

---

# Java Network (Socket)

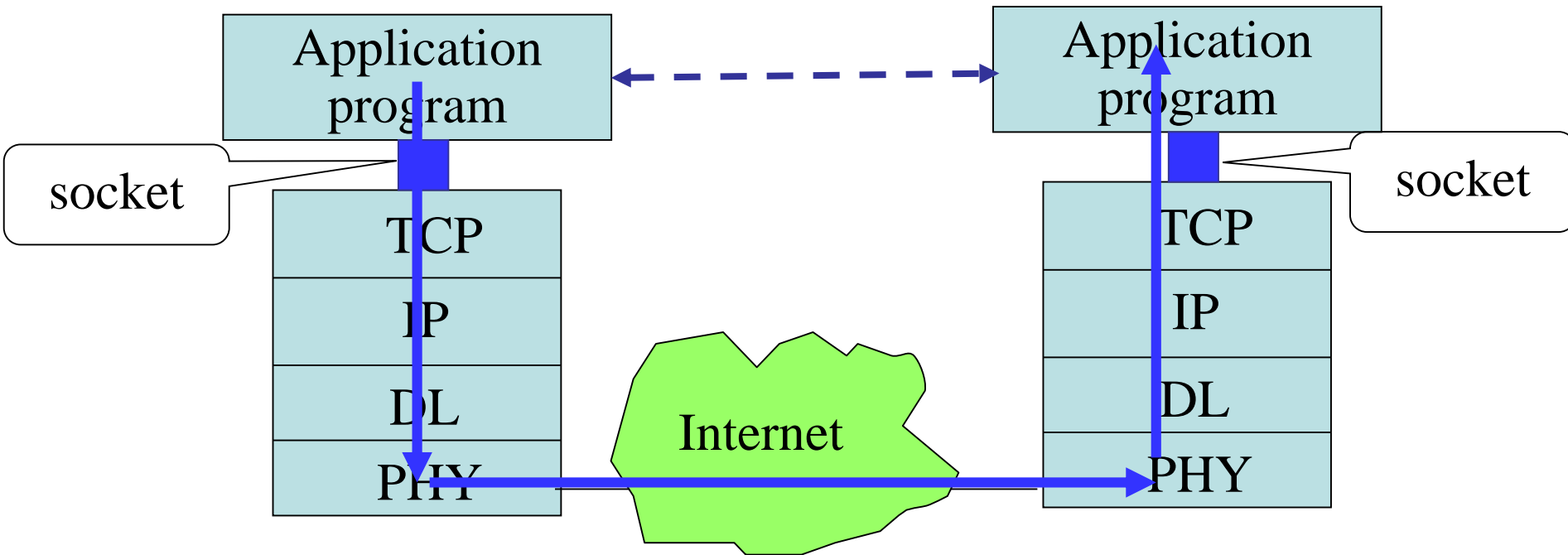
# Contents

---

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

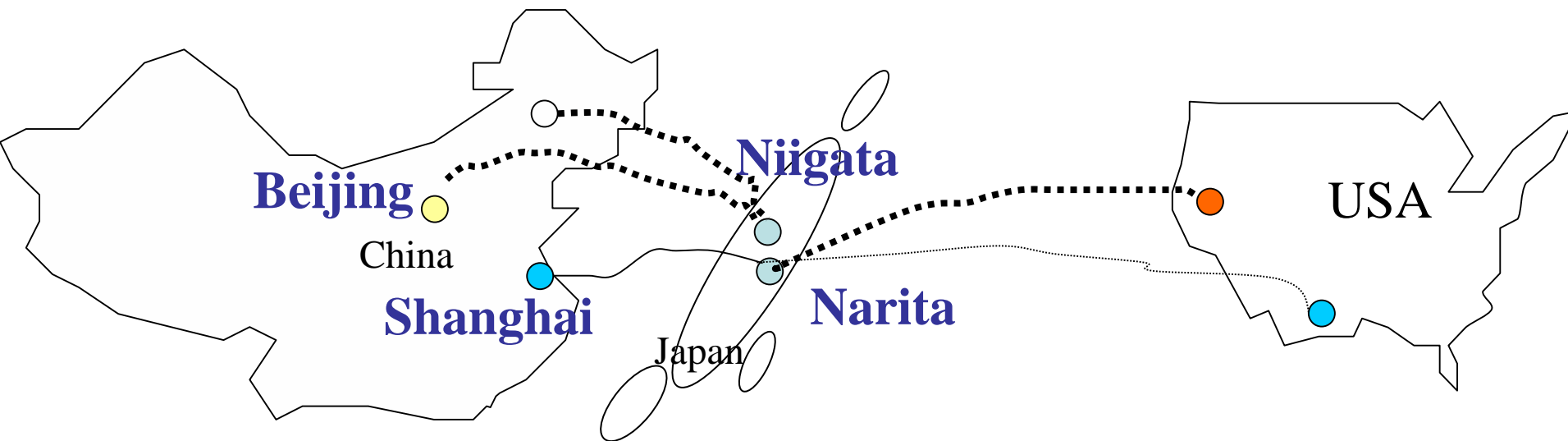
# Socket: Interface for Application Layer

## Echo Program

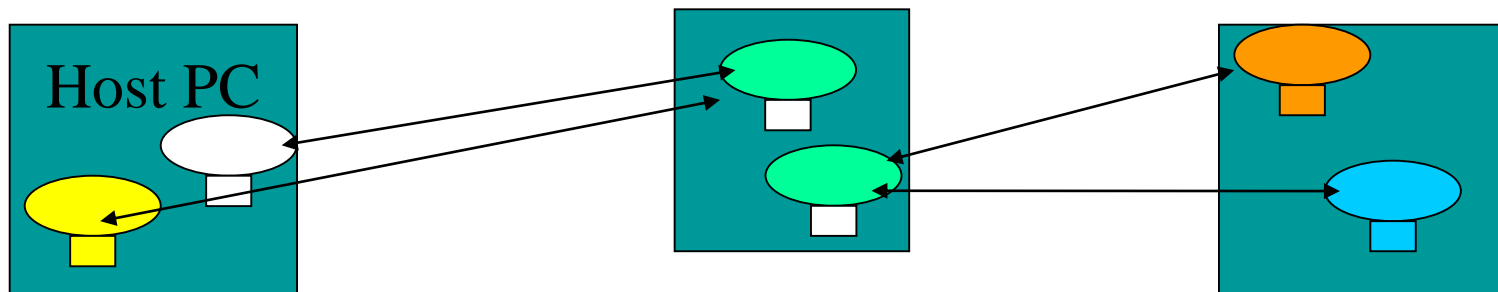


# Socket Address: IP and Port

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



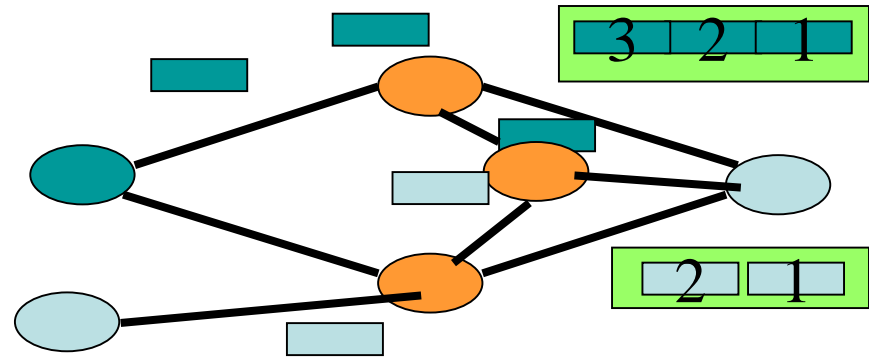
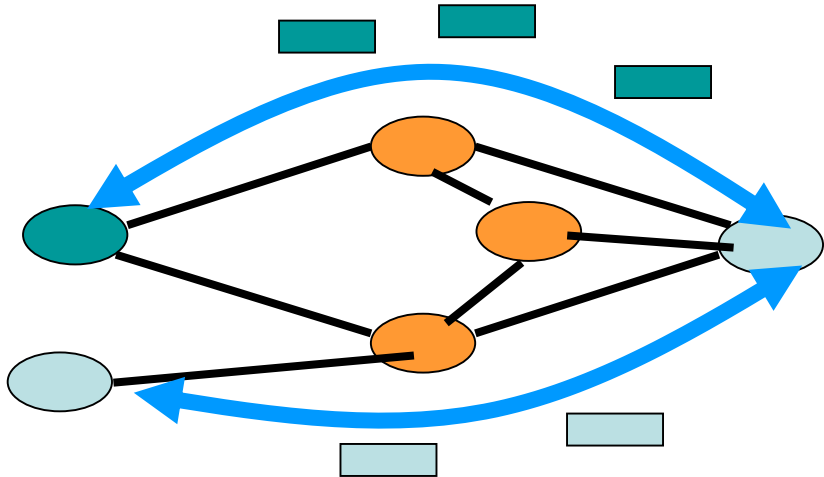
**communication: A host Q port to B host R port**      **1 port: N ports**



# Two types of Data Transportation Services

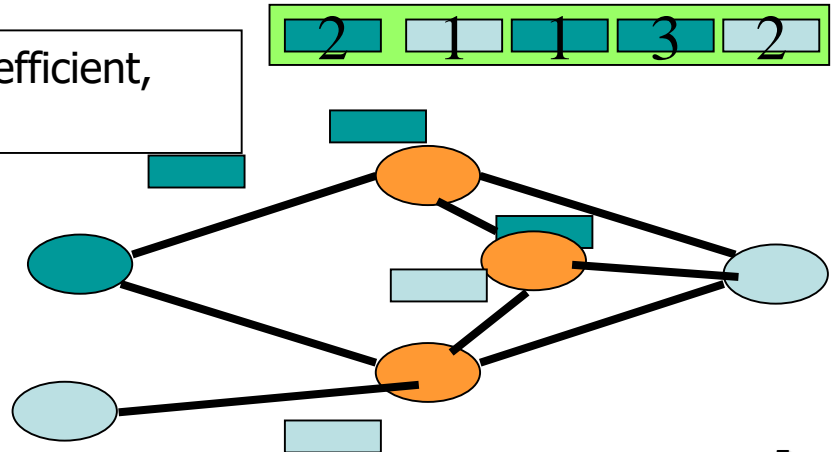
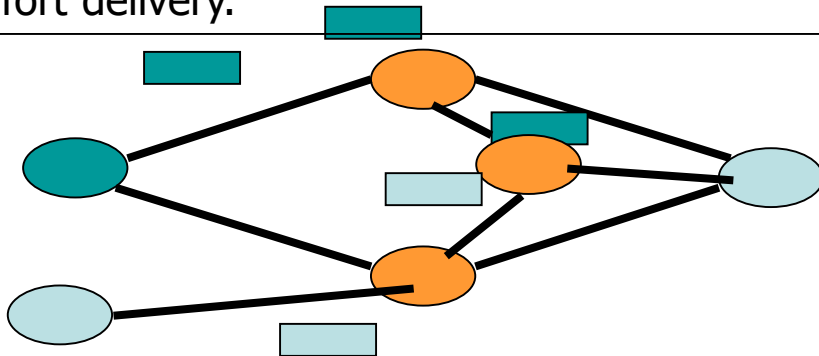
## 1. Connection oriented

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

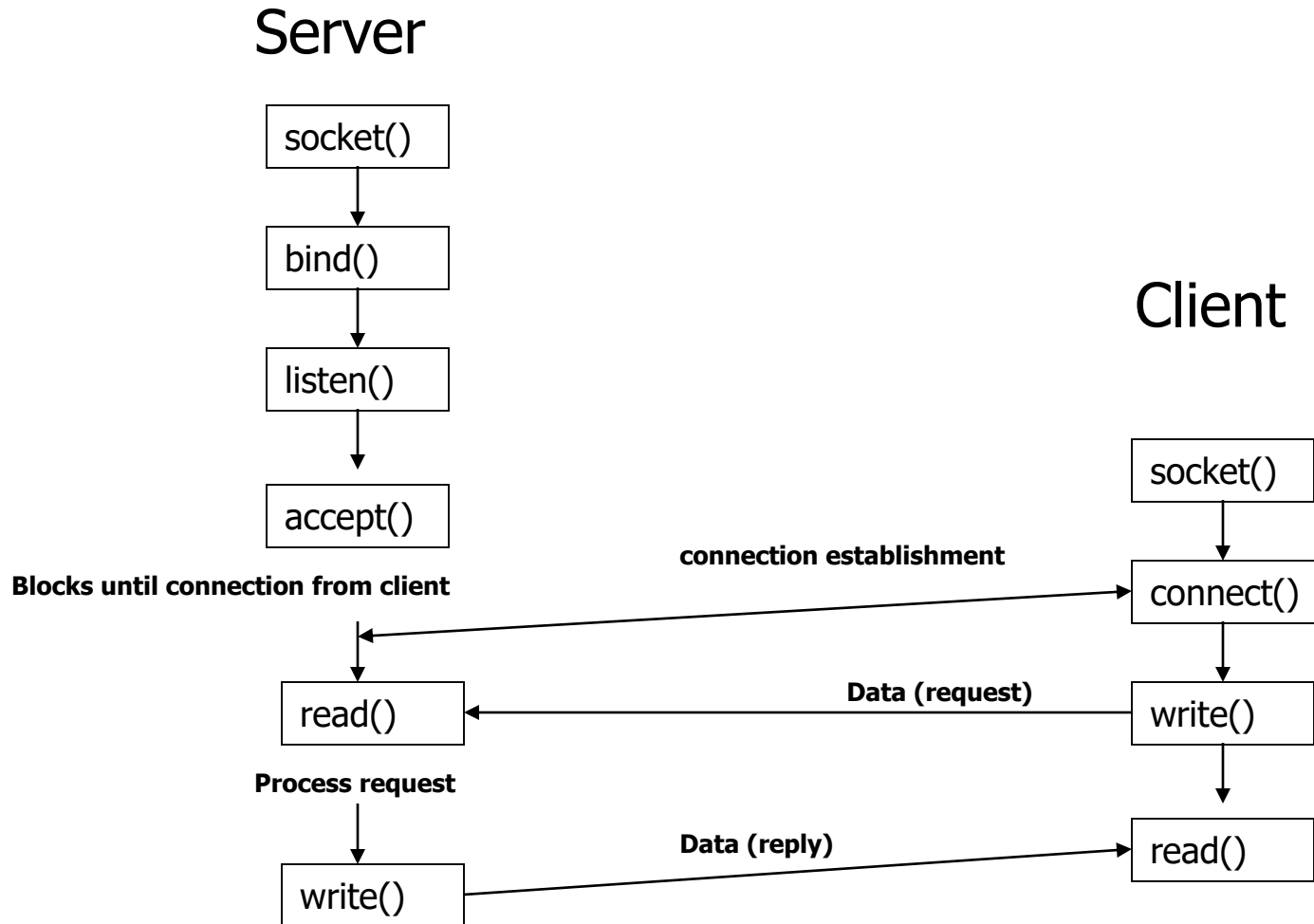


## 2. Connectionless

◆ 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 socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.

*ServerSocket(int port) throws IOException*

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

## ◆ **accept() Method**

*Socket accept() throws IOException*

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

## ◆ **close() Method**

*void close() throws IOException*

- Closes this socket.

## ◆ **Socket Class**

This class implements client sockets (also called just "sockets"). A socket is an endpoint for communication 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.

## ◆ **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.

## ◆ **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);
        }
    }
}
```

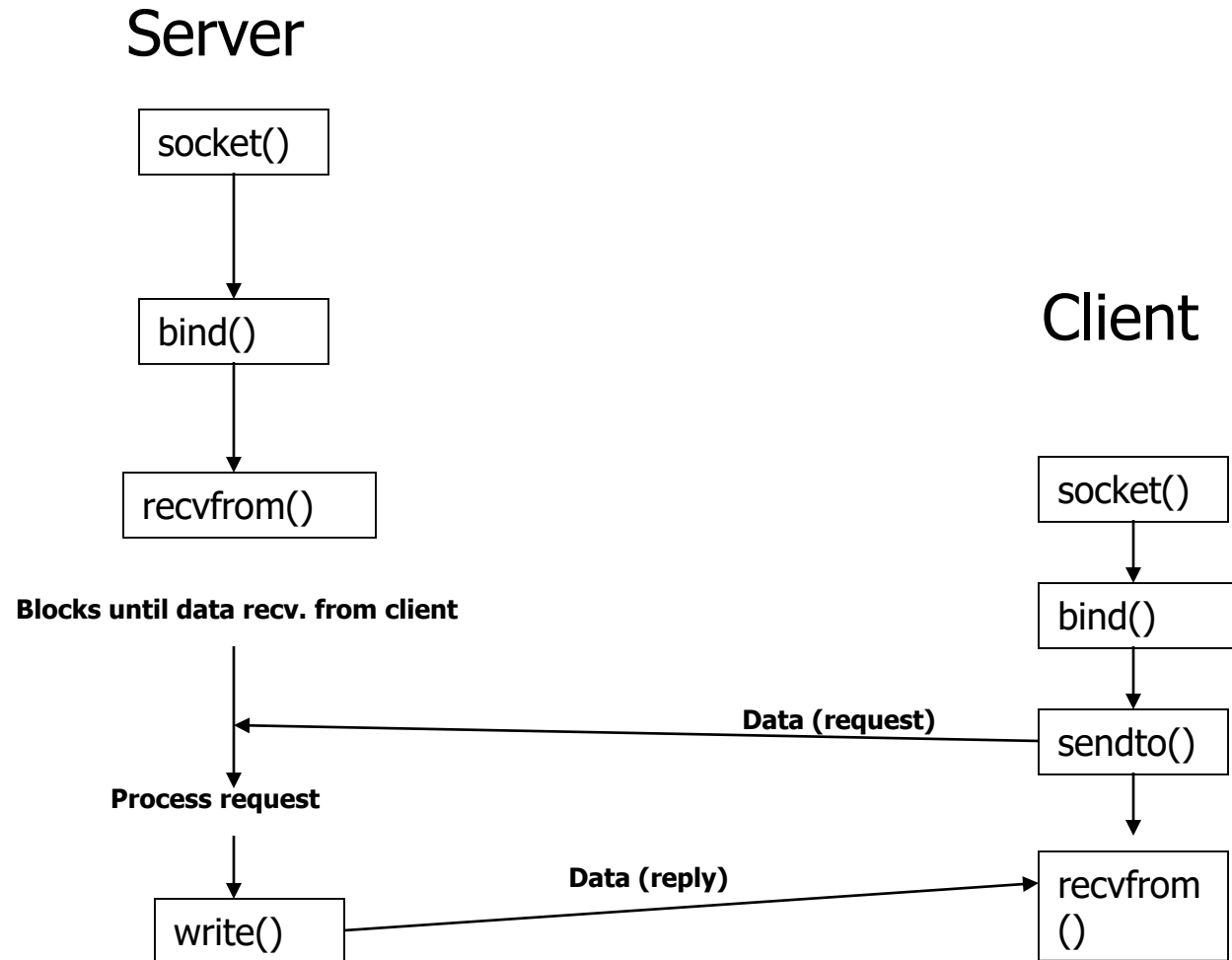
Running :

```
% java ServerSocketDemo 4321
```

```
% java SocketDemo 127.0.0.1 4321
```



# 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*

## ◆ **send() Method**

*void send(DatagramPacket dp) throws IOException*

## ◆ **DatagramPacket Constructor**

*DatagramPacket(byte buffer[], int size)*

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

## ◆ **close() Method**

*void close()*

## ◆ **receive() Method**

*void receive(DatagramPacket dp) throws IOException*

<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
```

# 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
```

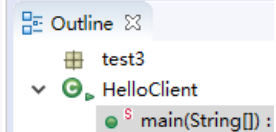
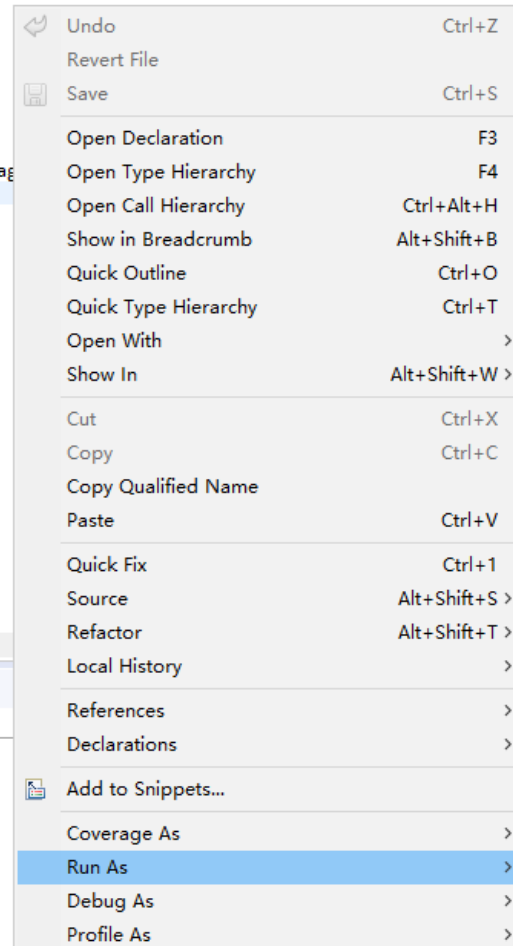
```
import java.net.*;  
import java.io.*;  
  
public class Client {  
  
    public static void main(String args[]) {  
        // Initialize the stream  
        OutputStream outputStream = null;  
        DataOutputStream dataOutputStream = null;  
        InputStream inputStream = null;  
        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);
```

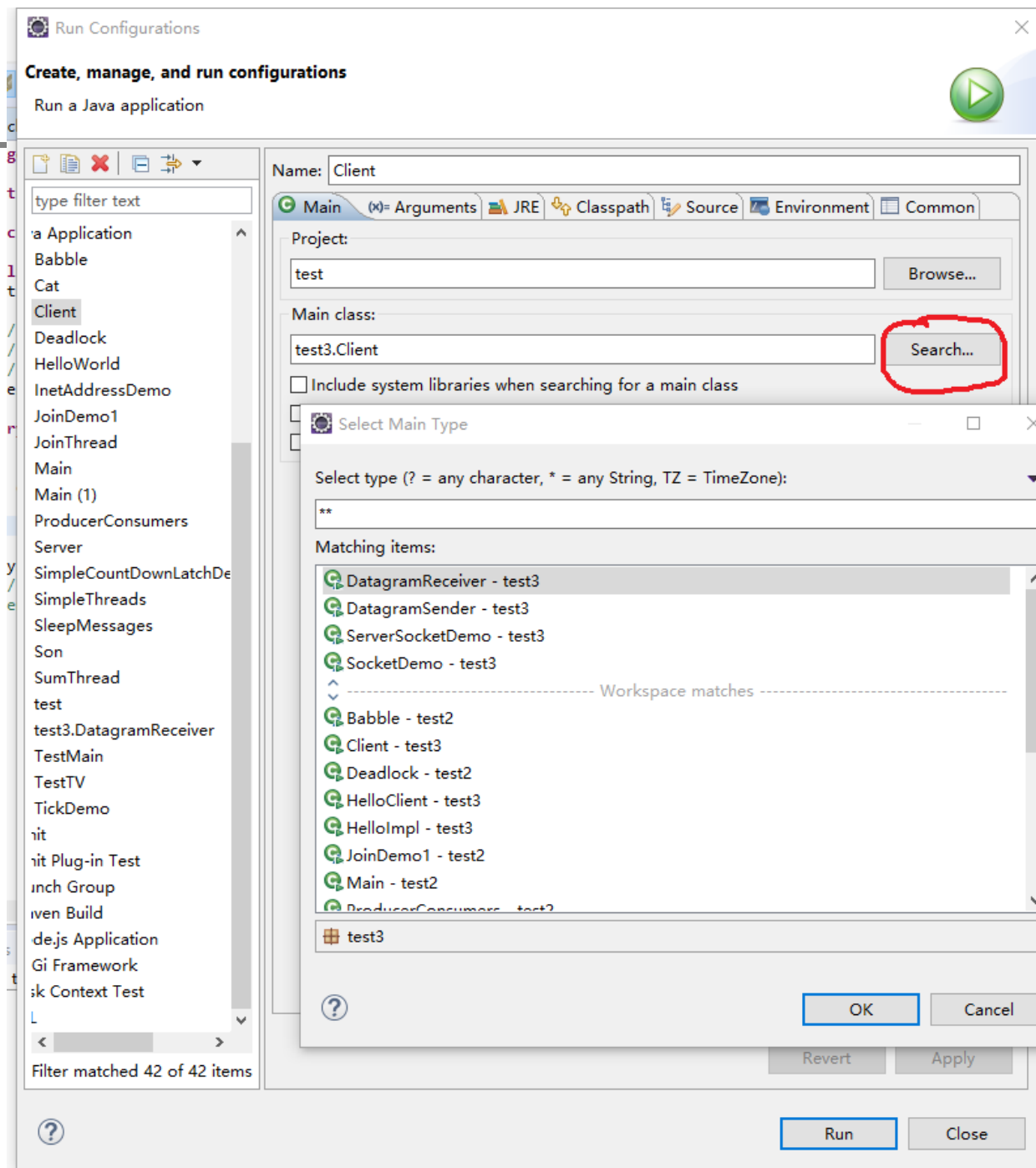
```
                dataStream = new DataInputStream(inputStream);  
                outputStream = socket.getOutputStream();  
                dataOutputStream = new  
                DataOutputStream(outputStream);  
                dataOutputStream.writeUTF(message);  
            } catch(UnknownHostException 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
```

# Run Configuration in Eclipse

right click → run as → run configuration

```
5
6 public class HelloClient {
7
8     public static void main(String args[]) {
9         String message = "Hello: This is my test message";
10
11         // "obj" is the identifier that we'll use to refer
12         // to the remote object that implements the "Hello"
13         // interface
14         Hello obj = null;
15
16         try {
17             obj = (Hello)Naming.lookup("//" + "/HelloServer");
18             message = obj.sayHello();
19         } catch (Exception e) {
20             System.out.println("HelloClient exception: " + e.getMessage());
21             e.printStackTrace();
22         }
23         System.out.println("Message = " + message);
24     } // end of main
25 } // end of HelloClient
26
27
```





Search the program  
you want to run

## Create, manage, and run configurations

Run a Java application



type filter text

Application  
Babble  
Cat  
Client  
Deadlock  
HelloWorld  
InetAddressDemo  
JoinDemo1  
JoinThread  
Main  
Main (1)  
ProducerConsumers  
Server  
SimpleCountDownLatch  
SimpleThreads  
SleepMessages  
Son  
SumThread  
test  
test3.DatagramReceiver  
TestMain  
TestTV  
TickDemo  
uit  
uit Plug-in Test  
inch Group  
iven Build  
de.js Application  
Gi Framework  
sk Context Test

Filter matched 42 of 42 items

Name: Client

Main (X) Arguments JRE Classpath Source Environment Common

Program arguments:

localhost 4321 Message

Variables...

VM arguments:

Variables...

Working directory:

☒ Default: \${workspace\_loc:test}☐ Other:

Workspace...

File System...

Variables...

Revert

Apply

Run

Close

Input parameters (space  
between parameters)  
→ Apply → run



```
47     } catch (IOException e) {  
48         System.out.println("Error : I/O Error." + e);  
49     }  
50 } // end of while  
51 } // end of main method  
52 } // end of Client Constructor  
53
```

Problems @ Javadoc Declaration Console Coverage

Server [Java Application] C:\Program Files\Java\jre1.8.0\_144\bin\javaw.exe (2017年12月1日 上午10:27:11)

Initializint Port...

Listen...

1 Server [Java Application] C:\Program Files\Java\jre1.8.0\_144\bin\javaw.exe (2017年12月1日 上午10:27:11)  
2 Client [Java Application] C:\Program Files\Java\jre1.8.0\_144\bin\javaw.exe (2017年12月1日 上午10:27:11)

test3  
Client  
S ma