CS 246	Software Design and Development

Quickness

Emergency Backup Recovery *Exploration*

Goal

The purpose of this exploration is for you to explore certain characteristics of the java language and I/O library, learn how to learn, and do so collaboratively and **quickly**.

Background

"**HELP!!**" Your doctor friend called, his voice drenched with desperation. "You've got to help me recover my database files!" You feel keenly obligated, since after all, you were the one to write the Medical Office Management software on which his practice depends. He's usually very good at making backups of his data, but something happened this time that he is powerless to fix.

Somehow amidst your friend's frantic cries for help, you get the gist of what happened. He entered the command `copy *.dbf backup` and instead of copying all DBF files to a "backup" directory, it made one big file called `backup` containing all the DBF files "smooshed" together. This would happen if somehow the backup directory was removed, which is apparently what happened, because then the copy command would treat `backup` as the destination file name. To make matters much worse, Murphy's Law kicked in to make all the original DBF files somehow lose their contents and become empty!

Fortunately, he could tell you the names of the seven individual DBF files inside the one "backup" file, because, of course, that information was lost when the file was made. (Unlike a .zip or a .jar file, for instance, the path/filename information for contained files is not stored in the containing file itself.) Unfortunately, you don't know how big each file was before being "nullified", which means you would have to determine the boundary between one file and the next some other way.

Requirements

Your task is to successfully split the "backup" file into the seven original DBF files. To help you meet your deadline (your friend needs his data **now** of course) a `SplitDBF.java` stub file is provided. You must quickly get familiar with the **java.io.RandomAccessFile** class, and use an instance of that class in your hasty implementation of the `getDataAndOffsets` method.

Hints

Google "DBF header format" and apply the **gersy principle** — but don't try to be too meticulous, you just don't have time!

See the `main` method for an example of using `RandomAccessFile` for writing. Of course, it should go without saying that your `getDataAndOffsets` method should only be using `RandomAccessFile` for reading.

Use the following file-size information to verify that your `getDataAndOffsets` method works correctly:

```
    73  acctno.dbf
  1468  check.dbf
    73  claimno.dbf
 13800  dicode.dbf
1044600 policy.dbf
120090  recall.dbf
 45911  trcode.dbf
```

Use the `backup` file attached, i.e., save it to disk and pass a path to that file as your command-line argument. (For your convenience, this file is also available at `/home/cs246/files/backup` in the Linux lab, and is the default name of the input file if you merely invoke `java SplitDBF` with no command-line argument.)

Grading Criteria

42	Split the backup file into the 7 named files (6 points each).
14	Got the 7 files *exactly* correct (all or nothing, 2 points each).
14	Used `RandomAccessFile` efficiently in `getDataAndOffsets`.
20	Applied the **gersy principle**, avoided the **gersy gotchas**.
10	Collaborated, and identified collaborators.

Comments

For your consideration, here are some comments past CS246 students have made about this exploration:

- I liked how different this exploration was, it wasn't just a normal assignment with set instructions. Still, I would have liked it better if I had done something like it before, instead of it being my first time.
- This was the first time that I actually was able to use the information of a header. I have studied TCP headers, IP headers, and other packet types, but I have never had an occasion to actually do my own code to do anything with it. I also enjoyed the challenge of doing something that I could see myself actually using in my work.
- Well, I don't like working with code I didn't write. I enjoy knowing exactly what is going on in MY code, I take ownership of it, and often when trying to figure out someone else's methods of doing things it is difficult. I think that in a career of programming that you would frequently have to do this. I believe that this class will assist me in overcoming this and help me to work better with others, but it isn't easy for me.
- I liked doing some "low-level" stuff with Java. Previously I've been under the impression that Java wouldn't allow for system level file access or even bit level work. I also liked the challenge of not knowing exactly how to do something and having to research it online. This is exactly how it was for me at work, so I felt like we weren't just doing busy work. It was good practice.
- I enjoyed the fact that we started with already written code - many times that allows learning the same things faster.
- I did like the concept of the program and the accompanying mock-real-world scenario. The story gave it some sort of context of "why am I doing this?"
- I had a good time figuring out this exploration. It took me a little while to figure out that this assignment was much less complicated than I realized. It always feels good to get a program to work right. The most useful thing I found out was when I realized that the entire file was read into the "buffer" array. That made everything else easier.

- I enjoyed this exploration quite a bit. It did take me a little bit to get going and fully understand the procedures necessary to find a solution. I know that the collaboration with other classmates really helped out a lot.
- After finishing the project in the computer lab it was fun talking with other people and seeing how they got their solutions.
- This exploration taught me that sometimes the answer is right in front of me but that I am trying to see beyond the horizon. After completing this assignment I can already think of ways to make it better.
- This wasn't a keep-your-code-reusable kind of assignment, it was more like a quick hack which is the kind of real-world code we may have to write sometimes.
- I liked learning on my own rather than having every step of the process spoon-fed to me. In this assignment I had to go out and do a bit of research on my own. Again, that's a much more real-world kind of thing.
- I liked the collaboration aspect though I probably should have done more of it. That part has real-world written all over it!
- I didn't like the part of this assignment where I went way off in the wrong direction before getting it right. Maybe a bit more collaboration would have prevented this though.
- Well, I have a tendancy to over-think things, but this assignment only took me an hour and a half, more than half of which was spent looking at the DBF format and trying to understand it.
- This assignment brings new meaning to the term: "You're thinking too hard." Well I thought so hard I suffered indigestion. And then I calmed down and remembered the **gersy principle**!
- This exploration was definitely an exercise in finding the simplest solution in the shortest amount of time; something that many of us already struggle with. I like that this exploration forced us to think quickly and find quick solutions. And I really like the real-world applications of what we were doing.
- This assignment was an interesting one, it didn't work for me until I shared what I was trying and then that student got it to work then I also got it to work. The solution was simple once I understood what truly was needed.
- It was nice having a "real" problem to solve. You always get these assignments that don't always seem practical. Having to split someones smooshed files apart is a real thing. However, I didn't like having to delete all my files while debugging, due to the behavior I observed that once the .dbf files are created their size doesn't shrink, they only grow (if needed).