**CSE 1325 – Summer 2013**

**Object-Oriented and Event-driven Programming in Java**

**Lab 2**

Design Document Due Date: **Thursday, June 27, 2013 at 11:59pm**

Lab Due Date: **Tuesday, July 2, 2013 at at 11:59pm**

(see instructions on website for how to turn this in - “lab submission info”)

Grade value: 8% out of 100% for all grades

**Objective:**

The theme of this project is to design classes and methods for managing books in the library (textbook, magazine, journal, newspaper, and novel) using object-oriented design (OOD) principles. Please make sure you adhere to OO principles throughout the design and implementation. The projects are designed to give you an opportunity to exercise the principles taught in the course. Remember, Java is only a vehicle for supporting OOD. Make sure you fully leverage the features available in Java for your benefit as long as we’ve touched on them in class. Ask before jumping into totally new concepts for your lab.

In this project, you will be designing the data abstractions and required operations on those abstractions for managing books in the library.

Be sure to check the DEDUCTIONS section at the end of this assignment to avoid penalties

You are also required to design your program in advance before you begin writing code. You will document your design and turn in the design document at least a week before the lab assignment is due. The goal of the design document is to assist you in developing the actual program.

**Description:**

Design classes, fields (attributes) and methods for the problem description below. Make sure you understand and differentiate between public/private fields and methods. Use static fields/methods where appropriate. Make sure you have at least one constructor (more as needed) with arguments. Each class should have appropriate get (accessor) and set (mutator) methods. Each class should have a toString method (and other formatting methods as needed).

Use a “Test” class to run the application (to test/call each method defined in classes). The test class should print the following menu, accept user input and execute the menu to provide understandable output. All inputs (as appropriate for the menu) should be accepted from the main screen (for this lab) and the program should prompt for the input with appropriate messages. If bounds are appropriate, please indicate it in the prompt and check for input errors. Test your source code—be sure every method works correctly.

1) Add a new book

2) Remove a book

3) Check out a book (input ISBN and the due date)

4) Check in a book (input ISBN)

5) Display all book details (along with their specialization)

6) Display all available book details

7) Display all checked-out books (including due date and how many days are left before the due date)

0) Exit program

The “Test” class has a “Library” class to manage books.

A. The Library

1. Has books: Use “ArrayList” and “Book” class

2. Adds a new book

3. Removes a book

4. Checks out a book: input ISBN

5. Checks in a book: input ISBN

6. Display all book

7. Display all available books

8. Display all checked-out books

B. The Book

1. Has an ISBN (10 digits): e.g., 1001234567

2. Has a title (string): e.g., Java Programming Language

3. Has a type (string): e.g., textbook, magazine, journal, newspaper, novel, and other

4. Has a publisher (string): e.g., UT Arlington Press

5. Has a page information (how many pages) (integer): e.g., 506

6. Has a price (float): e.g., 20.00

7. Has a publication year (integer): e.g., 2013

8. Has a status (integer): e.g., 0=available, 1=checked-out

9. Has a name who checked out the book (string): e.g., Tommy

10. Has a due date (LibraryDate): use DATE class

11. Calculate how many days are left before the due date

C. The LibraryDate (Don’t use Calendar or Date class that JAVA provides)

1. Has a day (integer): e.g., 1

2. Has a month (integer): e.g., 1

3. Has a year (integer): e.g., 2013

4. Calculate the date difference (implementing “-“ operator function)

D. Each employee has the following attributes EXAMPLE

1. First name John

2. Last name Smith

3. Identification number 006

4. Monthly salary $2500.00

5. Area of specialization (use enum) reference, special collections, research, desk, media

Each patron has the following attributes EXAMPLE

1. First name Mary

2. Last name Jones

3. Address "123 Sesame St. Baltiyork, TD 43564"

4. Gender male/female

5. The last item checked out by ISBN 1001234567

BONUS:

Java provides mechanisms for comments and input for Javadoc. Please make sure your code contains both. Also, submit the files generated by the Javadoc for your program. Look up the convention for specifying parameters (@param) and user information for javadoc. Every method should have its meaningful description for use by someone else. Use the relevant Javadoc tags specified in -http://en.wikipedia.org/wiki/Javadoc or <http://java.sun.com/j2se/javadoc/writingdoccomments/>

**Standard Java conventions and coding style:**

Be sure to observe standard Java naming conventions and style. These will be observed across all projects for this course, hence it is necessary that you understand and follow them correctly. They include:

* Class names begin with an upper-case letter, as do any subsequent words in the class name.
* Method names begin with a lower-case letter, and any subsequent words in the method name begin with an upper-case letter. This is called "camelCase".
* Class, instance and local variables begin with a lower-case letter, and any subsequent words in the name of that variable begin with an upper-case letter.
* Each set of opening/closing curly braces must be in the same column.
* All user prompts must be clear and understandable
* Give meaningful names for classes, methods, and variables even if they seem to be long. The point is that the names should be easy to understand for a person looking at your code
* All static final variables will be upper case identifiers (i.e.constants)

In addition, ensure that your code is properly documented in terms of comments and other forms of documentation wherever necessary.

**What and How to Submit**

For Lab #2 WRITE A DESIGN DOCUMENT FIRST. The design must include

a) all the classes you expect to write,

b) brief (one line) descriptions of each data member of the class

c) brief (one line) descriptions of each method (member function) of the class, and

d) some indication of the relationships between the classes.

The design document may be written as lines of text or as diagrams (such as a diagram that start with the main function at the top and all others below it, a tree, UML, etc.) or as some combination of those, but it must include the information listed for a), b), c), and d) above. Each class should encapsulate one thing and each piece of data at a lower level should have a more specific purpose than the class that owns it. Be sure to include all the classes that are described in this lab assignment. See the website or this assignment for the DESIGN DOCUMENT due date. It is usually ONE WEEK PRIOR to the lab due date.

A Lab #2 Design Document must be turned in in order for your Lab #2 assignment to be graded.

For this lab project itself, please submit using the Blackboard system at elearn.uta.edu. Please submit a .zip file that contains:

* Each .java file used in development with the appropriate name for the included class
* A document listing each test case that is in your test class and what the purpose is of each test. This is your test plan.
* A script file of a test session on omega
* A development notes file (standard text or word file) that describes at least three logical errors that you found while developing your code, how you solved it, and the total amount of time you spent on this project in terms of hours.

The .zip file containing your project must have the name format of: **proj2\_100number\_firstname\_lastname**

Upload this code to Blackboard at elearn.uta.edu

**A Few Helpful Tips:**

* Netbeans by default recompiles your class every time you save your file. If you want to force everything to be recompiled, press the ‘clean and build’ button.
* Ctrl + space in most IDE’s will show a list of available members (or variables) and methods (or functions).

**Grading Rubric:**

**Code: (50%)**

Class header and method headers for all classes (4 points)

Comments (line comments and block comments) (3 points)

Style (following Java conventions, indentation, consistency, meaningful identifiers, lateral separation of code from line comments, etc.) (3 points)

Modularity (division of the problem into small parts) (5 points)

Correct declaration of the required classes with data and methods and any other needed classes

Library (5 points)

Book (7 points)

LibraryDate (5 points)

Employee (5 points)

Other classes (5 points)

Test plan includes at least two tests for each of the required

functions of add, remove, check in, check out, and uses the

display functions to verify test results (8 points)

**Output: (50%)**

User clearly understands what is being requested for input (3 points)

Required tasks performs correctly

( add, remove, check in, check out,display (3), exit) (15 points)

Set functions handle inputs correctly including error checking (6 points)

Get functions provide values safely (4 points)

Output gives clear information to explain the values to the user (3 points)

Output contains all the given test data (if any) and at least two additional data sets, i.e at least one other library, two add’l books, and two add’l dates up to the maximum of 5 (12 points)

Output demonstrates all elements of test plan (7 points)

**Grading Deductions:**

Labs which have errors in them and do not terminate normally will receive an overall grade of 0 (zero) [-100 deduction]

Late submission of softcopy of design document, code and/or script file or other documents to Blackboard will result in an overall grade of 0 (zero) UNLESS student has obtained prior instructor approval [-100 deduction]

Use of Java language elements not yet discussed in class by the lab due date will result in potential deduction of points - discuss with instructor before using.

Use of Java public variables (C equivalent to global variables) will result in an overall grade of 0 (zero) [-100 deduction]

Use of unstructured control elements, like the break (except in the case of switch statements) or continue keywords, will result in an overall grade of 0 (zero) [-100 deduction]

See lab re-grading policy to understand how to gain back some points on a lab.

Miscellaneous: If you have questions, e-mail Dr. T ([tiernan@uta.edu](mailto:tiernan@uta.edu)) and the TA ([mingon.kang@mavs.uta.edu](mailto:seanp@uta.edu) )