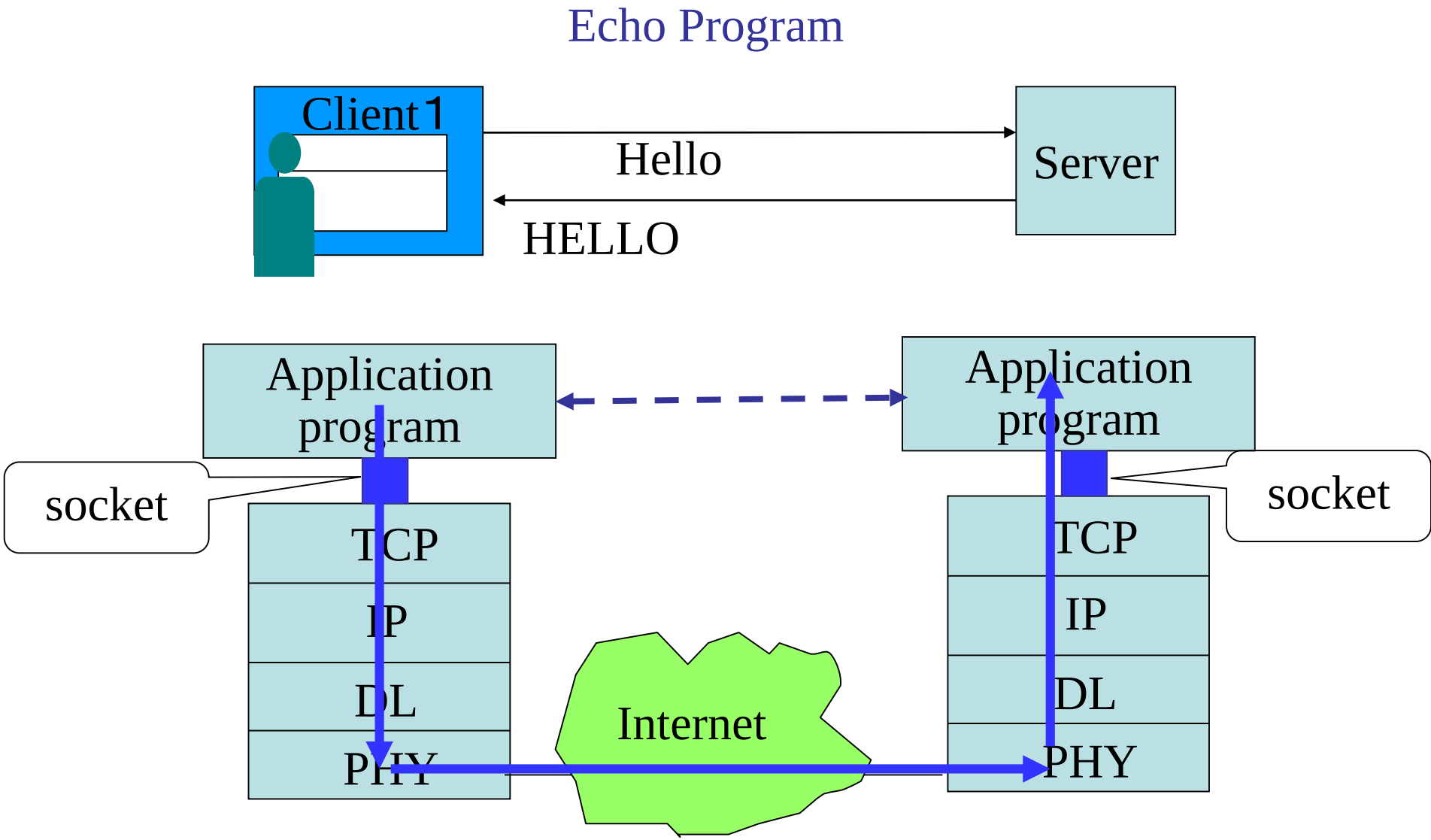
Java Network
(Socket)

Contents

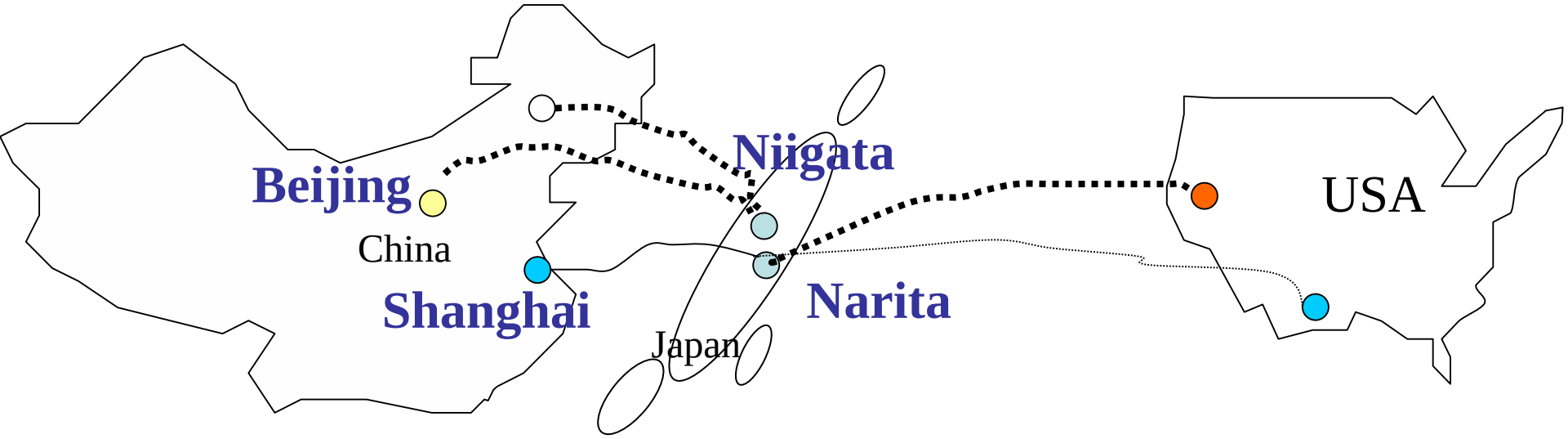
- What is Socket
- Server Sockets and Sockets
- Datagram Sockets and Packets
- Client and Server Application

Socket : Interface for Application Layer



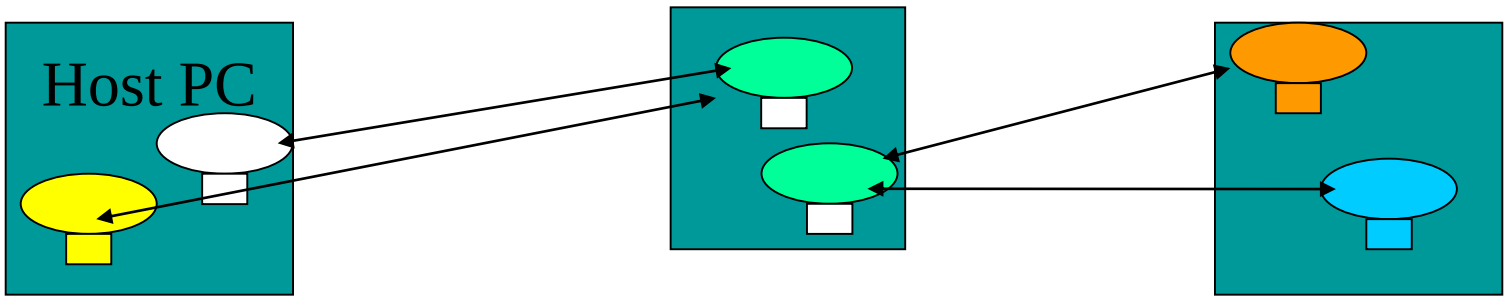
Socket Address：IP and Port

For example: Country and Airport
Air lines：country A airport Q to
country B airport R



communication：A host Q port to B host R port

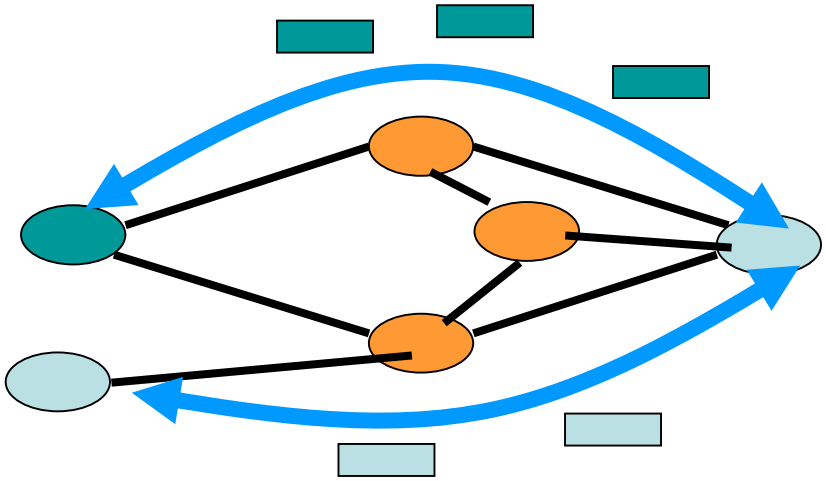
1port：N ports

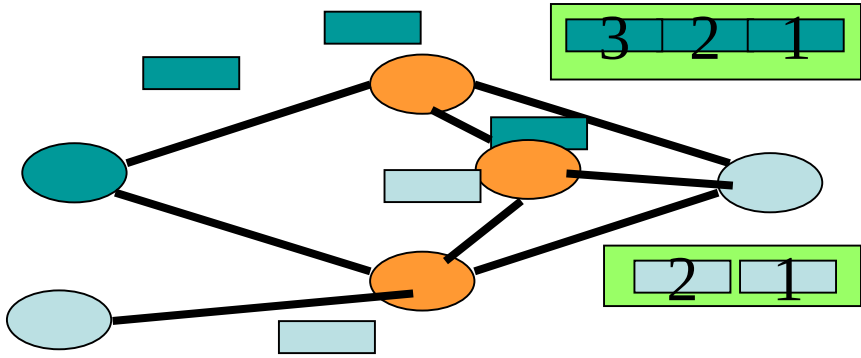


Two types of Data Transportation Services

1. Connection oriented

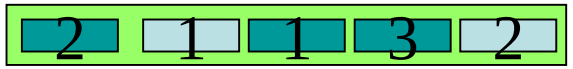
- Transmission Control Protocol (TCP) : To obtain reliable, sequenced data exchange.

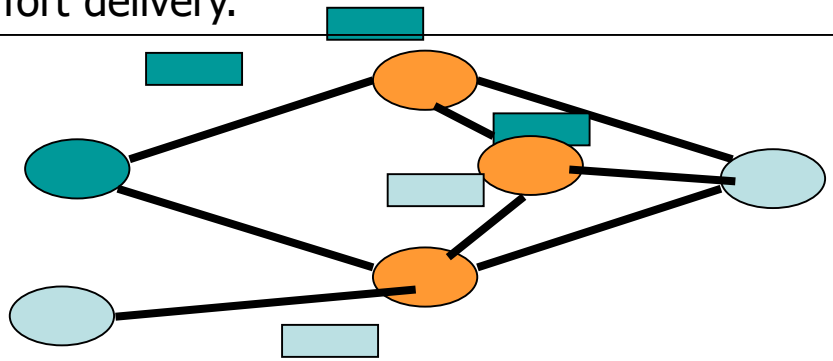


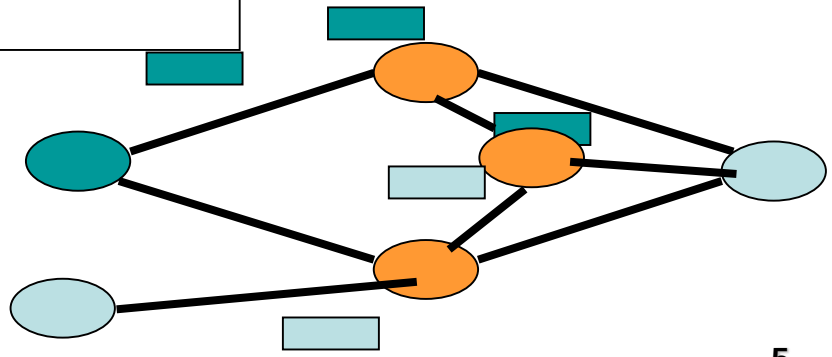


2. Connectionless

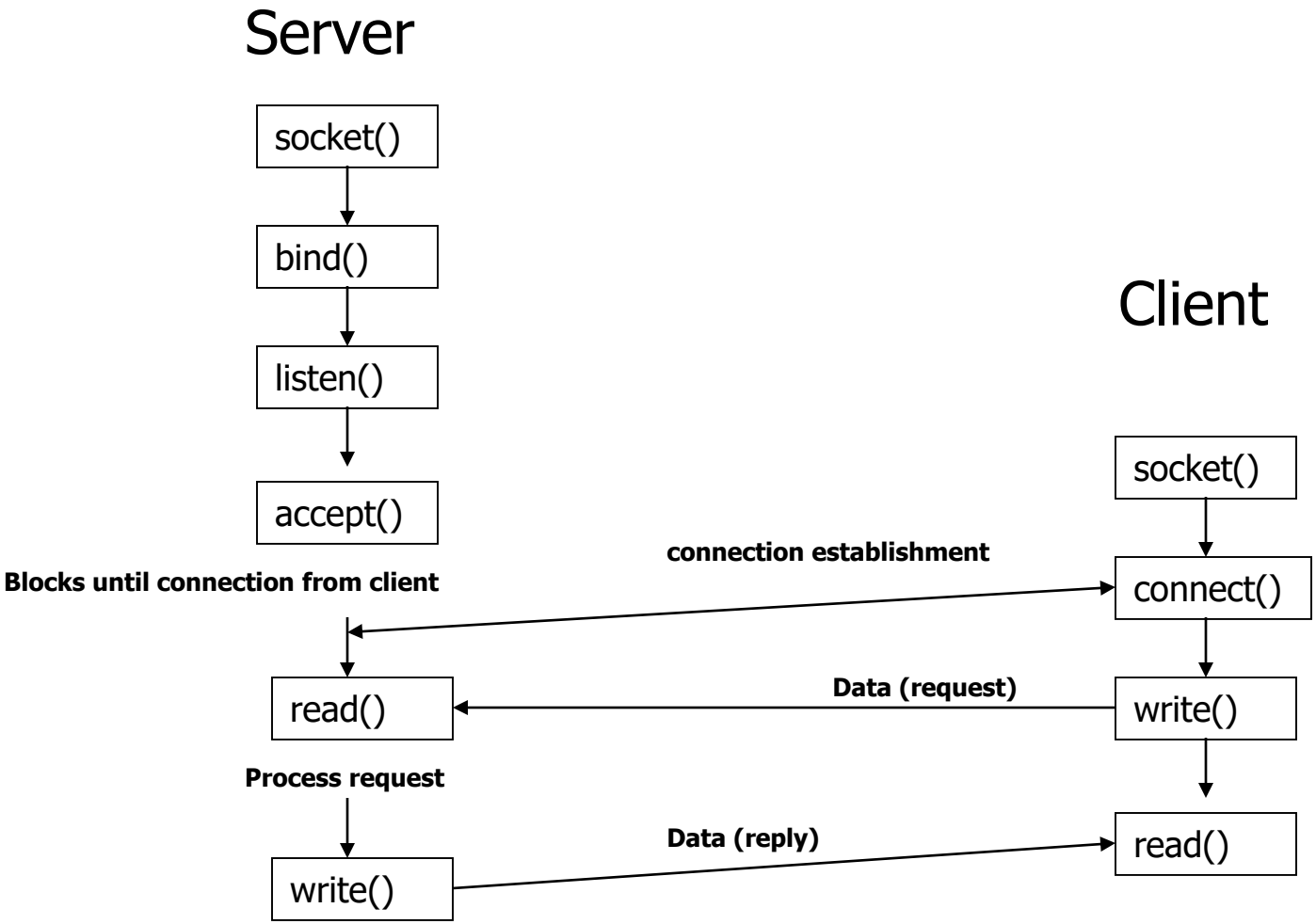
- User Datagram Protocol (UDP) : To obtain a more efficient, best-effort delivery.







Socket Call for Connection-Oriented Protocol



Server Sockets and Sockets

ServerSocket Constructor

This class implements server sockets. A server sock et waits for requests to come in over the network. It performs some operation based on that request, an d then possibly returns a result to the requester.

ServerSocket(int port) throws IOException

- Creates a server socket, bound to the specified port.

```
ServerSocket ss = new ServerSocket(port);
```

accept() Method

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

```
Socket s = ss.accept();
```

close() Method

void close() throws IOException
- Closes this socket.

```
s.close();
```

Socket Class

This class implements client sockets (also called j ust "sockets"). A socket is an endpoint for commu nication between two machines.

Socket(String hostName, int port) throws UnknownHostException, IOException
- Creates a stream socket and connects it to the specified port number at the specified IP address.

```
Socket s = new Socket(server, port);
```

getInputStream(), getOutputStream Method

InputStream getInputStream() throws IOException
-Returns an input stream for this socket.
OutputStream getOutputStream() throws IOException
- Returns an output stream for this socket.

```
OutputStream os = s.getOutputStream();
```

close()

void close() throws IOException
- Closes this socket.

http://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html

http://docs.oracle.com/javase/8/docs/api/java/net/Socket.html

Server Sockets and Sockets

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

class ServerSocketDemo {
  public static void main(String args[]) {
    try {

      // Get Port
      int port = Integer.parseInt(args[0]);
      Random random = new Random();
      //Create Server Socket
      ServerSocket ss = new ServerSocket(port);
      //Create Infinite Loop
      while(true) {
        //Accept Incoming Requests
        Socket s = ss.accept();

        //Write Result to Client
        OutputStream os = s.getOutputStream();
       DataOutputStream dos = new
DataOutputStream(os);
        dos.writeInt(random.nextInt());

        //Close socket
        s.close();
      }
    }
    catch (Exception e) {
      System.out.println("Exception: " + e);    }
  }
}
```

```
class SocketDemo {
  public static void main(String args[]) {
    try {
      //Get Server and Port
      String server = args[0];
      int port = Integer.parseInt(args[1]);
      //Create socket
      Socket s = new Socket(server, port);
      //Read random number from server
      InputStream is = s.getInputStream();
     DataInputStream dis = new
DataInputStream(is);
      int i = dis.readInt();
      //Display Result
      System.out.println(i);
      //Close Socket
      s.close();
    }
    catch (Exception e) {
      System.out.println("Exception: " + e);    }
  }
}
```

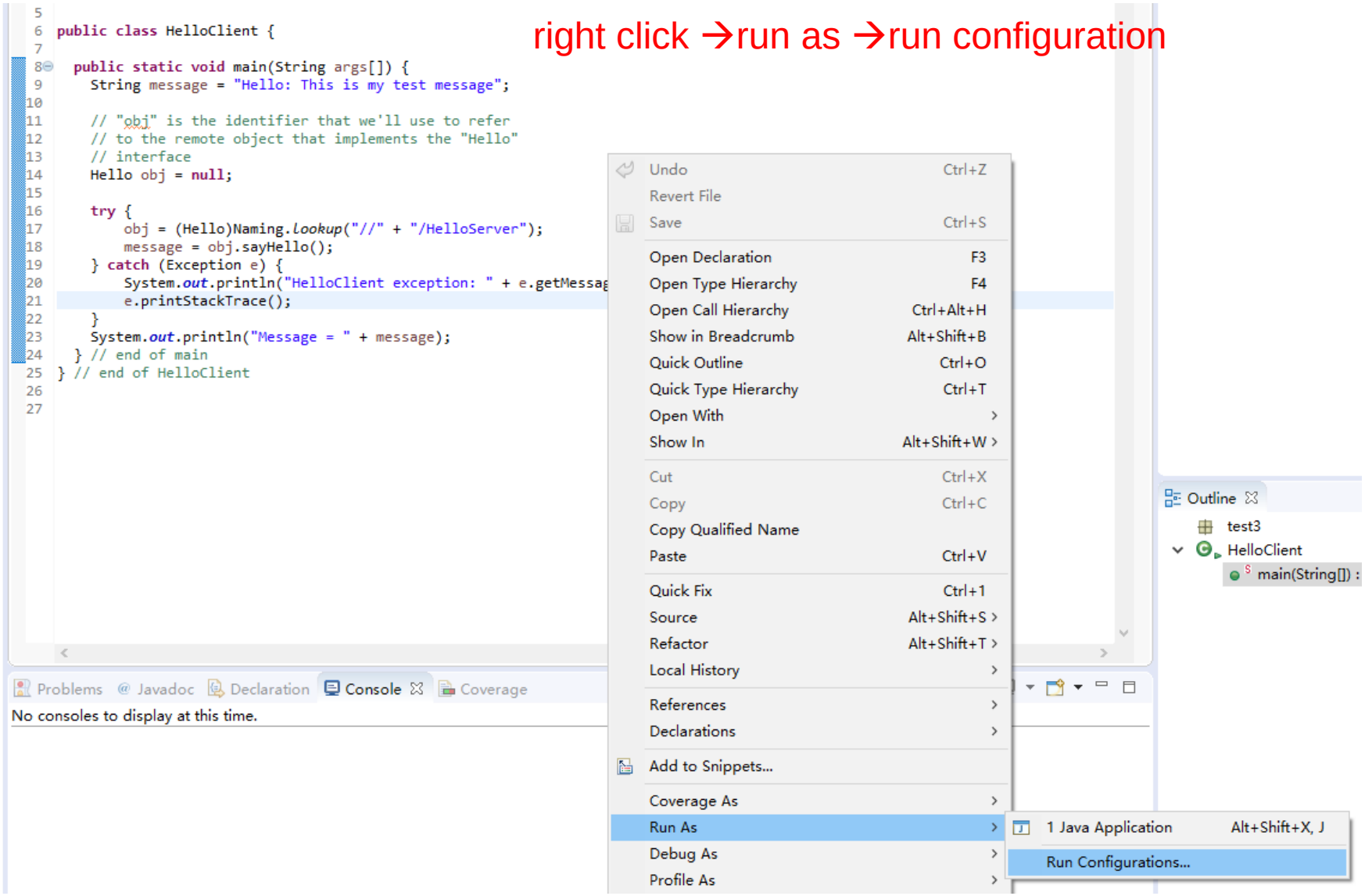
Running :

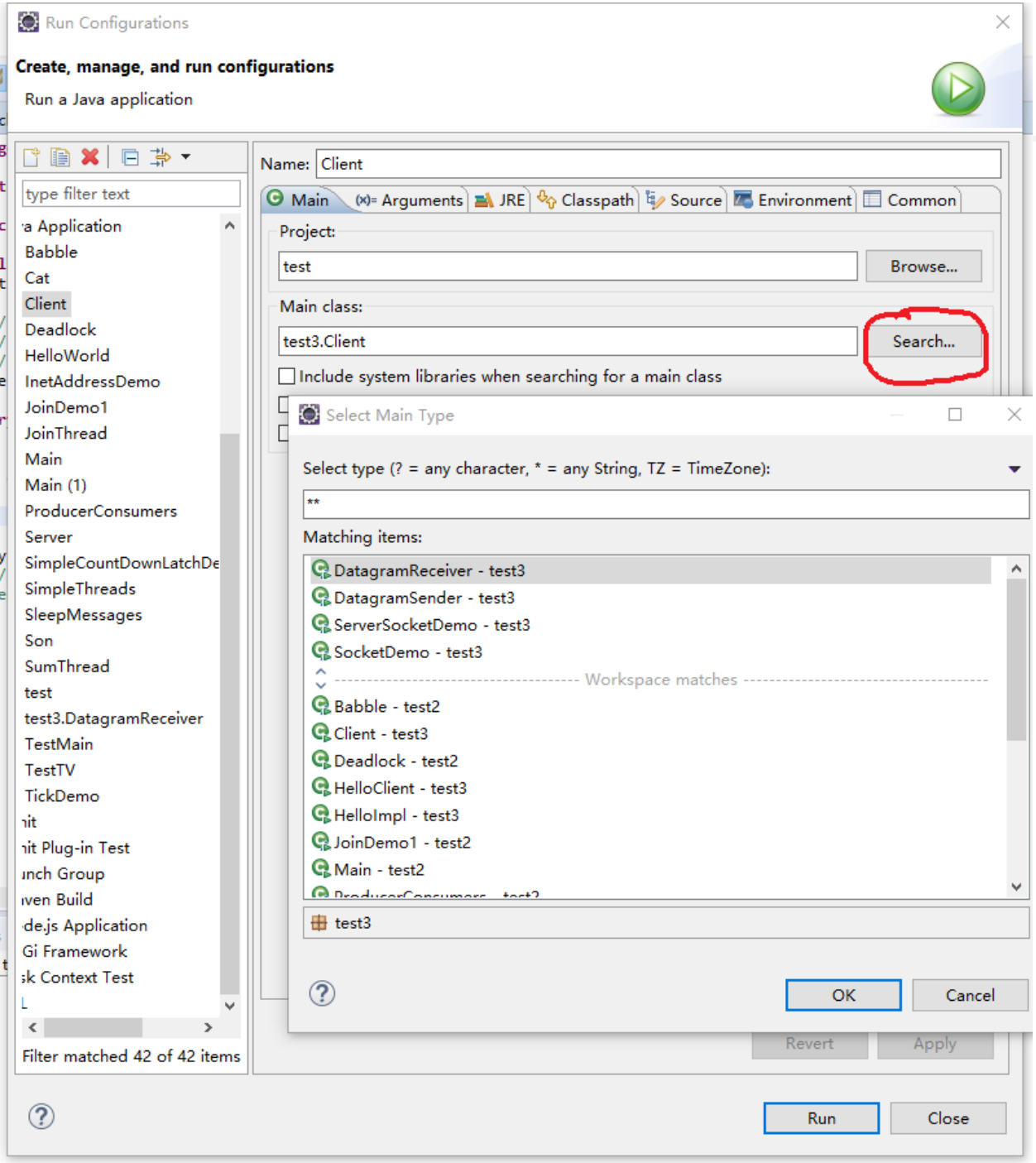
% java ServerSocketDemo 4321

% java SocketDemo 127.0.0.1 4321

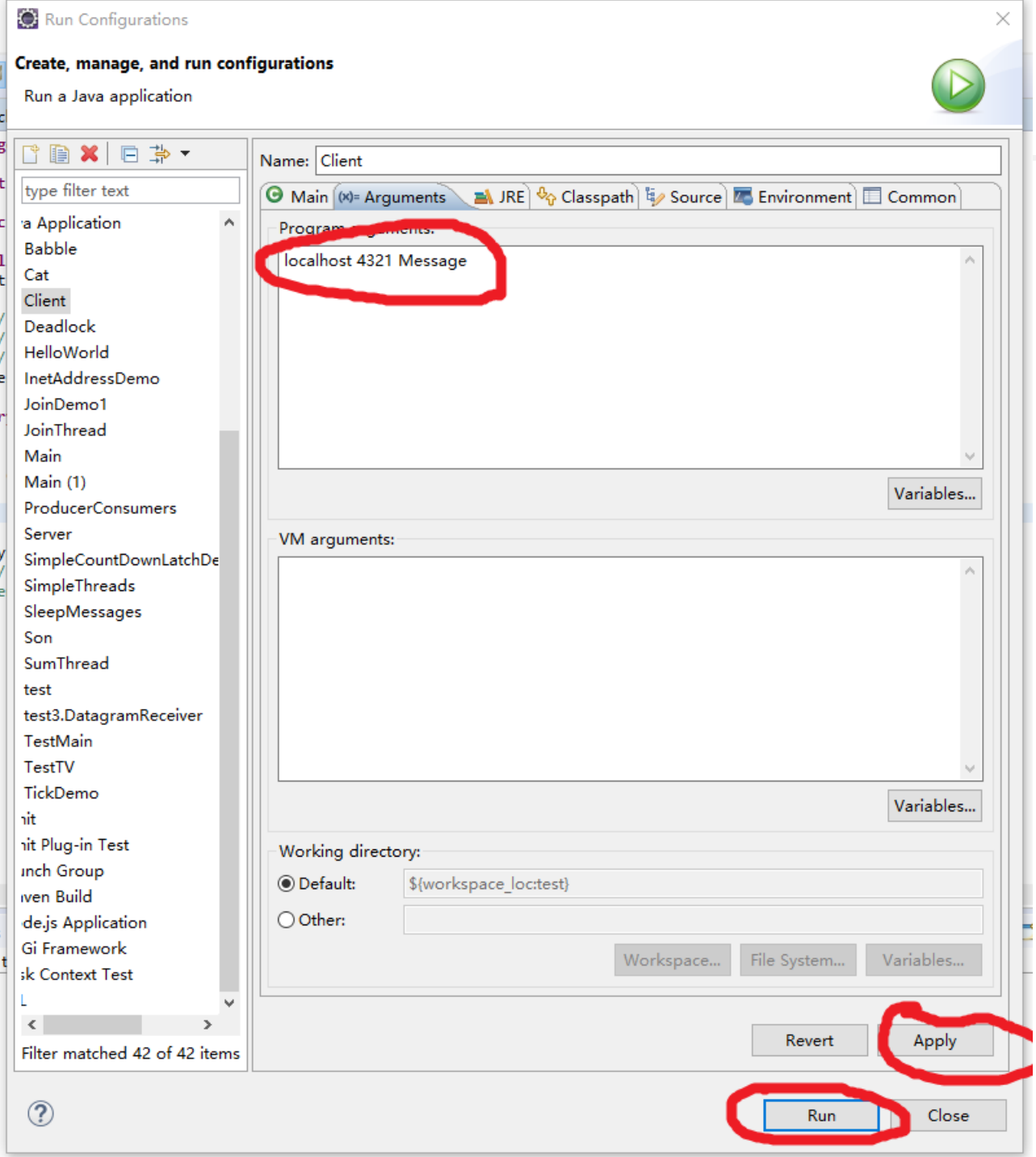
Run Configuration in Eclipse

right click →run as →run configuration





Search the program you want to run



Input parameters (space between parameters) →Apply →run

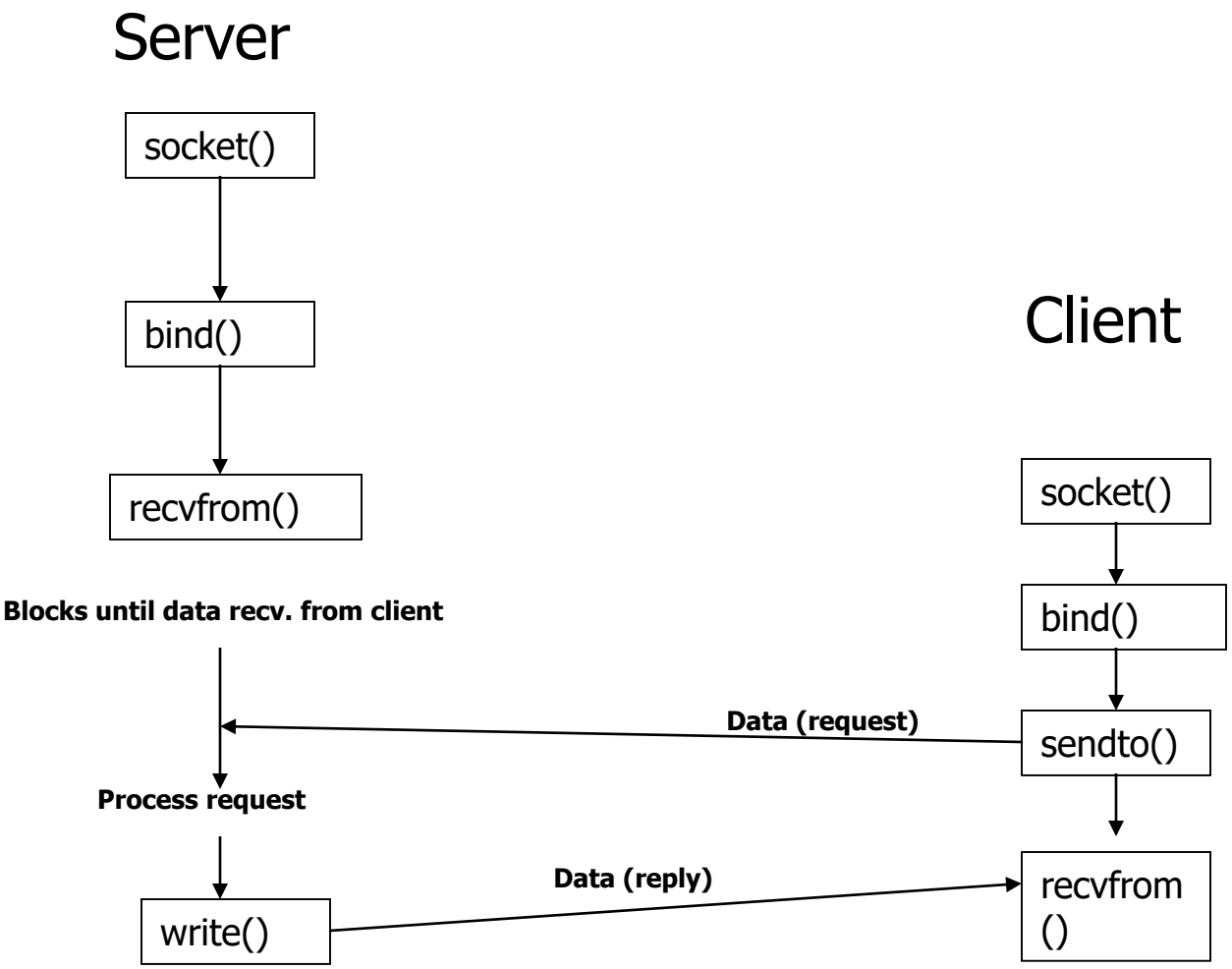
Switch Application

Example 2

```java
public class SocketServerExample {
 //static ServerSocket variable
 private static ServerSocket server;
 //socket server port on which it will listen
 private static int port = 9876;
 public static void main(String args[])
         throws IOException, ClassNotFoundException{
  //create the socket server object
  server = new ServerSocket(port);
  //keep listens indefinitely until receives 'exit' call or program terminates
  while(true){
   System.out.println("Waiting for the client request");
   //creating socket and waiting for client connection
   Socket socket = server.accept();
   //read from socket to ObjectInputStream object
   ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
   //convert ObjectInputStream object to String
   String message = (String) ois.readObject();
   System.out.println("Message Received: " + message);
   //create ObjectOutputStream object
   ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
   //write object to Socket
   oos.writeObject("Hi Client "+message);
   //close resources
   ois.close();
   oos.close();
   socket.close();
   //terminate the server if client sends exit request
   if(message.equalsIgnoreCase("exit")) break;
  }
  System.out.println("Shutting down Socket server!!");
  //close the ServerSocket object
  server.close();
 }
}
```

```java
public class SocketClientExample {
 public static void main(String[] args)
         throws UnknownHostException,IOException,
         ClassNotFoundException, InterruptedException{
  /*get the localhost IP address, if server
   is running on some other IP, you need to use that*/
  InetAddress host = InetAddress.getLocalHost();
  Socket socket = null;
  ObjectOutputStream oos = null;
  ObjectInputStream ois = null;
  for(int i=0; i<5;i++){
  //establish socket connection to server
  socket = new Socket(host.getHostName(), 9876);
  //write to socket using ObjectOutputStream
  oos = new ObjectOutputStream(socket.getOutputStream());
  System.out.println("Sending request to Socket Server");
  if(i==4)oos.writeObject("exit");
  else oos.writeObject(""+i);
  //read the server response message
  ois = new ObjectInputStream(socket.getInputStream());
  String message = (String) ois.readObject();
  System.out.println("Message: " + message);
  //close resources
  ois.close();
  oos.close();
  Thread.sleep(100);
  }
 }
}
```

Socket Call for Connectionless Protocol



Datagram Sockets and Packets

◆ **UDP does not guarantee reliable, sequenced data exchange, and therefore requires much less overhead**.

◆ **DatagramSocket() Method**

DatagramSocket() throws SocketException
DatagramSocket(int port) throws SocketException

```
DatagramSocket ds = new DatagramSocket(port);
```

◆ **DatagramPacket Constructor**

DatagramPacket(byte buffer[], int size)

DatagramPacket(byte buffer[], int size, InetAddress ia, int port)

```
DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
DatagramPacket dp = new DatagramPacket(buffer, buffer.length, ia, port);
```

◆ **receive() Method**

void receive(DatagramPacket dp) throws IOException

```
ds.receive(dp);
```

◆ **send() Method**

void send(DatagramPacket dp) throws IOException

```
ds.send(dp);
```

◆ **close() Method**

void close()

http://docs.oracle.com/javase/8/docs/api/java/net/DatagramPacket.html

Datagram Sockets and Packets

```
class DatagramReceiver {
  private final static int BUFSIZE = 20;
  public static void main(String args[]) {
    try {
      //Obtain port
      int port = Integer.parseInt(args[0]);

      //Create a DatagramSocket object for the port
      DatagramSocket ds = new DatagramSocket(port);

      //Create a buffer to hold incoming data
      byte buffer[] = new byte[BUFSIZE];

      //Create infinite loop
      while(true) {
        //Create a datagram packet
        DatagramPacket dp =
          new DatagramPacket(buffer, buffer.length);

        //Receive data
        ds.receive(dp);

        //Get data from the datagram packet
        String str = new String(dp.getData());

        // Display the data
        System.out.println(str);
      }
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```
class DatagramSender {
  public static void main(String args[]) {
    try  {
      // Create destination Internet address
      InetAddress ia =
        InetAddress.getByName(args[0]);
      // Obtain destination port
      int port = Integer.parseInt(args[1]);

      // Create a datagram socket
      DatagramSocket ds = new DatagramSocket();

      //Create a datagram packet
      byte buffer[] = args[2].getBytes();
      DatagramPacket dp =
        new DatagramPacket(buffer, buffer.length,
          ia, port);
      // Send the datagram packet
      ds.send(dp);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Running :

% java DatagramReceiver 4321

% java DatagramSender localhost 4321 Message

Example 2

```
public class QuoteServerThread extends Thread {
 protected DatagramSocket socket = null;
 protected BufferedReader in = null;
 protected boolean moreQuotes = true;

 public QuoteServerThread() throws IOException {
  this("QuoteServerThread");
 }

 public QuoteServerThread(String name) throws IOException {
  super(name);
  socket = new DatagramSocket(4445);

  try {
   in = new BufferedReader(new FileReader("one-liners.txt"));
  } catch (FileNotFoundException e) {
   System.err.println("Could not open file. Serving time instead.");
  }
 }
public void run() {
  while (moreQuotes) {
   try {
    byte[] buf = new byte[256];
    // receive request
    DatagramPacket packet = new DatagramPacket(buf, buf.length);
    socket.receive(packet);
    // figure out response
    String dString = null;
    if (in == null)
     dString = new Date().toString();
    else
     dString = getNextQuote();

    buf = dString.getBytes();
    // send the response to the client at "address" and "port"
    buf = dString.getBytes();
    // send the response to the client at "address" and "port"
    InetAddress address = packet.getAddress();
    int port = packet.getPort();
    packet = new DatagramPacket(buf, buf.length, address, port);
    socket.send(packet);
   } catch (IOException e) {
    e.printStackTr ace();
    moreQuotes = false;
   }
  }
  socket.close();
 }

    protected String getNextQuote() {
        String returnValue = null;
        try {
            if ((returnValue = in.readLine()) == null) {
                in.close();
                moreQuotes = false;
                returnValue = "No more quotes. Goodbye.";
            }
        } catch (IOException e) {
            returnValue = "IOException occurred in server.";
        }
        return returnValue;
    }

    public static void main(String[] args) throws IOException {
        new QuoteServerThread().start();
    }
}
```

Example 2

```
public class QuoteClient {
  public static void main(String[] args) throws IOException {
    if (args.length != 1) {
      System.out.println("Usage: java QuoteClient <hostname>");
      return;
    }
    // get a datagram socket
    DatagramSocket socket = new DatagramSocket();
    // send request
    byte[] buf = new byte[256];
    InetAddress address = InetAddress.getByName(args[0]);
    DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 4445);
    socket.send(packet);
    // get response
    packet = new DatagramPacket(buf, buf.length);
    socket.receive(packet);
    // display response
    String received = new String(packet.getData(), 0, packet.getLength());
    System.out.println("Quote of the Moment: " + received);
    socket.close();
  }
}
```

Client and Server Application

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

public class Server
{
  public ServerSocket svrSocket = null;
  public Socket socket = null;
  public InputStream inputStream = null;
  public OutputStream outputStream = null;
  public DataInputStream dataStream = null;
  public PrintStream printStream = null;
  public DataOutputStream dataoutputStream = null;
  public String message;
  public BufferedReader charStream = new
BufferedReader(new InputStreamReader(System.in));

  public Server() {
    try {
      svrSocket = new ServerSocket(1056);
      System.out.println("\nInitializint Port...");
      System.out.println("\nListen...");
      socket = svrSocket.accept();
      System.out.println("\nConnect to Client!\n");
      inputStream = socket.getInputStream();
      dataStream = new DataInputStream(inputStream);
      outputStream = socket.getOutputStream();
      dataoutputStream = new
DataOutputStream(outputStream);

      message = dataStream.readUTF();
      System.out.println(message + "\n");
    }  catch( UnknownHostException e) {
       System.out.println("Error : Cannot find server." + e);
    }
    catch( IOException e ) {
      System.out.println("Error : I/O Error." + e);
    }
  }
```

```
public void readSocket(){
    try {
      message = dataStream.readUTF();
      System.out.println(message + "\n");
      if(message.equals("Exit")){
        System.exit(0);
                                                  }
    }
    catch( UnknownHostException e) {
      System.out.println("Error : Cannot find server." + e);
    }
    catch( IOException e ) {
      System.out.println("Error : I/O Error." + e);
    }
  }

  public void writeSocket(){
    try {
      String initmsg_r = new String("Enter your message: ");
      dataoutputStream.writeUTF(initmsg_r);
      System.out.print("Enter please for ready... ");
      message = charStream.readLine();
      if  (! Message.equals("Exit")) return;
      else {dataoutputStream.writeUTF("Exit");
              System.exit(0); }
    }
    catch( UnknownHostException e) {
      System.out.println("Error : Cannot find server." + e);
    }
    catch( IOException e ) {
      System.out.println("Error : I/O Error." + e);
    }
  }
```

Client and Server Application

```
  public static void main(String args[]) {
    Server svr = new Server();
    for(;;){
      svr.writeSocket();
      svr.readSocket();
    }
  }
}
// End of Server
```

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

public class Client {

  public static void main(String args[]) {
  // Initialize the stream
  OutputStream outputStream = null;
  DataOutputStream dataoutputStream = null;
  InputStream inputStream = null;
  DataInputStream dataStream = null;
  BufferedReader charStream = null;

  // Initialize Socket
  Socket socket = null;
  String message;

  try {
      charStream = new BufferedReader(new
InputStreamReader(System.in));
 message = new String("Hi! I am a client");
      socket = new Socket("127.0.0.1", 1056);
```

```
dataStream = new DataInputStream(inputStream);
      outputStream = socket.getOutputStream();
      dataoutputStream = new
DataOutputStream(outputStream);
      dataoutputStream.writeUTF(message);
   } catch(UnknownHostException e) {
       System.out.println("Error : Cannot find server." + e);
     }
     catch(IOException e) {
       System.out.println("Error : I/O Error." + e);
     }

 while(true) {
   try {
      inputStream = socket.getInputStream();
      dataStream = new DataInputStream(inputStream);
      message = dataStream.readUTF();
      System.out.print(message);
      if(message.equals("Exit")){ System.exit(0); }
        message = charStream.readLine();
        dataoutputStream.writeUTF(message);
    } catch(UnknownHostException e) {
        System.out.println("Error : Cannot find server." + e);
      }
      catch(IOException e) {
        System.out.println("Error : I/O Error." + e);
      }
  } // end of while
 } // end of main method
} // end of Client Constructor
```

Exercise 1: Echo Program

Echo program using socket:

1) client reads line from standard input (**`inFromUser`** stream) , and sends to server via socket (**`outToServer`** stream)
2) server reads line from socket
3) server converts line to uppercase, sends back to client
4) client readsfrom socket (**`inFromServer`** stream) , and show the modified line through (**outToUser** stream)