Vanna Moore

CMPS 390

Program 4 – Sacry Linked List

/\* This program uses a linked list to take in a file of names from the user and sort them alphabetically. It

starts with asking the user for the file name. Once they enter the name, a menu displays giving them the option to 1) display the list, 2) Display the length of the list 3) Print names that start with a specific letter,

4)Display the length of a section of the list by first letter, 5) Delete a name from the list,

and they can enter 0 to exit. The menu returns after every action the user chooses is complete until they press option 0 to exit. It uses methods called delete to delete names. It uses a method called addNode that uses methods called addNode and addFirst to insert names alphabetically. It uses methods called convertName to convert names into a base 26 number to insert them in order. It also uses a method called firstLetter to make an index. It has methods showList to print it out and showIndex to print out a section of the list. The last method is display menu, which gives the user a menu to choose their next action from.

\* \*/

A screenshot of a computer program

Description automatically generated

After the user enters a file name, the display menu appears. The menu keeps prompting after every action is completed until the user presses option 0 to exit the menu.

A screenshot of a computer program

Description automatically generated

Option 1 is to show the list in alphabetical order.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Option 2 will give the user the length of the list.

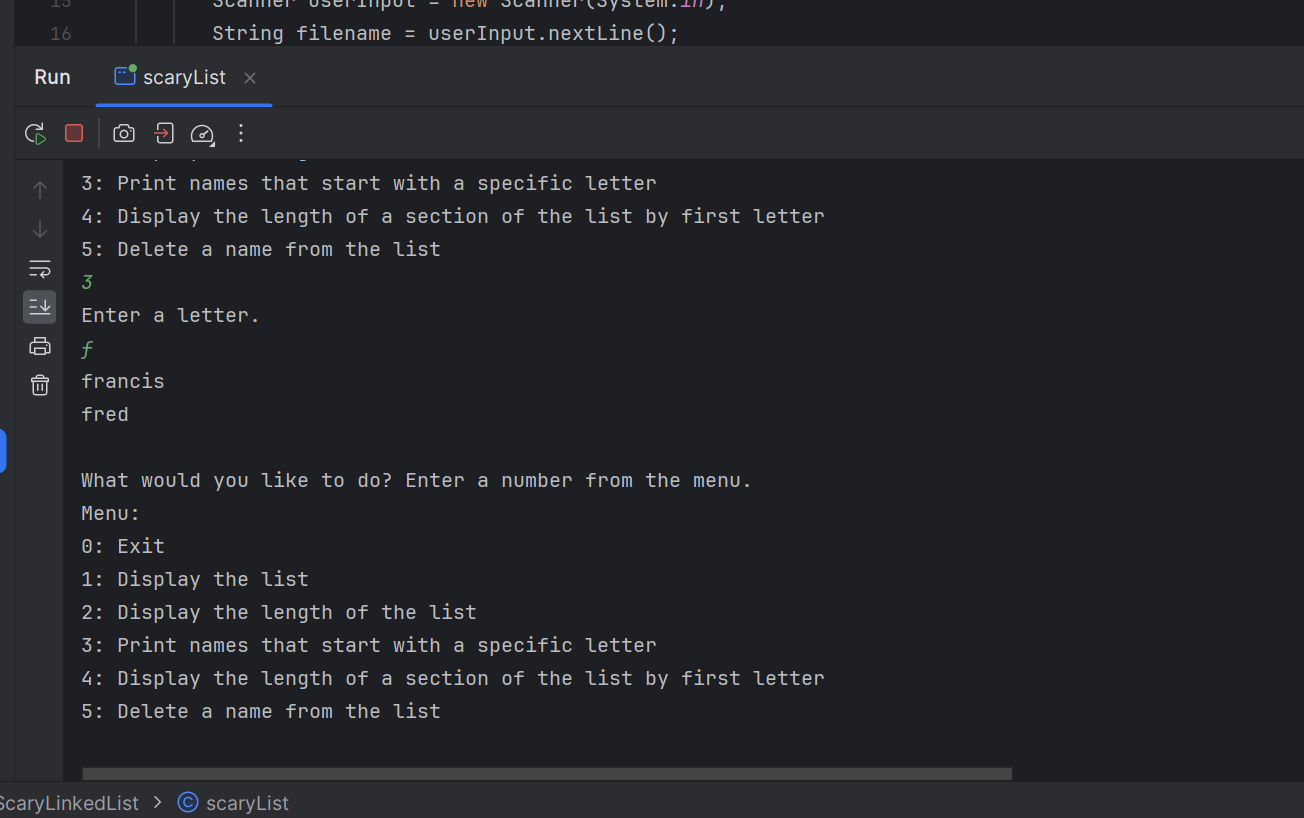
A screenshot of a computer program

Description automatically generated

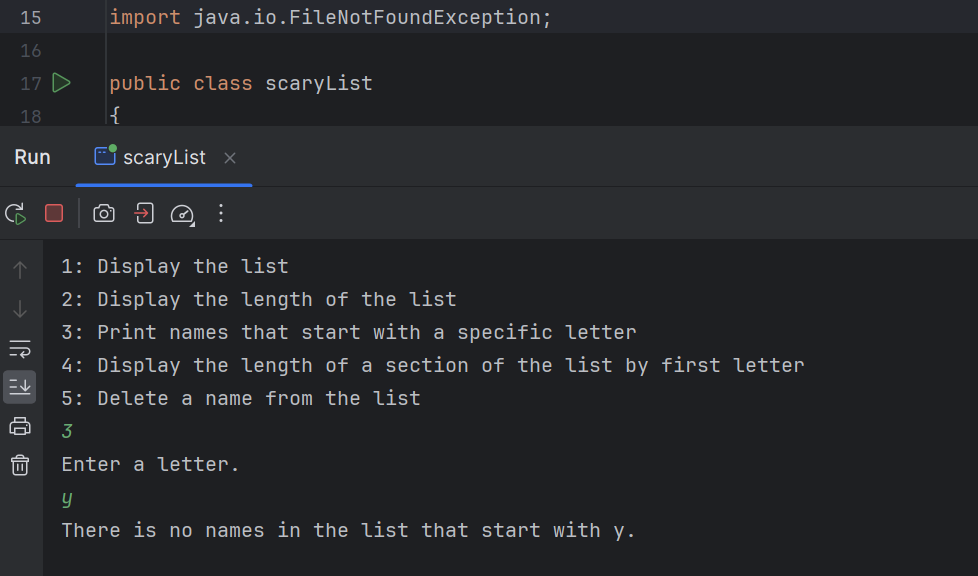
Option 3 will let them print a name of a certain index.

A screen shot of a computer

Description automatically generated



If there are no names in the list that start with that letter, it will give the user a message telling them that.



Option 4 will display the length of a specified index.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Option 5 will let the user delete a name from the list.

A screenshot of a computer program

Description automatically generated

Option 0 will let them exit the program.

A screenshot of a computer program

Description automatically generated

**scaryList.java (main method)**

import java.io.File;  
import java.util.Scanner;  
import java.io.FileNotFoundException;  
  
public class scaryList  
{  
 public static void main(String[] args) throws FileNotFoundException {  
 int code;  
 String[] listArray = new String[100];  
 int[] codeArray = new int[100];  
 letterIndex nameCode = new letterIndex(); // tester  
   
// Scanner for user file io  
 System.*out*.println("Enter the file name:");  
 Scanner userInput = new Scanner(System.*in*);  
 String filename = userInput.nextLine();  
 File file = new File(filename);  
 Scanner scan = new Scanner(file);  
  
 list list = new list();  
 list.init();  
  
 while(scan.hasNextLine()){  
   
 list.addNode(scan.nextLine());  
  
 } //close while loop  
  
 //Display Menu  
 list.displayMenu();  
  
 } //close main  
} // close scary list class

**list.java**

import java.util.Scanner;  
  
public class list {  
 Scanner sc = new Scanner(System.*in*);  
 String data;  
 node next;  
 node curr;  
 node front;  
 node tail;  
 node spot;  
 node prev;  
 node newNode;  
 node temp;  
 int count = 0;  
  
 // initialize list  
 **public void init()** {  
 front = null;  
 } // close init  
  
 // makeNode: Method to create a new node  
 **public node makeNode(String data)** {  
 newNode = new node();  
 newNode.data = data;  
 newNode.next = null;  
  
 return newNode;  
 } // close make node  
  
 // addFirst: Method to add to the beginning of a list in alphabetical order  
 **public node addFirst(String n)** {  
 front = curr;  
 if (front == null) {  
 front = makeNode(n);  
 } else {  
 newNode = makeNode(n);  
 newNode.next = curr;  
 front = newNode;  
 }  
 return front;  
 }// close add last  
  
 // addLast: Method to add to the end of a list  
 **public node addLast(String data)** {  
 if (front == null) {  
 front = makeNode(data);  
 tail = front;  
 } else {  
 tail = findTail();  
 tail.next = makeNode(data);  
 tail = tail.next;  
 tail.next = null;  
 }  
 return tail;  
 } // close add last  
  
 **public void delete(String n){**  
 curr = front;  
 while(curr.next != null) {  
 if (convertName(curr.next.data) == convertName(n)) {;  
 temp = curr;  
 curr = curr.next;  
 temp.next = curr.next;  
 System.*out*.println(n + " has been deleted: \n");  
 showList();  
 displayMenu();  
 }  
 else if(convertName(curr.data) != convertName(n)){  
 curr = curr.next;  
 }  
 }  
 }  
  
 // findTail: method to find last node in the list  
 **public node findTail()** {  
 node curr;  
 curr = front;  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
 return curr;  
 } // close findTail  
  
 **public void listLength()** {  
 node curr;  
 curr = front;  
 count = 0;  
 while (curr != null) {  
 count++;  
 curr = curr.next;  
 }  
 System.*out*.println("The length of the list is: " + count);  
 } // close listLength  
  
 **public void indexLength(String x)** {  
 int indexIt = firstChar(x);  
 curr = front;  
 boolean isInList = false;  
 while (curr != null) {  
 if(firstChar(curr.data) == indexIt){  
 curr = curr.next;  
 isInList = true;  
 count++;  
 }  
 else if(curr.next == null && !isInList){  
 System.*out*.println("There is no names in the list that start with " + x + ".");  
 curr = curr.next;  
 }  
 else if(firstChar(curr.data) != indexIt){  
 curr = curr.next;  
 }  
 }  
 System.*out*.println("The length of the list is: " + count);  
 } // close indexLength  
  
 // showList: Method to print out a list  
 **public void showList()** {  
 node curr;  
 curr = front;  
 while (curr != null) {  
 System.*out*.println(curr.data);  
 curr = curr.next;  
 }  
 }// close show list  
  
 // Method to add a Node after a node in alphabetical order  
 **public node addNext(String n)** {  
 newNode = makeNode(n);  
 newNode.next = curr.next;  
 curr.next = newNode;  
 return newNode;  
 }// close add next  
  
 // Method to calculate base 26 value for a string.   
 **public int convertName(String n)** {  
 int x, y, z, nameCode;  
 x = n.charAt(0) - 'a';  
 y = n.charAt(1) - 'a';  
 z = n.charAt(2) - 'a';  
  
 nameCode = (x \* (26 \* 26)) + (y \* (26)) + (z \* 1);  
  
 return nameCode;  
 }// close method convertName  
  
 //Method to analyze the first char in a string  
 **public char firstChar(String n**) {  
 int x, charCode;  
 char first = n.charAt(0);  
  
 return first;  
 }  
  
 // "insert" method called addNode  
  **public node addNode(String n)** {  
 curr = front;  
 // Case1: add the first node in the list  
 if (front == null) {  
 newNode = addFirst(n);  
 front = newNode;  
 }  
 // Case 2: add to front of list when newNode is < front node  
 else if (convertName(front.data) > convertName(n)) {  
 newNode = addFirst(n);  
 front = newNode;  
 }  
 curr = front;  
 int compare = 0;  
 while (convertName(n) > convertName(curr.data)){  
 //System.out.println("test");  
 if(curr.next == null){  
 newNode = addLast(n);  
 }  
 else if (convertName(n) < convertName(curr.next.data)){  
 newNode = addNext(n);  
 }  
 else{  
 curr = curr.next;  
 }  
 }  
 return newNode;  
 } // close addNode  
   
 // Menu: Method for user menu  
 public void displayMenu(){  
  
 boolean menuOn = true;  
 while(menuOn) {  
 System.*out*.println("\nWhat would you like to do? Enter a number from the menu. \nMenu: ");  
 System.*out*.println("0: Exit"); // done  
 System.*out*.println("1: Display the list");// method done  
 System.*out*.println("2: Display the length of the list"); // method done  
 System.*out*.println("3: Print names that start with a specific letter"); // method done  
 System.*out*.println("4: Display the length of a section of the list by first letter"); // method done  
 System.*out*.println("5: Delete a name from the list");  
  
 int x = sc.nextInt();  
   
 if(x == 0){  
 menuOn = false;  
 }  
 else if(x == 1){  
 showList();  
 }  
 else if(x == 2){  
 listLength();  
 }  
 else if(x == 3){  
 System.*out*.println("Enter a letter.");  
 showIndex(sc.next().charAt(0));  
 }  
 else if(x == 4){  
 System.*out*.println("Enter the letter for the index length you want.");  
 String y = String.*valueOf*(sc.next().charAt(0));  
 indexLength(y);  
 }  
 else if(x == 5){  
 System.*out*.println("Enter the name you would like to delete.");  
 String nameToDelete = sc.next();  
 delete(nameToDelete);  
 }  
 }  
 } //close displayMenu  
  
 **public void showIndex(char x)** {  
 int indexIt;  
  
 curr = front;  
 boolean isInList = false;  
 while (curr != null) {  
 if(firstChar(curr.data) == x){  
 System.*out*.println(curr.data);  
 curr = curr.next;  
 isInList = true;  
 count++;  
 }  
 else if(curr.next == null && !isInList){  
 System.*out*.println("There is no names in the list that start with " + x + ".");  
 curr = curr.next;  
 }  
 else if(firstChar(curr.data) != x){  
 curr = curr.next;  
 }  
 }  
 } // close showIndex method  
} // close List Class