**9. Constructors of whole classes and parent classes**

* Which classes are collections of other classes? Check all constructors of the main classes to see if they initialize their parts.

**Collections:**

* Store contains Media.
* Cart contains Media.
* CompactDisc contains Track.

**Store Class**

* **Attributes:** Likely contains a collection of Media objects.
* **Constructor:** Initializes the list of Media.
* **Aggregation:** The Store class contains Media objects because Media can exist on its own, without the Store.

**Cart Class**

* **Attributes:** Likely contains a collection of Media objects.
* **Constructor:** Initializes the list of Media.
* **Aggregation:** The Cart class contains Media objects, similar to the Store.

**Disc Class**

* **Attributes:** May have extra information like length and director.
* **Constructor:** Sets up properties for Disc, and through inheritance, also initializes Media attributes.
* **Aggregation:** Does not contain any other objects but inherits from Media.

**CompactDisc Class**

* **Attributes:** Contains a list of tracks and an artist.
* **Constructor:** Likely initializes the list of tracks.
* **Aggregation:** The CompactDisc class contains tracks because tracks can exist independently of the CompactDisc.

**Track Class**

* **Attributes:** Title and length.
* **Constructor:** Initializes these properties.
* **Aggregation:** Not a collection class since it doesn’t contain any other objects.

**DigitalVideoDisc Class**

* **Attributes:** Inherits attributes and methods from Disc.
* **Constructor:** Sets specific properties for DigitalVideoDisc and also initializes inherited properties.
* **Aggregation:** None; it directly inherits from Disc.

**10. What happens if the passed object is not an instance of Media?**

* If the object passed to the equals() method is not an instance of Media or Track, it will return false. This makes sure the type is correct and avoids ClassCastException (if type checking isn’t handled properly).

**12. Sorting media in the cart**

**Question:** Instead of using Comparator to compare items in the cart, we can use the Comparable interface and override the compareTo() method. You can check the Java documentation for information on this interface.

**Suppose we use the Comparable interface approach.**

* **Which class should implement the Comparable interface?**
  + The Media class should implement the Comparable interface because we want a default ordering for media objects.
* **How should you implement the compareTo() method in those classes to reflect the ordering we want?**
* **Can we have two ordering rules for items (like by title then cost, and by cost then title) if we use the Comparable interface?**
  + No, it is very hard because the Comparable interface allows only one natural ordering for a class. If we need multiple ordering rules (such as sorting by title or cost), we should use Comparator instead.
  + 
* **If DVDs have a different sorting rule from other media types, which is by title, then decreasing length, then cost, how would you modify the code to support this?**
  + Since DVDs have a different sorting rule from other media types, we can override the compareTo() method in the DigitalVideoDisc class. Then, DVDs will be compared using the overridden compareTo() method.



