CSci 1933 Lab 6

October 20, 2015

1. Introduction

The purpose of this lab is to create classes to illustrate basic object-oriented programming techniques, including inheritance and polymorphism, and to implement the Comparable interface.

2. Your task

In today's lab, you will implement classes necessary to make a rudimentary media library. You will implement an abstract class RateableMedia in steps 2.4-2.6. The most important step is to override the compareTo() method so that the subclasses Movie and Music can be sorted based on their rating (e.g., a Movie rated 5 stars should come before a Movie rated 4 stars); we provide this code in step 5. In Step 7 you will implement reading in and processing Media objects from a file.

To complete this lab, do the following:

2.1. In IntelliJ, create a new Java project named Lab 6.

2.2. Import the following files into our Lab 6 project.

- Media.java The abstract class you will extend. Place this in the src/ directory.
- Picture.java An object to represent a picture in the Media Library. Place this in the src/ directory.
- Music.java An object to represent a song in the Media Library. Place this in the src/ directory.
- Movie.java An object to represent a movie in the Media Library. Place this in the src/ directory.
- MediaReader.java A class to read in entries from a text file and output Media objects. Place this in the src/ directory.
- MediaTest.java A set of JUnit test for this lab. Place this in the src/ directory.
- Media.txt The Media Library.

2.3. Add JUnit to your project. To do this, do the following:

- Select “Project structure” from the “File” menu.
- Go to the "Libraries" group, click the little green plus (look up), and choose "From Maven...".

- Search for "junit" -- you're looking for something like "junit:junit:4.8.1" (latest stable option), and click OK.

2.4. Create the abstract class RateableMedia.

- Make RateableMedia extend the class Media.
- Define an instance variable of type int to store the rating; name this instance variable rating. It will need to be declared protected instead of private for the child classes to access it.
- Define a public accessor method getRating() to return the rating.
- Override the compareTo method so that RateableMedia objects will be ordered by their ratings, not by their names. Please see the following logic of the compareTo method
    - If the input Media is RateableMedia…
        - ...with a rating greater than that of the instance of RateableMedia on which this method is operating, return what you are supposed to do when the input media is greater than the current instance.
        - ...with a rating less than that of the instance of RateableMedia on which this method is operating, do the opposite
        - ...with an equal rating, do sorting behavior according to the name of the media (getName())
    - Else
        - do sorting behavior according to the name of the media

> **Note**
>
> 1. You can take advantage of the IntelliJ Generating Code function auto-implement the definition of the override methods. https://www.jetbrains.com/idea/help/generating-code.html
> 2. For more information on the compareTo() method, please see http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html

2.5. Run the unit tests in MediaTest.java to ensure that you implemented these methods correctly.

2.6. Create a class called Library with a main() method.

- Import the Collection interface.

> **Hint**
>
> You can do “import” by:
> 1) moving the cursor to the red-highlighted “Collection” keyword and using the keyboard shortcut “Intention Actions” https://www.jetbrains.com/idea/help/keyboard-shortcuts-you-cannot-miss.html. IntelliJ will prompts you the class to import.

> 2) manually adding a statement to import java.util.Collections to the top of the file, outside of the class declaration.

- Create a MediaReader object to read from the file named Media.txt. The constructor of MediaReader takes in a String as an argument, just like TweetReader.
- Create two ArrayLists. One will contain all the elements read in using the MediaReader object. The second will contain only those items that are a subclass of RateableMedia, i.e., objects of type Music and Movies. Make sure to use the above hint to import java.util.ArrayList to the top of the file outside of the class declaration.
- Read in the media entries using MediaReader's getMedia() method. Add each entry to the appropriate list or lists. The code you need for this task will have a similar structure to the code you wrote for Lab 3:

```
while advance() returns true
    process a record: that is, add the media entry to the list of all
    media objects and add it to the list of RateableMedia if
    it is an instanceOf RateableMedia.
```

- After adding all the items to the lists, sort the RateableMedia list using the Collections.sort() static method.
- Print out each list using a single static method. Sample output is given below. Note that the toString() method of each class returns precisely the output you need to produce for each line of output.

```
All Media
Movie: Hackers by Iain Softley is rated 5 stars
Music: Lights by Ellie Goulding is rated 5 stars
Movie: Contagion by Stephen Soderbergh is rated 4 stars
Music: Finally Found by Late Night Alumni is rated 4 stars
Picture: Mona Lisa
Music: Stairway to Heven by Led Zepplein is rated 5 stars

RateableMedia
Movie: Hackers by Iain Softley is rated 5 stars
Music: Lights by Ellie Goulding is rated 5 stars
Music: Stairway to Heven by Led Zepplein is rated 5 stars
Movie: Contagion by Stephen Soderbergh is rated 4 stars
Music: Finally Found by Late Night Alumni is rated 4 stars
```

2.7. The following part is intended to get you thinking about some of the topics we'll be discussing in class over the next few weeks and to help you prepare for upcoming homework. The following part is intentionally a bit hard for you and we're intentionally not holding your hand. You should do what you think you need to do to get this working.

- Import the following files into your Lab6 project in the src/ directory.
    - SortedLinkedList.java
    - SortedArrayList.java
- Open SortedLinkedList.java. Change the method signature for getItem(int index) so that it returns an int equal to the number of comparisons needed to find the element at index.
- Open SortedArrayList.java. Change the method signature for getItem(int index) so that it returns an int equal to the number of comparisons needed to find the element at index.
- Add code to the end of your main() method in Library.java to create a SortedArrayList and SortedLinkedList.
- Using the getRandom() method from MediaReader to add the same 1,000 random elements to both the SortedArrayList and SortedLinkedList.
- Using getItem() from each of the lists print out the number of comparisons needed to find the last (1,000th) element of the two lists. Try this with different numbers of elements in the two lists (5000, 10000, etc.) and look at how the number of comparisons changes with the number of elements in the list.

---END---

---Solutions will be posted on moodle page 30 minutes before the lab completion time---