CIS*2430 (Fall 2010) Assignment Three

Instructor: F. Song

Due Time: November 15, 2010 by midnight.

In Assignment Two, you created three different classes (Book, Journal, and LibrarySearch) and were able to add and search for the relevant references. In Assignment Three, you will simplify your code and add some improvements. In particular, you will create a super class for all books and journals and reduce the two ArrayLists to one in order to minimize the code redundancy. Additionally, you will load the existing references from a file and save all references (existing plus new) to another file every time you run your program. Finally, you will create a HashMap for the title keywords so that the search performance can be greatly improved.

Specific Requirements for Assignment Three

(1) Create a super class *Reference* that contains *Book* and *Journal* as its subclasses. Note that all the common members of the sub-classes should be pushed into the corresponding parent class so that they can be inherited by the sub-classes. Any new additions and overriding methods should be defined in a subclass. In particular, you should revise your "equals" methods to make them truly overriding for objects. All of this will help simplify the two concrete classes of Book and Journal considerably due to the code reuse of inheritance.

(2) Reduce two ArrayLists to one ArrayList so that you can keep all the references in one place. The new list should have the type "ArrayList<Reference>" and will help simplify your code for adding and searching relevant references considerably.

(3) Expand your code so that you can load the existing references from a file at the start and save all the references (including the existing and new references) to another file at the end of your program. This implies that you need to specify two filenames at the command line, i.e., "java LibrarySearch <input> <output>". For the input file, we recommend the following format for all references:

```
type = “book”
callnumber = “QA76.73.J38S265”
authors = “Walter Savitch, Kenrich Mock”
title = “Absolute Java”
publisher = “Addison-Wesley”
year = “2009”

type = “journal”
callnumber = “P98.C6116”
title = “Computational Linguistics”
organization = “Association for Computational Linguistics”
year = “2008”
```

For the output file, you will write the references in the same format so that it can be used as an input file for the next run of your program.

(4) In addition to the sequential search in Assignment Two, you can use a HashMap as an index for the title keywords. To create such an index, you will tokenize the titles of all activities and create a mapping entry for each keyword. For example, if the word “Java” appears in the titles of references 0, 3, 7, and 10, the mapping entry will map “java” to a list of [0, 3, 7, 10]. Similarly, the mapping entry for “Programming” may map to a list of [0, 5, 7, 12, 15]. To maximize the possible matches, all the keywords in the index and from the search request need to be normalized to lower cases so that “Java”, “JAVA”, and “java” can all be matched.

To search for title keywords “Java Programming”, you will first find “java” in the index to get a list of [0, 3, 7, 10] and then find “programming” to get a list of [0, 5, 7, 12, 15]. After that, you can reduce the search to the intersection of [0, 7], since only the references at these locations in the ArrayList will contain both keywords “java” and “programming”. If there are also requirements for call number and a time period, you will then search the reduced list of [0, 7] sequentially for all matched activities. Obviously, this can greatly speed up the search performance for a request that contains title keywords. If no title keywords are specified in a search request, you will still need to search the entire ArrayList sequentially to find all matched references.

Note that this implementation assumes that all references are stored in a single ArrayList so that their locations can be used to identify them. The title index needs to be created as soon as you load the existing references from the input file and then updated each time you add a new reference into the ArrayList. You do not need to save the index before terminating the program since it can easily be computed in the next run of the program.

(5) You should keep all of your classes in one package and use Javadoc to create a set of external documents so that the TA can examine the contents of your package easily for marking.

Deliverables:

Same as Assignment Two.