Assignment 4 – Concurrency

SEG2106 – Software Construction

Due: April 9th at 11:59 PM

Background

You are developing software for a genetics research laboratory called *GenResearch*. The scientists at this lab want to be able to search for particular nucleotide or peptide sequences in the human genome. The human genome at their disposal is encoded in FASTA format. If the last few sentences sound like mumbo jumbo to you, do not worry, you do not need to be a biologist to solve this assignment. You simply need to be a Java programmer.

Chromosomes

As you might have studied back in high school, human cells have 23 pairs of chromosomes (Figure 1). The genetic information pertaining to humans is encoded in these chromosomes.

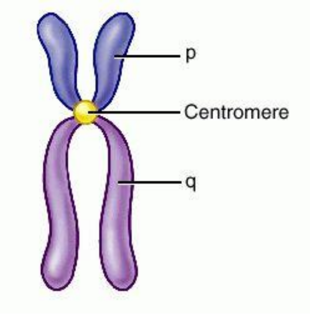

Figure 1 - Nice little diagram of a chromosome

FASTA Format

The DNA sequence of an organism can be described in plain text using various formats. FASTA format is one of the popular ones. It specifies sequences of amino acids using single-letter codes. Therefore, for your purpose, **you can suppose that a FASTA file is simply a text file that contains a bunch of letters.**

Part A

The human genome in question is stored in a folder, where the genetic code for each chromosome is stored in a separate FASTA file. The scientists at *GenResearch* have developed a small Java program that searches for a particular sequence in the human genome and specifies where it was found (which chromosome and at which line in the corresponding FASTA file). The Java program (**given with this assignment**) requires an argument which specifies the path of the folder containing the FASTA files.

In order to get the FASTA files, perform the following:

1. Go to the following link: http://www.site.uottawa.ca/~halosman/SEG2106/
2. Download the file: **HumanGenome.zip** ***(make sure you have enough space on your disk as the file size is huge: 944 MB)***
3. Extract the **.zip** file ***(again, the extracted folder is even bigger, around 2.93 GB)***

Once you have the **HumanGenome** folder extracted, you can use the Java program (associated with this assignment) as follows:

1. Specify the **path** of the HumanGenome folder (**the folder containing the .fa files**) as an argument to the program
2. Run the program
3. When the program prompts you for a pattern to look for in the files, enter the following string: **gagaccactctggcca**
   (For more possible sequences that you can look for, check the appendix at the end.)
4. Observe the results on the screen

Now that you can run the program, study its code (especially the **Main**, **SearchTask** and **SearchJob** classes). Note that you do not have to understand the **KMP** class; it suffices to know that it provides a search method to find a particular pattern in a string and returns the index of that pattern within the string if it finds it; otherwise, it returns -1.

> **SIDE NOTE:**
> *For those of you who are interested in the KMP class, it provides an implementation of the Knuth Morris Pratt algorithm that you have seen in your data structures course.*

Part B

Exercise 1

Instead of performing the search on all the files in one Thread, you will use a multi-threaded approach. Therefore, change the given code so that you run one Thread for every search job (one thread per file). Create a **ThreadGroup** to keep track of all your threads, and make sure all of them complete their execution before you exit. You might want to take a look at the **ThreadGroup** Javadoc: http://docs.oracle.com/javase/7/docs/api/java/lang/ThreadGroup.html

Exercise 2

The previous exercise assigns a single search job for every Thread. This results in a large number of Threads. In this exercise, you will limit the size of your **ThreadGroup to 4 threads**. Start by assigning a search job for every one of these Threads. Whenever they are all done, assign to them the next **4** search jobs. Keep performing this operation until you run out of search jobs.

Good Luck!

Appendix

Search Patterns

The following are patterns that you can look for in the FASTA files (using the Java Program).

- **tcccactcttattat**
- **AGACGCCCACCTACGAGCAAAGT**
- **aagtttgataaatttctggtac**
- **atcaccactaaagaacttatccatgtaactaaacaccacctgttccccaa**
- **ATTTCAGATTGACTCTGG**
- **TGAACAAATAATTCATCTGAAACATTCAGGCAA**