1. README: It is quite simple to install this program. All one needs to do is go to the Assignment Uploader and through some teacher magic, one should be able to get all of my files, including my .java files, .class files, and all of my resource files. Once all of these files are downloaded, you can open them using any java IDE. I recommend IntelliJ because that’s what I coded in and it has the greatest probability of success. Once the files are opened, navigate to the .java file called JsonParsing and hit the run button to start the program. If you experience any difficulty opening or running this file, please don’t hesitate to email me at [grant.hugh@lakesideschool.org](mailto:grant.hugh@lakesideschool.org)
2. You have the right to look at the files in any order you want, but I would recommend starting with the text resource files that end in “Test” to see what the data I am parsing in looks like. Notice that while all of the files are in the JSON format, the JSONObjects within the JSONArrays are quite different and need to be collated in odd manners due to the missing information in some of the data. After looking through the test text files, you can look at the real thing and get a sense for how big that is. Once you have looked through the text files, you should have a pretty good sense of what JSON is all about if you were unfamiliar with that beforehand and now you can look at the Collate file. The Collate file collates all of the Excel spreadsheets, which was not a simple task. Next, you can look at the Query file. The Query file does most of the main work. Finally, you should look at the JsonParsing file, which puts all of these things together. I don’t think you need to look at the Inputter file because it is basically what we did in CSII plus a couple of additions for additional data types.
3. Again, you have the right to perform any tests other than shutting down your computer, which is guaranteed to crash my program. The program will take a while to start because of the amount of data it has to read in. The stuff you have to type in is a bit odd since I started to run out of time, but I will explain it clearly right here:
   1. First, you will be asked for the Initial Data Set. You can try typing in random stuff here and my program should be able to re-prompt you. Once you feel like you are satisfied with my error-handling, I would recommend typing in T, for total data set.
   2. Next, you will be prompted for a parameter name. Again, you can try typing random stuff but you will be re-prompted for an actual parameter. Once you feel satisfied with keyboard mashing, I would recommend typing in “Help” (without quotes) to get a list of all the possible parameters that you can type in. You can choose any parameter you like from this list. Just type in the first part; ignore everything after “Type:”. For example, my favorite query is GradeSpan.
   3. I’ll assume you entered in GradeSpan. If you didn’t, hopefully the program will be intuitive enough so that you’ll understand. If you entered GradeSpan or any other parameter with a String value, you’ll be prompted with a value. If you type in something random, such as “I don’t believe in the grade system that society imposes on us”, the program will tell you that there are no schools that satisfy your specified query and it’ll just make you reenter a parameter. This gives you the opportunity to actually enter in GradeSpan. Once you do that, you’ll again be prompted for a value; this time, enter in “9-12”, without the quotes.
   4. You’ll be prompted for another parameter. Again, you can mash your keyboard, have your pet lizard crawl across your keyboard for a random string of characters, but alas, my program will not crash (hopefully, if it does crash, please email me at [grant.hugh@lakesideschool.org](mailto:grant.hugh@lakesideschool.org)). This time, you should type in TotalEnrollment.
   5. My program will now ask for a lower bound on the parameter. I think you’re probably bored of randomly mashing your keyboard by now; if I assume wrongly, go ahead and do it. If you are bored, great! We can move on more quickly now. Type in 500.
   6. Type in 5000. If you’re really ambitious, try typing in 50000. Ooohhh
   7. My program will once again ask you for a parameter. I think we’re bored of parameters now though. Type in “end”.
   8. Now you will be prompted for a field you want to keep. I will give you a little sovereignty now:
      1. You can enter in “all”, in which case all parameters will be kept
      2. Or you can enter in “help” and then choose the parameters you want to keep. Same as last time, don’t enter in the stuff after and including “Type: “.
      3. Or, if you really want to be difficult, you can once again mash your keyboard. You must really like mashing your keyboard Dr. Bricker…
   9. The prompt will ask you if you want to print the dataset. Please don’t mash your keyboard on this one; just type in “print”. If you type in something else, you will miss out on a one in a lifetime opportunity of watching JSONObjects get printed.
   10. After applauding at the spectacular show, you are asked if you want to convert the file. If you want to convert the file, please type in “convert”. Please don’t mash your keyboard on this one either; it’s a hit or miss, and if you miss, you have to start over at the very beginning, all the way up at a. You don’t want to do that. Type it in carefully.
   11. Now you will be prompted for a file name. Choose a school appropriate file name.
   12. Now you will be prompted for a directory name. You will need to add an extra \\ to each \ in your directory. For example, I used the directory C:\\Users\\Grant.Hugh\\Desktop\\School\_Tests
   13. You will be asked if you want to do any additional queries. I think you would be pretty tired at this point; after all, you went from a-m. If you are truly obsessed with my program you could type in “yes” but I would mash on the keyboard, forcing the program to break. Just kidding, it just stops if you type in anything other than “yes” or “Yes”.
   14. You can look in the directory you saved your file in, and there should be a new file. Yay! You can open this file using Excel by opening Excel, selecting Open in the menu, and letting Excel open any file, not just .xml files. Afterwards, walk through the wizard and make sure you select “Delimited” and “Comma” as one of your delimiters.
   15. Make cool graphs, do whatever you want in Excel. You are officially done.

Grading Rubric:

**\_\_\_\_Total project grade –**

**\_\_\_ Late penalty (days \* 2 pts each)**

**\_\_\_Correctness:** Does the program compile without any errors? Does the program run without any errors? Are all exceptions handled properly?

0 == More than one major error

2 == one major error

4 == Some exceptions not handled, but they were difficult to raise

6 == One or two exceptions not handled, but they were difficult to raise

8 == one to three minor problems

10 == no errors found (typos don’t count as errors.)

Comments:

**\_\_\_\_Program Design:** Procedural decomposition is natural: easy to understand and follow. No methods are extremely long without reason. Recursion is used properly. Global variables are instantiated only when necessary.

0 == unacceptable structure: One extremely long method, magic numbers scattered everywhere, makes Dr. Bricker wonder if she is teaching well enough

2 == poorly structured: A few long methods, magic numbers show up, long-winded solutions.

4 == reasonably well structured but uses many global variables or magic numbers when unnecessary. Program is divided into methods, but the divisions don’t make much sense. Scanner initialized in main.

6 == reasonably well structured but needs improvement. May use some magic numbers but methods divisions are reasonable and scanner is not initialized in main.

8 == some insufficiently justified violations but no serious problems. Scanner is not initialized in main. There are still some places which could be refactored.

10 == all small, coherent, independent modules unless well justified. Everything that should be refactored is refactored and procedural decomposition is nearly flawless.

Comments:

**\_\_\_\_User Friendliness:** This program might be used by students in math classes who are unfamiliar with programming. When the program **is paired with the README,** (I will probably be making a handout explaining how to use my program for the students) is it easy to understand and use? Can you do everything that you want to do by combining this program with Excel? Is there minimal typing involved?

0 == unacceptable; interface is extremely confusing, user is forced to type pargraph after paragraph.

2 == difficult to use; it is possible to make the program do what I want it to do, but it feels painstaking and I would not want this to be assigned as homework

4 == generally good, but still a lot of typing involved. Also, it seems as if the user needs a background in computer science to use the program well

6 == generally good, but several imperfections. When the user messes up, there is no re-prompting and the user is forced to stop the program and restart it.

8 == pretty good, user doesn’t need to be familiar with computer science, everything works fluidly and prompts are easily understandable with README. One or two areas which the user is a little confused about and might have to ask the teacher.

10 == perfect. Easy homework for the user. They might even enjoy using the program a little (highly unlikely) Maybe they’ll enjoy reading the README

Comments:

**\_\_\_\_Efficiency:** Efficiency is pretty important given how much data is involved. There is not really a good workaround to the time it takes to load the data, but the queries should be instantaneous as well as the file conversions.

0 == grossly inefficient, unnecessary repetition of steps, etc. I’ve waited hours for my query to finally go through and then it gave me an exception.

2 == some clearly unnecessary steps were performed and code is often repeated. The user has to wait a minute before a query goes through.

4 == any reasonable method for implementation was used. The queries go through after a small delay.

6 == Good methods for implementation were used with some minor problems. Queries go through almost instantaneously.

8 == Good methods for implementation were used. JSON is used correctly and the premade methods have been used extensively. Queries go through basically instantaneously.

10 == Very efficient programming methods were used correctly, while the code continued to be understandable. Seems to have mastered JSON and the conversions to CSV files. Queries go through instantaneously.

Comments: