МІНИСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Дніпровський національний університет
залізничного транспорту ім. академіка В. Лазаряна

Кафедра КІТ

ЛАБОРАТОРНА РОБОТА № 12

з дисципліни: «Операційні системи»

на тему: «Створення багатопотокових серверних застосувань»

Виконав: студент групи ПЗ1712

Нікольський О.В.

Приняла: Нежуміра О.І.

Дніпро, 2020

Тема: Створення багатопотокових серверних застосувань.

Мета роботи:

1. Написати програми, які реалізують взаємодію клієнтів і сервера через сокети з використанням потоків;

2. Клаєнт повинен надіслати дані серверу, а сервер обробити отримані дані і надіслати результат клієнтові.

Опис специфікацій

SimpleThreadServer.java	
Методи	
run	Викликає методи GetClientsStreams() та GetArrayFromClient(); є перевантаженим методом, який можна використовувати як поток
GetClientsStreams	Отримує вхідний/вихідний потоки клієнтського сокету
Sort	Вхід – невідсортований масив Вихід – відсортований масив Виконує сортування масиву
GetArrayFromClient	Отримує масив від клієнта
SendArrayToClient	Вхід – відсортований масив Надсилає відсортований масив клієнту
Поля	
Socket client	Клієнтський сокет
OutputStream clientOutputStream	Поток виводу клієнтського сокету
InputStream clientInputStream	Поток вводу клієнтського сокету

ConnectionFactory.java	
Методи	
ConnectionFactory	Вхід: threadCount – к-сть потоків, що буде запущена у пулі; port – номер порту для запуску сервера. Ініціалізує пул потоків розміром <threadCount> та запускає сервер на порту <port>
ServerWork	Слухає порт, при підключеннія нового клієнта передає його задачу в чергу до пулу потоків
Поля	
ServerSocket server	Серверний сокет
Socket client	Клієнтський сокет
ExecutorService threadPool	Пул потоків

Код програми

ConnectionFactory

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ConnectionFactory {

    ConnectionFactory(int threadCount, int port) throws IOException {
        threadPool = Executors.newFixedThreadPool(threadCount);
        server = new ServerSocket(port);
    }

    public void ServerWork() throws IOException {
        while(!server.isClosed()){
            client = server.accept();
            threadPool.execute(new SimpleThreadServer(client));
        }
    }

    ServerSocket server;
    Socket client;
    ExecutorService threadPool;
}
```

SimpleServerThread

```
import java.io.*;
import java.net.Socket;
import java.util.Arrays;

public class SimpleThreadServer implements Runnable {

    SimpleThreadServer(Socket cl) {
        SimpleThreadServer.client = cl;
    }

    @Override
    public void run() {
        while (!client.isClosed()) {
            try{
                GetClientsStreams();
                GetArrayFromClient();
                Thread.sleep(1000);
                break;
            } catch (Exception e){
                System.err.println("Error: " + e.getMessage());
```

```
        }
    }
}

public void GetClientsStreams() throws IOException {
    clientInputStream = client.getInputStream();
    clientOutputStream = client.getOutputStream();
    System.out.println("Got client's streams");
}

private int[] Sort(int[] array) {
    Arrays.sort(array);
    return array;
}

public void GetArrayFromClient() throws IOException, ClassNotFoundException {
    ObjectInputStream inputStream = new ObjectInputStream(clientInputStream);
    int[] array = (int[]) inputStream.readObject();
    System.out.println("Received array: ");
    for (int count = 0; count < array.length; ++count) {
        System.out.print(array[count] + " ");
    }
    SendArrayToClient(array);
}

private void SendArrayToClient(int[] array) throws IOException {
    ObjectOutputStream outputStream = new ObjectOutputStream(clientOutputStream);
    outputStream.writeObject(Sort(array));
    outputStream.flush();
    System.out.println("\nSending array to client");
}

private static Socket client;
private OutputStream clientOutputStream;
private InputStream clientInputStream;
```

Висновки

Отримане завдання реалізовано мовою програмування Java з використанням пулу потоків, який має зменшити навантаження на сервер. Пул потоків — структура, що дозволяє використовувати обмежену к-сть потоків для необмеженої к-сті завдань, ставлячи їх в чергу обробки. До того ж, потоки у пулі не завершуються, а чекають, поки до них не буде передане завдання; створюються лише на початку роботи пула, що значно зменшує використовувані ресурси.