Міністерство освіти та науки України

Черкаський політехнічний фаховий коледж

**Звіт**

З Практичної роботи №1

Виконав студент групи ПС 4-1

Кудь Д.О.

Перевірив викладач:

Поломаренко Н.В.

2021 р.

**Індивідуальне завдання**

Організувати інформаційну систему за типом клієнт-серверної

архітектури на базі сокетної технології, де на сервері виконується обробка

запитів від клієнтів відповідно варіанту (таблиця 1.1), причому\*.

11 | 4

Сервер звертається до клієнтів через випадкові проміжки часу з проханням ввести будь-яке слово (перевіряти чи це слово). Якщо клієнт не вводить за фіксований час (до 10 секунд слово), то сервер сам пише в колекцію слово «alert». Кожній 10 хвилин сервер виводить повідомлення з підрахунком кількості слів alert та часом, коли вони були занесені в колекцію.

\* - результати роботи програми повинні виводитися при завершенні її роботи

(передбачити для цього спеціальне кодове слово в enum).

\*\* -

https://uk.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB%D1%96%D1%87%D0%BD%D0%B8%

D0%B9\_%D0%B1%D1%83%D1%84%D0%B5%D1%80

Передбачити обробку можливих нештатних подій ти помилок при

роботі створеної системи та окремих її частин.

Зробити висновки по практичній роботі з самоаналізом за принципом:

- що вдалося реалізувати по завданням на 100% (не виходячи за

рамки тем, перерахованих в теоретичних відомостях);

- що вдалося реалізувати по завданням на частково (не виходячи за

рамки тем, перерахованих в теоретичних відомостях) і чому;

- що не вдалося реалізувати по завданням взагалі (не виходячи за

рамки тем, перерахованих в теоретичних відомостях) і чому.

**Лістинг коду:**

Phone

package com;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class Phone implements Closeable *{* private final Socket socket;  
 private final BufferedWriter writer;  
 private final BufferedReader reader;  
  
 public Phone*(*String ip, int port*) {* try *{* this.socket = new Socket*(*ip,port*)*;  
 this.reader = createReader*()*;  
 this.writer = createWriter*()*;  
 *}* catch *(*IOException e*) {* throw new RuntimeException*(*e*)*;  
 *}  
 }* public Phone*(*ServerSocket server*){* try *{* this.socket = server.accept*()*;  
 this.reader = createReader*()*;  
 this.writer = createWriter*()*;  
 *}* catch *(*IOException e*) {* throw new RuntimeException*(*e*)*;  
 *}  
 }* public void writeLine*(*String message*) {* try *{* writer.write*(*message*)*;  
 writer.newLine*()*;  
 writer.flush*()*;  
 *}* catch *(*IOException e*) {* throw new RuntimeException*(*e*)*;  
 *}  
 }* public String readLine*() {* try *{* return reader.readLine*()*;  
 *}* catch *(*IOException e*) {* throw new RuntimeException*(*e*)*;  
 *}  
 }* private BufferedReader createReader*()* throws IOException *{* return new BufferedReader*(*new InputStreamReader*(*socket.getInputStream*()))*;  
 *}* private BufferedWriter createWriter*()* throws IOException *{* return new BufferedWriter*(*new OutputStreamWriter*(*socket.getOutputStream*()))*;  
 *}* @Override  
 public void close*()* throws IOException *{* writer.close*()*;  
 reader.close*()*;  
 socket.close*()*;  
 *}  
}*

Server

import com.Phone;  
  
import java.io.IOException;  
import java.net.ServerSocket;  
  
public class Server *{* public static void main*(*String*[]* args*) {  
 serverOperations()*;  
 *}* public static void serverOperations*() {* try *(*ServerSocket serverSocket = new ServerSocket*(*8000*)) {* System.*out*.println*(*"Server started!"*)*;  
 while *(*true*) {* Phone phone = new Phone*(*serverSocket*)*;  
 new Thread*(()* -> *{* String request = phone.readLine*()*;  
 System.*out*.println*(*"Request: " + request*)*;  
 String response = "HELLO FROM SERVER: " + request.length*()*;  
// try {  
// Thread.sleep(4000);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
 phone.writeLine*(*response*)*;  
 System.*out*.println*(*"Response" + response*)*;  
 try *{* phone.close*()*;  
 *}* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}  
 })*.start*()*;  
 *}  
 }* catch *(*IOException e*) {* throw new RuntimeException*(*e*)*;  
 *}  
 }  
}*

Client

import com.Phone;  
  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Client *{* public static void main*(*String*[]* args*) {* while *(*true*){* try *(*Phone phone = new Phone*(*"127.0.0.1", 8000*)) {* System.*out*.println*(*"Connected to server!"*)*;  
 Scanner scanner = new Scanner*(*System.*in)*;  
 System.*out*.print*(*"Input line: "*)*;  
 while *(*!scanner.hasNextLine*()){* System.*out*.print*(*"Input line: "*)*;  
 scanner.nextLine*()*;  
 *}* String request = scanner.nextLine*()*;  
 System.*out*.println*(*"Request: " + request*)*;  
 phone.writeLine*(*request*)*;  
  
 String response = phone.readLine*()*;  
 System.*out*.println*(*"Response: " + response*)*;  
  
 *}* catch *(*IOException e*) {* e.getStackTrace*()*;  
 *}  
 }  
  
 }  
}*

**Результат роботи:**



