Lab Assignment 3:
Inheritance - Library Catalog System

Due Dates	
Wednesday Labs	**Wednesday 13th Mar. @ 11:59**
Thursday Lab	**Thursday 14th Mar. @ 11:59**

Objectives

- Review inheritance
- Review overriding superclass methods in subclasses
- Follow the Software Life Cycle model, with pseudocode and code refinement steps
- Review UML diagrams

Problem Specification

Following the Software Life Cycle model, write a Java application to keep track of a library's collection of materials. The collection will be read from an **input** file; users should be able to display a list of materials and check out materials using their call numbers.

The library materials are either books or periodicals. The information stored for each type of library material is shown below.

- Books (checked out for 21 days)
  - Book Title
  - Author
  - Genre
- Periodicals (checked out for 7 days)
  - Periodical Title
  - Volume
  - Issue
  - Subject

All library materials have a **call number**, and **can be checked out**. The **date checked out** should be the current date, and the **due date** depends on the type of material (21 days after current date for **books**; 7 days for **periodicals**). When displaying the item, show the specific information, along with the call number, whether or not the item is checked out, and if checked out, the check-out and due dates (see example output below).

The input file with data about available library materials is organized as follows:

- The first line is the number of books in the library's collection.

- The second line is the number of periodicals in the library's collection.
- Each line after that contains a single library item (fields separated by commas)
    - Books (indicated by the letter B)
        - Fields included: *Call number, Book title, Author, Genre*
    - Periodicals (indicated by the letter P)
        - Fields included: *Call number, Periodical title, Volume, Issue, Subject*

Example Input:

2
2
B,C124.S17,The Cat in the Hat,Dr. Seuss,Children's Literature
P,QJ072.C23.37.4,Computational Linguistics,37,4,Computational Linguistics
P,QJ015.C42.55.2,Communications of the ACM,55,2,Computer Science
B,F380.M1,A Game of Thrones,George R. R. Martin,Fantasy Literature

The application should exhibit the following functionality (see the sample output below):

- Read the contents of the library's collection from the input file.
    - The file name should be **hardcoded** in the application.
- Allow the user to choose from a menu of 3 options:
    - ***Display collection***
        - Display the full list of library materials including the check-out status (YES or NO);
            - If checked out, display the check-out date and the due date.
    - ***Check out materials***
        - Ask the user for the call number, and then find the matching item.
        - If the item is already checked out, do not allow the user to check it out; display a message indicating it is not available.
            - Display the menu again to the user.
        - If the item is not checked out, set the **check-out** date and **due date** accordingly.
        - Display the item information
    - ***Quit***
        - Exit the application
- Allow the user to continue making requests (selecting options) until s/he selects the ***Quit*** option.

Example Output:

---------- Menu ----------
1) Display collection
2) Check out materials
3) Quit

Please choose an option: 1

Book Title: The Cat in the Hat
Author: Dr. Seuss
Genre: Children's Literature
Call Number: C124.S17
Checked Out: NO

Periodical Title: Computational Linguistics
Volume: 37
Issue: 4
Subject: Computational Linguistics
Call Number: QJ072.C23.37.4
Checked Out: NO

Periodical Title: Communications of the ACM
Volume: 55
Issue: 2
Subject: Computer Science
Call Number: QJ015.C42.55.2
Checked Out: NO

Book Title: A Game of Thrones
Author: George R. R. Martin
Genre: Fantasy Literature
Call Number: F380.M1
Checked Out: NO

---------- Menu ----------
1) Display collection
2) Check out materials
3) Quit

Please choose an option: 2

Enter the call number: F380.M1

Book Title: A Game of Thrones
Author: George R. R. Martin
Genre: Fantasy Literature
Call Number: F380.M1
Checked Out: YES
Date Out: 02/08/16
Date Due: 02/29/16

---------- Menu ----------
1) Display collection
2) Check out materials
3) Quit

Please choose an option: 2

Enter the call number: F380.M1
Item is not available.

```
---------- Menu ----------
   1) Display collection
   2) Check out materials
   3) Quit
--------------------------------
Please choose an option: 3
```

Design Requirements

Your application **MUST** make use of a proper inheritance relationship. It should have a **superclass** representing a general library item, and **subclasses** representing the specific types of library items included in the collection (**books** and **periodicals**).

Specifically, your project should have a class **LibraryItem** which implements the **ILibrary** interface. This interface is provided and **should not be changed**. All methods in the interface must be implemented by class **LibraryItem**.

Class **LibraryItem** has two subclasses - **Book** and **Periodical**. Your application should store objects (library items) of a similar type in an array of the appropriate type. You should therefore have two arrays to store the library materials: one array for books and another one for periodicals.

There is also a **Controller** class which implements the **IController** interface. This class is used by the main class to run the program.

The code for the main class **LA3Test** (which has only the ***main*** method) is provided and **should not be modified**. This serves as the test class for your program.

NOTE: ***Any common fields or methods between the subclasses should be defined in the superclass. Specific functionality, such as additional data members or overriding methods, should be defined in the specific subclass.***

The **interfaces** and the **main class** are provided below.

```
public interface IController {

        /**
         * Displays the collection of library items on the screen
         */
        public void displayCollection();

        /**
         * Requests for the call number from the user, uses the findItem()
         * method to check if that item exists in the library, and if it does
         * calls the checkOut() method for that item and prints out the item
         * that has been checked out.
         */
        public void checkoutMaterials();

        /**
```

```
        * Searches in both the array of books and the array of periodicals
        * for the book with the call number received as a parameter.
        * @param callNum The call number of the item requested by the user
        * @return The requested item, or 'null' if item does not exist.
        */
       public ILibrary findItem(String callNum);

       /**
        * Displays the menu options to the user.
        */
       public void showMenu();

       /**
        * Reads data from the input file and stores the items in the
        * appropriate array.
        * @param fileName The name of the input file.
        * @throws IOException Included in case input file is not found.
        */
       public void readInput(String fileName) throws IOException;

}      // End of interface IController


public interface ILibrary {

       /**
        * Sets the boolean value checkedOut to true, and
        * initializes the dateChecked out attribute (a
        * GregorianCalendar object).
        */
       public void checkOut();

       /**
        * Generates a string with the details of the library item
        * whose call number has been input by the user (see example output)
        * and returns that string.
        * If the user wants to check out the library item, the string to be
        * returned also includes information that the item has been checked out,
        * the date it was checked out, and the due date by which the item should be returned.
        * @return A string with details of this library item.
        */
    public String toString();

    /**
     * Returns the call number of this object.
     * @return the callNumber
     */
    public String getCallNumber();

    /**
     * Returns true or false depending on if this item has been checked out.
     * @return the boolean value for isCheckedOut
     */
```

```
    public boolean isCheckedOut();

    /**
     * Returns the date this item was checked out.
     * @return the dateCheckedOut
     */
    public GregorianCalendar getDateCheckedOut();

    /**
     * Returns the date this item is due to be returned.
     * @return the dateDue
     */
    public GregorianCalendar getDateDue();

    /**
     * Sets the dateDue to the parameter received.
     * @param dateDue the dateDue to set
     */
    public void setDateDue(GregorianCalendar dateDue);

}       // End of ILibrary interface


public class LA3Test {

        public static void main(String[] args) throws IOException {
                // TODO Auto-generated method stub

                Scanner keyboard = new Scanner(System.in);
                IController control = new Controller(keyboard);
                control.readInput("input.txt");
                String response = "";
                boolean quitFlag = false;

                while (!quitFlag) {
                        control.showMenu();
                        response = keyboard.nextLine();
                        System.out.println();
                        switch (response) {
                                case "1":
                                        control.displayCollection();
                                        break;
                                case "2":
                                        control.checkoutMaterials();
                                        break;
                                case "3":
                                        quitFlag = true;
                                        break;
                        }
                }

                System.out.println("Good bye!");
                keyboard.close();
```

```
        }

}       // End of main class
```

Hints

1) Working with dates in Java is very easy! You need to use the **GregorianCalendar** class (the name just refers to the type of calendar familiar to most of the western world). You can get the current date by instantiating a new **GregorianCalendar** object. You can copy the date using the clone method, and modify the date by using the add method. The add method below adds 50 days to the date.

```
import java.util.Calendar;
import java.util.GregorianCalendar;
...
dateCheckedOut = new GregorianCalendar();
dateDue = (GregorianCalendar)dateCheckedOut.clone();
dateDue.add(Calendar.DAY_OF_YEAR, 50);
String.format("Date Out: %tD\n", dateCheckedOut);
```

The **%tD** format specifier will print the date in *mm/dd/yy* format
Output for this code snippet will be (assuming today's date is 02/08/19):

```
Date Out: 03/30/19
```

2) Remember that the split method can be used to split a string using a specified delimiter, and returns an array of strings. The following line of code splits a string (inputString) delimited by commas.

```
String[ ] line = inputString.split(",");
```

Additional Requirements

Software Life Cycle Report with UML Class diagram

You are required to follow the Software Life Cycle (SLC) methodology presented in class, and write the SLC Report, explaining how you followed the nine phases of the Software Life Cycle in this assignment.

For this LA, you are also required (for the first time) to generate the **UML Class diagram** for your application, showing all classes, their fields/attributes, the methods they provide, and the relationships between classes (subclasses and superclasses, classes and the interfaces they implement, and associations between classes where applicable). The process of generating a UML diagram in Eclipse has been covered in the Labs; you will need to follow that process to generate the UML Class diagram for your project.

A proper *design* (with stepwise pseudocode refinement), a proper *coding* method (with stepwise code refinement starting from the most detailed pseudocode refinement), and proper *testing* are all essential. For reference, please see the **Sample SLC Report** (covered in class) on Elearning.

Note: The SLC report with correct pseudocode development will be worth 40% of the total LA grade.

You will need to **generate Javadoc** for your project. If Javadoc is correctly generated, a "**doc**" folder (for Javadoc) should be created in your project.

Coding Standards

You must adhere to all conventions in the CS 1120 Java coding standards (available on Elearning for your Lab). This includes the use of white spaces and indentations for readability, and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions also for naming classes, variables, method parameters and methods.

Assignment Submission

- Generate a .zip file that contains all your files including:
  - Program Files
  - Any input or output files
  - The SLC Report (a text with description of all nine phases of the Software Life Cycle)

- Submit the .zip file to the appropriate folder on ELearning.

NOTE: The eLearning folder for LA submission will remain open beyond the due date but will indicate how many days late an assignment was submitted where applicable. The dropbox will be inaccessible seven days after the due date by which time no more credit can be received for the assignment.

The penalty for late submissions as stated in the course syllabus will be applied in grading any assignment submitted late.