CS5010, Fall 2019

Assignment 6

Abi Evans

ab.evans@northeastern.edu

- **This assignment is due by 6pm on Monday, November 11. Submissions should be made using your group repo.**

Assignment objectives:

This assignment builds on your concurrent solution from Assignment 5, incorporating data from more of the OULAD and adding additional analyses.

Resources

You might find the following online resource useful while tackling this assignment:

- Java 8 Tutorial: Concurrency

Submission Requirements

Gradle

The Gradle requirements are the same as for the previous assignments.

GitHub and branches

You will be building on your Assignment 5 code but you should still create a new folder for Assignment 6. As with Assignment 5, each individual group member should create their own branch while working on this assignment. Only merge to the master branch when you're confident that your code is working well. You may submit pull requests and merge to master as often as you like but you should not tag your TA until you are ready to submit. Only one person will need to tag the TA when you're ready to submit your finished assignment.

New: Please include a README.md file that will help your grader understand your code. List the entry point to your program (for example, where to find the main() method) and give a high level description of the purpose of key classes and/or methods. You should not need to write a

lot...just enough to help your grader follow your thinking. You may also want to include any assumptions you made about the nature of the problem and the steps you took to ensure correctness.

The Problem

The work you did on Assignment 5 will make it possible to identify high activity days across module presentations. However, to enable any serious data analysis to take place, the raw click totals generated in Assignment 5 will need to be normalized. For example, a larger course may have more high activity days simply because it has more students. For this assignment, you will transform the data to make it possible to compare activity across courses and find out if the click activity increases around the time an assignment is due.

You will need to process three files from the Open University Learning Analytics Dataset (OULAD): courses.csv, assessments.csv, and studentVle.csv. **Your program should accept the same command line inputs and output the same files**. The contents of the click count summary files will change but you should have the same number of files with the same names.

Again, your solution must have producer and consumer threads running concurrently, and communicating via some suitable shared data structure. Producers and consumers should be loosely coupled and communicate only through some queue or channel.

You will need to add three new columns to the click count summary files generated in part 2 of Assignment 5. You should still create a file for each code_presentation of a code_module using the same naming convention as in Assignment 5. This time, however, the columns in the generated files should be: date, normalized_date, total_clicks, relative_clicks, and assessment_type:

- date is the date relative to the start of the course (same as Assignment 5).
- normalized_date is the date normalized by the length of the course. This is calculated as date divided by the length of the course, given by the module_presentation_length column in courses.csv. For example, if the course is 280 days long, the normalized_date for date 0 would be 0 / 280 = 0 and the normalized_date for date 280 would be 280 / 280 = 1.
- total_clicks is the total number of clicks that occurred in the presentation across all id_sites and all id_students (same as Assignment 5).
- relative_clicks is the day's clicks relative to the daily average number of clicks for the presentation. To calculate this metric, you will need:
    - The active period of the presentation, which is the first date that clicks were recorded subtracted from the length of the course. For example, if the first date that clicks were recorded was -10 and the length of the course is 280 days, the active period is 280 - -10 = 290 days.
    - The total number of clicks in the presentation. This is the sum of all clicks across all days for all students.

    - The daily average number of clicks. This is the total number of clicks in the presentation divided by the active period of the presentation.
    - The relative clicks can then be calculated as the total_clicks for the day divided by the daily average number of clicks. For example, if the average daily clicks for a presentation is 100 and the total_clicks for a given date is 82, then the relative_clicks will be 0.82.
- assessment_type is the type of an assessment taking place on the given date in the given module presentation. You can find the assessment information in assessments.csv. You only need to populate this field if there was an assessment on the given date. For example, code_presentation "2013J" of code_module "AAA" had a "TMA" assessment on day 19. So, for date 19 in the file AAA_2013J.csv, the value of assessment_type should be "TMA". There was no assessment on date 18, so the assessment_type entry would be left blank for date 18.

A key difference between this assignment and the previous assignment is that you now need to process all three csv files in order to write the summary files. You should have a producer thread for each file that needs to be read in. You will need to consider how best to design your consumer(s). Remember that you can have multiple sets of producers and consumers and that they are not limited to file I/O.

Your main() should process the command line arguments. Then, it should create all the threads and necessary communication mechanisms between threads. Once everything has been constructed, your program should record a "start" timestamp and start all threads simultaneously. Finally, when all threads are completed, record an "end" timestamp and **print out how long in seconds your program took to run**. The actual processing time will be dependent on your hardware as well as your design—we are not looking for specific numbers when it comes to speed, just an estimation of how long your program takes.