

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ
УКРАЇНИ**

Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

«Проектування комп'ютерних мереж»

для студентів фаху 7.05010201, 8.05010201 «Комп'ютерні системи та мережі» та 7.05010203, 8.05010203 «Спеціалізовані комп'ютерні системи», усіх форм навчання

Програмування. Розробка додатків.

2013

Методичні вказівки до лабораторних робіт з дисципліни “Проектування комп’ютерних мереж” для студентів фаху 7.05010201, 8.05010201 “Комп’ютерні системи та мережі” та 7.05010203, 8.05010203 “Спеціалізовані комп’ютерні системи”, усіх форм навчання. Програмування. Розробка додатків. / Укл.: Г.Г. Киричек, Т.М. Семерюк. – Запоріжжя: ЗНТУ, 2013. – 50 с.

Укладачі:

Г.Г. Киричек, доцент, к.т.н.,
Т.М. Семерюк, асистент

Рецензент:

О.І. Вершина, доцент, к.т.н.

Відповідальний за випуск:

Г.Г. Киричек, доцент, к.т.н.

Затверджено
на засіданні кафедри КСМ

Протокол № 6
від 23.02.2013

ЗМІСТ

1 Лабораторна робота. Програмування TCP-сокетів.....	4
1.1 Загальні відомості	4
1.2 Перевірка налаштувань мережевого інтерфейсу комп'ютерів.....	8
1.3 Установка середовища програмування Java	10
1.4 Створення клієнтської програми	11
1.5 Створення серверної програми	17
1.6 Компіляція програм.....	19
1.7 Перевірка роботи мережевого додатку.....	20
1.8 Завдання на модифікацію мережевого додатку.....	20
1.9 Відновлення комп'ютерів в початковий стан	21
1.10 Зміст письмового звіту.....	21
1.11 Контрольні питання.....	21
2 Лабораторна робота. Програмування UDP-сокетів	22
2.1 Загальні відомості	22
2.2 Перевірка налаштувань мережевого інтерфейсу комп'ютерів	24
2.3 Установка середовища програмування Java	26
2.4 Створення клієнтської програми	28
2.5 Створення серверної програми	32
2.6 Компіляція програм.....	35
2.7 Перевірка роботи мережевого додатку.....	36
2.8 Завдання на модифікацію мережевого додатку.....	36
2.9 Зміст письмового звіту.....	37
2.10 Контрольні питання.....	37
3 Лабораторна робота. Розробка простого web-сервера	38
3.1 Загальні відомості	38
3.2 Перевірка налаштувань мережевого інтерфейсу комп'ютерів	39
3.3 Установка середовища програмування Java	40
3.4 Створення серверної програми	42
3.5 Компіляція програм.....	47
3.6 Перевірка роботи web-сервера	48
3.7 Завдання на модифікацію web-сервера.....	48
3.8 Відновлення комп'ютерів в початковий стан	49
3.9 Зміст письмового звіту.....	49
3.10 Контрольні питання.....	49
Рекомендована література	50

1 ЛАБОРАТОРНА РОБОТА.

Програмування TCP-сокетів

Мета роботи – ознайомлення з питаннями розробки додатків для Інтернету.

1.1 Загальні відомості

Мережевий додаток складається з двох програм — клієнта і серверу. Коли ці програми запускаються, створюються клієнтський і серверний процеси, які взаємодіють один з одним, обмінюючись повідомленнями через сокети.

При створенні мережевого додатку головною задачею розробника є написання програмного коду для клієнтської і серверної частин додатку.

Існують два види додатків з архітектурою клієнт/сервер. До першого виду відносяться додатки, які підтримують стандартний протокол, описаний в документах RFC. В цьому випадку як клієнтська, так і серверна частини додатку повинні відповідати всім описам, приведеним в RFC.

Наприклад, якщо додаток використовує протокол FTP, то клієнтська і серверна програми додатку повинні бути побудовані відповідно до вимог документа RFC 959, що описує FTP-додатки.

Дотримання стандартів дозволяє незалежним розробникам підключатися до створення різних частин одного і того ж додатку.

Для додатків, розроблених за стандартом RFC, слід застосовувати порт з номером, відповідним протоколу, який використовується.

Другий вид додатків з архітектурою клієнт/сервер — нестандартні додатки. В таких додатках підтримка клієнтом і сервером яких-небудь угод, прийнятих в RFC, не обов'язкова. Розробник в особі одного або декількох чоловік розробляє клієнтську і серверну частини додатку, повністю контролюючи процес написання коду.

При цьому інші розробники не зможуть створювати програми, які будуть здатні взаємодіяти з нестандартним додатком. Для

нестандартних додатків необхідно використовувати порти з номерами, відмінними від добре відомих, вказаних в RFC.

В роботі будуть розглянуті ключові принципи розробки нестандартних додатків з архітектурою клієнт/сервер. Одне з перших рішень, яке вимушений приймати розробник, торкається вибору протоколу транспортного рівня (TCP або UDP) для свого додатку. Протокол TCP передбачає встановлення логічного з'єднання між хостами і забезпечує надійну передачу даних. Протокол UDP, навпаки, не встановлює логічного з'єднання, і передача пакетів здійснюється без гарантії їх доставки адресату.

В цій роботі ви створите простий мережевий додаток, який підтримує протокол TCP. Програми написані на мові Java. Ці програми можна було з рівним успіхом писати на мові C або C++, проте вибір обумовлений декількома істотними причинами. По-перше, Java-код простіше, зрозуміліше (що особливо важливо для новачків) і складається з меншого числа рядків, ніж код на C/C++. По-друге, програмування додатків клієнт/сервер на мові Java останніми роками отримало велику популярність, і не виключено, що в найближчі роки стане загальноприйнятим.

Процеси, що виконуються на різних обчислювальних машинах, взаємодіють один з одним за допомогою повідомлень, що посилаються через сокети. Процеси можна представити у вигляді будинків, а сокети — у вигляді дверей. Як показано на рис. 1.1, сокет є дверима між процесом і протоколом TCP. Розробник додатку повністю контролює ту частину сокета, яка відноситься до прикладного рівня, майже не маючи нагоди впливати на «транспортну частину» (за винятком завдання декількох параметрів протоколу TCP, таких як максимальний розмір буфера і сегменту).

Тепер розглянемо процес взаємодії клієнтської і серверної програм більш детально. У функції клієнта входить ініціювання з'єднання з сервером, а сервер повинен бути готовим до встановлення з'єднання. Це означає, що, по-перше, програма-сервер повинна бути запущена раніше, ніж клієнт зробить спробу встановити з'єднання, і, по-друге, що сервер повинен мати в своєму розпорядженні сокет, за допомогою якого встановлюється з'єднання.

Коли серверний процес запущений, клієнт може ініціювати установку TCP-з'єднання з сервером. Першою дією клієнтської програми є створення сокета, при цьому програма вказує адресу

серверного процесу, що складається з IP-адреси і номера порту процесу.

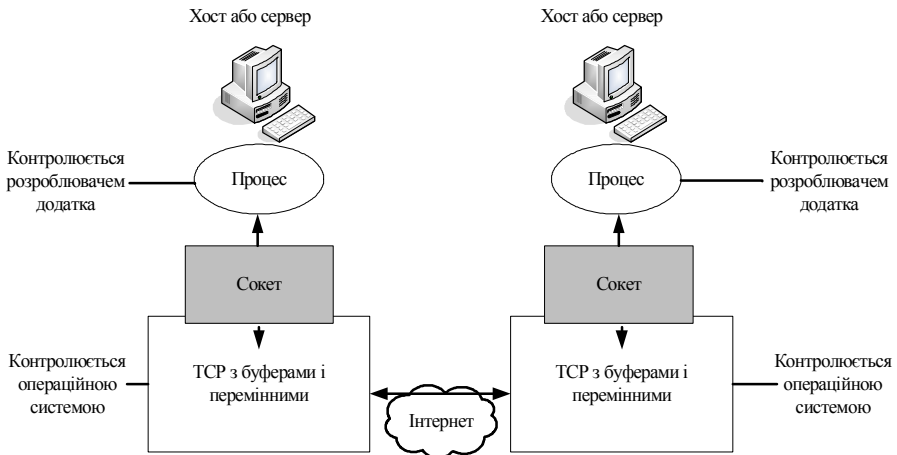


Рисунок 1.1 - Взаємодія процесів за допомогою TCP-сокетів

Після створення сокета клієнтська сторона протоколу TCP здійснює процедуру потрібного рукостискання з сервером, що закінчується встановленням з'єднання. Помітимо, що процедура рукостискання ніяк не позначається на роботі додатку.

В ході потрібного рукостискання клієнтський процес стукає у вхідні двері серверного процесу. Коли сервер чує стукіт, він створює нові двері (тобто новий сокет), що відносяться до поточного клієнта.

В прикладі, який послідує нижче, вхідними дверима є об'єкт **ServerSocket** з ім'ям **welcomeSocket**. Коли клієнт стукає в ці двері, викликається метод **accept()** об'єкту **welcomeSocket**, що створює нові двері для клієнта.

Після закінчення процедури рукостискання встановлюється TCP-з'єднання між сокетом клієнта і новим сокетом серверу, який називають *сокетом з'єднання* (рис. 1.2).

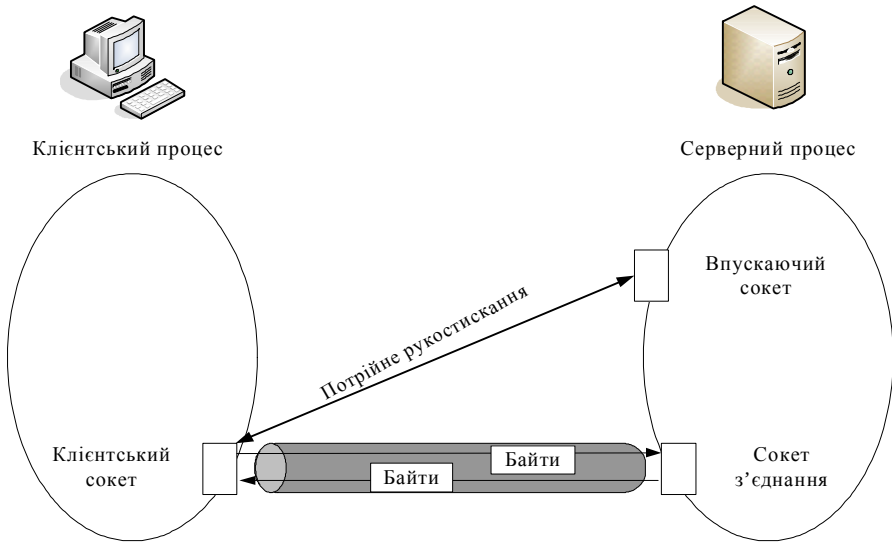


Рисунок 1.2 - Клієнтський сокет, пускаючий сокет і сокет з'єднання

З погляду додатку TCP-з'єднання є прямим віртуальним каналом між сокетом з'єднання клієнта і серверу. Клієнт може здійснювати передачу будь-яких байтів через свій сокет, при цьому протокол TCP гарантує, що сервер отримає ці байти через свій сокет без спотворень і в тому ж порядку, в якому вони були передані.

Оскільки сокети грають центральну роль в роботі додатків клієнт/сервер, розробку таких додатків часто називають програмуванням сокетів.

Перш ніж навести приклад додатку клієнт/сервер, розглянемо поняття *поток даних*. Під потоком даних розуміється послідовність символів, що передається між двома процесами. По відношенню до процесу виділяють *вхідний* і *вихідний* потоки. Вхідний потік процесу зв'язується з деяким вхідним пристроєм (клавіатурою, сокетом і т.д.), а вихідний потік — з деяким вихідним пристроєм (монітором, сокетом і т.д.).

Даний додаток функціонує таким чином:

- клієнт читає з 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).

Програма-клієнт (**TCPClient.java**) і програма-сервер (**TCPServer.java**) лише ілюструють ключові фрагменти додатку і не претендують на те, щоб називатися «добрими» з погляду практичного застосування. Для поліпшення програм приведені тексти слід було би забезпечити декількома додатковими рядками.

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 7** 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.

1.2 Перевірка настройок мережевого інтерфейсу комп'ютерів

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

- завантажить **Microsoft Windows 7** (ім'я користувача – **admin**, пароль – **1234567**);
- перевірте настройки мережевого інтерфейсу;
- задайте **Пуск => Панель управления => Сеть и Интернет => Центр управления сетями и общим доступом => Изменение параметров адаптера**;
- у вікні **Сетевые подключения** запустіть - **Подключение по локальной сети**;
- у вікні **Состояние Подключение по локальной сети** клацніть **Свойства**;

– установіть курсор на **Протокол Інтернета (TCP/IPv4)** і клацніть **Свойства**;

– виберіть опцію **Использовать следующий IP-адрес** 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 подсети: **255.255.255.0**

Основной шлюз: **10.0.9.100**

Предпочитаемый DNS-сервер: **10.0.2.1**

і клацніть **ОК**,

– з поверненням у вікно **Подключение по локальной сети** - **свойства** клацніть **ОК**,

– закрийте вікно **Состояние–Подключение по локальной сети**.

– закрийте вікно **Просмотр основных сведений о сети и настройка подключений**.

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

– завантажте **Microsoft Windows 7** (ім'я користувача – **admin**, пароль – **1234567**);

– перевірте настройки мережевого інтерфейсу:

– задайте **Пуск => Панель управления => Сеть и Интернет => Центр управления сетями и общим доступом => Изменение параметров адаптера**;

– у вікні **Сетевые подключения** запусіть - **Подключение по локальной сети**;

– у вікні **Состояние Подключение по локальной сети** клацніть **Свойства**;

– установіть курсор на **Протокол Інтернета (TCP/IPv4)** і клацніть **Свойства**;

– виберіть опцію **Использовать следующий IP-адрес** і введіть:

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

Маска подсети: **255.255.255.0**

Основной шлюз: **10.0.9.100**

Предпочитаемый DNS-сервер: 10.0.2.1

і клацніть **ОК**,

- з поверненням у вікно **Подключение по локальной сети - свойства** клацніть **ОК**,
- закрийте вікно **Состояние–Подключение по локальной сети**.
- закрийте вікно **Просмотр основных сведений о сети и настройка подключений**.

1.3 Установка средовища 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 v1.4.0;

див. **Панель управления => Программы => Программы и компоненты**), 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:

1. Встановіть середовище програмування Java, для чого:

- запустіть:

C:\Java_install\JDK1_4\j2sdk-1_4_0-win.exe;

– з відкриттям вікна **Welcome to the InstallShield Wizard for Java2 SDK, SE v1.4.0** клацніть **Next**;

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

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

G:\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**;

2. Створіть перемінні оточення, для чого:

- виконайте:

Пуск => Настройка => Панель управления => Система и безопасность => Система.

- оберіть вкладку **Защита системы**;
- у вікні **Свойства системы** перейдіть на вкладку **Дополнительно** і клацніть **Переменные среды**;
- додайте в системну перемінну **Path** ще один шлях:
- у вікні **Переменные среды** у розділі **Системные переменные** виділіть рядок з перемінною **Path**;
- клацніть **Изменить**;
- у вікні **Изменение системной переменной** у полі **Значение переменной** додайте в кінці рядка
;%G:\j2sdk1.4.0\bin
і клацніть **ОК**;
- створіть системну перемінну **CLASSPATH**;
- у вікні **Переменные среды** у розділі **Системные переменные** клацніть **Создать**;
- у вікні **Новая системная переменная**:
- у полі **Имя переменной** задайте **CLASSPATH**;
- у полі **Значение переменной** задайте
;%G:\j2sdk1.4.0\jre\lib\rt.jar
і клацніть **ОК**;
- для завершення процесу створення нових перемінних середовища клацніть **ОК** (у вікні **Переменные среды**), клацніть **ОК** (у вікні **Свойства системы**);
- закрийте вікно **Панель управления**.

3. Створіть на диску **G** папку **Work** з підкаталогом **TCPLAB**.

1.4 Створення клієнтської програми

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

1. Запустити програму:

Пуск => Все программы => Стандартные => WordPad

2. Набрати наступний текст програми:

```
import java.io.*;
import java.net.*;
class TCPClient {
    public static void
    main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;
        BufferedReader inFromUser =
            new BufferedReader(
                new InputStreamReader(System.in));
        Socket clientSocket = new Socket ("hostname", 6789);
        DataOutputStream outToServer =
            new DataOutputStream(
                clientSocket.getOutputStream());
        BufferedReader inFromServer =
            new BufferedReader(new InputStreamReader(
                clientSocket.getInputStream()));
        sentence = inFromUser.readLine();
        outToServer.writeBytes(sentence + '\n');
        modifiedSentence = inFromServer.readLine();
        System.out.println("FROM SERVER: " + modifiedSentence);
        clientSocket.close();
    }
}
```

Далі даються пояснення для кожного рядка коду

Як показано на рис. 1.3, програма **TCPClient** створює три потоки даних і один сокет.

Сокет має назву **clientSocket**.

Потік **inFromUser** є вхідним потоком даних програми і пов'язаний із стандартним пристроєм вводу (клавіатурою). Всі символи, що вводяться з клавіатури користувачем, потрапляють в потік **inFromUser**.

Інший вхідний потік даних програми, **inFromServer**, пов'язаний з сокетом і складається з символів, що передаються серверною стороною додатку.

Вихідний потік даних програми **TCPClient** має назву **outToServer** і також пов'язаний з сокетом. Цей потік містить символи, що передаються серверу для обробки.

Перші два рядки програми містять імена двох Java-пакетів, **java.io** і **java.net**:

```
import java.io.*;
import java.net.*;
```

Пакет **java.io** містить класи вхідних і вихідних потоків даних. Звичайно цими класами є **BufferedReader** і **DataOutputStream**, які і були використані в програмі.

Пакет **java.net** береже класи, що підтримують роботу з комп'ютерною мережею (**Socket** і **ServerSocket**). Об'єкт **clientSocket** програми породжений класом **Socket**.

```
class TCPClient {
    public static void
    main(String argv[]) throws Exception
    {.....}
}
```

Конструкції, аналогічні приведеній вище, є стандартними в практиці програмування на Java. Перший рядок є початком блоку визначення класу. В ньому присутнє ключове слово **class** і ім'я визначуваного класу **TCPClient**. Клас може містити змінні і методи; в оголошенні класу вони укладені у фігурні дужки і фактично складають блок визначення класу. В класі **TCPClient** немає змінних, він містить єдиний метод з ім'ям **main()**. Методи схожі на процедури і функції мов, подібних C; метод **main()** також грає роль, схожу з функцією **main()** в C або C++.

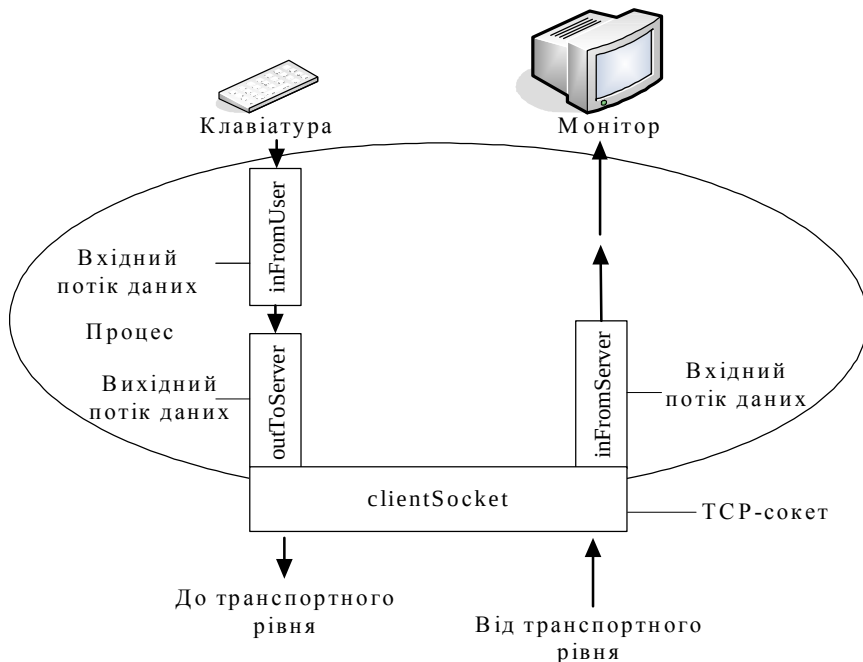


Рисунок 1.3 - Програма **TCPClient** організовує три потоки даних для символів

Коли інтерпретатор Java виконує додаток (будучи викликаним управляючим класом додатку), він звертається до методу **main()** цього класу. Метод **main()** викликає всю решту методів, що використовуються в додатку. Якщо в даний момент ви вперше стикаєтеся з Java-додатком, можете не звертати уваги на ключові слова **public**, **static**, **void**, **main** і **throws Exception**.

String sentence;

String modifiedSentence;

Приведені два рядки є оголошеннями об'єктів типу **String**. Об'єкт **sentence** призначений для зберігання рядка, що вводиться користувачем і передається серверу. Об'єкт **modifiedSentence** містить рядок, що приймається від серверу і виводиться на стандартний пристрій виводу.

```
BufferedReader inFromUser =  
new BufferedReader(  
new InputStreamReader(System.in));
```

Тут відбувається створення потокового об'єкту **inFromUser** типу **BufferedReader**. Ініціалізація потоку даних проводиться об'єктом **System.in**, що пов'язує потік із стандартним пристроєм вводу. Після виконання цієї команди клієнту починають передаватися символи, що вводяться користувачем з клавіатури.

```
Socket clientSocket = new Socket ("hostname", 6789);
```

В приведеному рядку відбувається створення об'єкту **clientSocket** типу **Socket**. Крім того, при цьому ініціюється TCP-з'єднання між клієнтом і сервером. Слово **hostname** необхідне замінити ім'ям хоста, зберігши лапки (наприклад, "**fling.seas.upenn.edu**"). Перед початком установки TCP-з'єднання клієнт проводить DNS-запит IP-адреси хоста. Значення **6789** — це номер порту; ви можете вибрати інше число, проте необхідно пам'ятати, що клієнтська і серверна сторони повинні використовувати один і той же номер порту. Як згадувалося раніше, IP-адреса хоста в сукупності з номером порту додатку ідентифікує серверний процес.

```
DataOutputStream outToServer =  
new DataOutputStream (  
clientSocket.getOutputStream());  
BufferedReader inFromServer =  
new BufferedReader(new InputStreamReader(  
clientSocket.getInputStream());
```

Тут відбувається створення потокових об'єктів, пов'язаних з сокетом. Потік **outToServer** забезпечує вивід програми через сокет, а потік **inFromServer** призначений для прийому даних від серверної сторони (рис. 1.3).

```
sentence = inFromUser.readLine();
```

Цей рядок поміщає послідовність символів, що вводяться користувачем, в змінну **sentence**. Введення закінчується натисненням користувачем клавіші **Enter**. Як бачимо, для поміщення символів в змінну використовується потоковий об'єкт **inFromUser**.

outToServer.writeBytes(sentence + '\n');

Тут рядок **sentence** разом з символом повернення каретки поміщається у вихідний потік **outToServer**. Після цього він буде переданий через сокет в канал, що з'єднує клієнта з сервером.

modifiedSentence = inFromServer.readLine();

Відповідь сервера приймається у вхідний потік **inFromServer**, звідки прийнятий рядок копіюється в змінну **modifiedSentence**. Поміщення символів в рядок **modifiedSentence** продовжується до тих пір, поки не буде отриманий символ повернення каретки.

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

Тут відбувається вивід прийнятого від сервера рядка на монітор.

clientSocket.close();

Цей рядок закриває сокет, а отже, і TCP-з'єднання між клієнтом і сервером. Виконання цієї команди веде до відправки TCP-клієнтом повідомлення TCP-серверу.

3. Збережіть файл з текстом програми:

- клацніть **Файл => Сохранить как;**
- у вікні **Сохранить как;**
- у полі **Папка** укажіть **G:\Work\TCPLAB;**
- у полі **Тип файла** виберіть **Текстовый документ MS-DOS;**
- у полі **Имя файла** укажіть **TCPClient.java;**
- клацніть **Сохранить;**
- у підвікні **WordPad** клацніть **Да;**
- закрийте вікно **TCPClient.java.txt-WordPad;**
- відкрийте папку **G:\Work\TCPLAB;**
- зніміть прапорець **Скрывать расширения для зарегистрированных типов файлов (Пуск => Панель управления => Оформление и персонализация);**
- оберіть вкладку **Параметры папок => Вид;**
- перейменуйте файл **TCPClient.java.txt** на **TCPClient.java;**
- закрийте вікно **TCPLAB.**

4. Внесіть зміну в текст програми:

- відкрийте файл **G:\Work\TCPLAB\TCPClient.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 => Сохранить**, клацніть **Да**;
- закрийте вікно з файлом, що редагується;
- закрийте вікно **TCPLAB**.

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):

1. Запустіть програму:

Пуск => Все программы => Стандартные => WordPad

2. Наберіть наступний текст програми:

```
import java.io.*;
import java.net.*;
class TCPServer {
    public static void
    main(String argv[]) throws Exception
    {
        String clientSentence;
        String capitalizedSentence;
        ServerSocket welcomeSocket = new ServerSocket (6789);
        while (true) {
            Socket connectionSocket = welcomeSocket.
            accept();
            BufferedReader inFromClient =
            new BufferedReader(
                new InputStreamReader(
                    connectionSocket.getInputStream()));
            DataOutputStream outToClient =
            new DataOutputStream(
```

```

        connectionSocket.getOutputStream());
    clientSentence = inFromClient.readLine();
    capitalizedSentence =
        clientSentence.toUpperCase() + '\n';
    outToClient.writeBytes(capitalizedSentence);
    }
}
}

```

Програма **TCPServer** має багато спільного з програмою **TCPClient**; розглянемо її докладніше, опустивши рядки, спільні з програмою **TCPClient** і описані вище.

```
ServerSocket welcomeSocket = new ServerSocket (6789);
```

Тут відбувається створення об'єкту **welcomeSocket** типу **ServerSocket**. Об'єкт **welcomeSocket** є впускаючим сокетом, за допомогою якого клієнт встановлює первинний контакт з сервером. Для цього сокета використовується порт з номером **6789**, співпадаючим з номером порту сокета клієнта.

Протокол TCP встановлює прямий віртуальний канал між сокетом **clientSocket** на клієнтській стороні і **connectionSocket** на серверній стороні, після чого клієнт і сервер можуть вільно здійснювати обмін інформацією. Після створення сокета **connectionSocket** сервер може знов використовувати сокет **welcomeSocket** для ініціювання нових з'єднань з клієнтами (наведена програма не обробляє нових з'єднань, однак її можна модифікувати відповідним чином).

Далі програма створює декілька потокових об'єктів, аналогічних **clientSocket**.

```
capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

Ця команда основна в додатку. Вона виконує отримання рядка, переданого клієнтом, переведення рядка у верхній регістр за допомогою методу **toUpperCase()** і додавання в кінець символу повернення каретки.

Всі інші команди програми є периферійними і не стосуються взаємодії серверу з клієнтом.

3. Збережіть файл з текстом програми:

- клацніть **Файл => Сохранить как**,
- у вікні **Сохранить как**:
- у полі **Папка** укажіть **G:\Work\TCPLAB**;
- у полі **Тип файла** виберіть **Текстовый документ MS-DOS**;
- у полі **Имя файла** укажіть **TCPServer.java**;
- клацніть **Сохранить**;
- у підвікні **WordPad** клацніть **Да**;
- закрийте вікно **TCPServer.java.txt-WordPad**;
- відкрийте папку **G:\Work\TCPLAB**;
- зніміть прапорець **Скрывать расширения для зарегистрированных типов файлов (Пуск => Панель управления => Оформление и персонализация)**;
- оберіть вкладку **Параметры папок => Вид**;
- перейменуйте файл **TCPServer.java.txt** на **TCPServer.java**;
- закрийте вікно **TCPLAB**.

1.6 Компіляція програм

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

- задайте **Пуск => Выполнить**;
- у полі **запуску команди** наберіть **cmd** і клацніть **ОК**;
- у вікні **G:\Windows\System32\cmd.exe**:
- виконайте команди

G: і **cd G:\Work\TCPLAB**;

- виконайте команду компіляції

javac TCPServer.java;

- щоб переконатися у тому, що компіляція пройшла успішно

і з'явився файл

G:\Work\TCPLAB\TCPServer.class;

- виконайте команду

dir.

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

- задайте **Пуск > Выполнить**;
- у полі **запуску команди** наберіть **cmd** і клацніть **ОК**;
- у вікні **G:\Windows\System32\cmd.exe**;

- виконайте команди

G: i cd G:\Work\TCPLAB;

- виконайте команду компіляції

javac TCPClient.java;

- щоб переконатися у тому, що компіляція пройшла успішно і з'явився файл

G:\Work\TCPLAB\TCPClient.class;

- виконайте команду

dir.

1.7 Перевірка роботи мережевого додатку

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

java TCPServer

(при цьому програма перейде в режим очікування).

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

java TCPClient

- 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

1. 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.
2. Змініть додаток так, щоб поведінка додатку була наступною:
 - одна серверна програма чекає з'єднання відразу по двох портах;

- 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.

1.9 Відновлення комп'ютерів в початковий стан

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

1.10 Зміст письмового звіту

1. Хід роботи.
2. Основні функції.
3. Відповіді на контрольні питання.

1.11 Контрольні питання

1. Що таке сокет?
2. Укажіть розташування сокета у моделі взаємодії відкритих систем OSI.
3. Назвіть етапи установки TCP та з'єднання з сервером.
4. Розкрийте поняття потоку даних.

2 ЛАБОРАТОРНА РОБОТА. Програмування UDP-сокетів

Мета роботи – ознайомлення з питаннями розробки додатків для Інтернету.

2.1 Загальні відомості

З погляду процесів взаємодія за протоколом TCP виконується через канал, який існує до тих пір, поки один з процесів не закриє його. Передача процесом інформації зводиться до того, що процес «скидає» байти безпосередньо в канал, при цьому немає необхідності забезпечувати байти адресою призначення, оскільки канал логічно пов'язаний з адресатом. Крім того, передача по каналу являється надійною, тобто послідовність байтів, що приймається, в точності відповідає послідовності, що передається.

Протокол UDP, як і TCP, реалізує взаємодію між двома процесами, які виконуються на різних хостах; проте UDP-взаємодія принципово відрізняється від TCP-взаємодії:

- по-перше, логічне з'єднання між хостами відсутнє, оскільки не існує процедури рукостискання, в ході якої встановлювався б канал;
- по-друге, будь-яка послідовність байтів, що передається, повинна забезпечуватися адресою призначення, що є сукупністю IP-адреси хоста і номера порту процесу.

IP-адреса, номер порту і послідовність інформаційних байтів ми будемо називати пакетом.

Після того, як пакет створений, процес відправляє його адресату через свій сокет. Транспортний рівень (протокол UDP) уживає дії для доставки пакету за адресою призначення, проте не дає щодо доставки ніяких гарантій. Іншими словами, протокол UDP забезпечує ненадійну передачу даних між процесами.

В роботі, у вигляді прикладу, ми модифікуємо для протоколу UDP мережевий додаток з попередньої лабораторної роботи. Як ви побачите, коди програм для протоколів TCP і UDP значно різняться:

- по-перше, через відсутність процедури рукостискання немає необхідності у впускаючому сокеті;

- по-друге, вхідні і вихідні потоки даних не пов'язані з сокетом;
- по-третє, при пересиланні кожної групи байтів вказується IP-адреса хоста-одержувача і номер порту сервера;
- по-четверте, приймаючій стороні доводиться виділяти інформаційні байти з прийнятих пакетів.

Нагадаємо порядок дій, що виконуються нашим додатком:

- клієнт зчитує з 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. 2.1 приведена схема взаємодії клієнта і сервера при використанні протоколу UDP.

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 7** 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.

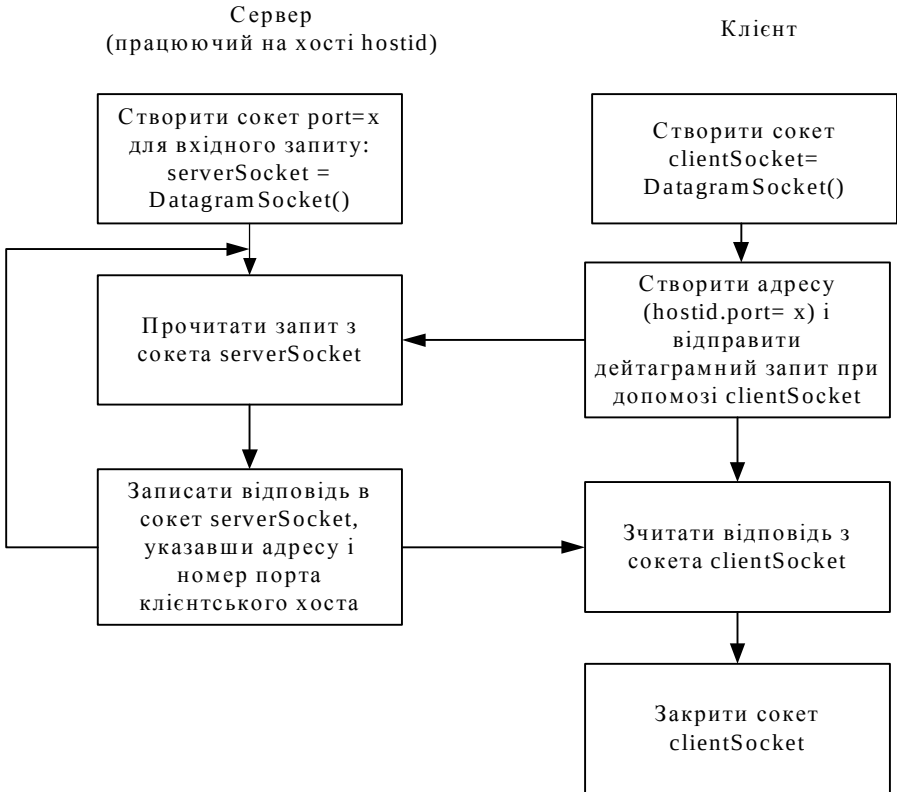


Рисунок 2.1 - Додаток клієнт/сервер, який використовує транспортні служби без установлення логічного з'єднання

2.2 Перевірка налаштувань мережевого інтерфейсу комп'ютерів

1. На комп'ютері з фізичним ім'ям **RMF01**:
 - завантажить **Microsoft Windows 7** (ім'я користувача – **admin**, пароль – **1234567**);
 - перевірте настройки мережевого інтерфейсу:

– задайте **Пуск => Панель управления => Сеть и Интернет => Центр управления сетями и общим доступом => Изменение параметров адаптера;**

– у вікні **Сетевые подключения** запустіть - **Подключение по локальной сети;**

– у вікні **Состояние Подключение по локальной сети** клацніть **Свойства;**

– установіть курсор на **Протокол Интернета (TCP/IPv4)** і клацніть **Свойства;**

– виберіть опцію **Использовать следующий IP-адрес** і введіть:

IP-адрес: **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 подсети: **255.255.255.0**

Основной шлюз: **10.0.9.100**

Предпочитаемый DNS-сервер: **10.0.2.1**

і клацніть **ОК,**

– з поверненням у вікно **Подключение по локальной сети - свойства** клацніть **ОК,**

– закрийте вікно **Состояние–Подключение по локальной сети.**

– закрийте вікно **Просмотр основных сведений о сети и настройка подключений.**

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

– завантажте **Microsoft Windows 7** (ім'я користувача – **admin**, пароль – **1234567**);

– перевірте настройки мережевого інтерфейсу:

– задайте **Пуск => Панель управления => Сеть и Интернет => Центр управления сетями и общим доступом => Изменение параметров адаптера;**

– у вікні **Сетевые подключения** запустіть - **Подключение по локальной сети;**

– у вікні **Состояние Подключение по локальной сети** клацніть **Свойства;**

– установіть курсор на **Протокол Интернета (TCP/IPv4)** і клацніть **Свойства;**

– виберіть опцію **Использовать следующий IP-адрес** і введіть:

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

Маска подсети: 255.255.255.0

Основной шлюз: 10.0.9.100

Предпочитаемый DNS-сервер: 10.0.2.1

і клацніть **ОК**,

– з поверненням у вікно **Подключение по локальной сети - свойства** клацніть **ОК**,

– закрийте вікно **Состояние–Подключение по локальной сети**.

– закрийте вікно **Просмотр основных сведений о сети и настройка подключений**.

2.3 Установка средовища 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 v1.4.0;

див. **Панель управления => Программы => Программы и компоненты**), 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:

1. Установіть середовище програмування Java, для чого:

– запустіть:

C:\Java_install\JDK1_4\j2sdk-1_4_0-win.exe;

– з відкриттям вікна **Welcome to the InstallShield Wizard for Java2 SDK, SE v1.4.0** клацніть **Next**;

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

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

G:\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**;

2. Створіть перемінні оточення, для чого:

- виконайте:

Пуск => Настройка => Панель управления => Система и безопасность => Система.

- оберіть вкладку **Защита системы**;
- у вікні **Свойства системы** перейдіть на вкладку **Дополнительно** і клацніть **Переменные среды**;

- додайте в системну перемінну **Path** ще один шлях:
- у вікні **Переменные среды** у розділі **Системные переменные** виділіть рядок з перемінною **Path**;

- клацніть **Изменить**;

- у вікні **Изменение системной переменной** у полі **Значение переменной** додайте в кінці рядка

;G:\j2sdk1.4.0\bin

і клацніть **ОК**;

- створіть системну перемінну **CLASSPATH**:

- у вікні **Переменные среды** у розділі **Системные переменные** клацніть **Создать**;

- у вікні **Новая системная переменная**:

- у полі **Имя переменной** задайте **CLASSPATH**;

- у полі **Значение переменной** задайте

.;G:\j2sdk1.4.0\jre\lib\rt.jar

і клацніть **ОК**;

- для завершення процесу створення нових перемінних середовища клацніть **ОК** (у вікні **Переменные среды**), клацніть **ОК** (у вікні **Свойства системы**);

- закрийте вікно **Панель управления**.

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

2.4 Створення клієнтської програми

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

1. Запустіть програму:

Пуск > Все программы > Стандартные > WordPad

2. 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();
    }
}
```

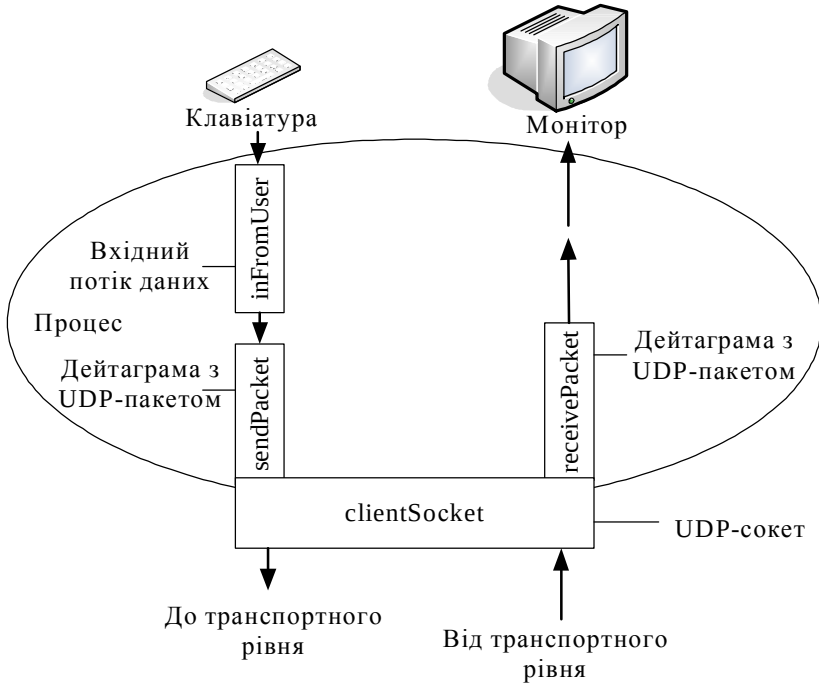


Рисунок 2.2 - В програмі **UDPCliient** є єдиний потік даних; сокет приймає і передає пакети процесу

Як показано на рис. 2.2, програма **UDPCliient** створює один потік даних і один сокет.

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

Зверніть увагу, що при використанні протоколів UDP і TCP типи сокетів розрізняються: у випадку TCP звичайно 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 був 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 **UDPCClient**.

```
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, що 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).

3. Збережіть файл з текстом програми:

- клацніть **Файл => Сохранить как;**
- у вікні **Сохранить как;**
- у полі **Папка** укажіть **G:\Work\UDPLAB;**
- у полі **Тип файла** виберіть **Текстовый документ MS-DOS;**
- у полі **Имя файла** укажіть **UDPClient.java;**
- клацніть **Сохранить;**
- у підвікні **WordPad** клацніть **Да;**
- закрийте вікно **UDPClient.java.txt-WordPad;**
- відкрийте папку **G:\Work\UDPLAB;**
- зніміть прапорець **Скрывать расширения для зарегистрированных типов файлов (Пуск => Панель управления => Оформление и персонализация);**
- оберіть вкладку **Параметры папок>Вид;**
- перейменуйте файл **UDPClient.java.txt** на **UDPClient.java;**
- закрийте вікно **UDPLAB.**

4. Внесіть зміну в текст програми:

- відкрийте файл **G:\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 => Сохранить**, клацніть **Да;**
- закрийте вікно з файлом, що редагується;
- закрийте вікно **UDPLAB.**

2.5 Створення серверної програми

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

1. Запустіть програму:

Пуск => Все программы => Стандартные => WordPad

2. 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 рис. 2.3, програма **UDPServer** створює єдиний сокет з ім'ям **serverSocket** типу **DatagramSocket**; цей же тип використовувався нами в програмі-клієнті додатку.

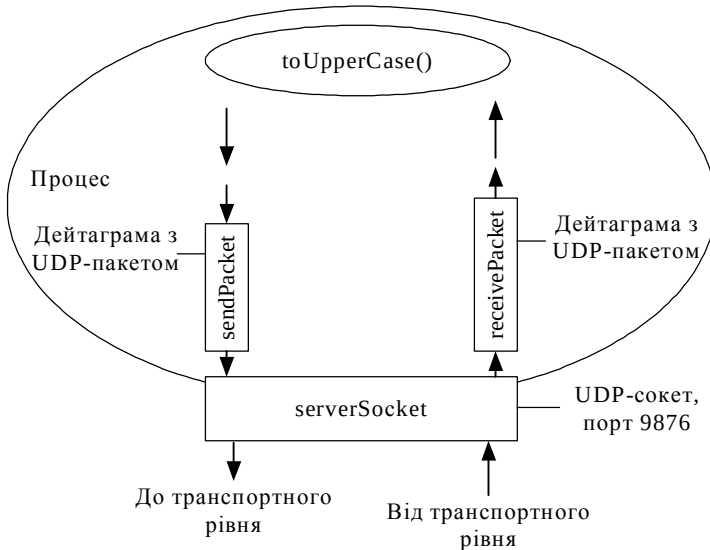


Рисунок 2.3 - Програма **UDPServer** не має потоків; сокет приймає і передає пакети процесу

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 рукоштовування 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 **UDPClien**).

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

3. Збережіть файл з текстом програми:

- клацніть **Файл => Сохранить как**;
- у вікні **Сохранить как**;
- у полі **Папка** укажіть **G:\Work\UDPLAB**;
- у полі **Тип файла** виберіть **Текстовый документ MS-DOS**;
- у полі **Имя файла** укажіть **UDPServer.java**;
- клацніть **Сохранить**;
- у підвікні **WordPad** клацніть **Да**;
- закрийте вікно **UDPServer.java.txt-WordPad**;
- відкрийте папку **G:\Work\UDPLAB**;
- зніміть прапорець **Скрывать расширения** для зареєстрованих типів файлів (**Пуск => Панель управления => Оформление и персонализация**);
- оберіть вкладку **Параметры папок => Вид**;
- перейменуйте файл **UDPServer.java.txt** на **UDPServer.java**;
- закрийте вікно **UDPLAB**.

2.6 Компіляція програм

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

- задайте **Пуск => Выполнить**;
- у полі **запуску команди** наберіть **cmd** і клацніть **ОК**;
- у вікні **G:\Windows_2\System32\cmd.exe**;

- виконайте команди: **G: i cd G:\Work\UDPLAB;**
- виконайте команду компіляції: **javac UDPServer.java;**
- щоб переконатися у тому, що компіляція пройшла успішно і з'явився файл: **G:\Work\UDPLAB\UDPServer.class;**
- виконайте команду: **dir.**

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

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

2.7 Перевірка роботи мережевого додатку

1. На комп'ютері з фізичним ім'ям **RMF01** у тому ж вікні **G:\Windows\System32\cmd.exe** виконайте команду **java UDPServer** (при цьому програма перейде в режим очікування).

2. На комп'ютері з фізичним ім'ям **RMF02** у тому ж вікні **G:\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.

2.8 Завдання на модифікацію мережевого додатку

1. 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.

2. Змініть додаток так, щоб поведінка додатку була наступною:

- одна серверна програма чекає з'єднання відразу по двох портах;

- 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 **G:\Work**.

2.9 Зміст письмового звіту

1. Хід роботи.
2. Основні функції.
3. Відповіді на контрольні питання.

2.10 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?

3 ЛАБОРАТОРНА РОБОТА. Розробка простого web-сервера

Мета роботи – ознайомлення з питаннями розробки додатків для Інтернету.

3.1 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 7** 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.

3.2 Перевірка налаштувань мережевого інтерфейсу комп'ютерів

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

- завантажить **Microsoft Windows 7** (ім'я користувача – **admin**, пароль – **1234567**);
- перевірте налаштування мережевого інтерфейсу:
- задайте **Пуск => Панель управління => Сеть и Интернет => Центр управления сетями и общим доступом => Изменение параметров адаптера**;
- у вікні **Сетевые подключения** запустіть - **Подключение по локальной сети**;
- у вікні **Состояние Подключение по локальной сети** клацніть **Свойства**;
- установіть курсор на **Протокол Интернета (TCP/IPv4)** і клацніть **Свойства**;
- виберіть опцію **Использовать следующий IP-адрес** 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 подсети: **255.255.255.0**

Основной шлюз: **10.0.9.100**

Предпочитаемый DNS-сервер: **10.0.2.1**

і клацніть **ОК**,

- з поверненням у вікно **Подключение по локальной сети - свойства** клацніть **ОК**,
- закрийте вікно **Состояние–Подключение по локальной сети**.
- закрийте вікно **Просмотр основных сведений о сети и настройка подключений**.

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

- завантажить **Microsoft Windows 7** (ім'я користувача – **admin**, пароль – **1234567**);
- перевірте налаштування мережевого інтерфейсу:

– задайте **Пуск => Панель управления => Сеть и Интернет => Центр управления сетями и общим доступом => Изменение параметров адаптера;**

– у вікні **Сетевые подключения** запустіть - **Подключение по локальной сети;**

– у вікні **Состояние Подключение по локальной сети** клацніть **Свойства;**

– установіть курсор на **Протокол Интернета (TCP/IPv4)** і клацніть **Свойства;**

– виберіть опцію **Использовать следующий IP-адрес** і введіть:

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

Маска подсети: 255.255.255.0

Основной шлюз: 10.0.9.100

Предпочитаемый DNS-сервер: 10.0.2.1

і клацніть **ОК,**

– з поверненням у вікно **Подключение по локальной сети - свойства** клацніть **ОК,**

– закрийте вікно **Состояние–Подключение по локальной сети.**

– закрийте вікно **Просмотр основных сведений о сети и настройка подключений.**

3.3 Установка средовища 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 v1.4.0;

див. **Панель управления => Программы => Программы и компоненты**), 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:

1. Установіть середовище програмування Java, для чого:

- запустіть:
C:\Java_install\JDK1_4\jdk-1_4_0-win.exe;
- з відкриттям вікна **Welcome to the InstallShield Wizard for Java2 SDK, SE v1.4.0** клацніть **Next;**
- у вікні **License Agreement** клацніть **Yes;**
- у вікні **Choose Destination Location** погодьтеся з пропозицією створити папку:
G:\jdk1.4.0;
- 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;**

2. Створіть перемінні оточення, для чого:

- виконайте:

Пуск => Настройка => Панель управления => Система и безопасность => Система.

- оберіть вкладку **Защита системы;**
- у вікні **Свойства системы** перейдіть на вкладку **Дополнительно** і клацніть **Переменные среды;**
- додайте в системну перемінну **Path** ще один шлях:
- у вікні **Переменные среды** у розділі **Системные переменные** виділіть рядок з перемінною **Path;**
- клацніть **Изменить;**
- у вікні **Изменение системной переменной** у полі **Значение переменной** додайте в кінці рядка
;G:\jdk1.4.0\bin
і клацніть **ОК;**
- створіть системну перемінну **CLASSPATH:**
- у вікні **Переменные среды** у розділі **Системные переменные** клацніть **Создать;**
- у вікні **Новая системная переменная:**
- у полі **Имя переменной** задайте **CLASSPATH;**

- у полі **Значение переменной** задайте
.;G:\j2sdk1.4.0\jre\lib\rt.jar
і клацніть **ОК**;
- для завершення процесу створення нових перемінних середовища клацніть **ОК** (у вікні **Переменные среды**), клацніть **ОК** (у вікні **Свойства системы**);
- закрийте вікно **Панель управления**.

3.4 Створення серверної програми

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

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

2. Запустіть програму:

Пуск => Все программы => Стандартные => WordPad

3. 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 numOfBytes = (int) file.length();
    FileInputStream inFile = new FileInputStream (fileName);
    byte[] fileInBytes = new byte[numOfBytes];
    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: "
+ numOfBytes + "\r\n");
    outToClient.writeBytes("\r\n");
    outToClient.write(fileInBytes, 0, numOfBytes);
    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 файла, 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 файла знаходиться в полі, що слідує відразу після слова **GET**. Ім'я файла поміщається в рядок з ім'ям **fileName**.

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

```
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.

4. Збережіть файл з текстом програми:

- клацніть **Файл => Сохранить как**;
- у вікні **Сохранить как**;
- у полі **Папка** укажіть **G:\Work\WEBSRVLAB**;
- у полі **Тип файла** виберіть **Текстовый документ MS-DOS**;
- у полі **Имя файла** укажіть **WebServer.java**;
- клацніть **Сохранить**;
- у підвікні **WordPad** клацніть **Да**;
- закрийте вікно **WebServer.java.txt-WordPad**;
- відкрийте папку **G:\Work\WEBSRVLAB**;
- зніміть прапорець **Скрывать расширения** для

зарегистрированных типов файлов (Пуск => Панель управления
=> Оформление и персонализация);

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

5. Створіть документ **G:\Work\WEBSRVLAB\test.html:**

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

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

- відкрийте документ (клацніть правою кнопкою миші на документі **test.html**, виберіть **Открыть с помощью => Блокнот => OK**);

- 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 => Сохранить**);
- закрийте вікно **test.html-Блокнот**;
- закрийте вікно **WEBSRVLAB.**

3.5 Компіляція програм

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

- задайте **Пуск => Выполнить**;
- у полі **запуску команды** наберіть **cmd** і клацніть **OK**;
- у вікні **G:\Windows\System32\cmd.exe**:

- виконайте команди: **G: i cd G:\Work\WEBSRVLAB;**
- виконайте команду компіляції: **javac WebServer.java;**
- щоб переконатися у тому, що компіляція пройшла успішно і з'явився файл: **G:\Work\WEBSRVLAB\WebServer.class;**
- виконайте команду: **dir.**

3.6 Перевірка роботи web-сервера

1. На комп'ютері з фізичним ім'ям **RMF01** у тому ж вікні **G:\Windows\System32\cmd.exe** виконайте команду
java WebServer
 (при цьому програма перейде в режим очікування).

2. 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 **Остановить**;

- відключитесь від проксі-сервера (**Сервис => Свойства обозревателя => вкладка Подключение => Настройка сети => зніміть прапорець Использовать прокси-сервер для локальных подключений => 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.

3.7 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, тобто обробляла запити безперервно (без перезапуску).

3.8 Відновлення комп'ютерів в початковий стан

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

1. На комп'ютері з фізичним ім'ям **RMF01** видалить папку **G:\Work**.

2. На комп'ютері з фізичним ім'ям **RMF02** підключить проксі-сервер:

– запустить **Internet Explorer**, для чого клацніть по його значку на панелі задач, і у вікні веб-браузера, яке відкрилося, клацніть по значку **Остановить**;

– підключить проксі-сервер (**Сервис => Свойства обозревателя => вкладка Подключение => Настройка сети => установить прапорець Использовать прокси-сервер для локальных подключений => ОК => ОК**);

– закрийте вікно веб-браузера.

3.9 Зміст письмового звіту

1. Хід роботи.
2. Основні функції.
3. Відповіді на контрольні питання.

3.10 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 ЛИТЕРАТУРА

1. Олифер В. Г. Компьютерные сети. Принципы, технологии, протоколы: / В. Г. Олифер, Н. А.Олифер. // Учебник для вузов. — 4-е изд. — СПб.: Питер, 2011. — 944с.: ил.

2. Закер Крэйг. Планирование и поддержка сетевой инфраструктуры Microsoft Windows Server 2003. Учебный курс MCSE / Закер Крэйг. // Пер. с англ.— М.: Издательско-торговый дом «Русская Редакция», 2005. — 544с. : ил.

3. Нортроп Т. Проектирование сетевой инфраструктуры Windows Server 2008. Учебный курс Microsoft / Т. Нортроп, Дж. К. Макин // Пер. с англ. — М. : Издательство «Русская Редакция», 2009. — 592с. : ил.

4. Палмер М. Проектирование и внедрение компьютерных сетей / М. Палмер, Р. Синклер. — СПб.:БХВ-Петербург, 2004. — 752с.; ил.

5. Колисниченко Д.Н. Беспроводная сеть дома и в офисе / Д.Н. Колисниченко. — СПб.: БХВ-Петербург, 2009. — 480с.; ил.