COP 5618/CIS 4930 HW2

Due: Nov 7 at 11:50pm

Suppose that there are three kinds of operations on singly-linked list: search, insert and remove. Searching threads merely examine the list; hence they can execute concurrently with each other. Inserting threads add new items to the front of the list; insertions must be mutually exclusive to preclude two inserters from inserting new items at about the same time. However, one insert can proceed in parallel with any number of searches. Finally, the remove operation removes items from anywhere in the list. At most one thread can remove items at a time, and a remove must also be mutually exclusive with searches and inserts.

Start with the skeleton class in ConcurrentSearcherList.java. Your task is to modify the class to allow access by multiple threads according to the description above. To help you design your solution, answer questions 1-4 as a pencil and paper exercise:

1. Which variables will you add and what is their interpretation and initial value?
2. Give an invariant that captures the synchronization requirements above? Give your invariant as a logical formula and express it in English. (Hint: Make sure that your invariant allows for no threads accessing the list. You may recall from discussions of student solutions in class when we were looking at the readers/writers problem that forgetting this corner case is a common error.)
3. Provide implementations for start_search, end_search, start_insert, end_insert, start_remove, and end_remove *in terms of atomic_await statements*.
4. The start_search, and start_insert methods should allow an inserter to execute concurrently with searchers. If you convert your atomic await solution to a legal Java implementation are there conflicting operations within the provided bodies of the search and insert methods? Do these conflicts create data races? Are new Nodes created by the insert method safely published?
5. Modify the given class in accordance with your answers above. You may add additional variables, add keywords, and modify the body of the constructor (but not its signature). You should not need to modify class Node. There is a solution that does not require modification of the public methods insert, remove, and search. Your solution should not have data races.
6. (COP5618 students only) Think about whether this class is likely to be useful in practice. Briefly discuss its pros and cons.

You should, of course, create tests for your solution. However, in this assignment we will not collect or evaluate your test cases.

Turn in:
1. Your answers to Questions 1-4, and 6 if applicable.
2. One uncompressed Java file called ConcurrentSearcherList.java containing your solution for Question 5. (Use assignment HW2_code to turn in)