Part 11's questions: Iterate through the list and print out the information of the media by using `toString()` method. Observe what happens and explain in detail.

```
public class PolymorphismTest {
	public static void main(String[] args) {
		ArrayList<Media> mediaList = new ArrayList<Media>();

		ArrayList<String> authors = new ArrayList<String>();
		authors.add("Sarah Rose Summer");
		authors.add("Catriona Gray");
		Book b = new Book("Snow White", "Comic book", 50.99, authors);

		ArrayList<Track> tracks = new ArrayList<Track>();
		Track track1 = new Track("Let it go", 30);
		Track track2 = new Track("Hero", 70);
		tracks.add(track1);
		tracks.add(track2);
		CompactDisc cd = new CompactDisc("Frozen", "Animation", 30.95, 200, "Elsa",
"Diana Mendez", tracks);

	   DigitalVideoDisc dvd = new DigitalVideoDisc("Snow white", "Animation", 40.15,
150, "Anna");

		mediaList.add(dvd);
		mediaList.add(cd);
		mediaList.add(b);

		for(Media m : mediaList) {
			System.out.println(m.toString());
		}
	}
}
```

Result in cosole application:

```
DVD Title: Snow white - Category: Animation - Cost: 40.15$ - Length: 150 - Director:
Anna
CD Title: Frozen - Category: Animation - Cost: 30.95$ - Length: 200 - Director: Elsa -
Artist: Diana Mendez - Track: [(Title: Let it go , Length: 30), (Title: Hero , Length:
70)]
Book Title: Snow White - Category: Comic book - Cost: 50.99$ - Authors: [Sarah Rose
Summer, Catriona Gray]
```

Some explanations:

Polymorphism in behavior: The `mediaList` holds objects of different types (`Book`, `CompactDisc`, `DigitalVideoDisc`), but each object calls its own overridden `toString()` method when printed.

Part 12's questions: 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.

1. What class should implement the `Comparable` interface?
   It is used when an object of a class needs to compare itself with other objects of the same class. This means that the `Media` class should implement the `Comparable` interface. By implementing this interface, instances of the class will be able to be compared with each other using the `compareTo()` method.
2. In those classes, how should you implement the `compareTo()` method be to reflect the ordering that we want?
   For example, if we want to sort the media in the list by title first, then by cost, we can implement the `compareTo()` method in the `Media` class like this one:

```
@Override
public int compareTo(Media other) {
     return this.title.compareTo(other.title);
     // Only one comparison rule (by title)
}
```

3. 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, if you use the `Comparable` interface approach, you can only define **one natural ordering** for the objects of that class. The `Comparable` interface allows you to define a single comparison method, `compareTo()`, which specifies how two objects of the class should be compared with each other based on one criterion (e.g., by `title` or `cost`).
   If you want to have multiple ordering rules (e.g., by `title` then `cost`, or by `cost` then `title`), the `Comparable` interface alone won't be sufficient, because it only allows one comparison logic to be defined.
4. Suppose the DVDs has 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?
   import java.util.*;

```
public class MediaComparatorByTitleLengthCost implements Comparator<Media> {
    @Override
    public int compare(Media m1, Media m2) {
        if (m1 instanceof DigitalVideoDisc && m2 instanceof DigitalVideoDisc) {
            DigitalVideoDisc dvd1 = (DigitalVideoDisc) m1;
            DigitalVideoDisc dvd2 = (DigitalVideoDisc) m2;

            int titleComparison = dvd1.getTitle().compareTo(dvd2.getTitle());
            if (titleComparison != 0) {
                return titleComparison;
            }

            int lengthComparison = Integer.compare(dvd2.getLength(),
                                                   dvd1.getLength());
            if (lengthComparison != 0) {
                return lengthComparison;
            }

            return Double.compare(dvd1.getCost(), dvd2.getCost());
        }
        return 0;
    }
}
```