CS 351 Data Organization and Management

FALL 2011 - Programming Project #1

Union of Sequential Files

Due: November 3, 2011 at 11:59 PM

In this project you are asked to implement a Java console application which aims to find the union of two piles (F1 and F2) and write it to an output text file (F12). For this purpose, you must think as if you have limited memory so that you are able to understand the intuition behind processing block-by-block reading and writing in a practical way. Therefore, the memory capacity is much less than the sizes of input files.

You are not given any particular algorithm for finding the union; however, you are expected to write an algorithm which efficiently finds the union (i.e. as minimum read/write operations as possible). You are also supposed to write a short report which explains your algorithm. Moreover, record your execution time for finding the union, and state it in the report. Look at "How to time your Execution?" section to record your time.

Please read the following specifications carefully before starting to your implementation.

1. File Structures

Both input files consist of student records which have the following format:

Student ID	Student Name	Department

Each record is uniquely identified by its student id field. Henceforth there are no duplicate records in each input file.

A record as a whole can be read into a string (i.e. you can read records line by line from the input files).

You are supposed to do read/write file operations in Java as explained in the next section.

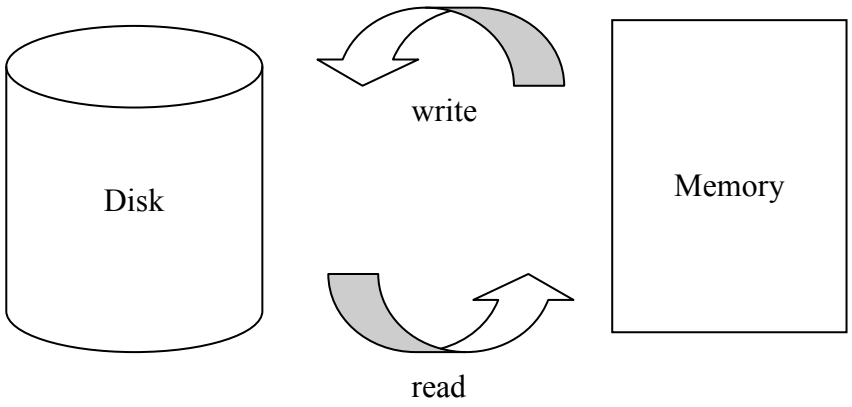
You can find sample input files in the link below:

http://cs.bilkent.edu.tr/~ctoraman/cs351fall11/project1/samples.rar

2. Memory

Suppose that the memory holds at most 100 records. In other words, you can think the memory as an array of 100 elements and you must use only that portion while processing the input files and writing the output file.

0	1	……..…	99



Read the following explanation for analogy. Input files are placed into the disk, and to be able to read them, you must fetch a block of records into the memory. Since you have two input files, you should think carefully for what portion of memory you use for F1 and F2 to process the files efficiently. Note that the files may contain different number of records. You are also supposed to use the memory to write to the output file.

2. Tutorial: How to R/W File in Java

In Java, there are various ways to handle r/w a file; however, you must use the following way. Other implementations will **not** be accepted.

You have to use BufferedReader and BufferedWriter classes to read from and write to a file, respectively. Consider using the following methods of BufferedReader:

- BufferedReader(Reader in): Opens file for reading. The argument is an instance of Reader class. See Java API for more detailed information.
- readLine( ): Reads a line from the file.
- close( ): Closes the file.

Use the following methods of BufferedWriter:

- BufferedWriter(Writer out): Opens file for writing. The argument is an instance of Writer class. See Java API for details.
- write(String s): Writes a string to the file.
- close( ): Closes the file.

3. How to Time your Execution?

```
private static long startTime, endTime;

public static void startTimer(){
    startTime = System.currentTimeMillis();
}

public static void endTimer(){
    endTime = System.currentTimeMillis();
}

public static long duration(){
    return endTime - startTime;
}

public static void reset(){
    startTime = 0;
    endTime = 0;
}
```

The following code segment calculates execution time by finding the difference between two different time points.

```
public static void main(String[] args) throws IOException {
    System.out.println("PROCESSING...");
    reset();
    startTimer();
    findUnion();
    endTimer();
    long time = duration();
    System.out.println("PROCESS COMPLETED SUCCESSFULLY.");
    System.out.println("Duration: " + time + " ms");
}
```

4. General & Submission Rules

- Name your program as “FileUnion.java”.
- You are free how to test your implementation. We will use our own method to grade your projects. Therefore do not put any essential code segments into your main method.
- The input files will always be stdlist1.txt and stdlist2.txt. The output file which your program creates at the end of execution should be named as union.txt. The output file should be placed in the same folder that your Java project folder exists.
- Note that we will test your programs with input files different than those given to you. Additionally, we plan to use a supportive tool to be able to detect plagiarism.
- Name your report as “**StudentID.doc**” (e.g. 20802629.doc).
- Put your program and report into a **rar** file. Do not put any other file.
- Name your rar file as **StudentID_cs351p1.rar** (e.g. 20802629_cs351p1.rar). Files with other formats will be **ignored** automatically.
- Use the submission page to upload your rar file:
  http://cs.bilkent.edu.tr/~ctoraman/cs351fall11/project1/submission/
  Read the upload rules in the submission page as well. Late submissions will **not** be accepted!
- You can resubmit until the due date and the previous file will be replaced.
- For questions, contact with hayrettin@cs.bilkent.edu.tr