НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ

**Лабораторна робота № 3**

**з предмету «Програмування комп’ютерних та віртуальних мереж» на тему «ЗНАЙОМСТВО З ПРОТОКОЛОМ UDP»**

Виконала:

студентка групи ІТ-з03мп

Козак О. С.

Перевірив:

Галушко Д.О.

**Київ – 2020р.**

**Лабораторна робота № 3**

**Мета:** ознайомитись з протоколом UDP.

**Завдання:** Реалізувати наступний клієнт-серверний застосунок, що працює через протокол UDP:

Сервер:

Сервер має вміти «відповідати» на такі запити клієнта:

- сьогоднішня дата;

- поточний час;

- номер та прізвища усіх студентів у бригаді.

Клієнт:

Консольний застосунок, який демонструє функціональність сервера.

**Хід роботи**:

Для роботи з UDP протоколом та встановлення і підтримання з’єднання між клієнтом та сервером ми використовуємо класи DatagramSocket та DatagramPacket.

1. Створюємо клас сервера.

а) Створюємо сокет сервера, використовуючи клас DatagramSocket. Для отримання та відправлення даних, ми створюємо екземпляр класу DatagramPacket.

**Лістинг Server:**

package CNVP\_Lab3\_Server;  
  
import java.io.IOException;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.SocketException;  
  
public class Server {  
 private DatagramSocket socket;  
  
 public Server() throws SocketException {  
 socket = new DatagramSocket(5548);  
 System.*out*.println("socket os running");  
 }  
  
 public void run(boolean status) {  
  
 try {  
 DatagramPacket receivePacket = SendReceive.*receiveData*(socket);  
 String received = new String(receivePacket.getData(), 0, receivePacket.getLength());  
 if (status) {  
 SendReceive.*sendData*(receivePacket, socket, received);  
 } else {  
 System.*out*.println("client ends connection");  
 }  
  
 } catch (IOException exception) {  
 exception.printStackTrace();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {   
 socket.close();  
 }  
 }  
}

б) Метод SendReceive відповідає за відправлення та отримання пакетів. Метод *receive* записує в об'єкт DatagramPacket прийняті дані, а метод *send* – перешле дані з об'єкта класу DatagramPacket вузлу, адреса якого вказана в пакеті.

**Лістинг SendReceive**

package CNVP\_Lab3\_Server;  
  
import java.io.IOException;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
  
public class SendReceive {  
 public static void sendData(DatagramPacket receivedPacket, DatagramSocket socket, String received) throws Exception {  
 InetAddress address = receivedPacket.getAddress();  
 int port = receivedPacket.getPort();  
 byte[] dataToSend = DataToSend.*getDataToSend*(received);  
 DatagramPacket sendPacket = new DatagramPacket(dataToSend, dataToSend.length, address, port);  
 socket.send(sendPacket);  
 }  
  
 public static DatagramPacket receiveData(DatagramSocket socket) throws IOException {  
 byte[] dataToReceive = new byte[256];  
 DatagramPacket receivePacket = new DatagramPacket(dataToReceive, dataToReceive.length);  
 socket.receive(receivePacket);  
 return receivePacket;  
 }  
}

в) На сервер від клієнта приходять три запити: сьогоднішня дата; поточний час; номер та прізвища усіх студентів у бригаді. Клас OperationDispatch відповідає за диспетчеризацію даних запитів. Визначаючи операцію, сервер надає відповідні дані для відправлення.

**Лістинг OperationDispatch:**

package CNVP\_Lab3\_Server;  
  
import com.fasterxml.jackson.core.JsonProcessingException;  
  
import java.io.IOException;  
import java.time.LocalDate;  
import java.time.LocalTime;  
  
public class OperationDispatch {  
 public static String getOperation(String receivedData) throws IOException {  
 return JsonParser.*convertFromJson*(receivedData);  
 }  
  
 public static boolean getStatusOfOperation(String receivedData) throws IOException {  
 String operation = *getOperation*(receivedData);  
 if (operation.equals("exit")) return false;  
 else return true;  
 }  
  
 public static byte[] getTodayData() throws JsonProcessingException {  
 LocalDate localDate = *getLocalDate*();  
 String todayDate = localDate.toString();  
 return DataToSend.*convertJsonToBytes*(todayDate);  
 }  
  
 public static LocalDate getLocalDate() {  
 return LocalDate.*now*();  
 }  
  
 public static byte[] getCurrentTime() throws JsonProcessingException {  
 LocalTime localTime = *getLocalTime*();  
 String currentTime = localTime.toString();  
 return DataToSend.*convertJsonToBytes*(currentTime);  
 }  
  
 public static LocalTime getLocalTime() {  
 return LocalTime.*now*();  
 }  
  
 public static byte[] getSurnames() throws JsonProcessingException {  
 String dataToSend = String.*join*(" ", "ПОЛІЩУК", "ШЕВЧУК", "ПОПОВА", "ХАЧАТУРЯН", "СЕРДЮК");  
 return DataToSend.*convertJsonToBytes*(dataToSend);  
 }  
}

г) Всі запити та відповіді пересилаються у вигляді Json. За серіалізацію та десеріалізацію відповідає клас JsonParser, створюючи екземпляр класу ObjectMapper.

**Лістинг JsonParser**

package CNVP\_Lab3\_Client;  
  
import com.fasterxml.jackson.core.JsonProcessingException;  
import com.fasterxml.jackson.databind.ObjectMapper;  
  
import java.io.IOException;  
  
public class JsonParser {  
 public static String ConvertToJson(int input) throws JsonProcessingException {  
 ObjectMapper objectMapper = new ObjectMapper();  
 ServerRequest serverRequest = new ServerRequest();  
 serverRequest.setOperation(OperationDispatch.*operation*(input));  
 return objectMapper.writeValueAsString(serverRequest);  
 }  
  
  
 public static String convertFromJson(String received) throws IOException {  
 ObjectMapper objectMapper = new ObjectMapper();  
 ServerRequest serverRequest = new ServerRequest();  
 serverRequest = objectMapper.readValue(received, ServerRequest.class);  
 String result = serverRequest.getOperation();  
 return result;  
 }  
}

д) Після завершення сесії закриваємо сокет сервера у методі Server. *run*.

е) У main створюємо екземпляр класу Server та викликаємо метод run.

**Лістинг Main:**

package CNVP\_Lab3\_Server;  
  
import java.net.SocketException;  
  
public class Main {  
 public static void main (String [] args){  
 try {  
 boolean status = true;  
 while (status){  
 Server server = new Server();  
 server.run(status);  
 }  
  
 } catch (SocketException exception) {  
 exception.printStackTrace();  
 }  
 }  
}

2. Створюємо клієнт.

а) Створюємо клас Client, який відповідає за надсилання та отримування запитів на сервер та відповідей від сервера. Дані передаються у вигляді Json.

**Лістинг Client:**

package CNVP\_Lab3\_Client;  
  
import java.io.IOException;  
import java.net.\*;  
import java.nio.charset.StandardCharsets;  
  
public class Client {  
 public static void send(DatagramSocket socket, InetAddress address, byte[] dataToSend) throws IOException {  
 DatagramPacket sendPacket = new DatagramPacket(dataToSend, dataToSend.length, address, 5548);  
 socket.send(sendPacket);  
 }  
  
 public static String receive(DatagramSocket socket) throws IOException {  
 byte[] dataToReceive = new byte[256];  
 DatagramPacket receivePacket = new DatagramPacket(dataToReceive, dataToReceive.length);  
 socket.receive(receivePacket);  
 String received = new String(receivePacket.getData(), 0, receivePacket.getLength(), StandardCharsets.*UTF\_8*);  
 String stringFromJson = JsonParser.*convertFromJson*(received);  
  
 return stringFromJson;  
 }  
}

б) За диспетчеризацію операцій відповідає клас OperationDispatch. Запит вводиться у командний рядок. Клас OperationDispatch надає відповідну запиту операцію.

**Лістинг OperationDispatch**

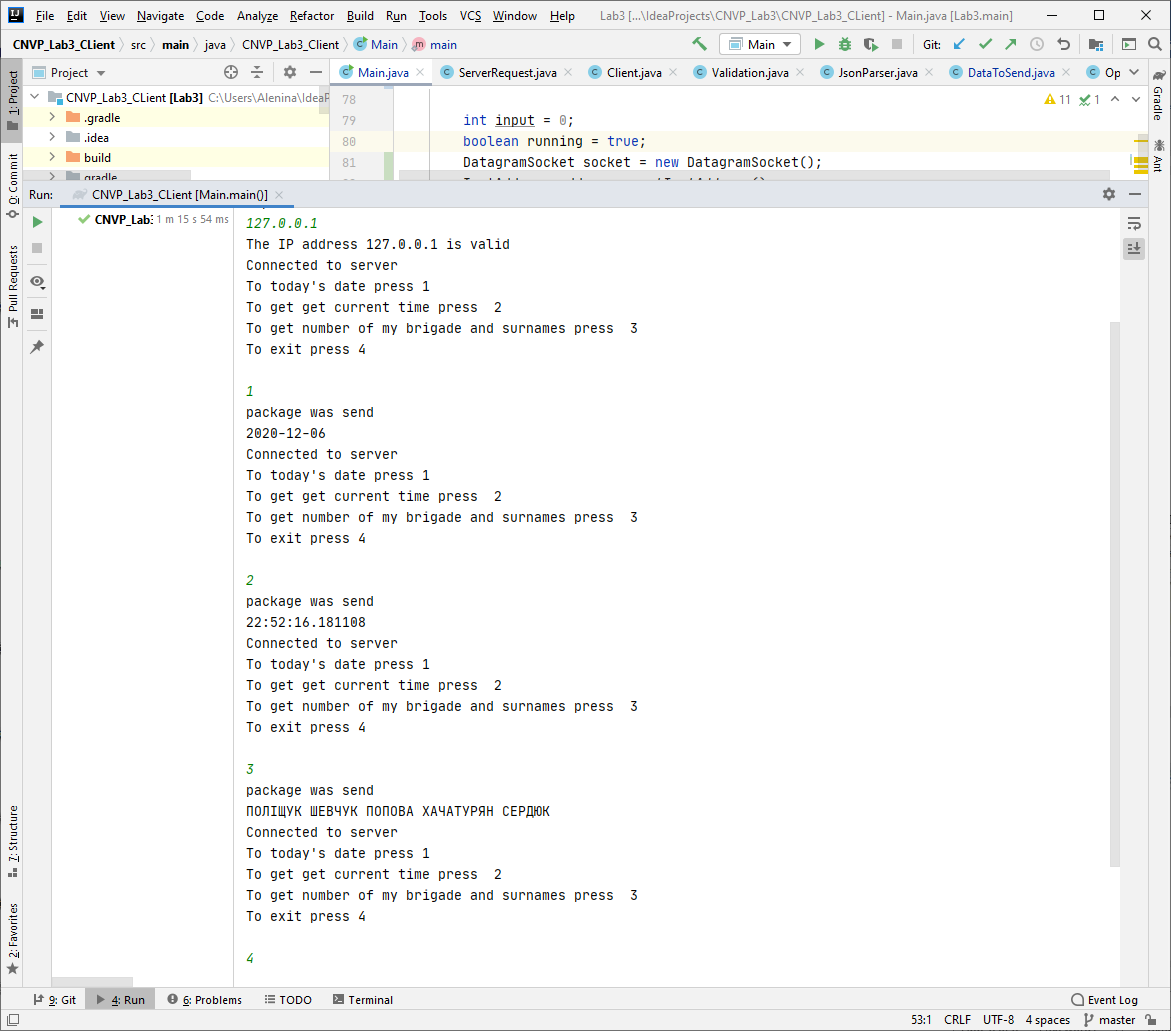
package CNVP\_Lab3\_Client;  
  
public class OperationDispatch {  
 public static String operation(int input) {  
 if (input == 1) {  
 return "GetTodayDate";  
 } else if (input == 2) {  
 return "GetCurrentTime";  
 } else if (input == 3) {  
 return "GetNumberOfMyBrigadeAndSurnames";  
 } else {  
 throw new UnsupportedOperationException("operation is not found");  
 }  
 }  
}

в) У класі Main ми маємо методи, які показують доступні опції для запитів, зчитуємо ввід клієнта, отримуємо IP-адресу для підключення. У методі main створюється сокет клієнта, відбувається підключення до сервера, зчитування запиту з командного рядку, конвертація запиту у Json та надсилання запиту на сервер, а також отримання відповіді від серверу.

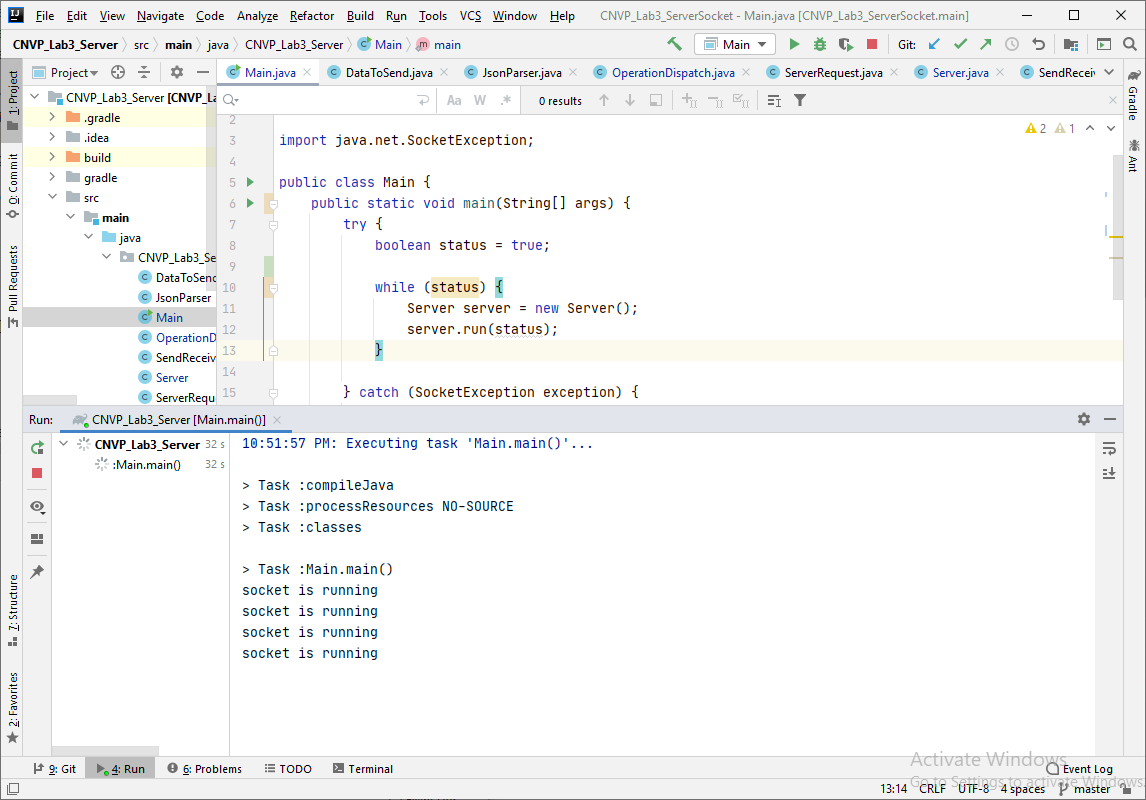
**Лістинг Main:**

package CNVP\_Lab3\_Client;  
  
import com.fasterxml.jackson.core.JsonProcessingException;  
  
import java.io.IOException;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
import java.net.SocketException;  
import java.net.UnknownHostException;  
import java.util.Scanner;  
  
public class Main {  
 public static void displayOptions() {  
 System.*out*.println("To today's date press 1");  
 System.*out*.println("To get get current time press 2");  
 System.*out*.println("To get number of my brigade and surnames press 3");  
 System.*out*.println("To exit press 4");  
 System.*out*.println();  
 }  
  
 public static boolean tryParseInt(String value) {  
 try {  
 Integer.*parseInt*(value);  
 return true;  
 } catch (NumberFormatException e) {  
 return false;  
 }  
 }  
  
 public static int readUserInput() {  
 *displayOptions*();  
 Scanner in = new Scanner(System.*in*);  
 while (true) {  
 String input = in.nextLine();  
 if (!*tryParseInt*(input)) {  
 System.*out*.println("Input is not correct: input should be int");  
 *displayOptions*();  
 continue;  
 }  
 int inputAsInt = Integer.*parseInt*(input);  
 if (inputAsInt == 1 || inputAsInt == 2 || inputAsInt == 3 || inputAsInt == 4) {  
 return inputAsInt;  
 } else {  
 System.*out*.println("Input is not correct: input should be 1, 2, 3 or 4");  
 *displayOptions*();  
 }  
 }  
 }  
  
 public static boolean validation(String inputedIp) {  
 boolean validation = false;  
 if (Validation.*isValidInetAddress*(inputedIp)) {  
 System.*out*.print("The IP address " + inputedIp + " is valid");  
 return true;  
 } else {  
 System.*out*.print("The IP address " + inputedIp + " isn't valid");  
 return false;  
 }  
  
 }  
  
 public static InetAddress getInetAddress() throws UnknownHostException {  
 System.*out*.println("Input server's IP address or use default 127.0.0.1");  
 Scanner in = new Scanner(System.*in*);  
 while (true) {  
 String ipAddressOfServer = in.nextLine();  
 boolean validation = *validation*(ipAddressOfServer);  
 if (validation) {  
 return InetAddress.*getByName*(ipAddressOfServer);  
 } else {  
 System.*out*.println("Input server's IP address or use default 127.0.0.1");  
 }  
 }  
 }  
  
  
 public static void main(String[] args) throws SocketException, UnknownHostException {  
  
 int input = 0;  
 boolean running = true;  
 DatagramSocket socket = new DatagramSocket();  
 InetAddress address = *getInetAddress*();  
 while (running)  
 try {  
 System.*out*.println("Connected to server");  
 Client client = new Client();  
 input = *readUserInput*();  
 if (input == 4) {  
 break;  
 } else {  
 String dataToSendInJson = JsonParser.*ConvertToJson*(input);  
 byte[] dataToSend = DataToSend.*getDataToSend*(dataToSendInJson);  
 client.*send*(socket, address, dataToSend);  
 System.*out*.println("package was send");  
 String received = client.*receive*(socket);  
 System.*out*.println(received);  
 }  
  
 } catch (SocketException e) {  
 e.printStackTrace();  
 } catch (UnknownHostException e) {  
 e.printStackTrace();  
 } catch (JsonProcessingException exception) {  
 exception.printStackTrace();  
 } catch (IOException exception) {  
 exception.printStackTrace();  
 }  
 }

3. Результат роботи клієнта:



4. Результат роботи сервера:



**Висновки:**

Я ознайомилась з принципами роботи протоколу UDP та реалізували клієнт-серверний застосунок, що працює через протокол UDP.