This assignment is due January 24 at 5:00pm. You must work in pairs on this assignment.

Name your package for this assignment "assignment2". Do not change the package name on any provided files.

The purpose of this assignment is to practice your knowledge of generic programming, specifically through the use of inheritance, Java generics, and function objects.

The assignment must be completed in pairs. For information on pair programming and finding a partner, see this document.

1. The Problem
   We have been asked 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.)

   To make our task more challenging, 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.

   This is a big job that is best completed in three phases.

   - *Phase 1:* Build the library using a specific representation of the library book holder (a `String` name). In this phase, the library is *not* generic. This phase uses the classes `Book`, `LibraryBook`, `Library`, and `LibraryTest`(see below).

This phase gives you practice using inheritance and Java basics.

- *Phase 2:* Modify the specific implementation completed in Phase 1 to make it generic. This will require using a placeholder for the type of the library book holder (rather than an actual `String` type). This phase uses the classes `Book`, `LibraryBookGeneric`, `LibraryGeneric`, and `LibraryGenericTest`(see below).
  This phase gives you practice using Java 5 generics.

- *Phase 3:* Add to the generic implementation completed in Phase 2 to include the extra operations of getting a sorted list of library books (based on 2 different orderings) and getting a list of overdue library books. This phase will add to the class `LibraryGeneric`(see below).
  This phase will give you practice using the Java `Comparator` interface and function objects.

2. Requirements

- *Phase 1*

  - `Book` and `LibraryBook` classes
    The base class `Book` has been started for you. The `equals` method is left for you to fill in. *Do not make any other changes to the* `Book` *class.*

    Construct a class `LibraryBook` derived from `Book` and containing the library book's holder (a `String`) and due date represented by a `GregorianCalendar` .

    (`import java.util.GregorianCalendar`)

    If a library book is checked in, its holder and due date should be set to `null`. The `LibraryBook` class must include the following methods (you may add other methods as needed).

    - `public LibraryBook(long isbn, String author, String title)`
    - `public String getHolder()`
    - `public GregorianCalendar getDueDate()`
    - Methods for checking a book in and out.
    - *Do not override the* `equals` *method in* `Book`.

assignment 2 files

Analysis Document

Algorithm analysis; Data Structures

Basic Sorting Algorithms

Recursive Sorting Algorithms

Linked Lists

Stacks and Queues

Trees

Graphs

Spring Break!

Hash Tables

Binary Heaps

File Compression

Comprehensive Project; Multithreading

Wrap Up

Final Exam and Review

29 April - 5 May

SETTINGS

Assignment administration

Submission

Course administration

- `Library` class
  The `Library` class has been started for you. Fill in the method implementations as indicated. *Do not change the method signatures.*

- `LibraryTest` class
  The sample test class LibraryTest has been provided for you. You should improve this class to include more exhaustive testing.

  The `LibraryTest` class references a text file of books, Mushroom_Publishing.txt. Download this file and place in your Eclipse project folder.

- *Phase 2*

  - `LibraryBookGeneric` class
    Make a copy of your `LibraryBook` class and modify it to make the type of the library book's holder generic. The header for your new class should be the following.

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

    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`.)
  - `LibraryGeneric` class
    Make a copy of your `Library` class and 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 LibraryGeneric<Type>
    ```

    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<LibraryBookGeneric<Type>>` in the new class.
  - `LibraryGenericTest` class
    The sample test class `LibraryGenericTest` has been provided for

you. You should extend this class to include more exhaustive testing.

The `LibraryGenericTest` class creates two libraries, one that identifies holders with (`String`) names and another that identifies holders with `PhoneNumber` objects. To use the test class, you must download and import the `PhoneNumber` class.

- *Phase 3*
  - Adding features to the `LibraryGeneric` class
    Add the code given here to the `LibraryGeneric` class you completed in Phase 2. (Copy the code and paste it just inside the curly braces of the class. Note that this code requires that you import `java.util.Comparator`.) *Do not change the method signatures.*

    Notice that 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.

  - *Feature 1:* retrieving a list of library books sorted by ISBN. This feature has been implemented for you. The `getInventoryList` method first makes a copy of the list of library books and then invokes the `sort` method with an instance of the `OrderByIsbn` class (a Comparator function object).

  - Feature 2: retrieving a list of library books sorted by author. This is similar to Feature 1, except you must provide the code for the `OrderByAuthor`comparator (for which the class declaration is provided, you must fill in the code), and you must fill in the `getOrderedByAuthor` method. 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

    The `OrderByAuthor` comparator 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. Then invoke the `sort` method with an instance of `OrderByAuthor`.

- *Feature 3:* retrieving a list of overdue library books sorted by due date (oldest first). This feature is left for you to implement.
  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). Then invoke the `sort` method on the overdue list with an instance of the `OrderByDueDate` class (a Comparator), for which the class declaration is provided, but you must fill in the rest of the implementation.

- **Be sure to test these features by adding to the `LibraryGenericTest` or creating a new test class!**

1. When preliminary coding is complete and your program compiles without error or warning, test the program thoroughly and systematically.
2. Your code should be well-commented (Javadoc comments are recommended) and formatted such that it is clear and easy to read. Be sure to put the names of both programming partners in the header comment of each file. You must have at least: comments for every method, describing what the method does, what the arguments are (and what they mean), and what the return value is, as well as any special cases that the method handles or can run in to. You should also have comments on any line or block of code that is not self-explanatory.

3. Zip your .java source code files (no `.class`, `.java` only) and **upload the zip file here by 5p on January 24**. Please submit just one solution per pair (i.e., one partner should upload the zip file and an analysis document, the other should only upload an analysis document).

4. Analysis Document (must be written **separately** and submitted by each programming partner) ***due January 24, at 5pm***

Due date:	Thursday, 24 January 2013, 5:05 PM

Submission feedback

Paymon Saebi
Tuesday, 5 February 2013, 6:32 AM

Grade: 78.00 / 100.00

Please see the new attached file for grading details.

(-10) for the permitted lateness due to exception throwing code.

Barsketis.txt

Submission

Analysis1.pdf
Final.zip

No further submissions are allowed.