CIS351

Homework 5

Academic Integrity

- Remember the Academic Integrity Policies, of this course.

Objectives

- Implement a specific design from UML diagrams.
- Validate the arguments before updating an object.
- Write toString methods that use string formatting.
- Search for given values within an array of objects.
- Solve problems using object-oriented techniques.

Download Materials

Download the following files from Blackboard

- DateUtils.java - DO NOT MODIFY. This utility class is provided for you.
- Driver.java - DO NOT MODIFY. This is the driver class that runs the entire project for your testing.
- Library.java - DO NOT MODIFY. The Library class is provided for you. It is simulating a real Library.
- lflbooks.txt - Example books holding books information.
- patrons.txt - Example patrons holding patron information.
- SampleOutput.txt - A sample output of the entire program

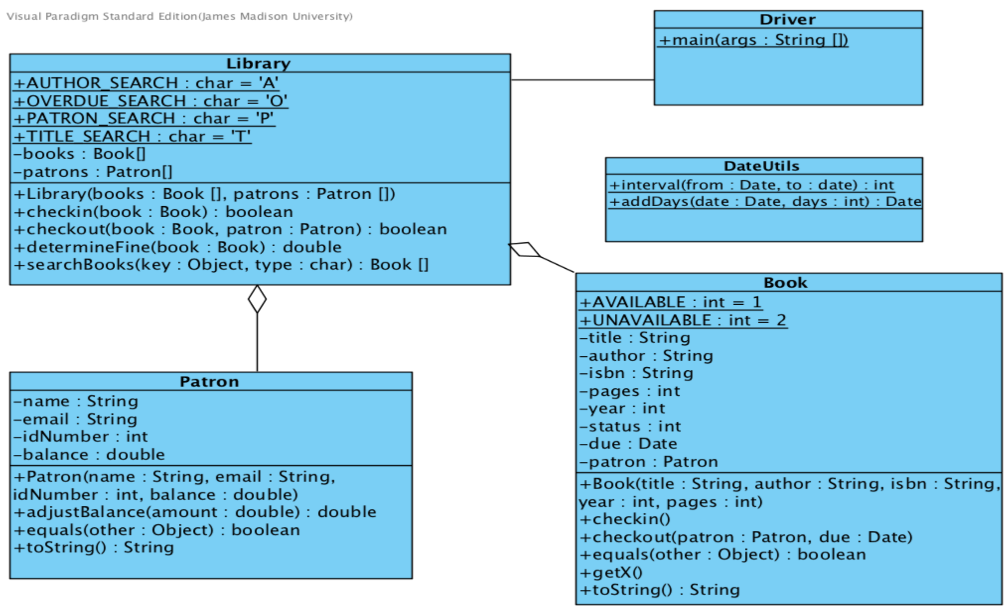
Background

This assignment is inspired by the Little Free Library organization which has managed to charter small "libraries" across the world. Each "library" encourages patrons to take a book or leave a book to promote the accessibility of books and reading in general. They are community spaces, many with benches or seats nearby and all are maintained by volunteers. We will be simulating an information management system to be used in these libraries.

Requirements

In this assignment, imagine that the Little Free Library organization has contracted with the SU EECS department to track how much money they could earn if they were a commercial organization that charged late fees for overdue books. This information will then be used for local librarians to apply for funding from other non-profit foundations or the government. Each book and patron will be modeled by classes with relevant data and methods as outlined in the UML diagram and descriptions shown below. The Library class will then have arrays of books and patrons along with methods that allow for books to be checked in, checked out, and found in addition to determining fines.

Look at the following UML Diagram of this library system. All of the required files are provided for you except **Patron.java and Book.java**, which you need to implement in this homework.


Visual Paradigm Standard Edition(James Madison University)
Driver
+main(args : String [])
Library
+AUTHOR_SEARCH : char = 'A'
+OVERDUE_SEARCH : char = 'O'
+PATRON_SEARCH : char = 'P'
+TITLE_SEARCH : char = 'T'
-books : Book[]
-patrons : Patron[]
+Library(books : Book [], patrons : Patron [])
+checkin(book : Book) : boolean
+checkout(book : Book, patron : Patron) : boolean
+determineFine(book : Book) : double
+searchBooks(key : Object, type : char) : Book []
DateUtils
+interval(from : Date, to : date) : int
+addDays(date : Date, days : int) : Date
Patron
-name : String
-email : String
-idNumber : int
-balance : double
+Patron(name : String, email : String, idNumber : int, balance : double)
+adjustBalance(amount : double) : double
+equals(other : Object) : boolean
+toString() : String
Book
+AVAILABLE : int = 1
+UNAVAILABLE : int = 2
-title : String
-author : String
-isbn : String
-pages : int
-year : int
-status : int
-due : Date
-patron : Patron
+Book(title : String, author : String, isbn : String, year : int, pages : int)
+checkin()
+checkout(patron : Patron, due : Date)
+equals(other : Object) : boolean
+getX()
+toString() : String

You must read through the code of **Library** class to understand the structure and interaction of this program. You don't actually need the Driver to complete the assignment; it is simply provided to show how your classes might be used in a larger application. And it is provided in case you want to test your written two classes and observe how these are acting like a puzzle piece in a large puzzle (which in our case is the Library system).

HELP: DO NOT MODIFY Library.java. It is provided for you. Please read this class very carefully to understand what each method is doing. This will also help you to code *Patron.java* and *Book.java*.

Specification of classes that YOU NEED TO CODE

Book.java (YOU NEED TO CODE)

1. The Book `constructor` will be used to initialize all attributes of the Book object. The initial `status` of the book should be AVAILABLE, and the patron and due date should be null.

2. Like the constructor, the `checkin` method should set the book's status to AVAILABLE and assign null to both patron and due date.

3. The `checkout` method should make the book UNAVAILABLE and save the given patron and due date in this book the object.

4. The `equals` method checks whether two books have the same isbn. If the given parameter is an instance of a Book object, compare the other book's isbn with this books isbn. If the given parameter is a String, compare other books isbn with this given string to determine if the two books are same.

5. In the UML diagram, there is a `getX` method, which actually represents eight methods, meaning you should write one accessor method corresponding to each of the eight attributes. Note: that you may choose to write the setX methods too. But it is upto you and it is not a requirement.

6. The `toString` method should return a String representation of the book. If a book is initialized with the following parameters `("aaa", "bbb", "1234BBBBBBBBB", 2013, 10)`, it should return a String as follows: `"Title: aaa, Author: bbb, ISBN: 1234BBBBBBBBB, Year: 2013, Pages: 10."`

Patron.java (YOU NEED TO CODE)

1. The Patron `constructor` will be used to initialize all attributes of the Patron object.

2. The `adjustBalance` will update the current balance of the patron. For example if the current balance of a Patron is 2.0 and the amount parameter is 4.5, this method should assign 6.50 to the balance of this Patron and return the balance.

3. The `equals` method checks whether two patrons have the same id number. If the given parameter is an instance of a Patron object, compare the other Patron's idNumber with this Patron object's idNumber. If the given parameter is an instance of an Integer object, compare the given Integer objects value with this patron object's idNumber to determine if the two Patrons are same.

4. The `toString` method should return a String representation of the Patron. If a patron is initialized with the following parameters ("bob", "eee", 2, 5.5), it should return a String *exactly* as follows: "Name: bob, Email: eee, ID: 2, Balance: $5.50."

Hints and Tips

To implement both equals methods, you'll need to use the instanceof operator. `instanceof` is a keyword that is used for checking if a reference variable is containing a given type of object reference or not. Read more here.

In yout Book equals method, you need to check if the given parameter is an instance of Book or is it an instance of String. Here is an example just so that you can get started:

```
if (other instanceof Book)  {
    Book b = (Book) other; // casting the other Object to Book
    //now check if two Book object's isbn are equal
}
else if (other instanceof String){
    // looks like the passed parameter is not Book, rather a string
    // so now check if this String isbn is equal to the other Books isbn
}
```

Testing your program

In order to test your code, first you need to have all the provided java files and the two text files in the same folder. Now compile them all and execute the Driver.java. If your two classes are implemented correctly then you should be able to interact with the Libray system. it will present you different options. See a sample run of the program is provided in `SampleOutput.txt` file.

Submission Instruction

Submit a zip folder consisting of the following

1. completed Book.java and Patron.java
2. filled out Reflection Form (https://farahman.github.io/reflection-form.pdf)
3. filled out cover sheet (https://farahman.github.io/CoverSheet.pdf)

Grading Criteria

Total points: 100 points

Correctness of Patron.java - 45pt
Correctness of Book.java - 45pt
Reflection sheet - 10pt

Farzana Rahman / frahman@syr.edu