# Java Sockets

COMP476
Networked Computer Systems

## Class java.net.Socket

- This class implements TCP client sockets (also called just "sockets"). A socket is an endpoint for communication between two machines.
- Sockets create streams that can be used exactly like file streams.
- Reading and writing to a socket is identical to reading and writing to a file.

## Socket Constructor

```
public Socket(String host, int port)
        throws UnknownHostException,
               IOException
```

- Creates a stream socket and connects it to the specified port number on the named host.
- Other constructors are available.

## Methods

```
public InputStream getInputStream(
            throws IOException
```

- Returns an input stream for this socket.

```
public OutputStream getOutputStream()
            throws IOException
```

- Returns an output stream for this socket.

## java.net.ServerSocket

- Creates a TCP socket for use by a server program.
- A server socket waits for requests to come in over the network.

## ServerSocket Constructor

```
public ServerSocket(int port)
            throws IOException
```

- Creates a server socket on a specified port.
- A port of 0 creates a socket on any free port.

## ServerSocket Methods

```
public Socket accept()
            throws IOException
```

- Listens for a connection to be made to this socket and accepts it.
- The method blocks until a connection is made.
- Returns a new Socket for communicating with the client.

## UDP Sockets

- **DatagramSocket** is a class to create a Java socket that uses UDP.
- **DatagramSocket** objects send and receive objects of the **DatagramPacket** class.
- A **DatagramPacket** object contains the data transmitted along with the address of the sender or destination.

## Java Security

- Java applets running in a browser can only connect to the server that hosts the applet.
- Java applications generally have no restrictions.
- Experience shows that network programming is easiest on a PC with Java.

## Sample Java Client

```
public class Tclient {
   final static String serverIPname = "whatever.ncat.edu";
   final static int      serverPort   = 3456;
  public static void main(String[] args) {
      java.net.Socket                sock = null;
      java.io.PrintWriter            pw   = null;
      java.io.BufferedReader         br   = null;
```

```
try {
      sock = new java.net.Socket(serverIPname,serverPort);
      pw   = new java.io.PrintWriter(sock.getOutputStream(),true);
      br   = new java.io.BufferedReader(new
                java.io.InputStreamReader(sock.getInputStream()));
      pw.println("Message from the client");
      String answer = br.readLine();
      System.out.println("Response from the server >" + answer);
      pw.close();
      br.close();
      sock.close();
} catch (Exception e) {
```

## Sample Java Server

```
public class Tserver {
   final static int serverPort   = 3456;
   public static void main(String[] args) {
      java.net.ServerSocket   sock = null;
      java.net.Socket         clientSocket = null;
      java.io.PrintWriter     pw   = null;
      java.io.BufferedReader  br   = null;
```

```
try {
    sock = new java.net.ServerSocket(serverPort);
    clientSocket = sock.accept();
    pw   = new java.io.PrintWriter(
                              clientSocket.getOutputStream());
    br   = new java.io.BufferedReader(new
          java.io.InputStreamReader( clientSocket.getInputStream()));
    String msg = br.readLine();
    System.out.println("Message from the client >" + msg);
    pw.println("Got it!");
    pw.flush();
    pw.close();
    br.close();
    clientSocket.close();
    sock.close();
} catch (Throwable e) {
```