R•I•T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences & Technology

ISTE-121 Computational Problem Solving for the Information Domain 2

Lab 5 – Threads & Byte I/Os

Problem Statement:

Sample files of city, state, zip code and other information were broken down into several smaller files for emailing. These are stored in several sequentially numbered binary files within **Lab5Data.zip** that you can download from MyCourses. Your job is to write a program that reads as many files as exist. As you read each record load only the city names into an ArrayList, and write a subset of what you read in into another file containing only a few fields from the record. Input and output file details are given below.

You are writing this program to show the feasibility of the processing above, which will be done on a much larger scale with world wide city, state and postal codes. For speed, you are to create one thread per file being read. For example: the filenames are in the format Lab5FileN.dat, where N starts with 1. So for the first file you create a thread object that reads in Lab5File1.dat, file number 2 a thread is created to process Lab5File2.dat, and so on until your program discovers that some file does not exist.

Once the files are all read in, the ArrayList is created, and the output file ZipCityState.dat is created, write out how many cities were stored in the ArrayList, and how many records were written to the new binary file.

Input Datafile layout:

Neither input nor output files contain any heading information, only the data shown.

Each input file of the format **Lab5FileN.dat** has this binary file layout.

Field contains	Data Type
Zip	Integer
City	UTF
State	UTF
Longitude	Double
Latitudes	Double
Time Zone	Integer
DST – Daylight savings time observed	Integer

Write the ArrayList object as an object to file **CityArrayList.ob**. Display the size of the file as described below.

The output binary file **ZipCityState.dat** is to contain only this information. The other information is ignored and not used.

Field contains	**Data Type**
Zip	Integer
City	UTF
State	UTF

Requirements:

- The number of files is unknown, so the program needs to handle this unknown
- Create a thread for each input Lab5FileN.dat file being processed by the program
- As each thread reads its Lab5FileN.dat file:
    - The city name, not the state, is stored into an ArrayList. You must use an ArrayList, not any other collection, and synchronization must take place as to not lose any Cities.
    - The Zip, City and State are written into ZipCItyState.dat. Synchronization separate from the ArrayList must take place here, along with counting how many records are being written to this file. Must use a counter defined common to all threads.
    - When the thread is finished reading its file, print a formatted line containing the filename and record count. A sample is shown here:
      ```
      File Lab5File5.dat completed, record count is   4,950
      File Lab5File2.dat completed, record count is   9,431
      ```
- After all thread finish, the code waiting for the treads to complete is to print:
    - Count of how many cities were loaded into the ArrayList
    - Count of how many records were written to the ZipCityState.dat file
    - Write the ArrayList object to file CityArrayList.ob and then print how many bytes the file contains.
      ```
         Loaded list size has 43,191 cities.
         ZipCityState.dat has 43,191 cities.
         CityArrayList.ob is 500,406 bytes.
      ```
- Following this, take the city names from the command line and check if they can be found in the ArrayList. This is more for testing purposes for later programming.
    - Any number of city names can be requested, or none.
    - If no cities are on the command line, no messages should display.
    - For each city or word on the command line, show that it was found, by displaying the city name and position in the ArrayList. Using ArrayList's indexOf() method might work well for this. For example on the command line:
      ```
         java Lab5 Rochester buffalo Albany "Salt Lake City" abcde
      ```
      The output following the counts and bytes might be:
       (notice lower case city, buffalo & enclosing a multi word city in "…"):
      ```
         Rochester       was found at position   5,456
         buffalo         was NOT found
         Albany          was found at position   2,777
         Salt Lake City  was found at position   8,742
         abcde           was NOT found
      ```

Hints:

The programming difficulty and time of this Lab5 and the Practicum 1 can be greatly reduced by typing into a blank "public class Lab5 { … }" file the steps that are needed to complete this program.

Not to give anything away, but for example you know there is a main method, a class where you will put the threaded code, and other constructors. Don't type those, but only comments such as // main, and all the rest of the high level steps to program this problem. Then start writing details under each major comment. Keep doing this until you see there will only be code and no further refinement. Then you will be ready to write code very fast and in an organized (verses disorganized) manner.

The one hint that may give something away is, depending on how you write your program; you may have to use a Lab5 constructor. Look for the *static context* error.

Dropbox:

Send to the dropbox your best version of Lab5, even if it was checked off in the class.

Instructor or T/A: ______________________ (remember to dropbox)

R•I•T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences & Technology

ISTE-121 – Lab 5 – Threads Byte I/Os (Practice practical)

Item	**Possible points**	**Earned points**
Driver code:		
Start one thread per file found	10	
Wait for all threads to finish	10	
Properly end use of ZipCityState.dat file	5	
Print size of ArrayList	5	
Print number of records written to ZipCityState.dat	5	
Write ArrayList as an object to CityArrayList.ob	5	
Prints correct size of the CityArrayList.ob file	5	
Take command line city names, print position, or not found	10	
Threaded code:		
Opens this thread's data file for reading	5	
Properly reads the data items	10	
Properly places city in array list	5	
Properly writes ZipCityState.dat file	10	
Prints this file's name and record count	5	
Code works as expected	10	
Deductions: Coding standard violations, etc.		
Total:	100	0

Comments: