**Name:** Hung Viet Luu

**Link to Project:** <https://github.com/hvluu/CS380/tree/master/Exercise2>

**Ex2Client.java**

*import java.io.IOException;*

*import java.net.Socket;*

*import java.net.UnknownHostException;*

*public class Ex2Client*

*{*

*private static Socket socket;*

*public static void main(String[] args)*

*{*

*connect();*

*}*

*/\*\**

*\* Connects the client to the server and*

*\* creates a thread.*

*\*/*

*public static void connect()*

*{*

*String hostName = "18.221.102.182";*

*int portNumber = 38102;*

*try*

*{*

*socket = new Socket(hostName, portNumber);*

*new Connection(socket).start();*

*System.out.println("Connected to server.");*

*}*

*catch (UnknownHostException e)*

*{*

*System.err.println("ERROR: Unknown host " + hostName + ".");*

*}*

*catch (IOException e)*

*{*

*System.err.println("ERROR: Could not connect to " + hostName + ".");*

*}*

*}*

*/\*\**

*\* Disconnects the client from the server.*

*\*/*

*public static void disconnect()*

*{*

*try*

*{*

*socket.close();*

*System.out.println("Disconnected from server.");*

*}*

*catch (IOException e)*

*{*

*System.err.println("ERROR: " + e.getMessage());*

*}*

*}*

*}*

**ListenerThread.java**

*import java.io.\*;*

*import java.net.Socket;*

*import java.nio.ByteBuffer;*

*import java.util.zip.\*;*

*public class Connection extends Thread*

*{*

*public volatile static boolean endThread = false;*

*private Socket socket = null;*

*public Connection(Socket socket)*

*{*

*super("Connecting Thread");*

*this.socket = socket;*

*}*

*@Override*

*/\*\**

*\* Override run() function*

*\* run() function will handle the communication between the server*

*\* and the client.*

*\*/*

*public void run()*

*{*

*try*

*{*

*// Counter to track the amount bytes received*

*int counter = 1;*

*final short AMOUNT\_BYTES\_RECEIVED = 100;*

*// Variables and array to hold the server's responses.*

*int firstHalfByte, secondHalfByte, reconstructedByte;*

*byte receivedBytes[] = new byte[AMOUNT\_BYTES\_RECEIVED];*

*String receivedBytesString = " ";*

*// Constants for bitwise operations.*

*final short HALF\_BYTE\_SIZE = 4;*

*final int BITMASK = 0xFF;*

*// InputStream object to receive and read server's responses.*

*InputStream inputStream = socket.getInputStream();*

*// The main loop of execution.*

*// InputStream.read() returns -1 when the stream ends*

*// and there is nothing else to read*

*while((firstHalfByte = inputStream.read()) != -1*

*&& (secondHalfByte = inputStream.read()) != -1)*

*{*

*// The full byte obtained from combining the first and second half byte sent by the server.*

*// The first half of the byte gets LEFT SHIFTED by the size of 4 bits.*

*// The second half gets ANDed with the 0xFF bitmask.*

*// Both halves are ORed with each other to combine them into a single byte.*

*reconstructedByte = (firstHalfByte << HALF\_BYTE\_SIZE) | (secondHalfByte & BITMASK);*

*receivedBytesString += Integer.toHexString(reconstructedByte).toUpperCase();*

*receivedBytes[counter - 1] = (byte) reconstructedByte;*

*// This only separates the received bytes*

*// into groups of 20 for user convenience.*

*if(counter % 10 == 0*

*&& counter != AMOUNT\_BYTES\_RECEIVED)*

*receivedBytesString += "\n ";*

*else if(counter == AMOUNT\_BYTES\_RECEIVED)*

*{*

*System.out.println("Received bytes:\n" + receivedBytesString);*

*long checksum = verify(receivedBytes);*

*System.out.println("Generated CRC32: "*

*+ Long.toHexString(checksum).toUpperCase() + ".");*

*respond(checksum);*

*break;*

*}*

*counter++;*

*}*

*Ex2Client.disconnect();*

*}*

*catch (IOException e)*

*{*

*System.err.println("ERROR: Connection lost with server.");*

*}*

*catch (Exception e)*

*{*

*System.err.println("ERROR: " + e.getMessage());*

*}*

*}*

*/\*\**

*\* Verifies data by calculating its CRC32 checksum.*

*\* @param data - The data to calculate the checksum for.*

*\* @return - Returns the checksum.*

*\*/*

*private long verify(byte[] data)*

*{*

*Checksum checksum = new CRC32();*

*checksum.update(data, 0, data.length);*

*return checksum.getValue();*

*}*

*/\*\**

*\* Breaks up the checksum into a sequence*

*\* of 4 bytes to send to the server.*

*\* @param checksum - The checksum to be broken into 4 bytes.*

*\* @return - A byte array filled with the 4 bytes.*

*\*/*

*private byte[] prepare(long checksum)*

*{*

*// Allocates 8 spaces in the ByteBuffer.*

*ByteBuffer buffer = ByteBuffer.allocate(Long.BYTES);*

*// Writes 8 bytes into the ByteBuffer*

*// containing the given long value.*

*buffer.putLong(checksum);*

*// Gets the array that backs the ByteBuffer.*

*byte checksumReturn[] = buffer.array();*

*// A new array to hold the response because*

*// we only need to send 4 bytes back to the server.*

*byte response[] = new byte[4];*

*for(int i = 0; i < 4; i++)*

*response[i] = checksumReturn[i + 4];*

*return response;*

*}*

*/\*\**

*\* Responds to the server with the 4 byte sequence*

*\* obtained from the given checksum.*

*\* @param checksum - The checksum to be sent to the server.*

*\*/*

*private void respond(long checksum)*

*{*

*try*

*{*

*socket.getOutputStream().write(prepare(checksum));*

*int serverResponse;*

*if((serverResponse = socket.getInputStream().read()) == 1)*

*System.out.println("Response good.");*

*else*

*System.out.println("Bad response. Server returns " + serverResponse);*

*}*

*catch (IOException e)*

*{*

*System.err.println("ERROR: " + e.getMessage());*

*}*

*}*

*}*