Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу “Программные на языке Java”

на тему: “ **Сетевое взаимодействие в Java**”

Выполнил студент группы 22ВВП1:

Аргаткин А.А.

Горбатов К.В.

Коробкин В.Ю.

Приняли:

Юрова О.В.

Карамышева Н.С.

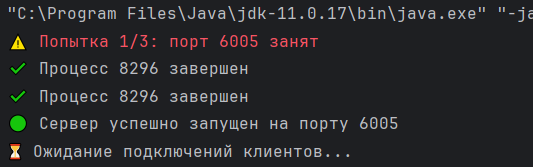
Пенза 2024

**Цель работы**:научиться создавать клиент-серверные приложения c использованием стандартных классов Java.

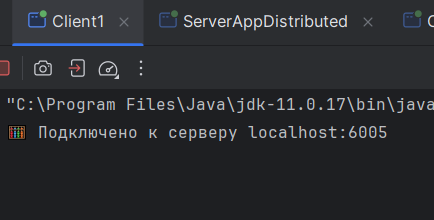
**Вариант: №1**

**Ход работы:**

Запуск сервера:

****

Работа клиента:

****

**Листинг:**

Сервер:

import network.IntegralRequest;  
import network.IntegralResponse;  
  
import java.io.\*;  
import java.net.InetSocketAddress;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
import static integrals.IntegrationCalculator.integrateSimpson;  
  
public class ServerApp {  
 public static void main(String[] args) {  
 final int PORT = 6005;  
 System.out.println("🔵 Сервер запущен...");  
  
 try (ServerSocket serverSocket = new ServerSocket()) {  
 // Настройка возможности повторного использования адреса  
 serverSocket.setReuseAddress(true);  
 // Привязка к порту с таймаутом  
 serverSocket.bind(new InetSocketAddress(PORT), 100);  
  
 System.out.println("🟢 Сервер слушает порт " + PORT);  
  
 while (true) {  
 Socket socket = serverSocket.accept();  
 System.out.println("🟢 Клиент подключился");  
  
 try {  
 ObjectInputStream in = new ObjectInputStream(socket.getInputStream());  
 ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());  
  
 IntegralRequest request = (IntegralRequest) in.readObject();  
 double result = integrateSinX2Parallel(request.a, request.b, request.h);  
  
 out.writeObject(new IntegralResponse(result));  
 } catch (Exception e) {  
 System.err.println("Ошибка обработки клиента: " + e.getMessage());  
 } finally {  
 try {  
 socket.close();  
 } catch (IOException e) {  
 System.err.println("Ошибка закрытия сокета: " + e.getMessage());  
 }  
 }  
 }  
 } catch (Exception e) {  
 System.err.println("Фатальная ошибка сервера: " + e.getMessage());  
 e.printStackTrace();  
 }  
 }  
  
 public static double integrateSinX2Parallel(double a, double b, double h) throws InterruptedException {  
 int NUM\_THREADS = 8;  
 double stepSize = (b - a) / NUM\_THREADS;  
 Thread[] threads = new Thread[NUM\_THREADS];  
 double[] partialResults = new double[NUM\_THREADS];  
  
 for (int i = 0; i < NUM\_THREADS; i++) {  
 final int index = i;  
 double localA = a + index \* stepSize;  
 double localB = (index == NUM\_THREADS - 1) ? b : localA + stepSize;  
  
 threads[i] = new Thread(() -> {  
 partialResults[index] = integrateSimpson(localA, localB, h);  
 });  
  
 threads[i].start();  
 }  
  
 for (Thread thread : threads) {  
 thread.join();  
 }  
  
 double total = 0;  
 for (double res : partialResults) {  
 total += res;  
 }  
  
 return total;  
 }  
}

Клиент:

package distributed;  
  
import network.IntegralRequest;  
import network.IntegralResponse;  
  
import java.io.\*;  
import java.net.Socket;  
  
public class Client1 {  
 private static final String HOST = "localhost";  
 private static final int PORT = 6005;  
  
 public static void main(String[] args) {  
 try (Socket socket = new Socket(HOST, PORT)) {  
 System.out.println("🧮 Подключено к серверу " + HOST + ":" + PORT);  
  
 ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());  
 ObjectInputStream in = new ObjectInputStream(socket.getInputStream());  
  
 IntegralRequest request = (IntegralRequest) in.readObject();  
 System.out.printf("📤 Получено задание: [%.2f, %.2f] h=%.5f\n",  
 request.a, request.b, request.h);  
  
 double result = calculateIntegral(request.a, request.b, request.h);  
 out.writeObject(new IntegralResponse(result));  
  
 System.out.printf("✅ Результат отправлен: %.8f\n", result);  
  
 } catch (Exception e) {  
 System.err.println("Ошибка клиента: " + e.getMessage());  
 e.printStackTrace();  
 }  
 }  
  
 private static double calculateIntegral(double a, double b, double h) {  
 // Реализация метода Симпсона  
 int n = (int) ((b - a) / h);  
 if (n % 2 != 0) n++;  
 double sum = 0;  
  
 for (int i = 0; i <= n; i++) {  
 double x = a + i \* h;  
 double fx = Math.sin(x \* x);  
 if (i == 0 || i == n) {  
 sum += fx;  
 } else if (i % 2 == 0) {  
 sum += 2 \* fx;  
 } else {  
 sum += 4 \* fx;  
 }  
 }  
 return h / 3 \* sum;  
 }  
}

**Вывод**: научились создавать клиент-серверные приложения c использованием стандартных классов Java.