**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Н-Н ІНСТИТУТ ПІДПРИЄМНИЦТВА ТА ПЕРСПЕКТИВНИХ ТЕХНОЛОГІЙ**



**ЗВІТ**

до виконаної лабораторної роботи №1

з дисципліни *«Клієнт-серверне програмування»*

на тему

**« Мережеве програмування на Java »**

Виконав:

студент гр. КН-214

**Дубницький Ю.І.**

Прийняв:

**Скобилко І.Я.**

**Львів–2019**

**Лабораторна робота №1**

**Тема:***Мережеве програмування на Java*

**Мета:***Навчитися програмувати в мережі на мові Java*

**Теоретичні відомості**

Одна із сильних сторін Java полягає в безпроблемній мережевій роботі. Дизайнери мережевої бібліотеки Java зробили її досить простою для читання і запису файлів, за винятком випадку, коли "файл" існує на віддаленій машині і віддалена машина може вирішувати що їй робити з інформацією, яку ви запитуєте або посилаєте. Наскільки це можливо, нижче лежачі деталі мережевої взаємодії були абстраговані і про них піклується ядро ​​JVM і локальний пакет установки Java. Програмна модель, яку ви використовуєте для такого файлу, фактично, це обгортка мережевого з'єднання ( "сокет") з об'єктом потоку, так що в кінцевому рахунку ви використовуєте ті ж виклики методів, які ви використовуєте для інших потоків. Крім того, вбудована многопоточністьJava виключно зручна, коли ви маєте справу з такою мережевий можливістю, як обробка безлічі з'єднань одночасно.

**Код**

**Клієнт:**

import java.io.\*;

import java.net.\*;

importjava.util.\*;

publicclass SocketClient11

{

publicstaticvoidmain(Stringargs[])

{

bytebKbdInput[] = newbyte[256];

Socket s;

InputStreamis;

OutputStreamos;

try

{

System.out.println(

"SocketClientApplication" +

"\nEnteranystringor" +

" 'quit' toexit...");

}

catch(Exceptionioe)

{

System.out.println(ioe.toString());

}

try

{

s = newSocket("localhost",9999);

is = s.getInputStream();

os = s.getOutputStream();

bytebuf[] = newbyte[512];

intlength;

Stringstr;

while(true)

{

length = System.in.read(bKbdInput);

if(length != 1)

{

str = newString(bKbdInput, 0);

StringTokenizerst;

st = newStringTokenizer(

str, "\r\n");

str = newString(

(String)st.nextElement());

System.out.println("> " + str);

os.write(bKbdInput, 0, length);

os.flush();

length = is.read(buf);

if(length == -1)

break;

str = newString(buf, 0);

st = newStringTokenizer(

str, "\r\n");

str = newString(

(String)st.nextElement());

System.out.println(">> " + str);

if(str.equals("quit"))

break;

}

}

is.close();

os.close();

s.close();

}

catch(Exceptionioe)

{

System.out.println(ioe.toString());

}

try

{

System.out.println(

"Press<Enter>to " +

"завершення програми...");

System.in.read(bKbdInput);

}

catch(Exceptionioe)

{

System.out.println(ioe.toString());

}

}

}

**Сервер:**

import java.io.\*;

import java.net.\*;

importjava.util.\*;

publicclassSocketServ//nameofserversocket

{

publicstaticvoidmain(Stringargs[])

{

bytebKbdInput[] = newbyte[256];

ServerSocketss;

Socket s;

InputStreamis;

OutputStreamos;

try

{

System.out.println(

"Socket Server Application");

}

catch(Exceptionioe)

{

System.out.println(ioe.toString());

}

try

{

ss = newServerSocket(9999);

s = ss.accept();

is = s.getInputStream();

os = s.getOutputStream();

bytebuf[] = newbyte[512];

intlenght;

while(true)

{

lenght = is.read(buf);

if(lenght == -1)

break;

Stringstr = newString(buf, 0);

StringTokenizerst;

st = newStringTokenizer(

str, "\r\n");

str = newString(

(String)st.nextElement());

System.out.println("> " + str);

os.write(buf, 0, lenght);

os.flush();

}

is.close();

os.close();

s.close();

ss.close();

}

catch(Exceptionioe)

{

System.out.println(ioe.toString());

}

try

{

System.out.println("Press<Enter>toterminateapplication...");

System.in.read(bKbdInput);

}

catch(Exceptionioe)

{

System.out.println(ioe.toString());

}

}

}

**Висновок:***В цій лабораторній роботі я навчився програмувати в мережі на Java.*