**Exercise 03**

**Name:** Jose Juan Sandoval

**Link to Project:** <https://github.com/Juanchiselo/CS380/tree/master/Exercises/Exercise%2003>

**Java Code**

Ex3Client.java

package Exercise03;  
  
import java.io.IOException;  
import java.net.Socket;  
import java.net.UnknownHostException;  
  
public class Ex3Client  
{  
 private static Socket *socket*;  
  
 public static void main(String[] args)  
 {  
 *connectToServer*();  
 }  
  
 */\*\*  
 \* Connects the client to the server and  
 \* creates a Listener thread.  
 \*/* public static void connectToServer()  
 {  
 String hostName = "codebank.xyz";  
 int portNumber = 38103;  
  
 try  
 {  
 *socket* = new Socket(hostName, portNumber);  
 System.*out*.println("Connected to server.");  
 new ListenerThread(*socket*).start();  
 }  
 catch (UnknownHostException e) {  
 System.*err*.println("ERROR: Unknown host " + hostName + ".");  
 } catch (Exception e) {  
 System.*err*.println("ERROR: Could not connect to " + hostName + ".");  
 }  
 }  
  
 */\*\*  
 \* Disconnects the client from the server.  
 \*/* public static void disconnectFromServer()  
 {  
 try {  
 *socket*.close();  
 } catch (IOException e) {  
 System.*err*.println("ERROR: " + e.getMessage() + ".");  
 }  
 }  
}

ListenerThread.java

package Exercise03;  
  
import java.io.\*;  
import java.net.Socket;  
  
public class ListenerThread extends Thread  
{  
 public volatile static boolean *endThread* = false;  
 private Socket socket = null;  
 private InputStream inputStream;  
 private OutputStream outputStream;  
  
 public ListenerThread(Socket socket)  
 {  
 super("Listener Thread");  
 this.socket = socket;  
 }  
  
 */\*\*  
 \* The overridden run() function belonging to the Thread class.  
 \* This is what handles the communication between the server and the client.  
 \*/* public void run()  
 {  
 try  
 {  
 inputStream = socket.getInputStream();  
 outputStream = socket.getOutputStream();  
  
 int dataSize = inputStream.read();  
 System.*out*.println("Reading " + dataSize + " bytes.");  
  
 byte[] data = new byte[dataSize];  
 int currentByte;  
 System.*out*.print("Data received:\n\t");  
 for(int i = 0; i < dataSize; i++)  
 {  
 currentByte = inputStream.read();  
 data[i] = (byte)currentByte;  
  
 if((i != 0) && (i % 10 == 0))  
 System.*out*.print("\n\t");  
  
 System.*out*.printf("%02X", data[i]);  
 }  
  
 short checksum = *internetChecksum*(data);  
 System.*out*.printf("\nChecksum calculated: 0x%02X\n", checksum);  
  
 respondToServer(checksum, 2);  
 Ex3Client.*disconnectFromServer*();  
 }  
 catch (IOException e) {  
 System.*err*.println("ERROR: Connection lost with server.");  
 }  
 }  
  
 */\*\*  
 \* Calculates the Internet Checksum from a given array of data.  
 \* The algorithm maintains a 32-bit number as the sum and goes through  
 \* the array two bytes at a time, forms a 16-bit number out of each pair  
 \* of bytes and adds it to the sum. After each time it adds,  
 \* it checks for overflow. If overflow occurs, it is cleared and  
 \* added back in to the sum (acting like a wrap-around). Finally,  
 \* when the sum is calculated we perform one's complement and return  
 \* the rightmost 16 bits of the sum.  
 \** ***@param*** *data - The array of bytes to calculate the Internet Checksum from.  
 \** ***@return*** *- Returns the Internet Checksum.  
 \*/* private static short internetChecksum(byte[] data)  
 {  
 int sum = 0;  
 int firstByte;  
 byte secondByte;  
 boolean allPairs = (data.length % 2 == 0);  
  
 for(int i = 0; i < data.length; i += 2)  
 {  
 firstByte = data[i];  
 // If the last byte doesn't have a pair.  
 if(!allPairs && (i == data.length - 1))  
 secondByte = 0;  
 else  
 secondByte = data[i + 1];  
  
 // Forms a 16-bit number from two consecutive bytes  
 // and adds it to the sum.  
 sum += ((firstByte << 8 & 0xFF00) | (secondByte & 0xFF));  
  
 // Checks for overflow on the sum.  
 if((sum & 0xFFFF0000) > 0)  
 {  
 sum &= 0xFFFF;  
 sum++;  
 }  
 }  
  
 // Get one's complement and return the 16 rightmost bits.  
 return (short)~(sum & 0xFFFF);  
 }  
  
 */\*\*  
 \* Responds to the server with the 2 byte sequence  
 \* obtained from the given Internet Checksum.  
 \** ***@param*** *checksum - The Internet Checksum to be sent to the server.  
 \*/* private void respondToServer(short checksum, int sequenceSize)  
 {  
 try  
 {  
 for(int i = sequenceSize - 1; i >= 0; i--)  
 outputStream.write(checksum >> (8 \* i));  
  
 if(inputStream.read() == 1)  
 System.*out*.println("Response good.");  
 else  
 System.*out*.println("Response bad.");  
 }  
 catch (IOException e) {  
 System.*err*.println("ERROR: " + e.getMessage());  
 }  
 }  
}