



Library Management System

21.03.2019

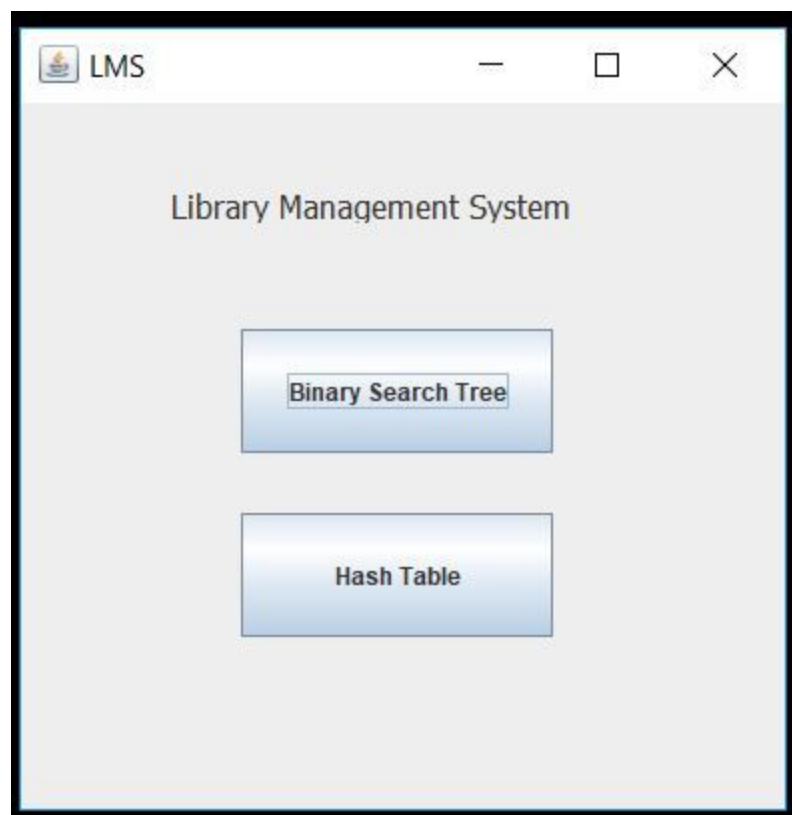
Nikit Vaswani
A00258753

Introduction

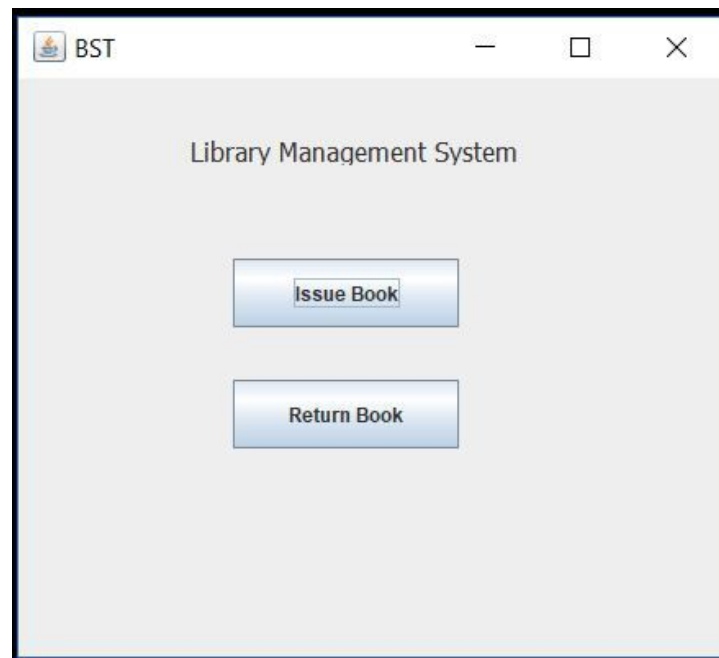
Library Management System will be using the concept of both the Hash Table and Binary Search Tree. The user has an option to choose which one they want to use. This application will make it easier to main the library which will allow the user to issue and return books. It will save important information at the time of issuing the book like student ID, Book ID(Key) , Book name and Date of issue. It will be using the search feature at the time of returning the book it will also tell if the user has to pay some fine in case of late return of the book.

Classes:-

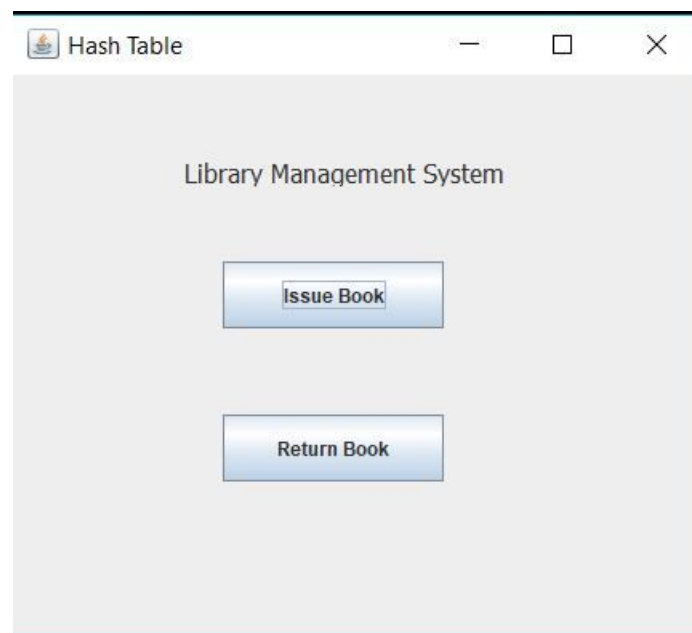
1. **Home-** This class is the user interface of the home page of the application and is from gui package.



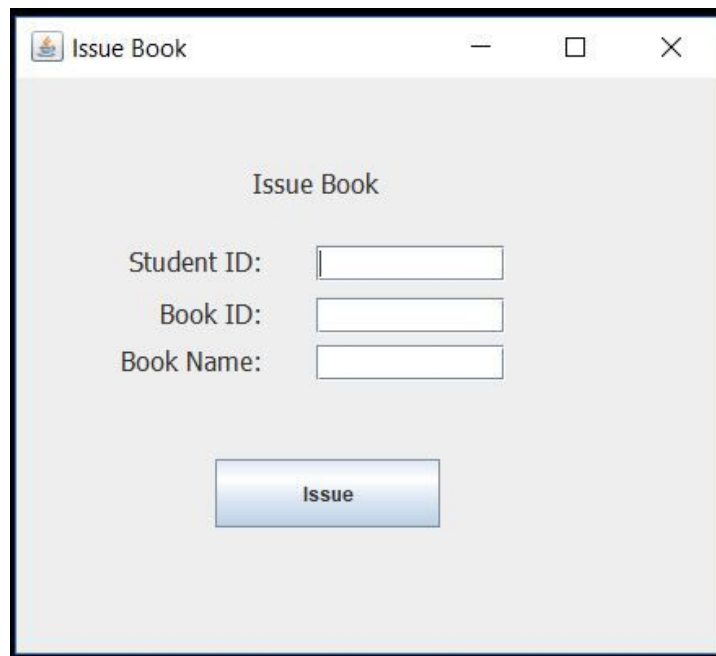
2. **Bst** - This class is also inside the gui package and will appear on clicking Binary Search Tree button on the home page.



3. **HashTable**- This class is also inside the gui package and will appear on clicking Hash Table button on the home page.

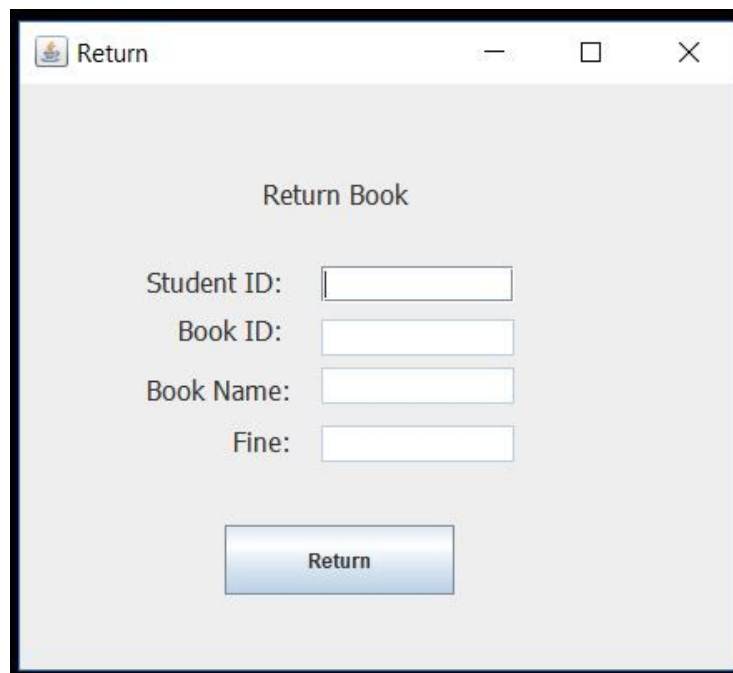


4. **Issue-** This class is inside gui package and will be used when the user wants to issue the book.



The screenshot shows a Java Swing window titled "Issue Book". The window has a light gray background and a title bar with standard Windows controls (minimize, maximize, close). The main content area is titled "Issue Book" in a bold, black font. Below the title, there are three text labels: "Student ID:", "Book ID:", and "Book Name:". Each label is followed by a white text input field with a thin gray border. At the bottom center of the window, there is a blue button with the word "Issue" in white text.

5. **Return-** This class is inside gui package and will be used when the user wants to return the book.



The screenshot shows a Java Swing window titled "Return". The window has a light gray background and a title bar with standard Windows controls (minimize, maximize, close). The main content area is titled "Return Book" in a bold, black font. Below the title, there are four text labels: "Student ID:", "Book ID:", "Book Name:", and "Fine:". Each label is followed by a white text input field with a thin gray border. At the bottom center of the window, there is a blue button with the word "Return" in white text.

Limitations:-

The Biggest Limitation being time, as I am inserting the date at the time of insertion. This application only remains on the server side, database can be implemented to store the book details.

Scope:-

The scope of the project is limited to company side. No client can access the platform to issue or return the book.

Functionalities:-

These above classes clearly depict how the user interface will look like. The **Home** class also shows that there will be two different data structures (Binary Search Tree and Hash Tables) used for **insertion** and **searching**, which are the two main core functionalities. Other than that their will be many other functionalities like:-

- Generating fine which will be calculating if the book has been issued for more than 7 days there will be a fine of €1/day.
- Checking if there is already a book issued by that student ID while issuing the book.
- Iterator to for searching the books.
- Binary Search Tree and Hash Table data structures to insert and search data.
- Inheritance of node class.
- The same class issue and return are used for implementing Binary Search Tree and Hash Table using complex GUI features.

Code Snippets:-

Iterator

```
package gui;

import java.util.Stack;

public class BSTIterator {
    Stack<Node> stack;

    public BSTIterator(Node root) {
        stack = new Stack<Node>();
        while (root != null) {
            stack.push(root);
            root = root.left;
        }
    }

    public boolean hasNext() {
        return !stack.isEmpty();
    }

    public int next() {
        Node node = stack.pop();
        int result = 0;
        if (node != null) {
            result = node.key;
        }
        if (node.right != null) {
            node = node.right;
            while (node != null) {
                stack.push(node);
                node = node.left;
            }
        }
        return result;
    }
}
```

```
@Override
public boolean hasNext() {
    // TODO Auto-generated method stub
    return index < head.length;
}

@Override
public Object next() {
    // TODO Auto-generated method stub
    HNode res = head[index++];
    return res;
}

@Override
public HtabIterator iterator() {
    // TODO Auto-generated method stub
    return new HashTable();
}

public static HashTable getInstance() {
    if(!created) {
        single = new HashTable();
    }
    return single;
}
}
```

Generate Fine

```
public String cost(HNode x) throws ParseException {
    SimpleDateFormat sdf = new SimpleDateFormat("dd-M-yyyy hh:mm:ss");
    String p = sdf.format(new Date());
    Date d2 = sdf.parse(p);
    int t = (int) ChronoUnit.DAYS.between(d2.toInstant(), x.d.toInstant());
    System.out.println(t);
    if(t>0) {
        t= t-7;
        return "€"+t;
    }
    else {
        return "No fine before Time";
    }
}
```