Indian Institute of Technology Kharagpur

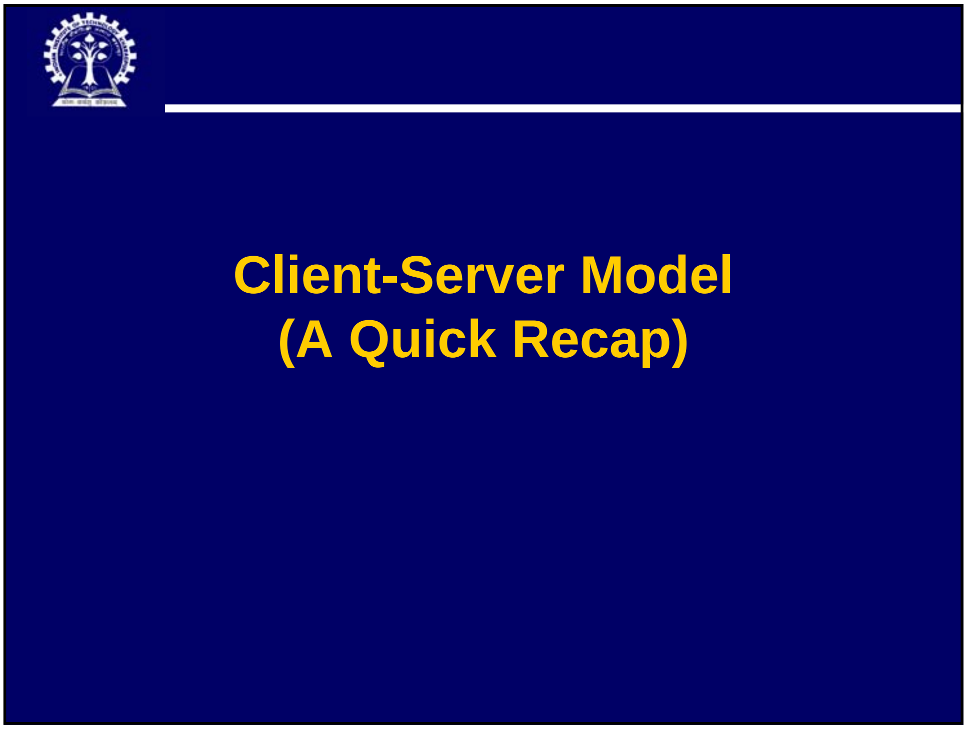
Client-Server Programming in Java

Prof. Indranil Sen Gupta
Dept. of Computer Science & Engg.
I.I.T. Kharagpur, INDIA

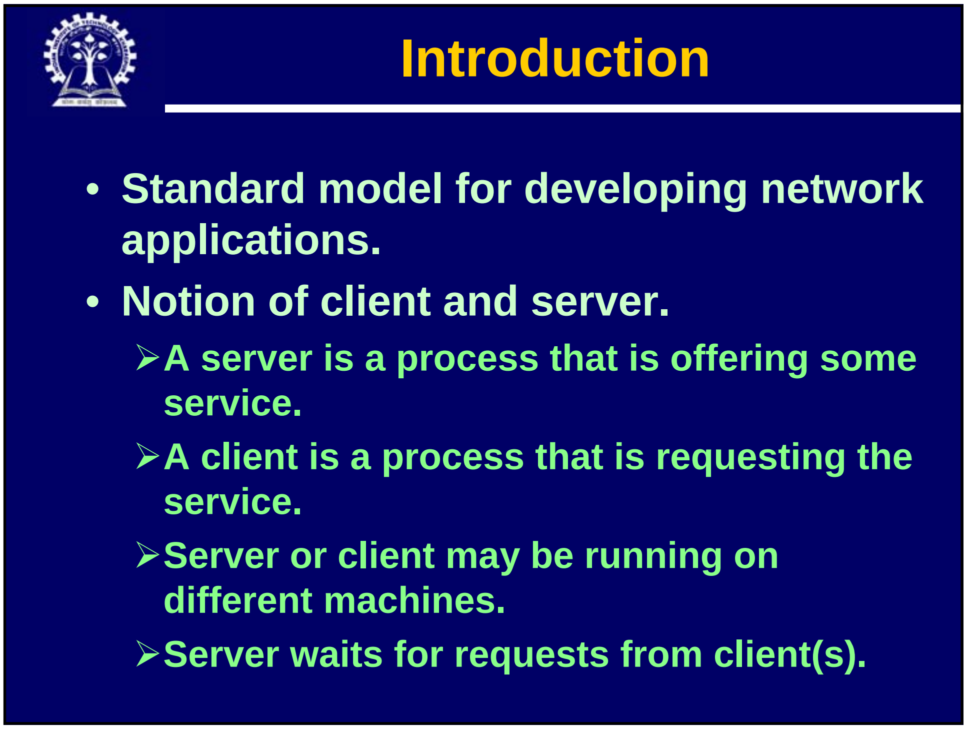
Lecture 30: Client-server programming in Java

On completion, the student will be able to:

1. Explain the process of writing client-server applications in java.
2. Illustrate the writing of connection-oriented server, and the corresponding client, in java.
3. Illustrate the writing of connection-less server, and the corresponding client, in java.



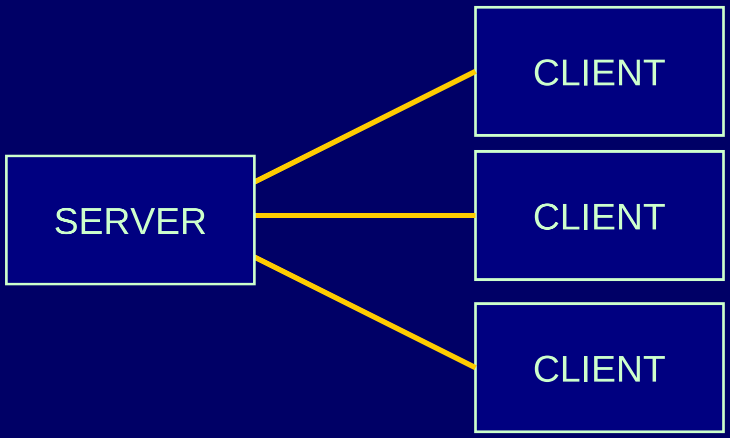
Client-Server Model
(A Quick Recap)



Introduction

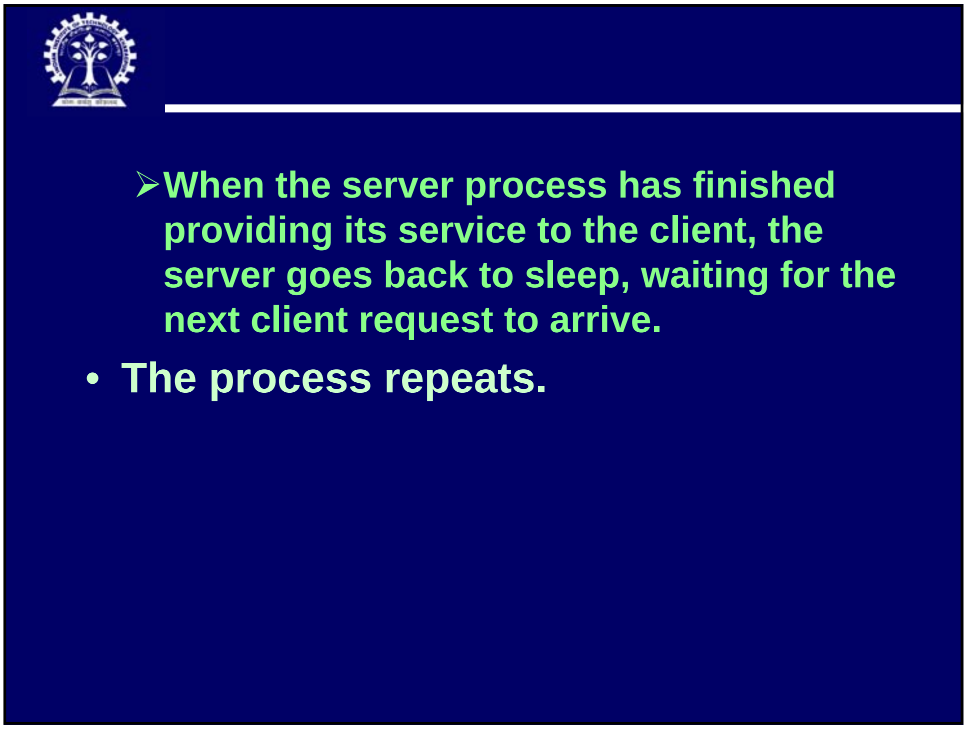
- Standard model for developing network applications.
- Notion of client and server.
  - ➢A server is a process that is offering some service.
  - ➢A client is a process that is requesting the service.
  - ➢Server or client may be running on different machines.
  - ➢Server waits for requests from client(s).

Client-server Model (contd.)

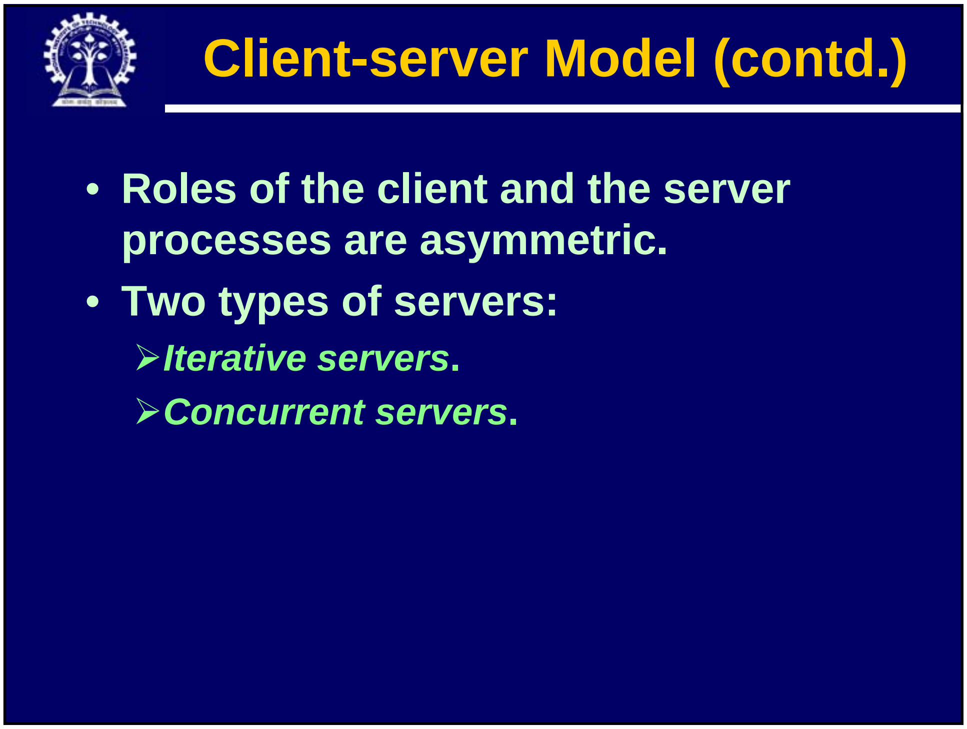


Client-server Model (contd.)

- Typical scenario:
  - ➢The server process starts on some computer system.
    - ▪ Initializes itself, then goes to sleep waiting for a client request.
  - ➢A client process starts, either on the same system or on some other system.
    - ▪ Sends a request to the server.

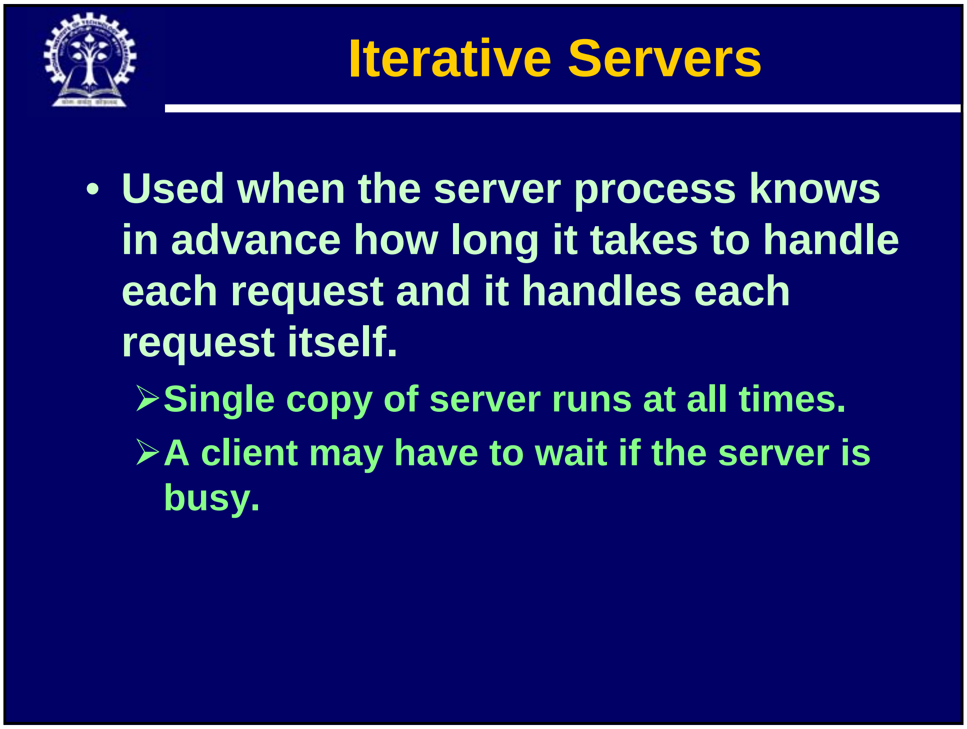


  - When the server process has finished providing its service to the client, the server goes back to sleep, waiting for the next client request to arrive.
- The process repeats.



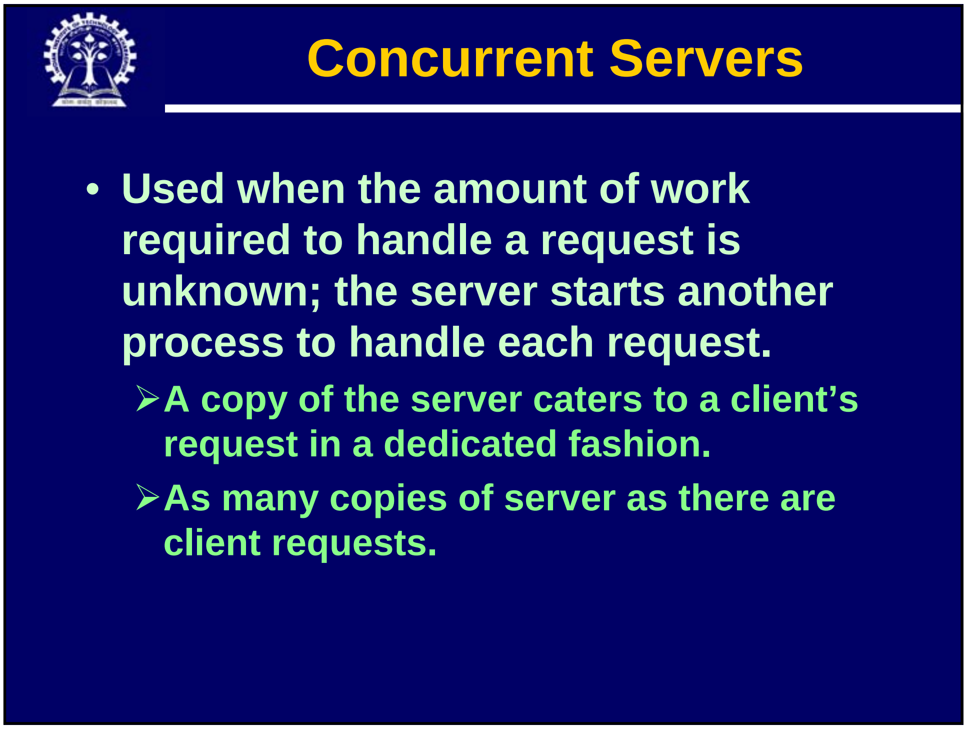
Client-server Model (contd.)

- Roles of the client and the server processes are asymmetric.
- Two types of servers:
  - *Iterative servers*.
  - *Concurrent servers*.



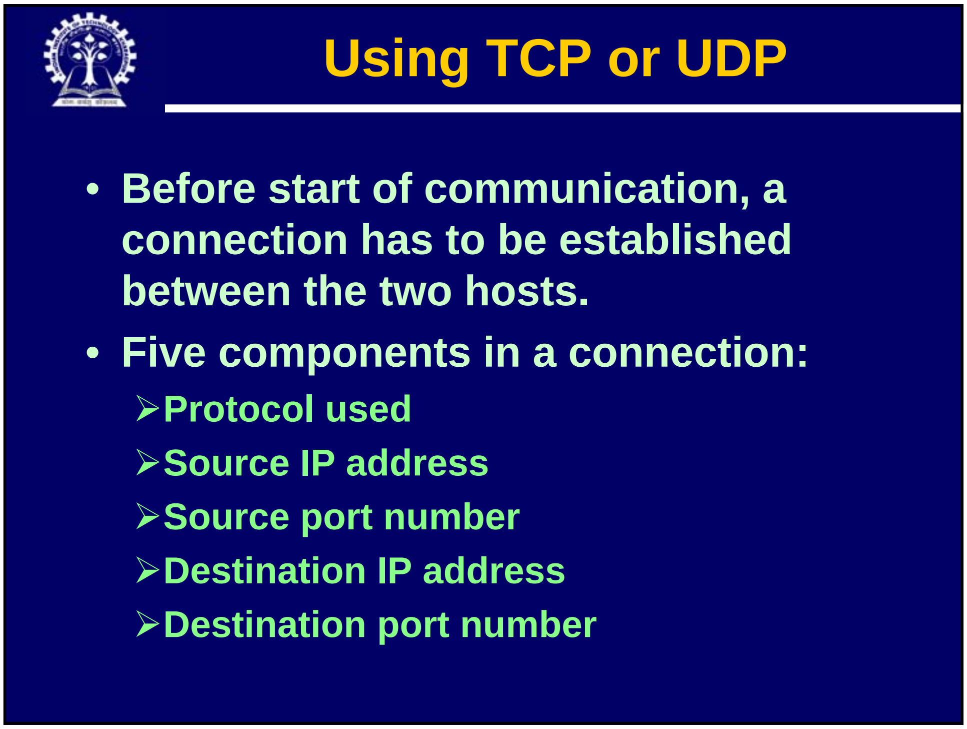
Iterative Servers

- Used when the server process knows in advance how long it takes to handle each request and it handles each request itself.
  - ➢Single copy of server runs at all times.
  - ➢A client may have to wait if the server is busy.



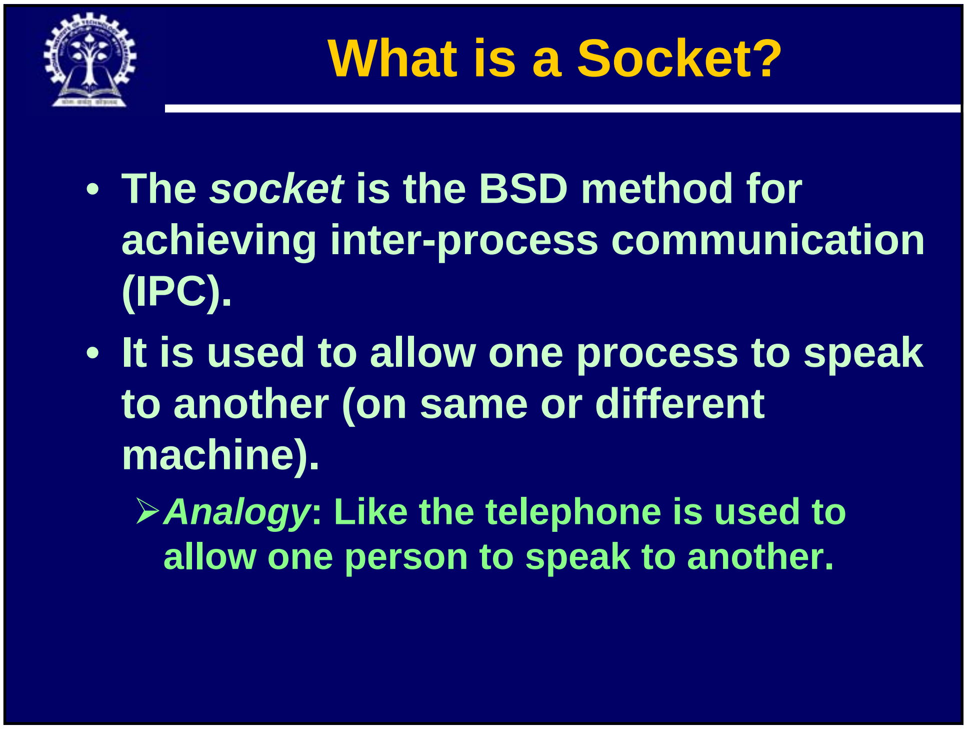
Concurrent Servers

- Used when the amount of work required to handle a request is unknown; the server starts another process to handle each request.
  - ➢A copy of the server caters to a client’s request in a dedicated fashion.
  - ➢As many copies of server as there are client requests.



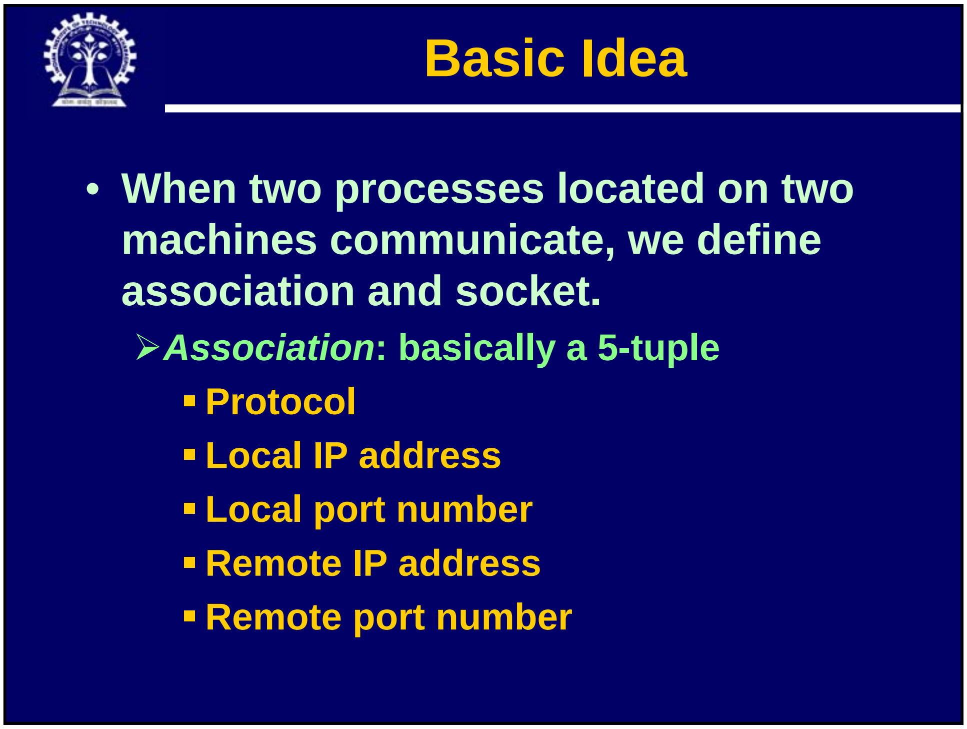
Using TCP or UDP

- Before start of communication, a connection has to be established between the two hosts.
- Five components in a connection:
  - ➢Protocol used
  - ➢Source IP address
  - ➢Source port number
  - ➢Destination IP address
  - ➢Destination port number



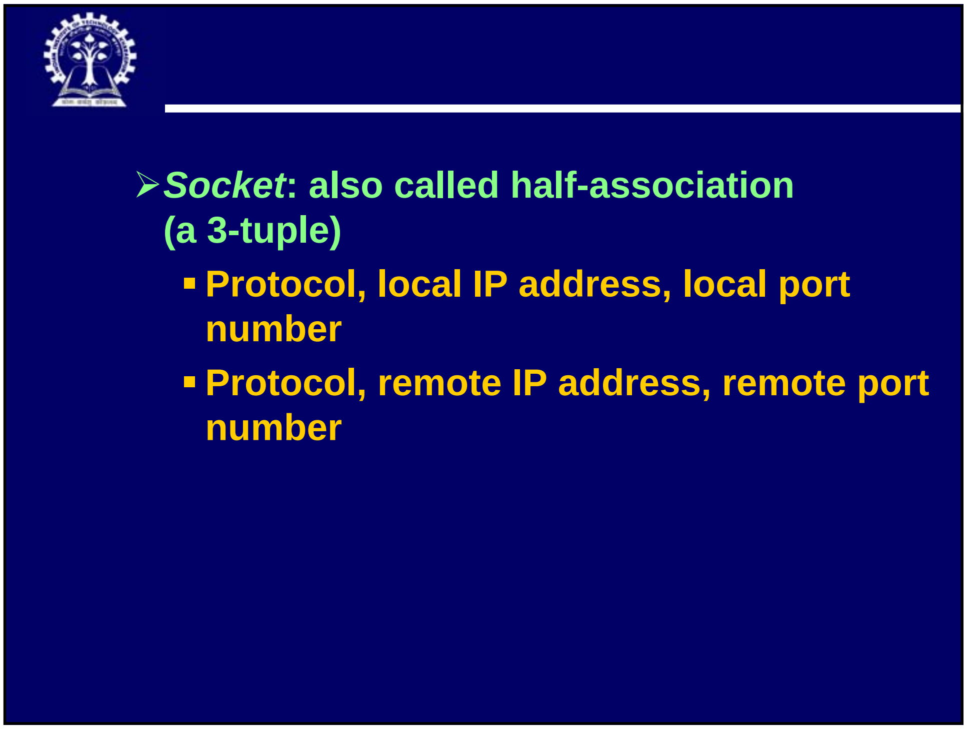
What is a Socket?

- The *socket* is the BSD method for achieving inter-process communication (IPC).
- It is used to allow one process to speak to another (on same or different machine).
  - ➢*Analogy*: Like the telephone is used to allow one person to speak to another.

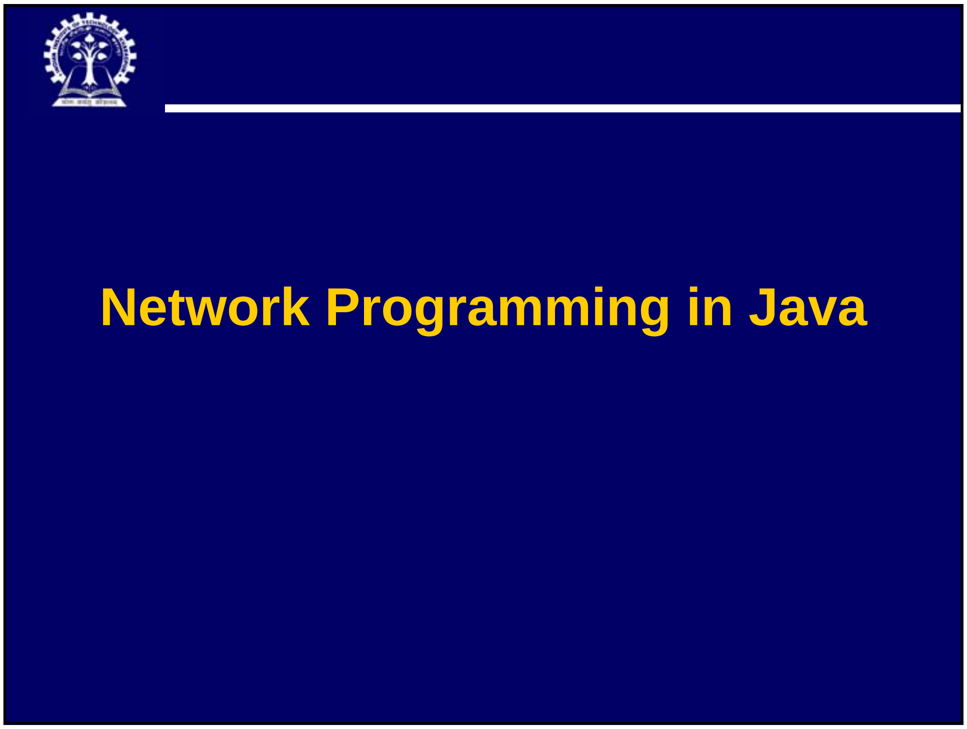


Basic Idea

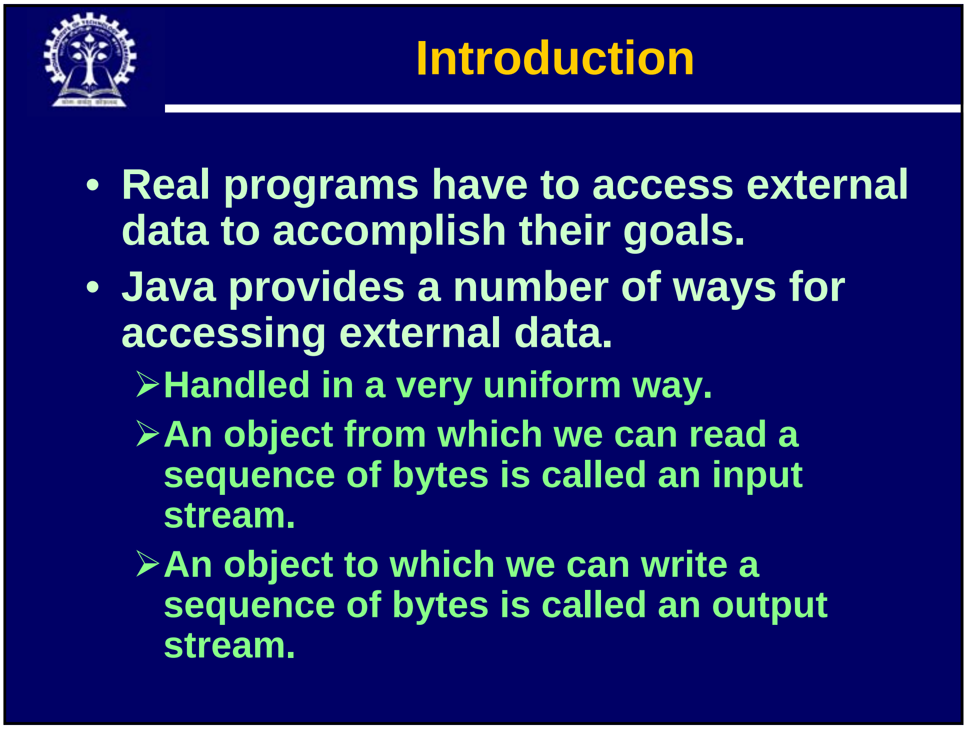
- When two processes located on two machines communicate, we define association and socket.
  - ➢ *Association*: basically a 5-tuple
    - Protocol
    - Local IP address
    - Local port number
    - Remote IP address
    - Remote port number



- ➢ *Socket*: also called half-association (a 3-tuple)
  - Protocol, local IP address, local port number
  - Protocol, remote IP address, remote port number

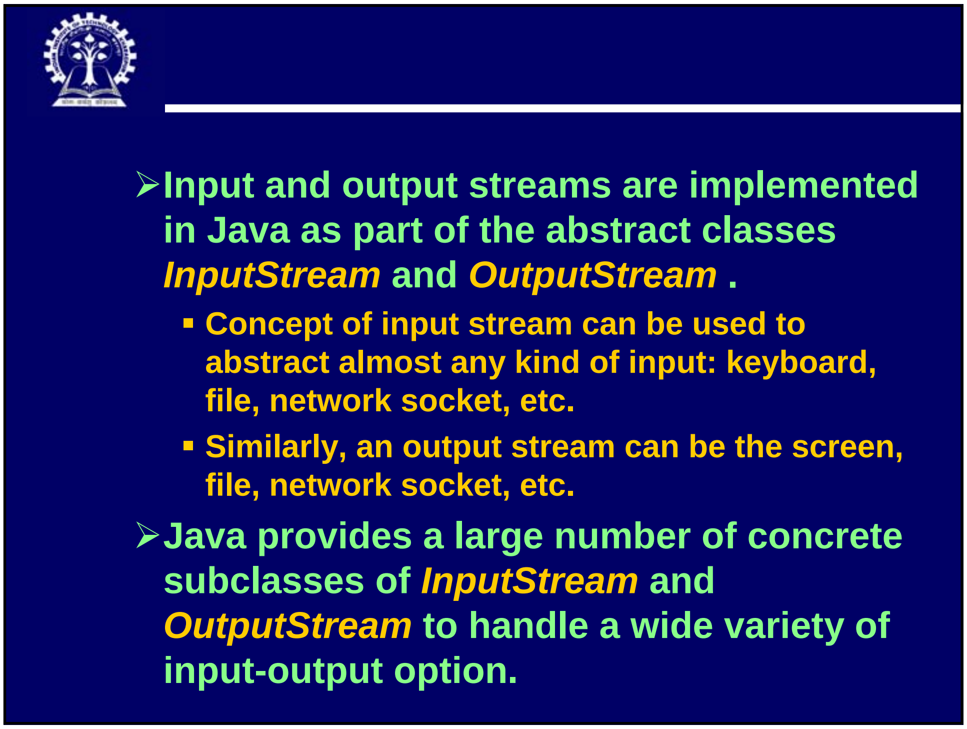


Network Programming in Java

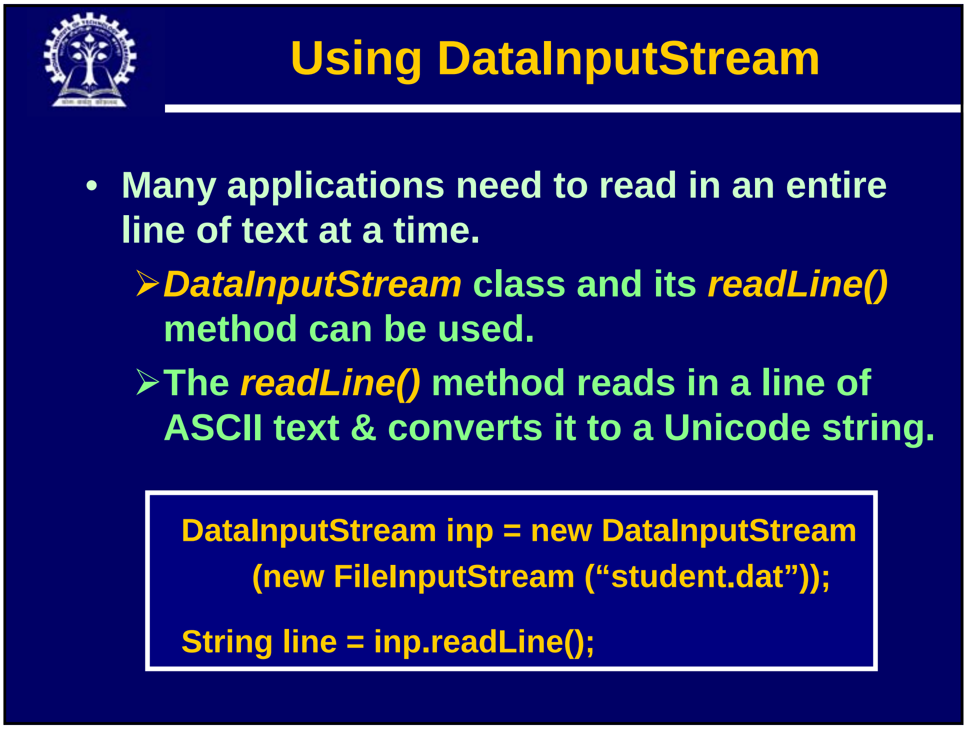


Introduction

- Real programs have to access external data to accomplish their goals.
- Java provides a number of ways for accessing external data.
  - ➢Handled in a very uniform way.
  - ➢An object from which we can read a sequence of bytes is called an input stream.
  - ➢An object to which we can write a sequence of bytes is called an output stream.



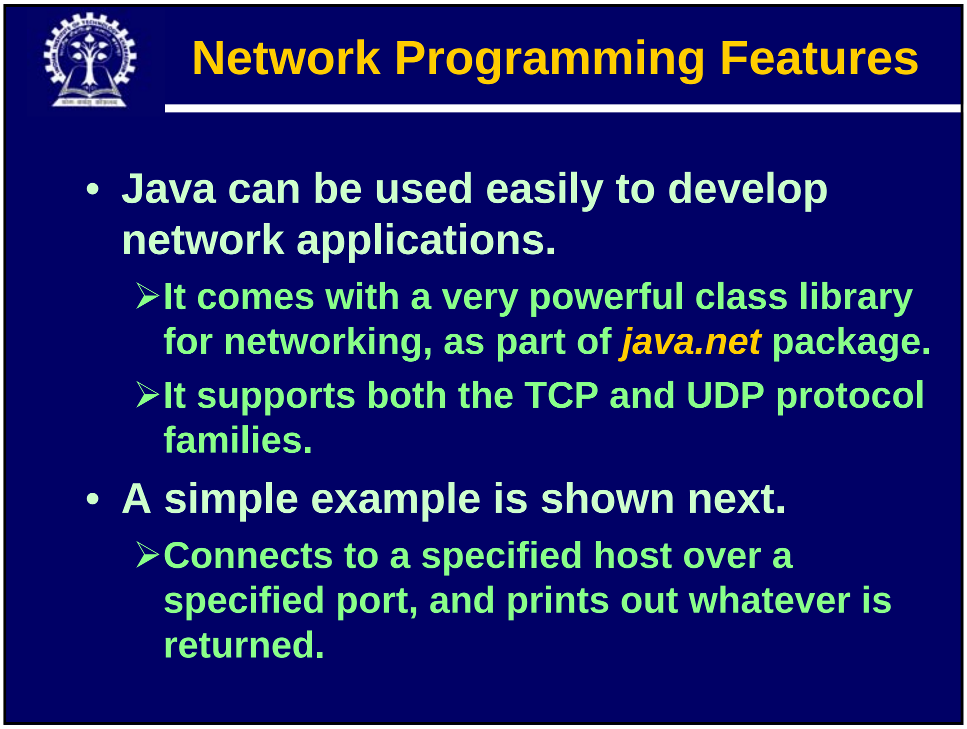
- Input and output streams are implemented in Java as part of the abstract classes *InputStream* and *OutputStream* .
  - Concept of input stream can be used to abstract almost any kind of input: keyboard, file, network socket, etc.
  - Similarly, an output stream can be the screen, file, network socket, etc.
- Java provides a large number of concrete subclasses of *InputStream* and *OutputStream* to handle a wide variety of input-output option.



Using DataInputStream

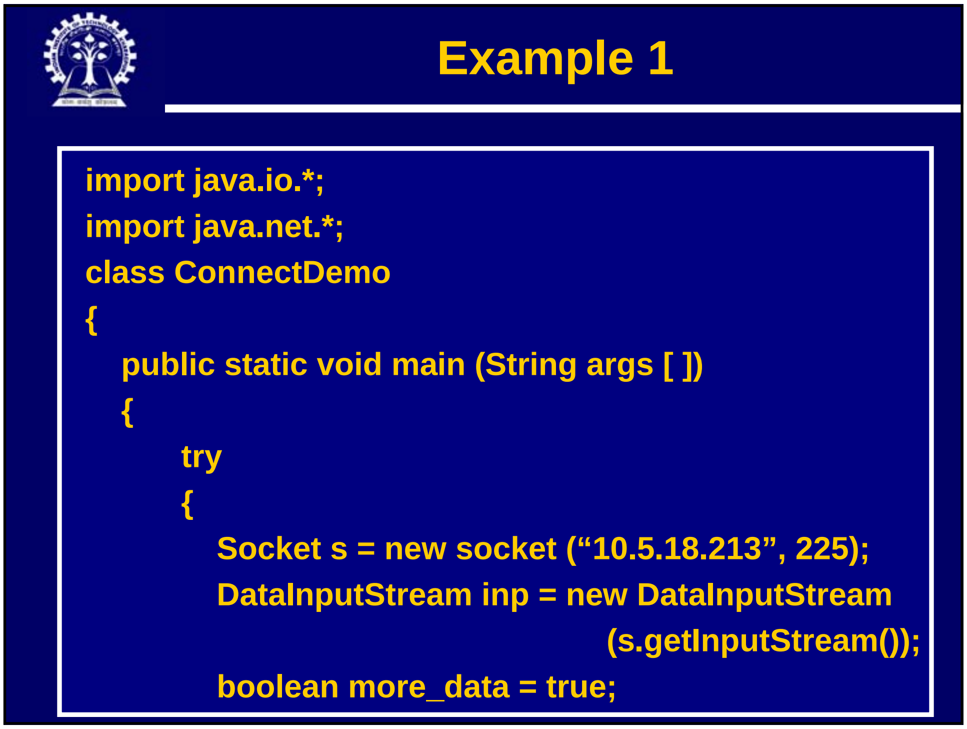
- Many applications need to read in an entire line of text at a time.
  - *DataInputStream* class and its *readLine()* method can be used.
  - The *readLine()* method reads in a line of ASCII text & converts it to a Unicode string.

```
DataInputStream inp = new DataInputStream
        (new FileInputStream (“student.dat”));

String line = inp.readLine();
```



Network Programming Features

- Java can be used easily to develop network applications.
  - It comes with a very powerful class library for networking, as part of *java.net* package.
  - It supports both the TCP and UDP protocol families.
- A simple example is shown next.
  - Connects to a specified host over a specified port, and prints out whatever is returned.



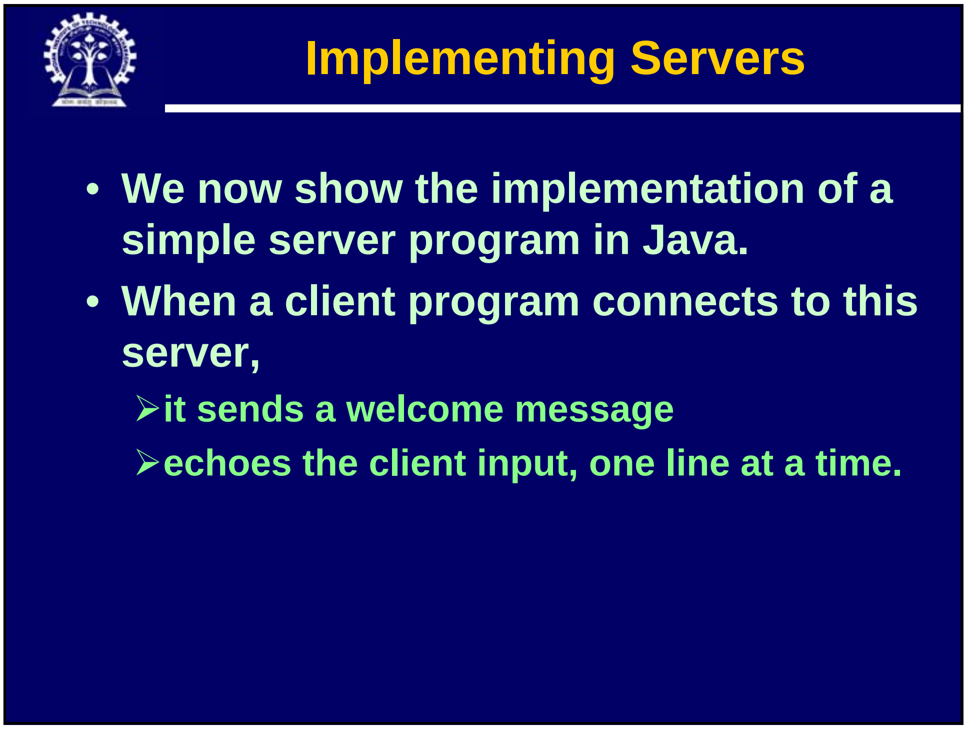
Example 1

```
import java.io.*;
import java.net.*;
class ConnectDemo
{
   public static void main (String args [ ])
   {
        try
        {
           Socket s = new socket ("10.5.18.213", 225);
           DataInputStream inp = new DataInputStream
                                                  (s.getInputStream());
           boolean more_data = true;
```

Example 1 (contd.)

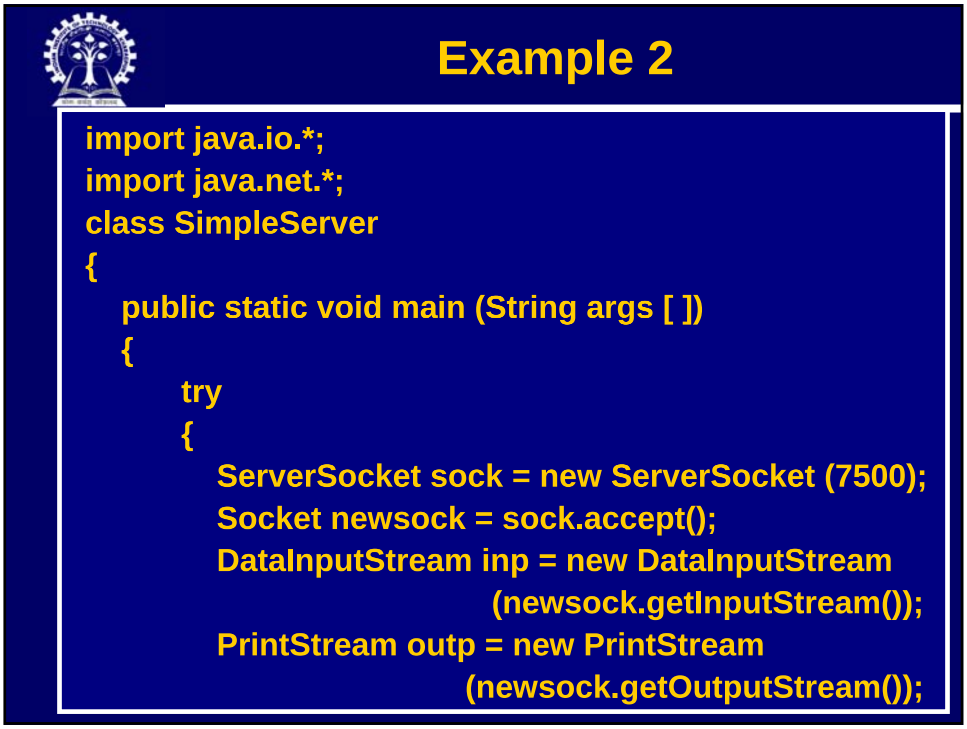
```
             System.out.println (“Established connection”);
             while (more_data)
             {
                  String line = inp.readLine();
                  if (line = = null)    more_data = false;
                  else  System.out.println (line);
             }
          }
          catch (IOException e)
          {   System.out.println (“IO error “ + e)    }
          }
  }
```

- Some points:
  - ➢All networking code are enclosed in the *try … catch* block.
  - ➢Most of the network-related methods throw *IOException* whenever some error occurs.



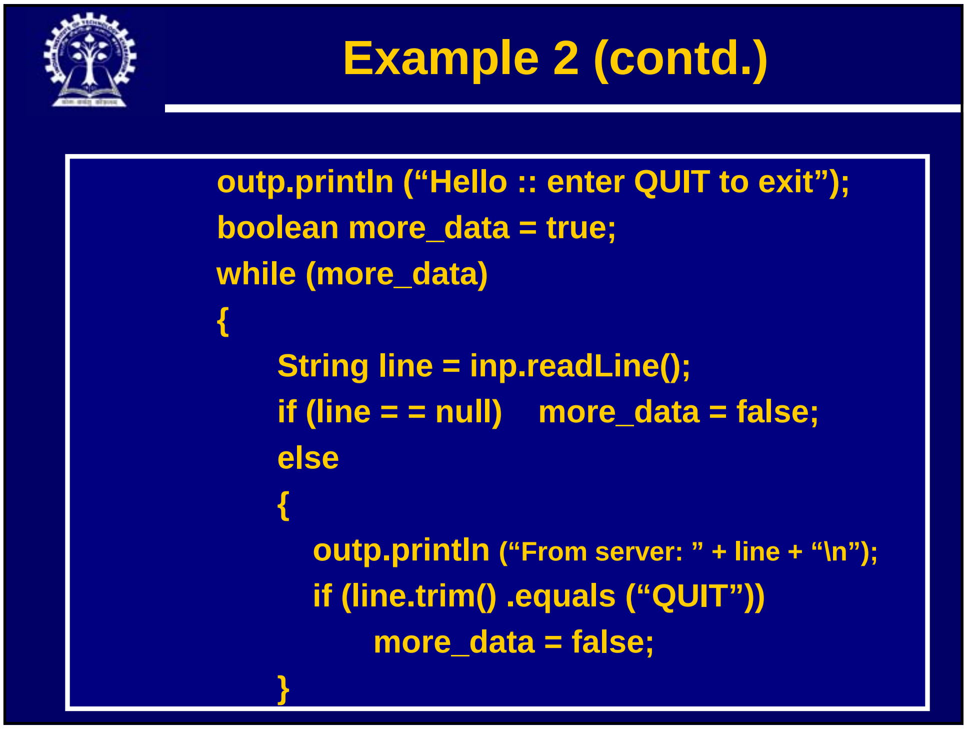
Implementing Servers

- We now show the implementation of a simple server program in Java.
- When a client program connects to this server,
  - ➢it sends a welcome message
  - ➢echoes the client input, one line at a time.



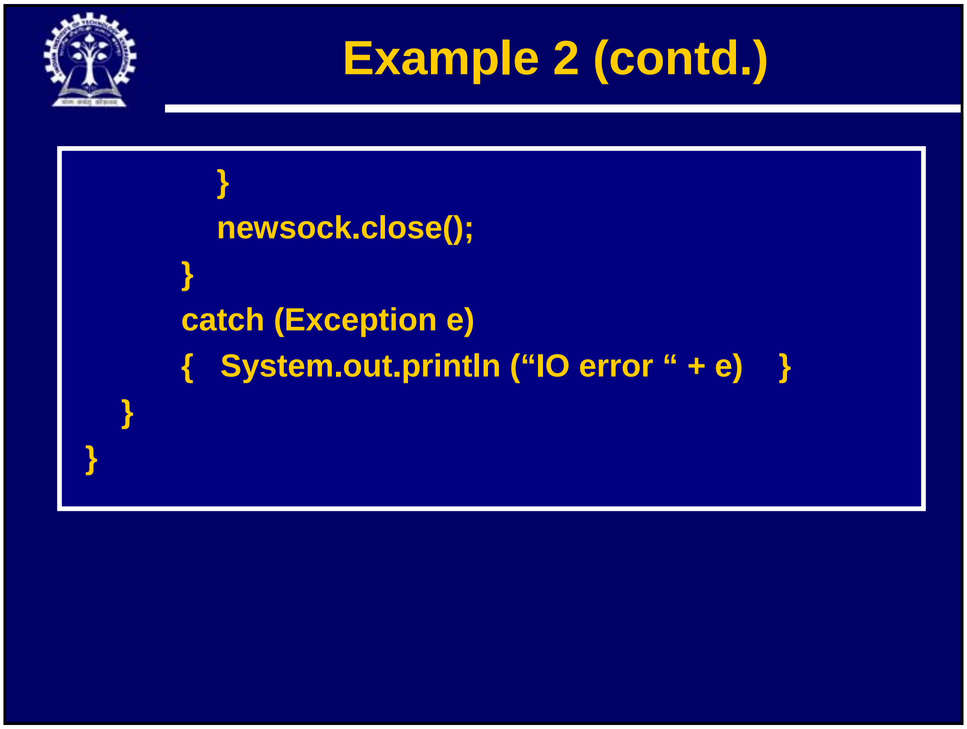
Example 2

```
import java.io.*;
import java.net.*;
class SimpleServer
{
   public static void main (String args [ ])
   {
        try
        {
           ServerSocket sock = new ServerSocket (7500);
           Socket newsock = sock.accept();
           DataInputStream inp = new DataInputStream
                                 (newsock.getInputStream());
           PrintStream outp = new PrintStream
                               (newsock.getOutputStream());
```



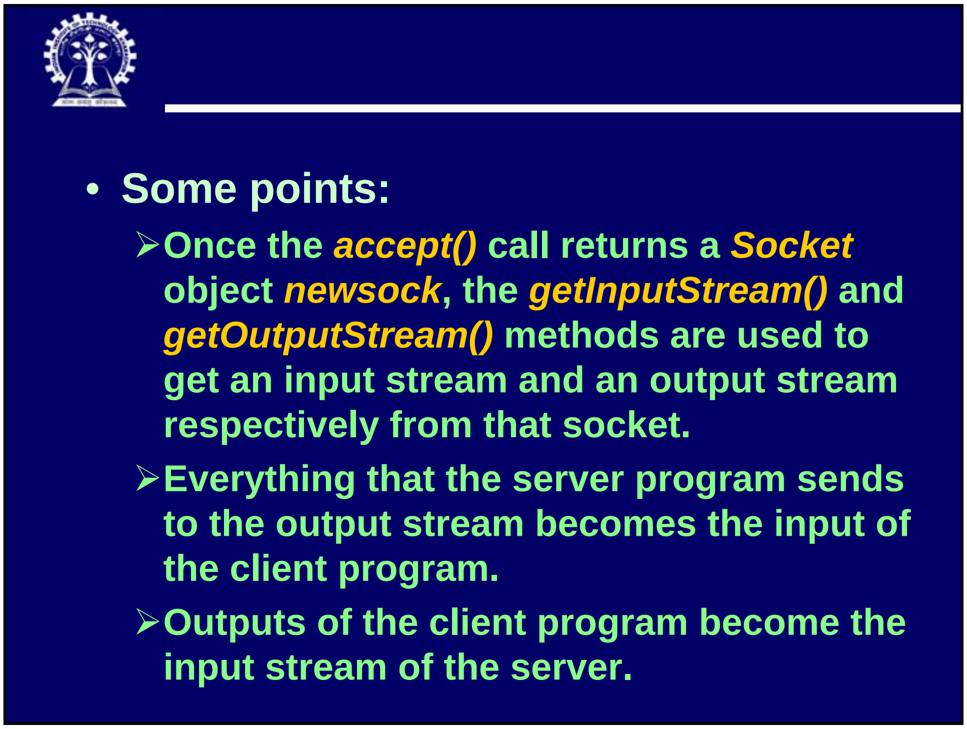
Example 2 (contd.)

```
outp.println (“Hello :: enter QUIT to exit”);
boolean more_data = true;
while (more_data)
{
    String line = inp.readLine();
    if (line = = null)    more_data = false;
    else
    {
      outp.println (“From server: ” + line + “\n”);
      if (line.trim() .equals (“QUIT”))
          more_data = false;
    }
```

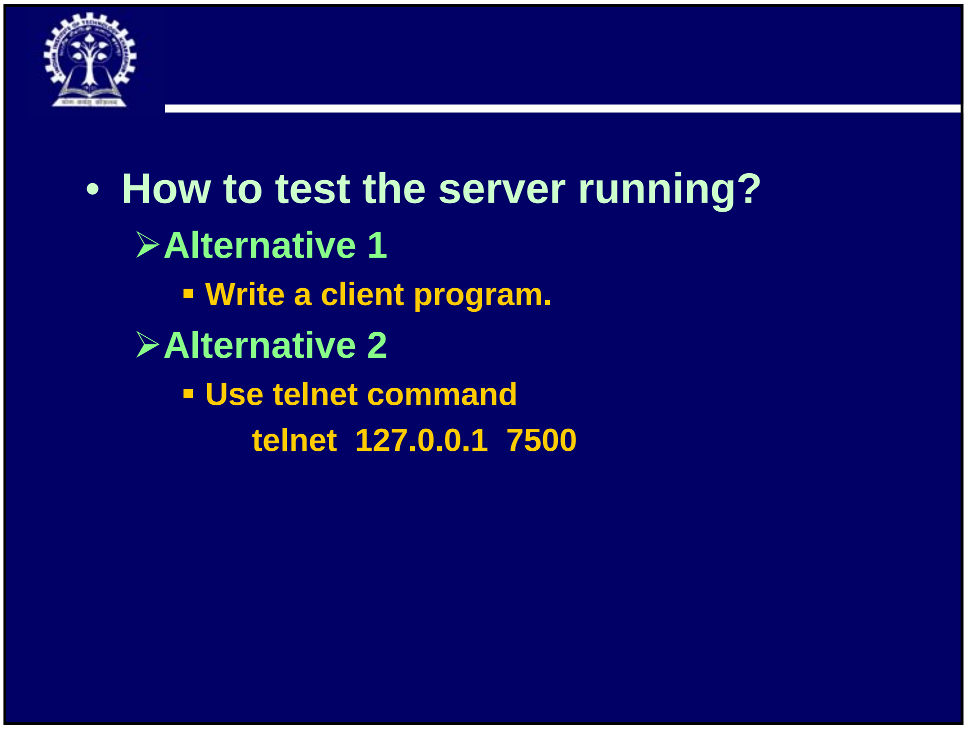


Example 2 (contd.)

```
        }
        newsock.close();
      }
      catch (Exception e)
      {   System.out.println (“IO error “ + e)    }
  }
}
```

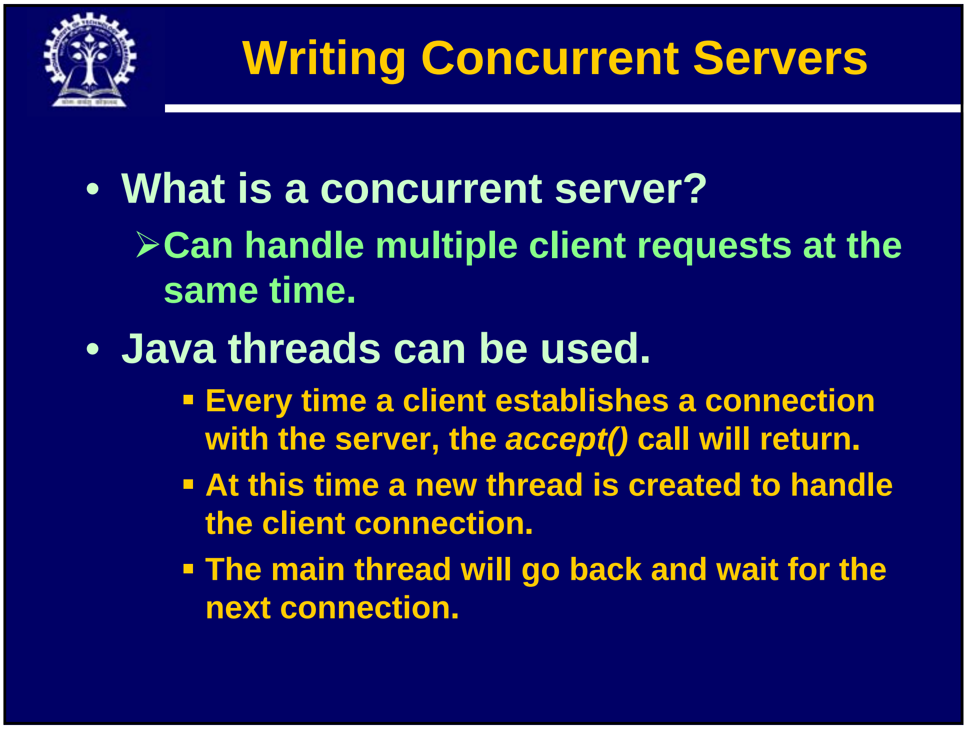


- Some points:
  - Once the ***accept()*** call returns a ***Socket*** object ***newsock***, the ***getInputStream()*** and ***getOutputStream()*** methods are used to get an input stream and an output stream respectively from that socket.
  - Everything that the server program sends to the output stream becomes the input of the client program.
  - Outputs of the client program become the input stream of the server.



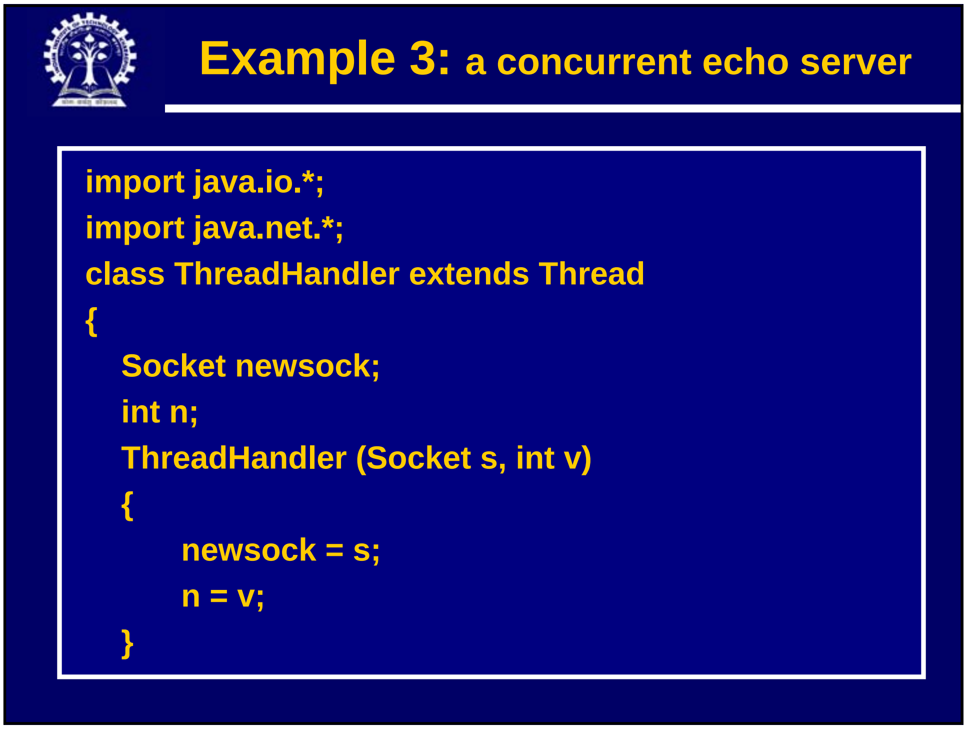
- How to test the server running?
  - Alternative 1
    - Write a client program.
  - Alternative 2
    - Use telnet command

```
telnet  127.0.0.1  7500
```



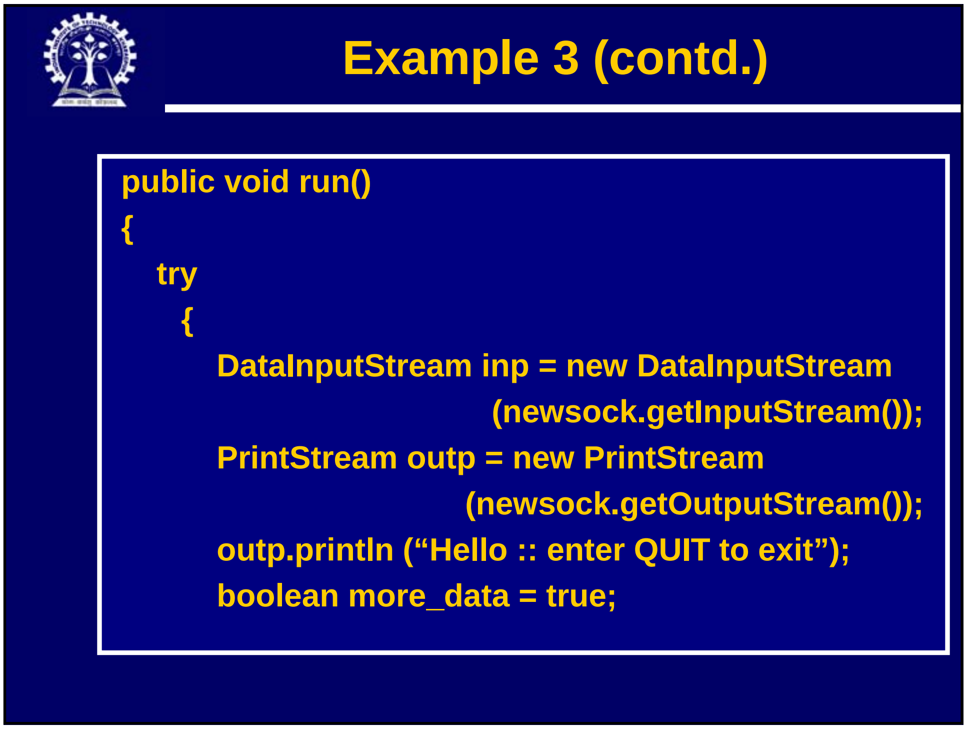
Writing Concurrent Servers

- What is a concurrent server?
  - Can handle multiple client requests at the same time.
- Java threads can be used.
  - Every time a client establishes a connection with the server, the *accept()* call will return.
  - At this time a new thread is created to handle the client connection.
  - The main thread will go back and wait for the next connection.



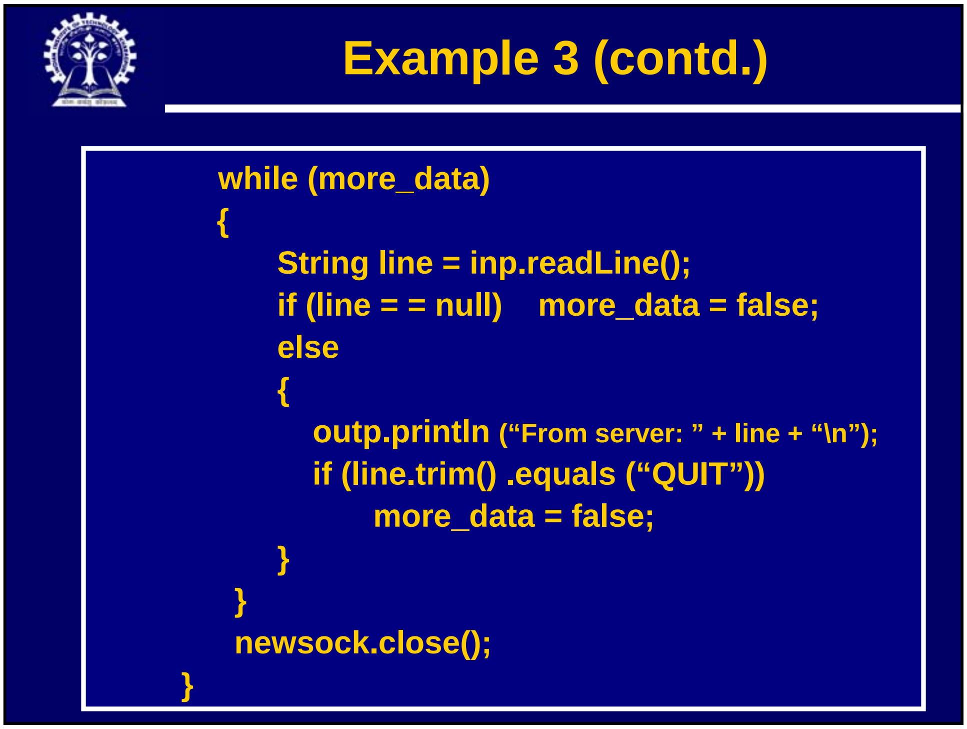
Example 3: a concurrent echo server

```
import java.io.*;
import java.net.*;
class ThreadHandler extends Thread
{
   Socket newsock;
   int n;
   ThreadHandler (Socket s, int v)
   {
        newsock = s;
        n = v;
   }
```



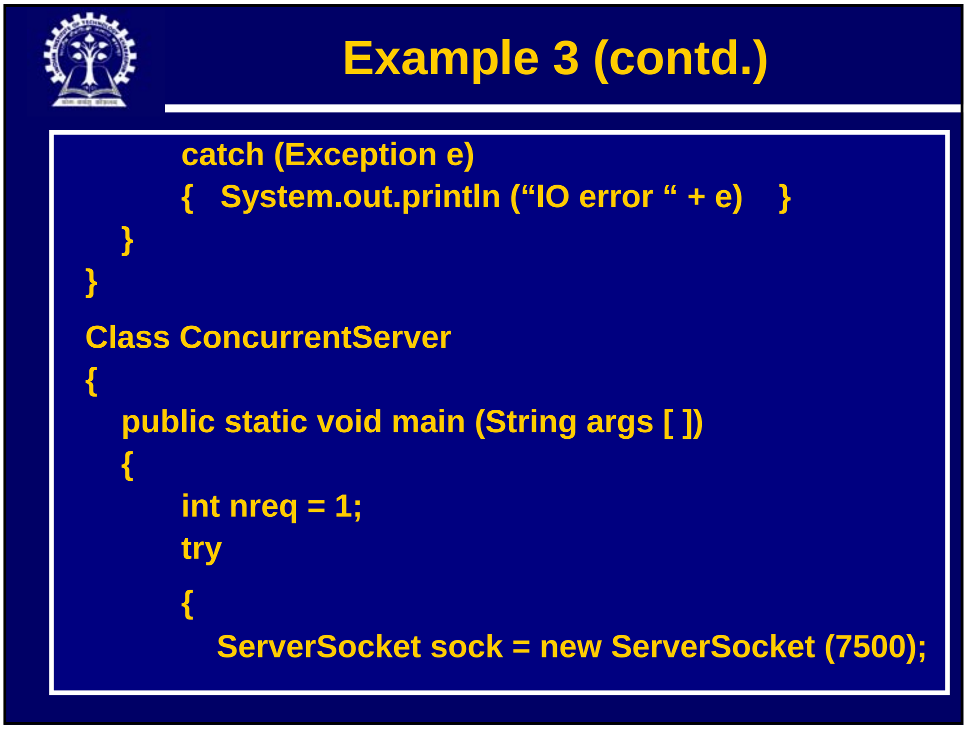
Example 3 (contd.)

```
public void run()
{
   try
     {
          DataInputStream inp = new DataInputStream
                                      (newsock.getInputStream());
          PrintStream outp = new PrintStream
                                    (newsock.getOutputStream());
          outp.println (“Hello :: enter QUIT to exit”);
          boolean more_data = true;
```



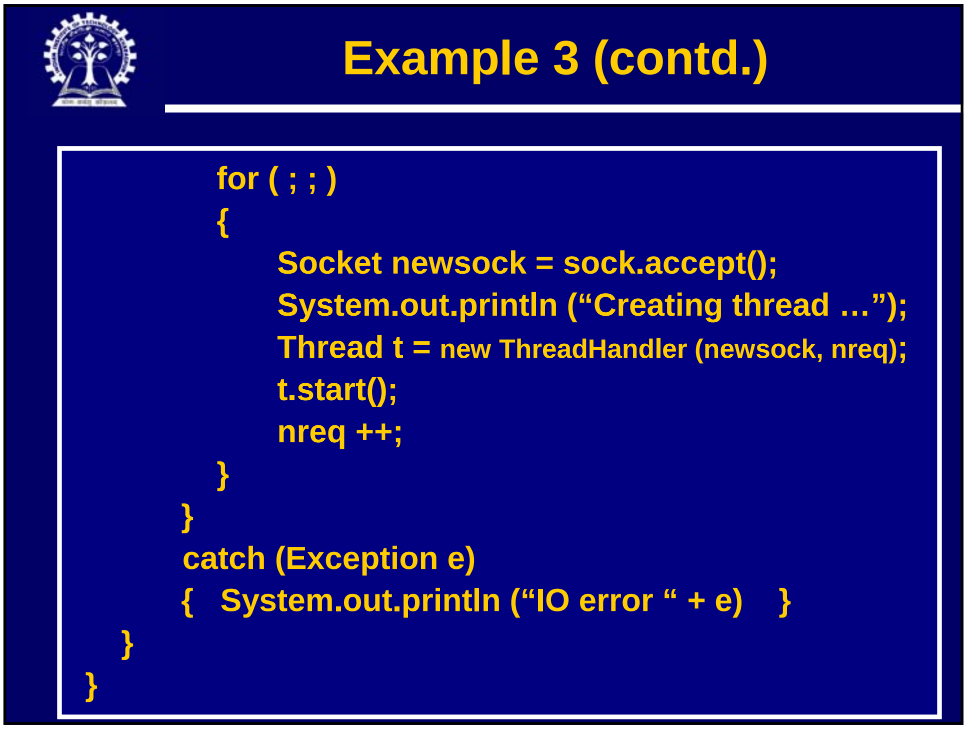
Example 3 (contd.)

```
          while (more_data)
          {
               String line = inp.readLine();
               if (line = = null)    more_data = false;
               else
               {
                  outp.println (“From server: ” + line + “\n”);
                  if (line.trim() .equals (“QUIT”))
                       more_data = false;
               }
           }
           newsock.close();
     }
```



Example 3 (contd.)

```
            catch (Exception e)
            {   System.out.println (“IO error “ + e)    }
      }
}
Class ConcurrentServer
{
      public static void main (String args [ ])
      {
            int nreq = 1;
            try
            {
                  ServerSocket sock = new ServerSocket (7500);
```



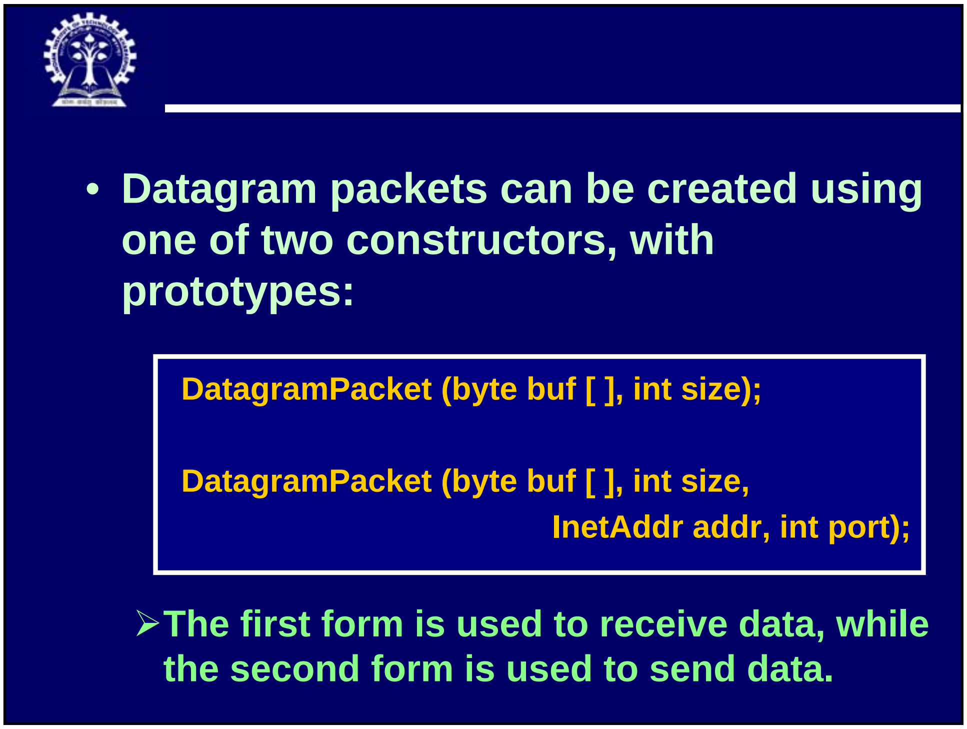
Example 3 (contd.)

```
                  for ( ; ; )
                  {
                        Socket newsock = sock.accept();
                        System.out.println (“Creating thread …”);
                        Thread t = new ThreadHandler (newsock, nreq);
                        t.start();
                        nreq ++;
                  }
            }
            catch (Exception e)
            {   System.out.println (“IO error “ + e)    }
      }
}
```

Connectionless Servers

- The examples shown so far are based on connection-oriented transfers.
  - ➢Provides reliable, bidirectional, point-to-point, stream based connections between two hosts.
  - ➢Based on the TCP protocol.
- Java also supports connectionless transfers using datagrams.
  - ➢Based on the UDP protocol.

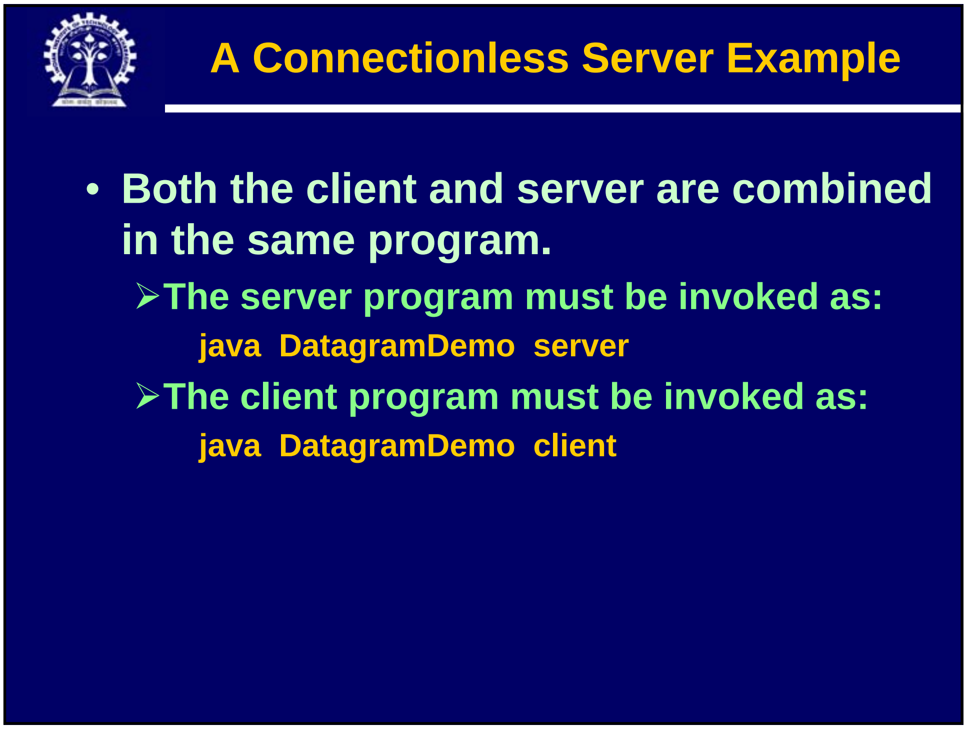
  - ➢May be used in situations where optimal performance is required, and the overhead of explicit data verification is justified.
- Java implements datagrams on top of the UDP protocol using two classes.
  - ➢The *DatagramPacket* class acts as the data container.
  - ➢The *DatagramSocket* class is the means for sending and receiving datagrams.



- Datagram packets can be created using one of two constructors, with prototypes:

```
DatagramPacket (byte buf [ ], int size);

DatagramPacket (byte buf [ ], int size,
                                  InetAddr addr, int port);
```

  - The first form is used to receive data, while the second form is used to send data.



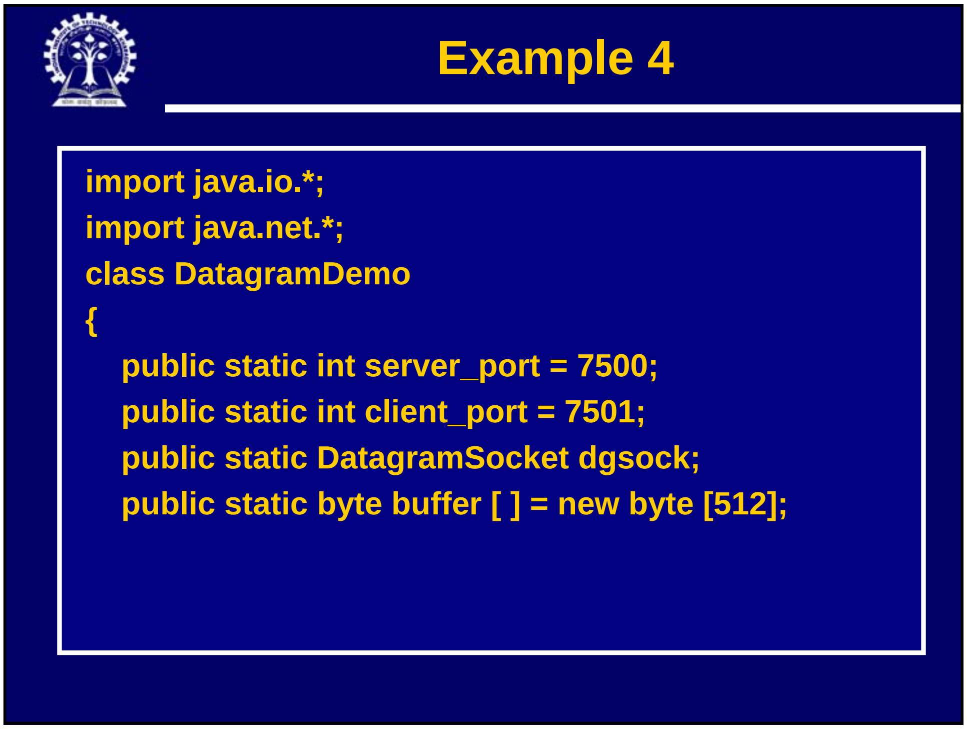
A Connectionless Server Example

- Both the client and server are combined in the same program.
  - The server program must be invoked as:

    ```
    java  DatagramDemo  server
    ```

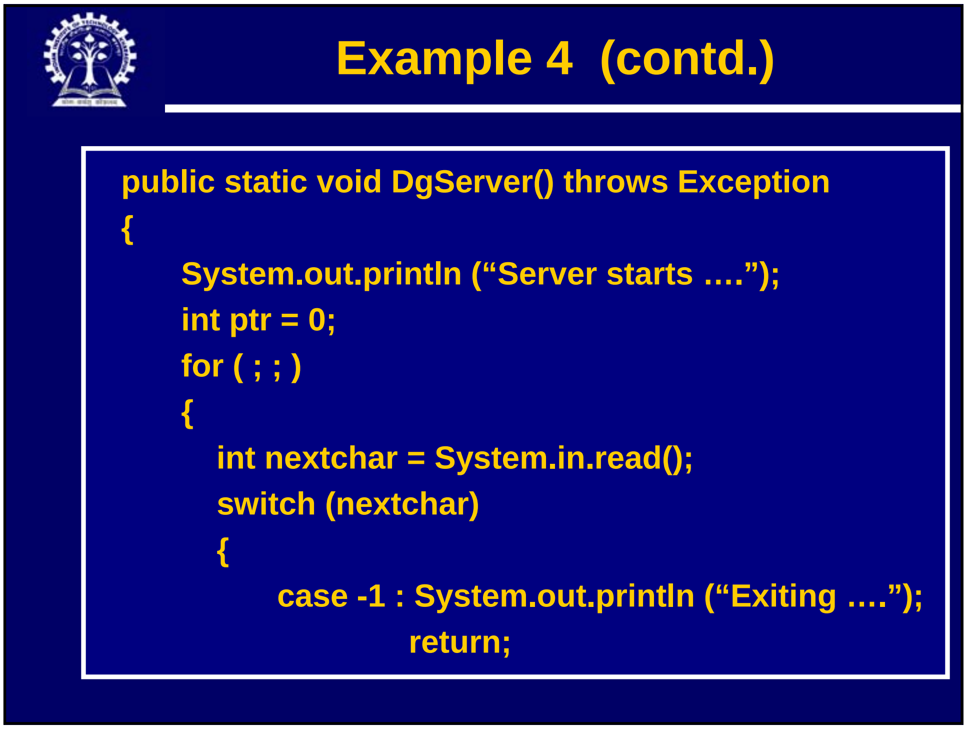
  - The client program must be invoked as:

    ```
    java  DatagramDemo  client
    ```



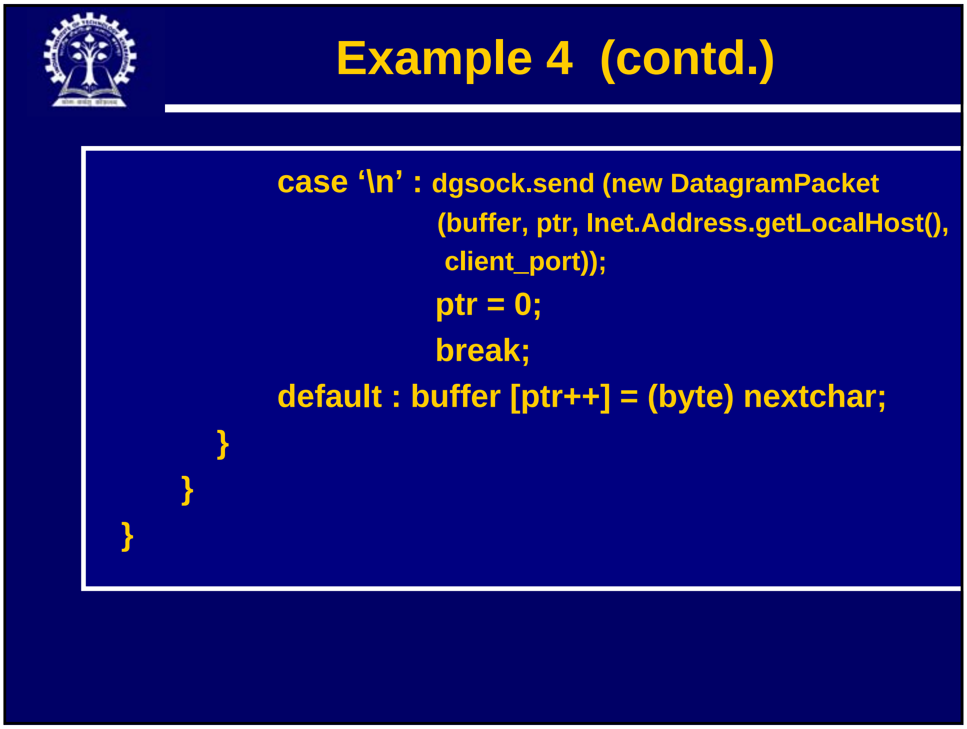
Example 4

```
import java.io.*;
import java.net.*;
class DatagramDemo
{
   public static int server_port = 7500;
   public static int client_port = 7501;
   public static DatagramSocket dgsock;
   public static byte buffer [ ] = new byte [512];
```



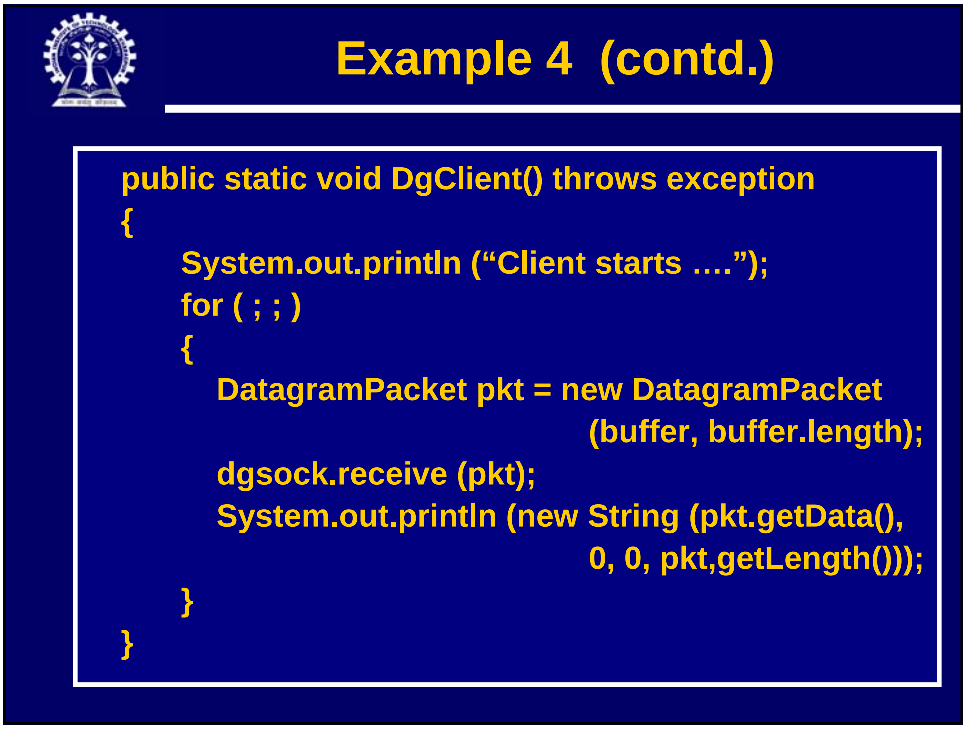
Example 4 (contd.)

```
   public static void DgServer() throws Exception
   {
        System.out.println (“Server starts ….”);
        int ptr = 0;
        for ( ; ; )
        {
           int nextchar = System.in.read();
           switch (nextchar)
           {
                case -1 : System.out.println (“Exiting ….”);
                           return;
```



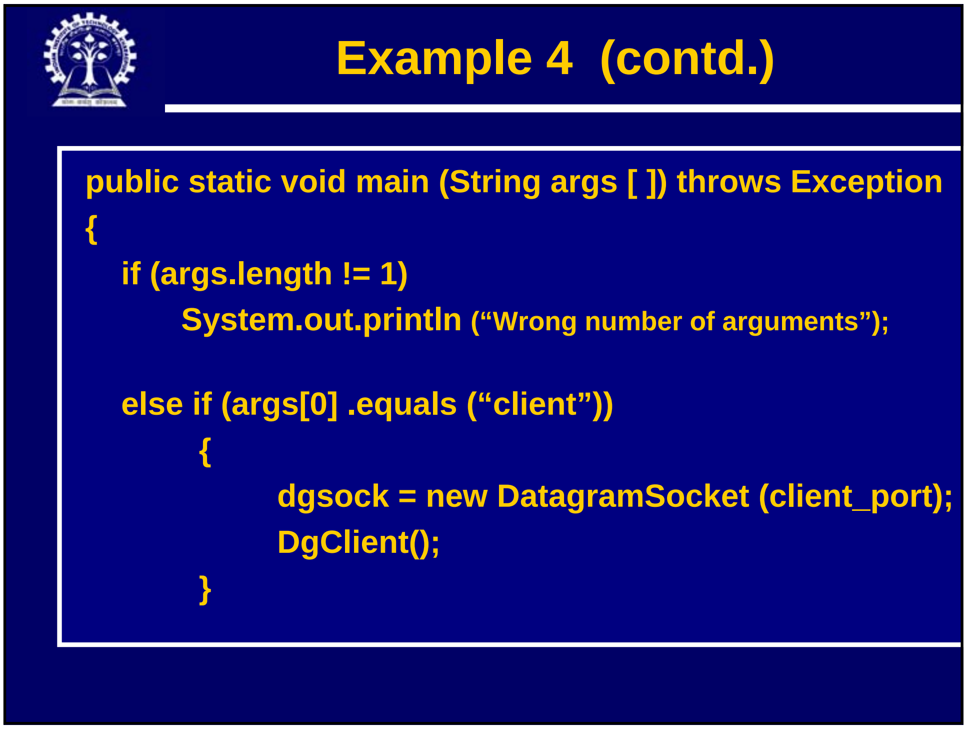
Example 4 (contd.)

```
                case ‘\n’ : dgsock.send (new DatagramPacket
                                (buffer, ptr, Inet.Address.getLocalHost(),
                                 client_port));
                            ptr = 0;
                            break;
                default : buffer [ptr++] = (byte) nextchar;
            }
        }
}
```



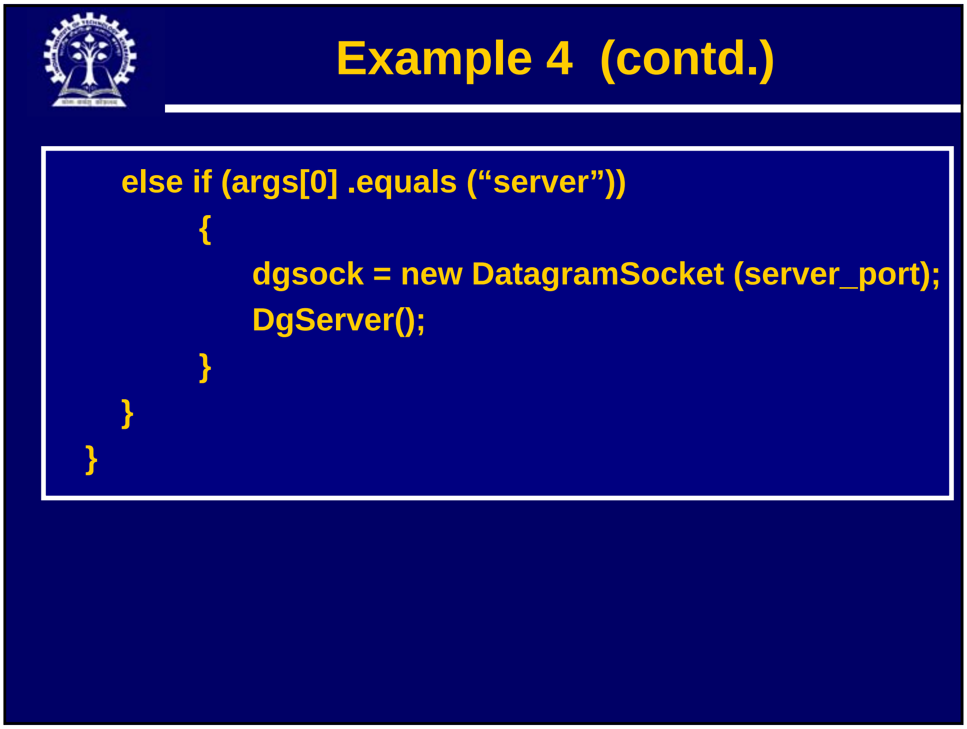
Example 4 (contd.)

```
public static void DgClient() throws exception
{
        System.out.println (“Client starts ….”);
        for ( ; ; )
        {
            DatagramPacket pkt = new DatagramPacket
                                        (buffer, buffer.length);
            dgsock.receive (pkt);
            System.out.println (new String (pkt.getData(),
                                        0, 0, pkt,getLength()));
        }
}
```



Example 4 (contd.)

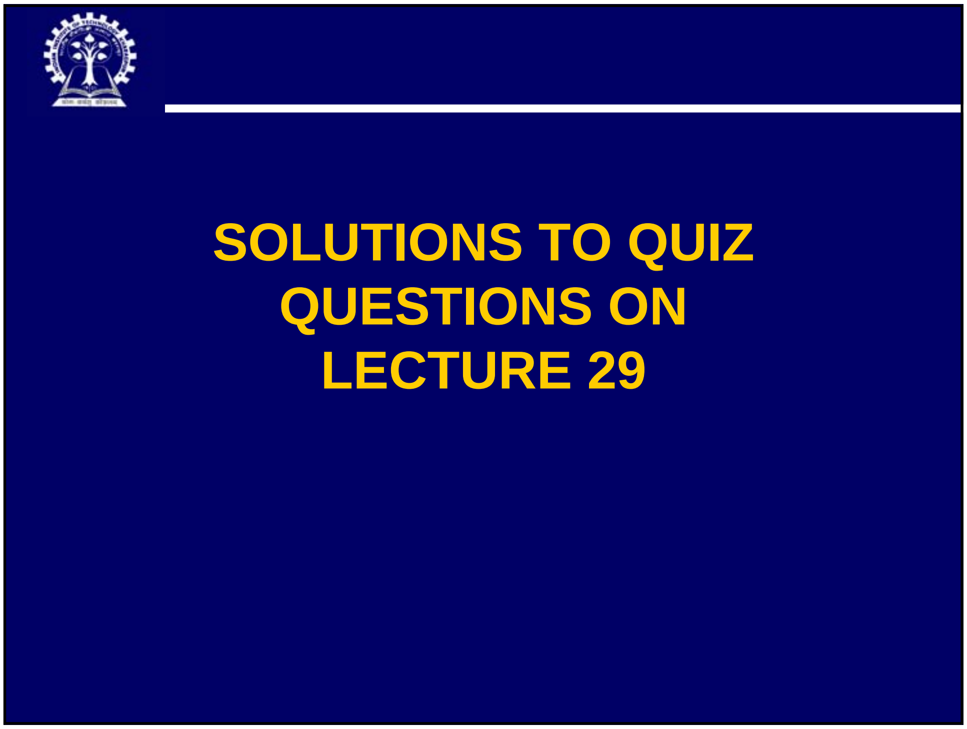
```
public static void main (String args [ ]) throws Exception
{
   if (args.length != 1)
        System.out.println ("Wrong number of arguments");

   else if (args[0] .equals ("client"))
         {
                dgsock = new DatagramSocket (client_port);
                DgClient();
         }
```



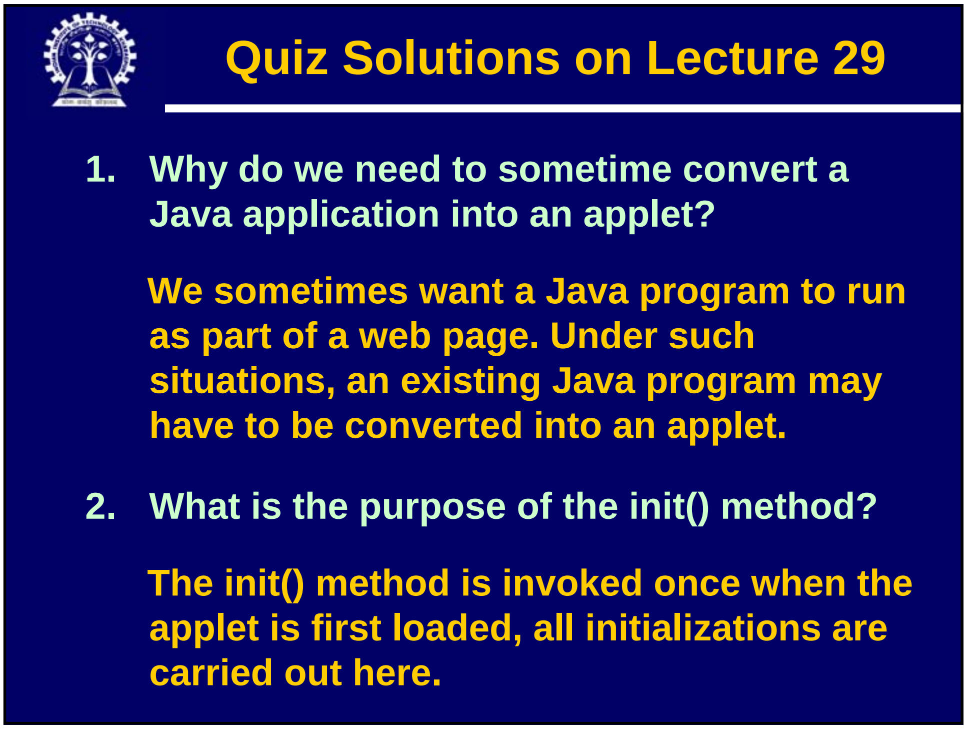
Example 4 (contd.)

```
   else if (args[0] .equals ("server"))
         {
              dgsock = new DatagramSocket (server_port);
              DgServer();
         }
   }
}
```

End of Lecture 30



SOLUTIONS TO QUIZ QUESTIONS ON LECTURE 29



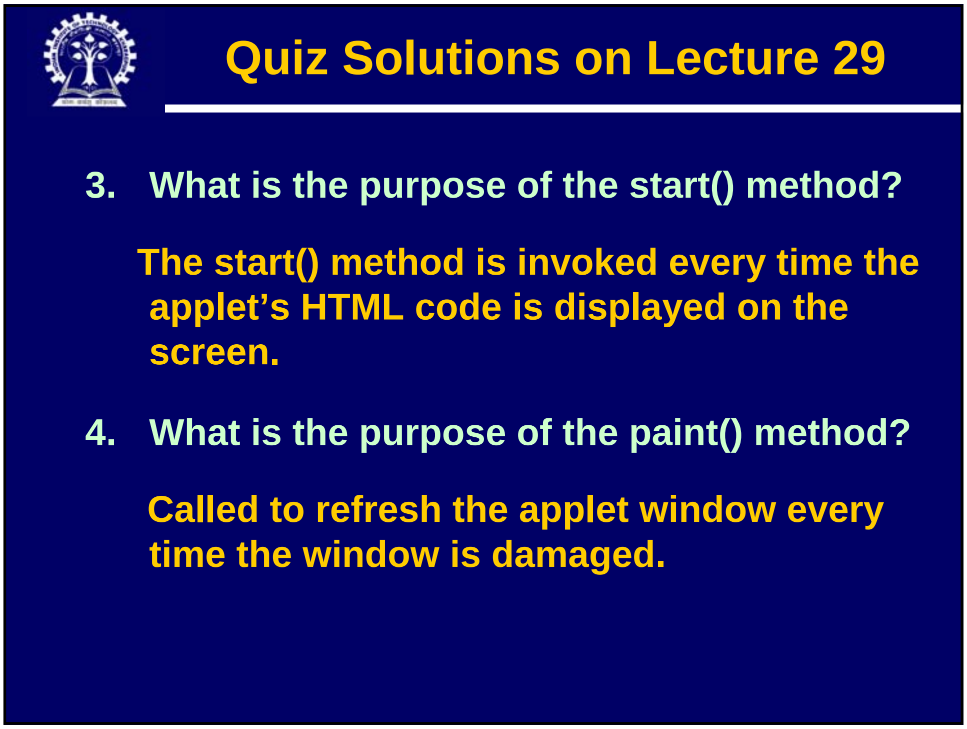
Quiz Solutions on Lecture 29

1. Why do we need to sometime convert a Java application into an applet?

   We sometimes want a Java program to run as part of a web page. Under such situations, an existing Java program may have to be converted into an applet.

2. What is the purpose of the init() method?

   The init() method is invoked once when the applet is first loaded, all initializations are carried out here.



Quiz Solutions on Lecture 29

3. What is the purpose of the start() method?

   The start() method is invoked every time the applet’s HTML code is displayed on the screen.

4. What is the purpose of the paint() method?

   Called to refresh the applet window every time the window is damaged.

Quiz Solutions on Lecture 29

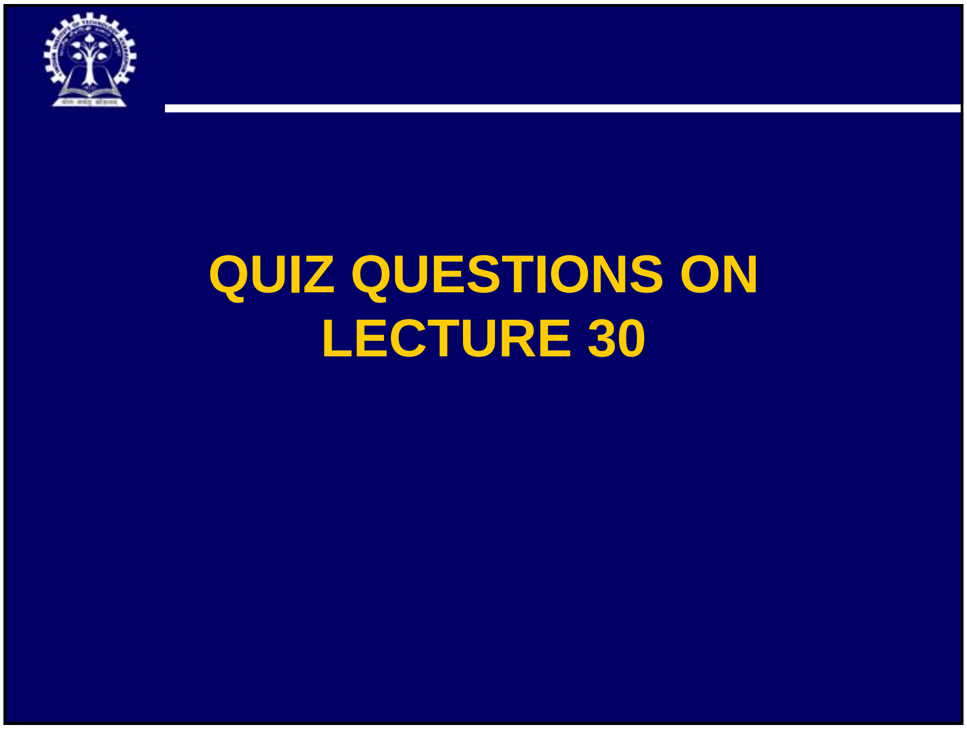
5. How can an applet A invoke a method of applet B, where both A and B are included in the same HTML page?

   Applet A must first call the getAppletContext() method to gain access to applet B. A can then proceed to access the public variables or invoke the public methods in B.

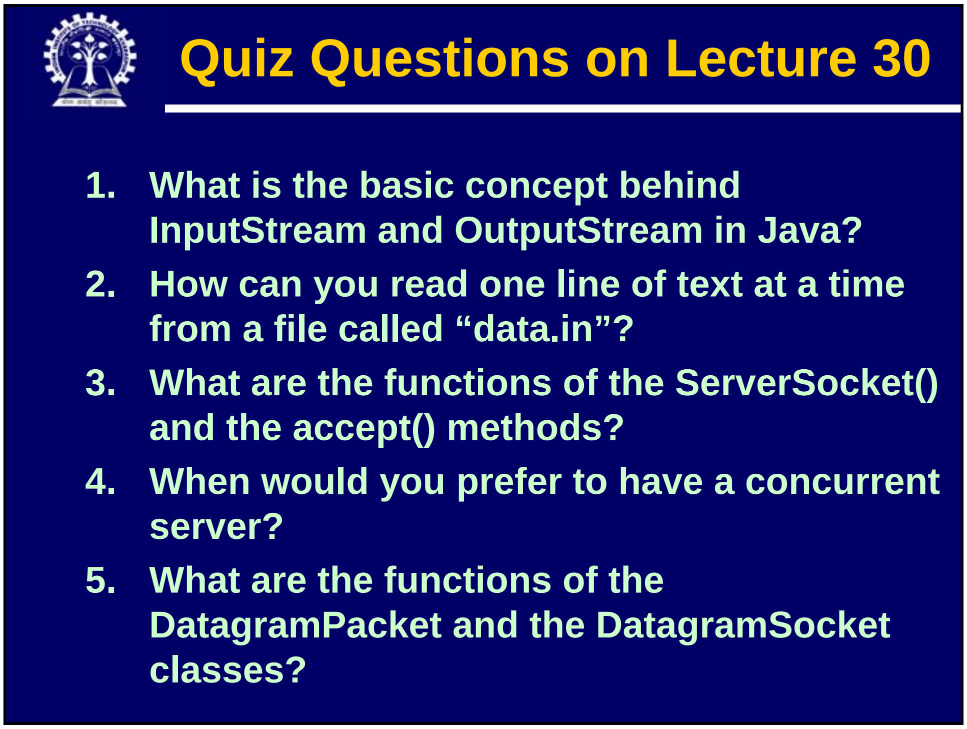
Quiz Solutions on Lecture 29

6. How do you change the displayed image on an applet?

   By invoking the drawImage() method.



QUIZ QUESTIONS ON LECTURE 30



Quiz Questions on Lecture 30

1. What is the basic concept behind InputStream and OutputStream in Java?
2. How can you read one line of text at a time from a file called “data.in”?
3. What are the functions of the ServerSocket() and the accept() methods?
4. When would you prefer to have a concurrent server?
5. What are the functions of the DatagramPacket and the DatagramSocket classes?