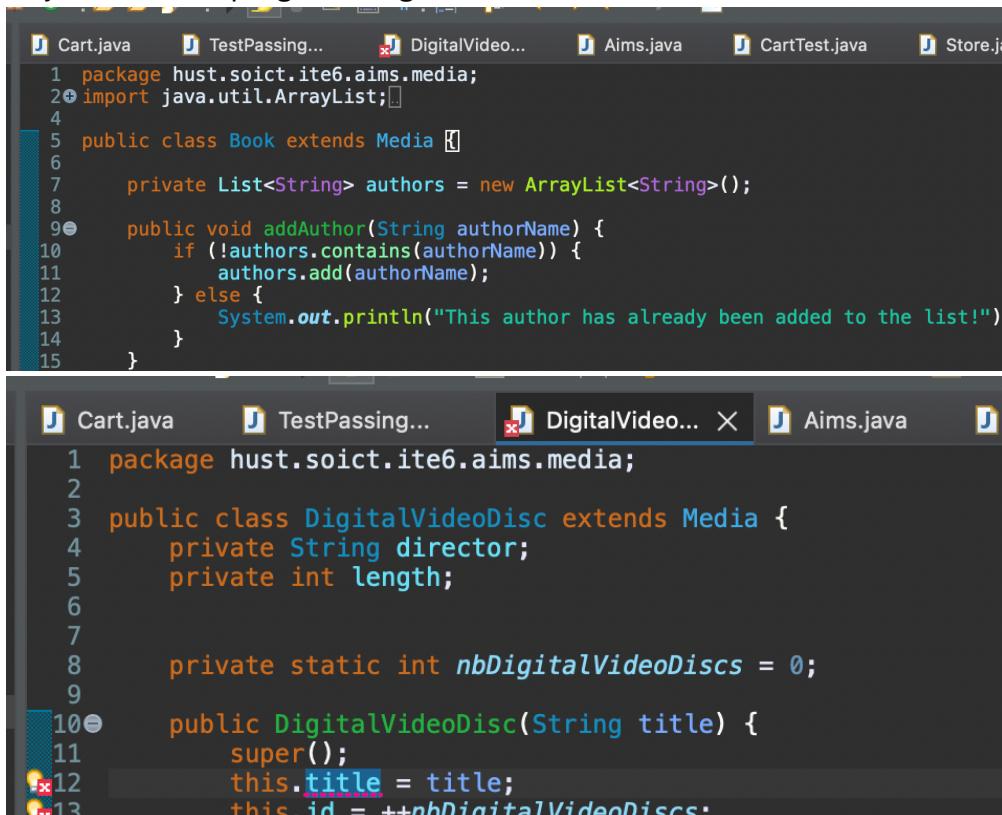


Figure 0-12 Move 'DigitalVideoDisc' to 'hust.soict.ite6.aims.media'



```

1 package hust.soict.ite6.aims.media;
2 import java.util.ArrayList;
3
4
5 public class Book extends Media {
6
7     private List<String> authors = new ArrayList<String>();
8
9     public void addAuthor(String authorName) {
10         if (!authors.contains(authorName)) {
11             authors.add(authorName);
12         } else {
13             System.out.println("This author has already been added to the list!");
14         }
15     }
16 }

```

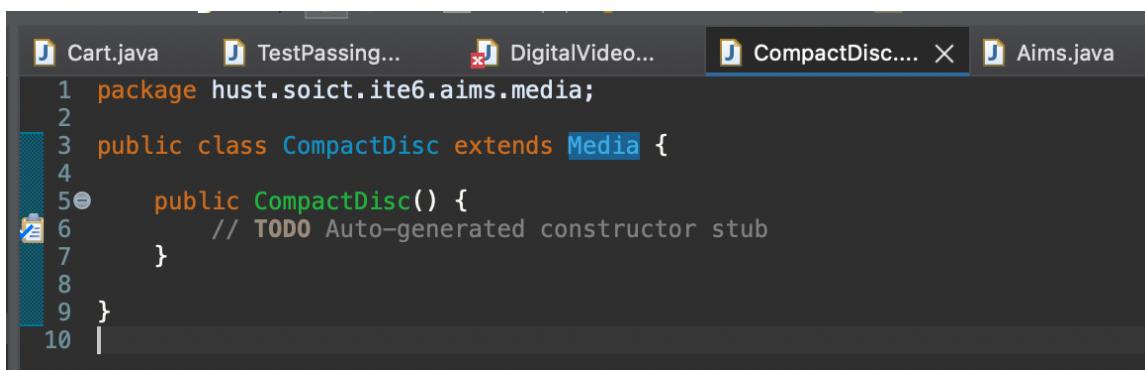
```

1 package hust.soict.ite6.aims.media;
2
3 public class DigitalVideoDisc extends Media {
4     private String director;
5     private int length;
6
7
8     private static int nbDigitalVideoDiscs = 0;
9
10    public DigitalVideoDisc(String title) {
11        super();
12        this.title = title;
13        this.id = ++nbDigitalVideoDiscs;
14    }
15 }

```

Figure 0-13 Extends 'Book' and 'DigitalVideoDisc'

## Section 5: Creating the 'CompactDisc' class

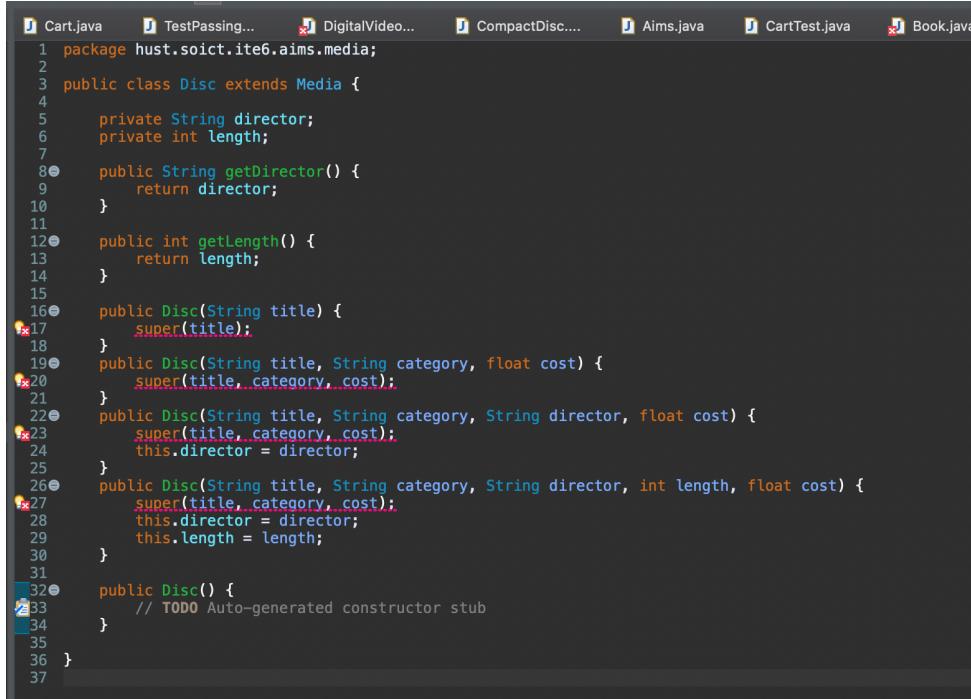


```

1 package hust.soict.ite6.aims.media;
2
3 public class CompactDisc extends Media {
4
5     public CompactDisc() {
6         // TODO Auto-generated constructor stub
7     }
8
9 }
10

```

Figure 0-14 Creating the 'CompactDisc' class which extends 'Media'

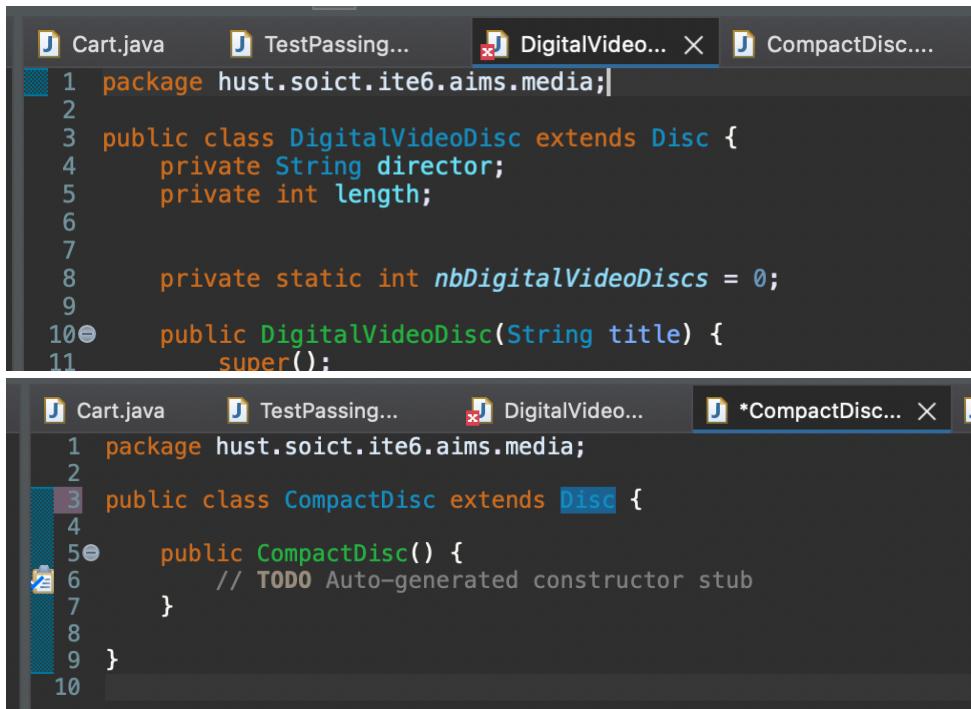


```

1 package hust.soict.ite6.aims.media;
2
3 public class Disc extends Media {
4
5     private String director;
6     private int length;
7
8     public String getDirector() {
9         return director;
10    }
11
12    public int getLength() {
13        return length;
14    }
15
16    public Disc(String title) {
17        super(title);
18    }
19    public Disc(String title, String category, float cost) {
20        super(title, category, cost);
21    }
22    public Disc(String title, String category, String director, float cost) {
23        super(title, category, cost);
24        this.director = director;
25    }
26    public Disc(String title, String category, String director, int length, float cost) {
27        super(title, category, cost);
28        this.director = director;
29        this.length = length;
30    }
31
32    public Disc() {
33        // TODO Auto-generated constructor stub
34    }
35
36 }
37

```

Figure 0-15 Creating 'Disc' class and declare attributes and generate getters and constructors



```

1 package hust.soict.ite6.aims.media;
2
3 public class DigitalVideoDisc extends Disc {
4     private String director;
5     private int length;
6
7     private static int nbDigitalVideoDiscs = 0;
8
9     public DigitalVideoDisc(String title) {
10        super();
11    }

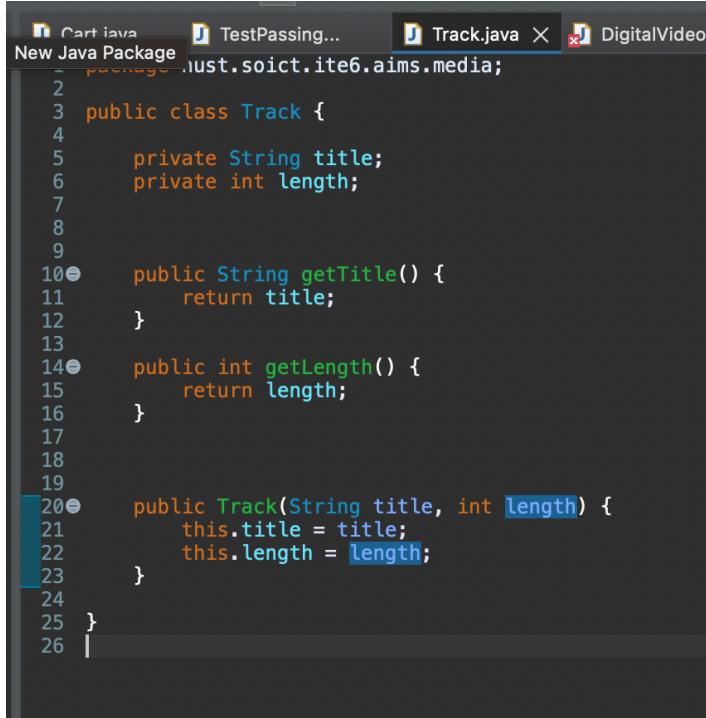
```

```

1 package hust.soict.ite6.aims.media;
2
3 public class CompactDisc extends Disc {
4
5     public CompactDisc() {
6         // TODO Auto-generated constructor stub
7     }
8
9 }
10

```

Figure 0-16 Modify 'DigitalVideoDisc' and 'CompactDisc' to extend 'Disc'

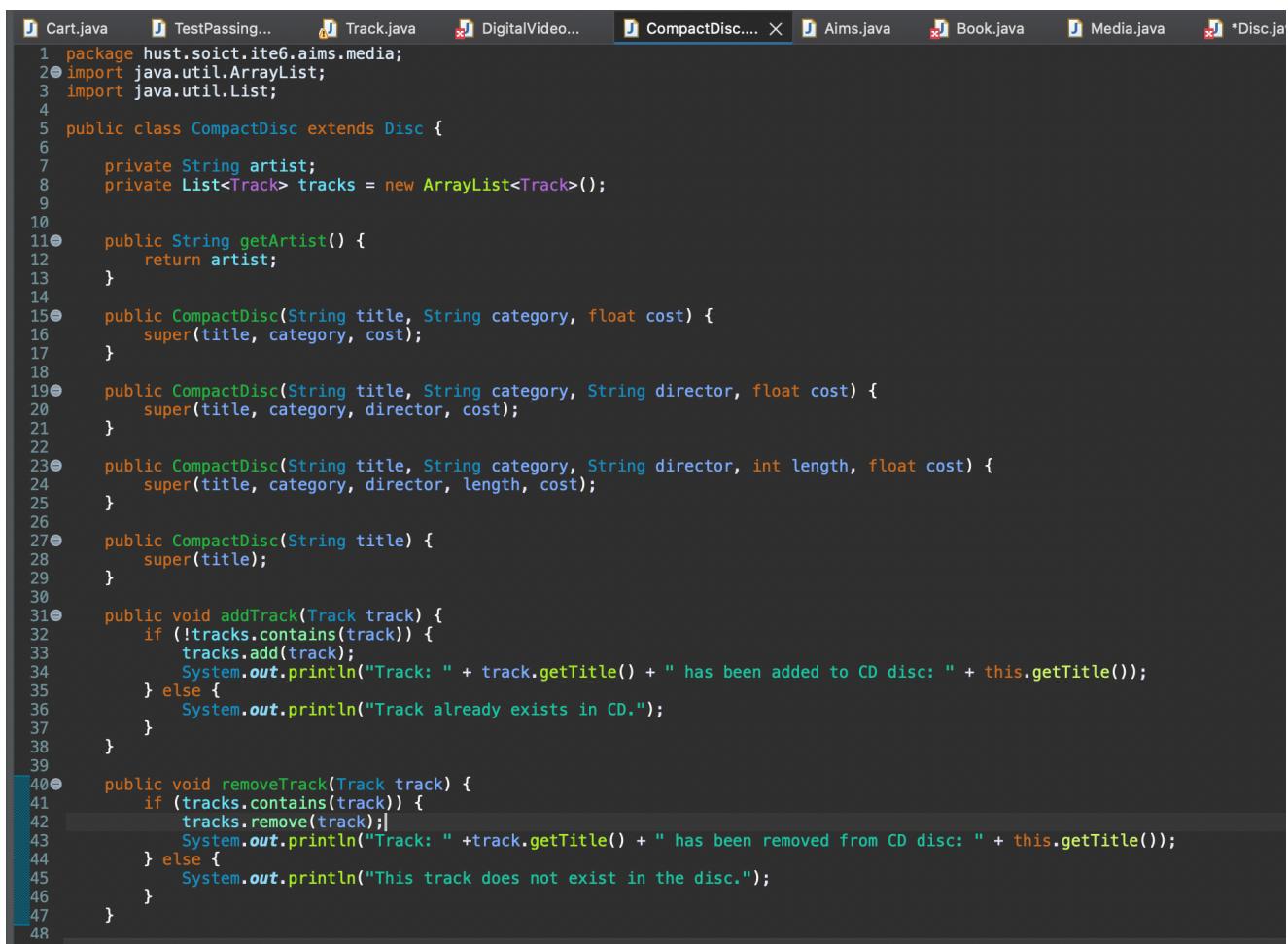


```

1 package hust.soict.ite6.aims.media;
2
3 public class Track {
4
5     private String title;
6     private int length;
7
8
9
10    public String getTitle() {
11        return title;
12    }
13
14    public int getLength() {
15        return length;
16    }
17
18
19
20    public Track(String title, int length) {
21        this.title = title;
22        this.length = length;
23    }
24
25 }
26

```

Figure 0-17 Create new 'Track' class, declare it's attributes and generate getters, constructors



```

1 package hust.soict.ite6.aims.media;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class CompactDisc extends Disc {
7
8     private String artist;
9     private List<Track> tracks = new ArrayList<Track>();
10
11    public String getArtist() {
12        return artist;
13    }
14
15    public CompactDisc(String title, String category, float cost) {
16        super(title, category, cost);
17    }
18
19    public CompactDisc(String title, String category, String director, float cost) {
20        super(title, category, director, cost);
21    }
22
23    public CompactDisc(String title, String category, String director, int length, float cost) {
24        super(title, category, director, length, cost);
25    }
26
27    public CompactDisc(String title) {
28        super(title);
29    }
30
31    public void addTrack(Track track) {
32        if (!tracks.contains(track)) {
33            tracks.add(track);
34            System.out.println("Track: " + track.getTitle() + " has been added to CD disc: " + this.getTitle());
35        } else {
36            System.out.println("Track already exists in CD.");
37        }
38    }
39
40    public void removeTrack(Track track) {
41        if (tracks.contains(track)) {
42            tracks.remove(track);
43            System.out.println("Track: " + track.getTitle() + " has been removed from CD disc: " + this.getTitle());
44        } else {
45            System.out.println("This track does not exist in the disc.");
46        }
47    }
48

```

Figure 0-18 Adding attributes, making public getters, constructors and creating 'addTrack', 'removeTrack' methods for 'CompactDisc'

```
47     }
48
49     public int getLength() {
50         int totalLength = 0;
51         for (Track track : tracks) {
52             totalLength += track.getLength();
53         }
54         return totalLength;
55     }
56
57 }
58 }
```

Figure 0-19 Write 'getLength' method for 'CompactDisc'

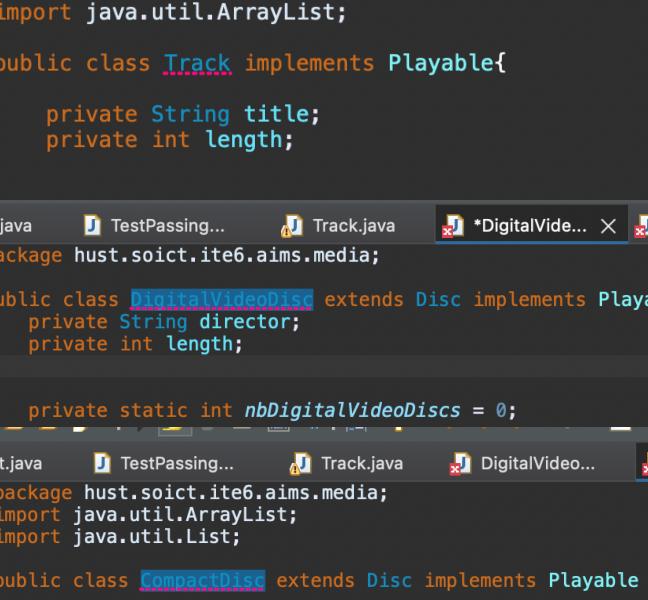
## Section 6: Create the ‘Playable’ interface



Cart.java    TestPassing...    Track.java    DigitalVideo...

```
1 package hust.soict.ite6.aims.media;
2
3 public interface Playable {
4     public void play();
5 }
6 |
```

Figure 0-20 Create 'Playable' interface and add method prototype



Cart.java    TestPassing...    Track.java    DigitalVideoDisc    CompactDisc    Disc

```
1 package hust.soict.ite6.aims.media;
2 import java.util.ArrayList;
3
4 public class Track implements Playable{
5
6     private String title;
7     private int length;
8
9 }
```

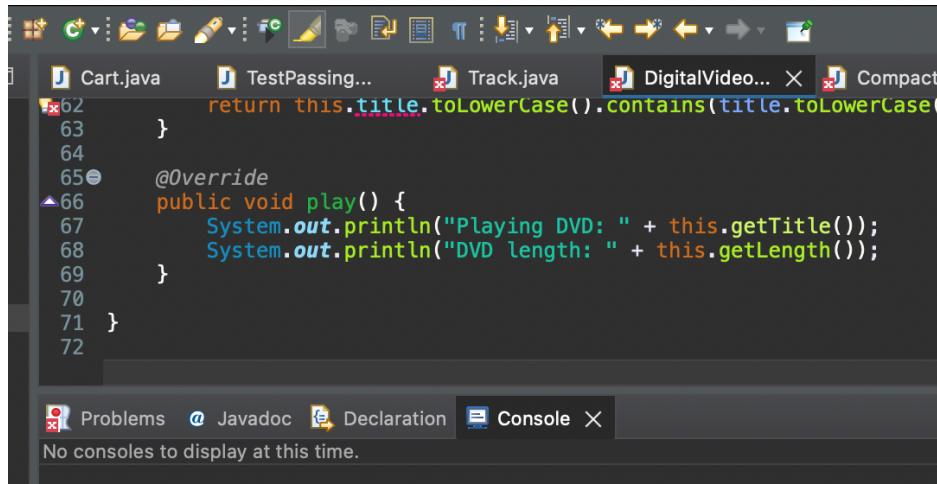
```
Cart.java    TestPassing...    Track.java    *DigitalVideoDisc    CompactDisc    Disc
```

```
1 package hust.soict.ite6.aims.media;
2
3 public class DigitalVideoDisc extends Disc implements Playable {
4     private String director;
5     private int length;
6
7
8     private static int nbDigitalVideoDiscs = 0;
```

```
Cart.java    TestPassing...    Track.java    DigitalVideoDisc    CompactDisc    Disc
```

```
1 package hust.soict.ite6.aims.media;
2 import java.util.ArrayList;
3 import java.util.List;
4
5 public class CompactDisc extends Disc implements Playable {
6
7     private String artist;
8     private List<Track> tracks = new ArrayList<Track>();
9
10 }
```

Figure 0-21 Implements 'Playable' with 'CompactDisc', 'DigitalVideoDisc' and 'Track'

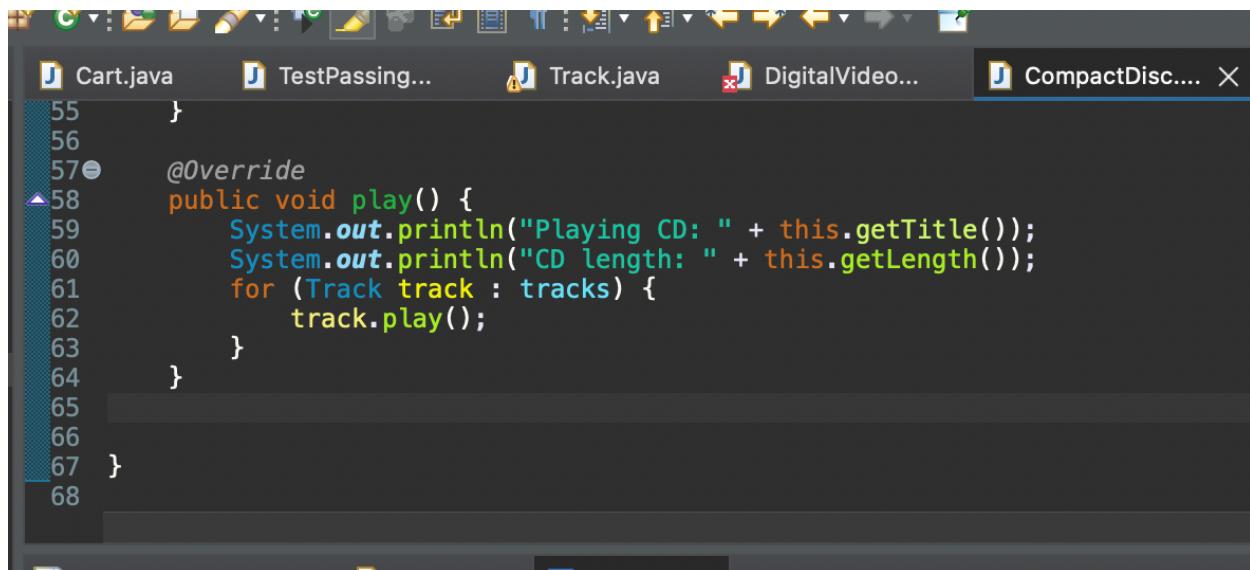


The screenshot shows a Java IDE interface with several tabs at the top: Cart.java, TestPassing..., Track.java, DigitalVideo..., and Compact. The DigitalVideo... tab is active. The code editor displays the DigitalVideoDisc class with the following content:

```
62     }
63 }
64
65 @Override
66 public void play() {
67     System.out.println("Playing DVD: " + this.getTitle());
68     System.out.println("DVD length: " + this.getLength());
69 }
70
71 }
72
```

Below the code editor, there are tabs for Problems, Javadoc, Declaration, and Console. The Console tab is selected, showing the message: "No consoles to display at this time."

Figure 0-22 Add new method 'play()' to 'DigitalVideoDisc' and 'Track'

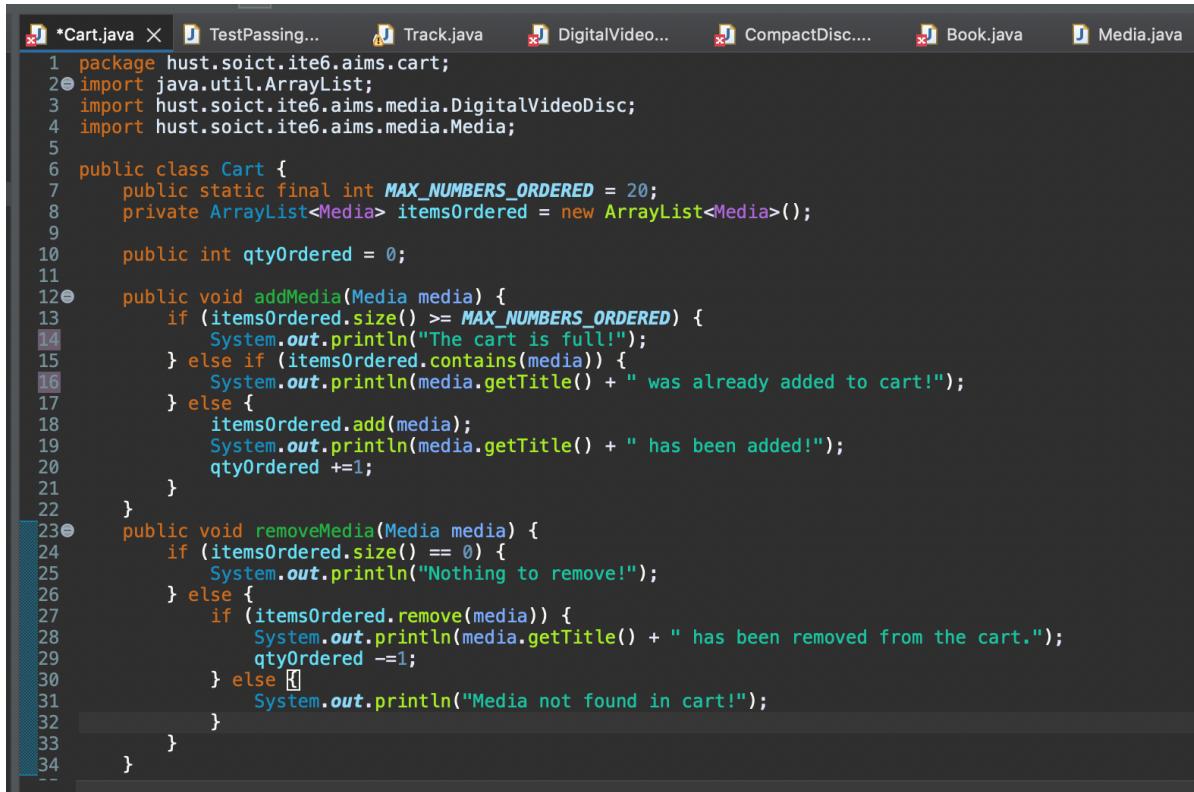


The screenshot shows a Java IDE interface with several tabs at the top: Cart.java, TestPassing..., Track.java, DigitalVideo..., and CompactDisc.... The CompactDisc.... tab is active. The code editor displays the CompactDisc class with the following content:

```
55     }
56
57 @Override
58 public void play() {
59     System.out.println("Playing CD: " + this.getTitle());
60     System.out.println("CD length: " + this.getLength());
61     for (Track track : tracks) {
62         track.play();
63     }
64 }
65
66
67 }
68
```

Figure 0-23 'play()' method for 'CompactDisc' which loops through every tracks of a disc

## Section 7: Update the 'Cart' class to work with 'Media'

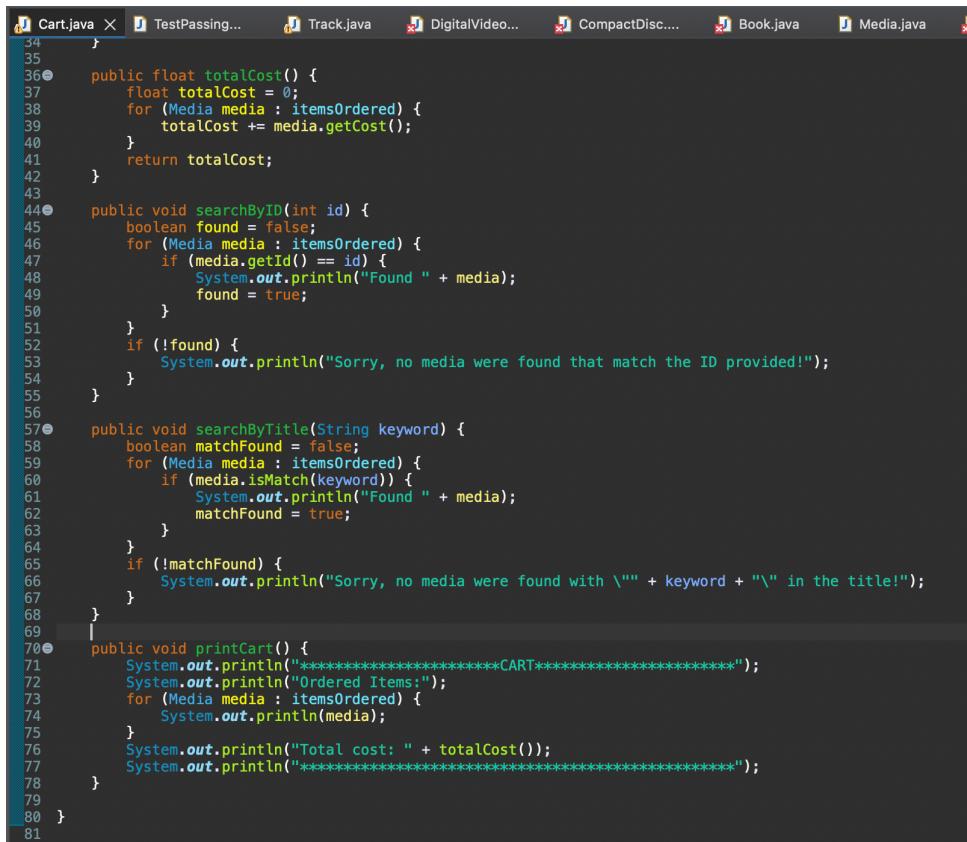


```

1 package hust.soict.ite6.aims.cart;
2 import java.util.ArrayList;
3 import hust.soict.ite6.aims.media.DigitalVideoDisc;
4 import hust.soict.ite6.aims.media.Media;
5
6 public class Cart {
7     public static final int MAX_NUMBERS_ORDERED = 20;
8     private ArrayList<Media> itemsOrdered = new ArrayList<Media>();
9
10    public int qtyOrdered = 0;
11
12    public void addMedia(Media media) {
13        if (itemsOrdered.size() >= MAX_NUMBERS_ORDERED) {
14            System.out.println("The cart is full!");
15        } else if (itemsOrdered.contains(media)) {
16            System.out.println(media.getTitle() + " was already added to cart!");
17        } else {
18            itemsOrdered.add(media);
19            System.out.println(media.getTitle() + " has been added!");
20            qtyOrdered += 1;
21        }
22    }
23    public void removeMedia(Media media) {
24        if (itemsOrdered.size() == 0) {
25            System.out.println("Nothing to remove!");
26        } else {
27            if (itemsOrdered.remove(media)) {
28                System.out.println(media.getTitle() + " has been removed from the cart.");
29                qtyOrdered -= 1;
30            } else {
31                System.out.println("Media not found in cart!");
32            }
33        }
34    }
}

```

Figure 0-24 Removing the old 'addDigitalVideoDisc()', 'removeDigitalVideoDisc()' methods and the 'itemsOrdered' array to work with the new 'Media' class



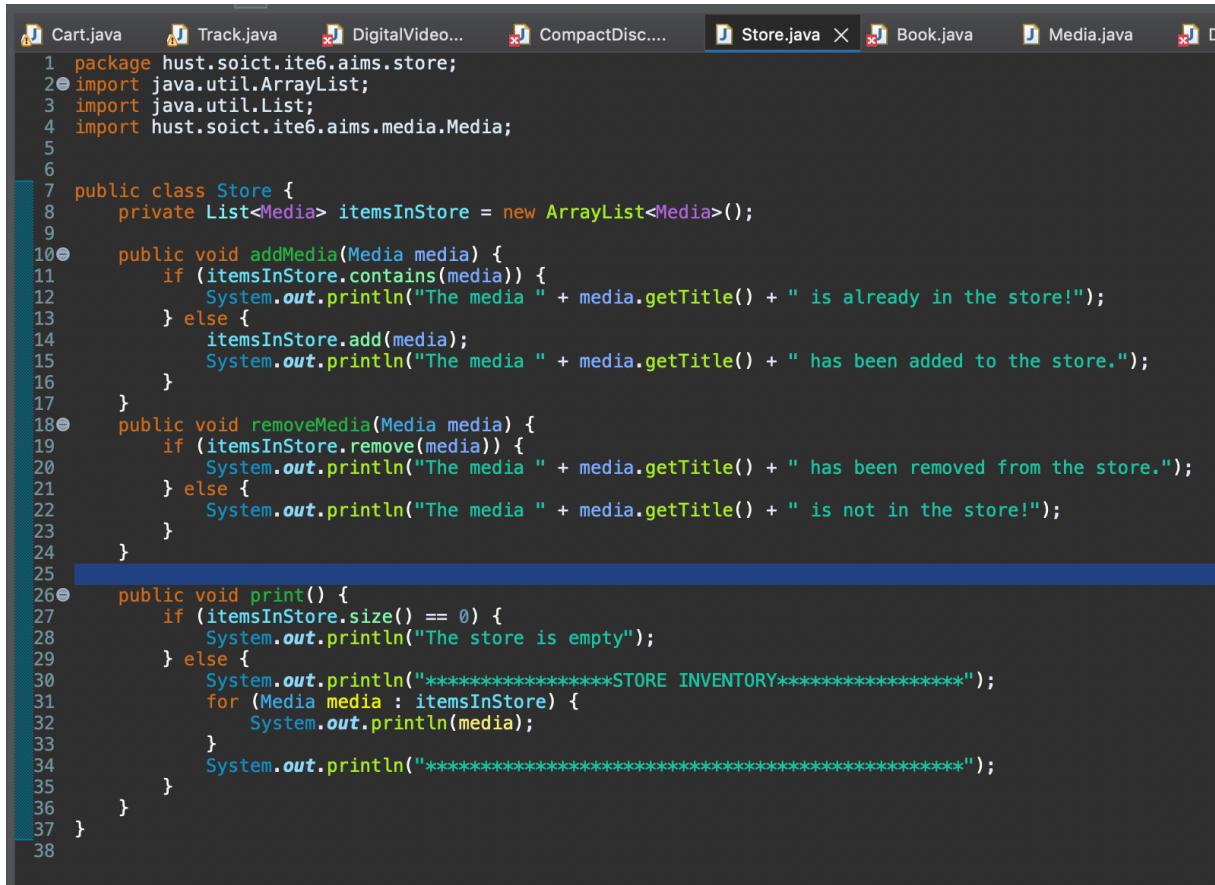
```

34 }
35
36    public float totalCost() {
37        float totalCost = 0;
38        for (Media media : itemsOrdered) {
39            totalCost += media.getCost();
40        }
41        return totalCost;
42    }
43
44    public void searchByID(int id) {
45        boolean found = false;
46        for (Media media : itemsOrdered) {
47            if (media.getId() == id) {
48                System.out.println("Found " + media);
49                found = true;
50            }
51        }
52        if (!found) {
53            System.out.println("Sorry, no media were found that match the ID provided!");
54        }
55    }
56
57    public void searchByTitle(String keyword) {
58        boolean matchFound = false;
59        for (Media media : itemsOrdered) {
60            if (media.isMatch(keyword)) {
61                System.out.println("Found " + media);
62                matchFound = true;
63            }
64        }
65        if (!matchFound) {
66            System.out.println("Sorry, no media were found with '" + keyword + "' in the title!");
67        }
68    }
69
70    public void printCart() {
71        System.out.println("*****CART*****");
72        System.out.println("Ordered Items:");
73        for (Media media : itemsOrdered) {
74            System.out.println(media);
75        }
76        System.out.println("Total cost: " + totalCost());
77        System.out.println("*****");
78    }
79
80 }

```

Figure 0-25 Update other methods to work with the new 'itemsOrdered' object

## Section 8: Update the 'Store' class to work with 'Media'



```
1 package hust.soict.ite6.aims.store;
2 import java.util.ArrayList;
3 import java.util.List;
4 import hust.soict.ite6.aims.media.Media;
5
6
7 public class Store {
8     private List<Media> itemsInStore = new ArrayList<Media>();
9
10    public void addMedia(Media media) {
11        if (itemsInStore.contains(media)) {
12            System.out.println("The media " + media.getTitle() + " is already in the store!");
13        } else {
14            itemsInStore.add(media);
15            System.out.println("The media " + media.getTitle() + " has been added to the store.");
16        }
17    }
18    public void removeMedia(Media media) {
19        if (itemsInStore.remove(media)) {
20            System.out.println("The media " + media.getTitle() + " has been removed from the store.");
21        } else {
22            System.out.println("The media " + media.getTitle() + " is not in the store!");
23        }
24    }
25
26    public void print() {
27        if (itemsInStore.size() == 0) {
28            System.out.println("The store is empty");
29        } else {
30            System.out.println("*****STORE INVENTORY*****");
31            for (Media media : itemsInStore) {
32                System.out.println(media);
33            }
34            System.out.println("*****");
35        }
36    }
37 }
38 }
```

Figure 0-26 Updated 'itemsInStore' object and all methods to use 'Media'

## Section 9: Constructors of whole classes and parent classes

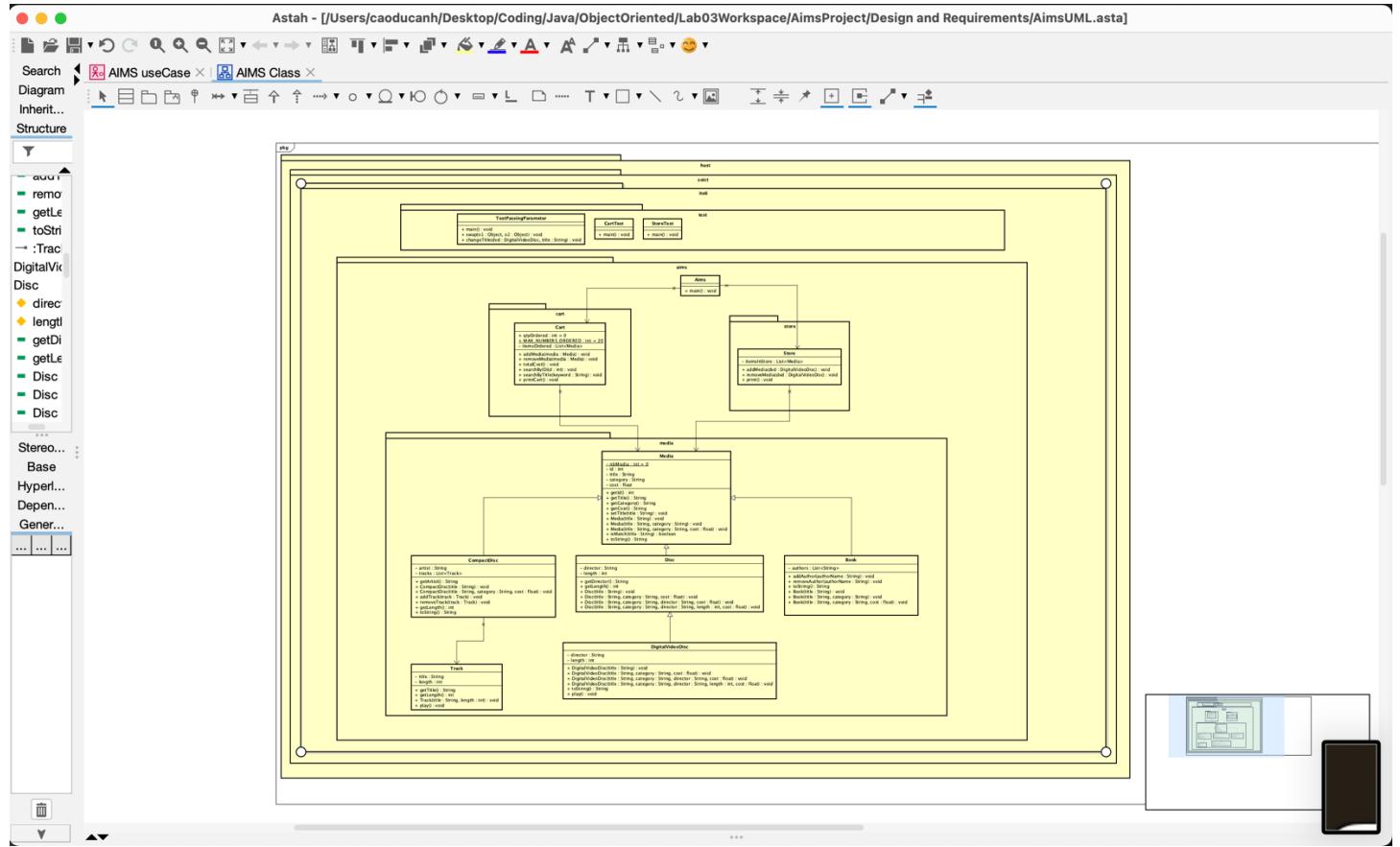


Figure 0-27 Update the project's UML class diagram

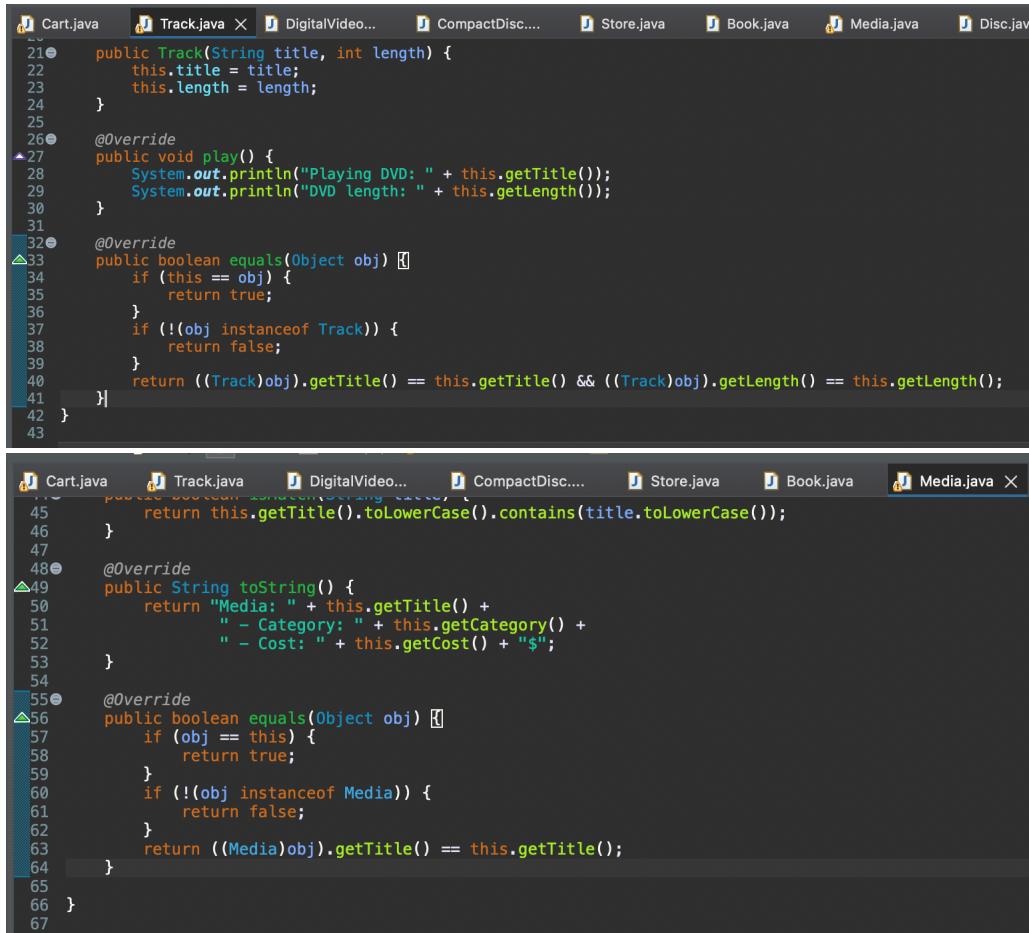
```

1  Cart.java  2  Track.java  3  DigitalVideo...  4  CompactDisc...  5  Store.java  6  Book.java  7  Media.java  8  Disc.java  9  Playable.java  10
11
12  private float cost;
13
14  public int getId() {
15      return id;
16  }
17  public String getTitle() {
18      return title;
19  }
20  public String getCategory() {
21      return category;
22  }
23  public float getCost() {
24      return cost;
25  }
26  public void setTitle(String title) {
27      this.title = title;
28  }
29
30  public Media(String title) {
31      this.title = title;
32  }
33  public Media(String title, String category) {
34      this.title = title;
35      this.category = category;
36  }
37  public Media(String title, String category, float cost) {
38      this.title = title;
39      this.category = category;
40      this.cost = cost;
41  }
42

```

Figure 0-28 Rewrite constructors of parent class

## Section 10: Unique item in a list



```

1 Cart.java   2 Track.java X  3 DigitalVideo...  4 CompactDisc...  5 Store.java  6 Book.java  7 Media.java  8 Disc.java
21  public Track(String title, int length) {
22      this.title = title;
23      this.length = length;
24  }
25
26  @Override
27  public void play() {
28      System.out.println("Playing DVD: " + this.getTitle());
29      System.out.println("DVD length: " + this.getLength());
30  }
31
32  @Override
33  public boolean equals(Object obj) {
34      if (this == obj) {
35          return true;
36      }
37      if (!(obj instanceof Track)) {
38          return false;
39      }
40      return ((Track)obj).getTitle() == this.getTitle() && ((Track)obj).getLength() == this.getLength();
41  }
42 }
43

```

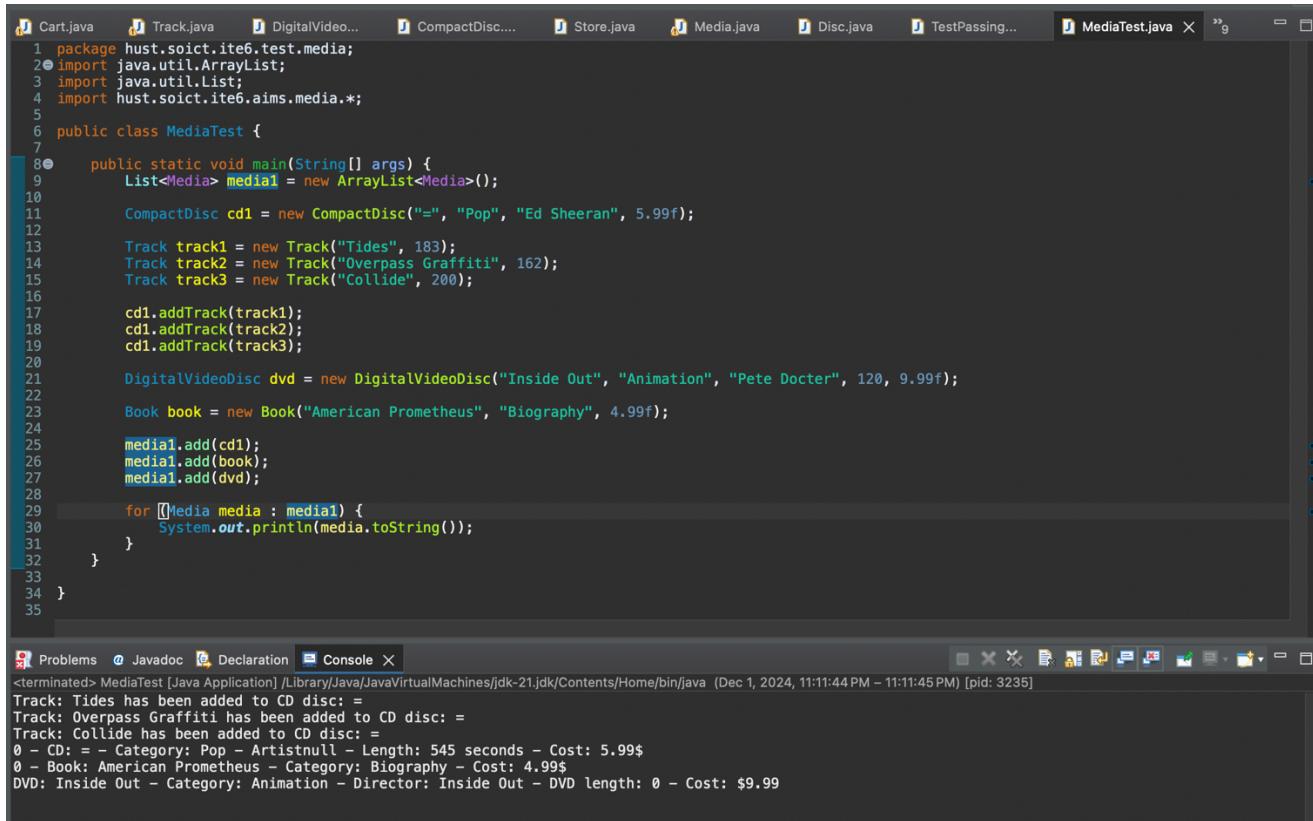
```

1 Cart.java   2 Track.java  3 DigitalVideo...  4 CompactDisc...  5 Store.java  6 Book.java  7 Media.java X
45      return this.getTitle().toLowerCase().contains(title.toLowerCase());
46  }
47
48  @Override
49  public String toString() {
50      return "Media: " + this.getTitle() +
51          " - Category: " + this.getCategory() +
52          " - Cost: " + this.getCost() + "$";
53  }
54
55  @Override
56  public boolean equals(Object obj) {
57      if (obj == this) {
58          return true;
59      }
60      if (!(obj instanceof Media)) {
61          return false;
62      }
63      return ((Media)obj).getTitle() == this.getTitle();
64  }
65
66 }
67

```

Figure 0-29 Override 'equals()' method of 'Media' and 'Track'

## Section 11: Polymorphism with `toString()` method



```

1 package hust.soict.ite6.test.media;
2 import java.util.ArrayList;
3 import java.util.List;
4 import hust.soict.ite6.aims.media.*;
5
6 public class MediaTest {
7
8     public static void main(String[] args) {
9         List<Media> medial = new ArrayList<Media>();
10
11         CompactDisc cd1 = new CompactDisc("=", "Pop", "Ed Sheeran", 5.99f);
12
13         Track track1 = new Track("Tides", 183);
14         Track track2 = new Track("Overpass Graffiti", 162);
15         Track track3 = new Track("Collide", 200);
16
17         cd1.addTrack(track1);
18         cd1.addTrack(track2);
19         cd1.addTrack(track3);
20
21         DigitalVideoDisc dvd = new DigitalVideoDisc("Inside Out", "Animation", "Pete Docter", 120, 9.99f);
22
23         Book book = new Book("American Prometheus", "Biography", 4.99f);
24
25         medial.add(cd1);
26         medial.add(book);
27         medial.add(dvd);
28
29         for (Media media : medial) {
30             System.out.println(media.toString());
31         }
32     }
33 }
34

```

Output in the Console:

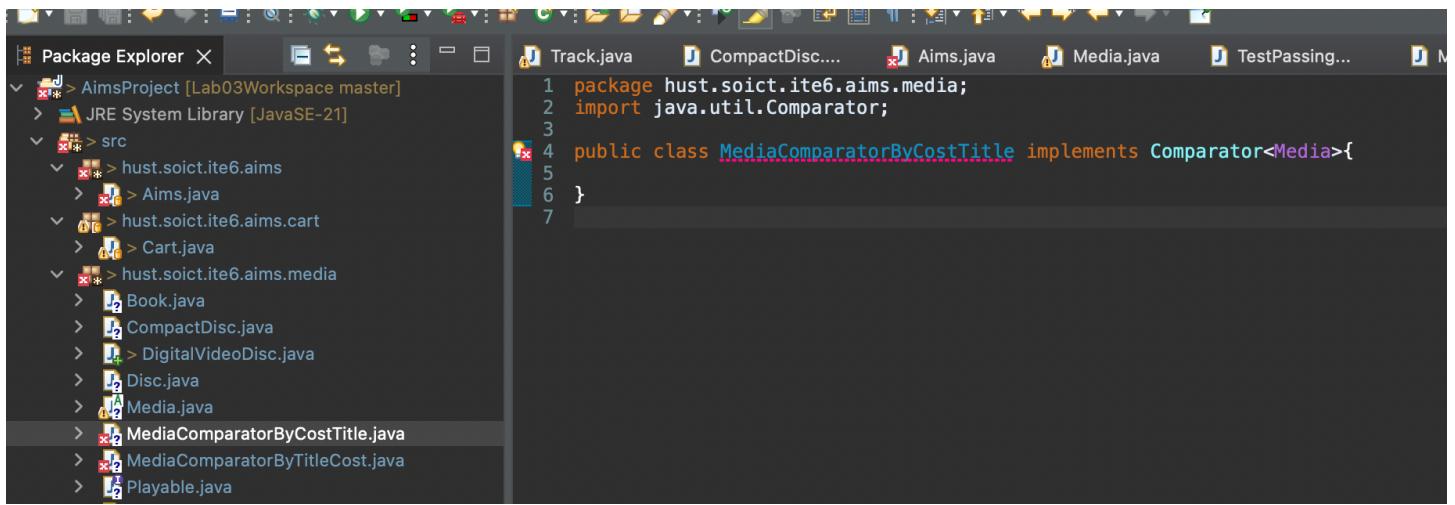
```

Track: Tides has been added to CD disc: =
Track: Overpass Graffiti has been added to CD disc: =
Track: Collide has been added to CD disc: =
0 - CD: = - Category: Pop - Artist: null - Length: 545 seconds - Cost: 5.99$
0 - Book: American Prometheus - Category: Biography - Cost: 4.99$
DVD: Inside Out - Category: Animation - Director: Inside Out - DVD length: 0 - Cost: $9.99

```

Figure 0-30 Create a new test class 'MediaTest' to test the `toString()` method

## Section 12: Sort media in the cart



Project Explorer:

- AimsProject [Lab03Workspace master]
  - JRE System Library [JavaSE-21]
  - src
    - hust.soict.ite6.aims
      - Aims.java
    - hust.soict.ite6.aims.cart
      - Cart.java
    - hust.soict.ite6.aims.media
      - Book.java
      - CompactDisc.java
      - DigitalVideoDisc.java
      - Disc.java
      - Media.java
      - MediaComparatorByCostTitle.java
      - MediaComparatorByTitleCost.java
      - Playable.java

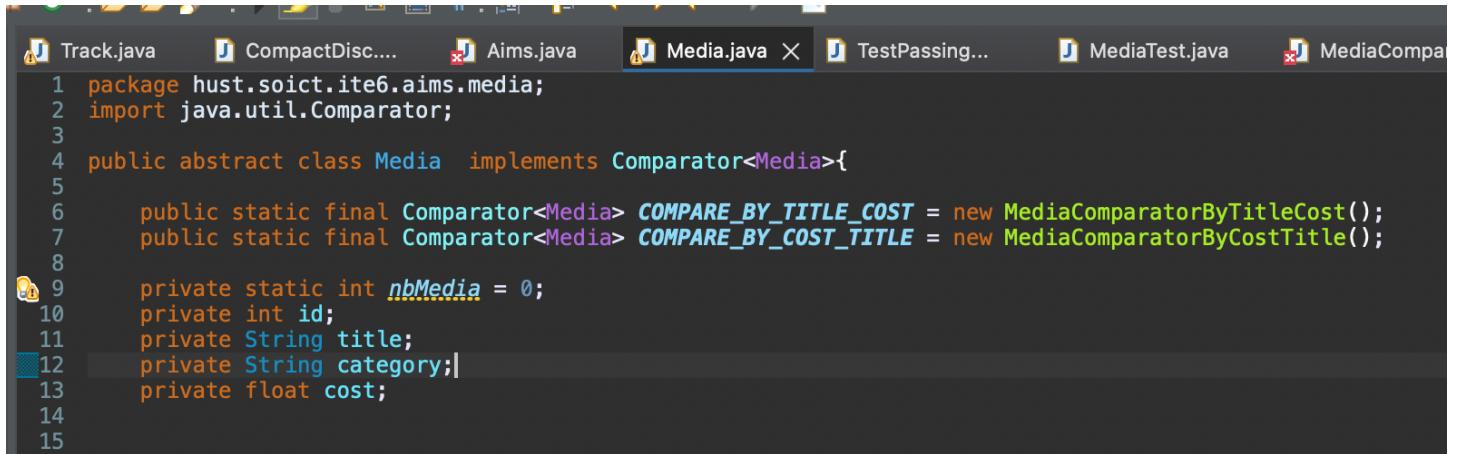
Code in MediaComparatorByCostTitle.java:

```

1 package hust.soict.ite6.aims.media;
2 import java.util.Comparator;
3
4 public class MediaComparatorByCostTitle implements Comparator<Media>{
5
6 }
7

```

Figure 0-31 Create 2 classes of comparators



```

1 package hust.soict.ite6.aims.media;
2 import java.util.Comparator;
3
4 public abstract class Media implements Comparator<Media>{
5
6     public static final Comparator<Media> COMPARE_BY_TITLE_COST = new MediaComparatorByTitleCost();
7     public static final Comparator<Media> COMPARE_BY_COST_TITLE = new MediaComparatorByCostTitle();
8
9     private static int nbMedia = 0;
10    private int id;
11    private String title;
12    private String category;
13    private float cost;
14
15

```

Figure 0-32 Add comparators as attributes in 'Media' class

### **Questions:**

- *What class should implement the Comparable interface?*

*The 'Media' class should implement the 'Comparable' interface.*

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

*The implementation of the 'compareTo()' method can be found in the 'Media' class.*

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

*No, we cannot. The 'Comparable' interface supports only one natural ordering for the objects it compares.*

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

*The compareTo() method in the Disc class can be overridden to implement the specific ordering rule.*

## Section 13: Create a complete console application in the Aims class

For full implementation please check the source code.