DM 519 - concurrent Programming

Troels P. Have
E-mail: trhav17@student.sdu.dk
Supervisor: Fabrizio Montesi

May 7, 2018

Contents

1 The methodology

1.1 The tasks

The three assignments in this project was to:

1. Recursively visit all text files with a ".txt" suffix in a given directory and its subdirectories. The method was to return a list of all the files and the lowest number found in the file.
2. Recursively visit all text files with a ".dat" suffix in a given directory and its subdirectories. The method looks for a file that contains a line where the numbers added together amount to at least the given parameter, `min`. When the method encounters such a file, the method needs to return immediately.
3. Computes overall statistics of the occurrences of numbers in a given directory. The statistics of interrest are: The number of occurrences of a given number, `n`, The most, as well as the least frequent number, and a list of the files, ordered by the total value in the files.

1.2 The methodology

The program consists of three classes, respectively named, `assignment1`, `assignment2` & `assignment3`, which all have their own constructor. They are all being given a path-parameter, but `assignment2` also receive a integer-parameter, `min`.

The first assignment, `m1`, calls the `dial()` function, that recursively visits all subdirectories, and opens all text files with a ".txt" suffix as a new instance of the first constuctor, `assignment1`. The constructor calls the `lowestNumber`-functions that finds the smallest number in the file, and add it to the list of Results, from the `Result`-interface required.

The second assignment, `m2`, calls the function, `dial_2()`, which recursivelt visits all subdirectories and opens all text files with a ".dat" suffix as a new instance of the second constructor, `assignment2`. The second constructor calls the function, `find`, that for each line in the document, adds the numbers together and checks whether the total value amounts to the value, `min`. If that conditions is reached, the method returns the line number, as well as the path.

The third assignment, `m3`, calls the function, `dial_3()`, that recursively visits all subdirectories and opens all text files with either a ".txt" or a ".dat" suffix as a new instance of the third constructor, `assignment3`. The constructor calls the function `DictMaker`, that adds the values of each number on every line to a ConcurrentHashMap, that counts the instance of each number. When the dictionary is completed, the `m3` method implements the methods from the interface `Stats`.

For this code to become concurrent, the constructors of each assignment implements the Runnable-interface, and that way, each time a `dial`-function is called, it gets treated as a new thread. To manage the thread, an executorService, workstealingPool, is constructed. To make sure the gathered results in the methods are thread safe, the block of code where its added is synchronized on that list.

2 Advantages

One of the advantages of this program is the readability. The way of using small, concise private functions and synchronizing single blocks of code, gives the reader a nice overview of the code, instead of cramming everything into one, giant block of chaoticness.

The way of using a workStealingPool, is also an advantage, due to the fact that it handles recursive functions so well. The WorkStealinPool uses parallelism and is much faster than a FixedThreadPool, that takes a fixed amount of threads and queues work when that number of threads is used. instead, a WorkStealingPool dynamically uses the threads and finds balance that way.

Further more, to ensure the best- and fastest results, the program scales, according the what number of processors the computer has. By defining the variable, `cores`, the program adapts to the processing power and ensures maximum speed.

3 Limitations

To handle my threads, I was already from the start going to use an executorService, and the natural choice fell on a FixedThreadPool. However after some time i discovered that it handled recursion very poorly. And since a big part of the project is recursion, i then switched to a WorkStealingPool instead.

Another flaw of the program is that it does not handle a number of errors, or exceptions. For example if the program gets a directory with no ".txt" or ".dat" files in it, the program will keep running forever, doing nothing, due to the fact that the executorService never is created, but never shut down, however, this could probably be handled with a try-catch statement.

Furthermore, if one were to give the program a directory with text files, containing other than numbers, or "," characters, the program would give yet another exception error. this, too could probably be fixed with another try-catch statement.