Media, sau đó kiểm tra sự bình đẳng của các thuộc tính của hai đối tượng như các yêu cầu trên (tức là `tiêu đề` cho `Media`; `tiêu đề` và `độ dài` cho `Track`). Nếu đối tượng đi qua không phải là một thực thể của `Media`, điều gì sẽ xảy ra?

Nếu so sánh một đối tượng của lớp Media với một đối tượng của lớp Track, phương thức equals() trong Track cần đảm bảo:

- Kiểm tra xem obj có thuộc cùng hệ thống phân cấp (Media) không.
- Đảm bảo loại cụ thể của obj (ví dụ: Track) để so sánh các thuộc tính một cách phù hợp.

Nếu không làm như vậy, kết quả của equals() có thể không chính xác hoặc dẫn đến lỗi runtime khi ép kiểu không hợp lệ.

- Lặp lại danh sách và in ra thông tin của phương tiện bằng phương thức toString(). Quan sát những gì xảy ra và giải thích chi tiết.

Trong bài làm toString() ⇔print()

```
// hàm print
public void print() {
    float totalCost = 0.00f;
    StringBuilder output = new StringBuilder("**********CART**********\nOrderItems\n");
    for (Media i: itemsOrdered) {
        output.append(i.getId() + " - " +i.getTitle() + " - " + i.getCategory() +" : "+ i.getCost() + " $\n");
        totalCost = +i.getCost();
    }
    output.append("Total cost: " + totalCost + " $")
            .append("\n**************************************************");
    System.out.println(output);
}
```

```java
public class Cart {

    private ArrayList<Media> itemsOrdered = new ArrayList<Media>();
    //addMedia
    public boolean equals(Media media) {
        for (Media i : itemsOrdered) {
            if ((i.getTitle()).equals(media.getTitle()))
                return true;
        }
        return false;
    }
    public void addMedia(Media media) {
        if(!equals(media))
        itemsOrdered.add(media);
        System.out.println("Added: " + media.getTitle());
    }
//removeMedia
    public int removeMedia(Media media) {
        if (itemsOrdered.remove(media)) {
            System.out.println("Removed: " + media.getTitle());
            return 1;
        } else {
            System.out.println("Media not found in the cart.");
            return 0;
        }
    }
```

```java
    public static void main(String[] args) {
    //Create a new cart
    Cart cart = new Cart();
    //Create new dvd objects and add them to the cart
    DigitalVideoDisc dvd1 = new DigitalVideoDisc ("The Lion King", "Animation", "Roger Allers", 87, 19.95f); cart.addMedia(dvd1);
    DigitalVideoDisc dvd2 = new DigitalVideoDisc ("Star Wars", "Science Fiction", "George Lucas", 87, 24.95f); cart.addMedia(dvd2);
    DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin", "Animation", 18.99f); cart.addMedia(dvd3);
    Book book1=new Book("Banh troi nuoc","ho xuan huong");cart.addMedia(book1);
    Book book2=new Book("Doraemon","anime",10.00f);cart.addMedia(book2);
    CompactDisc track1= new CompactDisc("Power");cart.addMedia(track1);
    //Test the print method
    cart.print();
    //To-do: Test the search methods here
    cart.search(2);
    cart.search("Aladin");
    cart.search(1);
    cart.search("Banh troi nuoc");
    dvd1.play();
    }
    }
```

```
OrderItems
1 - The Lion King - Animation : 19.95 $
2 - Star Wars - Science Fiction : 24.95 $
3 - Aladin - Animation : 18.99 $
1 - Banh troi nuoc - null : 0.0 $
2 - Doraemon - anime : 10.0 $
1 - Power - null : 0.0 $
Total cost: 0.0 $
**********************************************
```

Câu hỏi 1: What class should implement the Comparable interface?

- Class Media nên implement interface Comparable nếu muốn cung cấp một thứ tự sắp xếp mặc định cho các đối tượng Media.
- Lý do: Interface Comparable được sử dụng để định nghĩa thứ tự tự nhiên (natural ordering) của một class. Các lớp khác như DVD có thể kế thừa từ Media và sử dụng hoặc ghi đè phương thức sắp xếp này.

Câu hỏi 2: In those classes, how should you implement the compareTo() method to reflect the ordering that we want?

- Thứ tự mặc định (natural ordering) sẽ được định nghĩa bằng cách override phương thức compareTo() trong class Media.

Ví dụ, nếu muốn sắp xếp mặc định theo **tiêu đề tăng dần**, và nếu tiêu đề giống nhau thì sắp xếp theo **giá giảm dần**:

```
@Override
public int compareTo(Media other) {
    int titleComparison = this.title.compareTo(other.title); // Tiêu đề tăng dần
    if (titleComparison != 0) {
        return titleComparison;
    }
    return Double.compare(other.cost, this.cost); // Giá giảm dần
}
```

Câu hỏi 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?

- **Không thể.**
    - Interface Comparable chỉ cho phép một thứ tự sắp xếp mặc định (một phương thức compareTo()).
    - Nếu cần nhiều cách sắp xếp (ví dụ: byTitleThenCost và byCostThenTitle), **phải sử dụng các lớp Comparator riêng biệt.**

Câu hỏi 4: 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?

- **Giải pháp:** Tạo một lớp con DVD kế thừa từ Media và sử dụng một Comparator riêng biệt để xử lý thứ tự sắp xếp.

Ví dụ:

```
public class DVD extends Media {
    public DVD(String title, double cost) {
        super(title, cost);
    }

    public static final Comparator<DVD> COMPARE_BY_TITLE_LENGTH_COST =
new Comparator<DVD>() {
```

```
        @Override
        public int compare(DVD d1, DVD d2) {
            // Sắp xếp theo tiêu đề (tăng dần)
            int titleComparison = d1.getTitle().compareTo(d2.getTitle());
            if (titleComparison != 0) {
                return titleComparison;
            }

            // Nếu tiêu đề giống nhau, sắp xếp theo độ dài tiêu đề (giảm dần)
            int lengthComparison = Integer.compare(d2.getTitle().length(),
d1.getTitle().length());
            if (lengthComparison != 0) {
                return lengthComparison;
            }

            // Nếu độ dài tiêu đề giống nhau, sắp xếp theo giá (tăng dần)
            return Double.compare(d1.getCost(), d2.getCost());
        }
    };
}
```