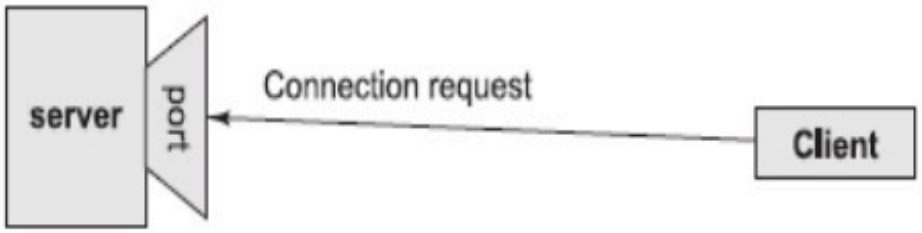
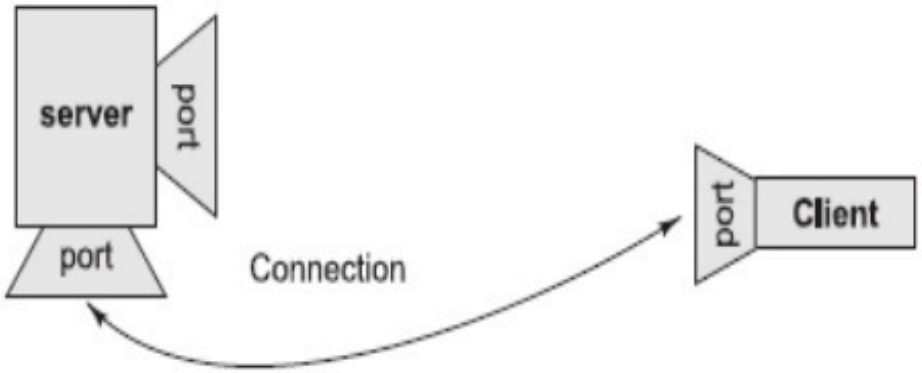
Networking

Networking

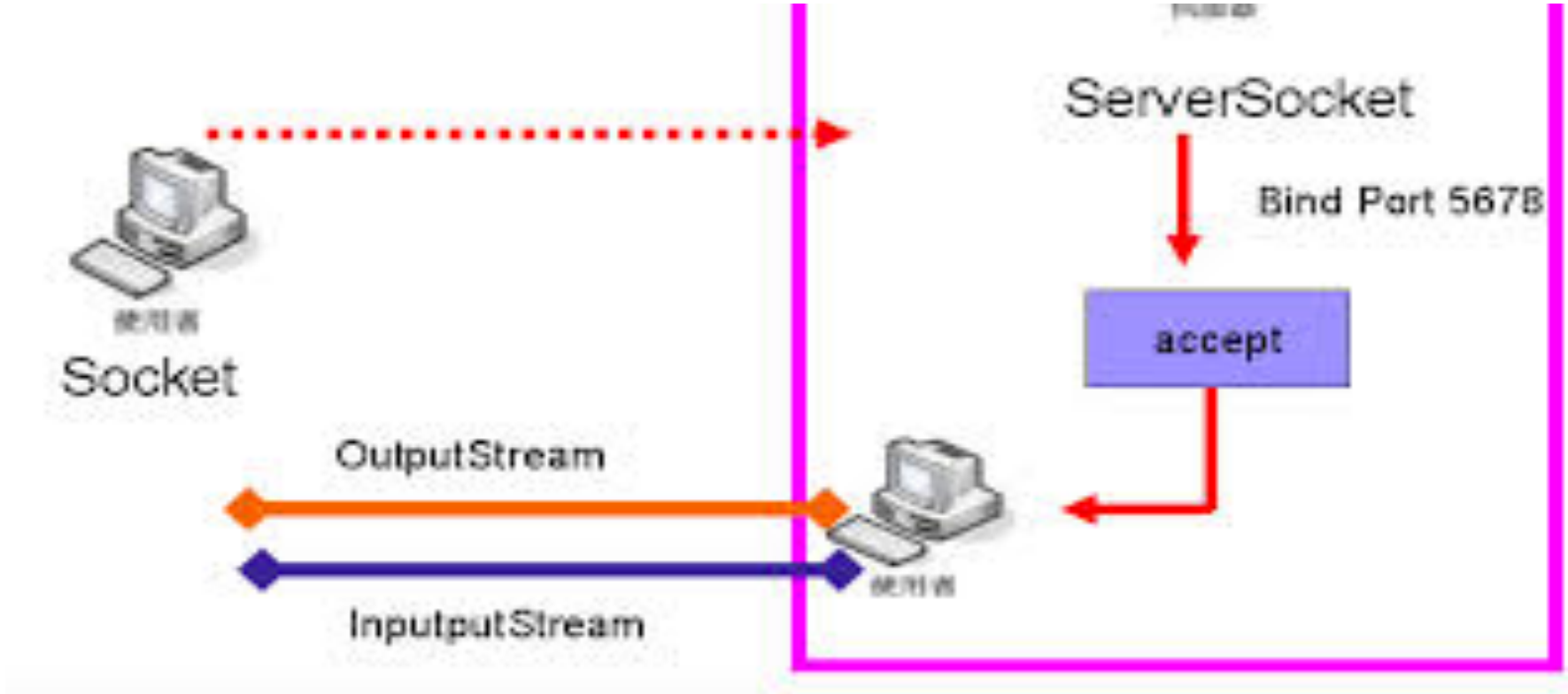


[a]: a client making a connection request to the server



[b]: session established with temporary ports used for two way communication.

java.net.ServerSocket



accept() - Listens for a connection to be made to this socket and accepts it.

java.net.ServerSocket

`ServerSocket``()`
Creates an unbound server socket.

`ServerSocket``(int port)`
Creates a server socket, bound to the specified port.

`ServerSocket``(int port, int backlog)`
Creates a server socket and binds it to the specified local port number, with the specified backlog.

`ServerSocket``(int port, int backlog,` **`InetAddress`** `bindAddr)`
Create a server with the specified port, listen backlog, and local IP address to bind to.

java.net.ServerSocket

Socket	**accept**() Listens for a connection to be made to this socket and accepts it.
void	**bind**(**SocketAddress** endpoint) Binds the ServerSocket to a specific address (IP address and port number).
void	**bind**(**SocketAddress** endpoint, int backlog) Binds the ServerSocket to a specific address (IP address and port number).
void	**close**() Closes this socket.
ServerSocketChannel	**getChannel**() Returns the unique **ServerSocketChannel** object associated with this socket, if any.
InetAddress	**getInetAddress**() Returns the local address of this server socket.
int	**getLocalPort**() Returns the port number on which this socket is listening.
SocketAddress	**getLocalSocketAddress**() Returns the address of the endpoint this socket is bound to, or null if it is not bound yet.

Socket

Constructors

Modifier	Constructor and Description
	Socket() Creates an unconnected socket, with the system-default type of SocketImpl.
	Socket(**InetAddress** address, int port) Creates a stream socket and connects it to the specified port number at the specified IP address.
	Socket(**InetAddress** host, int port, boolean stream) **Deprecated.** *Use DatagramSocket instead for UDP transport.*
	Socket(**InetAddress** address, int port, **InetAddress** localAddr, int localPort) Creates a socket and connects it to the specified remote address on the specified remote port.
	Socket(**Proxy** proxy) Creates an unconnected socket, specifying the type of proxy, if any, that should be used regardless of any other settings.
protected	**Socket**(**SocketImpl** impl) Creates an unconnected Socket with a user-specified SocketImpl.
	Socket(**String** host, int port) Creates a stream socket and connects it to the specified port number on the named host.
	Socket(**String** host, int port, boolean stream) **Deprecated.** *Use DatagramSocket instead for UDP transport.*
	Socket(**String** host, int port, **InetAddress** localAddr, int localPort) Creates a socket and connects it to the specified remote host on the specified remote port.

Socket

Methods

Modifier and Type	Method and Description
void	**bind**(**SocketAddress** bindpoint) Binds the socket to a local address.
void	**close**() Closes this socket.
void	**connect**(**SocketAddress** endpoint) Connects this socket to the server.
void	**connect**(**SocketAddress** endpoint, int timeout) Connects this socket to the server with a specified timeout value.
SocketChannel	**getChannel**() Returns the unique **SocketChannel** object associated with this socket, if any.
InetAddress	**getInetAddress**() Returns the address to which the socket is connected.
InputStream	**getInputStream**() Returns an input stream for this socket.
boolean	**getKeepAlive**() Tests if SO_KEEPALIVE is enabled.
InetAddress	**getLocalAddress**() Gets the local address to which the socket is bound.
int	**getLocalPort**() Returns the local port number to which this socket is bound.
SocketAddress	**getLocalSocketAddress**() Returns the address of the endpoint this socket is bound to, or `null` if it is not bound yet.
boolean	**getOOBInline**() Tests if OOBINLINE is enabled.
OutputStream	**getOutputStream**() Returns an output stream for this socket.
int	**getPort**()

InputStream

Methods

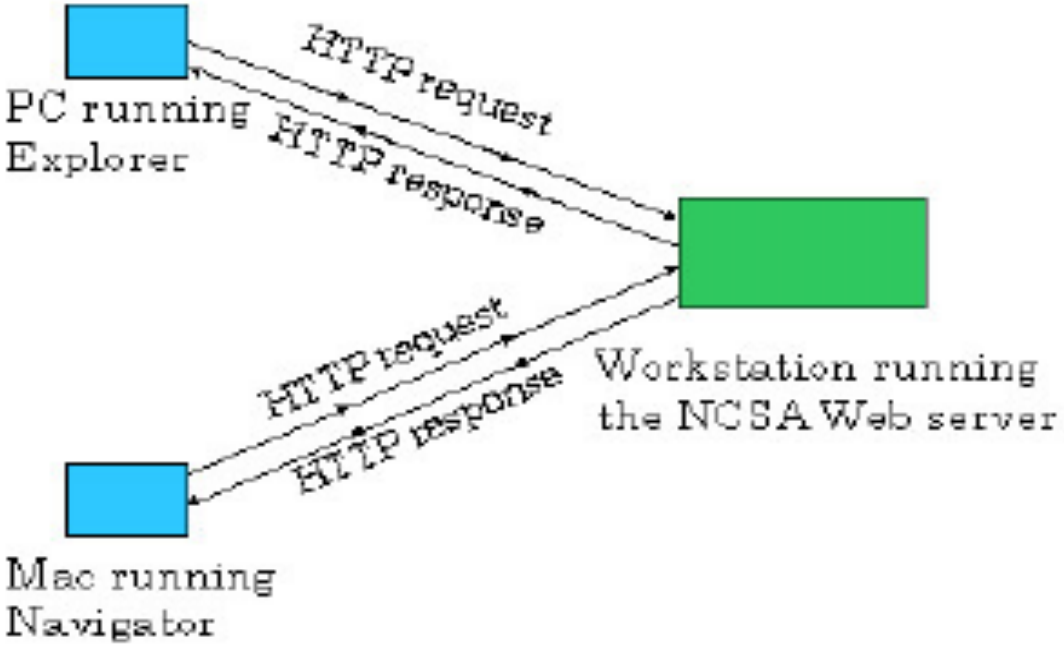
Modifier and Type	Method and Description
`int`	**`available`**`()` Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
`void`	**`close`**`()` Closes this input stream and releases any system resources associated with the stream.
`void`	**`mark`**`(int readlimit)` Marks the current position in this input stream.
`boolean`	**`markSupported`**`()` Tests if this input stream supports the `mark` and `reset` methods.
`abstract int`	**`read`**`()` Reads the next byte of data from the input stream.
`int`	**`read`**`(byte[] b)` Reads some number of bytes from the input stream and stores them into the buffer array `b`.
`int`	**`read`**`(byte[] b, int off, int len)` Reads up to `len` bytes of data from the input stream into an array of bytes.
`void`	**`reset`**`()` Repositions this stream to the position at the time the `mark` method was last called on this input stream.
`long`	**`skip`**`(long n)` Skips over and discards `n` bytes of data from this input stream.

OutputStream

Methods

Modifier and Type	Method and Description
`void`	**`close`**`()` Closes this output stream and releases any system resources associated with this stream.
`void`	**`flush`**`()` Flushes this output stream and forces any buffered output bytes to be written out.
`void`	**`write`**`(byte[] b)` Writes `b.length` bytes from the specified byte array to this output stream.
`void`	**`write`**`(byte[] b, int off, int len)` Writes `len` bytes from the specified byte array starting at offset `off` to this output stream.
`abstract void`	**`write`**`(int b)` Writes the specified byte to this output stream.

Web


HTTP request
HTTP response
PC running
Explorer
HTTP request
HTTP response
Workstation running
the NCSA Web server
Mac running
Navigator

HTTP

HTTP requests

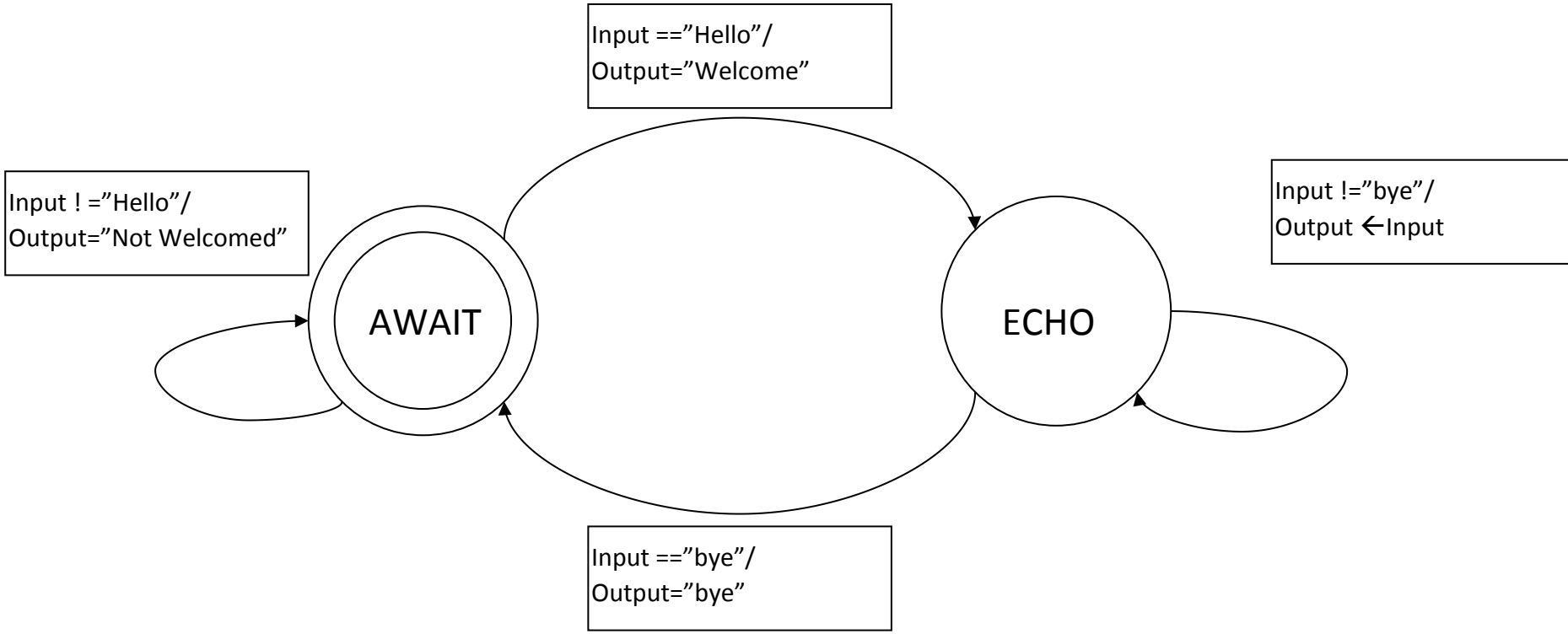
```
GET /lessons/index.htm HTTP/1.1
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language:gr

(extra carriage return, line feed)
```

HTTP respond

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html
data data data data data ...
```

GET / HTTP/1.0\n\n

Echo Server


Input ==”Hello”/
Output=”Welcome”
Input ! =”Hello”/
Output=”Not Welcomed”
AWAIT
ECHO
Input !=”bye”/
Output ←Input
Input ==”bye”/
Output=”bye”

```
 BufferedReader instream = new BufferedReader (new InputStreamReader
(csocket.getInputStream()));
 BufferedWriter outstream = new BufferedWriter(new
OutputStreamWriter(csocket.getOutputStream()));

          //String strin = instream.readLine();

          String strin ;
   while ((strin = instream.readLine()) != null) {
          if (strin.equals("Hi")){
                    outstream.write("Hi");

          }else{

           System.out.println(strin);
          }

    if (strin.equals("Bye."))
      break;
   }
```

Multi Thread

```
public static void main(String args[])
                    throws Exception {
                        ServerSocket ssock = new ServerSocket(8080);
                        System.out.println("Listening");
                        while (true) {
                            Socket sock = ssock.accept();
                            System.out.println("Connected");
                            new Thread(new MultiThreadServer(sock)).start();
                    }
                   }
```

Runnable

```
public class MultiThreadServer implements Runnable {
    Socket csocket;
    MultiThreadServer(Socket csocket) {
        this.csocket = csocket;
 }
```

Run()

```
public void run() {
      try{


    BufferedReader instream = new BufferedReader (new InputStreamReader
(csocket.getInputStream()));
           BufferedWriter outstream = new BufferedWriter(new
OutputStreamWriter(csocket.getOutputStream()));

           //String strin = instream.readLine();
           String strin;
    while ((strin = instream.readLine()) != null) {


       // csocket.close();
  }
  catch (IOException e) {
         System.out.println(e);
  }
 }
```