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?

- If the object passed to equals() is not an instance of Media or Track, it returns false. This ensures type safety and avoids ClassCastException (in case of not handling type checking).

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.

```java
    public boolean search(String title) {
        return this.title.toLowerCase().contains(title.toLowerCase());
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }

        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }

        Media media = (Media) obj;
        return title.equals(media.title);
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Title: " + title + ", Category: " + category + ", Cost: " + cost;
    }
}
```

- **How should you implement the compareTo() method to reflect the ordering that we want?**
- **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 very hard because the Comparable interface allows only one natural ordering for a class. If we need multiple ordering rules (like sorting by title or cost), we must replace it with Comparator instead.
- **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?**
  - As DVDs have different ordering rules from other media types, we can override the compareTo() method in the DigitalVideoDisc class. After that, DVDs will be compared by the overridden compareTo() method.