B.Sc. In Internet Systems Development.
Concurrent Programming.
Networking.

LIMERICK INSTITUTE
OF TECHNOLOGY
SCHOOL OF SCIENCE,
ENGINEERING & I.T.
Department of Information Technology

Networking – The Basics

- Networking is tightly integrated in Java. Socket based communication is provided that enables programs to communicate through designated sockets.

- A socket is an abstraction that facilitates communication between a client and a server.

- Java treats sockets communication much as it treats I/O operations; thus programs can read from and write to sockets as easily as they can read from or write to files.

Networking – The Basics

- Java supports stream sockets and datagram sockets.
- Stream sockets use TCP (Transmission Control Protocol) for data transmission, whereas datagram sockets use UDP (User Datagram Protocol).
- Since TCP can detect lost transmission and resubmit them, transmissions are lossless and reliable.
- On the other hand UDP cannot guarantee lossless transmission.
- Therefore, stream sockets are used in most areas of Java programming.

Networking – The Basics

- Networking programming usually involves a server and one or more clients.
- The client sends requests to the server and the server then responds to these requests.
- The client begins by attempting to establish a connection to the server.
- The server can accept or deny the connection.
- Once a connection is established, the client and the server communicate through sockets.
- The server MUST be running before a client makes a request.

Networking – The Basics

- Once started, the server then waits for a client to make a request.
- To establish a server, you need to create a server socket and attach it to a port.
  - This is where the server listens for communications.
- The port identifies the TCP service on the socket.
- Port numbers between 0 – 1023 are reserved for privileged processes.
  - EMail is on 25.
  - Web Servers are on 80.

Networking – The Basics

- You can choose any port number that is not currently used by any other process.
- Attempting to create a server socket on a port already in use would cause the `java.set.BindException`
- The following statement creates a server socket s.

```
ServerSocket s = new
                 ServerSocket(port)
```

- After a server socket is created the server can use the following statement to listen for connections.

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

Networking – The Basics

- This statement waits until a client connects to the server socket.
- The client issues the following statement to request a connection to a server.

```
Socket connectToServer =
    new Socket(ServerName, port);
```

- This statement opens a socket so that the client program can communicate with the server.
- `ServerName` is the servers Internet host name or IP address.

Networking – The Basics

- The following statement creates a socket at port 8000 on the clients machine to connect to the host lit.ie:

```
 Socket connectToServer =

      new Socket(“lit.ie”, 8000);
```

- Alternatively, you can use the IP address to create a socket as follows:

```
 Socket connectToServer =

      new Socket(“192.1.88.11”, 8000);
```

Networking – The Basics

- An IP address, consisting of four dotted decimal numbers between 0 and 255, such as 192.1.88.11 is a computers unique identity on a network.
- Since these numbers are not that easy to remember, they are often mapped to more meaningful names called host names such as lit.ie.
- A program can use the name `localhost` or the IP address 127.0.0.1 to refer to the machine on which a client is running.
- After the server accepts the connection, communications between server and client is conducted the same as for I/O streams.

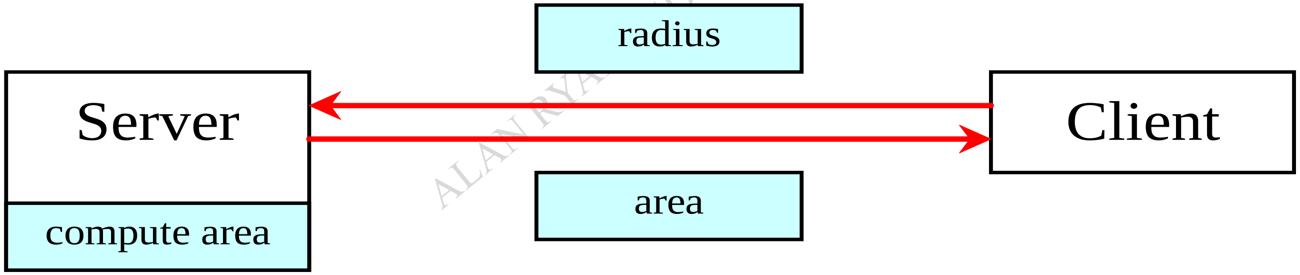
Coding the Client and the Server

- To get an input stream and an input stream, use the getInputStream() and getOutputStream() methods on a socket object.

```
InputStream isFromServer =
      connectToServer.getInputStream();


OutputStream osToServer =
      connectToServer.getOutputStream();
```

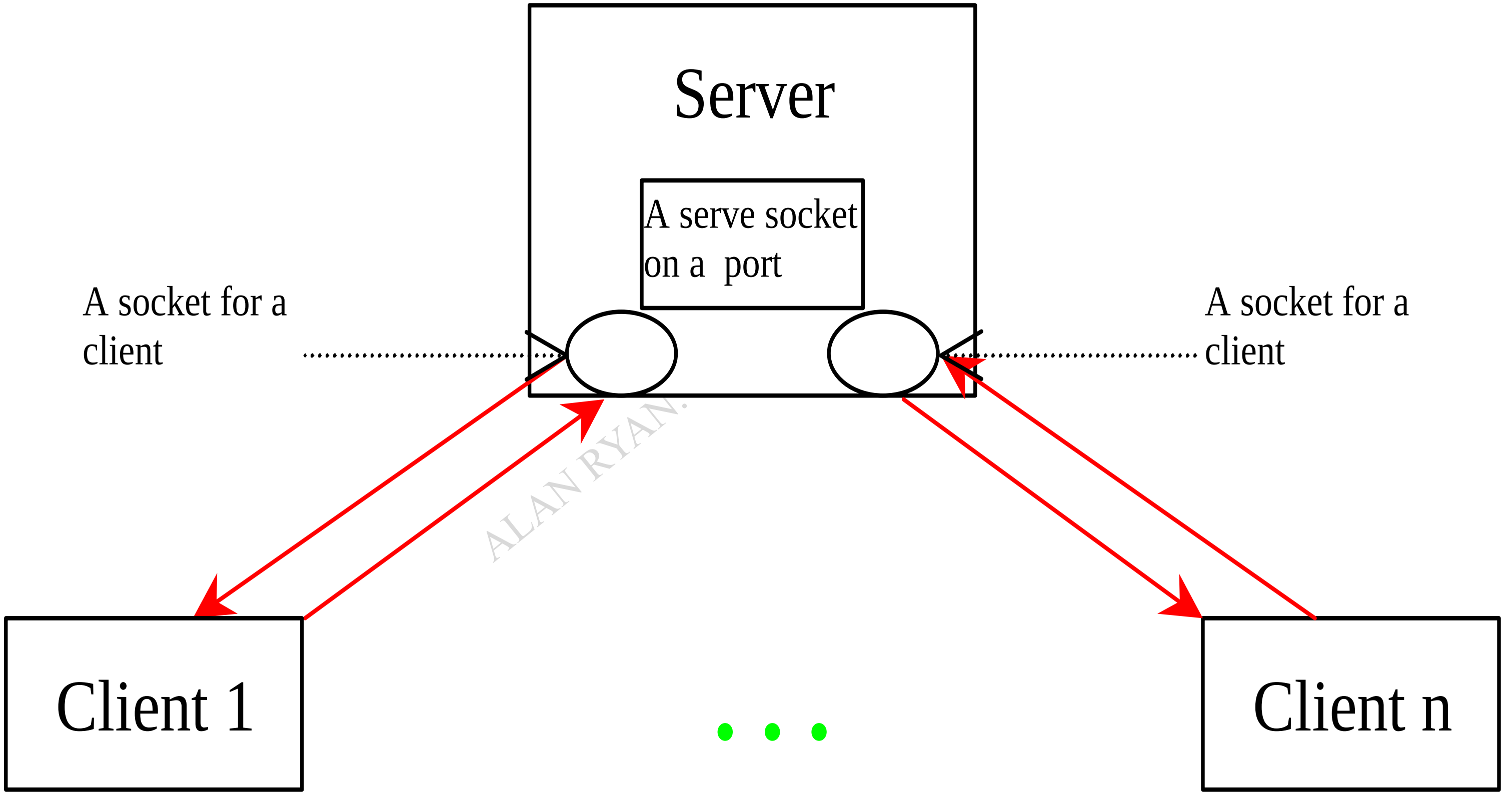
Example 1 – Simple Client/Server App

- **Objective**: Write a client to send data to a server. The server receives the data, uses it to produce a result, and then sends the result back to the client. The client then displays the result. In this example, the data sent from the client is the radius of a circle, and the result produced by server is the area of the circle.



- Consult CircleClient.java and CircleServer.java. These classes are in a package called Example1.

Example 2 – Serving Multiple Clients



Example 2 – Serving Multiple Clients

- Multiple clients are quite often connected to a single server at the same time.
- Typically, a server runs constantly on a server computer and clients from all over the Internet can connect to it.
- You can use threads to handle the servers multiple clients simultaneously.
- Simply create a thread for each connection.
- The following code is how the server handles the establishment of a connection.
- Consult MultiThreadServer.java and CircleClient.java. These classes are in a package called Example2.

Example 3 – Passing Objects Over a Network

- **Objective:** Develop an application that creates a Student object on the client side. The client will then pass this object to the (single-threaded) server, where the objects state will be made persistent (saved to a file).

- Consult the following code:

  1. RegistrationClientUsingObjectStream.java
  2. RegistrationServerUsingObjectStream
  3. Student.java

- All these classes are contained within a package called "Example3_Passing_Objects".

Programming With Datagram Sockets

- Clients and servers that communicate via stream sockets have a dedicated point-to-point channel between them.
  - Stream sockets communicate using TCP as its very reliable.
- Clients and servers that communicate using datagram's do not have a dedicated point-to-point channel.
  - Data is transmitted using packets.
  - Datagram’s use UDP to communicate but its not as reliable as TCP.
- Where a dedicated point-to-point connection is not required UDP is more efficient than TCP.

Programming With Datagram Sockets

- Stream sockets communicate using TCP as its very reliable. The java.net package contains two classes which help you develop Java programs that use datagram's to send and receive packets over the network.

- **DatagramPacket**: represents a datagram packet and are used to implement a connectionless packet-delivery service. Each message is routed from one computer to another based solely on information contained within the packet.

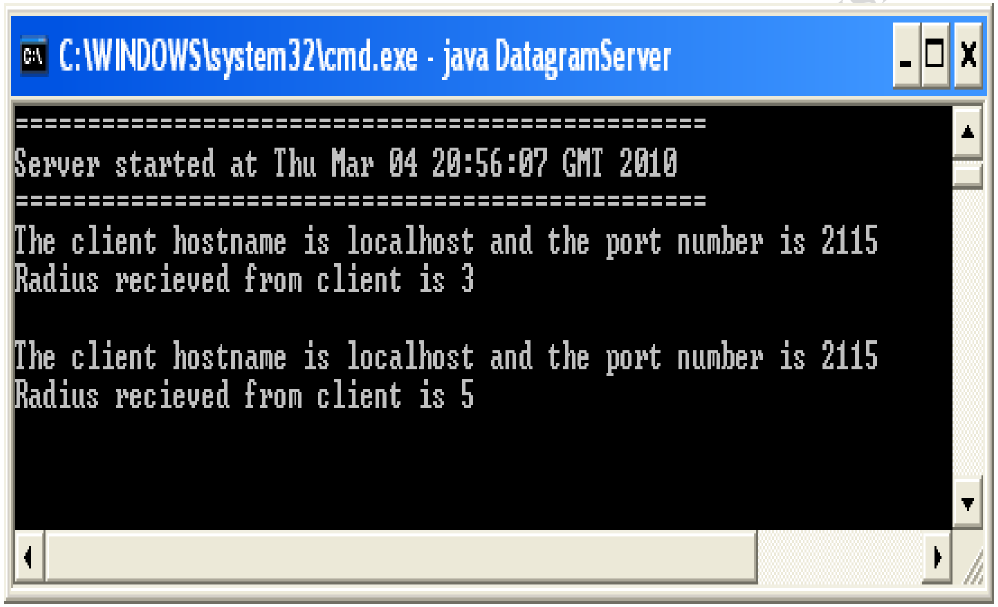
- **DatagramSocket**: represents a socket for sending and receiving datagram packets. A datagram socket is the sending/receiving point for a packet-delivery service. Each packet sent or received on a datagram socket is individually addressed and routed.

Programming With Datagram Sockets

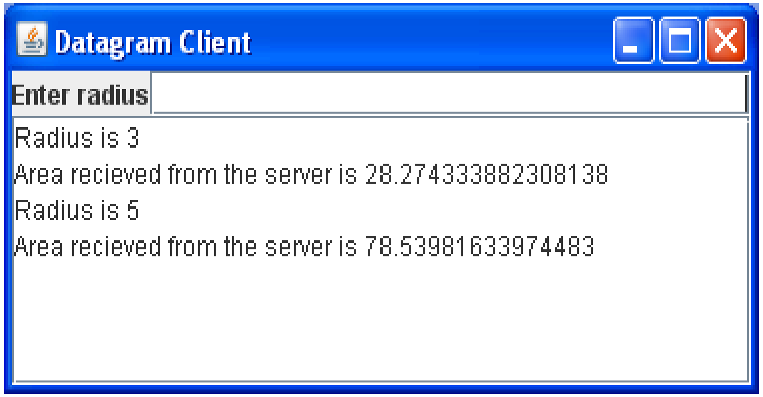
- Datagram programming is different from stream socket programming in the sense that there is no concept of a StreamSocket for datagram's.
- Both the client and the server use DatagramSocket to send and receive packets.
- Now we are going to designate one application as the server and create a DatagramSocket with a specified port.
- A client can then also create a DatagramSocket (without specifying a port number).
- When a client sends a packet to the server, the client’s IP address and port number are contained within the packet.
    - The server can extract these details from the packet and use them to send information back to the client.

Example 4: Rework Example 1 Using Datagram’s

- We are now going to rewrite example 1 using datagram's rather than socket streams.
- The client sends the radius to the server.
- The server receives this information, uses it to find the area and then sends it back to the client.



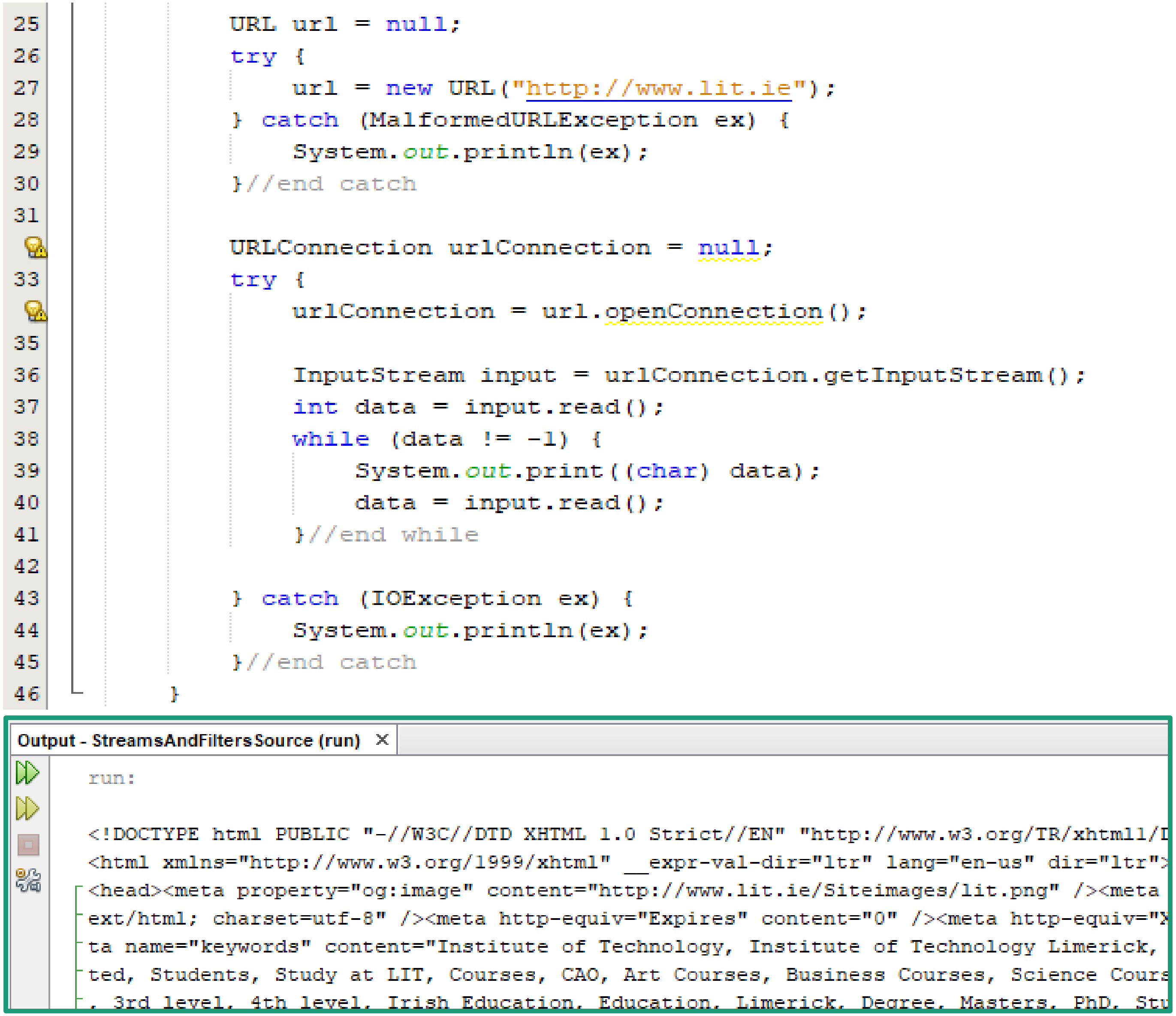
```
================================================
Server started at Thu Mar 04 20:56:07 GMT 2010
================================================
The client hostname is localhost and the port number is 2115
Radius recieved from client is 3

The client hostname is localhost and the port number is 2115
Radius recieved from client is 5
```



Enter radius

```
Radius is 3
Area recieved from the server is 28.274333882308138
Radius is 5
Area recieved from the server is 78.53981633974483
```

Example 5: Working With URL’s



```
        URL url = null;
        try {
            url = new URL("http://www.lit.ie");
        } catch (MalformedURLException ex) {
            System.out.println(ex);
        }//end catch

        URLConnection urlConnection = null;
        try {
            urlConnection = url.openConnection();

            InputStream input = urlConnection.getInputStream();
            int data = input.read();
            while (data != -1) {
                System.out.print((char) data);
                data = input.read();
            }//end while

        } catch (IOException ex) {
            System.out.println(ex);
        }//end catch
    }
```

```
run:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/D
<html xmlns="http://www.w3.org/1999/xhtml" __expr-val-dir="ltr" lang="en-us" dir="ltr">
<head><meta property="og:image" content="http://www.lit.ie/Siteimages/lit.png" /><meta
ext/html; charset=utf-8" /><meta http-equiv="Expires" content="0" /><meta http-equiv="X
ta name="keywords" content="Institute of Technology, Institute of Technology Limerick,
ted, Students, Study at LIT, Courses, CAO, Art Courses, Business Courses, Science Cours
, 3rd level, 4th level, Irish Education, Education, Limerick, Degree, Masters, PhD, Stu
```

Example 6: Creating a Web Browser (Optional)

- JEditorPane can be used to display HTML files. Consult the package called “Example6_Web_Browser”. It contains three files:

  1. WebBrowser.java
  2. WebToolBar.java
  3. WebBrowserPane.java

Future Reading

- Java screen-scraping API's such as Jaunt or Jsoup or HTMLUnit.

References

Y. Daniel Liang (2017) *Intro to Java Programming and Data Structures, Comprehensive Version*. 11/E. Pearson. ISBN-13 978-0134670942 (Link)

Paul J Deitel(2016) *Java How To Program*. 10/E. ISBN-13 9780134800271 (Link)

Cay S. Horstmann (2018) Core Java SE 9 For the Impatient. 2/E. ISBN-13 978-0-13-469472-6 (Link)

http://tutorials.jenkov.com/java-networking/index.html