 Xi’AN JIAOTONG-LIVERPOOL UNIVERSITY

西 交 利 物 浦 大 学

COURSEWORK SUBMISSION COVER SHEET

Name (Surname)Li (Other Names) Zhi

Student Number 1718663

Programme Information and Computing Science

Module Title Data Structure

Module Code CSE104

Assignment Title Assignment 2

Submission Deadline Friday, May 24th, 2019

Module Leader Steven.Guan

By uploading or submitting this coursework submission cover sheet, I certify the following:

· I have read and understood the definitions of PLAGIARISM, COLLUSION, and the

FABRICATION Of DATA, as outlined in the Undergraduate Student Handbook of

Xi’an Jiaotong-Liverpool University and as posted on the University Website.

· This work is my own, original work produced specifically for this assignment. It

does not misrepresent the work of another person or institution as my own.

Additionally, it is a submission that has not been previously published, or

submitted to another module.

· This work is not the product of unauthorized collaboration between myself and

others.

· This work is free of embellished or fabricated data.

I understand that PLAGIARISM, COLLUSION and the FABRICATION OF DATA are serious

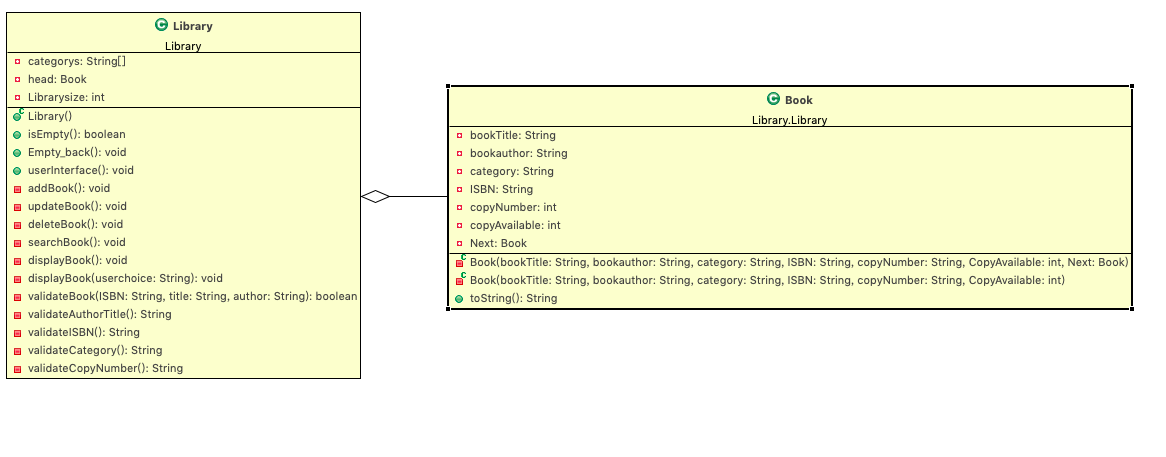
disciplinary offences. By uploading or submitting this cover sheet, I acknowledge that I am

subject to disciplinary action if I am found to have committed such acts.

Signature ……*.*………………Zhi.Li…………………… Date ……*………………………2019/5/19………*……………

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | For  Academic  Office use: |  | Date Received Days Late Penalty |  |

UML:

**

**pseudo code:**

public class Library{ //the whole Library is a LinkedList

int Librarysize; //used to record the node number in the Linkedlist

Book head = null; //initialized head node, which is null

private class book{ //book as a inner class is the LinkedList’s node

String bookTitle,bookauthor,category,ISBN;int copyNumber,copyAvailable;

private Book(String bookTitle,String bookauthor, String category, String ISBN,String copyNumber,int CopyAvailable, Book Next) { //constructor function

this.bookTitle=bookTitle;

this.bookauthor=bookauthor;

this.category = category;

this.ISBN = ISBN;

this.copyNumber= Integer.parseInt(copyNumber);

this.copyAvailable=CopyAvailable;

this.Next = Next;

}

}

Function addbook(){

Print(“Please input the book information……”)

Check the information which user input is illegal or not ….

Call the function validateBook(ISBN,Bookname,Bookauthor) to judge whether this book has already existed in the libaray;

If(exist) {

print(“the book has already exists”)

go back to the userInterface

}

else {

Book book = new Book(bookTitle,bookauthor,ISBN,category,copyNumber,copyAvailable,null);

book.Next = head; //make the node's next node point at the head of the LinkedList

head = book; //update the head node to the current node

Librarysize++; //maintain the length of the Linkedlist

}

}

Function displaybook {

ArrayList<Book> books = new ArrayList<Book> ();

Book cur = head ;

For (n = 0 to Librarysize){

books.add(cur);

cur = cur.Next;

}//traversal the whole linkedlist, and add each node into the ArrayList

If(user choose to sort by Name){

books.sort(new Comparator(){

int compare (Book a, Book b){

return a.bookName.compareTo(b.bookName)

}

})

}

else if(user choose to sort by category){

books.sort(new Comparator(){

int compare (Book a, Book b){

return a.category.compareTo(b.category) }

})

}

Function deleteBook, updateBook,searchBook{

Accept ISBN or Bookname+bookauthor

Book cur =head;

If(deleteBook){

For(n = 0 to Librarysize){

Judge whether (cur.bookName + cur.bookAuthor) equals (userinput) or (cur.ISBN equals userinput ISBN)

If(equal){

judge whether cur.avalable number == cur.Totalnumber

if(equal){

head = cur.Next;

cur = null;

Librarysize--;//after delete a book maintain the library size

}

else break;

}

else print(“Books not found”)

}

}

else If(updatebook){

For(n = 0 to Librarysize){

Judge whether (cur.bookName) or (cur.bookAuthor) or cur.category equals (userinput)

If(equal){

Update the information input by user

}

else print(“Books not found”)

}

}

Else if (searchBook){

ArrayList books = new ArrayList();

For(n = 0 to Librarysize){

Judge whether (cur.bookName) or (cur.bookAuthor) or cur.category equals (userinput)

If(equal) books.add(cur)

else print(“Books not found”)

}

books.sort(new Comparator(){

int compare (Book a, Book b){

return a.copyNumber – b.copyNumber

}

})

}

}

Complexity analyze: (ignored all the validate process and check process)

AddBook O(1): Add process only contains creating a new node, and make the node’s Next point to the current head node , the whole process does not involve traversal, thus it is O(1)

DisplayBook O(n^2logn): O(n) for traversal O(nlogn) for it’s sort process, since ArrayList.sort’s algorithm is Timsort. Timsort’s Average time efficiency is O(nlogn); thus the total is O(n^2logn);

SearchBook O(n^3logn): O(n^2) for traversal and compare each name or ISBN information with current node. O(nlogn) for it’s sort process:Timsort;

UpdateBook O(n^2): for traversal and compare each ISBN information process, and the update process is only O(1), thus total is O(n^2);

DeleteBook O(n^2): for traversal and compare each name+author or ISBN information process, and the update process is only O(1), thus total is O(n^2);

Test picture: since page limitation, only put one picture of adding book here

