**Проект по мрежово програмиране**

Изработен от Цветелина Илиева

ФН: 82232

Тема: Клиент-сървър приложение + паралелен quicksort

Проектът представлява клиент-сървър приложение на Java. Клиентът изпраща на сървъра числа, които иска да бъдат сортирани. След това сървърът сортира паралелно тези числа като използва алгоритъма quicksort и накрая връща резултата на клиента. Именно това са трите главни части на проекта.

**1.Сървър**

Първо създаваме ServerSocket, чрез който да се останови връзка между клиента и сървъра. След това изчакваме да се свърже клиент и създаваме сокет. След като имаме свързан клиент взимаме input и output стриймовете на сокета съответно за получаване и изпращане на информация от/към клиента. След това в безкраен цикъл започваме да чакаме някакво съобщение от клиента. Зависимо какво е то изпълняваме 4 различни стъпки. Ако то е „exit“ затваряме сокета се прекратява програмата. Ако то е число го запазваме в масив, който ще бъде сортиран. Ако то е „sort“ сортираме досега събрания масив и го изпращаме на клиента обратно сортиран. Ако не е нито едно от изброените просто продължаваме програмата, чакайки друго съобщение.

**2.Клиент**

Първо се свързваме със Сървъра използвайки сокет и взимаме output и input стриймовете на сокета съответно за получаване и изпращане на информация от/към сървъра. След което изпращаме на сървъра колкото числа искаме да сортира и с изпращането на команда “sort” получаваме обратно сортиран масив от изпратените числа. При изпращане на команда „exit“ към сървъра сокета се затваря и се прекратява програмата.

**3.Паралелен quicksort**

За реализирането на паралелния quicksort създаваме ForkJoinPool, който ще използваме за изпълнението на задачи. За целта ни е нужен клас който да репрезентира тези задачи-QuicksortTask. Compute метода разделя масива на 2 части. Едната с всички елементи по-малки от pivot другата със всички елементи по-големи от pivot. След което се създават две задачи – лява и дясна. leftTask.fork() метода съобщава на pool-а лявата задача да бъде изпълнена паралелно с дясната. След това rightTask.compute() изпълнява дясната задача на сегашния thread. И най накрая leftTask.join() казва на текущия тред да изчака лявата задача да приключи преди да продължи изпълнението на програмата.с