Bookotron 9001

Carson White	Nevin Mahilal	Geoff Yeung
1337034	1054694	1317141

Version 1.0

Friday April 10th 2015

Group 12

CS 2XB3

Computing and Software
McMaster University

TABLE OF CONTENTS

REVISIONS

April 1, 2015	● Initial classes and GUI frame created
	  ○ Recommendation Tree (Search Tree)
	  ○ JTable
	● Input of novels created and updated

April 4, 2015	● Tested JTable implementation ● Updated JTable implementation
April 5, 2015	● Input storing method created ● Created book data structure to hold novels' information
April 9, 2015	● Added Selection Sort Algorithm ● Added Insertion Sort Algorithm ● Added Three Way String Quicksort for Title and Author ● Updated JTable implementation ● Revised Selection Sort
April 10, 2015	● Revision of Three Way String Quicksorts ● Added Comments ● Updated Three Way String Quicksorts ● Updated the GUI and JTable to include all sorts ○ Added GUI buttons for each sort ● Updated the GUI and JTable to include recommendation tree

	Carson	Nevin	Geoff
Student Numbers	1337034	1054694	1317141
Roles and Responsibilities	Programmer Project Leader Log Admin	Programmer Research/Design	Programmer Researcher

CONTRIBUTIONS

Names	Roles	Contribution	Comments
Carson White	Programmer Log Keeper Project Leader	Book class Insertion/Selection Sort classes JTableUsage class Class Descriptions for above	
Nevin Mahilal	Programmer Designer	Recommendation Class MainFrame class Class Descriptions for above	
Geoff Yeung	Programmer Researcher	Quicksort Classes Class Descriptions for above	

EXECUTIVE SUMMARY

The **Bookotron 9001** is a simple application that allows users to organize their personal book collection by title, author, length or rating. This allows them to find a specific entry or set of entries very easily.

A selection sort and Insertion sort were implemented to organize novels by rating and length respectively. These were chosen due to the small amount of overhead required to run these algorithms. Giving the program a fast response and startup time if the lists are sorted provide a small amount of overhead.

A Threeway quicksort was used to sort by Title and Author. After research we found that the most efficient and easiest implementation of a string sort was the Quicksort. Our other options were an LSD or MSD. After extensive research and testing, these string sorting algorithms were not chosen because they depend on the strings to contain the same length.

The second component to this project is a recommendation system. Through a series of questions a user can find reading material that will be curated based on their personal interests.

1. Book Class

1.1 Description:

This class is used to represent a single book. Is has properties to hold the title, author, length and rating. It also has methods to return each piece of information

1.2 Interface:

2.2.1 Uses:

- None

1.2.2 Types:

String title
String author
int length
int rating
String[] books

1.2.3 Code breakdown:

public Book(String[] books)
Takes the Title, Author, Length, and Rating and stores them for future reference

public getTitle()
Returns the title of the book

public getAuthor()
Returns the author of the book

public getLength()
Returns the length of the book

public getRating()
Returns the rating of the book

1.3 Implementation:

1.3.1 Uses:
None

1.3.2 Types:
None

1.3.3 Code Breakdown:

No private methods. Since this is a data structure no calculations occur in this class

2. Book Length Insertion Sort Class

2.1 Description:

This class is an implementation of an Insertion Sort Algorithm. It receives an Array of Book objects. It uses the getLength() method from the Book class for each Book as a basis for the sort.

2.2 Interface:

2.2.1 Uses:

- None

2.2.2 Types:

Book[] book

2.2.3 Code Breakdown:

public insertionSort(Book[] book)

Returns the sorted array of books as follows

```
for each book
        valueToSort = book[i]
        j = i
        while(book[j-1] length > valueToSort length)
                book[j] = book[j-1]
        book[j] = valuetoSort
```

2.3 Implementation:

2.3.1 Uses:

None

2.3.2 Types:

Book valueToSort

2.3.3 Code Breakdown:

No private methods

3. Book Rating Selection Sort Class

3.1 Description:

This class is an implementation of a Selection Sort Algorithm. It receives an Array of Book objects. It uses the getRating() method from the Book class for each Book as input for the sort.

3.2 Interface:

3.2.1 Uses:

- None

3.2.2 Types:

Book[] book

3.2.3 Codebreakdown:

public selectionSort(Book[] book)

Returns the sorted array of books as follows

```
for each book from i=0; i++ starting from end of array
        highest index = last book in array
        for  j=0; j< book array length ; j++
                if book[i] rating > book[highest index] rating
                        highest index = j
        temp = book[i]
        book[i] = book[highest index]
        book[highest index] = temp
```

3.3 Implementation:

3.3.1 Uses:

None

3.3.2 Types:

int highestIndex

3.3.3 Code Breakdown:

No private methods

4. Title Sorting Class

4.1 Description:

This class sorts the novels based on its title. It takes the input of novels and implements a three way quicksort to organize the titles alphabetically. This algorithm is based off the Three Way String Quicksort located in “Algorithms 4th Edition” or online at http://algs4.cs.princeton.edu/51radix/Quick3string.java.html.

4.2 Interface:

4.2.1 Uses:

- None

4.2.2 Types:

Book[] book

4.2.3 Codebreakdown:

Public static void sort(Book[] book)

Takes a collection (array of novels as input), and calls the private sort() to implement a quicksort. This method also checks if the list is already sorted prior to activating the actual sort.

4.3 Implementation:

4.3.1 Uses:

None

4.3.2 Types:

final int CUTOFF

4.3.3 Code Breakdown:

Private static int charAt(Book book, int d)

Int bookLength = the length of the novel title in the inputted novel
Assert that the character spot being compared is within bounds
Assert that the novels are sorted

Private static void sort(Book[] book, int low, int high, int d)

If the greatest valued element in the array is<= the lowest value + cutoff point:
Switch the places of the novels
Break

Int lt = temporary store of lowest value to increment
Int gt = temporary store of highest value to decrement
Int I = lowest value + 1
Int v = the unicode code point of the novel author at character “d”

While (the lowest value +1 is less than or equal to the size of the array):
Int t = Unicode code point of the character at integer d in the book on top of array
If the lowest value is less than the temporary value:
Exchange the lowest value + 1 and the lowest value + 2
Else if the lowest value is greater than the temporary value:
Exchange the novels of the lowest value + 1 and the greatest value – 1
Else:
Increment i (lowest value +1) by one

sort(call itself to sort the sub array taking away the greatest valued element by the current character)

if the Unicode code point is greater or equal to zero:
sort(call itself recursively to sort the two novels by the next character)
sort(call itself recursively to sort the temporary greatest value +1, the highest value at the next character)

private static void insertion(Book[] book, int low, int high, int d)

for the range from zero the size of the array:
for the range of the lowest number to the smallest value novel:
exchange the two novels

private static void exchange(Book[] book, int I, int j)

switches the two novels at place I and j using a novel placeholder

private static boolean less(Book x, Book y, int d)

String firstBook = the title of book x
String secondBook = the title of the book y
Assert if the two titles are equal

For the range of the place of the character and the minimum length:
If the letter searched in the first novel is less than the second novel:
Return true
If the letter compared in the first novel is greater than the second novel:
Return false

If all other cases fail, return the boolean value of the comparison of the first and second novel

Private static boolean isSorted(Book[] book)

For the range of 1 and the book length
If the first novel is less than the second novel:

Return false
Return true otherwise

5. Author Sorting Class

5.1 Description:

This class sorts the novels based on the last names of the authors. It takes the input of novels and implements a three way quicksort to organize the authors alphabetically. This algorithm is based off the Three Way String Quicksort located in “Algorithms 4th Edition” or online at http://algs4.cs.princeton.edu/51radix/Quick3string.java.html.
va.html.

5.2 Interface:

5.2.1 Uses:

- None

5.2.2 Types:

Book[] book

5.2.3 Code Breakdown:

Public static void sort(Book[] book)

Takes a collection (array of novels as input), and calls the private sortAuthor to implement a quicksort. This method also checks if the list is already sorted prior to activating the actual sort.

5.3 Implementation:

5.3.1 Uses:

None

5.3.2 Types:

final int CUTOFF

5.3.3 Code Breakdown:

Private static int character(Book book, int d)

Int bookLength = the length of the novel author in the inputted novel
Assert that the character spot being compared is within bounds
Assert that the novels are sorted

Private static void sort(Book[] book, int low, int high, int d)

If the greatest valued element in the array is<= the lowest value + cutoff point:
Switch the places of the novels
Break

Int lt = temporary store of lowest value to increment
Int gt = temporary store of highest value to decrement
Int I = lowest value + 1
Int v = the unicode code point of the novel author at character “d”

While (the lowest value +1 is less than or equal to the size of the array):
Int t = Unicode code point of the character at integer d in the book on top of array
If the lowest value is less than the temporary value:
Exchange the lowest value + 1 and the lowest value + 2
Else if the lowest value is greater than the temporary value:
Exchange the novels of the lowest value + 1 and the greatest value – 1
Else:
Increment i (lowest value +1) by one

sort(call itself to sort the sub array taking away the greatest valued element by the current character)

if the Unicode code point is greater or equal to zero:
sort(call itself recursively to sort the two novels by the next character)
sort(call itself recursively to sort the temporary greatest value +1, the highest value at the next character)

private static void insertAuthor(Book[] book, int low, int high, int d)

for the range from zero the size of the array:
for the range of the lowest number to the smallest value novel:
exchange the two novels

private static void exchange(Book[] book, int I, int j)

switches the two novels at place I and j using a novel placeholder

private static boolean less(Book x, Book y, int d)
String firstBook = the author of book x
String secondBook = the author of the book y
Assert if the two titles are equal

For the range of the place of the character and the minimum length:
If the letter searched in the first novel is less than the second novel:
Return true
If the letter compared in the first novel is greater than the second novel:
Return false

If all other cases fail, return the boolean value of the comparison of the first and second novel

Private static boolean isSorted(Book[] book)

For the range of 1 and the book length
If the first novel is less than the second novel:
Return false

Return true otherwise

6. JTable Class

6.1 Description:

This class creates the Jtable that is used to display the books after they have been sorted in different ways

6.2 Interface:

6.2.1 Uses:

java.awt.FlowLayout;
java.io.File;
java.io.FileNotFoundException;
java.util.Scanner;

javax.swing.JFrame;
javax.swing.JScrollPane;
javax.swing.JTable;
javax.swing.table.DefaultTableModel;

6.2.2 Types:

6.2.3 Code breakdown:

public start(int flag)
Contains processes necessary to format JTable as well as fill JTable with sorted information as follows

set cell editable = false
set Title = Bookotron 9001
set size, visibility, layout, default close operation

```
new Scanner input from data/Input.txt
String[][] books = new String[20][5]
for all books
        book[i] = split line at ","
for all books
        create book data structure
Book[] list = new Book array
for all books
        add book[i] to list array


switch (flag)
        case 1 = sort with Insertion
        case 2 = sort with Selection
        case3 = sort with Quicksort by Author
        case4 = sort with Quicksort by Title

for books length
        set value of each row
                getTitle
                getAuthor
                getLength
                getRating
```

6.3 Implementation:

6.3.1 Uses:

None

6.3.2 Types:

None

6.3.3 Code Breakdown:

No private methods.

7. MainFrame Class

7.1 Description:

This class creates the menu screen with 5 buttons for different options for sorting and recommendation

7.2 Interface:

7.2.1 Uses:

java.awt.*;
java.awt.event.ActionEvent;
java.awt.event.ActionListener;
java.io.FileNotFoundException;
java.io.IOException;

7.2.2 Types:

None

7.2.3 Code breakdown:

public MainFrame()
Sets default behaviour for JFrame

7.3 Implementation:

7.3.1 Uses:
None

7.3.2 Types:

Jpanel contentPane
JButton button1
JButton button2
JButton button3
JButton button4
JButton button5
JLabel label

7.3.3 Code Breakdown:

private initContentPanel()

create content pane
setLayout to grid layout

private initContents()

create button panel
create button1
create button2
create button3
create button4
create button5
create action listeners for all buttons

add buttons to JFrame
add Title to JFrame screen

8. Binary Search Tree Class

8.1 Description:

This class is implemented to emulate a binary search tree to create recommendations for the user based on ratings of the novels.

8.2 Interface:

8.2.1 Uses:

- None

8.2.2 Types:

int key
String val
Node left
Node right
int N

8.2.3 Code breakdown:

public Node(int key, String val, int N)
creates a node containing a key value, which is an integer, a String for its title contents and an int N for its rating.

public int Size()
Returns the size of the root in the BST

public String get(int key)
Returns returns the root at a given value

public put(int key, String val)
adds a value of a novel into the BST

8.3 Implementation:

8.3.1 Uses:

None

8.3.2 Types:

None

8.3.3 Code Breakdown:

private int size(Node x)

if the given value is null:
    return 0

else:
    returns the size of BST at Node x

private String get(Node x, int key)

if the node is null:
    return null

if the key is less than the key at node x;
    return a recursive statement of get(the left subtree, at int key)

else if the key is greater than the key at node x:
    return a recursive statement of get(the right subtree, at int key)

else return the value at x

9. Recommend Tree Class

9.1 Description:

This class is used to create a Binary Search Tree of Books, using specific properties of the books.

9.2 Interface:

9.2.1 Uses:

- java.io.BufferedReader
- java.io.FileReader
- java.io.IOException
- java.util.ArrayList

9.2.2 Types:

BST recommendST

9.2.3 Code breakdown:

RecommendTree(String[][] fileInfo)
Constructor that builds the tree and populates it.

public String get(int Key)
Retrieves the String value at the specified key in the BST.

9.3 Implementation:

9.3.1 Uses:

None

9.3.2 Types:

None

9.3.3 Code Breakdown:

No private methods. Since this is a data structure no calculations occur in this class.

10. Recommender Class

10.1 Description:

This class is makes use of the BST of books data structure to search for and display highly rated books for the client.

10.2 Interface:

10.2.1 Uses:

- java.awt.FlowLayout;
- java.io.BufferedReader;
- java.io.FileReader;
- java.io.IOException;
- java.util.ArrayList;

10.2.2 Types:

None

10.2.3 Code breakdown:

Recommender()
Constructor that finds highly rated books and rates it using the BST of books. Also displays the results in a JTable.

10.3 Implementation:

10.3.1 Uses:

- javax.swing.JFrame;
- javax.swing.JScrollPane;
- javax.swing.JTable;
- javax.swing.table.DefaultTableModel;

10.3.2 Types:

private static final long serialVersionUID
DefaultTableModel model
JTable table
String col[]

10.3.3 Code Breakdown:

class toTable()
Subclass used to create the JTable that the information is displayed in.

start(String[][] data)
Takes a 2D String array of book information and outputs a JTable of the information.

Design Review:

The choice to use a 2D array to hold input from the dataset is optimal for this particular implementation. If in the future this project was to be expanded on to allow users to add their own entries through the interface another data structure would need to be implemented in order to allow for a dynamic dataset

The 5 button GUI gives the user explicit ways to get to each portion of the program. This makes it the interface more user friendly.

As mentioned in our Executive Summary, the algorithms that were chosen in the design of the project allow for fastest possible run times given the relative data set being used.