

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
"Проектування комп'ютерних мереж"
для студентів спеціальності 123 Комп'ютерна інженерія,
усіх форм навчання.
User Datagram Protocol. Логічні з'єднання

Методичні вказівки до виконання лабораторних робіт з дисципліни "Проектування комп'ютерних мереж" для студентів спеціальності 123 Комп'ютерна інженерія, усіх форм навчання. User Datagram Protocol. Логічні з'єднання / Укл.: Г.Г. Киричек. – Національний університет «Запорізька політехніка», 2024. – 30 с.

Укладач:

Г.Г. Киричек, доцент, к.т.н.

Рецензент:

М.Ю. Тягунова, доцент, к.т.н.

Відповідальний за випуск:

Г.Г. Киричек, доцент, к.т.н.

Затверджено
на засіданні кафедри КСМ
Протокол № 1 від 06.08.2024

Рекомендовано до видання
НМК ФКНТ
Протокол № 1 від 04.09.2024

ЗМІСТ

1 Лабораторна робота. User Datagram Protocol.....	4
1.1 Загальні відомості	4
1.2 Перевірка налаштувань мережевого інтерфейсу.....	6
1.3 Установка середовища програмування Java	8
1.4 Створення клієнтської програми	9
1.5 Створення серверної програми	14
1.6 Компіляція програм.....	16
1.7 Перевірка роботи мережевого додатку	17
1.8 Завдання на модифікацію мережевого додатку.....	17
1.9 Зміст письмового звіту.....	18
1.10 Контрольні питання	18
2 Лабораторна робота. Логічні з'єднання	19
2.1 Загальні відомості	19
2.2 Налаштування мережевого інтерфейсу.....	20
2.3 Установка середовища програмування Java	21
2.4 Створення серверної програми	23
2.5 Компіляція програм.....	28
2.6 Перевірка роботи web-сервера	28
2.7 Завдання на модифікацію web-сервера	29
2.8 Відновлення комп'ютерів в початковий стан	29
2.9 Зміст письмового звіту.....	29
2.10 Контрольні питання	29
Рекомендована література	30

1 ЛАБОРАТОРНА РОБОТА.

User Datagram Protocol

Мета роботи – ознайомлення з питаннями розробки додатків для Інтернету.

1.1 Загальні відомості

User Datagram Protocol (UDP) є простим, орієнтованим на дейтаграми протоколом без організації з'єднання. Він надає швидке, але необов'язково надійне транспортне обслуговування та підтримує взаємодії «один з багатьма», тому часто застосовується для широкомовної і групової передачі дейтаграм.

TCP/IP є набором протоколів (стеком протоколів), він має ще одну назву і це є «пакет протоколів Інтернету», який в моделі має чотири рівні. Тож маємо що TCP/IP не просто один протокол, а сімейство протоколів, який має у своєму складі багато інших низькорівневих протоколів, наприклад, таких як IP, TCP, UDP і протоколи базових мережевих технологій. TCP і UDP є протоколами транспортного рівня, які працюють поверх IP (протоколу мережного рівня). Транспортний рівень забезпечує взаємодію між мережами через шлюзи з використанням IP-адрес для відправки пакетів даних через Інтернет або інші мережі за допомогою різних драйверів пристроїв.

При передачі даних пакетом є послідовність бінарних цифр, які представляють дані і управляючі сигнали, що передаються і комутуються через кінцевий пристрій. Усередині пакета ця інформація розташована відповідно до спеціального стандартного формату.

Дейтаграмою є незалежний пакет даних, що має інформацію, достатню для передачі від відправника до пункту призначення, тому ніякого додаткового обміну між джерелом, адресатом і транспортною мережею не потрібно використовувати.

Для того, щоб поставити у відповідність вихідним даним конкретний процес, який виконується в кінцевому пристрої, UDP використовує порти.

У порівнянні з TCP, заголовок UDP набагато простіший, оскільки містить лише чотири поля і потребує 8 байтів (тобто 64 біти). Заголовок складається з чотирьох 2-байтових полів та містить:

- номер порту відправника - це 16-бітне поле, яке використовується для ідентифікації застосунку відправника за номером його порту;

- номер порту одержувача - це 16-бітне поле, яке використовується для ідентифікації застосунку призначення за номером його порту;

- довжина дейтаграми - це 16-бітне поле, яке вказує на довжину заголовка UDP-дейтаграми;

- контрольна сума - це 16-бітне поле, яке використовується для перевірки наявності помилок, як в даних так і в заголовку дейтаграм.

В .NET протокол UDP можна реалізувати, використовуючи такі можливості: клас `UdpClient`; клас `Socket`; елемент управління `Winsock` та некерований API `Winsock`.

В роботі, у вигляді прикладу, ми модифікуємо для протоколу UDP мережевий додаток. Коди програм для протоколу UDP значно відрізняються від TCP.

Порядок дій, що виконуються додатком:

- клієнт зчитує зі стандартного пристрою вводу (клавіатури) рядок символів і посилає цей рядок серверу через свій сокет;
- сервер приймає рядок через свій сокет;
- сервер переводить всі символи рядка у верхній регістр;
- сервер відсилає модифікований рядок клієнту;
- клієнт одержує рядок і друкує його за допомогою стандартного пристрою виводу (монітора).

На рис. 1.1 приведена схема взаємодії клієнта і сервера при використанні протоколу UDP.

Примітка. Вказівки для роботи складені з орієнтацією на виконання її на комп'ютерах з фізичними іменами **RMF01** і **RMF02**.

Для цих комп'ютерів, нагадаємо, що при завантаженні на них клієнтської ОС **Microsoft Windows** маємо:

IP-адреса комп'ютера: **10.0.9.51** і **10.0.9.52** відповідно,
мережеве ім'я комп'ютера: **rmf51** і **rmf52** відповідно.

Оскільки більшість з вас працює за комп'ютерами з іншими фізичними іменами, то при виконанні роботи вам доведеться використовувати відповідно інші ("свої") IP-адреси, короткі і повні мережеві імена.

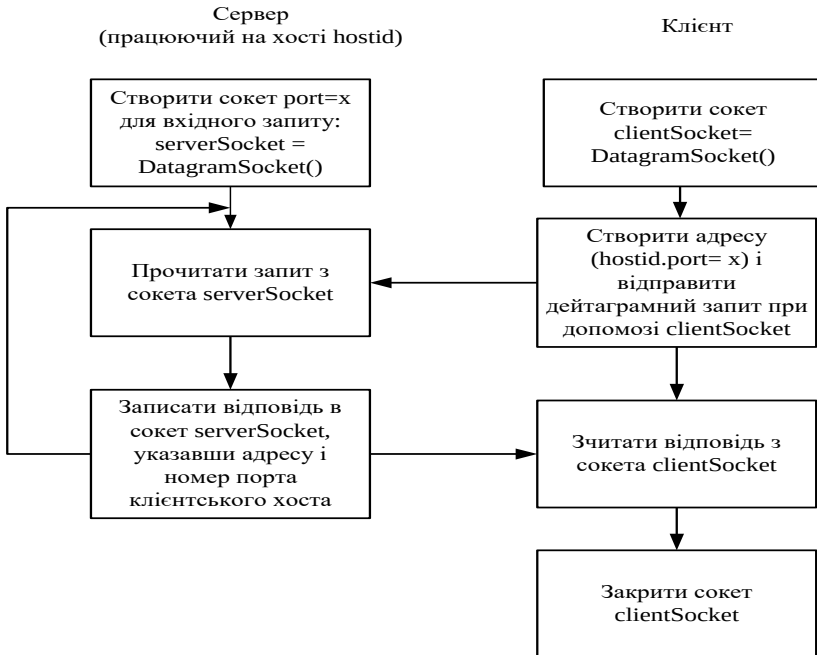


Рисунок 1.1 - Додаток клієнт/сервер, який використовує транспортні служби без устанавлення логічного з'єднання

1.2 Перевірка налаштувань мережевого інтерфейсу

На комп'ютері з фізичним ім'ям **RMF01**:

- завантажте **Microsoft Windows** (ім'я користувача – **admin**, пароль – **1234567**);
- перевірте настройки мережевого інтерфейсу;
- задайте **Пуск => Панель управління => Мережа та Інтернет => Центр управління мережами та спільним доступом => Зміна параметрів адаптера**;
- у вікні **Мережеві підключення** запустіть - **Підключення по локальній мережі**;
- у вікні **Стан Підключення по локальній мережі** клацніть **Властивості**;

– установіть курсор на **Протокол Інтернету (TCP/IPv4)** і клацніть **Властивості**;

– виберіть опцію **Використовувати наступну IP-адресу** і введіть:

IP-адреса: **10.0.9.51** (якщо фізичне ім'я вашого комп'ютера **RMF01+50**), (аналогічно, якщо фізичне ім'я вашого комп'ютера **RMF25**, то – **10.0.9.75**)

Маска підмережі: **255.255.255.0**

Основний шлюз: **10.0.9.100** (порт маршрутизатора)

Основний DNS-сервер: **10.0.2.1**

і клацніть **ОК**,

– з поверненням у вікно **Підключення по локальній мережі - Властивості** клацніть **ОК**,

– закрийте вікно **Стан–Підключення по локальній мережі**.

– закрийте вікно **Перегляд основних відомостей про мережу та налаштування підключень**.

На комп'ютері з фізичним ім'ям **RMF02**:

– завантажте **Microsoft Windows** (ім'я користувача – **admin**, пароль – **1234567**);

– перевірте настройки мережевого інтерфейсу:

– задайте **Пуск => Панель управління => Мережа та Інтернет => Центр управління мережами та спільним доступом => Зміна параметрів адаптера**;

– у вікні **Мережеві підключення** запустіть - **Підключення по локальній мережі**;

– у вікні **Стан Підключення по локальній мережі** клацніть **Властивості**;

– установіть курсор на **Протокол Інтернету (TCP/IPv4)** і клацніть **Властивості**;

– виберіть опцію **Використовувати наступну IP-адресу** і введіть:

IP-адреса: **10.0.9.52** (якщо фізичне ім'я вашого комп'ютера **RMF02+50**)

Маска підмережі: **255.255.255.0**

Основний шлюз: **10.0.9.100**

Основний DNS-сервер: **10.0.2.1**

і клацніть **ОК**,

- з поверненням у вікно **Підключення по локальній мережі** - **Властивості** клацніть **ОК**,
- закрийте вікно **Стан–Підключення по локальній мережі**.
- закрийте вікно **Перегляд основних відомостей про мережу та налаштування підключень**.

1.3 Установка середовища програмування Java

Якщо на комп'ютерах з фізичними іменами **RMF01** і **RMF02** вже встановлено середовище програмування Java (про це свідчить наявність в списку встановлених програм пункту

Java 2 SDK, SE v....;

див. **Панель управління => Програми => Програми та компоненти**), то перейдіть до виконання на обох комп'ютерах пункту 2 цього розділу (щоб продовжити роботу з перевірки правильності створення перемінних оточення студентами, які працювали до вас на цих комп'ютерах).

Якщо ні, то на обох комп'ютерах виконайте наступні операції:

Установіть середовище програмування Java, для чого:

- запустіть:

Java_install\JDK1_4\j2sdk-....-win.exe;

– з відкриттям вікна **Welcome to the InstallShield Wizard for Java2 SDK, SE v....** клацніть **Next**;

- у вікні **License Agreement** клацніть **Yes**;

– у вікні **Choose Destination Location** погодьтеся з пропозицією створити папку:

j2sdk1.4.0;

- тобто клацніть **Next**;

- у вікні **Select Components** клацніть **Next**;

– у вікні **Select Browsers** погодьтеся з запропонованим варіантом (**Microsoft Internet Explorer**), тобто клацніть **Next**;

– з відкриттям вікна **InstallShield Wizard Complete** клацніть **Finish**;

- закрийте вікно **JDK1_4**;

Створіть перемінні оточення, для чого:

- виконайте:

Пуск => Налаштування => Панель управління => Система та безпека => Система.

- оберіть вкладку **Захист системи**;
- у вікні **Властивості системи** перейдіть на вкладку **Додатково** і клацніть **Змінні середовища**;
- додайте в системну перемінну **Path** ще один шлях:
- у вікні **Змінні середовища** у розділі **Системні змінні** виділіть рядок з перемінною **Path**;

- клацніть **Змінити**;

– у вікні **Зміна системної змінної** у полі **Значення змінної** додайте в кінці рядка

диск:\j2sdk1.4.0\bin

і клацніть **ОК**;

- створіть системну перемінну **CLASSPATH**;

- у вікні **Змінні середовища** у розділі **Системні змінні** клацніть

Створити;

- у вікні **Нова системна змінна**:
- у полі **Ім'я змінної** задайте **CLASSPATH**;
- у полі **Значення змінної** задайте

диск:\j2sdk1.4.0\jre\lib\rt.jar

і клацніть **ОК**;

– для завершення процесу створення нових перемінних середовища клацніть **ОК** (у вікні **Змінні середовища**), клацніть **ОК** (у вікні **Властивості системи**);

- закрийте вікно **Панель управління**.

Створіть на диску папку **Work** з підкаталогом **UDPLAB**.

1.4 Створення клієнтської програми

На комп'ютері з фізичним ім'ям **RMF02** (мережевим ім'ям **rmf52**; у вашого комп'ютера відповідне “своє” мережеве ім'я):

Запустіть програму:

Пуск > Всі Програми > Стандартні > WordPad

Наберіть наступний текст програми:

import java.io.*;

import java.net.*;

```

class UDPClient {
    public static void
    main(String args[]) throws Exception
    {
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader
                (System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress =
            InetAddress.getByName("hostname");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket =
            new DatagramPacket(sendData, sendData.length, IPAddress,
9876);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket =
            new DatagramPacket(receiveData,
                receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence =
            new String(receivePacket.getData());
        System.out.println("FROM SERVER:" + modifiedSentence);
        clientSocket.close();
    }
}

```

Як показано на рисунку 1.2, програма **UDPClient** створює один потік даних і один сокет.

Сокет має тип **DatagramSocket** і ім'я **clientSocket**.

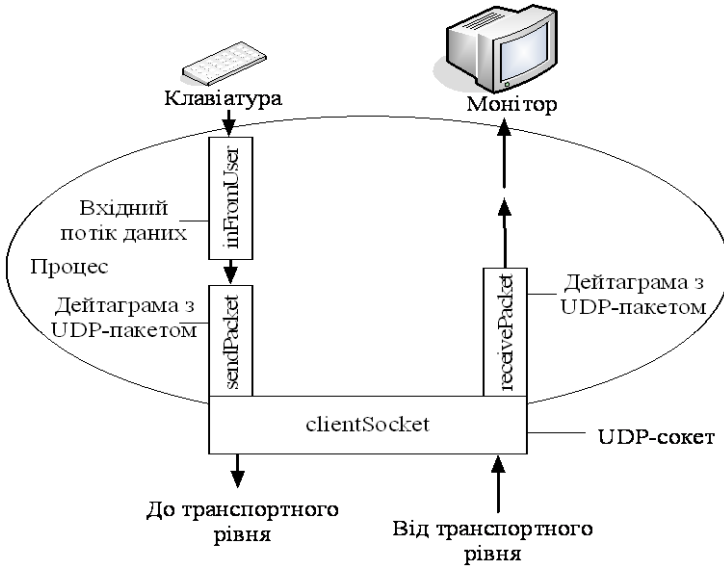


Рисунок 1.2 - В програмі **UDPClient** є єдиний потік даних; сокет приймає і передає пакети процесу

Зверніть увагу, що при використанні протоколів UDP і TCP типи сокетів розрізняються: у випадку TCP, звичай, застосовують тип **Socket**. Потік **inFromUser** є вхідним для програми і пов'язаний з стандартним пристроєм вводу (клавіатурою).

Коли користувач вводить символи, вони потрапляють в програму через вхідний потік даних. Аналогічним чином вхідний потік визначений в програмі **TCPClient**.

Слід зазначити одну важливу деталь: при використанні протоколу UDP жоден з потоків (ні вхідний, ні вихідний) не пов'язаний з сокетом. Відправка байтів здійснюється в пакетах за допомогою об'єкту типу **DatagramSocket**.

Тепер більш детально розглянемо рядки коду програми **UDPClient**.

```
DatagramSocket clientSocket =  
new DatagramSocket();
```

Ця команда створює об'єкт **clientSocket** типу **DatagramSocket**, однак на відміну від програми **TCPClient** не ініціює установлення TCP-

з'єднання. З цієї причини конструктор **DatagramSocket()** не приймає як аргументи ім'я хоста сервера і номер порту. Таким чином, дія приведеного рядка зводиться лише до створення сокета клієнта.

```
InetAddress IPAddress = InetAddress.getByName("hostname");
```

Для того, щоб мати можливість передавати віддаленому процесу інформацію, необхідно мати в своєму розпорядженні його адресу. Як ми знаємо, частиною адреси процесу є IP-адреса хоста, на якому він виконується. Даний рядок створює DNS-запит IP-адреси хоста з ім'ям **hostname** (як і в програмі **TCPClient**, слово **hostname** необхідно замінити реальним ім'ям хоста). Фактично робота з DNS-запитом здійснюється методом **getByName()**, який приймає як аргумент ім'я хоста і повертає отриману IP-адресу. IP-адреса поміщається в об'єкт **IPAddress** типу **InetAddress**.

```
byte[] sendData = new byte[1024];
```

```
byte[] receiveData = new byte[1024];
```

Байтові масиви **sendData** і **receiveData** призначені для зберігання інформації, що передається і приймається відповідно.

```
sendData = sentence.getBytes();
```

Рядок **sentence** за допомогою методу **getBytes()** перетворюється в масив байтів, який копіюється в перемінну **sendData**.

```
DatagramPacket sendPacket =
```

```
new DatagramPacket(sendData, sendData.length, IPAddress,  
9876);
```

Тут відбувається створення пакету **sendPacket**, призначеного для відправки серверу через сокет. В пакет входять дані, які містяться в перемінній **sendData**, розмір цих даних, IP-адреса хоста призначення і номер порту додатку (в даному випадку ми вибрали номер **9876**). Зверніть увагу на те, що пакет має тип **DatagramPacket**.

```
clientSocket.send(sendPacket);
```

Метод **send()** об'єкту **clientSocket** відправляє пакет **sendPacket** (як згадувалося раніше, **clientSocket** є клієнтським сокетом додатку). Звернемося до механізмів передачі даних, що здійснюються протоколами TCP і UDP. У випадку TCP рядок символів включається у вихідний потік, що має логічний зв'язок з сервером. У випадку UDP нам необхідно створити пакет, що містить адресу сервера. Після передачі пакета клієнт чекає прийому пакета від сервера.

```
DatagramPacket receivePacket =
```

```
new DatagramPacket(receiveData, receiveData.length);
```

Тут клієнт створює перемінну для зберігання пакету **receivePacket** типу **DatagramPacket**.

clientSocket.receive(receivePacket);

Клієнт знаходиться в стані очікування до тих пір, поки не буде початий прийом пакету від сервера; символи, що приймаються, поміщаються в перемінну **receivePacket**.

String modifiedSentence = new String(receivePacket.getData());

В цьому рядку відбувається витягання даних з об'єкту **receivePacket** і перетворення типу з масиву байтів в рядок з подальшим присвоюванням перемінній **modifiedSentence**.

System.out.println("FROM SERVER: " + modifiedSentence);

Цей рядок, присутній раніше в програмі **TCPClient**, друкує на екрані клієнта рядок **modifiedSentence**.

clientSocket.close();

Виклик методу **close()** об'єкту **clientSocket** приводить до закриття сокета. Оскільки протокол UDP не використовує логічне з'єднання, при закритті сокета клієнт не посилає серверу повідомлень транспортного рівня (як це було у випадку TCP).

Збережіть файл з текстом програми:

- клацніть **Файл => Зберегти як**;
 - у вікні **Зберегти як**;
 - у полі **Папка** укажіть **диск:\Work\UDPLAB**;
 - у полі **Тип файла** виберіть **Текстовий документ MS-DOS**;
 - у полі **Ім'я файлу** укажіть **UDPClient.java**;
 - клацніть **Зберегти**;
 - у підвікні **WordPad** клацніть **Так**;
 - закрийте вікно **UDPClient.java.txt-WordPad**;
 - відкрийте папку **диск:\Work\UDPLAB**;
 - зніміть прапорець **Приховувати розширення для зареєстрованих типів файлів (Пуск => Панель управління => Оформлення та персоналізація)**;
 - оберіть вкладку **Параметри папок>Вид**;
 - перейменуйте файл **UDPClient.java.txt** на **UDPClient.java**;
 - закрийте вікно **UDPLAB**.
- Внесіть зміну в текст програми:
- відкрийте файл **диск:\Work\UDPLAB\UDPClient.java** за допомогою **WordPad**;

- замініть слово **hostname** на мережеве ім'я комп'ютера з серверною програмою (**rmf51**; у вашого комп'ютера з серверною програмою відповідне “своє” мережеве ім'я);
- клацніть **Файл => Зберегти**, клацніть **Так**;
- закрийте вікно з файлом, що редагується;
- закрийте вікно **UDPLAB**.

1.5 Створення серверної програми

На комп'ютері з фізичним ім'ям **RMF01** (мережевим ім'ям **rmf51**; у вашого комп'ютера відповідне “своє” мережеве ім'я):

Запустіть програму:

Пуск => Всі Програми => Стандартні => WordPad

Наберіть наступний текст програми:

```
import java.io.*;
import java.net.*;
class UDPServer {
    public static void
    main(String args[]) throws Exception
    {
        DatagramSocket serverSocket = new DatagramSocket(9876);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];
        while(true)
        {
            DatagramPacket receivePacket =
                new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence = new String(receivePacket.getData());
            InetAddress IPAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            String capitalizedSentence = sentence.toUpperCase();
            sendData = capitalizedSentence.getBytes();
            DatagramPacket sendPacket =
                new DatagramPacket(sendData, sendData.length, IPAddress,
port);
            serverSocket.send(sendPacket);
```

```

    }
  }
}

```

Як показано на рисунку 1.3, програма **UDPServer** створює єдиний сокет з ім'ям **serverSocket** типу **DatagramSocket**; цей же тип використовувався нами в програмі-клієнті додатку.

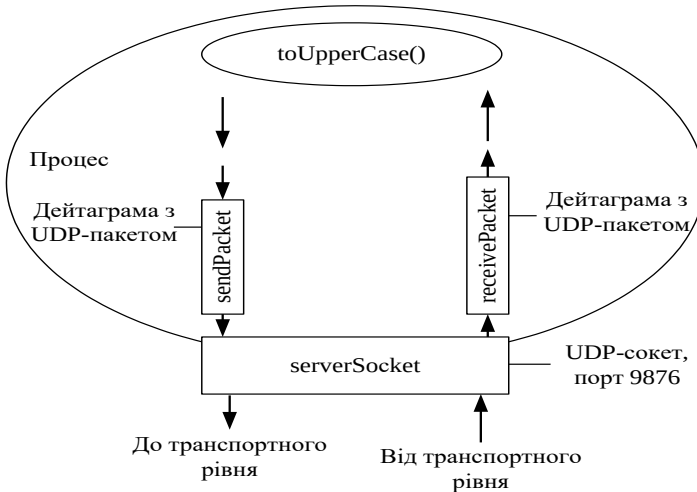


Рисунок 1.3 - Програма **UDPServer** не має потоків; сокет приймає і передає пакети процесу

Аналогічно, з сокетом не пов'язаний жоден з потоків даних. Розглянемо деякі з рядків приведенного коду.

```

DatagramSocket serverSocket =
new DatagramSocket(9876);

```

Цей рядок створює об'єкт **serverSocket** типу **DatagramSocket** з номером порту **9876**. Передача і прийом всіх даних сервером здійснюється за допомогою цього об'єкту.

Оскільки протокол UDP не передбачає встановлення логічного з'єднання, у нас немає необхідності підтримувати окремий сокет для кожного клієнта і спеціальний впускаючий сокет для рукошукань з новими клієнтами, як для протоколу TCP. У випадку декількох клієнтів

всі вони здійснюватимуть обмін даними з сервером через єдиний сокет **serverSocket**.

```
String sentence = new
    String(receivePacket.getData());
InetAddress IPAddress =
    receivePacket.getAddress();
int port = receivePacket.getPort();
```

Вищенаведені рядки здійснюють розпакування пакету, переданого клієнтом.

Перша команда витягає з пакету дані і поміщає їх в рядковий об'єкт **sentence** (аналогічна команда є в програмі **UDPClient**).

Наступна команда отримує IP-адресу, а остання, третя команда - номер порту сервера (в загальному випадку номер порту відрізняється від **9876**). Отримання IP-адреси і номера порту обумовлено необхідністю ідентифікації клієнта для передачі модифікованого рядка.

Збережіть файл з текстом програми:

- клацніть **Файл => Зберегти як**;
- у вікні **Зберегти як**;
- у полі **Папка** укажіть **диск:\Work\UDPLAB**;
- у полі **Тип файла** виберіть **Текстовий документ MS-DOS**;
- у полі **Ім'я файлу** задайте **UDPServer.java**;
- клацніть **Зберегти**;
- у підвікні **WordPad** клацніть **Так**;
- закрийте вікно **UDPServer.java.txt-WordPad**;
- відкрийте папку **диск:\Work\UDPLAB**;
- зніміть прапорець **Приховувати розширення для зареєстрованих типів файлів (Пуск => Панель управління => Оформлення та персоналізація)**;
- оберіть вкладку **Параметри папок => Вид**;
- перейменуйте файл **UDPServer.java.txt** на **UDPServer.java**;
- закрийте вікно **UDPLAB**.

1.6 Компіляція програм

На комп'ютері з фізичним ім'ям **RMF01**:

- задайте **Пуск => Виконати**;
- у полі **запуску команди** наберіть **cmd** і клацніть **ОК**;

- у вікні **диск:\Windows_2\System32\cmd.exe**;
 - виконайте команди: **диск: і cd диск:\Work\UDPLAB**;
 - виконайте команду компіляції: **javac UDPServer.java**;
 - щоб переконатися у тому, що компіляція пройшла успішно і з'явився файл: **диск:\Work\UDPLAB\UDPServer.class**;
 - виконайте команду: **dir**.
- На комп'ютері з фізичним ім'ям **RMF02**:
- задайте **Пуск => Виконати**;
 - у полі **запуску команди** наберіть **cmd** і клацніть **OK**;
 - у вікні **диск:\Windows\System32\cmd.exe**;
 - виконайте команди: **диск: і cd диск:\Work\UDPLAB**;
 - виконайте команду компіляції: **javac UDPClient.java**;
 - щоб переконатися у тому, що компіляція пройшла успішно і з'явився файл: **диск:\Work\UDPLAB\UDPClient.class**;
 - виконайте команду: **dir**.

1.7 Перевірка роботи мережевого додатку

На комп'ютері з фізичним ім'ям **RMF01** у тому ж вікні **диск:\Windows\System32\cmd.exe** виконайте команду

java UDPServer

(при цьому програма перейде в режим очікування).

На комп'ютері з фізичним ім'ям **RMF02** у тому ж вікні **диск:\Windows\System32\cmd.exe**:

- виконайте команду: **java UDPClient**;
- наберіть рядок, що містить латинські символи у нижнім і верхнім регістрі (наприклад, **aaaAAAbbbbBBB**), і натисніть **Enter**;
- запишіть результат роботи програм і проаналізуйте його;
- продемонструйте роботу програм викладачу і поясніть їх роботу.

1.8 Завдання на модифікацію мережевого додатку

Змініть додаток так, щоб поведінка додатку була наступною:

- клієнт відсилає серверу рядок, що містить прізвище студента;

- сервер при отриманні виводить його на екран і відсилає клієнту рядок, що містить ім'я студента;
- клієнт при отриманні виводить його на екран.

Змініть додаток так, щоб поведінка додатку була наступною:

- одна серверна програма чекає з'єднання відразу за двома портами;
- на клієнтській машині здійснюється запуск відразу двох клієнтських програм, що приєднуються до різних портів серверу. Ці програми зчитують по одному рядку з клавіатури і відправляють ці рядки серверу;
- сервер відправляє кожному з клієнтів рядок, рівний конкатенації отриманих від клієнтів рядків;
- клієнти виводять цей результат на екран.

Відновлення комп'ютерів в початковий стан

На обох комп'ютерах видаліть папку **диск:\Work**.

1.9 Зміст письмового звіту

- хід роботи;
- основні функції;
- відповіді на контрольні питання.

1.10 Контрольні питання

1. Укажіть принципові відзнаки протоколу UDP від протоколу TCP.
2. Розкрийте поняття UDP-дейтаграми.
3. У яких випадках доцільно здійснювати передачу даних повідомлення за протоколом UDP?

2 ЛАБОРАТОРНА РОБОТА. Логічні з'єднання

Мета роботи – ознайомлення з питаннями розробки додатків для Інтернету.

2.1 Загальні відомості

Web-сервер, який ви побудуєте, виконуватиме наступні функції:

- прийом і обробка єдиного HTTP-запиту;
- витягання необхідного файлу за допомогою файлової системи серверу;
- створення відповідного HTTP-повідомлення, що включає рядки заголовка і необхідний файл;
- передача відповідного повідомлення безпосередньо клієнту.

Програмний код, який буде представлений нижче, ілюструє лише основні аспекти роботи web-серверу і позбавлений багатьох деталей (наприклад, в ньому відсутня обробка виключень). Програма web-серверу зроблена максимально короткою, щоб ви могли сконцентрувати увагу на ключових складових серверу.

Крім того, програма **WebServer.java** написана з допущенням, що запитуваний файл є у файловій системі серверу.

Примітка. Методичні вказівки для цієї роботи складені з орієнтацією на виконання роботи на комп'ютерах з фізичними іменами **RMF01** і **RMF02**.

Для цих комп'ютерів, нагадаємо, при завантаженні на них клієнтської ОС **Microsoft Windows** маємо:

IP-адреса комп'ютера: **10.0.9.51** і **10.0.9.52** відповідно,
мережеве ім'я комп'ютера: **rmf51** і **rmf52** відповідно.

А оскільки більшість з вас працює за комп'ютерами з іншими фізичними іменами, то при виконанні роботи вам доведеться використовувати відповідно інші ("свої") IP-адреси, короткі і повні мережеві імена.

2.2 Налаштування мережевого інтерфейсу

На комп'ютері з фізичним ім'ям **RMF01**:

– завантажте **Microsoft Windows** (ім'я користувача – **admin**, пароль – **1234567**);

– перевірте настройки мережевого інтерфейсу:

– задайте **Пуск => Панель управління => Мережа та Інтернет**

=> Центр управління мережами та спільним доступом => Зміна параметрів адаптера;

– у вікні **Мережеві підключення** запустіть - **Підключення по локальній мережі**;

– у вікні **Стан Підключення по локальній мережі** клацніть **Властивості**;

– установіть курсор на **Протокол Інтернету (TCP/IPv4)** і клацніть **Властивості**;

– виберіть опцію **Використовувати наступну IP-адресу** і введіть:

IP-адреса: **10.0.9.51** (якщо фізичне ім'я вашого комп'ютера **RMF01+50**), (аналогічно, якщо фізичне ім'я вашого комп'ютера **RMF25**, то – **10.0.9.75**)

Маска підмережі: **255.255.255.0**

Основний шлюз: **10.0.9.100**

Основний DNS-сервер: **10.0.2.1**

і клацніть **ОК**,

– з поверненням у вікно **Підключення по локальній мережі** - **Властивості** клацніть **ОК**,

– закрийте вікно **Стан–Підключення по локальній мережі**.

– закрийте вікно **Перегляд основних відомостей про мережу та налаштування підключень**.

На комп'ютері з фізичним ім'ям **RMF02**:

– завантажте **Microsoft Windows** (ім'я користувача – **admin**, пароль – **1234567**);

– перевірте настройки мережевого інтерфейсу:

– задайте **Пуск => Панель управління => Мережа та Інтернет**

=> Центр управління мережами та спільним доступом => Зміна параметрів адаптера;

- у вікні **Мережеві підключення** запустіть - **Підключення по локальній мережі**;
- у вікні **Стан підключення по локальній мережі** клацніть **Властивості**;
- установіть курсор на **Протокол Інтернету (TCP/IPv4)** і клацніть **Властивості**;
- виберіть опцію **Використовувати наступну IP-адресу** і введіть:

IP-адреса:	10.0.9.52 (якщо фізичне ім'я вашого комп'ютера RMF02+50)
Маска підмережі:	255.255.255.0
Основний шлюз:	10.0.9.100
Основний DNS-сервер:	10.0.2.1

 і клацніть **ОК**,
- з поверненням у вікно **Підключення по локальній мережі** - **Властивості** клацніть **ОК**,
- закрийте вікно **Стан–Підключення по локальній мережі**.
- закрийте вікно **Перегляд основних відомостей про мережу** та налаштування підключень.

2.3 Установка середовища програмування Java

Якщо на комп'ютерах з фізичними іменами **RMF01** і **RMF02** вже встановлено середовище програмування Java (про це свідчить наявність в списку встановлених програм пункту

Java 2 SDK, SE v...;

див. **Панель управління => Програми => Програми та компоненти**), перейдіть до виконання на обох комп'ютерах пункту 2 цього розділу (щоб продовжити роботу з перевірки правильності створення перемінних оточення студентами, які працювали до вас на цих комп'ютерах).

Якщо ні, то на обох комп'ютерах виконайте наступні операції.

Установіть середовище програмування Java, для чого:

- запустіть:
диск:\Java_install\JDK1_4\j2sdk-1_4_0-win.exe;
- з відкриттям вікна **Welcome to the InstallShield Wizard for Java2 SDK, SE v...** клацніть **Next**;

- у вікні **License Agreement** клацніть **Yes**;
- у вікні **Choose Destination Location** погодьтеся з пропозицією створити папку:

диск:\j2sdk1.4.0;

- тобто клацніть **Next**;
- у вікні **Select Components** клацніть **Next**;
- у вікні **Select Browsers** погодьтеся з запропонованим варіантом (**Microsoft Internet Explorer**), тобто клацніть **Next**;

- з відкриттям вікна **InstallShield Wizard Complete** клацніть **Finish**;

- закрийте вікно **JDK1_4**;

Створіть перемінні оточення, для чого:

- виконайте:

Пуск => Налаштування => Панель управління => Система та безпека => Система.

- оберіть вкладку **Захист системи**;
- у вікні **Властивості системи** перейдіть на вкладку **Додатково** і клацніть **Змінні середовища**;
- додайте в системну перемінну **Path** ще один шлях:
- у вікні **Змінні середовища** у розділі **Системні змінні** виділіть рядок з перемінною **Path**;

- клацніть **Змінити**;

- у вікні **Зміна системної змінної** у полі **Значення змінної** додайте в кінці рядка

;диск:\j2sdk1.4.0\bin

і клацніть **ОК**;

- створіть системну перемінну **CLASSPATH**:

- у вікні **Змінні середовища** у розділі **Системні змінні** клацніть **Створити**;

- у вікні **Нова системна змінна**:

- у полі **Ім'я змінної** задайте **CLASSPATH**;

- у полі **Значення змінної** задайте

.;диск:\j2sdk1.4.0\jre\lib\rt.jar

і клацніть **ОК**;

- для завершення процесу створення нових перемінних середовища клацніть **ОК** (у вікні **Змінні середовища**), клацніть **ОК** (у

вікні **Властивості системи**);

- закрийте вікно **Панель управління**.

2.4 Створення серверної програми

На комп'ютері з фізичним ім'ям **RMF01** (мережевим ім'ям **rmf51**; у вашого комп'ютера відповідне “своє” мережеве ім'я):

Створіть на диску папку **Work** з підкаталогом **WEBSRVLAB**.

Запустіть програму:

Пуск => Всі програми => Стандартні => WordPad

Наберіть наступний текст програми:

```
import java.io.*;
import java.net.*;
import java.util.*;
class WebServer {
    public static void
    main(String argv[]) throws Exception
    {
        String requestMessageLine;
        String fileName;
        ServerSocket listenSocket = new ServerSocket(6789);
        Socket connectionSocket = listenSocket.accept();
        BufferedReader inFromClient =
            new BufferedReader(
                new InputStreamReader(
                    connectionSocket.getInputStream()));
        DataOutputStream outToClient =
            new DataOutputStream(
                connectionSocket.getOutputStream());
        requestMessageLine = inFromClient.readLine();
        StringTokenizer tokenizedLine =
            new StringTokenizer(requestMessageLine);
        if (tokenizedLine.nextToken().equals("GET"))
        {
            fileName = tokenizedLine.nextToken();
            if (fileName.startsWith("/") == true)
                fileName = fileName.substring(1);
```

```

        System.out.println("Requested file is: " + fileName);
        File file = new File(fileName);
        int numBytes = (int) file.length();
        FileInputStream inFile = new FileInputStream (fileName);
        byte[] fileInBytes = new byte[numBytes];
        inFile.read(fileInBytes);
        outToClient.writeBytes("HTTP/1.0 200 Document Follows\r\n");
        if (fileName.endsWith(".jpg"))
            outToClient.writeBytes("Content-Type: image/jpeg\r\n");
            outToClient.writeBytes("Content-Length: "
+ numBytes + "\r\n");
            outToClient.writeBytes("\r\n");
            outToClient.write(fileInBytes, 0, numBytes);
            connectionSocket.close();
        }
        Else
            System.out.println("Bad Request Message");
    } }

```

Як можна помітити, перша частина програми майже ідентична програмі **TCPClient.java**: спочатку ми включаємо в програму пакети **java.io** і **java.net**, а потім додатковий пакет **java.util**, що містить клас **StringTokenizer**, який використовується для обробки HTTP-запитів.

Усередині класу **WebServer** оголошені два рядкових об'єкта:

```

String requestMessageLine;
String fileName;

```

Об'єкт **requestMessageLine** призначений для зберігання першого рядка HTTP-запиту, а об'єкт **fileName** - для зберігання імені запитуваного файла.

```

ServerSocket listenSocket = new
    ServerSocket(6789);
Socket connectionSocket = listenSocket.accept();
BufferedReader inFromClient =
    new BufferedReader(
        new InputStreamReader(
            connectionSocket.getInputStream()));
DataOutputStream outToClient =
    new DataOutputStream(
        connectionSocket.getOutputStream());

```

В приведеному вище фрагменті створюються два сокета, **listenSocket** і **connectionSocket**, типів **ServerSocket** і **Socket** відповідно. Об'єкт **listenSocket** створюється до начала установлення TCP-з'єднання з клієнтом і призначений для «прослуховування» порта **6789**. При одержанні запиту на установлення з'єднання об'єкт **listenSocket** за допомогою метода **accept()** створює об'єкт **connectionSocket** типу **Socket**.

Далі створюються вхідний потік **inFromClient** типу **BufferedReader** і вихідний потік **outToClient** типу **DataOutputStream**. HTTP-запит за допомогою сокета **connectionSocket** попадає у потік **inFromClient**, а відправлення відповіді здійснюється через потік **DataOutputStream**, також зв'язаний з сокетом **connectionSocket**.

Частина програми **WebServer.java**, що залишилася, значно відрізняється від програми **TCPClient.java**, тому ми розглянемо її більш докладно.

```
requestMessageLine = inFromClient.readLine();
```

Ця команда здійснює читання першого рядка HTTP-запиту. Припускається, що рядок має наступний формат: **GET file_name HTTP/1.0**.

Для того щоб виділити ім'я запитуваного файлу, необхідно виконати обробку цього рядка, що і роблять наступні команди:

```
StringTokenizer tokenizedLine =  
    new StringTokenizer(requestMessageLine);  
if (tokenizedLine.nextToken().equals("GET"))  
{  
    fileName = tokenizedLine.nextToken();  
    if (fileName.startsWith("/") == true )  
        fileName = fileName.substring(1);  
    .....  
}
```

Об'єкт **tokenizedLine** представляє собою перший рядок повідомлення, розбитий на три сегмента (поля), що містять «слова» **GET**, **file_name** і **HTTP/1.0**. Згідно описам HTTP в документах RFC, ім'я запитуваного файлу знаходиться в полі, що слідує відразу після слова **GET**. Ім'я файлу поміщається в рядок з ім'ям **fileName**.

Конструкції **if** в останніх рядках призначені для видалення символу **/**, котрий може бути присутнім перед ім'ям файлу.

```
FileInputStream inFile = new FileInputStream (fileName);
```

Цей рядок зв'язує потік **inFile** з файлом **fileName**.

```
byte[] fileInBytes = new byte[numOfBytes];
```

```
inFile.read(fileInBytes);
```

Перша команда визначає розмір файлу і створює масив байтів **fileInBytes** такого ж розміру. Друга команда здійснює читання з потоку **inFile** в масив **fileInBytes**. Використання масиву байтів пояснюється тим, що вихідний потік **outToClient** працює з даними тільки в байтовій формі. Коли обробка запиту завершена, ми можемо почати створення відповідного повідомлення. Спочатку ми поміщаємо у вихідний потік заголовні рядки відповіді:

```
outToClient.writeBytes("HTTP/1.0 200 Document Follows\r\n");
```

```
if (fileName.endsWith(".jpg"))
```

```
outToClient.writeBytes("Content-Type: image/jpeg\r\n");
```

```
outToClient.writeBytes("Content-Length: " + numOfBytes +  
"\r\n");
```

```
outToClient.writeBytes("\r\n");
```

Перша команда відправляє в буфер TCP рядок стану **HTTP/1.0 200 Document Follows**, що закінчується парою символів повернення каретки і переходу на наступний рядок.

Наступні дві команди формують рядок заголовка, що містить тип файлу. Якщо потрібно передати JPEG-файл, то рядок має вигляд **Content-Type: image/jpeg**, а у разі GIF-файлу — **Content-Type: image/gif**.

Зверніть увагу на те, що в розглянутому простому прикладі для файлу, тип якого відрізняється від JPEG або GIF, рядок **Content-Type**: взагалі не формується.

Потім сервер створює рядок, що містить розмір файлу, і порожній рядок, що відокремлює заголовну частину відповідного повідомлення від об'єкту (файлу), що передається.

```
outToClient.write(fileInBytes, 0, numOfBytes);
```

Ця команда пересилає запитуваний файл **fileInBytes** в буфер TCP. Протокол TCP об'єднає створені раніше рядки заголовка з файлом, при необхідності проведе сегментацію об'єднання і здійснить передачу сегментів клієнтській стороні додатку.

```
connectionSocket.close();
```

Після обслуговування запиту сервер закриває сокет **connectionSocket** за допомогою наведеної команди.

Збережіть файл з текстом програми:

- клацніть **Файл => Зберегти як**;
- у вікні **Зберегти як**;
- у полі **Папка** укажіть **диск:\Work\WEBSRVLAB**;
- у полі **Тип файла** виберіть **Текстовий документ MS-DOS**;
- у полі **Ім'я файлу** укажіть **WebServer.java**;
- клацніть **Зберегти**;
- у підвікні **WordPad** клацніть **Так**;
- закрийте вікно **WebServer.java.txt-WordPad**;
- відкрийте папку **диск:\Work\WEBSRVLAB**;
- зніміть прапорець **Приховувати розширення для зареєстрованих типів файлів (Пуск => Панель управління => Оформлення та персоналізація)**;

- оберіть вкладку **Параметри папок => Вид**;
- перейменуйте файл **WebServer.java.txt** на **WebServer.java**;
- закрийте вікно **WEBSRVLAB**.

Створіть документ **диск:\Work\WEBSRVLAB\test.html**:

- відкрийте папку **диск:\Work\WEBSRVLAB**;
- створіть у ній текстовий документ (для чого: клацніть правою кнопкою миші, виберіть команду **Створити => Текстовий документ**, натисніть клавішу **Enter**);

– перейменуйте файл **Текстовий документ.txt** на **test.html** (для чого: клацніть на ньому правою кнопкою миші, виберіть команду **Перейменувати** і так далі);

– відкрийте документ (клацніть правою кнопкою миші на документі **test.html**, виберіть **Відкрити з допомогою=> Блокнот => ОК**);

– у вікні **test.html-Блокнот**, яке відкрилося, наберіть наступний текст:

```
<html>
<head>  <title> Test page</title>
</head>
<body>   This is test Web page !
</body> </html>
```

- збережіть текст (**Файл => Зберегти**);
- закрийте вікно **test.html-Блокнот**;
- закрийте вікно **WEBSRVLAB**.

2.5 Компіляція програм

На комп'ютері з фізичним ім'ям **RMF01**:

- задайте **Пуск => Виконати**;
- у полі **запуску команди** наберіть **cmd** і клацніть **OK**;
- у вікні **диск:\Windows\System32\cmd.exe**;
- виконайте команди: **диск: і cd диск:\Work\WEBSRVLAB**;
- виконайте команду компіляції: **javac WebServer.java**;
- щоб переконатися у тому, що компіляція пройшла успішно і з'явився файл: **диск:\Work\WEBSRVLAB\WebServer.class**;
- виконайте команду: **dir**.

2.6 Перевірка роботи web-сервера

На комп'ютері з фізичним ім'ям **RMF01** у тому ж вікні **диск:\Windows\System32\cmd.exe** виконайте команду

java WebServer

(при цьому програма перейде в режим очікування).

На комп'ютері з фізичним ім'ям **RMF02** виконайте запуск програми-клієнта:

- запустіть **Internet Explorer**, для чого клацніть по його значку на панелі задач, і у вікні веб-браузера, що відкрилося, клацніть по значку **Зупинити**;
- відключитесь від проксі-сервера (**Сервіс => Властивості оглядача => вкладка Підключення => Налаштування мережі => зніміть прапорець Використовувати проксі-сервер для локальних підключень => ОК => ОК**);
- у текстовому полі **Адрес** впишіть: **<http://rmf51:6789/test.html>** (у вас, відповідно, інша адреса);

Примітка. так як сервер використовує нестандартний номер порту **6789**, тому він вказується у запиті. Натисніть **Enter** (при цьому повинен з'явитися вміст тестової web-сторінки сервера);

- закрийте вікно веб-браузера.

2.7 Завдання на модифікацію web-сервера

Змініть програму-сервер так, щоб вона не припиняла роботу після обробки одного запиту, а переходила в режим очікування наступного запиту - обробляла запити безперервно (без перезапуску).

2.8 Відновлення комп'ютерів в початковий стан

(даний розділ підлягає **обов'язковому** виконанню!)

На комп'ютері з фізичним ім'ям **RMF01** видаліть папку **диск:\Work**.

На комп'ютері з фізичним ім'ям **RMF02** підключіть проксі-сервер:

- запустіть **Internet Explorer**, для чого клацніть по його значку на панелі задач, і у вікні веб-браузера, яке відкрилося, клацніть по значку **Зупинити**;
- підключіть проксі-сервер (Сервіс => **Властивості оглядача** => вкладка **Підключення** => **Налаштування мережі** => установіть прапорець **Використовувати проксі-сервер для локальних підключень** => **ОК** => **ОК**);
- закрийте вікно веб-браузера.

2.9 Зміст письмового звіту

- хід роботи;
- основні функції;
- відповіді на контрольні питання.

2.10 Контрольні питання

1. Опишіть формат повідомлення-запиту протоколу HTTP за методом GET.
2. Назвіть основні елементи повідомлення-відповіді, яка генерується HTTP-сервером.
3. Яку необхідно внести зміну в початковий текст розглянутої програми web-сервера для обробки запитів файлів типу GIF?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Odom, W. CCNA 200-301 Official Cert Guide, Volume 1. Cisco Press. 2019.
2. Odom, W. CCNA 200-301 Official Cert Guide, Volume 2. Cisco Press. 2019.
3. Киричек Г.Г. Конспект лекцій з дисципліни «Проектування комп'ютерних мереж», 2024. 38 с.
4. Устенко С. А., Устенко І. В., Борисенко В. Д. Мережне програмування. - 2019.
5. Hucaby D. CCNP Routing and Switching SWITCH 300-115 Official Cert Guide / D. Hucaby. – 2nd Edition. – USA: Cisco Press, 2015. – 578 p.
6. Tulloch M., Team WS Introducing Windows Server 2012 R2. - Microsoft press, 2013.
7. Krause J. Mastering Windows Server 2016. - Packt Publishing Ltd, 2016.
8. Жураковський Б. Ю., Зенів І. О. Комп'ютерні мережі. Частина 2. Навчальний посібник. 2020.
9. Киричек Г. Г. Лабораторні роботи із застосування мережевих сервісів та протоколів. 2024.
10. Киричек Г. Г. Сервіси та служби Windows Server. 2024.