Student name: Lê Ngọc Anh Vũ

Student ID: 20236014

14. 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?

- Media class is aggregate of Disc, DigitalVideoDisc, CompactDisc and Book class
- Disc class is aggregate of DigitalVideoDisc and CompactDisc class

15. Unique item in a list

- When overriding the `equals()` method of the `Object` class, you will have to cast the `Object` parameter `obj` to the type of `Object` that you are dealing with. For example, in the `Media` class, you must cast the `Object` obj to a `Media`, and then check the equality of the two objects' attributes as the above requirements (i.e. `title` for `Media`; `title` and `length` for `Track`). If the passing object is not an instance of `Media`, what happens?

When we override the equals() method in Java, and we cast the Object parameter to a more specific type (like Media), we should always check whether the passed object is actually an instance of that type before casting. Otherwise, we'll get a ClassCastException at runtime if the object is not of the expected type.

Question: Alternatively, to compare items in the cart, instead of using the `Comparator` class I have mentioned, you can use the `Comparable` interface[1] 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<Media> interface because it is the superclass of all media types (Books, DVDs, CDs) and we want to define default comparison logic for all media objects.

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

If the default ordering is by title (alphabetically), then by cost (descending). We can do vice versa.

- 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, the Comparable interface allows for only one natural ordering, defined by the compareTo() method. If you need multiple ways to sort (e.g., by cost then title), you

should use the Comparator interface instead, which allows for custom, reusable sorting logic.

- 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?

Firstly, we compare titles in alphabetical order. If there are two titles that are the same, then compare their length (descending order). Finally, if their length are also equal, we compare their cost.