ACCELERATED CS 1 & 2 - CSC 250 (FALL 2015)

Project 3: #Music

Randell Carrido
Department of Computer Science

David Pauls
Department of Physics

Theresa Pham
Department of Electrical & Computer Engineering

Instructor
Vinayak Elangovan, Ph.D.
Department of Computer Science

Submitted On: 12/3/15

TABLE OF CONTENTS

1. INTRODUCTION

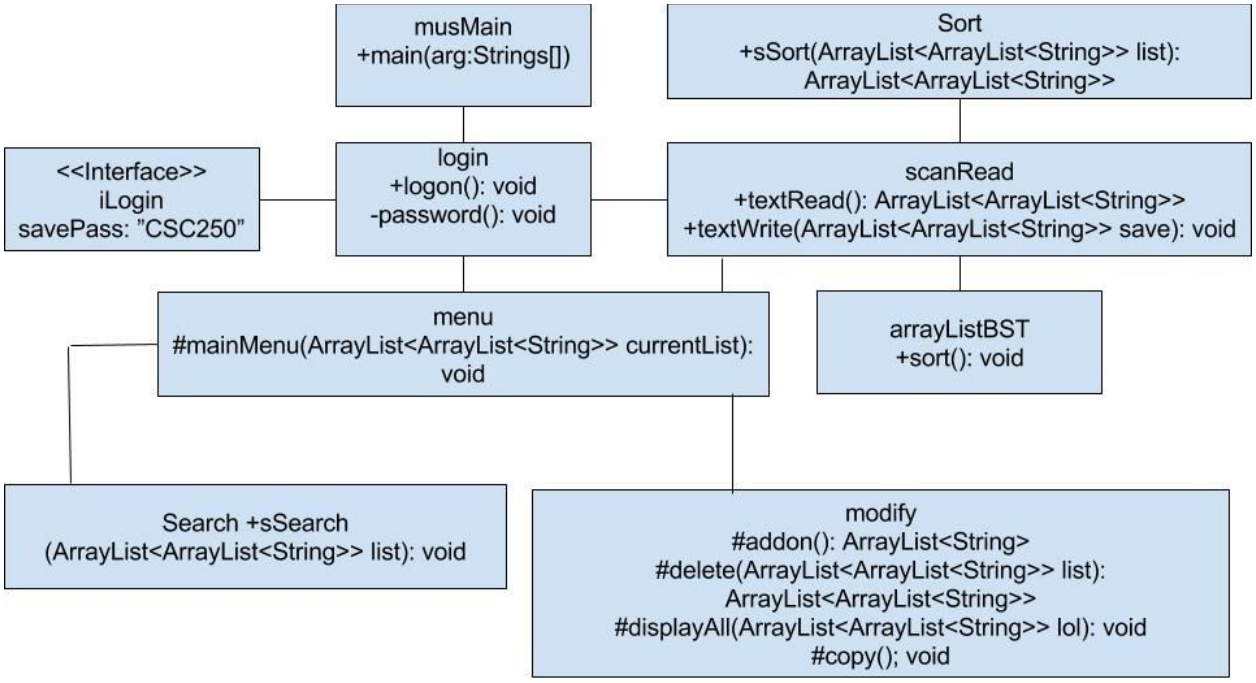
This program stores information about the inventory of a music store and allows those with the password to go in and make changes. Anyone can view the instruments available in the store with the prices and locations in the store, but only employees who work in the store can add instruments into the database when they come in to the store, modify the price of an instrument, or delete an instrument from the database after it is sold.

2. BACKGROUND

The program was coded in Java and utilizes the Java Collections API (through the use of different data structures such as an ArrayList and a non-linear data structure, a binary search tree), algorithms that come from the Collections API, as well as object oriented principles. More specifically, the data for each instrument was stored as an ArrayList of String Arraylists, allowing easy insertion or deletion of instruments or the corresponding parameters of model, brand, and price. An ArrayList allows easy access if you know what you want to search for, unlike and LinkedList, which requires you to iterate through it to find something. The Collections API contains its own sorting method, but we had to edit this method to fit our custom data, and sort by whichever parameter was necessary.

By implementing a Binary Search Tree, we could easily order our data by price, allowing customers to easily choose an instrument in their price range. The BST uses a recursive node creation algorithm, automatically sorting all inserted elements. Each newly created node is compared to the root node, then sorted left or right all the way down the tree. In addition to sorting by cost, however, the node also contains the object ArrayList for each instrument, allowing data to be sorted by cost but to display all information regarding the instrument as well.

3. DESIGN & IMPLEMENTATION

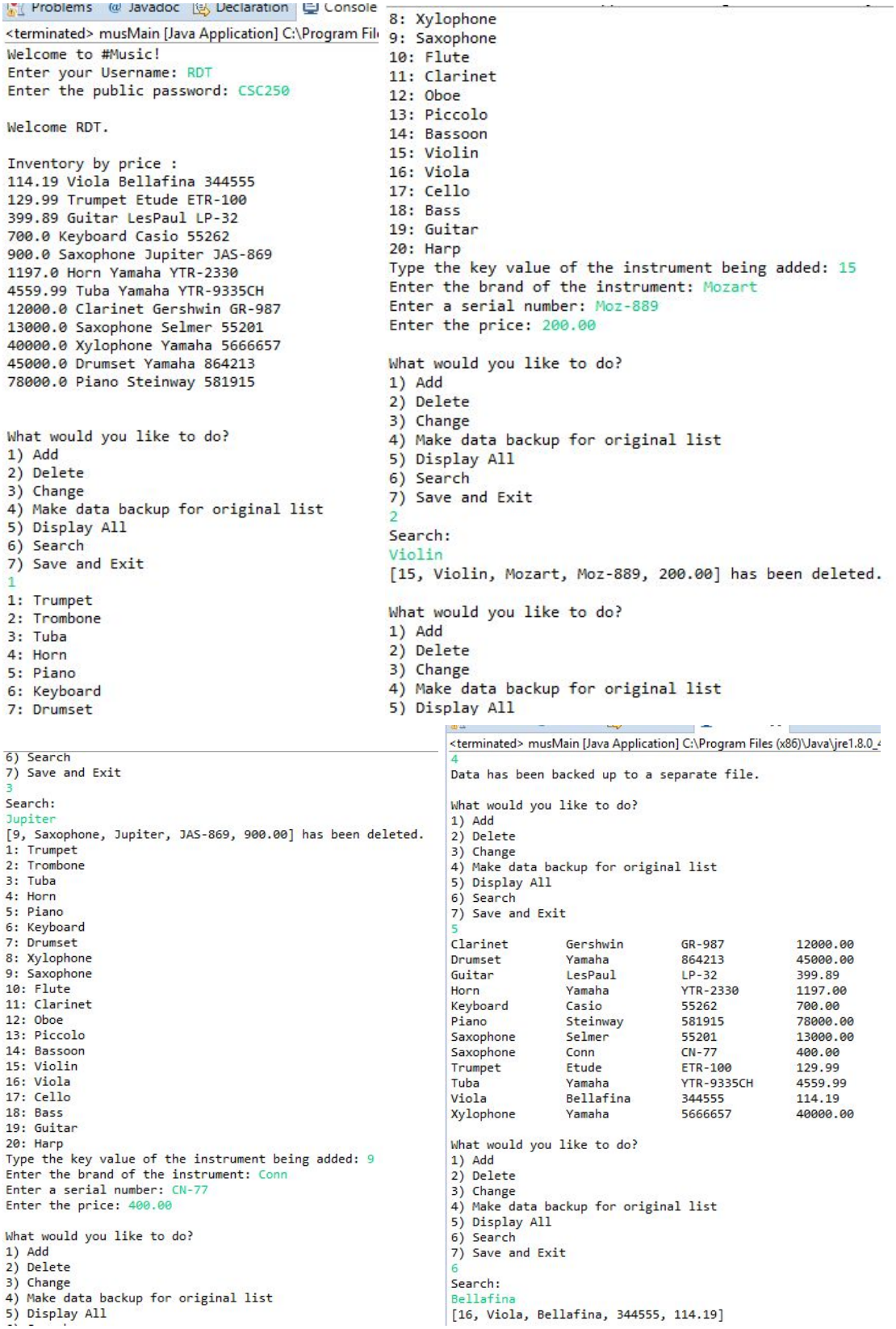


There were many things we had to remember to implement while we were doing this project. It had to include a login functionality, allow specific users to add, delete, search and modify the data, and allow users to backup the data. We also had to implement searching and sorting algorithms, as well as a non-linear data structure that would be useful to our program.

To have a login functionality for the program so that only people with the correct password could log in, we made a Password Interface. For adding and deleting, we used the .add() and .remove() methods from the Java Collections API. For searching, we implemented a linear search algorithm, which goes through every String in each ArrayList in order to find and output a match. For sorting in our linear data structure, we used a selection sort algorithm, which combines searching and sorting by finding the smallest to largest prices and then putting them in order in a new ArrayList. For our binary search tree, a non-linear data structure, we created our own algorithm to sort the prices, but because the prices are originally saved as Strings, we had to parse them into doubles before we could compare them as number values. Finally, to allow users to backup data, we used the FileReader to read every line from the original text file and copy it all over to a new text file using FileWriter.

Once the user successfully logs in, all of the other functionalities are shown to the user in a menu, where the user can select an option and the program will then go through any steps with the user to accomplish the task that the user wanted.

4. RESULTS / SAMPLE OUTPUTS



```
Problems  Javadoc  Declaration  Console
<terminated> musMain [Java Application] C:\Program Fil
Welcome to #Music!
Enter your Username: RDT
Enter the public password: CSC250

Welcome RDT.

Inventory by price :
114.19 Viola Bellafina 344555
129.99 Trumpet Etude ETR-100
399.89 Guitar LesPaul LP-32
700.0 Keyboard Casio 55262
900.0 Saxophone Jupiter JAS-869
1197.0 Horn Yamaha YTR-2330
4559.99 Tuba Yamaha YTR-9335CH
12000.0 Clarinet Gershwin GR-987
13000.0 Saxophone Selmer 55201
40000.0 Xylophone Yamaha 5666657
45000.0 Drumset Yamaha 864213
78000.0 Piano Steinway 581915


What would you like to do?
1) Add
2) Delete
3) Change
4) Make data backup for original list
5) Display All
6) Search
7) Save and Exit
1
1: Trumpet
2: Trombone
3: Tuba
4: Horn
5: Piano
6: Keyboard
7: Drumset
8: Xylophone
9: Saxophone
10: Flute
11: Clarinet
12: Oboe
13: Piccolo
14: Bassoon
15: Violin
16: Viola
17: Cello
18: Bass
19: Guitar
20: Harp
Type the key value of the instrument being added: 15
Enter the brand of the instrument: Mozart
Enter a serial number: Moz-889
Enter the price: 200.00

What would you like to do?
1) Add
2) Delete
3) Change
4) Make data backup for original list
5) Display All
6) Search
7) Save and Exit
2
Search:
Violin
[15, Violin, Mozart, Moz-889, 200.00] has been deleted.

What would you like to do?
1) Add
2) Delete
3) Change
4) Make data backup for original list
5) Display All
6) Search
7) Save and Exit
3
Search:
Jupiter
[9, Saxophone, Jupiter, JAS-869, 900.00] has been deleted.
1: Trumpet
2: Trombone
3: Tuba
4: Horn
5: Piano
6: Keyboard
7: Drumset
8: Xylophone
9: Saxophone
10: Flute
11: Clarinet
12: Oboe
13: Piccolo
14: Bassoon
15: Violin
16: Viola
17: Cello
18: Bass
19: Guitar
20: Harp
Type the key value of the instrument being added: 9
Enter the brand of the instrument: Conn
Enter a serial number: CN-77
Enter the price: 400.00

What would you like to do?
1) Add
2) Delete
3) Change
4) Make data backup for original list
5) Display All
<terminated> musMain [Java Application] C:\Program Files (x86)\Java\jre1.8.0_
4
Data has been backed up to a separate file.

What would you like to do?
1) Add
2) Delete
3) Change
4) Make data backup for original list
5) Display All
6) Search
7) Save and Exit
5
Clarinet        Gershwin        GR-987          12000.00
Drumset         Yamaha          864213          45000.00
Guitar          LesPaul         LP-32           399.89
Horn            Yamaha          YTR-2330        1197.00
Keyboard        Casio           55262           700.00
Piano           Steinway        581915          78000.00
Saxophone       Selmer          55201           13000.00
Saxophone       Conn            CN-77           400.00
Trumpet         Etude           ETR-100         129.99
Tuba            Yamaha          YTR-9335CH      4559.99
Viola           Bellafina       344555          114.19
Xylophone       Yamaha          5666657         40000.00

What would you like to do?
1) Add
2) Delete
3) Change
4) Make data backup for original list
5) Display All
6) Search
7) Save and Exit
6
Search:
Bellafina
[16, Viola, Bellafina, 344555, 114.19]
```

5. ALGORITHM / TIME COMPLEXITY ANALYSIS

Method	Time (ns) #1	Time (ns) #2	Time (ns) #3	Time Complexity
sSort()	3253196	2573523	3240372	O(n^2)
sort()	13158974	11257943	12568045	O(n^3)
sSearch()	1156212	246734	831509	O(n)

The time complexity of the linear search method sSearch() is O(n) and the selection sort algorithm, sSort(), is O(n^2). For our non-linear data structure, a Binary Search Tree, we sorted the prices using the sort() method, which had a time complexity of O(n^3). You can see from the time tabulations that they agree with the order of the time complexities. The sSearch() method took the shortest amount of time and had the smallest time complexity. sSort(), with its time complexity of O(n^2) took longer time than the sSearch() method to complete but shorter time than the sort() method. The sort() method of course had the longest times for completion, which coincides with its large time complexity of O(n^3).

6. CONCLUSION

While this was a good, all-encompassing final project, we found it difficult to do a coding project between 3 people, as when you are coding you need to make sure you are using the same variables and allow everything to connect to each other. Certain code will not work unless you have all of the code, but we figured out a system where we could each work on certain functions of the program that could work on their own and then we would send them to one person, Randell, who made sure all the functions in the program worked together. Theresa worked on the add function and the program's ability to backup the data. David worked on the program's abilities to write to files and the Binary Search Tree that was incorporated into the music inventory program for listing the prices in either increasing or decreasing order. Randell worked on the functions to search, modify, and delete, as well as putting the whole program together. Because putting the whole program together was a big task, Theresa and David communicated with Randell while he did it to help him when he needed it.

This project helped us take everything we learned though and implement it into something that can be used in real life. We were able to not only reuse our knowledge of object-oriented principles and searching and sorting algorithms, but we were also able to use our new knowledge of the Java Collections API to make use of data structures that, in some ways, are easier to use than regular arrays, which is what we would have used for this project otherwise. The Java Collections API also provides algorithms that makes creating certain functionalities for the program much simpler, so it is a great asset to us as programmers, and by combining that knowledge with everything else that we know from this course, we find that there are many possibilities when it comes to finding solutions using coding.

7. REFERENCES

None

8. APPENDIX

```
public class musMain {
public static void main(String args[]){
	login l = new login();
	l.logon(); //Login
}
}

public interface iLogin {
	String savePass = "CSC250"; //Password
}

public class login implements iLogin{ //interface
	public void logon(){
		Scanner scan = new Scanner(System.in);
		System.out.println("Welcome to #Music!");
		System.out.print("Enter your Username: "); //Any username acceptable
		String x = scan.nextLine();
		password();
		System.out.println("\nWelcome " + x + ".");
		scanRead s = new scanRead();
		ArrayList<ArrayList<String>> currentList = new ArrayList<ArrayList<String>>();
		currentList = s.textRead();
		menu m = new menu();
		m.mainMenu(currentList);
	}
	private void password(){
		Scanner scan = new Scanner(System.in);
		System.out.print("Enter the public password: "); //must be CSC250
		String y = scan.nextLine();
		if(savePass.equals(y)){
		}
		else{
			System.out.println("Sorry, wrong password.");
			password();
		}
	}
}
public class scanRead {
	ArrayList<ArrayList<String>> lol = new ArrayList<ArrayList<String>>();
```

```
	Sort s = new Sort();
	public ArrayList<ArrayList<String>> textRead(){
		String fileName = "C://Users/Randell/Desktop/Java/data.txt";
		String line = null;

		try{//Reads from txt
			FileReader fileReader = new FileReader(fileName);
			BufferedReader bufferedReader = new BufferedReader(fileReader);

			while((line = bufferedReader.readLine()) != null) {

				List<String> list = new ArrayList<String>(Arrays.asList(line.split("
")));
				lol.add((ArrayList<String>) list);
			}
			bufferedReader.close();
			lol = s.sSort(lol); //Sorts Alphabetically
			arraylistBST sortedbycost = new arraylistBST();
			sortedbycost.sort();//Sorts into a tree
		}
		catch(FileNotFoundException ex){
			System.out.println("Unable to open file '" + fileName + "'");

		}
		catch(IOException ex){
			System.out.println("Error reading file '" + fileName + "'");
		}
		return lol;
	}
	public void textWrite(ArrayList<ArrayList<String>> save){
		try{
			FileWriter fileWriter = new
FileWriter("C://Users/Randell/Desktop/Java/data.txt");
			BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

			for(int i = 0; i< save.size(); i++)
			{
				for(int j = 0; j< save.get(i).size(); j++)
				{
					bufferedWriter.write(save.get(i).get(j) + " ");
				}
				bufferedWriter.newLine();
			}

			//closes writer
			bufferedWriter.close();
```

```
                }
                catch(FileNotFoundException ex)
                {
                        System.out.println("Unable to open file '" + "filetoberead.txt" + "'");
                }
                catch(IOException ex)
                {
                        System.out.println("Error reading file '" + "filetoberead.txt" + "'");
                }
        }
}
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;



public class Sort{
        public ArrayList<ArrayList<String>> sSort(ArrayList<ArrayList<String>> list){
                Collections.sort(list, new Comparator<ArrayList<String>>(){
            @Override
            public int compare(ArrayList<String> arg0, ArrayList<String> arg1) {
               return arg0.get(1).compareTo(arg1.get(1));
            } //Selection Sort
          });
                return list;
        }
}
public class arraylistBST
{
        public static  Node root;
        public arraylistBST(){
                this.root = null;
        }

        public boolean find(double id)
        {
                Node current = root;
                while(current!=null){
                        if(current.data==id){
                                return true;
                        }else if(current.data>id){
                                current = current.left;
                        }else{
                                current = current.right;
                        }
```

```
        }
        return false;
}
public boolean delete(double id)
{
        Node parent = root;
        Node current = root;
        boolean isLeftChild = false;
        while(current.data!=id){
                parent = current;
                if(current.data>id){
                        isLeftChild = true;
                        current = current.left;
                }else{
                        isLeftChild = false;
                        current = current.right;
                }
                if(current ==null){
                        return false;
                }
        }
        //if i am here that means we have found the node
        //Case 1: if node to be deleted has no children
        if(current.left==null && current.right==null){
                if(current==root){
                        root = null;
                }
                if(isLeftChild ==true){
                        parent.left = null;
                }else{
                        parent.right = null;
                }
        }
        //Case 2 : if node to be deleted has only one child
        else if(current.right==null){
                if(current==root){
                        root = current.left;
                }else if(isLeftChild){
                        parent.left = current.left;
                }else{
                        parent.right = current.left;
                }
        }
        else if(current.left==null){
                if(current==root){
                        root = current.right;
```

```
                }else if(isLeftChild){
                        parent.left = current.right;
                }else{
                        parent.right = current.right;
                }
        }else if(current.left!=null && current.right!=null){

                //now we have found the minimum element in the right sub tree
                Node successor          = getSuccessor(current);
                if(current==root){
                        root = successor;
                }else if(isLeftChild){
                        parent.left = successor;
                }else{
                        parent.right = successor;
                }
                successor.left = current.left;
        }
        return true;
}

public Node getSuccessor(Node deleleNode){
        Node successsor =null;
        Node successsorParent =null;
        Node current = deleleNode.right;
        while(current!=null){
                successsorParent = successsor;
                successsor = current;
                current = current.left;
        }
        //check if successor has the right child, it cannot have left child for sure
        // if it does have the right child, add it to the left of successorParent.
        //      successsorParent
        if(successsor!=deleleNode.right){
                successsorParent.left = successsor.right;
                successsor.right = deleleNode.right;
        }
        return successsor;
}
public void insert(double id, ArrayList<String> list){
        Node newNode = new Node(id, list);
        if(root==null){
                root = newNode;
                return;
        }
        Node current = root;
```

```
		Node parent = null;
		while(true){
			parent = current;
			if(id<current.data){
				current = current.left;
				if(current==null){
					parent.left = newNode;
					return;
				}
			}else{
				current = current.right;
				if(current==null){
					parent.right = newNode;
					return;
				}
			}
		}
	}
	public void display(Node root){
		if(root!=null){
			display(root.left);
			System.out.println(root.data + " " + root.list.get(1) + " "
					+ root.list.get(2) + " " + root.list.get(3) + " " );
			display(root.right);
		}
	}
	public void sort(){
		String fileName = "C://Users/Randell/Desktop/Java/data.txt";
		String line = null;
		ArrayList<ArrayList<String>> lol = new ArrayList<ArrayList<String>>();
		try{
			FileReader fileReader = new FileReader(fileName);
			BufferedReader bufferedReader = new BufferedReader(fileReader);
			while((line = bufferedReader.readLine()) != null) {
				List<String> list = new ArrayList<String>(Arrays.asList(line.split("
")));
				lol.add((ArrayList<String>) list);
			}
			bufferedReader.close();
			arraylistBST b = new arraylistBST();


			for(int i = 0; i < lol.size(); i++)
			{
				b.insert(Double.parseDouble(lol.get(i).get(4)), lol.get(i));
			}
```

```
                        System.out.println("");
                        System.out.println("Inventory by price : ");
                        b.display(b.root);
                        System.out.println("");
                }
                catch(FileNotFoundException ex){
                        System.out.println("Unable to open file '" + fileName + "'");
                }
                catch(IOException ex){
                        System.out.println("Error reading file '" + fileName + "'");
                }
        }
}
class Node{
        double data;
        ArrayList<String> list;
        Node left;
        Node right;
        public Node(double data, ArrayList<String> list){
                this.data = data;
                this.list = list;
                left = null;
                right = null;
        }
public class menu extends modify{
        scanRead s = new scanRead();
        Scanner scan= new Scanner(System.in);
        Scanner scan1= new Scanner(System.in);
        Search find = new Search();
        modify e = new modify();
        Sort p = new Sort();


        protected void mainMenu(ArrayList<ArrayList<String>> currentList){ //Always return
here after method completed
                p.sSort(currentList);
                System.out.println("\nWhat would you like to do?\n"
                                + "1) Add\n"
                                + "2) Delete\n"
                                + "3) Change\n"
                                + "4) Make data backup for original list\n"
                                + "5) Display All\n"
                                + "6) Search\n"
                                + "7) Save and Exit");
                int choice = scan1.nextInt();
                if(choice == 1){
```

```
                ArrayList<String> z = e.addon();
                currentList.add(z);
                mainMenu(currentList);
            }
            else if(choice == 2){
                currentList = e.delete(currentList);
                mainMenu(currentList);
            }
            else if(choice == 3){
                currentList = e.delete(currentList);
                ArrayList<String> z = e.addon();
                currentList.add(z);
                mainMenu(currentList);
            }
            else if (choice == 4){
                copy();
                mainMenu(currentList);
            }
            else if (choice == 5){
                displayAll(currentList);
                mainMenu(currentList);
            }
            else if(choice == 6){
                Search g = new Search();
                g.sSearch(currentList);
                mainMenu(currentList);
            }
            else if(choice == 7){
                s.textWrite(currentList);
            }
        }
}
public class modify {
        protected ArrayList<String> addon(){
            int instrument;
            String brand;
            String serial;
            String price;
            Scanner scan = new Scanner(System.in);
            ArrayList<String> data = new ArrayList<String>();

            //listing instruments for user to see
            System.out.println("1: Trumpet");
            System.out.println("2: Trombone");
            System.out.println("3: Tuba");
            System.out.println("4: Horn");
```

```
System.out.println("5: Piano");
System.out.println("6: Keyboard");
System.out.println("7: Drumset");
System.out.println("8: Xylophone");
System.out.println("9: Saxophone");
System.out.println("10: Flute");
System.out.println("11: Clarinet");
System.out.println("12: Oboe");
System.out.println("13: Piccolo");
System.out.println("14: Bassoon");
System.out.println("15: Violin");
System.out.println("16: Viola");
System.out.println("17: Cello");
System.out.println("18: Bass");
System.out.println("19: Guitar");
System.out.println("20: Harp");

System.out.print("Type the key value of the instrument being added: ");
instrument = scan.nextInt();

//adds the key value and instrument into arraylist for you based on what you
want
switch(instrument)
{
	case 1:
		data.add("1");
		data.add("Trumpet");
		break;
	case 2:
		data.add("2");
		data.add("Trombone");
		break;
	case 3:
		data.add("3");
		data.add("Tuba");
		break;
	case 4:
		data.add("4");
		data.add("Horn");
		break;
	case 5:
		data.add("5");
		data.add("Piano");
		break;
	case 6:
		data.add("6");
```

```
            data.add("Keyboard");
            break;
case 7:
            data.add("7");
            data.add("Drumset");
            break;
case 8:
            data.add("8");
            data.add("Xylophone");
            break;
case 9:
            data.add("9");
            data.add("Saxophone");
            break;
case 10:
            data.add("10");
            data.add("Flute");
            break;
case 11:
            data.add("11");
            data.add("Clarinet");
            break;
case 12:
            data.add("12");
            data.add("Oboe");
            break;
case 13:
            data.add("13");
            data.add("Piccolo");
            break;
case 14:
            data.add("14");
            data.add("Bassoon");
            break;
case 15:
            data.add("15");
            data.add("Violin");
            break;
case 16:
            data.add("16");
            data.add("Viola");
            break;
case 17:
            data.add("17");
            data.add("Cello");
            break;
```

```
                case 18:
                        data.add("18");
                        data.add("Bass");
                        break;
                case 19:
                        data.add("19");
                        data.add("Guitar");
                        break;
                case 20:
                        data.add("20");
                        data.add("Harp");
                        break;
                default:
                        System.out.println("Instrument not found");
        }

        System.out.print("Enter the brand of the instrument: ");
        brand = scan.next();
        data.add(brand);

        System.out.print("Enter a serial number: ");
        serial = scan.next();
        data.add(serial);

        System.out.print("Enter the price: ");
        price = scan.next();
        data.add(price);
        return data;
}

protected ArrayList<ArrayList<String>> delete(ArrayList<ArrayList<String>> list){
        Scanner scan = new Scanner(System.in);
        System.out.println("Search: ");
        String search = scan.nextLine();
        for(int o = 0; o < list.size(); o++){
                if(list.get(o).contains(search) == true){
                        System.out.println(list.get(o)+" has been deleted.");
                        list.remove(list.get(o));
                        return list;
                }
        }
        return list;
}

protected void displayAll(ArrayList<ArrayList<String>> lol){
        for(int k = 0; k < lol.size(); k++)
```

```
      {
         for(int i = 1; i < lol.get(k).size(); i++)
         {
            System.out.print(lol.get(k).get(i) + "     " + "\t");
         }
         System.out.println("");
      }
          }

          protected void copy(){
                    String fileInput = "C://Users/Randell/Desktop/Java/data.txt";
                    String fileoutput = "C://Users/Randell/Desktop/Java/dataBACKUP.txt";
      try
      {
         FileReader fr=new FileReader(fileInput);
         FileWriter fw=new FileWriter(fileoutput);

         int c;
         while((c=fr.read())!=-1)
         {
            fw.write(c);
         }
         fr.close();
         fw.close();

      }
      catch(IOException e) {
         System.out.println(e);
      }

      System.out.println("Data has been backed up to a separate file.");
          }
}
public class Search {
          Scanner scan = new Scanner(System.in);

          public void sSearch(ArrayList<ArrayList<String>> list){
                    System.out.println("Search: ");
                    String search = scan.nextLine();

                    for(int o = 0; o < list.size(); o++){ //Linear Search
                              if(list.get(o).contains(search) == true){
                                        System.out.println(list.get(o));
                              }
                    }
          }
```

```
}
```