CSC-240 Lesson 11: Files, Streams and Object Serialization

Assignment 11

Please do the two programming projects described below. These are not projects from the textbook, so please read the instructions carefully.

Project One

For this project, write a program, **Summarize** (`Summarize.java`), containing the *main* method, that first writes 10,000 random positive **double** type numbers, to the full accuracy of the number (15/16 decimal places), to a text file named **DataValues.txt**, one number per line. The program must then close that file, and reopen it for reading. Read back the values from the file and write the following information to a file named `Summary.txt`:

- Your name;
- The class name and section number;
- The current date/time (*i.e.*, the date when the **Summarize** program is being run—which also means that the date/time when the Instructor runs that program should be the date seen);
- The total number of values in the file (which should be exactly the number you generated, but count them anyway);
- The average of the values in the file (*i.e.*, the sum, divided by the count of numbers);
- The largest value in the file; and,
- The smallest value in the file (these last three values being shown to 14/15 decimal places).

Please use as your random number generator a **Random** class object, declared and initialized as follows:

```
private static final Random RANDOM = new Random(-1L);
```

and using the method *nextDouble* to get the actual double values.

You may determine the presentation of the information in the output file, but the end result should be clear and easy to read (see sample below).

This exercise only uses a relatively small number of values written to and read from the file. However, the usual purpose of using file storage for such values is to be able to compute results on very large collections of values that would be a problem to keep in memory. So, in doing this exercise, please do not accumulate the individual data values into a memory structure: process them one at a time, and collect just the summary values needed for the final report.

Please submit at least three files for this part of the assignement:

- `Summarize.java`—the program that writes and then reads the text file of values and prints summary information about those values, plus any other subsidiary Java files that your solution may need;
- `DataValues.txt`—the file containing the numbers; and,
- `Summary.txt`—the file containing the summary information.

Project Two

This is an exercise in using both serialized input/output and plain text output. For this project you need to write a program that creates at least ***five*** **Employee** objects (you should use the class files provided through the link below, which are essentially the textbook examples from Chapter Eight (Classes and Objects: A Deeper Look)) and write serializations of those objects to a file named `Employees.dat`, using ***Java Object Serialization***. Make sure to vary the data values of the **Employee** objects.

Your program should then re-open that file for reading and should read back those objects and write their values to a simple text file named `Employees.txt`. You may use the *toString* method of the Employee class to format the data for that file.

The two Java files, `Employee.java` (similar to the one from Chapter Eight, see Figure 8.8 (Employee class with references to other objects)) and `Date.java` (similar to the one from Chapter Eight, see Figure 8.7 (Date class declaration)), are available in this zipfile, together with the corresponding Javadoc.

Additionally, your program needs to demonstrate the following:

- When you read the **Employee** objects back, do not assume you know how many objects are in the file. Use a loop that continues to read objects until the read throws an **EOFException** (just like in Figure 15.11 (Reading a file of objects sequentially with **ObjectInputStream** and displaying each record)).
- When you catch the **EOFException**, properly close the input object file and the output text file.

Please submit three files for this part of the assignment:

- `Employee.java`—an updated version of the above class file that supports Java object serialization;
- `Date.java`—an updated version of the above class file that supports Java object serialization;
- `EmployeeDat.java`—the program that implements this programming project, which writes and reads the file of **Employee** objects serialized using Java object serialization.

Sample Results

Here is an *example* of running a solution to the first program and viewing the resulting file (yours will be different, as it has different parameters specified):

```
Dr. Bruce K. Haddon
CSC-240-500 Java Programming
Sun Aug 12 19:37:18 MDT 2018
Count of values=13500
Sum of values=6753.881337433932000
Average=0.500287506476588
Maximum=0.999878150004852
Minimum=0.000040467287184
```

Here is an example of running a solution to the second program:

```
Creating and writing Employee objects
Employee Values--as written initially
Writing Employee objects to file "Employee.dat"
 1 Flint, Mike  Hired: 6/15/2005  Birthday: 1/2/1965
 2 Flint, Donna  Hired: 6/15/2006  Birthday: 1/2/1968
 3 Flint, George  Hired: 6/15/2007  Birthday: 1/2/1972
 4 Flint, Norma  Hired: 6/15/2002  Birthday: 1/2/1983
 5 Flint, Abby  Hired: 6/15/2006  Birthday: 1/2/1955
 6 Flint, John  Hired: 6/15/2002  Birthday: 1/2/1959
Reading Employee objects from file "Employee.dat"
 1 Flint, Mike  Hired: 6/15/2005  Birthday: 1/2/1965
 2 Flint, Donna  Hired: 6/15/2006  Birthday: 1/2/1968
 3 Flint, George  Hired: 6/15/2007  Birthday: 1/2/1972
 4 Flint, Norma  Hired: 6/15/2002  Birthday: 1/2/1983
 5 Flint, Abby  Hired: 6/15/2006  Birthday: 1/2/1955
 6 Flint, John  Hired: 6/15/2002  Birthday: 1/2/1959
```

> The following *About the Assignment* topic contains further explanation, directions, and hints about this assignment. Please read that topic before starting this assignment.

Important

Please follow the directions of Programming Assignment Identification in submitting your programming solutions.

If you have any questions or concerns about the Assignment or the *About the Assignment* topic, please use the Lesson Question discussion topic, or send a message by the D2L Internal Messaging system if you prefer.

Note: you may submit to the Assignment Submission Folders as many times and as often as you wish, up to the deadline time. Each submission is tagged with the date/time, and so each submission remains separate and distinct. Unless you leave instructions to the contrary, only the most recent of each file with the same name will be viewed for the purposes of grading. Details may be found in the topic *How To Submit and Get Feedback on Assignments* in the *How To* module.

Messages that accompany Assignment Submissions are read, and responded to, ***only*** when assignment submissions are graded (which is after the Assignment Submission Folder closing date/time). If you have a comment or question about an assignment, or a request for assistance, that needs an earlier response, then that comment, question or request should be made or asked *via* an Internal Message or the Discussion board, as these are usually read and answered every day.