Übung 07: Threads, Streaming und Networking

Abgabetermin: 7. 5. 2014, 8:15

Name: _____ **Matrikelnummer**: _____

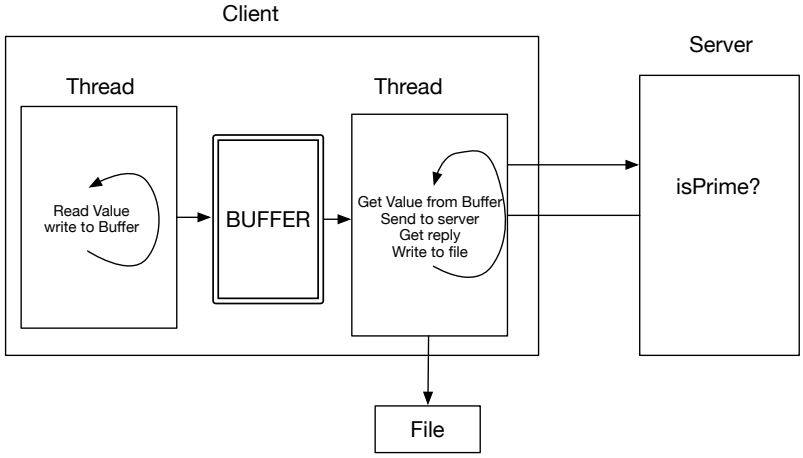
Informatik: ❑ G1 (Prähofer) ❑ G2 (Prähofer) ❑ G3 (Grimmer) ❑ G4 (Grimmer)

WIN: ❑ G1 (Khalil) ❑ G2 (Kusel) ❑ G3 (Kusel)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 6	24	Java-Programm, Ausgabe eines Ablaufs	Java-Programm	❑	

Network-based Producer-Consumer Example

In this exercise you should implement a client-server system for processing numbers. The client reads numbers from the console, writes them into a buffer, where they are retrieved to be sent to a server over the internet, the server checks the number if it is a prime, sends the response back to the client, and the client stores the result into a file. This program has multiple components, should use multiple threads, needs synchronization, and works with socket streaming as follows:



Server:

The server receives values from a client, tries to convert them to an integer, checks whether the integer value is a prime number (we provide you a `superSlowIsPrimeImplementation` function, which is really slow), and returns a reply, e.g. `"3 is a prime".`

Client:

The client has a small text-based user interface. The user can enter numbers, which are eventually sent to the server.

It consists of two threads: one thread that reads values from the terminal and puts them into a buffer; a second thread takes a value out of the buffer, sends it to the server, reads the server's reply, and finally writes the server's reply to a file.

Include a few test-runs (inputs, output, file content) to your submission.

Details

1) *Server*:
    a. Start a server (read the port number from the terminal)
    b. Wait for a client to connect
    c. Until the server receives 'x' from the client:
        i. Try to parse the received value to an integer
        ii. Test if the number is a prime (use the function `superSlowIsPrimeImplementation`)
            1. If the value is a prime: Return "XXX is a prime number"
            2. If the value is not a prime: Return "XXX is not a prime number"
            3. If the value is not a number: Return "XXX is not an integer"
    d. Ask if server should wait for another client.
2) Client:
    a. Connect to the server (read the IP and the port number from the terminal)
    b. Create a new Buffer
    c. Start 2 threads:
        i. Reader thread:
            1. Until the user inputs 'x':
                a. Read the input from the terminal
                b. Put the input into the buffer
            2. *Hint*: also put the terminating symbol 'x' into the buffer (the server will stop processing values once it received symbol 'x').
            3. Terminate thread

            Hint: this implementation needs synchronization!

            Hint: this thread only asks the user for a new input if the buffer is not full.

            Hint: you can assume that the user is patient and only enters a number if he/she is asked for an input (see example output)
        ii. Sender thread:
            1. Until you get an 'x' from the buffer:
                a. Get a value from the buffer
                b. Send it to the server
                c. Get the reply and write it to a file

            Hint: this implementation needs synchronization!

            Hint: If the reader thread terminates and the buffer is not empty, the sender thread needs to process the remaining values in the buffer

3) Buffer:
    a. Internally your buffer is an array of Strings (size 3).
    b. Add methods to put/get values to/from the buffer.
    c. Add methods to check isFull/isEmpty.

    Hint: Buffer needs synchronization!

Method for checking for prime number:

```
private static boolean superSlowIsPrimeImplementation(int n) {
	try {
		Thread.sleep(10000);
	} catch (InterruptedException e) {
	}
	if (n <= 1) {
		return false;
	}
	if (n == 2) {
		return true;
	}
	for (int i = 2; i <= Math.sqrt(n) + 1; i++) {
		if (n % i == 0) {
			return false;
		}
	}
	return true;
}
```

Sample session:

```
>java Server
Port number: 2222
Waiting for client request
++ Connected to client
Wait for next client? (y | n):
n


Output:
5 is a prime number
9 is not a prime number
8 is not a prime number
4 is not a prime number
7 is a prime number
mrks is not a number
13 is a prime number
```

```
Server IP: localhost
Port number: 2222
++ Connected to server
Enter 'x' to exit
Enter number: 5
Enter number: 9
Enter number: 8
// Buffer is full, you need to wait. You can assume that the user does not enter a value while waiting
Enter number: 4
Enter number: 7
Enter number: mrks
Enter number: 13
Enter number: x
Exiting...
// It took a few moments until the sender thread processed all remaining values.
```