Project #1 – 45 points

Due Date

Tuesday, February 9, by 11:59pm.

Submission

One of the team members is responsible to submit **the zipped project folder** to Canvas. The zipped file MUST include the following subfolders.

(1) Source folder **src**, including all Java files [30 points]
- MUST include both team members' names with @author in the comment block on top of EVERY Java class.
- In the **Date class**, you MUST include a testbed main, which implements the test cases in the test design document.

(2) Javadoc folder doc, including all the files generated. [5 points]

(3) Test design document, which is typed with a document editor, including all test cases you used to test the isValid() method in the Date class. [10 points]

Project Description

In this project, you will implement an array-based container class **Library** and use it to hold the information of a list of books owned by a library. An instance of the Library class is a growable bag data structure with an initial capacity of 4, and automatically grows (increases) the capacity by 4 whenever it is full. You are **NOT allowed** to use the **ArrayList class, or** you will **get 0 points** for this project. A Kiosk class must be included in the project to provide library services and process the command lines read from the console. The library services shall include **adding**, **removing** books to/from the library catalog, **checking out** and **returning** books, as well as **displaying** the list of books by the date published or by the books' serial numbers.

Scanner class and **StringTokener class** should be used to read the command lines from the console. When your project starts running, it shall display "`Library Kiosk running.`"; next, it will read and process the command lines continuously until the user enters the "`Q`" command to quit. Before the project stops running, display "`Kiosk session ended.`"

A command line always begins with a command and followed by a comma and some data tokens. Each data token is also delimited by a comma. Some examples of valid command lines are demonstrated below. All commands are case-sensitive, which means the commands with lowercase letters are invalid. You are required to deal with bad commands that are not supported.

- Adding a book

```
A,Programming in Java,11/20/2019
```

A is a command for adding a book to the library, followed by the name of the book, and the date published. If the bag is full, the bag automatically grows the capacity by 4. A serial number will be automatically generated to create the book instance with the name and the date published. The serial number is a five-digit number starting 10001, which will be increased by 1 for each subsequent instance of Book. The date will always be in **mm/dd/yyyy** format. However, you must check if the date is valid with the isValid() method in the Date class (see page #3 below.) Display "`Invalid Date!`" if the date is invalid; otherwise, display "`Programming in Java added to the Library.`" on the console when the book is added.

- Removing a book

  `R,10005`

  R is a command for removing a book from the library given a book's serial number. If the book doesn't exist, display `"Unable to remove, the library does not have this book.";` otherwise, display `"Book# 10005 removed."`.

- Checking out a book

  `O,10005`

  O is a command for checking out a book from the library. If the library doesn't own the book, or the book has already been checked out, display "`Book#1005 is not available.`", otherwise, display "`You've checked out Book#10005. Enjoy!`".

- Returning a book

  `I,10005`

  I is a command for returning a book to the library. If the book doesn't belong to the library, or the book is not checked out, display "`Unable to return Book#10005.`", otherwise, display "`"Book#10005 return has completed. Thanks!"`

- Printing the library catalog

```
PA  //output the list of books to the console with the current sequence
PD  //output the list of books by the dates published in ascending order
PN  //output the list of books by the book numbers in ascending order
```

- **Q** command will stop the program and display `"Kiosk session ended."`

Program Requirement

1. You MUST follow the software development ground rules. You will lose points if you are not following the rules.
2. There are sample input and output at the end of this document for your reference. The graders will be using the test cases in the order of the sample input to test your project. You will **lose 2 points** for each incorrect output.
3. Each Java class must go in a separate file. **-2 points** if you put more than one Java class into a file.
4. Your program MUST handle bad commands! **-2 points** for each bad command not handled.
5. You are not allowed to use any Java library classes, except Scanner, StringTokenizer and Calendar classes. **-5 points** for each violation,
6. You MUST include the classes below. **-5 points** for each class missing. Add additional classes if necessary.
   (a) **Book class**.

```
public class Book {
       private String number; //a 5-digit serial number unique to the book
       private String name;
       private boolean checkedOut; //set to true if the book has been checked out
       private Date datePublished;
       ...
       @Override
       public boolean equals(Object obj){ }
       @Override
       public String toString() { }
}
```

   - This class defines the abstract data type Book, which encapsulates the data fields and methods of a book. The book number is a 5-digit serial number automatically generated starting 10001. You CANNOT add other data members, EXCEPT static variables or constants.

- You CANNOT read from console or use System.out in this class. **-2 points** for each violation.
- **toString()** method returns a textual representation of a book in the following format.

```
Book#10007::Design Patterns::5/30/1996::is available.
```

- **equals()** method returns true if the serial numbers for the 2 book objects are the same.
- You CANNOT change the signatures of the **toString()** and **equals()** methods. You cannot remove the @Override tags. **-2 points** for each violation. You CAN add constructors, private methods (helper methods) and public methods.

(b) **Library class**.

```
public class Library {
     private Book[] books; // array-based implementation of the bag data structure
     private int numBooks; // the number of books currently in the bag

     public Library() { } //default constructor to create an empty bag
     private int find(Book book) { } // helper method to find a book in the bag
     private void grow() { }  // helper method to grow the capacity by 4
     public void add(Book book) { }
     public boolean remove(Book book) { }
     public boolean checkOut(Book book) { } //true if checking out is successful
     public boolean returns(Book book) { }
     public void print() { } //print the list of books in the bag
     public void printByDate() { } //print the list of books by datePublished (ascending)
     public void printByNumber() { } //print the list of books by number (ascending)
}
```

- This is the **container class** that defines the abstract data type Library to hold library catalog and its operations. You CANNOT add other data members, EXCEPT constants. You CANNOT change the signatures of the methods listed. **-2 points** for each violation. You must implement all the methods listed, **-2 points** for each method not implemented or not used. You can add other methods if necessary.
- You CANNOT use **System.out** in this class, except the 3 **print()** methods, **-2 points** for each violation.
- The **remove()** method calls the helper method **find()** and finds the index of the book to be removed. This method maintains the current sequence of books in the array after the removal, **-3 points** if this is not done.
- You can use any sorting algorithms for sorting.

(c) **Date class.**

```
public class Date {
       private int year;
       private int month;
       private int day;

       public Date(String date) { } //taking mm/dd/yyyy and create a Date object
       public Date() { } //create an object with today's date (see Calendar class)
       ...
       public boolean isValid() { }
}
```

- This class defines the properties of a Date object. You CANNOT add other data members to this class, and you CANNOT use **System.out** in this class. **-2 points** for each violation.
- The **isValid()** method checks if a date is valid.
  - For a date with the year less than 1900 or a date beyond today's date is invalid.
  - For the month, January, March, May, July, August, October and December, each has 31 days; April, June, September and November, each has 30 days; February has 28 days in a normal year, and 29 days in a leap year. To determine whether a year is a leap year, follow these steps:

Step 1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
Step 2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
Step 3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
Step 4. The year is a leap year.
Step 5. The year is not a leap year.

- You MUST design the test cases to thoroughly test the **isValid()** method. You must write a testbed main and implement the test cases. You must follow the instructions in the "Test Specification" section of the Software Development Ground Rules. You will **lose 10 points** if you do not submit the test document, or not follow the format. In the testbed main, you MUST write code to print out the test results to the console showing the test cases are passed or not passed; **-5 points** if the testbed is missing.

(d) **Kiosk class** for processing the command lines from the console.

```
public class Kiosk {
        ...
    public void run() { }
       ...
}
```

- This class is the **user interface class** that handles the input and output. You must define a **run()** method or you will **lose 5 points**.
- You can define the data members you need and define some private methods to handle the commands. Make sure you follow the ground rules and have a good programming style.

(e) **RunProject1 class** is a driver class to run your Project 1. The main method will call the **run()** method in the Kiosk class.

```
public class RunProject1 {
    public static void main(String[] args) {
        new Kiosk().run();
    }
}
```

7. You must follow the instructions in the Software Development Ground Rules and comment your code. You are required to **generate the Javadoc** after you properly commented your code. Your Javadoc must include the documentations for the constructors, private methods and public methods of all Java classes. Generate the Javadoc in a single folder and include it in your project folder to be submitted to Canvas. Please double check your Javadoc after you generated it and make sure the descriptions are not empty. You will lose points if the descriptions in the Javadoc is empty. You will **lose 5 points** for not including the Javadoc.

Sample Input

```
pa
a
pd
AA
PP
RR
PA
PD
PN
A,Programming in Java,3/20/2010
A,Software Methodology,4/1/2018
A,Artificial Intelligence,12/01/2005
A,Linear Algebra,1/31/2020
```

```
A,Clean Code,3/31/2010
A,Pro Git,4/30/2018
A,Design Patterns,5/30/1996
A,Data Science,2/29/2008
PA
PD
PN
A,Programming in C++,3/20/2011
A,Object Oriented Analysis and Design,4/30/2018
A,Refactoring,1/31/2005
A,Java for Dummies,1/31/2020
A,Database Management Systems,2/28/2010
A,eXtreme Programming,2/10/2010
A,Doing Agile Right,5/30/1996
A,Data Structure with Java,2/29/2008
PA
r,10012
R,10012
R,10012
PA
o,10012
O,10012
i,10012
I,10012
O,10015
O,10015
PA
i,10015
I,10015
PA
PD
R,10005
PA
A,A fun book,31/2/2000
A,A fun book,13/2/2020
A,A fun book,2/29/2021
A,A fun book,2/29/2009
A,A fun book,4/31/2009
A,A fun book,3/32/2009
A,A fun book,3/31/1800
A,A fun book,10/30/2022
A,A fun book,3/30/2021


Q

A,this line should not be processed,11/11/2020
```

Sample Output

```
Invalid command!
Invalid command!
Invalid command!
Invalid command!
Invalid command!
Invalid command!
Library catalog is empty!
```

```
Library catalog is empty!
Library catalog is empty!
Programming in Java added to the library.
Software Methodology added to the library.
Artificial Intelligence added to the library.
Linear Algebra added to the library.
Clean Code added to the library.
Pro Git added to the library.
Design Patterns added to the library.
Data Science added to the library.
**List of books in the library.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
Book#10005::Clean Code::3/31/2010::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10007::Design Patterns::5/30/1996::is available.
Book#10008::Data Science::2/29/2008::is available.
**End of list
**List of books by the dates published.
Book#10007::Design Patterns::5/30/1996::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10008::Data Science::2/29/2008::is available.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10005::Clean Code::3/31/2010::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
**End of list
**List of books by the book numbers.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
Book#10005::Clean Code::3/31/2010::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10007::Design Patterns::5/30/1996::is available.
Book#10008::Data Science::2/29/2008::is available.
**End of list
Programming in C++ added to the library.
Object Oriented Analysis and Design added to the library.
Refactoring added to the library.
Java for Dummies added to the library.
Database Management Systems added to the library.
eXtreme Programming added to the library.
Doing Agile Right added to the library.
Data Structure with Java added to the library.
**List of books in the library.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
Book#10005::Clean Code::3/31/2010::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10007::Design Patterns::5/30/1996::is available.
Book#10008::Data Science::2/29/2008::is available.
Book#10009::Programming in C++::3/20/2011::is available.
Book#10010::Object Oriented Analysis and Design::4/30/2018::is available.
Book#10011::Refactoring::1/31/2005::is available.
Book#10012::Java for Dummies::1/31/2020::is available.
Book#10013::Database Management Systems::2/28/2010::is available.
Book#10014::eXtreme Programming::2/10/2010::is available.
Book#10015::Doing Agile Right::5/30/1996::is available.
Book#10016::Data Structure with Java::2/29/2008::is available.
```

```
**End of list
Invalid command!
Book# 10012 removed.
Unable to remove, the library does not have this book.
**List of books in the library.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
Book#10005::Clean Code::3/31/2010::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10007::Design Patterns::5/30/1996::is available.
Book#10008::Data Science::2/29/2008::is available.
Book#10009::Programming in C++::3/20/2011::is available.
Book#10010::Object Oriented Analysis and Design::4/30/2018::is available.
Book#10011::Refactoring::1/31/2005::is available.
Book#10013::Database Management Systems::2/28/2010::is available.
Book#10014::eXtreme Programming::2/10/2010::is available.
Book#10015::Doing Agile Right::5/30/1996::is available.
Book#10016::Data Structure with Java::2/29/2008::is available.
**End of list
Invalid command!
Book#10012 is not available.
Invalid command!
Unable to return Book#10012.
You've checked out Book#10015. Enjoy!
Book#10015 is not available.
**List of books in the library.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
Book#10005::Clean Code::3/31/2010::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10007::Design Patterns::5/30/1996::is available.
Book#10008::Data Science::2/29/2008::is available.
Book#10009::Programming in C++::3/20/2011::is available.
Book#10010::Object Oriented Analysis and Design::4/30/2018::is available.
Book#10011::Refactoring::1/31/2005::is available.
Book#10013::Database Management Systems::2/28/2010::is available.
Book#10014::eXtreme Programming::2/10/2010::is available.
Book#10015::Doing Agile Right::5/30/1996::is checked out.
Book#10016::Data Structure with Java::2/29/2008::is available.
**End of list
Invalid command!
Book#10015 return has completed. Thanks!
**List of books in the library.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
Book#10005::Clean Code::3/31/2010::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10007::Design Patterns::5/30/1996::is available.
Book#10008::Data Science::2/29/2008::is available.
Book#10009::Programming in C++::3/20/2011::is available.
Book#10010::Object Oriented Analysis and Design::4/30/2018::is available.
Book#10011::Refactoring::1/31/2005::is available.
Book#10013::Database Management Systems::2/28/2010::is available.
Book#10014::eXtreme Programming::2/10/2010::is available.
Book#10015::Doing Agile Right::5/30/1996::is available.
Book#10016::Data Structure with Java::2/29/2008::is available.
**End of list
**List of books by the dates published.
Book#10007::Design Patterns::5/30/1996::is available.
```

```
Book#10015::Doing Agile Right::5/30/1996::is available.
Book#10011::Refactoring::1/31/2005::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10008::Data Science::2/29/2008::is available.
Book#10016::Data Structure with Java::2/29/2008::is available.
Book#10014::eXtreme Programming::2/10/2010::is available.
Book#10013::Database Management Systems::2/28/2010::is available.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10005::Clean Code::3/31/2010::is available.
Book#10009::Programming in C++::3/20/2011::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10010::Object Oriented Analysis and Design::4/30/2018::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
**End of list
Book# 10005 removed.
**List of books in the library.
Book#10007::Design Patterns::5/30/1996::is available.
Book#10015::Doing Agile Right::5/30/1996::is available.
Book#10011::Refactoring::1/31/2005::is available.
Book#10003::Artificial Intelligence::12/1/2005::is available.
Book#10008::Data Science::2/29/2008::is available.
Book#10016::Data Structure with Java::2/29/2008::is available.
Book#10014::eXtreme Programming::2/10/2010::is available.
Book#10013::Database Management Systems::2/28/2010::is available.
Book#10001::Programming in Java::3/20/2010::is available.
Book#10009::Programming in C++::3/20/2011::is available.
Book#10002::Software Methodology::4/1/2018::is available.
Book#10010::Object Oriented Analysis and Design::4/30/2018::is available.
Book#10006::Pro Git::4/30/2018::is available.
Book#10004::Linear Algebra::1/31/2020::is available.
**End of list
Invalid Date!
Invalid Date!
Invalid Date!
Invalid Date!
Invalid Date!
Invalid Date!
Invalid Date!
Invalid Date!
Invalid Date!
Kiosk session ended.
```