Object-Oriented Language and Theory

Lecturer: Nguyen Thanh Hung, [hungnt@soict.hust.edu.vn](mailto:trangntt@soict.hust.edu.vn)

Teaching Assistant: Nguyen Trung Thanh (thanh.nt176874@sis.hust.edu.vn)

**Lab 8: Polymorphism**

**\* Objectives:**

In this lab, you will practice with:

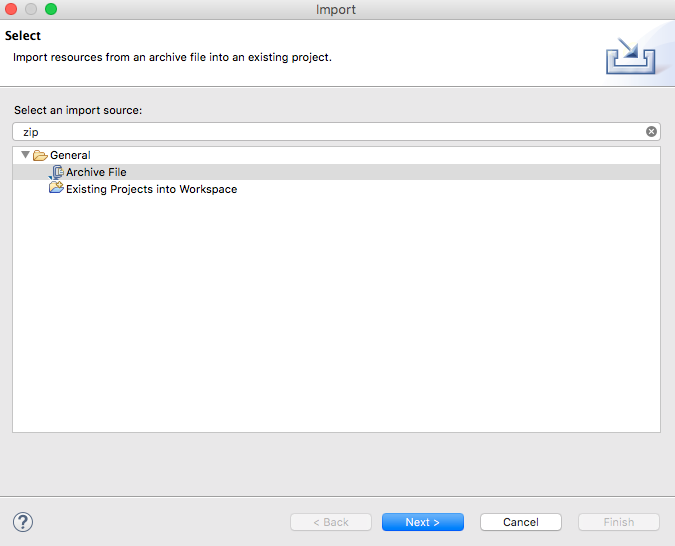
* Polymorphism
* Comparable interface for the purpose of sorting objects within a collection
* Template
* Map

You need to use the project that you did with the previous labs including AimsProject.

**1. Import the AimsProject**

**- Open Eclipse**

- Open File 🡪 Import. Type zip to find Archive File if you have exported as a zip file before. You may choose Existing Projects into Workspace if you want to open an existing project in your computer. Ignore this step if the AimsProject is already opened in the workspace.

****

- Click Next and Browse to a zip file or a project to open.

**2. Override** equals() **method of the Object class**

We can apply Release Flow here by creating a topic branch for the override of equals() method.

- In previous labs, you add a media to an order or a track to a CD. However, you may use the contains() methods of the list in the order or the CD to ensure that a similar object is not added to that list.

- However, the contains() method returns true if the list contains the specific element. More formally, it returns true if and only if the list contains at least one element e such that e such that (o==null?e==null:o.equals(e)). So the contains() method actually use equals() method to check equality.

- Please override the boolean equals(Object o) of the Media and the Track class so that two objects of these classes can be considered as equal if:

+ For the Media class: the id is equal

+ For the Track class: the title and the length are equal

- When overriding the equals() method of the Object class, you will have to cast the Object parameter obj to the type of Object that you are dealing with. For example, in the Media class, you must cast the Object obj to a Media, and then check the equality of the two objects’ attributes as the above requirements (i.e. id for Media; title and length for Track). If the passing object is not an instance of Media, what happens?

**3. Implement the** Comparable **interface and sort**

- We can apply Release Flow here by creating a topic branch for Comparable interface implementation.

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

***Note:*** The Comparable interface is part of the Java class library. It is in the java.lang package, so no import statement is needed. Please open the Java docs to see the information of this interface. Which method(s) do you need to implement from this interface?

+ 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:

Note: When you save your classes, you will now receive an error in the Tasks view. This is because classes which implement the Comparable interface must implement the compareTo method (from Comparable) - and this has not yet been completed.

**4. 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)

- When implementing the compareTo() method of the Comparable interface, you often simply use the compareTo() method on one of the fields of the class. 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?

- Since title is a field of Media and all of the above Comparable classes extend Media, you can create the same compareTo() methods for the CompactDisc, Track, and Book. Hence, you should think about the fact that we can move the compareTo() method to the Media class for re-use, and only Media class needs to implements the Comparable interface. If you do that, you can freely remove all implementations of the compareTo() method for the Comparable interface from the CD, DVD and Book classes.

**5. Test the compareTo() method**

- In **hust.soict.globalict.test.media** or **hust.soict.hedspi.test.media** package, create a new class for testing, e.g., TestMediaCompareTo.

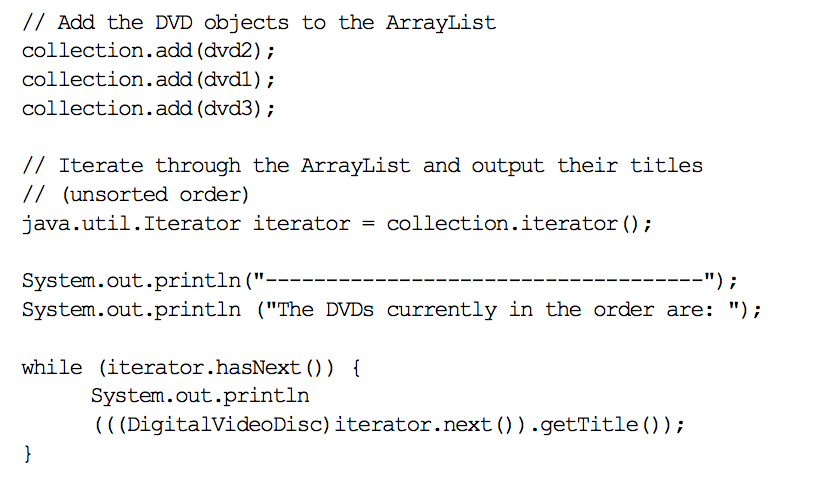
- Create a collection (for example, a ArrayList) and add some Media objects to it (for example, some DigitalVideoDiscs as the sample code below). Write more codes for other Media types.

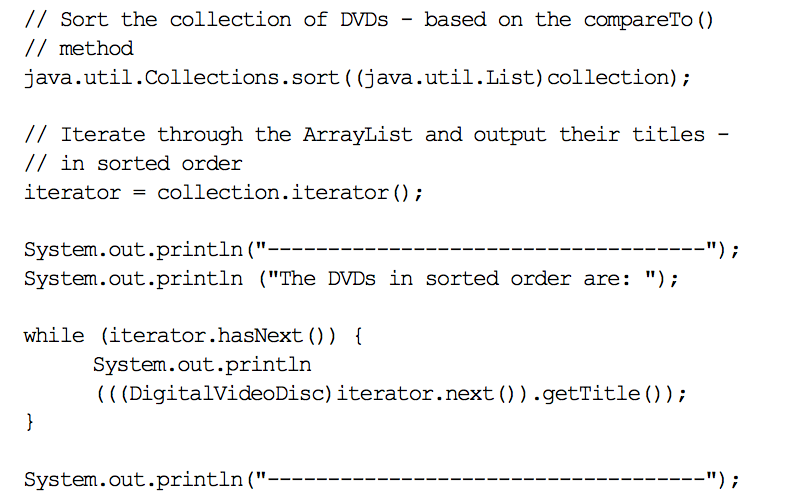
- Iterate through the entries in the collection and output them.

- 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.

- Below is only the suggestion, you need to complete it for a good menu presenting to the users:

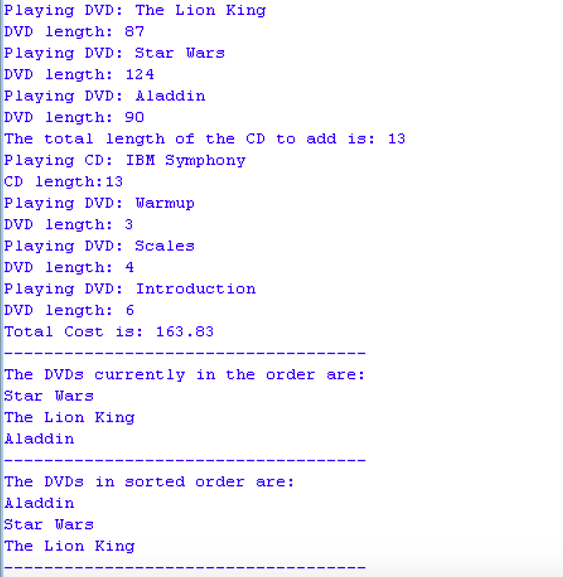
java.util.Collection collection = new java.util.ArrayList();





- 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):



**6. Change the criteria for sorting media**

We can apply Release Flow here by creating a topic branch for the addition of Media sorting criteria.

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.

**7. Using template for** Collection

We can apply Release Flow here by creating a topic branch for template usage.

Please modify all codes which work with Collection 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. Some suggestions can be found as below for the Media class, and similar ways for others.

public class Media implements Comparable<T> {

...

public int compareTo(T o){

...

}

…

}

Run and test again.

**8. Counting the frequency of** Book **content with** Map

- We can apply Release Flow here by creating a feature branch for book content processing.

- Add an attribute String content for Book with two additional attributes:

+ A sorted list List<String> contentTokens: *Hint: you may use String.split(String separator)to split the content by the separator, where the separator can be a regex (regular expression).*

+ 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 🡪 z and set to the contentTokens attribute list

+ Count the frequency of each token, sort by token from a 🡪 z and set to the wordFrequency attribute map *(Hint: use TreeMap to get an ordered 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 in a test package to test all above methods and display information of Book.