جافا 2

الدرس الثامن : جافا Network Programming

أعداد: د. عبدالحميد الواعر

جامعة طرابلس

كلية تقنية المعلومات – قسم هندسة البرمجيات

الفهرس

5 جافا Network Programming

5.1 أساسيات *Networking*

5.1.1 Network Programming

المصطلح Network Programming يشير إلى كتابة برامج يمكن أن تتصل مع البرامج الاخرى من خلال شبكة الحاسوب.

5.1.2 Server

Server عبارة عن تطبيق يقدم مجموعة من الخدمات إلى Clients الذين يطلبون هذه الخدمة,

5.1.3 Client

عبارة عن نطبيق يقوم بطلب خدمة من Server .

5.1.4 Protocol

هو عبارة عن مجموعةالقواعد و الخطوات اللازمة لحصول عملية الاتصال وحدوث تبادل المعلومات. وهناك مجموعة من Protocols المستخدمة في عملية الاتصال وتبادل المعلومات ومنها :

- **Hyper Text Transfer Protocol** (HTTP)
- **File Transfer Protocol** (FTP)
- **Telnet**
- **Transport Control Protocol** (TCP)
- **User Datagram Protocol** (UDP)

5.1.5 Port number

يستخدم في مشاركة نقطة الاتصال الفيزيائي بين هذه الخدمات (أنظر الشكل) وهو عدد بطول bit-16 مما يعطي يسمح بأستخدام الترقيم من 0 – 65535 والاعداد من 0 – 1023 محجوزة للخدمات المعروفة مثل العدد 80 محجوزة لخدمة HTTP.

POP3 - port 110

IMAP - port 143

SMTP - port 25

HTTP - port 80

Secure SMTP (SSMTP) - port 465

Secure IMAP (IMAP4-SSL) - port 585

IMAP4 over SSL (IMAPS) - port 993

Secure POP3 (SSL-POP) - port 995

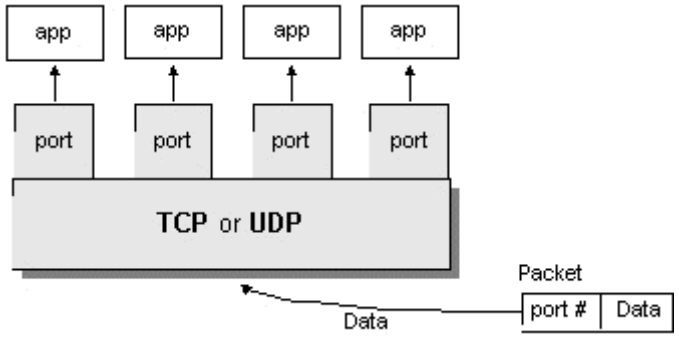
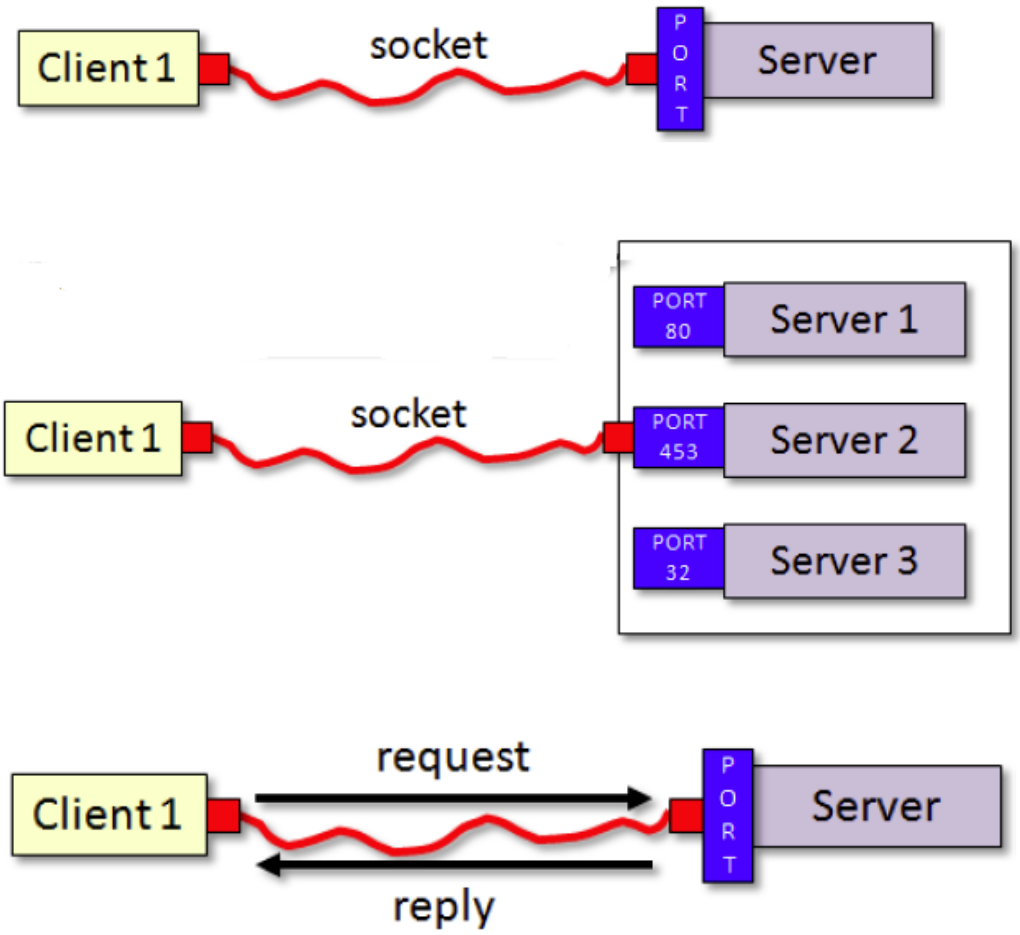


Figure 1أستخدام port

Socket 5.1.6

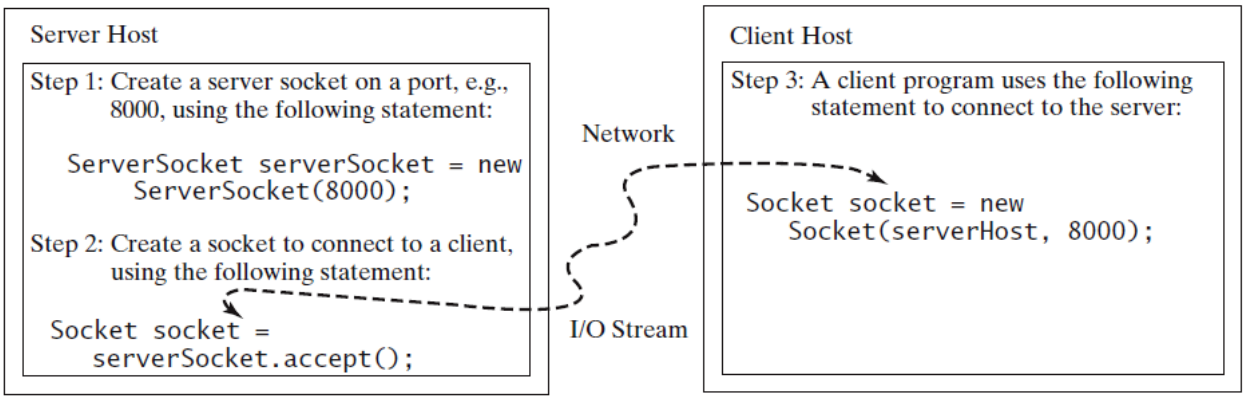
هي عبارة عن نقطة أتصال تستخدم في الربط بين برنامجين يتم تشغيلهم على شبكة الحاسوب. ويتم ربط Socket مع Port number حتى يمكن تحديد البرنامج أو الخدمة المرسلة لها البيانات.



Socket أستخدام2 Figure

5.2 *Java client/Server Programming*

تقدم لغة جافا مجموعة من classes التي تستخدم في أنشاء Socket . فمنها ServerSocket class التي تستخدم في أنشاء server socket و Socket class التي تستخدم في أنشاء client socket . أي برنامجيين يتصلان مع بعض عن طريق شبكة الحاسوب يستخدمان I/O Streams .



client/server sockets 3 **Figure**

5.2.1 أنشاء Server socket

الجملة التالية تستخدم في أنشاء Server socket

```
ServerSocket   server = new ServerSocket(port_number);
```

في حالة محاولة أنشاءِ Server socket **بأستخدام** port **قد تم أستخدامه سيتم حدوث الاستثناء**

java.net.BindException.

بعد أنشاء Server socket يستخدم Server الامر التالي لاستقبال طلبات الاتصال من clients .

```
Socket   serverSocket = server.accept();
```

5.2.2 أنشاء client socket

يقوم client **بأستخدام الامر التالي لانشاء** client socket **وعمل طلب أتصال مع** server .

```
Socket   clientSocket = new Socket(serverName,port_number);
```

مثال:

```
Socket socket = new Socket("130.254.204.33", 8000)
```

```
Socket socket = new Socket("liang.armstrong.edu", 8000);
```

5.2.3 نقل البيانات بأستخدام Sockets

عند حدوث الاتصال بين server و client يتم أستخدام I/O Strems لنقل البيانات بينهما. الشكل التالي يبين هذه العملية.

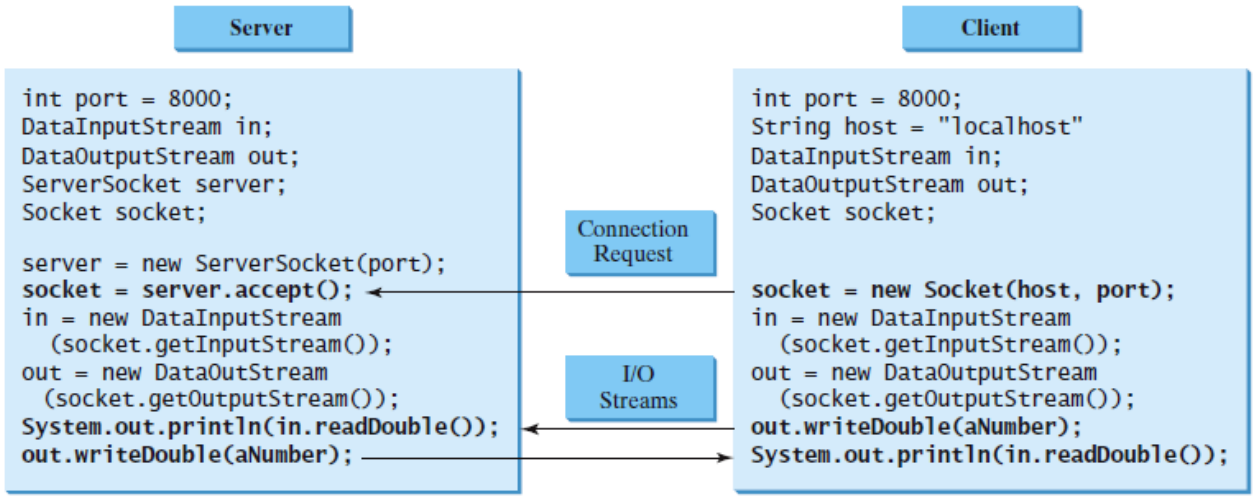


Figure 4 عملية تبادل المعلومات بأستخدام I/O streams

للحصول على input stream يتم أستخدام getInputStream() وللحصول على output stream يتم أستخدام getOutputStream() . يمكن تحويل كل من input/output streams أحد الانواع التالية من streams:

- Text-based stream
- Datatype-based stream
- Object-based stream

وذلك بأستخدام أحد الاوامر التالية:

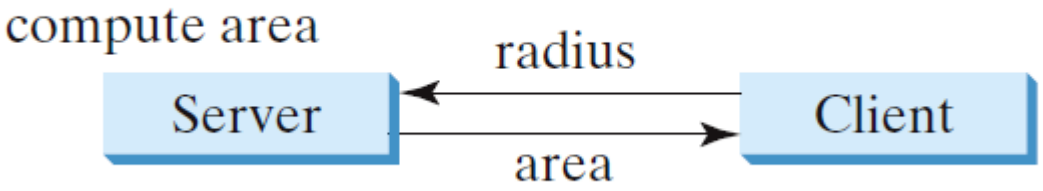
```
ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
PrintWriter out = new PrintWriter(socket.getOutputStream());

DataInputStream in = new DataInputStream(socket.getInputStream());
DataOutputStream out = new DataOutputStream(socket.getOutputStream());
```

مثال:

في هذا المثال سيتم أستخدام client/server في أنشاء client يقوم بأرسال بيانات إلى server الذي يقوم بأستخدامها في حساب المساحة وأرسال النتيجة إلى client الذي يقوم بعرض النتيجة على الشاشة.



الشكل التالي يبين كيف تتم عملية أرسال وأستقبال البيانات بأستخدام I/O streams .

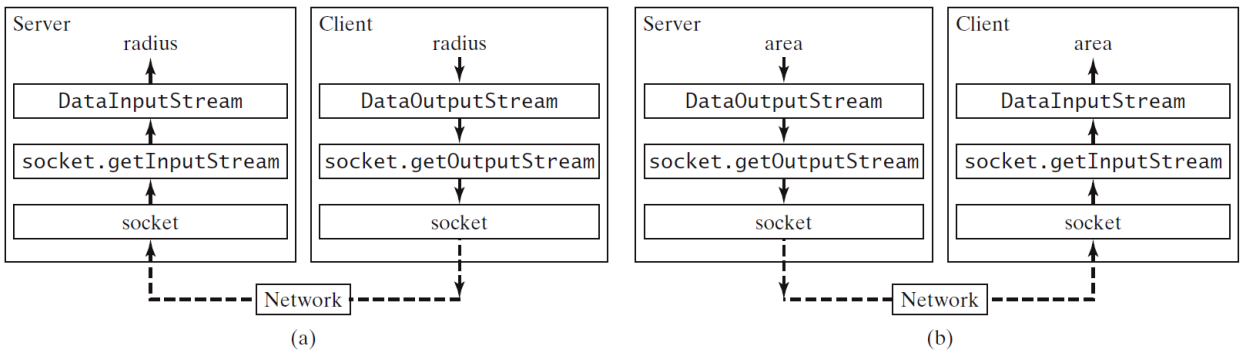


Figure 5عملية أرسال وأستقبال البيانات بأستخدام I/O streams

البرنامج التالي يقوم بأنشاء client يقوم بالاتصال مع server لحساب مساحة الدائرة التي نصف قطرها 5.

```java
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;

public class ClientProgram {

    public static void main(String[] args) {

        DataOutputStream toServer;
        DataInputStream fromServer;
        try {
            // 1- Create a socket to connect to the server
            Socket socket = new Socket("localhost", 8000);
            // 2- Create an input stream to receive data from the server
            fromServer = new DataInputStream(socket.getInputStream());

            // 3- Create an output stream to send data to the server
            toServer = new DataOutputStream(socket.getOutputStream());

            double radius = 40;

            // 4- Send the data to the server
            toServer.writeDouble(radius);
            toServer.flush();

            // 5- Get data from the server
            double area = fromServer.readDouble();

            // 6- Display data
            System.out.println("Radius is " + radius + "\n");
            System.out.println("Area received from the server is "+area+'\n');

            // 7- Close Socket
            socket.close();
        } catch (IOException ex) {
            System.err.println(ex.toString() + '\n');
        }
    }
}
```

البرنامج التالي يقوم بأنشاء server يقوم بأستقبال قيمة تصف القطر ثم حساب مساحة الدائرة وأرسال النتيجة إلى client.

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;

public class ServerProgram {

    public static void main(String[] args) {
        try {
            // 1- Create a server socket
            ServerSocket serverSocket = new ServerSocket(8000);
            System.out.println("Server started at " + new Date() + '\n');

            while (true) {
            // 2- Listen for a connection request
            Socket socket = serverSocket.accept();

            // 3- Create data input  streams
            DataInputStream inputFromClient = new DataInputStream(
                                                    socket.getInputStream());
            //4- Create data input and output streams
            DataOutputStream outputToClient = new DataOutputStream(
                                                     socket.getOutputStream());
            //5- Listen to clients requests

              // 6- Receive data  from the client
              double radius = inputFromClient.readDouble();
              System.out.println("Radius received from client: "+radius+ '\n');

                double area = radius * radius * Math.PI;
                System.out.println("Area found: " + area + '\n');
                // 6- Send data to the client
                outputToClient.writeDouble(area);

            }
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }

}
```

5.2.4 التعامل مع عدة clients

Server يمكن أن يتعامل مع عدة clients في نفس الوقت وذلك بأستخدام Threads حيث يقوم بخدمة كل client بأستخدام Thread خاص به.

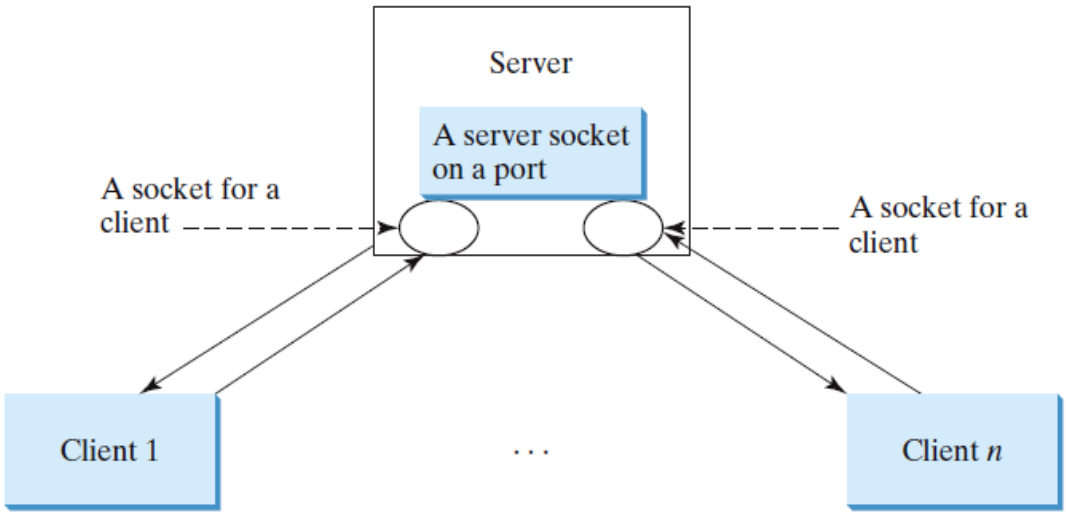


Figure 6التعامل مع عدة clients

البرنامج التالي يبين كيف تم تعديل برنامج ServerProgram للتعامل مع عدة clients.

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;

public class MultiServer {

    public static void main(String[] args) {

        try {
            ServerSocket serverSocket = new ServerSocket(8000);
            System.out.println("Server started at " + new Date() + '\n');

            int clientNo = 1;
            while (true) {

                Socket socket = serverSocket.accept();

                // Display the client number
                System.out.println("Starting thread for client " + clientNo
                        + " at " + new Date() + '\n');

                HandleAClient  clientHandler = new HandleAClient(socket);

                // Start the new thread for the comming request
                clientHandler.start();

                // Increment clientNo
                clientNo++;
            }
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }

}
```

الجزء التالي من البرنامج عبارة class بأستخدام Thread تستدعى كل مرة للتعامل مع client الجديد.

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;

public class HandleAClient extends Thread {

    private Socket socket;

    public HandleAClient(Socket socket) {
        this.socket = socket;
    }

    public void run() {
        try {
            // Create data input and output streams
            DataInputStream inputFromClient = new DataInputStream(
                    socket.getInputStream());
            DataOutputStream outputToClient = new DataOutputStream(
                    socket.getOutputStream());
            double radius = inputFromClient.readDouble();
            double area = radius * radius * Math.PI;
            outputToClient.writeDouble(area);
            System.out.println("radius received from client: "
                    + radius + '\n');
            System.out.println("Area found: " + area + '\n');
        } catch (IOException e) {
            System.err.println(e);
        }
    }

}
```