CSC220 Lab04
Generic Programming

The goal of this week's lab is:

1. To practice generic programming
2. Learn how to use Comparator interface
3. Continue experimenting with object types

Things you must do:

1. There are many details in this lab. Make sure you read and follow this instruction carefully and do things in order.
2. Always remember Java is case sensitive.
3. You must use file names, class names, package names as instructed.
4. Your assignment submission must include the methods you have written during the lab.

Things you must not do:

1. You must not change the signature of any of these methods (name, parameters, ...) and just fill in the missing code inside them, unless you have been asked to do so.
2. You must not create any different class.

In this lab we are going to construct another version of a program for libraries that allows books to be checked in and out electronically. Similar to the last lab, a book is represented by an **ISBN**, an **author**, and a **title**, all of which **cannot change** once the book has been created. (Please note that **ISBNs are unique**.) A **library book** is a book together with a **holder** (representation of the person who has the book checked out) and a **due date**, both of which **can change** as needed. (Please note that for our purposes, all **holders are unique**.)

To make our task more challenging this time, some of the libraries that will use our program **represent holders with names**. Others **represent holders with phone numbers**. Furthermore, we hope to sell our program to even more libraries whose representation of holders we cannot anticipate. Thus, our library program must be **generic**. Finally, we hope to squeeze even more money out of our customers by offering two additional features: an operation that **retrieves the list of all books in the library sorted either by ISBN or by author** (for inventory purposes), and an operation that **retrieves the list of all overdue library books**.

Part 0

1. You are going to create a new project. You learned how to create a project in Eclipse before. Create a new Java project and call it **Lab04** (with no space and no other name – notice the capital 'L')

2. Create a package inside your project's src folder and call it **lab04** (no space and no other name – all lowercase).

3. We are going to use our library implementation from Lab03 as a starting point. **Import the following files into the lab04 package from the AllYourBase.zip from Google Drive (link on blackboard assignment) so we all have a consistent starting point:**

    a. Book.java
    b. Library.java
    c. LibraryBook.java
    d. LibraryTest.java

4. Run **LibraryTest.java**. All you should see is "Testing done." If you see anything else (or any red text) start debugging your code and ask for help from TA.

You need a correct implementation of the code to continue.

Part 1 – Generic LibraryBook

1. Modify your LibraryBook class to make the type of the library book's holder generic. Be sure to also modify the header for the class to be as follows:

```
public class LibraryBook<Type> extends Book
```

2. For the most part, modification will involve **replacing the String type for the library book's holder in your original class to Type** in the new class. **Be careful. It is not correct to replace every occurrence of String with Type.**

Part 2 – Generic Library

1. We will now do the same for the Library. Modify it to make the list of library books a list of *generic* library books. The header for your new class should be the following.

```
public class Library<Type>
```

2. For the **most part**, modification will involve replacing the String type for the library book's holder in your original class to Type in the new class and replacing the ArrayList<LibraryBook> type for the library in your original class to ArrayList<LibraryBook<Type>> in the new class. **Be careful. Go through the class to make sure you have applied required changes to make your class generic.**

Part 3 – LibraryGenericTest

1. Grab a copy of **LibraryGenericTest.java** and **PhoneNumber.java** from the

Lab04_files.zip file on Google Drive (link on Blackboard assignment) and include them into your lab04 package.

2. Now run **LibraryGenericTest**, you should ONLY see "Testing done". If you see anything else (especially red text), you need to debug your implementation.

3. The LibraryGenericTest class creates two libraries, one that identifies holders with (String) names and another that identifies holders with PhoneNumber objects.

4. **You MUST add more tests to this class to include more exhaustive testing.**

Part 4 – Retrieving a list of library books sorted by ISBN

1. Import java.util.Comparator to Library.java.

2. Grab a copy of the code in **part4.txt** from the lab ZIP folder and add it to **Library.java**.

    a. copy and paste it **inside** the curly braces of the class.

3. The provided code includes a method for sorting an ArrayList of items. Both the type of items in the ArrayList and the order of the sort is generic. The ordering is specified by the Comparator object passed to the method. Most of this code has been implemented for you. The only missing part is for you to complete the implementation of compare method inside IsbnComparator that implements comparator interface.

    a. The first step is to specify how many parameters the compare method should receive and what is/are the type(s). You can consult JavaDoc if are in doubt:

    https://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html

    b. You need to think about how to implement this method and what this method needs to return – again use JavaDoc (and class lectures) to accomplish this task.

4. Your last task is to write tests in LibraryGenericTest.java to test your getInventoryList method.

As you saw during the lab, we are going to construct a program for libraries that allows books to be checked in and out electronically. A book is represented by an ISBN, an author, and a title, all of which cannot change once the book has been created. (Please note that ISBNs are unique.) A library book is a book together with a holder (representation of the person who has the book checked out) and a due date, both of which can change as needed. (Please note that for our purposes, all holders are unique.)

Your job is to complete several methods defined that are incomplete.

Part 1 - Retrieving a list of library books sorted by author

- This is similar to the feature you implemented in the lab getInventoryList. You can accomplish this task by finishing the implementation of the AuthorComparator (the class declaration is provided, you must fill in the code), and then you must fill in the getOrderedByAuthor.
- You do not need to sort by last name, just sort by the full author String as is. If two books have the same author, this Comparator should break the tie with the book title.
- For example, Mushroom_Publishing.txt contains the following two books:

Moyra Caldecott	The Eye of Callanish
Moyra Caldecott	Crystal Legends

- AuthorComparator should treat "Crystal Legends" as less than "The Eye of Callanish", even though they have the same author, but since 'C' is alphabetically less than 'T'. To perform these comparisons, simply use String's built-in compareTo method.
- Back in getOrderedByAuthor, invoke the sort method with an instance of AuthorComparator.

Part 2 - Retrieving a list of overdue library books sorted by due date (<u>oldest first</u>)

This feature is left for you to implement. Hints:

- In getOverdueList, first make a copy of the list of library books, but include only those that are overdue (note that GregorianCalendar has a compareTo method for comparing dates).
    - **Be careful** when iterating over a library book whose due date is **NULL**! (Why?)
- Then invoke the sort method on the overdue list with an instance of the DueDateComparator class (a Comparator), for which the class declaration is provided, but you must fill in the rest of the implementation. If you don't know how to use sort look at the implementation of getInventoryList.

Part 3 – Testing

Test the features you implemented by adding your own examples in the LibraryGenericTest! We have provided a file that includes a list of sample books called Mushroom_Publishing.txt. In order to use this file:

- Right click on the Lab04 (i.e., the project name – not the src folder) and then "new->file" and type "Mushroom_Publishing.txt". A file with the same name will show up under "JRE System Library".
- Now grab a copy of Mushroom_Publishing.txt from the assignment 4 ZIP folder from Google Drive and paste the content of it into the file you just created.
- Now you can uncomment lines after "test a medium library" and start testing your implementation.