**COP 5536 Fall 2023 Programming Project**

**REPORT**

**UF ID: 12427601**

**EMAIL: sa.ekbote@ufl.edu**

**NAME: Sai Aakash Ekbote**

Make-File for the Project



Fig 1

Step1: run “make” command in the linux terminal

Step2: run “make run1” command in the linux terminal similarly run “make run2” till “make run7”

Step3: run “make clean” to clean up the .class executable files (run this command before running step2)

Project Structure:

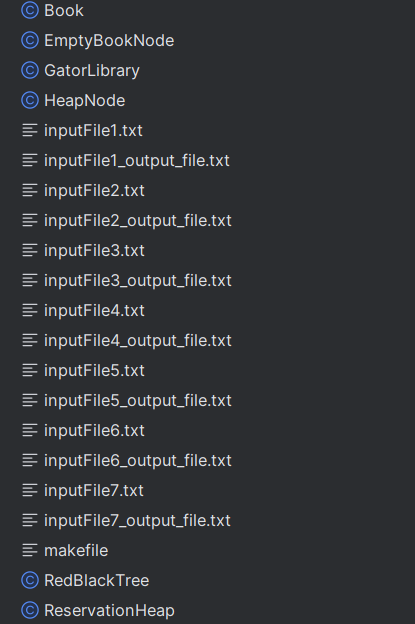


Fig 2

Main class is gatorLibrary.java

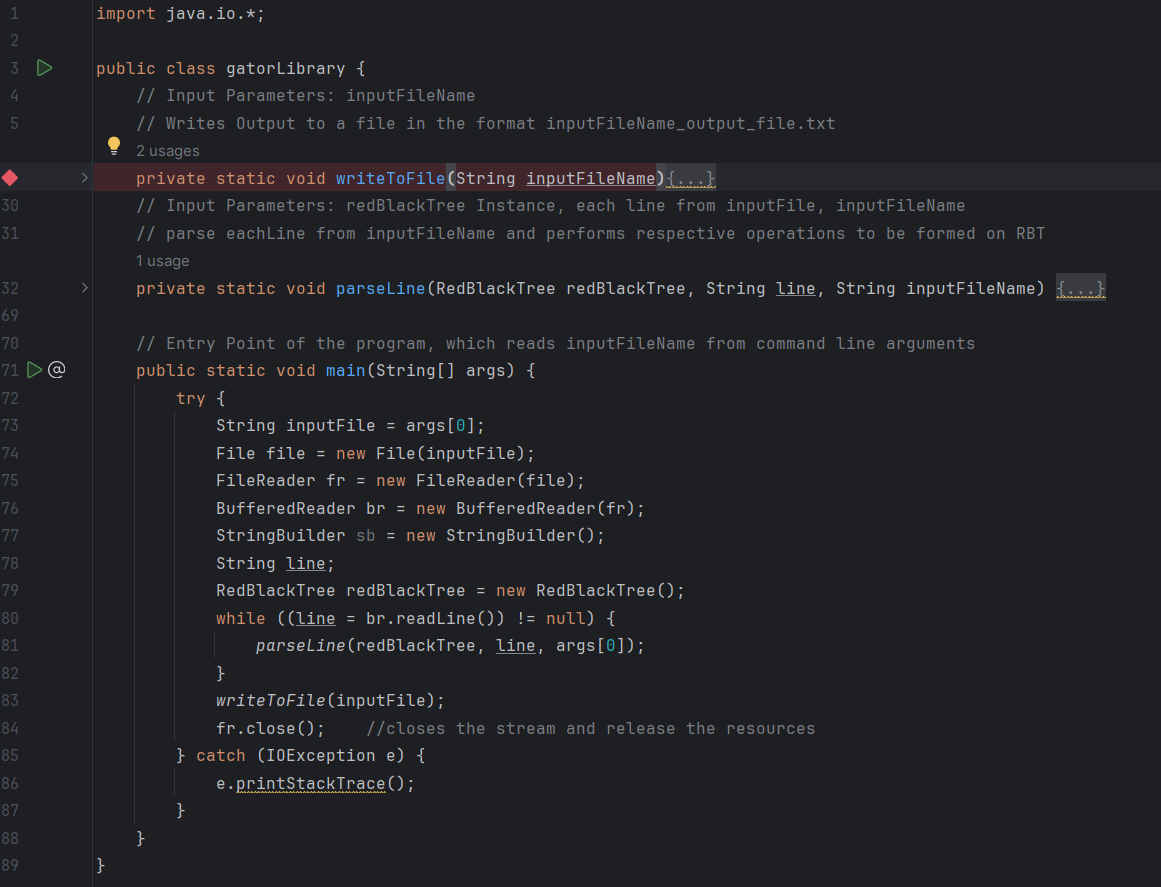


Fig3

* Main method reads inputFile from command line arguments and calls parseLine function which parses each line from inputFile and executes Respective operation
* We call writeToFile when we reached endOfInput or encountered Quit() operation

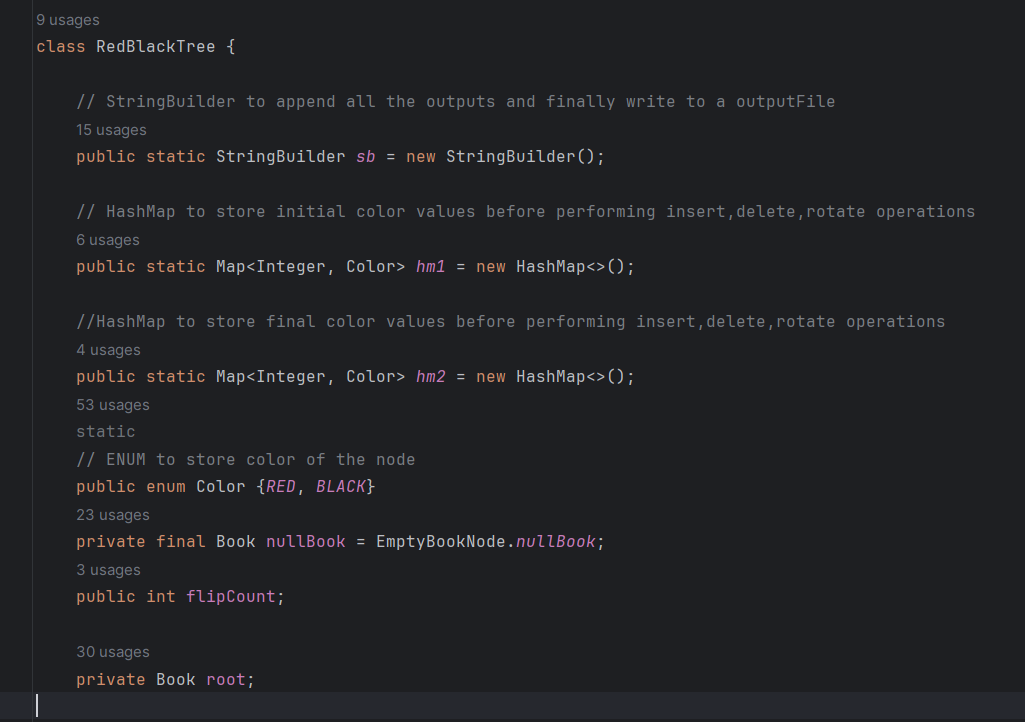


Fig4

The above image contains structure of RedBlackTree Node



Fig5: Book structure

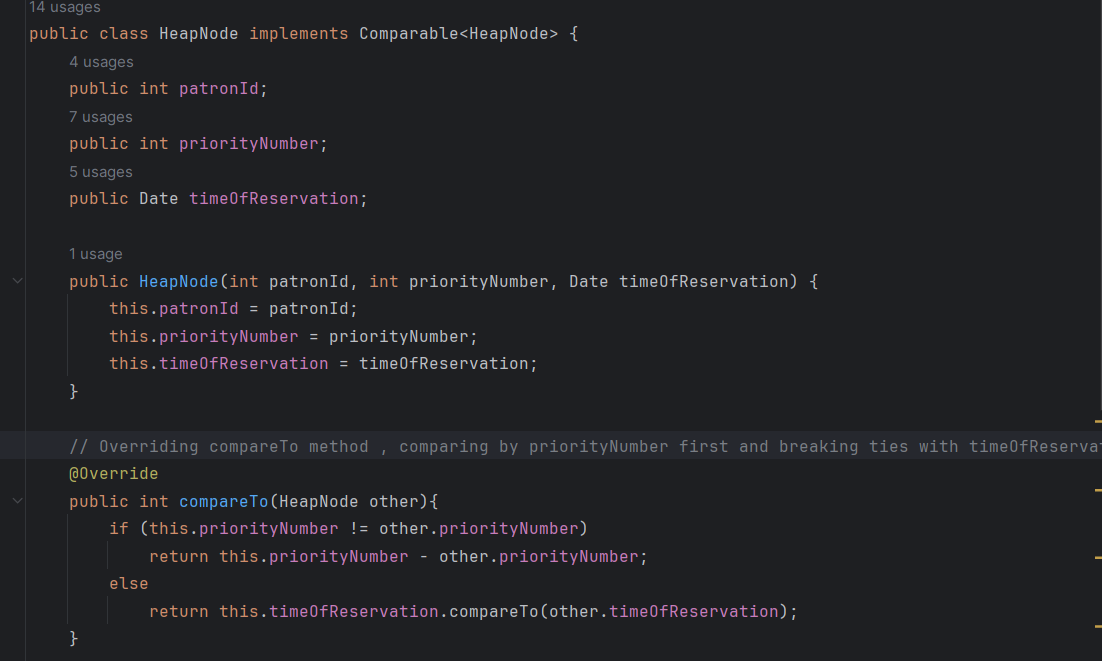


Fig6: HeapNode structure

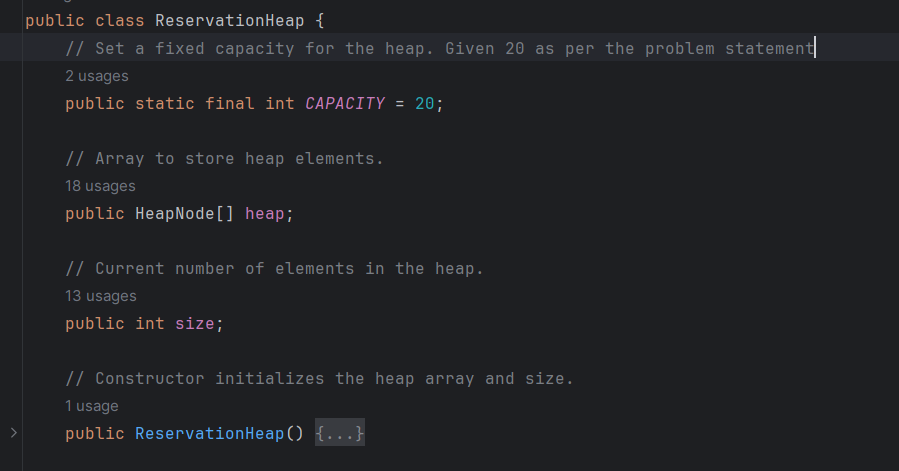


Fig7: Reservation Heap Structure

**Function Prototype:**

**insertBook():**

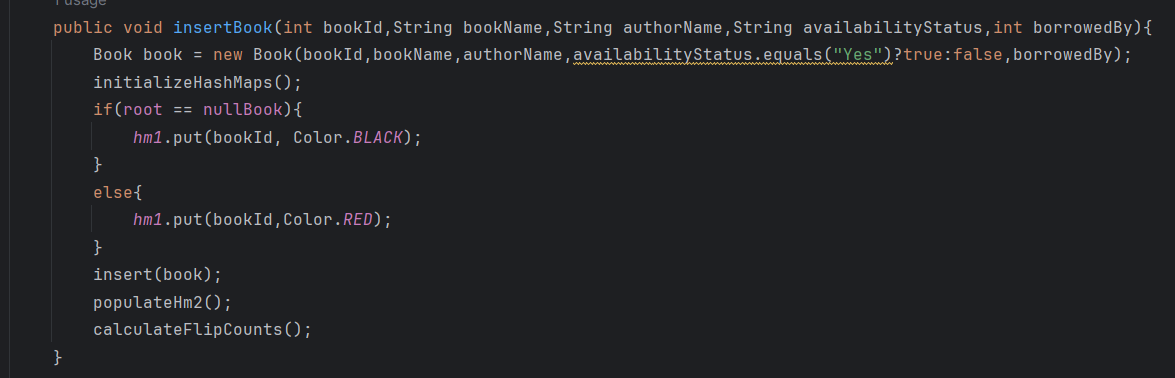


Fig8: InsertBook() structure

We pass borrowedBy as “-1” for initial case and Create a new Book to be inserted in the red black tree and pass book as argument to insert(book) .

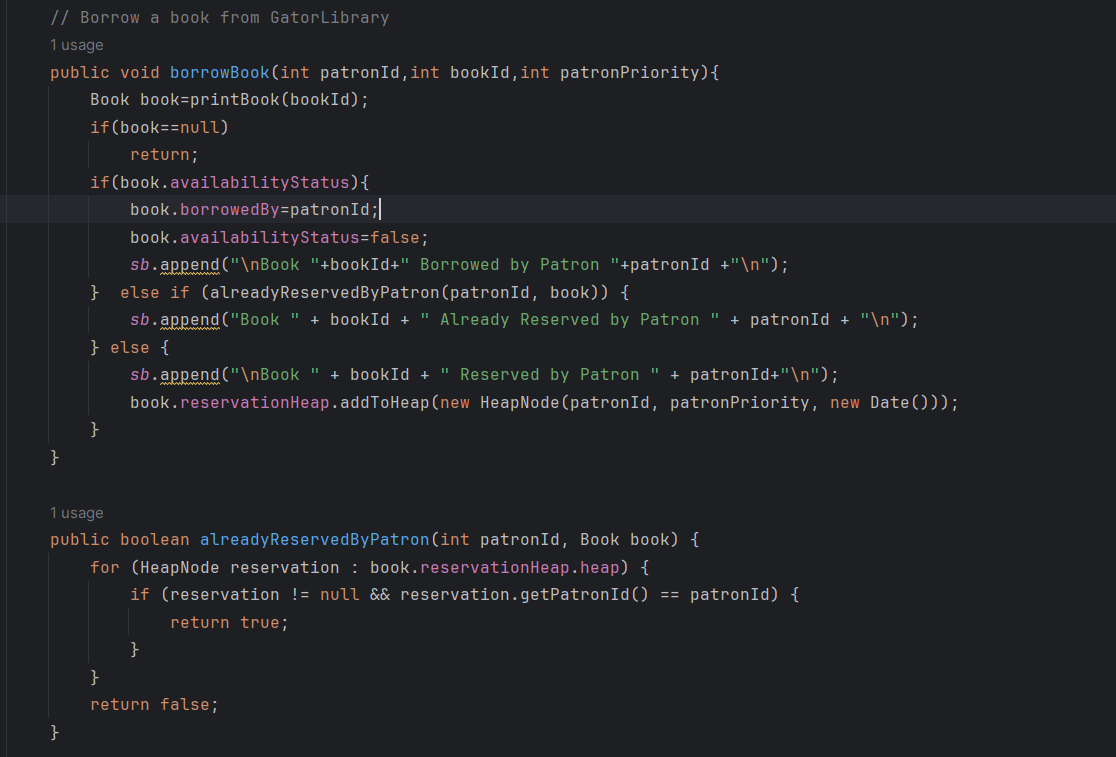
**borrowBook():**

Fig9: borrowBook() structure

* We do inorder search in the redblack tree to fetch book by patronId , and then check availability status for that book
* If book is available then we allot the book to patron who requested the borrow
* if book is not available we add the patron to the reservation heap, with the help of patronID, patronPriority and Current TimeofReservation

**DeleteBook():**

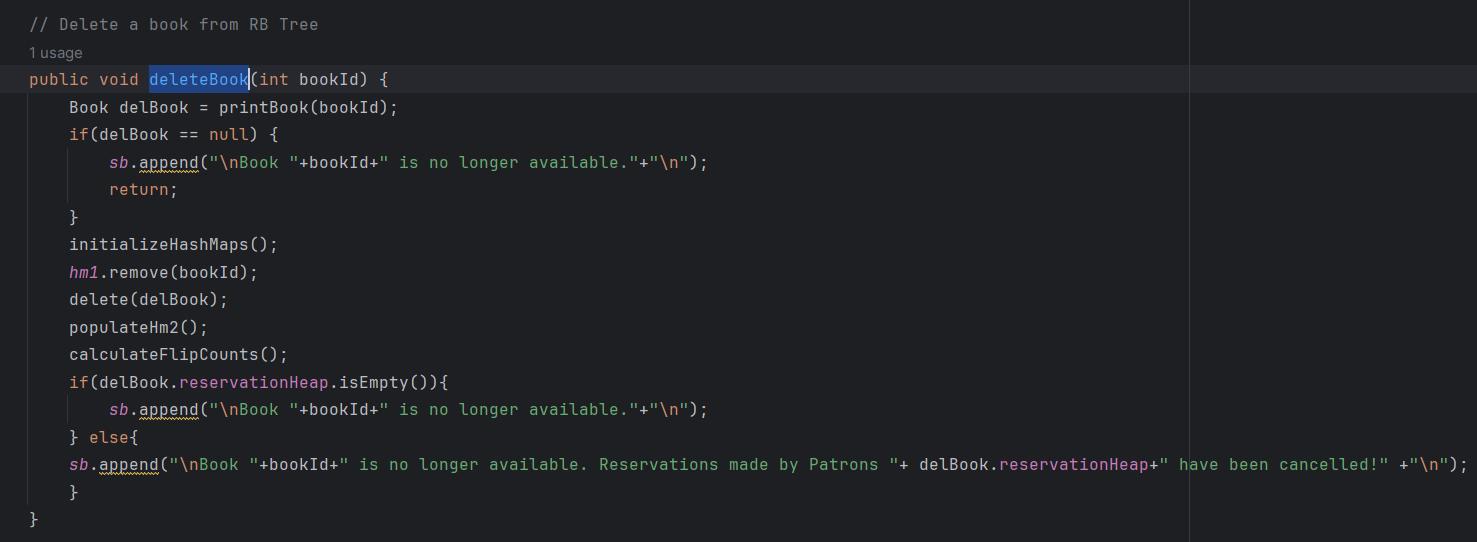


Fig10: deleteBook() structure

* we first do inorder search in the redblack tree to find if the bookId requested to be deleted is present or not,
* if present we delete from redblack Tree calling delete(book) and also compute flipCounts
* if not present we display appropriate message

**findClosestBook()**

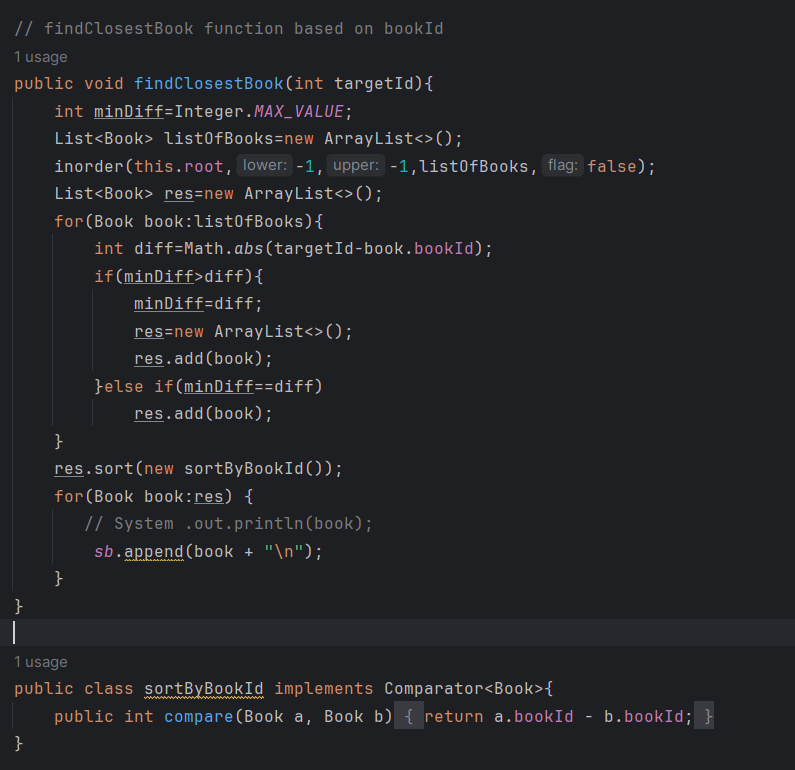


Fig11: findClosestBook() structure

* we do inorder Traversal and get all the books in an arraylist
* we calculate the difference between targetBookId and for each bookId from inorder result and compute closestBooks.
* We then sort the result based on bookId

**PrintBook()**

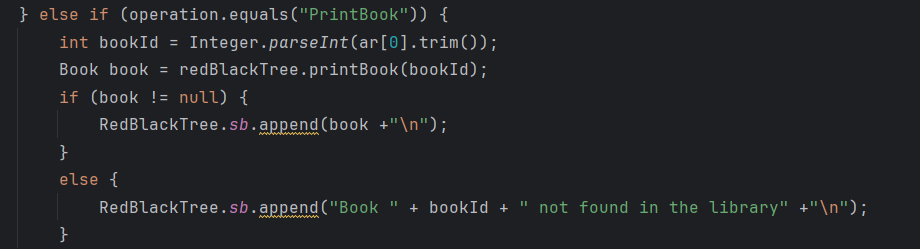
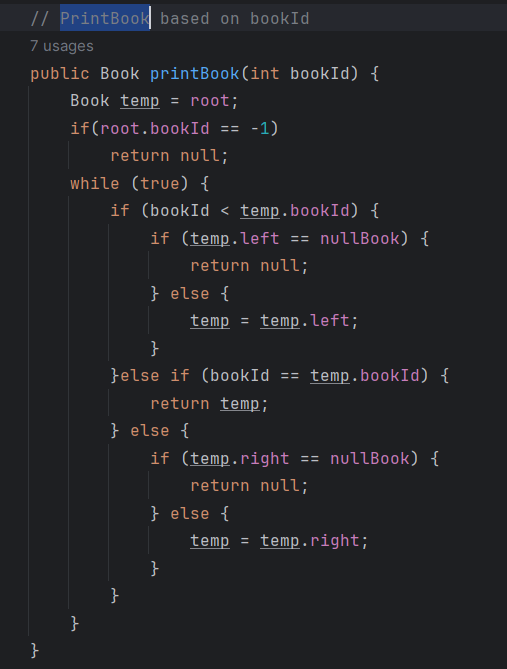
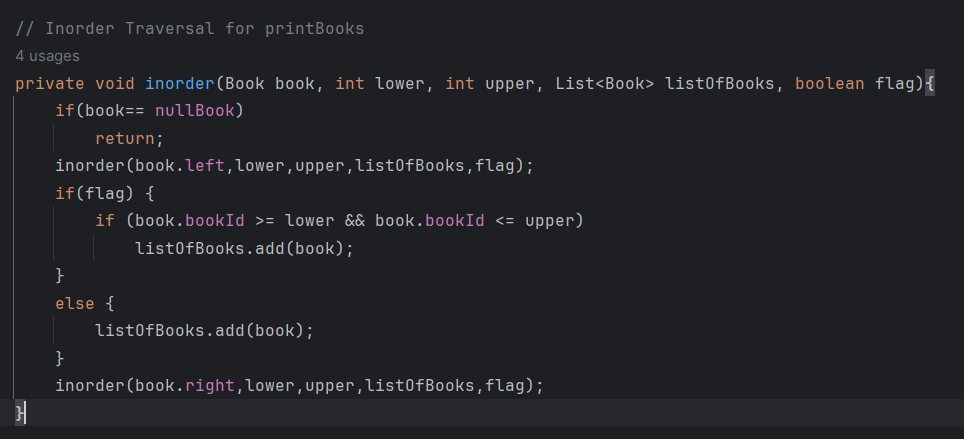
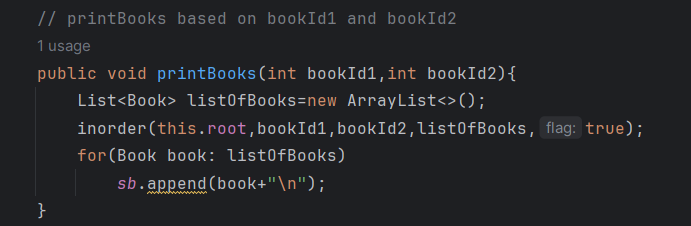


Fig 12: printBook() structure

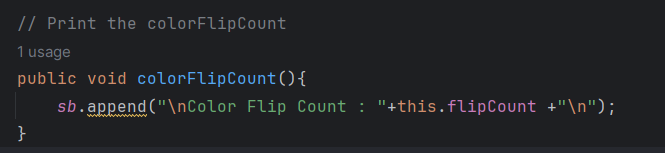
* We do inorder traversal on RBT based on bookId, and return that book if present or null
* Based on value returned we display appropriate message

**printBooks():**

 Fig 13: printBooks() structure

* We do inorder traversal on the RBT based on the bookId1 and bookId2 received as parameter from inputFile,
* We set lower as bookId1 and upper as bookId2 and flag to true
* If current node’s bookId is in the range , then we add to result arraylist
* We finally print the arraylist

**colorFlipCount()**

 Fig 14: colorFlipCount() structure

Explanation provided at the end

**ReturnBook()**



Fig14: ReturnBook() structure

* We do inorder search in the redblack tree and fetch the book associated with the book id
* We set borrowedBy parameter to -1 and set availability status of the book to true and print appropriate message
* We then, check if reservation heap for that book is empty or not,
* If it is Empty , we exit from the function
* Else, we allote the book to reserved Patron from the reservationHeap, and set borrowedBy to reservedPatronId and availabiltyStatus to false and print appriopriate message and exit from the function

**Quit()**

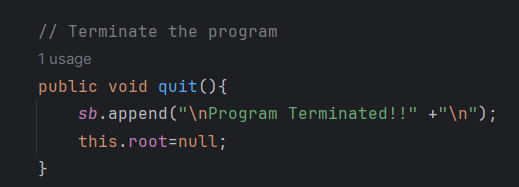
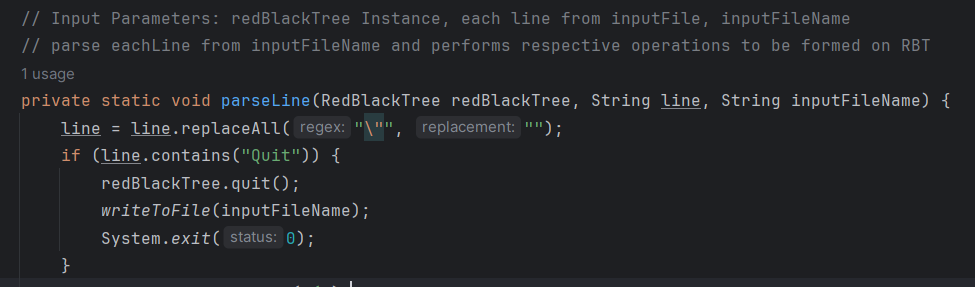
 

Fig: 15: Quit() structure

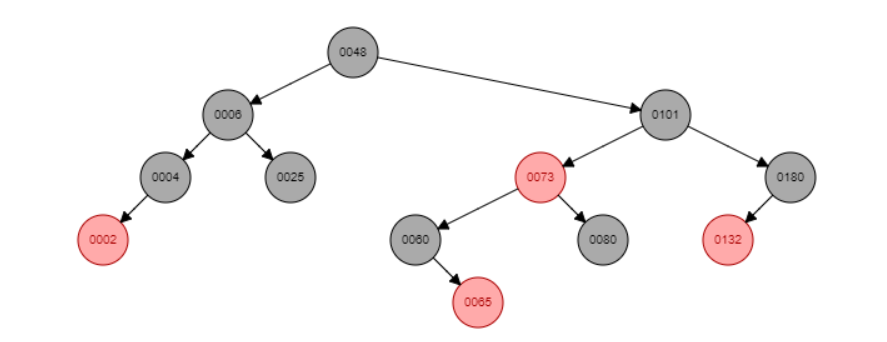
**Input and Outputs:**

All inputs and Outputs are matching with sample Input and Output provided except InputFile4.txt

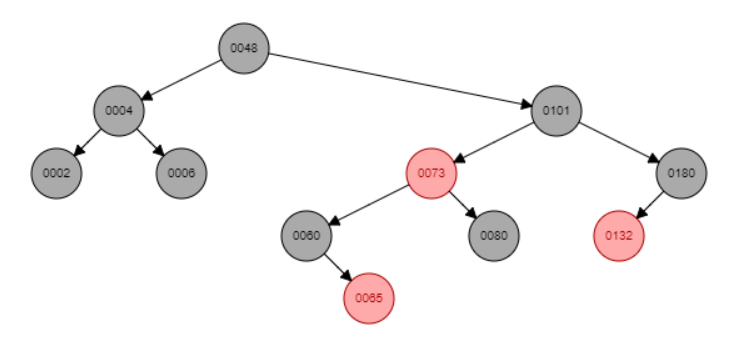
**For InputFile4.txt**

**Before DeleteBook(25) is encountered tree looks like**

**ColorFlipCount is 23**



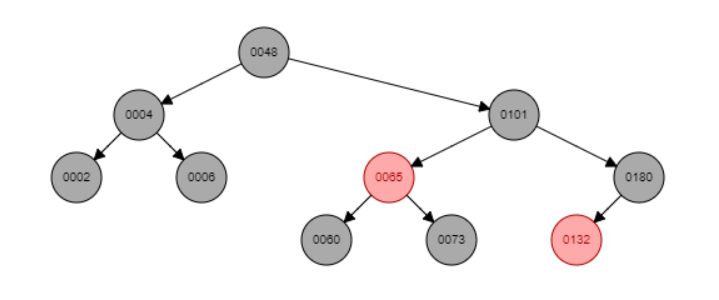
**After DeleteBook(25)**



**ColorFlipCount is 24 as red node 2 is changed to black**

**DeleteBook(80)**

**Tree Becomes:**

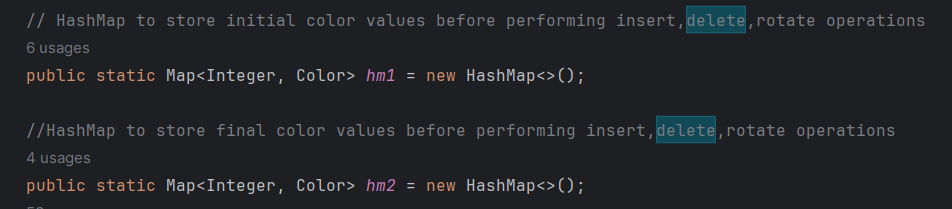


**Node 73 color is changed from red to black and rest of the color remains same**

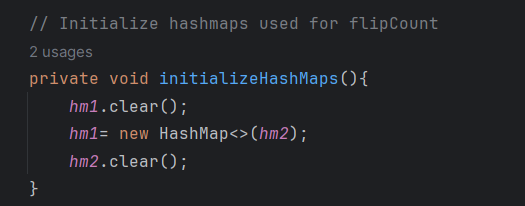
**So, ColorFlipCount is 25**

**Therefore, The final ColorFlipCount is 25 in my case instead of 27 given in the sample output.**

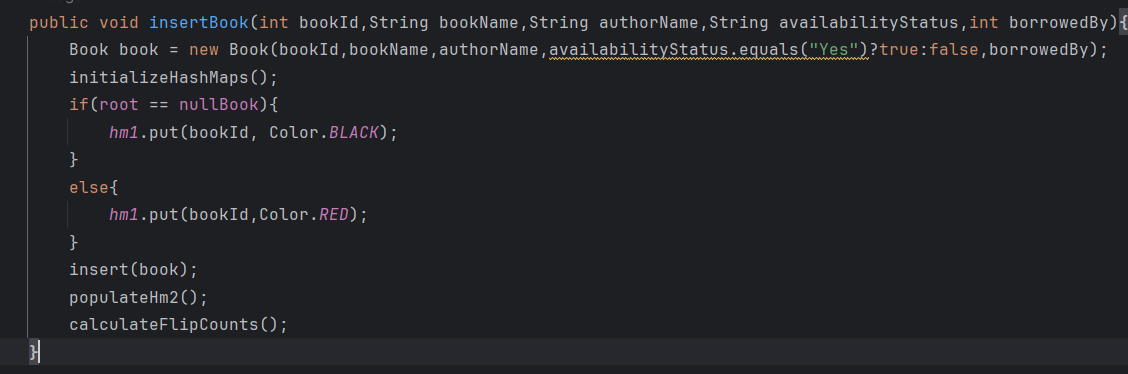
**How to calculate FlipCount():**



Initialize two hashmaps , hm1 and hm2

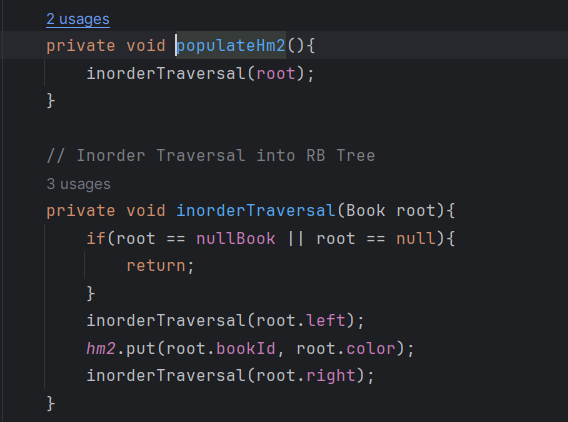


Before each insert and delete we invoke initializeHashMaps() which copies all the key value pairs from hm2 to hm1

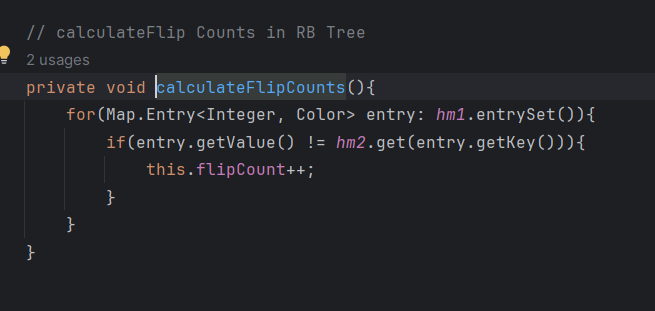


We insert bookId and Color as black if root is empty or else

We insert bookId and color as red



After Insert operation is perfomed, we perform inorder traversal on red black Tree and update hashmap with bookId and color of each Node



We then calculate the FlipCounts comparing key,value pairs from both the hashmaps.

**Sample Input Output**

Executed and tested under thunder server -> working as expected

Command1 : make

Command2: make run1

Command3(optional) : make clean (to clear executable files generated)

