LAB 04 reading assignment

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?

1. Aggregates:
   -Store aggregates Media.
   -Cart aggregates Media.
   -CompactDisc aggregates Track.
2. 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.
3. 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.
4. 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.
5. 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.
6. Track Class
   -Attributes: Title and length.
   -Constructor: Initializes these properties.
   -Aggregation: Not an aggregate class since it contains no other objects.
7. 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?

If the object passed to equals() is not an instance of Media or Track, I will false. This ensures type safety and avoids ClassCastException…

12. Sort media in the cart

- **Which class should implement the Comparable interface?**
  The `media` class should implement the `Comparable` interface since it represents the default ordering for media objects.

- **How should the `compareTo()` method be implemented to define the desired ordering?**
  In the `Media` class, the `compareTo()` method needs to be overridden to reflect the intended order of the objects. The specific implementation depends on the attributes used for comparison.

- **Can the Comparable interface support two ordering rules, like sorting by title and cost or cost and title?**
  No, it is challenging because the `Comparable` interface defines only one natural ordering for a class. If multiple sorting criteria are needed, such as sorting by title or cost, using the `Comparator` interface is a better approach.

- **What if DVDs need a different ordering rule compared to other media types?**
  If DVDs require a unique order (e.g., by title, then decreasing length, then cost), the `compareTo()` method can be overridden in the `DigitalVideoDisc` class. This ensures DVDs are compared using their custom ordering logic.