Homework 1

Concurrency

For this homework you will take an existing Java program (in Homework1.java) that is multi-threaded and make it “thread-safe,” meaning it runs correctly when the threads execute concurrently.

The program creates and prints a client list, where each client is represented by a name and an id. Five names (Albert, Bernice, Charlotte, Dedrick, and Edwin) are added by one thread, and six names (Ursula, Veronica, Willard, Xavier, Yolanda, and Zachary) are added by the other thread.

The threads run concurrently, so in the final list, each thread’s names appear in alphabetic order, but names from the two threads can be interleaved. Id numbers are assigned sequentially (by 10s) and each client has a unique id. Here is an example of what the correct output could look like:

Final list:

Client Albert has id 1000

Client Bernice has id 1010

Client Charlotte has id 1020

Client Ursula has id 1030

Client Veronica has id 1040

Client Willard has id 1050

Client Dedrick has id 1060

Client Xavier has id 1070

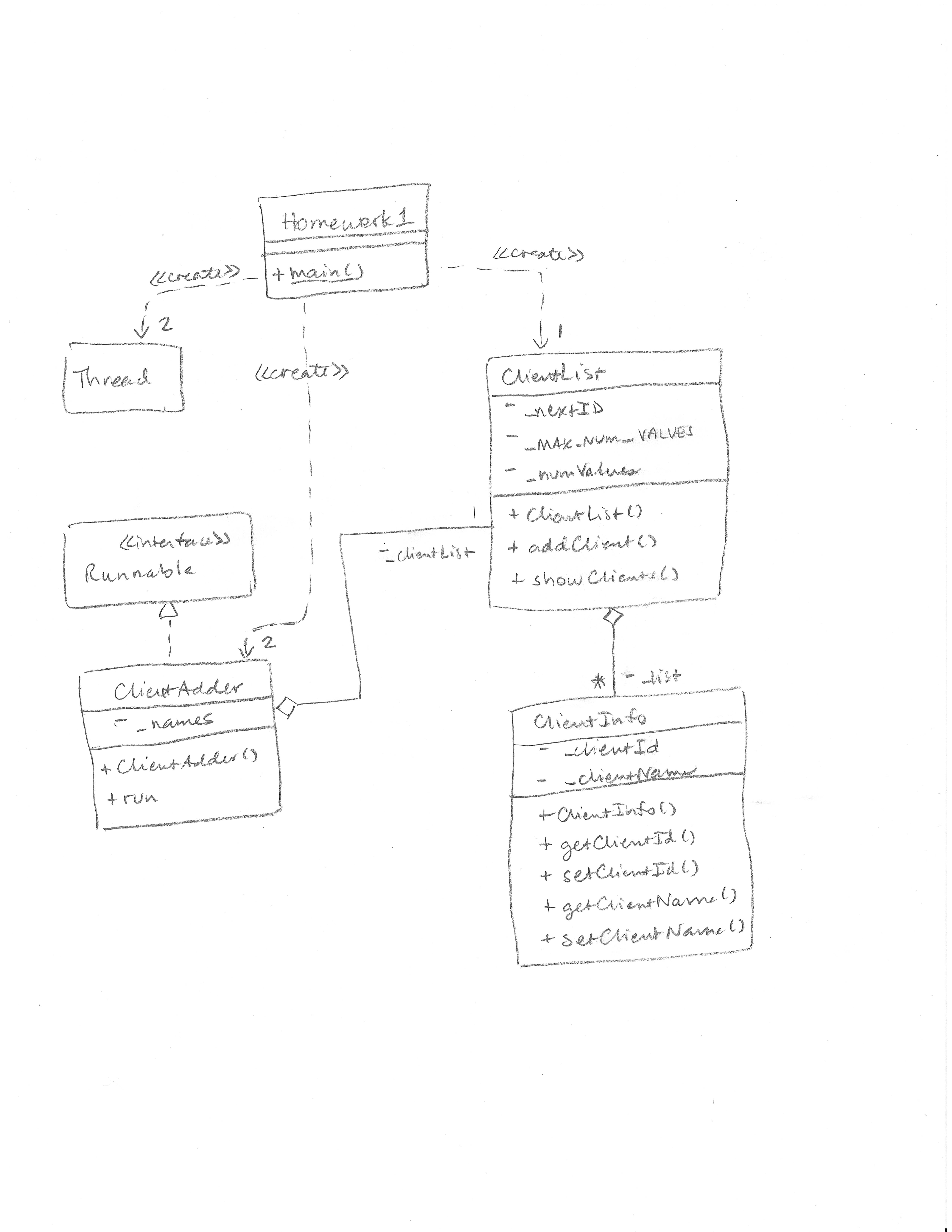
Client Yolanda has id 1080

Client Edwin has id 1090

Client Zachary has id 1100

The classes and relationships are shown in the diagram on the next page. The main() in class Homework1 creates an empty ClientList object. Next it creates a ClientAdder for the first five names, then associates that ClientAdder with a thread. It does the same for a second ClientAdder (with the last six names).

Next the main() starts both threads. When a thread runs, its ClientAddr adds the names in \_names to the ClientList. When both threads have completed, the main() prints the contents of the final ClientList object.



Currently the output of the program is this:

Final list:

Client Albert has id 1000

Client Bernice has id 1010

Client Charlotte has id 1020

Client Dedrick has id 1030

Client Edwin has id 1040

Client Ursula has id 1050

Client Veronica has id 1060

Client Willard has id 1070

Client Xavier has id 1080

Client Yolanda has id 1090

Client Zachary has id 1100

This output looks correct, but the program is not correct. Your task involves three steps:

1. Determine where the concurrency problems are.
2. Insert “Sleep” statements to expose the concurrency errors. (Keep these in permanently.)
3. Use Java’s concurrency mechanisms to fix the errors.

Step 2 is important because you won’t know whether you’ve solved the concurrency problem unless you can reveal the bad behavior. Here is an example of what your output could look like after step 2 but before step 3:

Final list:

Client Albert has id 1000

Client Bernice has id 1010

Client Charlotte has id 1030

Client Veronica has id 1030

No ClientInfo object.

Client Dedrick has id 1040

Client Willard has id 1040

No ClientInfo object.

Client Xavier has id 1050

Client Yolanda has id 1060

Client Zachary has id 1070

Notice that some of the client entries in the list have been overwritten (leaving others empty), and some clients have the same id. Remember that you’re looking for shared resources—resources that both threads use, and you need to make sure that each thread has exclusive access to the shared resources when it is updating them.

The program you turn in should be the corrected program, with the sleep statements intact. I’ll run it first as is, then I’ll remove your concurrency fixes (step 3) to see that you’ve exposed the bad behavior (step 2).