**PROJECT Java File Processing Application: Java Data Analytics**

**Objective** To type a simple Java program, execute ( run ) the program for some particular values, observe the output and then modify the program.

***PROJECT DESCRIPTION***

Design an application that uses sequential file processing to analyze a list of data values.

The initial starter code for your program is given in **Figure 1** .

First run and test the starter code, which primarily has file processing objects that declare a file and write data to the file.

After you test the original starter code and verify its functionality, you will then modify the program according to the instructions that follow.

Your completed program ( after modification ) will perform, at the minimum, each of the following tasks

• read from a text file and write to a text file

• include a method that displays the data array, when requested

• include a method to sort the data in the text file

• locate the maximum and minimum values in the array

• compute the average value of the data values

Basically, your program is outlined in the given starter code statements shown within **Figure 1** , which follows. Review the starter code to understand the mechanisms of the interactions between reading a file and writing to a file. Perform any modifications according to this project’s instructions.

Type, compile and run the basic Java program that is shown in **Figure 1** , which follows. Then modify the program accordingly.

***Information About This Project***

The types of file processing include:

*sequential access file processing* data is accessed one line at a time

*random access file processing* data can be assessed in any order

***Steps to Complete This Project***

**STEP 1**  **Open NetBeans**

Open NetBeans and create a Java project with the following details.

For Project Name include **Lab13**

For the Main Class include **lab13.DataAnalytics**

In your **Code** window for this class, shown below, copy the program code shown in **Figure 1** below, in the appropriate places, except substitute your own name in place of Sammy Student.

**PROJECT Java File Processing Application: Java Data Analytics**

**Figure 1 Source Code for the Data Analytics File Processing Program**

|  |
| --- |
| **import java.io.\*;**  **import java.util.ArrayList;**  **import java.util.Scanner;**  **// Sammy Student: Data Analysis with Java File Processing**  **class DataAnalytics**  **{**  **public static void main(String args[])**  **{**  **// declare an object to receive the data**  **Scanner scan = new Scanner(System.*in*);**  **// declare an array list to hold the data**  **ArrayList<Integer> list;**  **list = new ArrayList<Integer>();**  **int count = 0;**  **int num = 0;**  **int val = 0;**  **String line = "";**  **try**  **{**  **// create or append to the file**  **FileWriter fileOut = new FileWriter("outData.txt");**  **BufferedWriter fout = new BufferedWriter(fileOut);**    **System.*out*.println("how many data items?");**  **count = scan.nextInt();**  **for (int i = 1; i <= count; i++)**  **{**  **System.*out*.println("enter a data value");**  **val = scan.nextInt();**  **fout.write(val + "\n");**  **}**  **System.*out*.println("thank you ... the data has been recorded!");**    **// close the output stream objects**  **fout.close();**  **fileOut.close();**  **scan.close();**  **}**  **catch(Exception e)**  **{**  **// catch an exception if any arises**  **System.*err*.println("Error: " + e.getMessage());**  **}**  **}**  **}** |

**PROJECT Java File Processing Application: Java Data Analytics**

**STEP 2 Build, Compile and Run the Program**

From the NetBeans Run menu select Run Project ( Lab13 ) to run your app.

**STEP 3 Test the Program and Write the Data**

Once you have successfully compiled your program enter a valid list of data values as shown below. Your initial program code will request a count of data values to be recorded and then write the data values to a text file.

**[ Program Output ]**

how many data items?

5

enter a data value

20

enter a data value

50

enter a data value

10

enter a data value

80

enter a data value

90

thank you . . . the data has been recorded!

**STEP 4 Open the Data File and Review the Contents of the File**

Within your software development environment, click [ File ] and then [ Open ] to review the contents of the data file that was created by running the program. Locate the **outData.txt** text file and open the file to view its contents.

**STEP 5 Read the Data File**

Once you have successfully compiled and tested your starter program and opened the text file with the data, you will now supplement your program with some program code that will read the data in the **outData.txt** text file and individually place the data values into a Java ArrayList .

Place the statements that are shown in **Figure 2** before the closing curly brace

" } " of the try block.

**PROJECT Java File Processing Application: Java Data Analytics**

**Figure 2 Additional Source Code for the Data Analytics File Processing Program**

|  |
| --- |
| **// read the data**  **FileReader fileIn = new FileReader("outData.txt");**  **BufferedReader fin = new BufferedReader(fileIn);**    **while ((line = fin.readLine()) != null)**  **{**  **num = Integer.*parseInt*(line);**  **list.add(num);**  **System.*out*.println(num);**  **}**  **System.*out*.println("thank you ... the data has been received!");**  **fin.close();**  **fileIn.close();** |

Save your program and run the program with the sample data that was given earlier. Observe the output of your program.

**STEP 6 Supplement the Program Code Statements**

Supplement again the program by including a method that displays the data that was recorded into the Java ArrayList .

Within your program, create a method named DisplayData() , which has this signature and definition. Place the method below the main() method and before the closing curly brace " } " of the class definition.

**public static void DisplayData(ArrayList<Integer> num)**

**{**

**for (int i = 0; i < num.size(); i++)**

**System.*out*.println(num.get(i).toString());**

**}**

Then, before the closing curly brace " } " of the try block, place these two statements.

**System.*out*.println("display unsorted data");**

***DisplayData*(list);**

Save your program and run the program with the sample data that was given earlier.

Now include a method that will sort the data that was placed into the ArrayList . Place the method below the DisplayData() method and before the closing curly brace " } " of the class definition.

Use the Bubble Sort routine that follows in **Figure 3** .

In an appropriate place in the main() method include these lines of code to show that the data has been sorted.

**System.*out*.println("display sorted data");**

***DisplayData*(list);**

**PROJECT Java File Processing Application: Java Data Analytics**

**Figure 3 Source Code for the Data Analytics File Processing Program**

|  |
| --- |
| **public static void BubbleSort(ArrayList<Integer> num)**  **{**  **int j = 0;**  **boolean flag = true; // set the flag to true to begin first pass**  **int temp = 0; // define the holding variable**  **while (flag)**  **{**  **flag = false; //set flag to false awaiting a possible swap**  **for (j = 0; j < num.size() - 1; j++)**  **{**  **if (num.get(j) > num.get(j + 1))**  **// for descending sort change to <**  **{**  **temp = num.get(j); //swap the elements**  **num.set(j, num.get(j + 1));**  **num.set(j + 1, temp);**  **flag = true; //shows a swap occurred**  **}**  **}**  **}**  **}** |

**STEP 7 Modify the Program**

After testing your program that it displays the original data in an unsorted order and a sorted order, modify again the program to include each of these variations:

• Add a new method named MaxMin() that will find the smallest and largest value in the ArrayList after the data has been sorted.

• Add a new method named Average() that will find the average value in the ArrayList after the data has been sorted.

Save your program and perform a trial run of it. Test your program with data similar to that shown below.

**[ Program Output ]**

how many data items?

5

enter a data value

20

enter a data value

50

enter a data value

10

enter a data value

60

enter a data value

40

thank you . . . the data has been recorded!

**PROJECT Java File Processing Application: Java Data Analytics**

**STEP 8 Submit Your Project**

Once you have determined that your modified program is correctly displaying the required information, complete the submission process as follows:

Open MS Word and type a heading for a new document that includes your full name, course number, lab number and date.

Within the document paste snapshots of your modified program in action. Label the snapshots of your modified run with a reasonable description.

After your snapshot, paste in your finished source code as well copied in from your NetBeans editor.

Submit your MS Word document to Blackboard when complete.