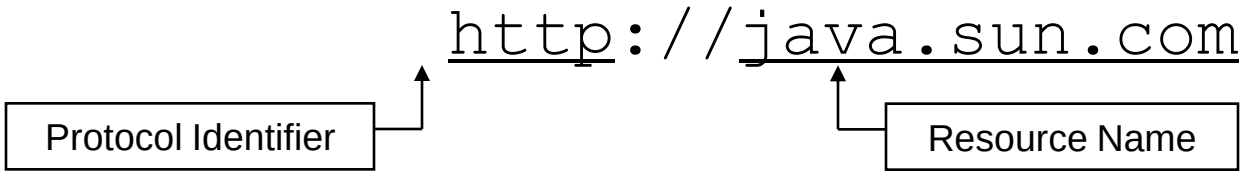
Network programming

By Pakita Shamoi, fall 2013

Manipulating URLs

- URL is an acronym for *Uniform Resource Locator* and is a reference (an address) to a resource on the Internet.
- Sample structure of a URL. The resource name part may contain: host name, file name, port number(optional) and reference (optional)



- So, URL is a description of a resource location on the Internet. Java provides a class—`java.net.URL`—to manipulate URLs.
- The URL class provides several methods implemented on URL objects. You can get the protocol, host name, port number, and filename from a URL.

Example code

```
import java.net.*;
import java.io.*;
public class ParseURL {
   public static void main(String[] args) throws Exception
{
      URL aURL = new
URL("http://java.sun.com:80/docs/books/" +
"tutorial/index.html#DOWNLOADING");
      System.out.println("protocol = " +
         aURL.getProtocol()); System.out.println("host = "
+
         aURL.getHost()); System.out.println("filename = "
+
         aURL.getFile()); System.out.println("port = " +
         aURL.getPort()); System.out.println("ref = " +
         aURL.getRef());
   }
}
```

Output of the above code:

protocol = http
host = java.sun.com
filename = /docs/books/tutorial/index.html
port = 80
ref = DOWNLOADING

Connecting with a URL (1)

- **`openStream()`:** returns a `java.io.InputStream` object, from which you can read easily as reading from an input stream. It may throw an IOException

Example code

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

public class ReadURL {
   public static void main(String[] args) throws Exception
  {
     URL osu = new URL("http://www.osu.edu/");
  BufferedReader in = new BufferedReader(
                  new InputStreamReader(osu.openStream()));
  String inputLine;
  while ((inputLine = in.readLine()) != null)
      System.out.println(inputLine);
  in.close();
    }
}
```

This prints out the source code for the webpage www.osu.edu

Connecting with a URL (2)

- **`openConnection()`:** Returns a `URLConnection` object that represents a connection to the remote object referred to by the URL. It may throws an IOException

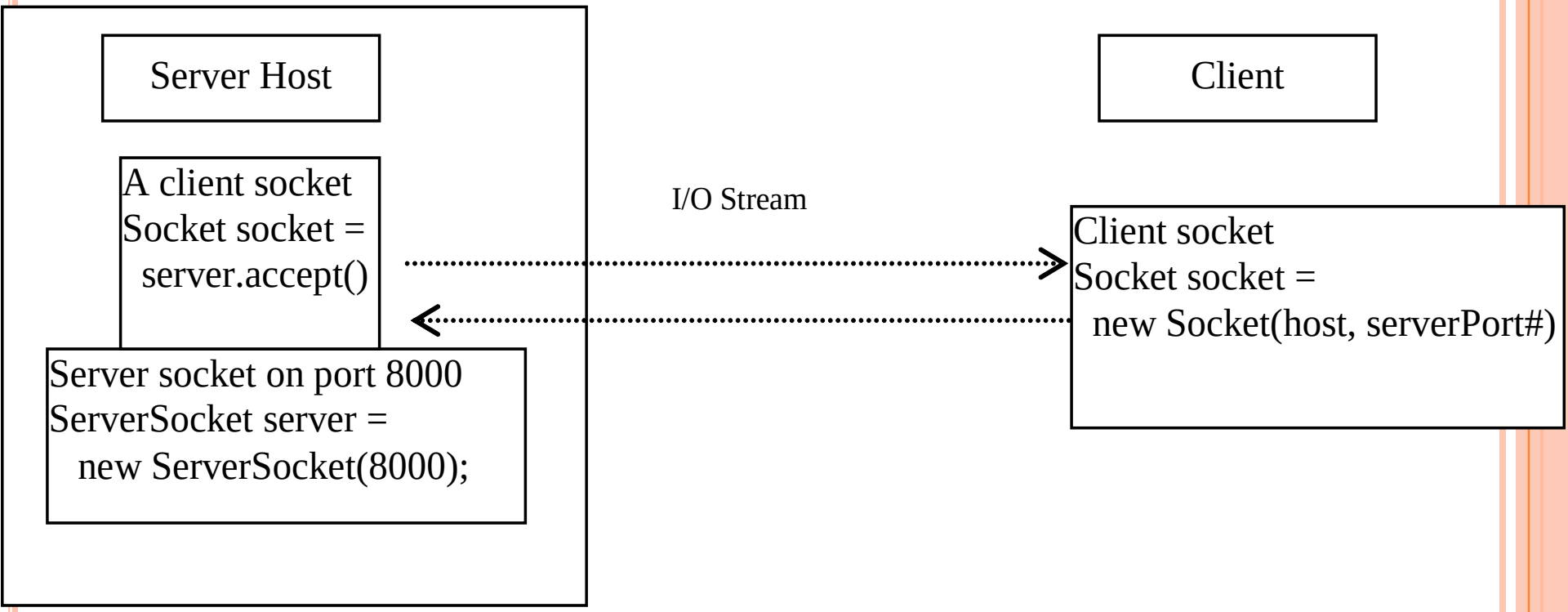
```
try {
    URL osu = new URL("http://www.osu.edu/");
    URLConnection osuConnection = osu.openConnection();
} catch (MalformedURLException e) { // new URL()
failed
    . . .
} catch (IOException e) {
    . . .
}
```

- The `URLConnection` class provides many methods to communicate with the URL, such as reading and writing.

Sockets

- Sometimes you need a low-level network communication, such as a client-server application
- The TCP protocol provides a reliable point-to-point communication channel via the **sockets.**
- A socket is an endpoint for reliable communication between two machines. To connect with each other, each of the *client and the server binds a socket to its end for reading and writing.*
- The **`java.net`** package provides two classes — **`Socket`** and **`ServerSocket`** — to implement the client and the server, respectively.

Client/Server Communications



Establishing a simple server

- Five steps:

1). Create a **`ServerSocket`** object

```
ServerSocket server=new
  ServerSocket(port,queueLength);
```

2). The server listens indefinitely (or blocks) for an attempt by a client to connect

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

3). Get the `OutputStream` and `InputStream` objects that enable the server to communicate with the client by sending and receiving bytes

```
InputStream input = connection.getInputStream();
OutputStream output = connection.getOutputStream();
```

4). ***Processing phase***: the server and the client communicate via the **`InputStream`** and the **`OutputStream`** objects

5). After the communication completes, the server closes the connection by invoking `close()` on the `Socket` and the corresponding streams

Establishing a simple client

- Four steps:
  1). Create a **`Socket`** object

```
Socket connection = new Socket (serverAddr, port);
```

  2). Get the **`OutputStream`** and **`InputStream`** of the Socket. The server and the client must send and receive the data in the same format

  3). ***Processing phase*:** the server and the client communicate via the **`InputStream`** and the **`OutputStream`** objects

  4). After the communication completes, the client closes the connection.

A SIMPLE SERVER/CLIENT PAIR EXAMPLE

The server side

```
import java.io.*;
import java.net.*;
class Server {
   public static void main(String args[]) {
      String data = "Let's test if we can connect...";
      try {
         ServerSocket server_socket = new ServerSocket(1234);
         System.out.println("I’ve started, dear clients...");

         Socket socket = server_socket.accept();

         System.out.print("Server has connected!\n");
         PrintWriter outToClient = new
  PrintWriter(socket.getOutputStream(), true);

         System.out.print("Sending string: ‘" + data + "’\n");
         outToClient.print(data);

         outToClient.close();
         socket.close();
         server_socket.close();
      }
      catch(Exception e) {
         System.out.print("Whoops! It didn't work!\n");
      }
   }}
```

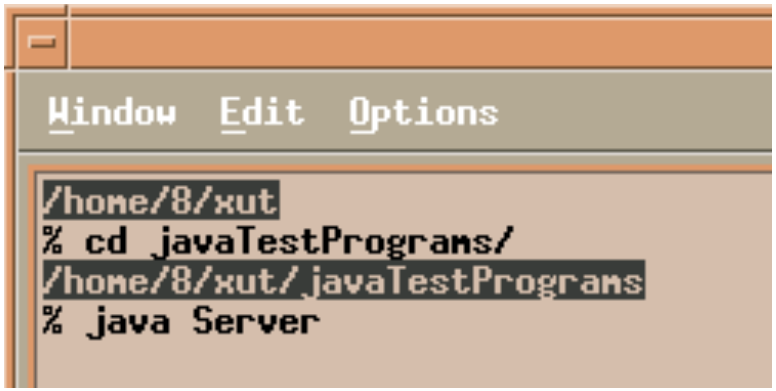
A SIMPLE SERVER/CLIENT PAIR EXAMPLE (CONT.)

The client side

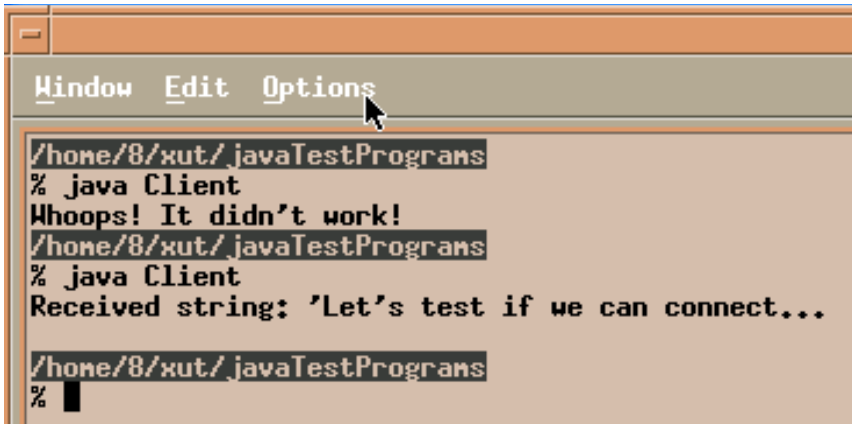
```
import java.io.*;
import java.net.*;

class Client {
   public static void main(String args[]) {
      try {
          Socket socket = new Socket("localhost", 1234);

          BufferedReader inFromServer = new BufferedReader(new
             InputStreamReader(socket.getInputStream()));

         System.out.print("Received string: '");
         while (inFromServer.ready())
             System.out.println(in.readLine()); //Read one line and
  output it

          inFromServer.close();
      }
      catch(Exception e) {
          System.out.print("Whoops! It didn't work!\n");
       }
   }}
```

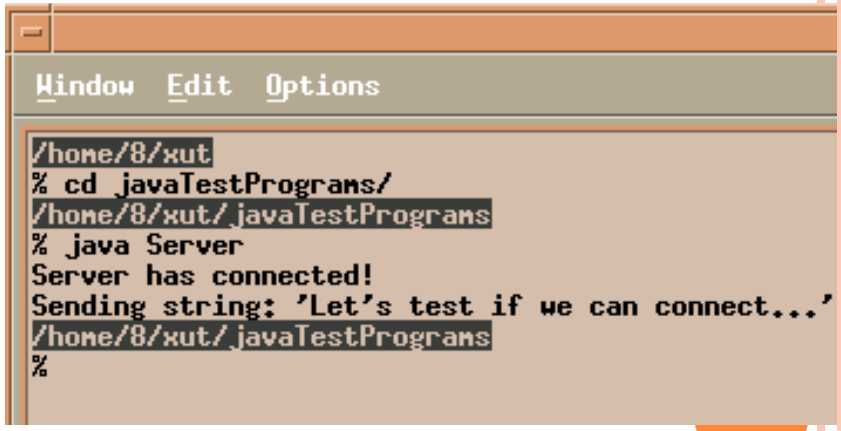
A SIMPLE SERVER/CLIENT PAIR EXAMPLE (CONT.)

- What happens on the screen if you run the code?
  - First run `Server.java`



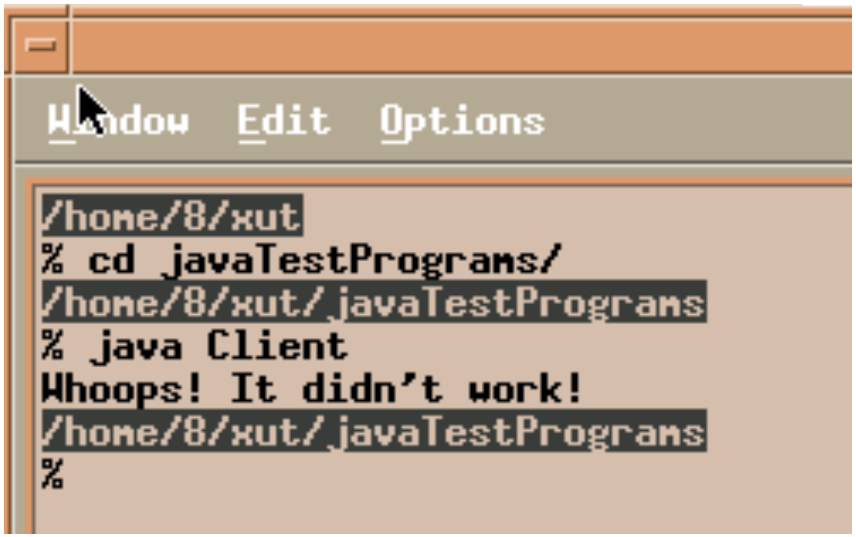
  - Then run `Client.java`





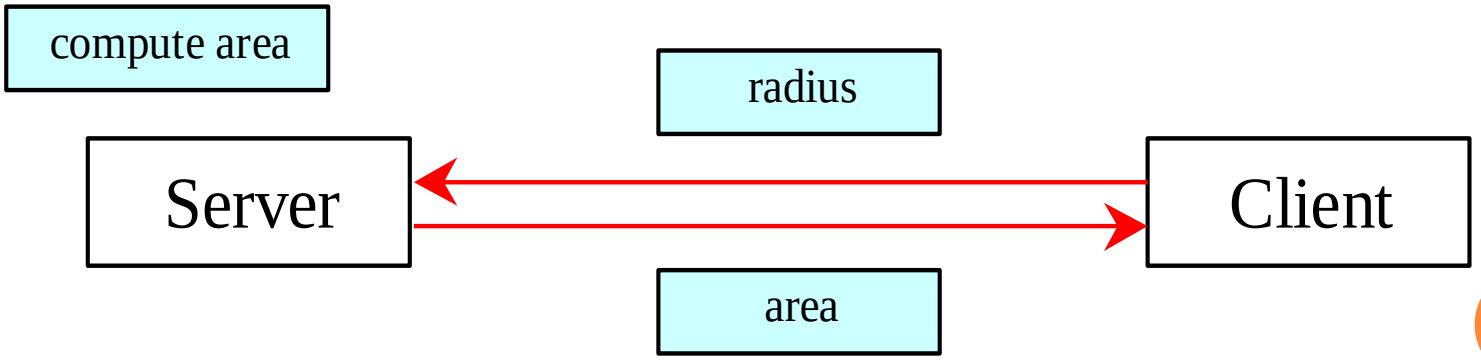
A simple server/client pair example (cont.)

- If you run `Client.java` without running `Server.java`



Example

- Objective: Write a client to send data to a server. The server receives the data, uses it to produce a result, and then sends the result back to the client. The client displays the result on the console. In this example, the data sent from the client is the radius of a circle, and the result produced by the server is the area of the circle.



Server part

```
System.out.println("-----Calculate Area service-----");
ServerSocket server = null;
Socket client = null;
try {
    server = new ServerSocket(1234);//1234 is an unused port number
} catch (IOException ie) {
    System.out.println("Cannot open socket.");System.exit(1);
}
while(true) {
    try {
        client = server.accept();
        OutputStream clientOut =client.getOutputStream();//Returns:an output stream for writing bytes
        PrintWriter pw = new PrintWriter(clientOut, true);

        InputStream clientIn =client.getInputStream();//Returns an input stream for reading bytes fro
        BufferedReader br = new BufferedReader(new InputStreamReader(clientIn));
        int r = Integer.parseInt(br.readLine());

        double answer = Math.PI*r*r;
        pw.println("Answer is: "+answer);
        System.out.println("1 client connected and requested the area of a circle with radius "+r+". A
    } catch (IOException ie) {}
    finally {
        client.close();
    }
}
```

```java
try {
    Socket client =new Socket(InetAddress.getLocalHost(),1234);
    //Socket client =new Socket("178.90.65.165",1234);
    InputStream clientIn =client.getInputStream();
    BufferedReader br = new BufferedReader(new InputStreamReader(clientIn));

    OutputStream clientOut =client.getOutputStream();
    PrintWriter pw = new PrintWriter(clientOut,true);

    BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter radius, please: ");
    pw.println(stdIn.readLine());
    System.out.println("Server message: "+ br.readLine());

    pw.close();
    br.close();
    client.close();
} catch (ConnectException ce) {
    System.out.println("Cannot connect to the server.");
} catch (IOException ie) {
    System.out.println("I/O Error.");
}
finally {
    System.out.println("Bye!");
}
```

CLIENT PART

Example Outputs

Problems | Console | @ Javadoc | Problem Statement | Console

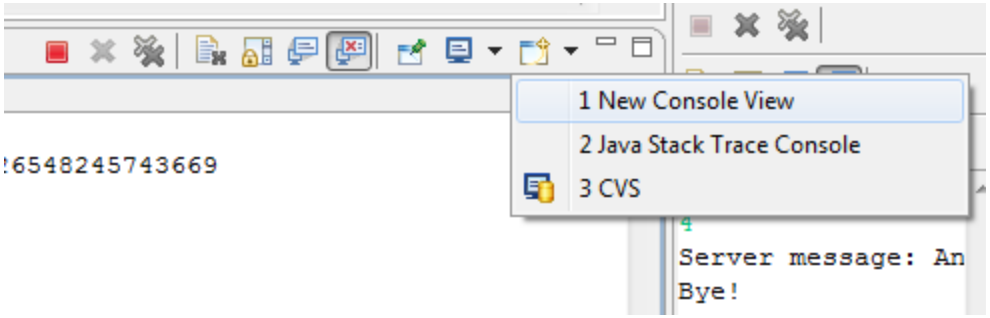
```
<terminated> MyServer [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (25.11.2013 4:57:10)
-----Calculate Area service-----
1 client connected and requested the area of a circle with radius 7. Answer is 153.93804002589985
```

Problems | Console | @ Javadoc | Problem Statement | Console

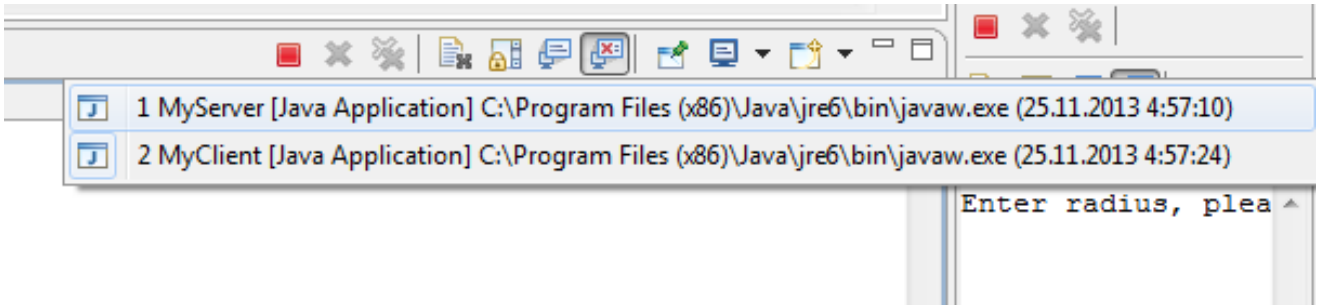
```
<terminated> MyClient [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (25.11.2013 4:57:24)
Enter radius, please:
7
Server message: Answer is: 153.93804002589985
Bye!
```

Launching client and server in Eclipse

- *Launch server*
- *Open the new console for the client*



- *If both consoles display client application, on one of them switch to server application:*



A BIT MORE COMPLEX EXAMPLE (INTRANET)

- Student.java

```
public class Student implements Serializable{
    String name;
    String id;
    double gpa;
    public Student(String name, String id, double gpa) {
        super();
        this.name = name;
        this.id = id;
        this.gpa = gpa;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public double getGpa() {
        return gpa;
    }
    public void setGpa(double gpa) {
        this.gpa = gpa;
    }
}
```

INTRANET SERVER

```
ServerSocket server = null;
Socket client = null;
boolean ok = true;
try {
    server = new ServerSocket(1234);
} catch (IOException ie) {
    System.out.println("Cannot open socket."); System.exit(1);
}
System.out.println("---Intranet Server at kbtu.kz---");
students = deserialize();
while(ok) {
    try {
        client = server.accept();
        OutputStream clientOut =client.getOutputStream();//Returns:an output stream for writing bytes t
        ObjectOutputStream  oos = new ObjectOutputStream(clientOut);

        InputStream clientIn =client.getInputStream();//Returns an input stream for reading bytes from
        BufferedReader br = new BufferedReader(new InputStreamReader(clientIn));

        oos.writeObject(students);

        String ans[] = (br.readLine()).split(" ");
        int index  = Integer.parseInt(ans[0]); double mark = Integer.parseInt(ans[1]);
        students.get(index).setGpa(mark);
        oos.close();
        System.out.println("1 teacher connected and put mark "+mark+" to  "+students.get(index).name);
    } catch (IOException ie) {ie.printStackTrace();}
}
serialize();
client.close();
```

Intranet Server (Serialization/Deserialization)

```
static Vector<Student> students = null;
static Vector<Student> deserialize() throws IOException, ClassNotFoundException{
    FileInputStream fis = new FileInputStream("students.out");
    ObjectInputStream oin = new ObjectInputStream(fis);
    Vector<Student> b = (Vector<Student>) oin.readObject(); fis.close(); oin.close();
    return b;
}
static void serialize() throws IOException, ClassNotFoundException {
    FileOutputStream fos = new FileOutputStream("students.out");
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(students);oos.flush();oos.close();fos.close();
}
```

```java
try {
    Socket client =new Socket(InetAddress.getLocalHost(),1234);
    InputStream clientIn =client.getInputStream();
    ObjectInputStream ois = new ObjectInputStream(clientIn);

    OutputStream clientOut =client.getOutputStream();
    PrintWriter pw = new PrintWriter(clientOut,true);

    Vector<Student> s = (Vector<Student>)ois.readObject();
    for(int i=0; i<s.size(); i++) System.out.println(i+ ")"+s.get(i).name+" "+s.get(i).gpa);

    BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter student index and mark ");
    pw.println(stdIn.readLine());

    System.out.println("Updated successfully!");
    pw.close();
    ois.close();
    client.close();
} catch (ConnectException ce) {
    System.out.println("Cannot connect to the server.");
} catch (IOException ie) {
    System.out.println("I/O Error.");
}
finally {
    System.out.println("Bye!");
}
```

CLIENT TEACHER

Outputs

Problems | Console | Javadoc | Problem Statement | Console

IntranetServer [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (25.11.2013 5:22:35)

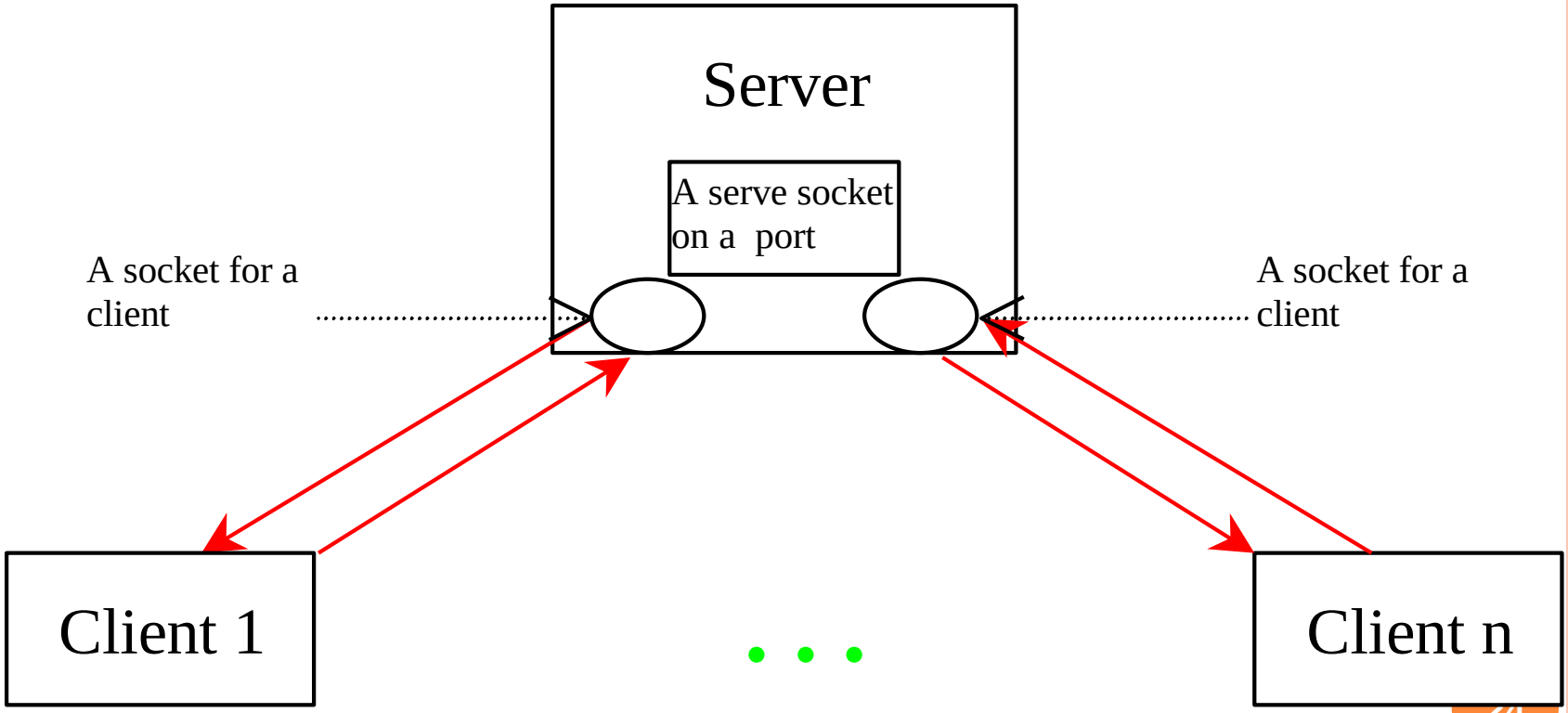
```
---Intranet Server at kbtu.kz---
1 teacher connected and put mark 73.0 to  Kanat
```

Problems | Console | Javadoc | Problem Statement | Consol

<terminated> ClientTeacher [Java Application] C:\Program Files (x86)\Java\jre6\bin\

```
0)Asel 84.0
1)Kanat 77.0
2)Ashat 40.0
Enter student index and mark
1 73
Updated successfully!
Bye!
```

Serving Multiple Clients



Supplemental reading

- *Custom networking*

  http://java.sun.com/docs/books/tutorial/networking/index.html

- *JavaTM Programming Language Basics, Socket Communications*

  http://developer.java.sun.com/developer/onlineTraining/Programming/BasicJava2/socket.html