

# OOP Lab 004 Report

## 9. Constructors of whole classes and parent classes

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

- **Aggregates:**

- **Store** aggregates Media.
- **Cart** aggregates Media.
- **CompactDisc** aggregates Track.

- **Store Class**

- **Attributes:** Likely contains a collection of Media objects.
- **Constructor:** Initializes the list of Media.
- **Aggregation:** The Store class aggregates Media objects because Media can exist independently of the Store.

- **Cart Class**

- **Attributes:** Likely contains a collection of Media objects.
- **Constructor:** Initializes the list of Media.
- **Aggregation:** The Cart class aggregates Media objects for the same reason as Store.

- **Disc Class**

- **Attributes:** May contain additional details like length and director.
- **Constructor:** Sets properties for Disc, and indirectly via inheritance, initializes Media attributes.
- **Aggregation:** Aggregates no separate objects but inherits from Media.

- **CompactDisc Class**

- **Attributes:** Contains a List<Track> and an artist.
- **Constructor:** Likely initializes the List<Track>.
- **Aggregation:** The CompactDisc aggregates Track because Track instances can exist independently of a CompactDisc.

#### • Track Class

- **Attributes:** Title and length.
- **Constructor:** Initializes these properties.
- **Aggregation:** Not an aggregate class since it contains no other objects.

#### • DigitalVideoDisc Class

- **Attributes:** Inherits Disc attributes and methods.
- **Constructor:** Sets properties specific to DigitalVideoDisc and initializes inherited ones.
- **Aggregation:** None - it directly inherits from Disc.

### 10. If the passing object is not an instance of Media, what happens?

**equals()** would return false if the object supplied to it wasn't an instance of **Media** or **Track**. In the event that type checking is not handled, this prevents **ClassCastException** and guarantees type safety.

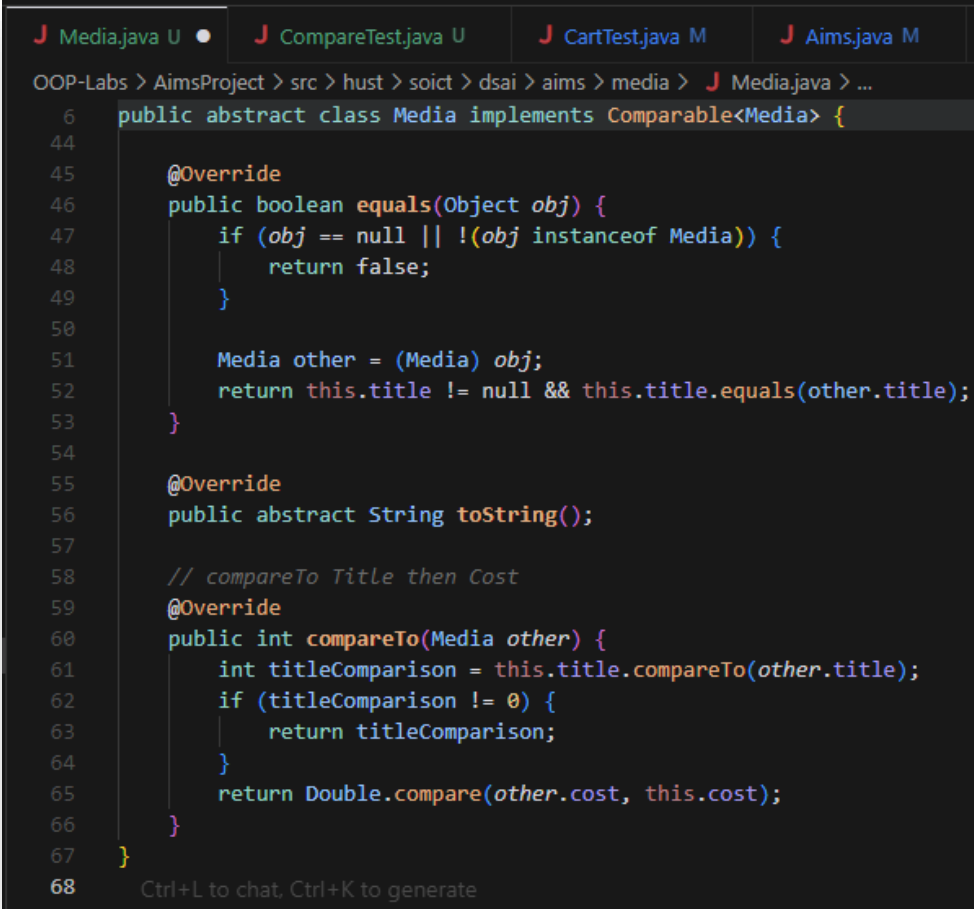
### 12. Sort media in the cart

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

- **What class should implement the Comparable interface?**

The Media class should implement the Comparable interface since we want to define a default ordering for media objects.

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



```
6 public abstract class Media implements Comparable<Media> {
44
45     @Override
46     public boolean equals(Object obj) {
47         if (obj == null || !(obj instanceof Media)) {
48             return false;
49         }
50
51         Media other = (Media) obj;
52         return this.title != null && this.title.equals(other.title);
53     }
54
55     @Override
56     public abstract String toString();
57
58     // compareTo Title then Cost
59     @Override
60     public int compareTo(Media other) {
61         int titleComparison = this.title.compareTo(other.title);
62         if (titleComparison != 0) {
63             return titleComparison;
64         }
65         return Double.compare(other.cost, this.cost);
66     }
67 }
68 Ctrl+L to chat, Ctrl+K to generate
```

If the title is the same, then it would compare by cost (item with higher cost should come first)

- **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, it's really challenging because a class can only have one natural ordering according to the Comparable interface. I believe that Comparator should be used in place of several ordering rules, such as sorting by cost or title.

- **Suppose the DVDs have 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?**

```

OOP-Labs > AimsProject > src > hust > soict > dsai > aims > media > J DigitalVideoDisc.java > DigitalVideoDisc > compareTo(Media)
3 public class DigitalVideoDisc extends Disc implements Playable {
4     @Override
5     public int compareTo(Media other) {
6         if (!(other instanceof DigitalVideoDisc)) {
7             return super.compareTo(other);
8         }
9
10        DigitalVideoDisc otherDVD = (DigitalVideoDisc) other;
11        int titleComparison = this.getTitle().compareTo(otherDVD.getTitle());
12        if (titleComparison != 0) {
13            return titleComparison;
14        }
15        int lengthComparison = Integer.compare(otherDVD.getLength(), this.getLength());
16        if (lengthComparison != 0) {
17            return lengthComparison;
18        }
19        return Double.compare(this.getCost(), otherDVD.getCost());
20    }
21 }
22

```

I believe we can override the `compareTo()` method in the `DigitalVideoDisc` class because DVDs have different ordering criteria than other media types. The altered `compareTo()` function will then compare the DVDs.

## Test result:

```

OOP-Labs > AimsProject > src > hust > soict > dsai > test > compare > J CompareTest.java > ...
1 package hust.soict.dsai.test.compare;
2
3 import hust.soict.dsai.aims.cart.Cart;
4 import hust.soict.dsai.aims.media.Book;
5 import hust.soict.dsai.aims.media.CompactDisc;
6 import hust.soict.dsai.aims.media.DigitalVideoDisc;
7
8 public class CompareTest {
9     public static void main(String[] args) {
10        Cart cart = new Cart();
11
12        CompactDisc cd1 = new CompactDisc(artist:"Taylor Swift", titles:"Album Red", category:"Pop", director:"Taylor Swift", cost:29.99f);
13        CompactDisc cd2 = new CompactDisc(artist:"Blackpink");
14
15        DigitalVideoDisc dvd1 = new DigitalVideoDisc(title:"Inception", category:"Science Fiction", director:"Christopher Nolan", length:148, cost:24.95f);
16        DigitalVideoDisc dvd2 = new DigitalVideoDisc(title:"Avatar", category:"Science Fiction", director:"James Cameron", length:162, cost:29.95f);
17
18        Book b1 = new Book(title:"Harry Potter", category:"Fantasy", cost:45.50f);
19        Book b2 = new Book(title:"The Hobbit", category:"Fantasy Adventure", cost:35.75f);
20        cart.addMedia(cd1);
21        cart.addMedia(cd2);
22        cart.addMedia(dvd1);
23        cart.addMedia(dvd2);
24        cart.addMedia(b1);
25        cart.addMedia(b2);
26
27        // Title, then Cost
28        cart.sortByTitleCost();
29        System.out.println("");
30        // Cost, then Title
31        cart.sortByCostTitle();
32    }
33 }
34

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\bcacb\OneDrive\Documents\OOP_Labs> c:\cd 'c:\Users\bcacb\OneDrive\Documents\OOP_Labs'; & 'C:\Users\bcacb\cursor\extensions\redhat.java-1.37.0-win32-x64\jre\17.0.13-
abs\OOP-Labs\AimsProject\bin' 'hust.soict.dsai.test.compare.CompareTest'
Sorted by Title, then Cost:
Compact Disc-Blackpink--0: 0.0$
Compact Disc-Taylor Swift-Pop-Taylor Swift-0: 29.99$
DVD-Avatar-Science Fiction-James Cameron-162: 29.95$
Book-Harry Potter-Fantasy: 45.5$
DVD-Inception-Science Fiction-Christopher Nolan-148: 24.95$
Book-The Hobbit-Fantasy Adventure: 35.75$

Sorted by Cost, then Title:
Book-Harry Potter-Fantasy: 45.5$
Book-The Hobbit-Fantasy Adventure: 35.75$
Compact Disc-Taylor Swift-Pop-Taylor Swift-0: 29.99$
DVD-Avatar-Science Fiction-James Cameron-162: 29.95$
DVD-Inception-Science Fiction-Christopher Nolan-148: 24.95$
Compact Disc-Blackpink--0: 0.0$
PS C:\Users\bcacb\OneDrive\Documents\OOP_Labs>

```