



# Lab 8 Polymorphism

---

Lý thuyết và ngôn ngữ hướng đối tượng  
(bài tập)



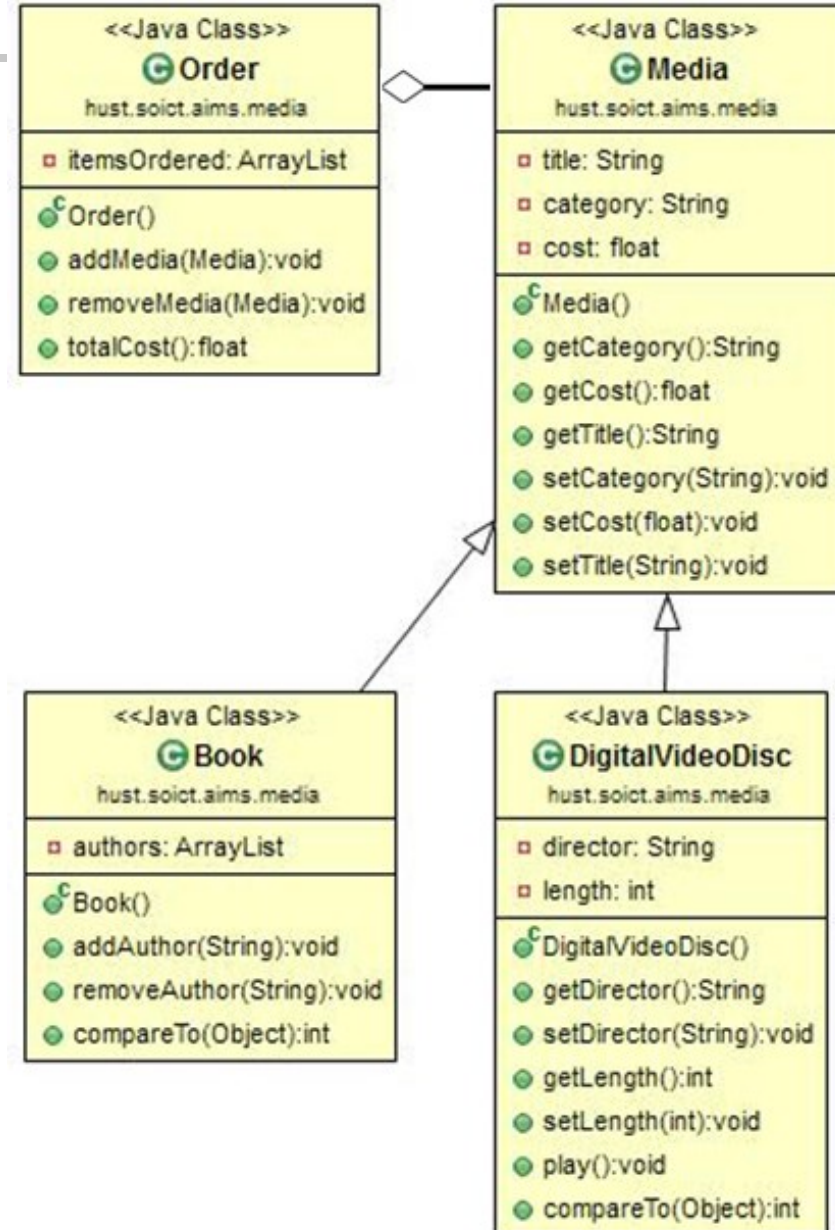
# Lab's Objectives

---

- In this lab, you will practice with:
  - Polymorphism
  - Comparable interface for the purpose of sorting objects within a collection
  - Template
  - Map

# 1. Continue the AimsProject

- Open Eclipse
- Open AimsProject





## 2. Implement the `Comparable` interface and sort

---

- To sort media products, implement the `Comparable` interface on `CompactDisc`, `DigitalVideoDisc`, `Track`, and `Book`.

```
public interface Comparable<T> {  
    public int compareTo(T o);  
}
```





## 2. Implement the `Comparable` interface and sort

---

- For each of `CompactDisc`, `DigitalVideoDisc`, `Track`, and `Book`, edit the class description to include the `Comparable` interface after the `implements` keyword in the class declaration.
- For example, the class declaration for `DigitalVideoDisc` becomes:

```
public class DigitalVideoDisc  
    extends Media implements Playable, Comparable {
```



### 3. Implement the compareTo() method

---

- For each of `CompactDisc`, `DigitalVideoDisc`, `Track`, and `Book`, create a method called `compareTo()` with the signature:

```
public int compareTo(Object obj)
```

- You will have to cast the `Object` parameter `obj` to the type of `Object` that you are dealing with.
- For example, in the `DigitalVideoDisc` class, you must cast the `Object obj` to a `DigitalVideoDisc`, and then return the value of a `compareTo()` on the title fields of the two objects.
- If the passing object is not an instance of `DigitalVideoDisc`, what happens?



## 4. Test the compareTo() method

---

- Open `Aims.java` in the editor.
- Locate the `main()` method. At the end of the `main()` method, create a collection (for example, a `ArrayList`) and add some `Media` objects to it (for example, some `DigitalVideoDiscs` as the sample code below). Write more codes for other `Media` types.
- Iterate through the entries in the collection and output them.
- Then use the `Collection.sort()` method to sort the entries in the collection, and output them again. They should now be in sorted order based on the `compareTo()` method that you created for that class.



## 4. Test the compareTo() method

---

```
java.util.Collection collection = new
java.util.ArrayList();

// Add the DVD objects to the ArrayList
collection.add(dvd2);
collection.add(dvd1);
collection.add(dvd3);

// Iterate through the ArrayList and output their titles
// (unsorted order)
java.util.Iterator iterator = collection.iterator();

System.out.println("-----");
System.out.println ("The DVDs currently in the order are: ");

while (iterator.hasNext()) {
    System.out.println
        (((DigitalVideoDisc) iterator.next()).getTitle());
}
```





## 4. Test the compareTo() method

---

```
// Sort the collection of DVDs - based on the compareTo()  
// method  
java.util.Collections.sort((java.util.List)collection);  
  
// Iterate through the ArrayList and output their titles -  
// in sorted order  
iterator = collection.iterator();  
  
System.out.println("-----");  
System.out.println ("The DVDs in sorted order are: ");  
  
while (iterator.hasNext()) {  
    System.out.println  
        ((DigitalVideoDisc) iterator.next()).getTitle();  
}  
  
System.out.println("-----");
```



## 4. Test the compareTo() method

- Execute your program (click the Run button as before).
- The output in the Console view may like the following (depends on objects you've created):

```
Playing DVD: The Lion King
DVD length: 87
Playing DVD: Star Wars
DVD length: 124
Playing DVD: Aladdin
DVD length: 90
The total length of the CD to add is: 13
Playing CD: IBM Symphony
CD length:13
Playing DVD: Warmup
DVD length: 3
Playing DVD: Scales
DVD length: 4
Playing DVD: Introduction
DVD length: 6
Total Cost is: 163.83
-----
The DVDs currently in the order are:
Star Wars
The Lion King
Aladdin
-----
The DVDs in sorted order are:
Aladdin
Star Wars
The Lion King
-----
```



## 5. Change the criteria for sorting media

---

- Suppose you wish to sort your DVDs by cost or length, rather than title.
  - What changes would you need to make to your code to do this?
  - Try making the necessary changes to the `compareTo()` method so that your DVDs are sorted by cost, from lowest to highest cost.
- Do the modification if you want to sort CDs by number of tracks then by length, rather than title. If two CDs have the same number of tracks, please compare lengths of these two CDs.
- Modify and run the class the `Aims` class to see the changes.



## 6. Using template for Collection

---

- Please modify all codes which work with `Collection` (from previous labs to this lab) to template style, for example:

```
Collection collection = new ArrayList();
```

- should be modified to:

```
List<CompactDisc> discs = new  
ArrayList<CompactDisc>();
```

- Or in the `Order` class should have:

```
List<Media> itemsOrdered = new  
ArrayList<Media>();
```

- and all related source codes. Run and test again.



## 7. Counting the frequency of Book content with Map

---

- Add an attribute `String content` for `Book` with two additional attributes:
  - A sorted list `List<String> contentTokens`
  - A sorted map `Map<String,Integer> wordFrequency`
- Write a method `processContent()` calculating the following information of the book content. This method is called when the content of the book is set/changed.
  - Split the content to tokens by spaces or punctuations, then sort these tokens from a  $\rightarrow$  z and set to the `contentTokens` attribute list
  - Count the frequency of each token, sort by token from a  $\rightarrow$  z and set to the `wordFrequency` attribute map



## 7. Counting the frequency of Book content with Map

---

- Override the method `toString()` to return all information of `Book`: all values of `Book` attributes, the content length (i.e. the number of tokens), the token list and the word frequency of the content.
- Write the `BookTest` class to test all above methods and display information of `Book`.