CS6326 Human-Computer Interactions

Fall 2022

Assignment: Multithreaded Text Search

Write a program that searches a text file for a string. It must find all occurrences of the string and show them in a list. Since the text file could be very long, and lists in C# and JavaFX do not display until the button-press code that fills them exits, your program must be multithreaded. The program should do the following:

1. Show a textbox for the name of a text file. You can also have a Browse button that will let you find a file to put in the textbox. This should bring up a FileOpenDialog or equivalent.
2. Another textbox will accept text to be found in the document.
3. A search button will cause the document to be searched.
4. A ListView control displays the entire line on which the text was found and the line number within the document. The lines aren’t numbered, so you have to keep track. Line numbers in the real world start at one, not zero. Blank lines count.
5. Search each line of text **as you read it** for the search string. You should do a case-insensitive comparison. Every time the string is found, put the entire line in which it occurs in the list along with its line number. To make it interesting, put in **a pause of 1 millisecond** every time you read a line. This will simulate a long document. The program should read the entire file exactly **once**.
6. The Search button text should be changed to “Cancel” once the search starts, and should cancel the operation when pressed.
7. Add any other features to this program that you think may make it more usable or understandable to the average computer user. The screen shown below is a possible starting point, and will not receive full points. (Do not e-mail me to ask if this or that feature is allowable. Use your knowledge of design, and material provided, to make the determination. In-class questions and discussion are welcome, however.)
8. You might need to use the .Net Queue class for this, since the secondary thread might find things faster than the main thread can display them. Here is a link <https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentqueue-1?view=net-5.0>

You don’t need to use regular expressions to do the comparison; a simple string search will suffice.

You do not need to use delegates; you can do everything using BackgroundWorker’s functions. Never interact with the UI directly from a secondary thread.

I have provided a long text file that is complete text of *The Divine Comedy* (Inferno, Purgatorio, and Paradiso) by Dante Alighieri. Searching for words like *paradise, fantasy, love, human, gold,* and others should yield interesting results.The document contains approximately 18,000 lines of text, so with the delay it should take over a minute to search. (My version takes nearly 4 minutes.)

You have your choice of language. I have covered C# BackgroundWorker in class, but you can also use SwingWorker in Java, which is very similar.

A possible partial screen design is shown below.

**To hand in via eLearning:** Hand in a Zip file named with your NetID followed by Asg4. For example, mine would be jxc064000Asg4. If you wrote in C#, it should contain your entire Visual Studio project. If you wrote in java, hand in only your java source files.

|  |  |
| --- | --- |
| **Grading: 100 points possible** | |
| Program is truly multithreaded and works according to the above specification | 50 |
| Program follows other UI design rules and guidelines | 20 |
| Additional features for usability/learnability and immediate versus eventual honesty | 20 |
| Program comments and good variable names | 10 |

Grading guidelines:

1. The program reads everything into memory, then searches an array. -50
2. The program waits until it finishes to display anything. -50
3. The program reads the entire file once to determine the line count, then uses that for a progress bar. -50.
4. The program uses only the screen below as is, without any further changes: -20
5. Screen is fixed size, not full-screen to show as much information as possible. -10
6. Search misses items that should be found. -10
7. The program doesn’t allow you to cancel the search. -10
8. The program can be made to crash for invalid input such as a bad file name. -10

