Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

**Лабораторна робота № 1**

по дисципліні «Програмування компютерних та віртуальних мереж інформаційно-комунікаційних систем»

Тема: «Знайомство з TCP sockets»

|  |  |
| --- | --- |
| Виконав:  студенти групи ІТ-91мн  Давиденко І. В.  Свириденко О.  Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Захищено з балом \_\_\_\_\_\_\_\_\_\_\_ | Перевірено:  ас. Галушко Д. О. |

Київ 2017

Завдання

1. Встановити та налаштувати JDK (версія – не нижче ніж JDK 8).

2. Встановити будь яку IDE на ваш вибір (NetBeans, Eclipse, IntelliJ IDEA,JDeveoper).

3. Реалізувати наступне клієнт-серверний застосунок:

Сервер:

Сервер має вміти «відповідати» на такі запити клієнта:

- Номер бригади;

- Прізвища усіх студентів у бригаді.

Клієнт:

Консольний застосунок, який демонструє функціональність сервера.

Контрольні питання

1. Як співвідносяться модель ISO OSI та стек протоколів TCP/IP?

2. Чим TCP відрізняється від UDP?

3. Що таке порт? Що таке сокет?

4. В чому полягає різниця між Socket та ServerSocket?

5. Що таке блокуючий та неблокуючий ввід-вивід даних?.

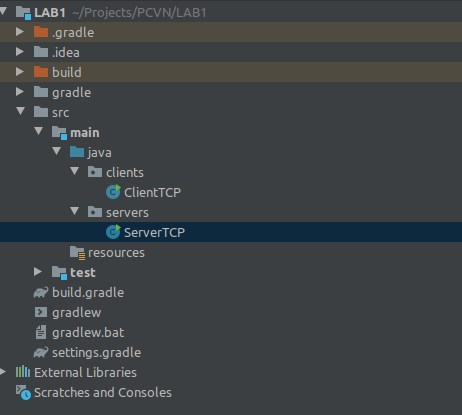
Короткі теоретичні відомості

UDP (англ. User Datagram Protocol — протокол пользовательских датаграмм) — один из ключевых элементов TCP/IP, набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных. Протокол был разработан Дэвидом П. Ридом в 1980 году и официально определён в RFC 768.

UDP использует простую модель передачи, без неявных «рукопожатий» для обеспечения надёжности, упорядочивания или целостности данных. Таким образом, UDP предоставляет ненадёжный сервис, и датаграммы могут прийти не по порядку, дублироваться или вовсе исчезнуть без следа. UDP подразумевает, что проверка ошибок и исправление либо не нужны, либо должны исполняться в приложении. Чувствительные ко времени приложения часто используют UDP, так как предпочтительнее сбросить пакеты, чем ждать задержавшиеся пакеты, что может оказаться невозможным в системах реального времени. При необходимости исправления ошибок на сетевом уровне интерфейса приложение может задействовать TCP или SCTP, разработанные для этой цели.

Хід роботи

Створю проект “LAB1”, котрий має наступну структуру:



Файл ClientTCP.java:

public class ClientTCP {

public static void main(String args[]) {

try {

Socket clientSocket = new Socket("localhost", 1500);

BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter((clientSocket.getOutputStream()), true);

System.out.println( in.readLine());

out.println("get\_group");

System.out.println(in.readLine());

out.println("get\_names");

System.out.println(in.readLine());

out.println("get\_name");

System.out.println(in.readLine());

out.close();

in.close();

clientSocket.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

Файл ServerTCP.java:

public class ServerTCP {

ServerSocket serverSocket = null;

public ServerTCP() {

try {

serverSocket = new ServerSocket(1500);

System.out.println("Starting the server ");

run();

} catch (Exception e) {

e.printStackTrace();

}

}

public void run() {

try {

Socket clientSocket;

while (true) {

clientSocket = serverSocket.accept();

System.out.println("Connection accepted from " + clientSocket.getInetAddress().getHostAddress());

BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);

writer.println("Client connected to sockets");

String inputLine;

while ((inputLine = in.readLine()) != null) {

if ("get\_names".equals(inputLine)) {

writer.println("I....");

}else if ("get\_group".equals(inputLine)) {

writer.println("IT-91mn");

}else {

clientSocket.close();

break;

}

System.out.println(inputLine);

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

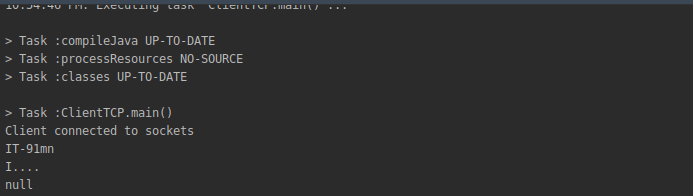
public static void main(String args[]) {

new ServerTCP();

}

}

В результаті виконання програми отримаю наступне:



Висновок: Під час лабораторної роботи я навчився створювати проект котрий за допомогою протоколу TCP/IP здійснює відправку повідомлень в реальному часі. Також створив програму котра отримує повідомлення та виводить їх на екран.