Reading Assignment

16. Polymorphism with toString() method

1. What are the advantages of Polymorphism?
   - Giúp code linh hoạt và dễ mở rộng
   - Dễ bảo trì: không cần biết kiểu cụ thể của đối tượng
   - Cho phép xử lý các đối tượng khác nhau theo cùng một cách, dựa vào cùng một interface hoặc lớp cha

2. How is Inheritance useful to achieve Polymorphism in Java?
   - Inheritance (kế thừa) là điều kiện cần để tạo nên đa hình
   - Các lớp con kế thừa từ lớp cha và ghi đè phương thức → cho phép gọi phương thức qua kiểu lớp cha nhưng hành vi của lớp con được thực thi

3. What are the differences between Polymorphism and Inheritance in Java?

Polymorphism	Inheritance
Là hành vi: nhiều hình thức thực thi	Là mối quan hệ: lớp con kế thừa lớp cha
Cho phép xử lý đối tượng qua kiểu cha	Cho phép chia sẻ lại code(reusability)
Dựa trên override, interface	Dựa trên từ extends

17. Sort media in the cart

1. What class should implement the Comparable interface?
   Lớp Media nên implement Comparable<Media>, vì nó là lớp cha chung cho các loại media như Book, DVD, CD, và là kiểu được lưu trong danh sách ArrayList<Media>.

2. In those classes, how should you implement the compareTo()method to reflect the ordering that we want:
   + Nếu muốn **sắp xếp mặc định theo tiêu đề (title) rồi theo giá (cost):**

```java
@Override
public int compareTo(Media other) {
    int titleCompare = this.title.compareToIgnoreCase(other.title);
    if (titleCompare != 0) return titleCompare;
    return Float.compare(this.cost, other.cost); // cost tăng dần
}
```

+ Nếu muốn sắp xếp mặc định theo giá giảm dần rồi đến tiêu đề, chỉ cần đảo chiều như sau:

```java
int costCompare = Float.compare(other.cost, this.cost);
if (costCompare != 0)
    return costCompare;
return this.title.compareToIgnoreCase(other.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 ?
   + Không . Vì Comparable chỉ cho phép một cách sắp xếp mặc định duy nhất (qua compareTo()).
   + Muốn linh hoạt sắp xếp theo nhiều cách (ví dụ: theo cost, hoặc theo title), phải dùng Comparator.

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
   + Cách 1: Tạo một Comparator riêng cho DVD:
   public class DVDComparatorByTitleLengthCost implements Comparator<DigitalVideoDisc>

   + Cách 2 : Trong DigitalVideoDisc, override compareTo() khác với Media
   Nhưng lại gặp một vấn đề ở đây là :

```java
public class Media implements Comparable<Media> {
    public int compareTo(Media o) {
        return this.title.compareToIgnoreCase(o.title); // mac dinh sort theo title
    }
}
```

```java
public class DigitalVideoDisc extends Media {
    @Override
    public int compareTo(Media o) {
        // sap xep theo: title -> length giam -> cost
    }
}
```

Nếu Media dùng compareTo() theo title,
→ Mà DigitalVideoDisc dùng title → length → cost
→ Hành vi của compareTo() không còn nhất quán

Garbage :

1. So sánh thời gian chạy :

```
Time with String + : 700ms
Time with StringBuilder: 3ms
Time with StringBuffer: 3ms
```

2. So sánh và đánh giá :

Cách nối chuỗi	Thời gian chạy	Đánh giá thực tế
String+	700ms	Chậm do tạo nhiều đối tượng rác
StringBuilder	3ms	Rất nhanh và tối ưu
StringBuffer	3ms	Nhanh gần bằng StringBuilder, có thread-safe