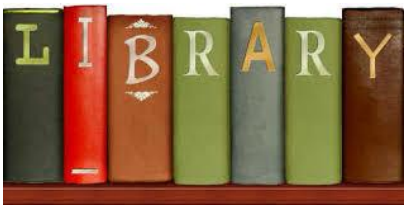
Programming Assignment #3 Library Information System

Information Systems allow us to store, manage and manipulate information. In this assignment we will create a library information system that will allow us to manage various books in a library.



In this assignment you must demonstrate your knowledge of the topics covered in Object Oriented Programming by making use of Interfaces, Classes, Abstract classes, Collections/Maps or other topics that may be suitable to complete the assignment.

The Library Information System is used to store **Books** also called **LibraryItems** which may be loaned to **Students**. The Library may store two (2) types of Books/LibraryItems, these are **GeneralWorks** and **ReservedWorks**. GeneralWorks are books in the general section of the Library. These books are considered ***Loanable*** and may be loaned to students so he can take them home. ReservedWorks are books in the reserved section of the Library and are considered ***Reservable***. These books **cannot be loaned** and **cannot be taken home**, they can only be used while in the library.

To create our Library information System we will do the following.

1. Create an **interface** named **Loanable** that will be implemented by all Loanable Books/LibraryItems. The Loanable interface should contain one method.
    1. **loanItem()** this method should accept no parameter. This method should return a boolean.

2. Create an **interface** named **Reservable** that will be implemented by all Reservable Books/LibraryItems. The Reservable interface should contain one method.
    1. **reserveItem()** this method should accept no parameter. This method should return a boolean.

3. Create an **abstract class** named **LibraryItem.** This will represent the base class of all LibraryItems. This class should have the following.
    1. Implements the Comparable interface.
    2. Four (4) instance variables, **String id, String title, String author and int availableUnits.**
    3. A Constructor that sets **all** the instance variables.
    4. The following accessors and mutators, **getAvailableUnits(), setAvailableUnits(), getTitle(), getAuthor(), getId()**. {you must use the appropriate parameters if needed}
    5. A **toString()** method that returns a string containing the Title, Id, Author and the Units Available.

```
Title: One Day in the Life of Ivan Denisovitch
ID: 4
Author: Alexander Solzhenitsyn
Units Available: 4
-----------------------------
```

    6. An **equals()** method. LibraryItems are equal if the **id** is equal.
    7. A **compareTo(Object obj)** method. LibraryItems are compared using their **id**.
    8. An abstract method named **returnItem()** this method does **not** accept a parameter and returns a boolean.

4. Create a **concrete class** named **GeneralWorks.** This class must extend LibraryItem and implement Loanable. The class should have the following.
    1. A Constructor that accepts **id, title, and author** as parameters. A GeneralWorks item is always created with a quantity of **5 available units**.
    2. It should implement the methods **loanItem()** and **returnItem()**.
       loanItem() method should
        - first check if the availableUnits > 0
        - if availableUnits > 0 then decrement the availableUnits and return **true**. Otherwise return **false**.

       returnItem() method should

        - first check if the availableUnits < 5
        - if availableUnits < 5 then increment the availableUnits and return true. Otherwise return false.

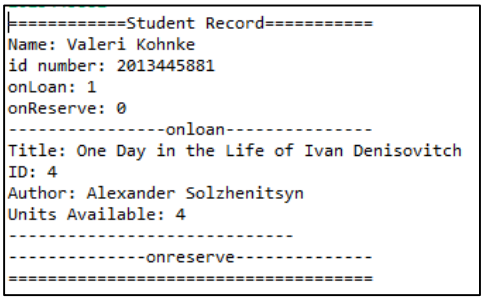
5. Create a **concrete class** named **ReservedWorks**. This class must extend LibraryItem and implement Reservable. The class should have the following.
    1. A Constructor that accepts **id, title, and author** as parameters. A ReservedWorks item is always created with a quantity of **1 available unit**.
    2. It should implement the methods **reserveItem()** and **returnItem()**.
       reserveItem() method should
        - first check if the availableUnits > 0
        - if availableUnits > 0 then decrement the availableUnits and return **true**. Otherwise return **false**.

       returnItem() method should

        - first check if the availableUnits = 0
        - if availableUnits = 0 then increment the availableUnits and return true. Otherwise return false.

6. Create **concrete class** named **Student**. This class should have the following
    1. Implements the Comparable interface.
    2. Instance variables **String name, String id, int numLoans, int numReserves**. There is also two (2) Collections/Maps named **onLoan** and **onReserve**. One contains Loanable items and the other contains Reservable items. {you must decide which structure is best for you to use}
    3. A Constructor that accepts the **name** and **id** of the student as parameters. When a student object is created the **numLoans** and the **numReserves** are **0**. Also the Collections/Maps should be created as empty.
    4. Accessors **getName(), getIdNumber(), getNumReserve() and getNumLoan().**
    5. An **equals()** method. Student objects are equal if their **idNumber** are equal.
    6. A **compareTo(Object obj)** method. Student objects are compared using their **idNumber**.

7. A **toString()** method that returns a string containing the **name, id, number of loans, number of reserves,** list of **items on Loan** and a list of **items on reserve**.



8. A method named **acquireItem()** that can be used to loan or reserve a Book/LibraryItem. This method accepts a **LibraryItem** object as a parameter and returns a **boolean**. The method returns true if the user can successfully loan or reserve a book.
    i. **A student can only have one (1) copy of a specific book. He cannot have two (2) copies of the same book.** If the Book/LibraryItem is already loaned or reserved by the student he cannot loan or reserve another copy of the same book.
    ii. The **loanItem()** method or **reserveItem()** method of the Book/LibraryItem must return true in order for **acquireItem()** to be successful.
    iii. The Book/LibraryItem must be added to the student's **onLoan** or **onReserve** Collection/Map. {this depends on the type of the book (Loanable or Reservable)}
    iv. The number books loaned or reserved by the student must be incremented. (numLoan or numReserve)

9. A method named **releaseItem()** that is used when a student releases a Book/LibraryItem that was loaned or reserved. This method accepts a **LibraryItem** object as a parameter and returns a **boolean**. The method returns true if the user can successfully release a book.
    i. **A student can only release a copy of a Book/LibraryItem that he has loaned or reserved.** If the student does not have that particular book then he cannot release it.
    ii. The **returnItem()** method of the Book/LibraryItem must return true in order for **releaseItem** to be successful.
    iii. The Book/LibraryItem must be removed from the student's **onLoan** or **onReserve** Collection/Map.
    iv. The number books loaned or reserved by the student must be decremented. (numLoan or numReserve)

7. Create the class **LibraryInfoSystem**. This class should contain two (2) Collections/Maps to store the Students and Books/LibraryItems. {you must decide which structure is best for you to use} This class should contain a methods **init()** that will be used to load information into the Information system by reading from two text files. A portion of the init() method is provided. You must complete this method.

   To test your Library management system a **main** method is needed. Create a main method that displays a menu system using the guidelines provided in the snippet bellow.

```
public class LibraryInfoSystem {


  private static final String BOOKSINPUTFILE = "Path to booksinputfile"
  private static final String STUDENTSINPUTFILE = "Path to studentsinputfile"

  private "Use an appropriate collection to hold the Books"
  private "Use an appropriate collection to hold the Students"

  private void init(){
        //load books
        try{
              //create a scanner to read from the file
              // read the library items and create the appropriate object
              //add the library item to the Collection/Map.

        }catch(/* catch an appropriate exception*/){
              System.out.println(e.getMessage());
        }
        //load Students
        try{
              //create a scanner to read from the file
              // read the student records and create the student object
              //add the student object to the Collection/Map.

        } catch(/* catch an appropriate exception*/){
              System.out.println(e.getMessage());
        }
  }



  public static void main(String[] args) {
        // create a main menu
        // (1) Borrow an Item:
        // (2) Return an Item:
        // (3) Search for Book
        // (4) Search for Student
        // (5) Exit the program
        The menu system should run continuously until the user selects option 5
        The menu system should handle the user entering incorrect options. {See the guidelines}

  }
}
```

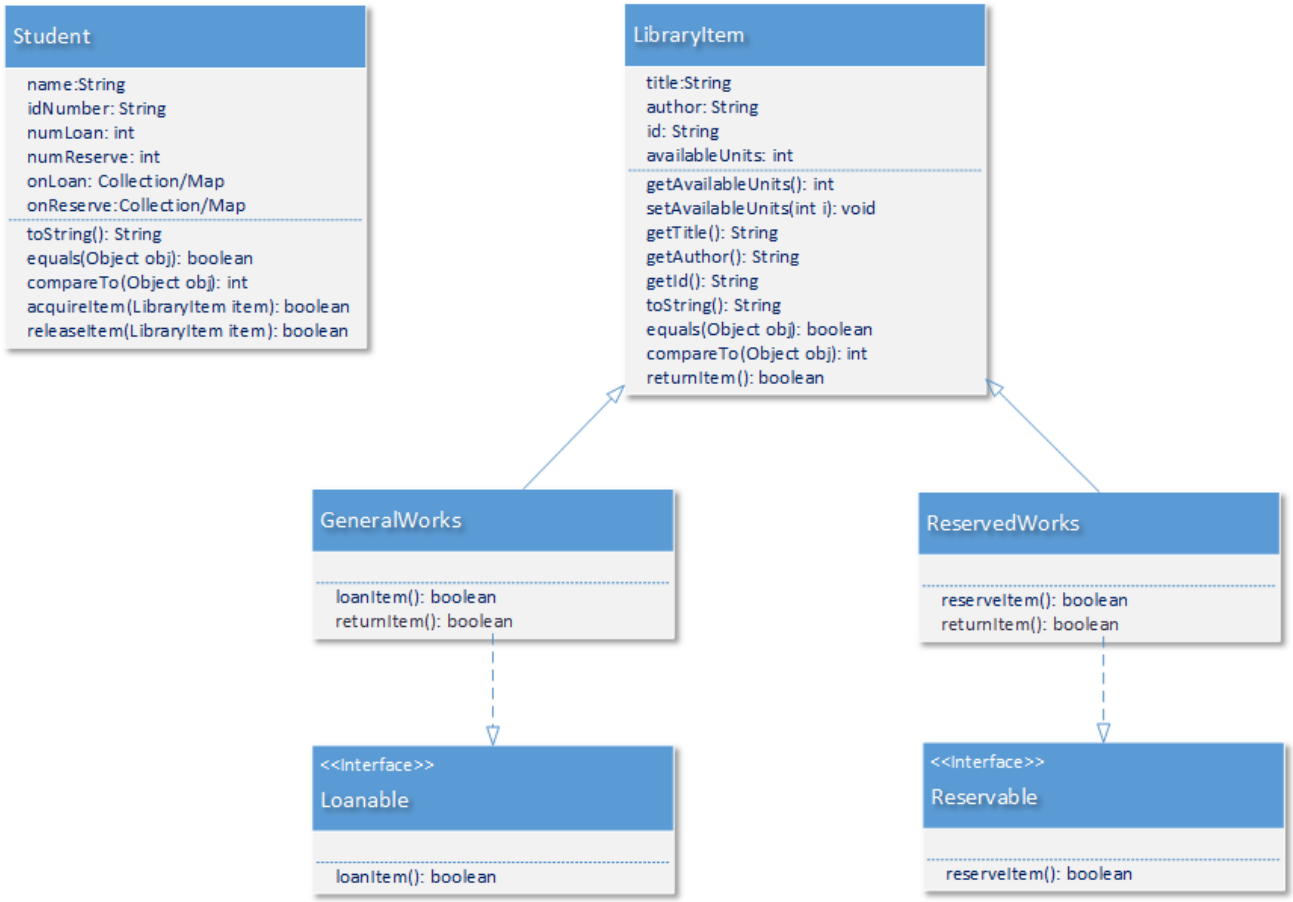
Guidelines

- The class loads the students and books/LibraryItems from two input files that are provided.
- The input files containing the student records and the book records are delimited using '**#**' character.
- The format of the bookinputfile is **idNumber#title#author#type#**
- The type may be either
    - G: GeneralWork
    - R: ReserveWork

  For example: **1#My experiments with Truth#Mahatma M.K.Gandhi#G#**

- The format of the student inputfile is **name#idNumber#**
- To **borrow** i.e. (loan or reserve) a book you must ask the user to enter the id number of the student and the id number of the book.
- To **return/release** a book you must ask the user to enter the id number of the student and the id number of the book.
- To **Search for a book** you must ask the user to enter the Id number of the book.
- To **Search for a Student** you must ask the user to enter the id number of the student.
- **You may implement other Interfaces such as Comparable if you need to.**
- **Your program should handle the situations where the user enters incorrect information. E.g.**
    - **If the user does not exist then he cannot loan or return a book**
    - **If the book does not exist then it cannot be loaned or returned.**

Class Structure of the application



Sample output

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
```

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
1
Please enter Student's ID:
2013445881
Please enter the id of the item to borrow:
4
Book successfully loaned.
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
```

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
3
Please enter the item's id number:
4
Title: One Day in the Life of Ivan Denisovitch
ID: 4
Author: Alexander Solzhenitsyn
Units Available: 4
-----------------------------

*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
```

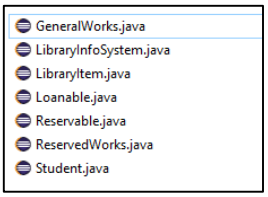
```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
4
Please enter the student's id number:
2013445881
==========Student Record==========
Name: Valeri Kohnke
id number: 2013445881
onLoan: 1
onReserve: 0
---------------onloan---------------
Title: One Day in the Life of Ivan Denisovitch
ID: 4
Author: Alexander Solzhenitsyn
Units Available: 4
-----------------------------
-------------onreserve-------------
====================================

*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
```

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
2
Please enter Student's ID:
2013445881
Please enter the id of the item to return:
4
Book successfully returned.
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
```

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
2
Please enter Student's ID:
2013445881
Please enter the id of the item to return:
4
Unable to returned.
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
```

Submission:

You have to submit the source code and the documentation file compressed into a **zip** file.



Written Documentation

In the documentation you should include a description of the implementation methodology as well as an explanation of the program design and structure. You should explain why you decided to use specific features to solve the program. **Also you have to include print screen or screen shot or snapshot of your program's output in the document.**

- Run the program by completing the test class given.
- Capture screenshots of the results for the operations of
  - Borrowing a book
  - Returning a book
  - Searching for a student
  - Searching for a book

Online Submission

Submit your assignment via the course on canvas.instructure.com. All students should have already been enrolled onto Canvas and be given access to the course.

Select **Assignment #3** from the Assignments menu and use the button to submit the assignment.

The compressed file should have the name "**ClassNo-OOP-StudentIdNumber-Surname**".

For example: **1-OOP-2008012409-kim**

2-OOP-2009003987-Bae

Hard-copy submission:

You **must** submit a hard-copy (printout) of Program, output and documentation no later than:

Monday (월요일) 20/06/2016. 17:00 오후 (5:00 PM)

Please bring the Hard-copy (printout) to IT/BT Room 402-2

Scoring Criteria:

- You will get 100 points if the program satisfies all the requested features, runs smoothly and handles incorrect choices and you properly described your program in the hard-copy document.
- Note: **50 Points Online Submission** and **50 Points Hard-Copy**.
- For missing features (e.g. not repeatedly asking for input or not handling users or books that don't exist) points will be deducted.
- This assignment does not carry an extension period. Submissions later than the deadline **will not** be accepted.

NOTE: This is an individual assignment. Copying other student's work is strictly prohibited!

Assistance and Explanations:

If you need assistance or more explanations you can visit me at

- IT/BT Room 402-2
- Mondays and Wednesdays from 2pm – 4pm.