```java
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class RemoteVariableClientTCP{
    static InetAddress aHost;
    static int serverPort;
    static Socket clientSocket = null;
    static String replyString = "";


    public static void main(String args[]){
        // args give message contents and server hostname
        System.out.println("The client is running." );
        try {
            // set to localhost to host on local machine and set port
            aHost = InetAddress.getByName("localhost");
            serverPort = 6789;

            // input server port to use
            System.out.print("Input a server side port number: ");
            Scanner readline = new Scanner(System.in);
            serverPort = readline.nextInt();

            clientSocket = new Socket("localhost", serverPort);


            String ID = "";
            String operation = "";
            String value = "";

            String packet = "";

            // create packet to server
```

```
35              while (true) {
36                  System.out.println("\n1. Add a
   value to your sum.\n" +
37                          "2. Subtract a value from
   your sum.\n" +
38                          "3. Get your sum.\n" +
39                          "4. Exit client");
40
41                  int in = readline.nextInt();
42
43                  if(in == 1){
44                      operation = "1 ";
45                      System.out.println("Enter value
    to add: ");
46                      value = String.valueOf(readline
   .nextInt());
47                      System.out.println("Enter your
   ID: ");
48                      ID = String.valueOf(readline.
   nextInt()) + " ";
49
50                      packet += operation + ID +
   value;
51
52                  }
53                  else if(in == 2){
54                      operation = "2 ";
55                      System.out.println("Enter value
    to subtract: ");
56                      value = String.valueOf(readline
   .nextInt());
57                      System.out.println("Enter your
   ID: ");
58                      ID = String.valueOf(readline.
   nextInt()) + " ";
59
60                      packet += operation + ID +
   value;
61
62                  }
63                  else if(in == 3){
```

```
64                      operation = "3 ";
65                      System.out.println("Enter your
    ID: ");
66                      ID = String.valueOf(readline.
   nextInt()) + " ";
67                      value = "";
68
69                      packet += operation + ID +
   value;
70                  }
71                  else if(in == 4){
72                      System.out.println("Client
   side quitting. The remote variable server is still
    running.");
73                      // send reply if not halt
74                      clientSocket.close();
75                      break;
76                  }
77
78                  System.out.println("The result is
   : " + communicate(packet));
79
80                  packet = "";
81
82
83              }
84
85              // catch potential exceptions
86          }catch (IOException e) {
87              System.out.println("IO Exception:" + e
   .getMessage());
88          } finally {
89              try {
90                  if (clientSocket != null) {
91                      clientSocket.close();
92                  }
93              } catch (IOException e) {
94                  // ignore exception on close
95              }
96          }
97
```

```
 98     }
 99     // ----------------------- Proxy style
    communication code -----------------------
100     // sends packet and reads reply
101     public static String communicate(String in)
    throws IOException {
102
103         BufferedReader read = new BufferedReader(
    new InputStreamReader(clientSocket.getInputStream
    ()));
104
105         PrintWriter out = new PrintWriter(new
    BufferedWriter(new OutputStreamWriter(clientSocket
    .getOutputStream())));
106
107         out.println(in);
108         out.flush();
109
110         String replyString = read.readLine();
111
112         return replyString;
113     }
114 }
```

```java
import java.net.*;
import java.io.*;
import java.util.Scanner;
import java.util.*;

public class RemoteVariableServerTCP{
    private static TreeMap<Integer, Integer> tree_map = new TreeMap<Integer, Integer>();

    static String instructions = "1. Add a value to your sum.\n" +
            "2. Subtract a value from your sum.\n" +
            "3. Get your sum.\n" +
            "4. Exit client";

    public static void main(String args[]){
        System.out.println("The server is running."); // lab instructions
        Socket clientSocket = null;
        byte[] buffer = new byte[1000];  // set up packet buffer for client message
        try{
            // set up ports
            System.out.print("Input a server port number to listen on: ");
            Scanner readline = new Scanner(System.in);
            int serverPort = readline.nextInt(); // convert to int

            ServerSocket listenSocket = new ServerSocket(serverPort);

            int id;
            int value;

            while(true){ // loop to continue until 'halt!' is sent
                clientSocket = listenSocket.accept();
```

```
31
32                  Scanner in;
33                  in = new Scanner(clientSocket.
   getInputStream());
34
35                  PrintWriter out;
36                  out = new PrintWriter(new
   BufferedWriter(new OutputStreamWriter(clientSocket.
   getOutputStream())));
37
38
39                  // operation request and reply
40                  while(in.hasNextLine()){
41                      String requestString = in.
   nextLine();
42
43                      String[] arrRequestString =
   requestString.split(" ");
44
45                      id = Integer.valueOf(
   arrRequestString[1]);
46
47                      // treemap logic to ensure
   unique id is generated
48                      if(!tree_map.containsKey(id)){
49                          if(arrRequestString[0].
   equals("1")){
50                              value = Integer.valueOf
   (arrRequestString[2]);
51                              tree_map.put(id, value
   );
52                          }
53                          else if(arrRequestString[0
   ].equals("2")){
54                              value = -1*Integer.
   valueOf(arrRequestString[2]);
55                              tree_map.put(id, value
   );
56                          }
57                          else{
58                              tree_map.put(id, 0);
```

```
58 // default for no id input
59                         }
60                     }
61                     else{
62                         if(arrRequestString[0].
   equals("1")){
63                             value = Integer.valueOf
   (arrRequestString[2]);
64                             tree_map.replace(id,
   tree_map.get(id)+value);
65                         }
66                         else if(arrRequestString[0
   ].equals("2")){
67                             value = Integer.valueOf
   (arrRequestString[2]);
68                             tree_map.replace(id,
   tree_map.get(id)-value);
69                         }
70                     }
71
72                     System.out.println("\nVisitorID
   : " + arrRequestString[1]);
73                     System.out.println("Operation
    #: " + arrRequestString[0]);
74                     System.out.println("Return
   Variable " + String.valueOf(tree_map.get(id)));
75
76
77                     out.println(String.valueOf(
   tree_map.get(id)));
78                     out.flush();
79                 }
80             }
81             // catch potential exceptions
82         } catch (IOException e) {
83             System.out.println("IO Exception:" + e.
   getMessage());
84
85             // If quitting (typically by you
   sending quit signal) clean up sockets
86         } finally {
```

```
87                 try {
88                     if (clientSocket != null) {
89                         clientSocket.close();
90                     }
91                 } catch (IOException e) {
92                     // ignore exception on close
93                 }
94             }
95         }
96 }
```