AP Computer Science

mr Hanley

Assignment 11/1011_2/13_8/B_{16} Version: 3.5

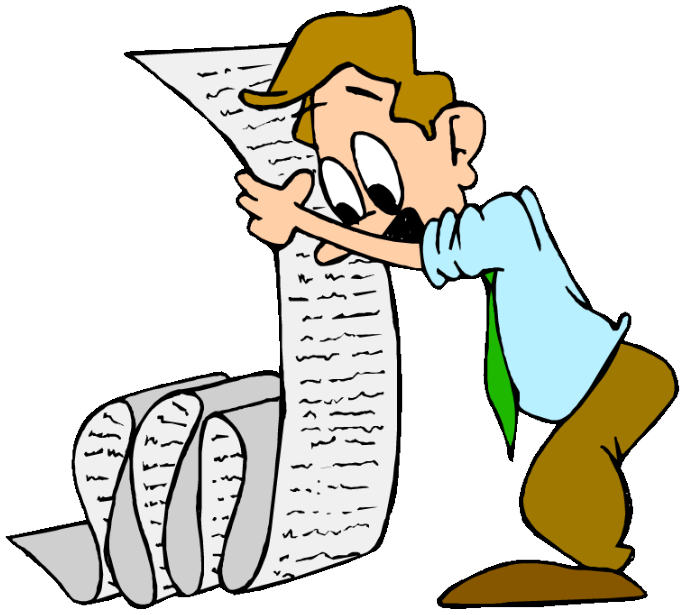
Binary							
				▨		▨	▨

Ones Complement							
▨	▨	▨	▨		▨		

Twos Complement							
▨	▨	▨	▨		▨		▨

Last Updated: 3/26/2019 12:52 PM

List Manager



Create your own list management program
Lists, lists, lists!
Playlists, friend lists, to do lists, password lists, they go on and on and on…
Fortunately, now that you are learning the art of software development, you are going to create a list manager! also knows as a CRUD (create, record, update, delete system)

Think of some sort of entity that you are going to keep track of in a list.
Examples include;

* Dynamic Memory *Big O Notation*Stacks *Extreme Programming*Selection Sort*Insertion Sort*Waterfall Model

String*Arrays*ArrayList*Client Server*Artificial Intelligence*Inheritance*Files*Video Games*Short circuit evaluation*

Colleges you are applying to, friend’s name and phone numbers and addresses, concerts attended, local restaurants, encryption methodologies, U2 songs, ski mountains, to do list, etc.

1. **Develop a class that implements the Comparable interface**

    a. This class must have at least 5 fields or data elements
    You may have only 3 Strings, other 2 FIELDS MUST BE DIFFERENT!!!

    b. While it is good OO design to have the variables private and the methods public, you may allow public variables for convenience

    c. Use good variable and method names (Charles, HiKC is not a good variable name!!!arghhhhhhhhh!!!)

    d. Provide at least 2 constructors!!!!
        i. Usually a default or zero arg which sets the values to empty or you can pick dummy fields like Jane Doe, 100 Main St. etc.
        ii. 5 arg constructor where each of the fields is passed in

    e. **Your compareTo method must be written to sort the instances of this class with some complexity**.

        i. For example, you can’t just have a compareTo method that compares names alphabetically

        ii. Use a secondary data element when the first is equal
        Example if(ranking == other.getRanking)){
                //rankings same, go to alphabetical
                return name.compareTo(other.getName())

        iii. If you want to be able to sort on multiple criteria, you may provide multiple classes that implement Comparator

        iv. In this case, at least one of your classes that implement Comparator must have some sort of complexity

    f. Comment your class

2. Develop another class that allows you to manage a list of your first class. This will have an ArrayList inside it and have methods that will;

   a. **Load the Text File**
      Read in data from a text file into the ArrayList, making an object for each "record" read in and copying each "field" into that record (this should happen each time the program starts)

   b. **Display the list**
      There are 2 ways to display the list:
      i. Print each record on its own line.
         1. Use the \t or the System.out.format command so the columns line up
         2. Put a number to the left of each record so you can refer to them individually when you edit or delete!
         3. Put this is a method so you can call it from edit or delete to select which one to choose
         4. DO NOT JUST USE A METHOD THAT PRINTS THE FIELDS SEPARATED BY A PIPE!!!!
         5. Have a heading for each field and print the actual data on individual lines underneath the heading
         6. I like to have the name of the author and the number of records loaded and the data changed boolean printed in my display

      ii. Print each record on multiple lines
         1. See the run through of Jon StreetFighter's character manager on my YouTube Channel

   c. Sort the list according to the compareTo method or multiple sorts using Comparator objects (May use Collections.sort here)

      i. There are **two** different approaches to sorting
         1. Use a compareTo for the records and sort based on some sort of complexity.
            a. For a restaurant, this could mean computing a score for the restaurant based on food rating, cost and consistency

   d. Edit a record in the list (Update the fields)
   e. Delete a record in the list

f. Save the list back to the text file **(should prompt user if they attempt to exit program without saving)**

g. Archive the list to a separate file name for safety (when working with files, it is ALWAYS a good idea to save a backup)

There are lots of different file formats you can use when writing your ArrayList to the text file

My preferred version is as follows;

```
import java.io.*;
import java.util.StringTokenizer;  //allows breaking  a String into fields

/*File could look like this
 McCarthy|Walter|255 Grapevine Rd|Wenham|MA|01984|12000.00
 NaSmith|Courtney|7 Main St.|Clifton Park|NY|12065|18000.00
 Anderson|Trinity|957 First St.|Hermosa Beach|CA|01954|19000.00
 */
```

Need to hit enter after last record in file so cursor sits on next line

```
//You read in one line of the file which represents a record, divided by a
//symbol known as the delimiter, in this case the pipe |

BufferedReader input = new BufferedReader(new FileReader("data.txt"));

    String line;
    //Attempt to read from the file, prime the pump
    line = input.readLine();
    while (line != null) {//goes to the end of file
      StringTokenizer st = new StringTokenizer(line, "|"); //| is the delimiter
      //Now break up the line
      lname = st.nextToken();
      fname = st.nextToken();
      streetAddr = st.nextToken();
      town = st.nextToken();
      state = st.nextToken();
      zip = st.nextToken();
      salary = Double.parseDouble(st.nextToken());
```

```
        System.out.println("Here's our info " + fname + " " + lname + " " +
                        streetAddr + " " + town + " " + state + " " + zip +
                        " " + salary);  //obviously you must make and add to list!!!!
        line = input.readLine();  //must be at end so when no data while condition
trips
      }
      input.close();
//That code MUST be surrounded by a try catch block
```

NOTE:Your menu must include your name and the type of list that you are managing.

Example:

```
Welcome to the Ski Mountain List Database by mr Hanley
------------------------------------------------------------------------
1 = Display Mountains
2 = Load in from Disk File
3 = Add a new Mountain
4 = Edit an existing Mountain
5 = Remove a Mountain
6 = Sort the List
7 = Save the List Back to Disk File
8 = Exit this program
------------------------------------------------------------------------
```

When someone chooses, delete or edit, MAKE SURE they know what to type, Many of the programs I have run are very UNCLEAR.

For example, in this case,

```
5
Which mountain to delete?
1 = Killington
2 = Mount Snow
3 = Gore Mtn.
4 = Sunday River Resort
3
Removing Gore Mtn!
1
#  Name              Trails      Cost        Distance     Rating
1  Killington        121         92          121 miles    8.9
2  Mount Snow        75          78          101 miles    8.1
3  Sunday River Res  89          88          211 miles    8.5
```

MAKE SURE PEOPLE CAN SAVE THEIR ARRAYLISTS FROM THE MENU WHEN THEY WANT TO MAKE A SELECTION. READING IN CAN BE DONE IN THE BEGINNING OF THE PROGRAM ONLY OR AS A MENU OPTION

3. Rakowsky Factor:
   For those of you using the Scanner to read in via nextLine and then nextInt or nextDouble, see the following code example:

```
while(inFile.hasNextLine()) {
   String tempName = inFile.nextLine();
   int tempAge = inFile.nextInt();

   …more reads
   if(inFile.hasNextLine()){
      inFile.nextLine();  //Skip the blank that the Scanner will choke on!
   }
}
```

4. You may use Swing or console for this project

5. BONUS: Encrypt your data using Blowfish encryption

Project Name	**Assign 11 – List Manager**
Class 1 Name	Restaurant.java (Example)
Class 2 Name	RestManager.java (Example)
Class 3 Name	Possible Comparator Classes (Optional)
Text File1 Name(Needs to be in project folder NOT src or classes)	Restaurants.txt
Text File2 Name(Needs to be in project folder NOT src or classes)	Restaurants.bak

Restaurants.txt text file stores information about restaurants
COULD look like this;
Forno Bistro|Italian|7.5|Moderate|Chic Atmosphere|Saratoga Springs
Bellinis|Italian|8|Moderate|Good Atmosphere|Clifton Park
….
read in at beginning of program into arraylist

write out at end of program in order to allow data to persist = persistency
write out a copy just in case to a separate file (files can become corrupted)

FoodGenre.java
```
enum FoodGenre {Italian,Chinese, American, Indian, BarbQue};
```

Restaurant.java
```
public class Restaurant implements Comparable{
        public String name;
        public FoodGenre gen;
        //etc

        public int compareTo(Object o) {  //Fixed 1/15/2016
                Restaurant other = (Restaurant) o;  //type cast so we can compare
                …

        }
}
```

RestManager.java

```
public static ArrayList<Restaurant> list = new ArrayList<Restaurant>();

public static void main() {

        //Has a menu which allows control of ArrayList
}
```
Remember to READ in all of your elements

Rubric	
Entity Class	
-compareTo	20
-good variable names with at least 5 data fields	10
main class or Frame for GUI	0
-menu and options are user friendly and easy to follow	15
	0
-reads from text file	15
-sorts based on compareTo	15
-edits	15
-adds	15
-deletes	15
-displays	15 -7 if you just use fields separated by pipes Use System.out.format with codes for field widths
-saves	15
-archives(must be a separate file)	10
-warns user if attempting to exit program without saving unsaved changes	5
-comments and variable names	10
TOTAL	175
BONUS	20

*Recursion*Linear Search*Binary Search*Grid World Case Study*File Processing *nlogn*Hangman*