

NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCE

Computer Network Lab (CL-307) READING MATERIAL FOR LAB-03

Introduction to Socket Programming

The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving the problem at hand.

The java.net package provides support for the two common network protocols:

- **TCP:** TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
- **UDP:** UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

Socket:

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

- The **java.net.Socket** class represents a socket
- The **java.net.ServerSocket** class provides a mechanism for the server program to listen for clients and establish connections with them.

The following steps occur when establishing a TCP connection between two computers using sockets:

1. The **server instantiates a ServerSocket object**, denoting which port number communication is to occur on.
2. The **server invokes the accept() method of the ServerSocket class**. This method waits until a client connects to the server on the given port.

3. After the server is waiting, a **client instantiates a Socket object**, specifying the server name and port number to connect to.
4. **The constructor of the Socket class attempts to connect the client to the specified server and port number.** If communication is established, the client now has a Socket object capable of communicating with the server.
5. On the server side, the **accept() method returns a reference to a new socket on the server** that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an OutputStream and an InputStream.

The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.

TCP is a two way communication protocol, so data can be sent across both streams at the same time. There are following useful classes providing complete set of methods to implement sockets.

ServerSocket Class Methods:

The **java.net.ServerSocket** class is used by server applications to obtain a port and listen for client requests

The ServerSocket class has four constructors:

SN	Methods with Description
1	public ServerSocket(int port) throws IOException Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
2	public ServerSocket(int port, int backlog) throws IOException Similar to the previous constructor, the backlog parameter specifies how many incoming clients to store in a wait queue.
3	public ServerSocket(int port, int backlog, InetAddress address) throws IOException Similar to the previous constructor, the InetAddress parameter specifies the local IP address to bind to. The InetAddress is used for servers that may have multiple IP addresses, allowing the server to specify which of its IP addresses to accept client requests on
4	public ServerSocket() throws IOException Creates an unbound server socket. When using this constructor, use the bind() method when you are ready to bind the server socket

If the `ServerSocket` constructor does not throw an exception, it means that your application has successfully bound to the specified port and is ready for client requests.

Here are some of the common methods of the `ServerSocket` class:

SN	Methods with Description
	public int getLocalPort() Returns the port that the server socket is listening on. This method is useful if
1	you passed in 0 as the port number in a constructor and let the server find a port for you.
	public Socket accept() throws IOException Waits for an incoming client. This method blocks until either a client connects to
2	the server on the specified port or the socket times out, assuming that the time-out value has been set using the <code>setSoTimeout()</code> method. Otherwise, this method blocks indefinitely
	public void setSoTimeout(int timeout)
3	Sets the time-out value for how long the server socket waits for a client during the <code>accept()</code> .
	public void bind(SocketAddress host, int backlog) Binds the socket to the specified server and port in the <code>SocketAddress</code> object.
4	Use this method if you instantiated the <code>ServerSocket</code> using the no-argument constructor.

When the `ServerSocket` invokes `accept()`, the method does not return until a client connects. After a client does connect, the `ServerSocket` creates a new `Socket` on an unspecified port and returns a reference to this new `Socket`. A TCP connection now exists between the client and server, and communication can begin.

Socket Class Methods:

The **`java.net.Socket`** class represents the socket that both the client and server use to communicate with each other. The client obtains a `Socket` object by instantiating one, whereas the server obtains a `Socket` object from the return value of the `accept()` method.

The Socket class has five constructors that a client uses to connect to a server:

SN	Methods with Description
	public Socket(String host, int port) throws UnknownHostException, IOException.
1	This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.
	public Socket(InetAddress host, int port) throws IOException
2	This method is identical to the previous constructor, except that the host is denoted by an InetAddress object.
	public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException.
3	Connects to the specified host and port, creating a socket on the local host at the specified address and port.
	public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException.
4	This method is identical to the previous constructor, except that the host is denoted by an InetAddress object instead of a String
	public Socket()
5	Creates an unconnected socket. Use the connect() method to connect this socket to a server.

When the Socket constructor returns, it does not simply instantiate a Socket object but it actually attempts to connect to the specified server and port.

Some methods of interest in the Socket class are listed here. Notice that both the client and server have a Socket object, so these methods can be invoked by both the client and server.

SN	Methods with Description
	public void connect(SocketAddress host, int timeout) throws IOException
1	This method connects the socket to the specified host. This method is needed only when you instantiated the Socket using the no-argument constructor.
	public InetAddress getInetAddress()
2	This method returns the address of the other computer that this socket is connected to.
	public int getPort()
3	Returns the port the socket is bound to on the remote machine.
	public int getLocalPort()
4	Returns the port the socket is bound to on the local machine.
	public SocketAddress getRemoteSocketAddress() Returns the address of the remote socket.
	public InputStream getInputStream() throws IOException
6	Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
	public OutputStream getOutputStream() throws IOException
7	Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket
	public void close() throws IOException
8	Closes the socket, which makes this Socket object no longer capable of connecting again to any server

InetAddress Class Methods:

This class represents an Internet Protocol (IP) address. Here are following useful methods which you would need while doing socket programming:

SN	Methods with Description
1	static InetAddress getByAddress(byte[] addr) Returns an InetAddress object given the raw IP address .
2	static InetAddress getByAddress(String host, byte[] addr) Create an InetAddress based on the provided host name and IP address.
3	static InetAddress getByName(String host) Determines the IP address of a host, given the host's name.
4	String getHostAddress() Returns the IP address string in textual presentation.
5	String getHostName() Gets the host name for this IP address.
6	static InetAddress InetAddress getLocalHost() Returns the local host.
7	String toString() Converts this IP address to a String.