Лабораторна робота № 9

Тема: Реалізація клієнтських та серверних сокетів

Мета: Вивчення принципів створення клієнтських та серверних сокетів

Класи, що визначено в пакеті java.net, забезпечують прості у використані засоби, за допомогою яких використовуються мережеві ресурси. В Java до основи роботи в мережі покладено поняття сокету. Сокет визначається за допомогою порта. Мережевий обмін ґрунтується на клієнт-серверній архітектурі. Клієнтська частина використовує динамічні порти, серверна — визначені статичні порти: 21 — FTP, 22 — ssh, 23 — Telnet, 25 — e-mail, 80 — HTTP, тощо. Для організації мережевого обміну використовується стек протоколів TCP/IP. Стек мережевих протоколів TCP/IP підтримується в Java через розширення наявних інтерфейсів потокового введення-виводу, а також впровадження нових засобів, необхідних для побудови об’єктів введення-виводу через мережу.

Інтерфейси пакету java.net:

ContentHandlerlFactory	ProtocolFamily
CookiePolicy	SocketImplFactory
CookieStore	SocketOption
DatagramSocketImplFactory	SocketOptions
FileNameMap	URLStreamHandlerFactory

Класи пакету java.net:

Authenticator	InetAddress	SocketAddress
CacheRequest	InetSocketAddress	SocketImpl
CacheResponse	InterfaceAddress	SocketPermission
ContentHandler	JarURLConnection	StandardSocketOption
CookieHandler	MulticastSocket	URI
CookieManager	NetPermission	URL
DatagramPacket	NetworkInterface	URLClassLoader
DatagramSocket	PasswordAuthentication	URLConnection
DatagramSocketImpl	Proxy	URLDecoder
HttpCookie	ProxySelector	URLEncoder
HttpURLConnection	ResponseCache	URLPermission
IDN	SecureCacheResponce	URLStreamHandler
Inet4Address	ServerSocket	
Inet6Address	Socket	

Клас InetAddress слугує для інкапсуляції як числової IP-адреси, так і її доменного імені. Для взаємодії з цим класом використовується ім’я IP-вузла. Цей клас може оперувати адресами як протоколу IPv4, так і протоколу IPv6. Клас не має доступних конструкторів. Для створення об’єкту класу InetAddress використовується один з доступних в ньому фабричних методів. Фабричні методи позначають певний зв’язок, за яким статичні методи класу

повертають екземпляр цього класу. Це робиться замість перевантаження конструктору з різними списками параметрів. Наявність однозначних імен методів робить більш чітким очікуваний результат. Найчастіше використовуються наступні фабричні методи цього класу:

static InetAddress getLocalHost( ) throws UnknownHostException

Метод, що повертає об’єкт типу InetAddress, який є локальним вузлом. Метод генерує виключення UnknownHostException, якщо він не може отримати адресу за іменем localhost.

static InetAddress getByName(String host_name)
throws UnknownHostException

Метод, що повертає об’єкт типу InetAddress, ім’я якого задано параметром host_name. Метод генерує виключення UnknownHostException, якщо він не може отримати адресу за іменем вузла.

static InetAddress[ ] getAllByName(String host_name)
throws UnknownHostException

Метод, що повертає масив об’єктів типу InetAddress, ім’я якого задано параметром host_name. Метод генерує виключення UnknownHostException, якщо він не може виконати принаймні одне перетворення імені у адресу.

Нижче наведено приклад використання методів:

Приклад 1

```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class Example {
    public static void main(String args[]) throws UnknownHostException {
        InetAddress Address = InetAddress.getLocalHost();
        System.out.println(Address);
        Address = InetAddress.getByName("www.ukr.net");
        System.out.println(Address);
        InetAddress gw[] = InetAddress.getAllByName( host: "www.google.com");
        for (int i = 0; i < gw.length; i++)
            System.out.println(gw[i]);
        byte[] addr = {(byte) 192, (byte) 168, (byte) 0, (byte) 1};
        Address = InetAddress.getByAddress(addr);
        System.out.println(Address.getHostName());
    }
}
```

До об’єктів класу InetAddress можуть застосовуватись наступні методи:

boolean equals(Object other) Повертає true, якщо об’єкт, до якого застосовується метод

	має таку саму адресу, що і об’єкт other
byte[ ] getAddress( )	Повертає масив байтів, який є IP адресою у порядку слідування байтів мережею
String getHostAddress( )	Повертає символьний рядок, який є адресою вузла об’єкту InetAddress
String getHostName( )	Повертає символьний рядок з іменем вузла об’єкту InetAddress
boolean isMulticastAddress( )	Повертає логічне значення true, якщо адрес є груповим (широкомовним), інакше — повертається значення false
String toString( )	Повертає символьний рядок з іменем та IP-адресою вузла

Сокети TCP/IP слугують для реалізації надійних двонаправлених, постійних, двовузлових, потокових з’єднань між вузлами в IP-мережі та Інтернет. Сокет може використовуватись для підключення системи введення-виводу в Java до інших програм, які можуть знаходитись як на локальному вузлі мережі, так і на будь-якому іншому вузлі мережі.

В Java підтримуються два різновиди сокетів протоколів TCP/IP: один - для серверів, інший - для клієнтів. Клас ServerSocket призначено для серверів, він слугує "приймачем", очікує підключення клієнтів до того, як почне діяти. Клас Socket призначено для клієнтів. Він слугує для підключення до серверних сокетів ті ініціювання обміном даними по мережевому протоколу. Клієнтські сокети частіш за все застосовуються у прикладних програмах на Java. При створені об’єкту типу Socket неявно встановлюється з’єднання клієнта з сервером. Це з’єднання на виявляється ні методами, ні конструкторами.

Клас Socket має наступні конструктори для створення сокетів:

Socket (String host_name , int port) throws UnknownHostException , IOException	Створює сокет, який підключається до вказаного вузлу та порту
Socket (InetAddress IP_addr , int port) throws IOException	Створює сокет, який використовує наявний об’єкт типу InetAddress та вказаний порт

У класі Socket визначено декілька методів екземпляру. Наприклад, із об’єкту типу Socket може бути отримано відомості о зв’язаних з ним адресою та портом.

InetAddress qetInetAddress( )	Повертає об’єкт типу InetAddress, який зв’язано з об’єктом типу Socket. Якщо сокет не підключено, то повертається пусте значення null.
int getPort( )	Повертає віддалений порт, з яким зв’язано об’єкт типу Socket. Якщо сокет не прив’язано, то повертається нульове значення.
int getLocalPort( )	Повертає локальний порт, до якого прив’язано об’єкт типу Socket. Якщо сокет не прив’язано, то повертається значення -1.

Для доступу до потоків введення-виводу, які зв’язано з класом Socket, можна використати методи getlnputStream( ) та getOuptutStream( ).

InputStream getInputStream( )	Повертає об’єкт типу InetAddress, який зв’язано із

throws IOException	сокетом, що викликається
OutputStream getOutputStream( ) throws IOException	Повертає об’єкт типу OutPutStream, який зв’язано із сокетом, що викликається

Методи isConnected( ), isBound( ), isClosed( ) дозволяють визначити стан сокету та з’єднання.

Приклад 2

```
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

public class Example1 {
   public static void main(String args[]) throws Exception {
      int c;
      // Create a socket connected to internic.net, port 43.
      Socket s = new Socket("whois.internic.net", 43);
      // Obtain input and output streams.
      InputStream in = s.getInputStream();
      OutputStream out = s.getOutputStream();
      // Construct a request string.
      String str = (args.length == 0 ? "google.com" : args[0]) + "\n";
      // Convert to bytes.
      byte buf[] = str.getBytes();
      // Send request.
      out.write(buf);
      // Read and display response.
      while ((c = in.read()) != -1) {
         System.out.print((char) c);
      }
      s.close();
   }
}
```

Наступний приклад виконує запит HTTP GET до веб-серверу. У дійсності HTTP є більш складним, але можна використовувати клієнтський код для обробки найпростішого випадку: зробити запит ресурсу у сервера, сервер повертає відповідь і закриває потік. Цей випадок потребує наступних кроків:

1. Створення сокету до веб-серверу, який прослуховує порт 80.
2. Отримання PrintStream на сервер, формування та відпрравлення запиту GET PATH HTTP/1.0, де PATH — запитуваний ресурс на сервері. Наприклад, для відкриття кореню веб-сайту в якості PATH використовуємо “/”.
3. Отримання InputStream на сервері, створення для нього BufferedReader і читання рядків відповіді до досягнення її кінця.

Приклад 3

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

public class Example1 {|
   public static void main(String args[]) throws Exception {
      int c;
      String path = "/";
      // Створення з'єднання Socket до www.google.com, порт 80
      Socket s = new Socket("www.google.com", 80);
      // Створення потоків input та output для відправлення запитів на сервер
      // та читання відповіді з нього
      PrintStream out = new PrintStream( s.getOutputStream() );
      BufferedReader in = new BufferedReader( new InputStreamReader( s.getInputStream() ) );
      // Конструкція HTTP запиту складається з GET <path> HTTP/1.0
      // та наступного пустого рядка
      out.println( "GET " + path + " HTTP/1.0" );
      out.println();
      // Читання даних з серверу до досягнення кінця відповіді
      String line = in.readLine();
      while( line != null )
      {
         System.out.println( line );
         line = in.readLine();
      }
      // Закриття потоків та сокету
      in.close();
      out.close();
      s.close();
   }
}
```

Завдання

1. Вивчить теоретичний матеріал лабораторної роботи.
2. Виконайте приклад 1 лабораторної роботи. Наведіть скрин-шот та надайте пояснення отриманому результату. Визначте IP адресу веб-серверу ЗНУ. Зробіть заміну параметрів у наступних рядках

```
Address = InetAddress.getByName("www.ukr.net");
InetAddress.getAllByName("www.google.com");
byte[] addr = {(byte)192,(byte)168,(byte)1,(byte)1};
```

на параметри веб-серверу ЗНУ. Отримайте результат роботи коду та наведіть його скрин-шот. Дайте пояснення отриманому результату.
3. Виконайте приклад 2 лабораторної роботи. Наведіть скрин-шот та надайте пояснення отриманому результату. В якості аргументу командного рядка задайте www.ukr.net. Наведіть скрин-шот виконання програми та надайте пояснення отриманому результату.
4. Виконайте приклад 3 лабораторної роботи. Наведіть скрин-шот та надайте пояснення отриманому результату. У який спосіб можна інтерпретувати отриману відповідь? Послідовно замініть адресу веб-серверу у рядку

```
Socket s = new Socket("www.google.com", 80);
```

на www.znu.edu.ua та www.ukr.net.
Наведіть скрин-шот отриманих результатів та надайте пояснення отриманому результату. Як можна інтерпретувати отримані відповіді для користувача?
5. Підготуйте та надайте звіт.