The goal of this assignment is to gain experience with network programming, specifically with sockets in Java. This assignment will also require you to develop a server and client simultaneously, which is an important skill in itself. The client will allow the user to enter an arbitrary number of integers which will be sent to the server. The server will then sum the integers and return the result back to the client for printing.

1. [20] Create a new Eclipse project named Assignment8. Create two classes, one for your client and one for your server. Each class will need a main method so that both classes can be executed. The client class should first prompt the user to enter integers until the user enters a blank line (i.e., presses enter immediately). These integers should be stored in a data structure for later retrieval.

2. [10] The server class should first create a java.net.ServerSocket on a port of your choice and then listen for a java.net.Socket connection. Note that the ServerSocket accept method is a blocking call. Your server does not need to handle multiple connections (i.e., the Server can terminate after handling a single Socket connection).

3. [20] The client should then create a Socket and connect to the ServerSocket (from part 2) on the local host (i.e., your machine). See java.net.InetAddress for more details on connecting to the local host. You’ll likely want to add some print statements at this point to ensure that the connection is being made successfully.

4. [20] The client should then send the number of integers to expect to the server followed by the integers to sum (from part 1). This will be much easier if you wrap the Socket’s java.io.OutputStream with a java.io.ObjectOutputStream, which provides write methods for specific types. Don’t forget to flush the stream when you’re done writing! Do not simply write an array or list to the socket as this makes the assignment trivial. ☺

5. [20] After accepting a Socket connection, the server should read the expected number of integers followed by the integers to sum from the Socket. Similarly, this will be much easier if you wrap the Socket’s java.io.InputStream with a java.io.ObjectInputStream, which provides blocking read methods for specific types. Sum the integers and then write the result to the Socket.

6. [10] Finally, the client should receive and print the resulting sum.

7. [+10] (Extra credit) Obfuscate the integers to sum before they are sent to the server (and deobfuscate them when they arrive) and the sum result before it is sent back to the client (and
deobfuscate it when it arrives). You may use any method of obfuscation you’d like, such as a simple XOR or a Caesar Cipher.

Zip the Assignment8 folder in your Eclipse workspace directory and upload the .zip file to Blackboard (see Assignment 8 assignment in the Course Documents area).