Networking Code

CSCI 201
Principles of Software Development

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu
Outline

- Server Networking
- Client Networking
- Program
Server Software

- A server application is only able to serve requests that are addressed to the computer on which the server application is running
  - A server program cannot listen to a port on another physical server
- So, the only data the server application needs is the port on which to listen
  - The `ServerSocket` constructor only takes a port as a parameter
- Multiple networked applications can be running on the same computer as long as they are all using different ports
Client connects to Server IP address on port 6789

Server accepts connection to Client IP address on whatever port it dynamically selected

Client

Server

Listening on port 6789 on what IP address?
Multi-Threading with Networking

- Multi-threading is usually necessary with networking since there are two things that often are done at the same time
  - The ability to send data
  - The ability to receive data

- If sending and receiving are not performed in series, multi-threading will be needed
  - Some applications may be synchronous and only need one program to send data then wait for a response – that would *not* require multi-threading
import java.io.*; // to save space
import java.net.*; // to save space
public class NetworkingServer {
    public NetworkingServer() {
        ServerSocket ss = null;
        Socket s = null;
        PrintWriter pw = null;
        BufferedReader br = null;
        try {
            System.out.println("Starting Server");
            ss = new ServerSocket(6789);
            s = ss.accept();
            br = new BufferedReader(new InputStreamReader(s.getInputStream()));
            pw = new PrintWriter(s.getOutputStream());
            String line = br.readLine();
            System.out.println("Line Received: " + line);
            String str="CSCI 201";
            System.out.println("Sending: " + str);
            pw.println(str);
            pw.flush();
        } catch (IOException ioe) {
            System.out.println("IOE: " + ioe.getMessage());
        }
    }
    public static void main(String[] args) {
        new NetworkingServer();
    }
}
Flushing

- Operating systems try to optimize networking similar to how they optimize file I/O
- Data is written into a buffer before it is sent along the socket
- The contents of the buffer will not be transmitted over the network until it fills up unless we explicitly flush the data
- Never forget to flush!
Outline

• Server Networking
• Client Networking
• Program
Client Software

- A client application is able to connect to any server application to which it has access, whether running on the same computer as the client or a different computer.
- The client application needs both the **IP address** of the server and the **port** on which the server application is listening.
  - Remember that multiple server applications can be running on the same computer as long as they are listening on different ports.
- A **Socket** is the combination of the IP address and port number needed by the client.
import java.io.*; // to save space
import java.net.Socket;
public class NetworkingClient {
    public NetworkingClient() {  
        Socket s = null;
        BufferedReader br = null;
        PrintWriter pw = null;
        try {
            System.out.println("Starting Client");
            s = new Socket("localhost", 6789);
            br = new BufferedReader(new InputStreamReader(s.getInputStream()));
            pw = new PrintWriter(s.getOutputStream());
            String str = "Line being sent";
            System.out.println("Sending: " + str);
            pw.println(str);
            pw.flush();
            String line = br.readLine();
            System.out.println("Line Received: " + line);
        } catch (IOException ioe) {
            System.out.println("IOE: " + ioe.getMessage());
        } catch (IOException ioe) {
            System.out.println("ioe: " + ioe.getMessage());
        } finally {
            try {
                if (pw != null)
                    pw.close();
                if (br != null)
                    br.close();
                if (s != null)
                    s.close();
            } catch (IOException ioe) {
                System.out.println("ioe: " + ioe.getMessage());
            }
        } // ends finally
    }
    public static void main(String [] args) {
        new NetworkingClient();
    }
}
Outline

• Server Networking
• Client Networking
• Program
Program

- Write a multi-threaded chat program that allows multiple clients to communicate with each other in an asynchronous manner. The clients should communicate with a server program.

C:>java ChatClient localhost 6789
Hello
How are you?
Them: Fine, and you?
Good, thanks.

C:>java ChatClient localhost 6789
Them: Hello
Them: How are you?
**Fine, and you?**
Them: Good, thanks.