Name: Aditi Gupta
Email: argupta@andrew.cmu.edu

Project 2

Project2Task0

"Project2Task0Client"

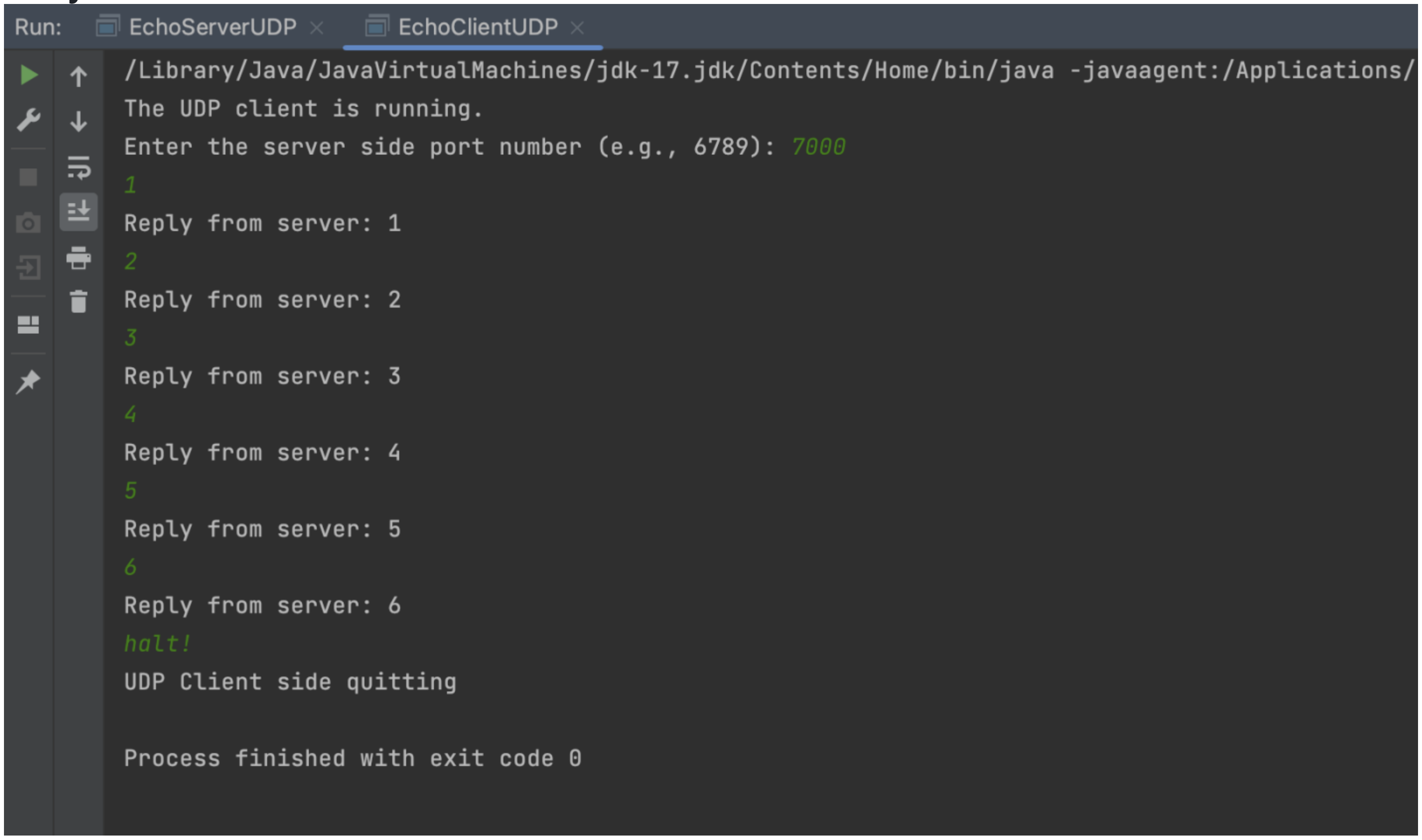
```
// Aditi Gupta - argupta@andrew.cmu.edu - Project2Task0
// Taken code from EchoClientUDP.java from Coulouris text

import java.net.*;
import java.io.*;

public class EchoClientUDP {
    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        try {
            // Announce that the client is running
            System.out.println("The UDP client is running.");

            // Change the server address to "localhost"
            InetAddress aHost = InetAddress.getByName("localhost");

            // Prompt the user for the server side port number
            System.out.print("Enter the server side port number (e.g., 6789): ");
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            int serverPort = Integer.parseInt(reader.readLine());

            // Create a DatagramSocket for sending and receiving UDP packets
            aSocket = new DatagramSocket();

            String nextLine;
            BufferedReader typed = new BufferedReader(new InputStreamReader(System.in));

            while ((nextLine = typed.readLine()) != null) {
                // Convert the input text into bytes and create a DatagramPacket
                byte[] m = nextLine.getBytes();
                DatagramPacket request = new DatagramPacket(m, m.length, aHost,
serverPort);
                aSocket.send(request);

                // Check if the client wants to halt
                if (nextLine.trim().equalsIgnoreCase("halt!")) {
                    System.out.println("UDP Client side quitting");
                    break;
                }

                // Receive a reply from the server
                byte[] buffer = new byte[1000];
                DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
                aSocket.receive(reply);

                // Extract and print the reply from the server
                int replyLength = reply.getLength();
                byte[] replyData = new byte[replyLength];

                // Code taken from this site:
                // https://stackoverflow.com/questions/5690954/java-how-to-read-an-
unknown-number-of-bytes-from-an-inputstream-socket-socke
                System.arraycopy(reply.getData(), 0, replyData, 0, replyLength);
                String replyString = new String(replyData);
                System.out.println("Reply from server: " + replyString);
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            // Ensure the socket is closed at the end
            if (aSocket != null)
                aSocket.close();
        }
    }
}
```

"Project2Task0Server"

```
// Aditi Gupta - argupta@andrew.cmu.edu - Project2Task0
// Taken code from EchoServerUDP.java from Coulouris text

import java.net.*;
import java.io.*;

public class EchoServerUDP {
    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        byte[] buffer = new byte[1000];
        try {
            // Announce that the server is running
            System.out.println("The UDP server is running");
            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            // Prompt the user for the port number to listen on
            System.out.print("Enter the port number for the server to listen on (e.g.,
6789): ");
            int serverPort = Integer.parseInt(reader.readLine());

            // Create a DatagramSocket to listen for incoming UDP packets
            aSocket = new DatagramSocket(serverPort);
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);

            while (true) {
                // Receive an incoming UDP packet (request) from a client
                aSocket.receive(request);

                // Calculate the length of actual data in the request
                int requestLength = request.getLength();
                byte[] requestData = new byte[requestLength];

                // Code taken from this site:
                // https://stackoverflow.com/questions/5690954/java-how-to-read-an-
unknown-number-of-bytes-from-an-inputstream-socket-socke
                System.arraycopy(request.getData(), request.getOffset(), requestData, 0,
requestLength);
                String requestString = new String(requestData, 0, requestLength);

                // Print the received request
                System.out.println("Echoing: " + requestString);

                // Check if the client wants to halt
                if (requestString.trim().equalsIgnoreCase("halt!")) {
                    System.out.println("UDP Server side quitting");
                    break;
                }
```

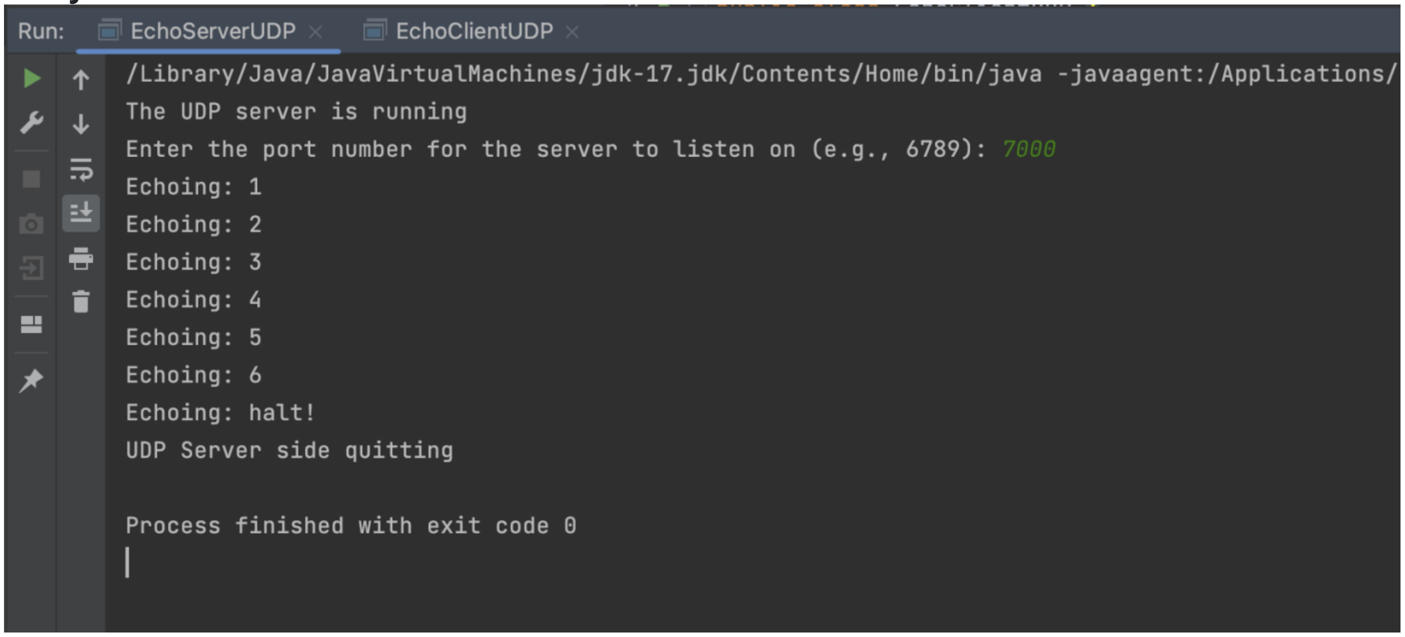
```java
                // Prepare and send a reply to the client
                DatagramPacket reply = new DatagramPacket(requestData, requestLength,
request.getAddress(), request.getPort());
                aSocket.send(reply);
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            // Ensure the socket is closed at the end
            if (aSocket != null)
                aSocket.close();
        }
    }

}
```

"Project2Task0ClientConsole"



```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/
The UDP client is running.
Enter the server side port number (e.g., 6789): 7000
1
Reply from server: 1
2
Reply from server: 2
3
Reply from server: 3
4
Reply from server: 4
5
Reply from server: 5
6
Reply from server: 6
halt!
UDP Client side quitting

Process finished with exit code 0
```

"Project2Task0ServerConsole"



Run: EchoServerUDP EchoClientUDP

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/
The UDP server is running
Enter the port number for the server to listen on (e.g., 6789): 7000
Echoing: 1
Echoing: 2
Echoing: 3
Echoing: 4
Echoing: 5
Echoing: 6
Echoing: halt!
UDP Server side quitting

Process finished with exit code 0
```

Project2Task1

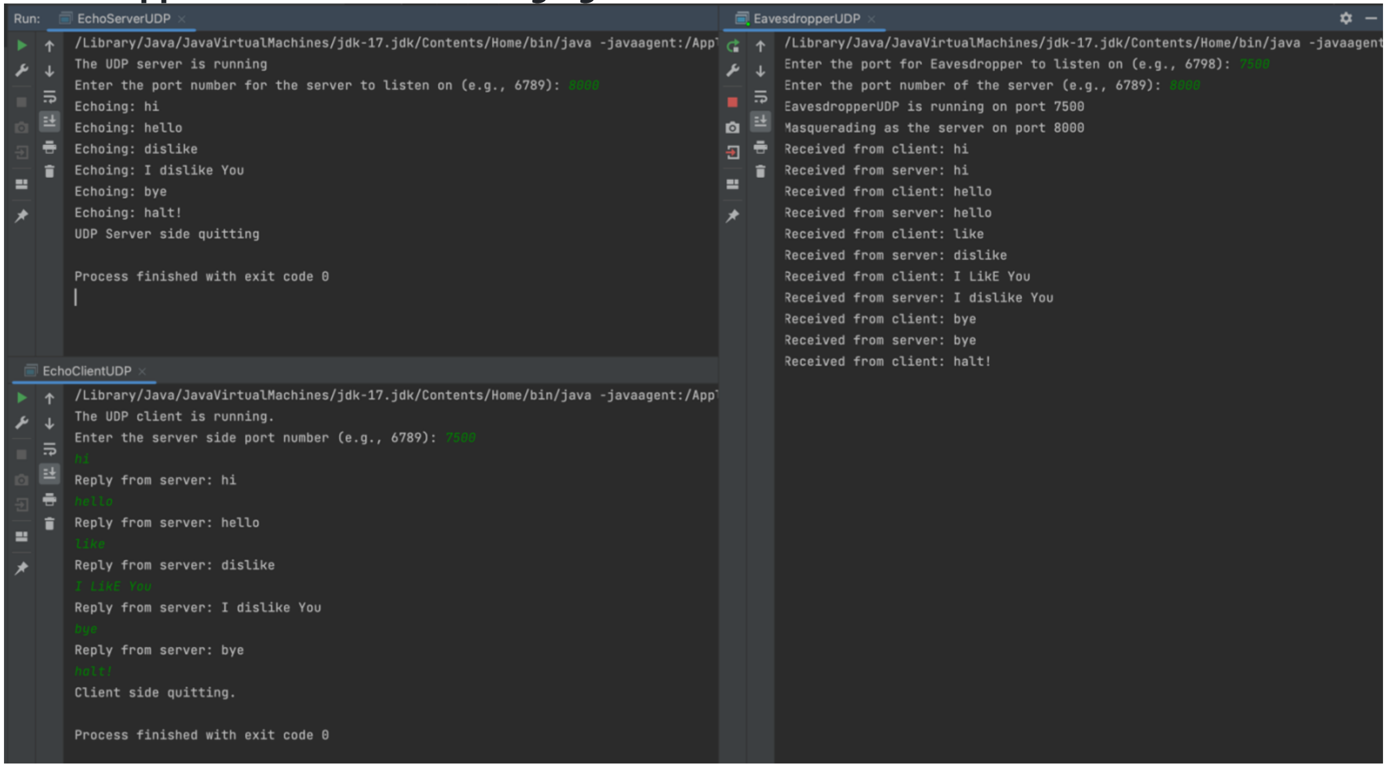
EavesdropperUDP.java program

```
//Aditi Gupta - argupta@andrew.cmu.edu - Project2Task1
// Taken reference from EchoServerUDP.java  and EchoClientUDP.java  from Coulouris
textbook to make the changes
//Combined them to make EavesdropperUDP.java

import java.net.*;
import java.io.*;

public class EavesdropperUDP {
    public static void main(String args[]) {
        DatagramSocket eavesdropperSocket = null;
        DatagramSocket serverSocket = null;

        try {
            // Prompt the user for the ports
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Enter the port for Eavesdropper to listen on (e.g., 6798):
");

            int eavesdropperPort = Integer.parseInt(reader.readLine());

            System.out.print("Enter the port number of the server (e.g., 6789): ");
            int serverPort = Integer.parseInt(reader.readLine());

            // Create a DatagramSocket for eavesdropping
            eavesdropperSocket = new DatagramSocket(eavesdropperPort);

            // Create a DatagramSocket to masquerade as the server
            serverSocket = new DatagramSocket();

            // Announce that the eavesdropper is running
            System.out.println("EavesdropperUDP is running on port " + eavesdropperPort);
            System.out.println("Masquerading as the server on port " + serverPort);
            while (true) {
                // Receive a message from the client
                byte[] clientBuffer = new byte[1000];
                DatagramPacket clientRequest = new DatagramPacket(clientBuffer,
clientBuffer.length);
                eavesdropperSocket.receive(clientRequest);

                // Extract and print the client's message
                int clientRequestLength = clientRequest.getLength();
                byte[] clientRequestData = new byte[clientRequestLength];

                // Code taken from this site:
                // https://stackoverflow.com/questions/5690954/java-how-to-read-an-
unknown-number-of-bytes-from-an-inputstream-socket-socke

                System.arraycopy(clientRequest.getData(), 0, clientRequestData, 0,
clientRequestLength);
                String clientMessage = new String(clientRequestData);
                System.out.println("Received from client: " + clientMessage);


                //Used ChatGPT for this line
                // Replace "like" with "dislike" in the client's message
                clientMessage = clientMessage.replaceAll("(?i)\\blike\\b", "dislike");

                // Forward the modified client's message to the server
                byte[] serverRequestData = clientMessage.getBytes();
                DatagramPacket serverRequest = new DatagramPacket(serverRequestData,
serverRequestData.length,
                        InetAddress.getLocalHost(), serverPort);
                serverSocket.send(serverRequest);

                // Receive the server's reply
                byte[] serverReplyBuffer = new byte[1000];
                DatagramPacket serverReply = new DatagramPacket(serverReplyBuffer,
serverReplyBuffer.length);
                serverSocket.receive(serverReply);

                // Extract and print the server's message
                int serverReplyLength = serverReply.getLength();
                byte[] serverReplyData = new byte[serverReplyLength];

                // Code taken from this site:
                // https://stackoverflow.com/questions/5690954/java-how-to-read-an-
unknown-number-of-bytes-from-an-inputstream-socket-socke

                System.arraycopy(serverReply.getData(), 0, serverReplyData, 0,
serverReplyLength);
                String serverMessage = new String(serverReplyData);
                System.out.println("Received from server: " + serverMessage);

                // Forward the server's reply to the client
                DatagramPacket clientReply = new DatagramPacket(serverReply.getData(),
serverReply.getLength(),
                        clientRequest.getAddress(), clientRequest.getPort());
                eavesdropperSocket.send(clientReply);
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            if (eavesdropperSocket != null)
```

```java
                    eavesdropperSocket.close();
                if (serverSocket != null)
                    serverSocket.close();
            }
        }
}
```

"Project2Task1ThreeConsoles"

Eavesdropper in between and changing like to dislike



Run: EchoServerUDP

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/App
The UDP server is running
Enter the port number for the server to listen on (e.g., 6789): 8000
Echoing: hi
Echoing: hello
Echoing: dislike
Echoing: I dislike You
Echoing: bye
Echoing: halt!
UDP Server side quitting

Process finished with exit code 0
```

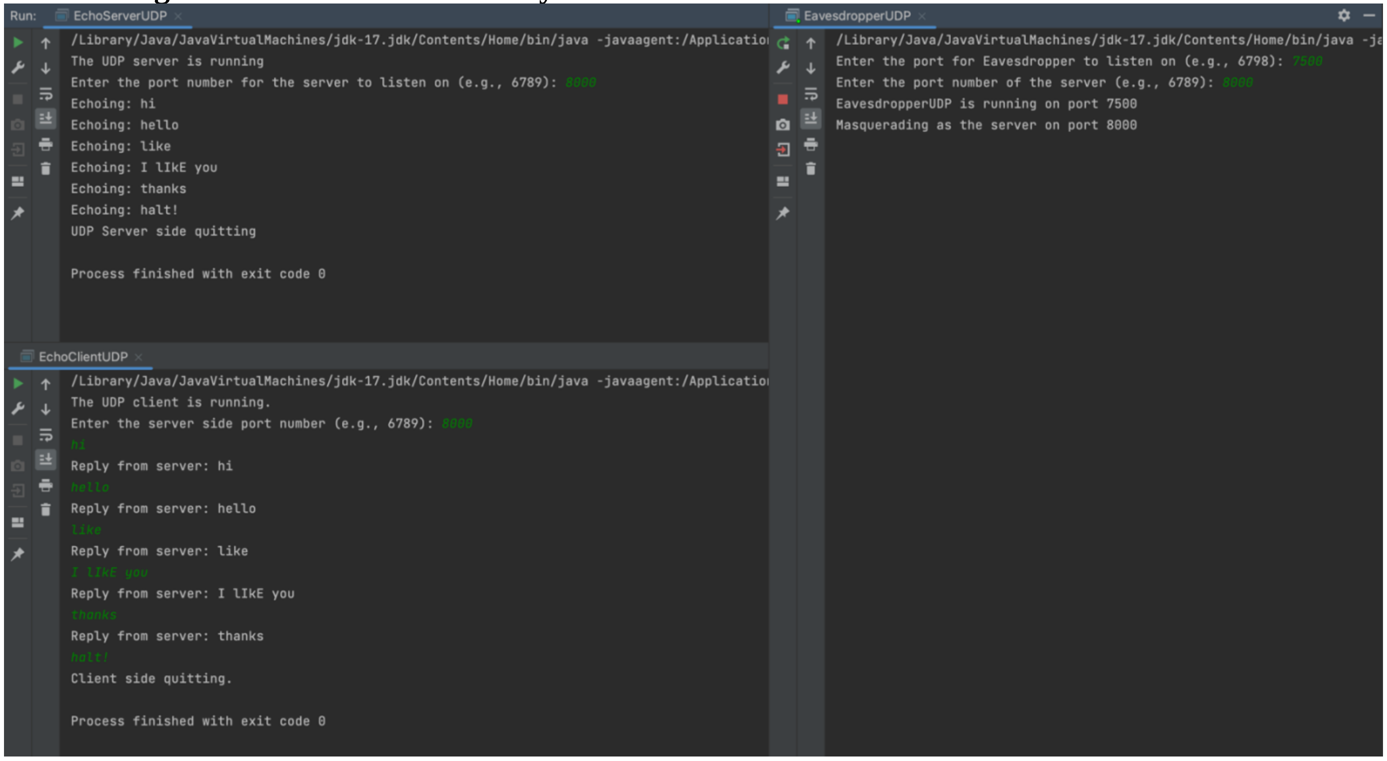
EchoClientUDP

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/App
The UDP client is running.
Enter the server side port number (e.g., 6789): 7500
hi
Reply from server: hi
hello
Reply from server: hello
like
Reply from server: dislike
I LikE You
Reply from server: I dislike You
bye
Reply from server: bye
halt!
Client side quitting.

Process finished with exit code 0
```

EavesdropperUDP

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent
Enter the port for Eavesdropper to listen on (e.g., 6798): 7500
Enter the port number of the server (e.g., 6789): 8000
EavesdropperUDP is running on port 7500
Masquerading as the server on port 8000
Received from client: hi
Received from server: hi
Received from client: hello
Received from server: hello
Received from client: like
Received from server: dislike
Received from client: I LikE You
Received from server: I dislike You
Received from client: bye
Received from server: bye
Received from client: halt!
```

Client being connected to server directly.



Run: EchoServerUDP

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applicatio
The UDP server is running
Enter the port number for the server to listen on (e.g., 6789): 8000
Echoing: hi
Echoing: hello
Echoing: like
Echoing: I lIkE you
Echoing: thanks
Echoing: halt!
UDP Server side quitting

Process finished with exit code 0
```

EchoClientUDP

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applicatio
The UDP client is running.
Enter the server side port number (e.g., 6789): 8000
hi
Reply from server: hi
hello
Reply from server: hello
like
Reply from server: like
I lIkE you
Reply from server: I lIkE you
thanks
Reply from server: thanks
halt!
Client side quitting.

Process finished with exit code 0
```

EavesdropperUDP

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -ja
Enter the port for Eavesdropper to listen on (e.g., 6798): 7500
Enter the port number of the server (e.g., 6789): 8000
EavesdropperUDP is running on port 7500
Masquerading as the server on port 8000
```

Project2Task2
"Project2Task2Client"

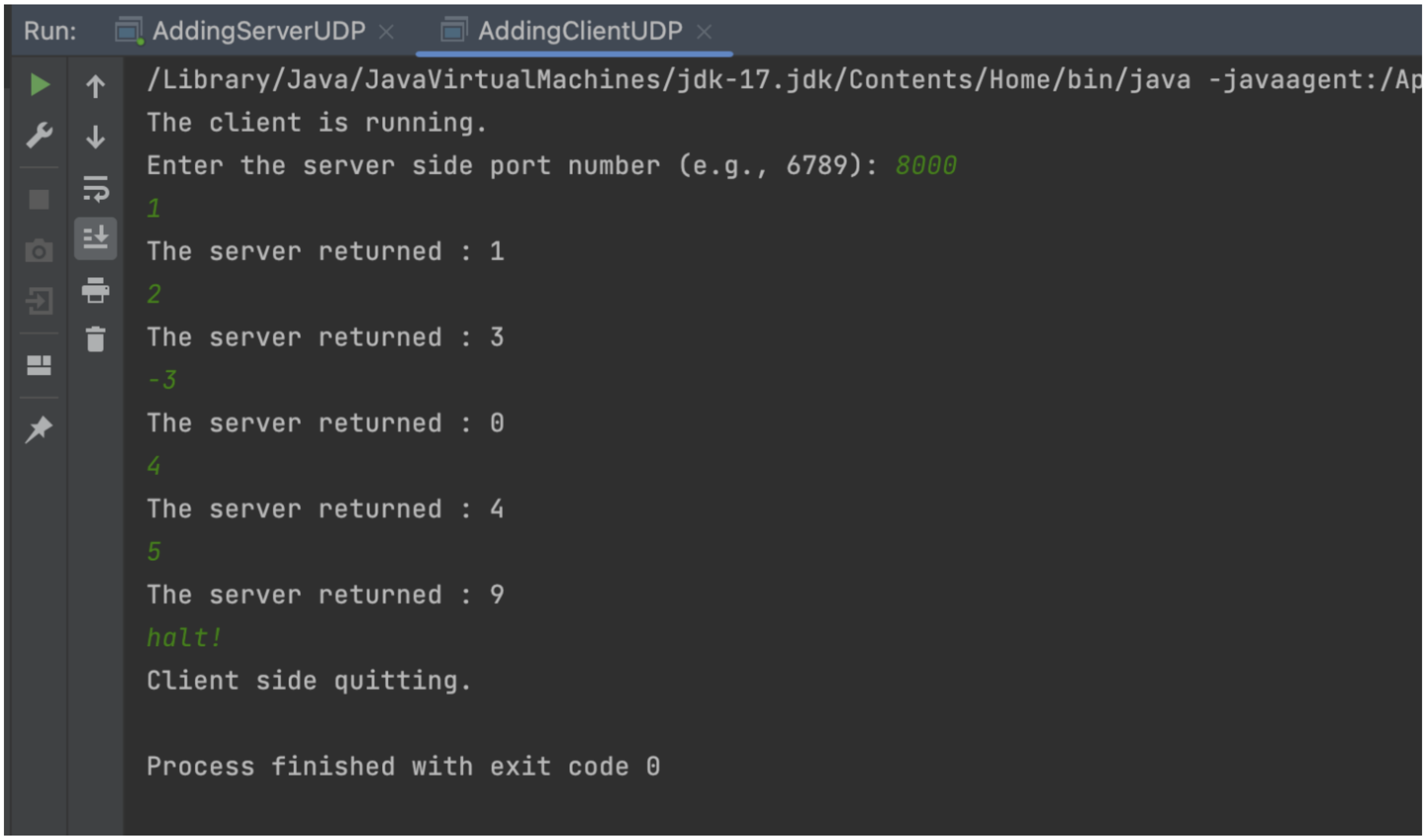
```java
//Aditi Gupta - argupta@andrew.cmu.edu - Project2Task4
// Used EchoClientUDP.java from Coulouris textbook to make the changes
//Used code from Lab 5 for separation of concerns

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.nio.ByteBuffer;

public class AddingClientUDP {
    // Server's listening port number
    private static int serverPort;

    public static void main(String args[]) {
        try {
            // Announce that the client is running
            System.out.println("The client is running.");

            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            // Prompt the user for the server side port number
            System.out.print("Enter the server side port number (e.g., 6789): ");
            serverPort = Integer.parseInt(reader.readLine());

            String nextLine;
            BufferedReader typed = new BufferedReader(new InputStreamReader(System.in));

            // Continue to process user input until "halt!" command is given
            while ((nextLine = typed.readLine()) != null) {
                if (nextLine.trim().equalsIgnoreCase("halt!")) {
                    System.out.println("Client side quitting.");
                    break;
                }

                // Convert the input integer to a byte array and send it to the server
                int num = Integer.parseInt(nextLine);
                int updatedSum = add(num);

                System.out.println("The server returned : " + updatedSum);
            }
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        }
    }

    // Used ChatGPT help to complete this method
    /**
     * Sends the provided integer value to the server and receives an updated sum
     * in response. This method handles the conversion of the integer to a byte array,
     * sending the request, and receiving and processing the server's reply.
     *
     * @param i The integer to be sent to the server for addition.
     * @return The updated sum received from the server, or -1 in case of an error.
     */
    public static int add(int i) {
        DatagramSocket socket = null;
        try {
            // Change the server address to "localhost"
            InetAddress host = InetAddress.getByName("localhost");

            // Create a DatagramSocket for sending and receiving UDP packets
            socket = new DatagramSocket();

            // Convert the integer 'i' to a 4-byte array
            byte[] intBytes = new byte[4];
            intBytes[0] = (byte) (i >> 24);
            intBytes[1] = (byte) (i >> 16);
            intBytes[2] = (byte) (i >> 8);
            intBytes[3] = (byte) i;

            // Send the byte array to the server
            DatagramPacket request = new DatagramPacket(intBytes, intBytes.length, host,
serverPort);
            socket.send(request);

            // Receive the reply from the server (containing the updated sum)
            byte[] buffer = new byte[4];
            DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
            socket.receive(reply);

            // Convert the received byte array back to an integer (the updated sum)
            // Used ChatGPT to get this line of code
            int updatedSum = ByteBuffer.wrap(reply.getData()).getInt();

            return updatedSum;
        } catch (IOException e) {
            System.out.println("Error in add method: " + e.getMessage());
            return -1; // Return an error value
        } finally {
            if (socket != null)
                socket.close();
        }
    }
}
```

"Project2Task2Server"

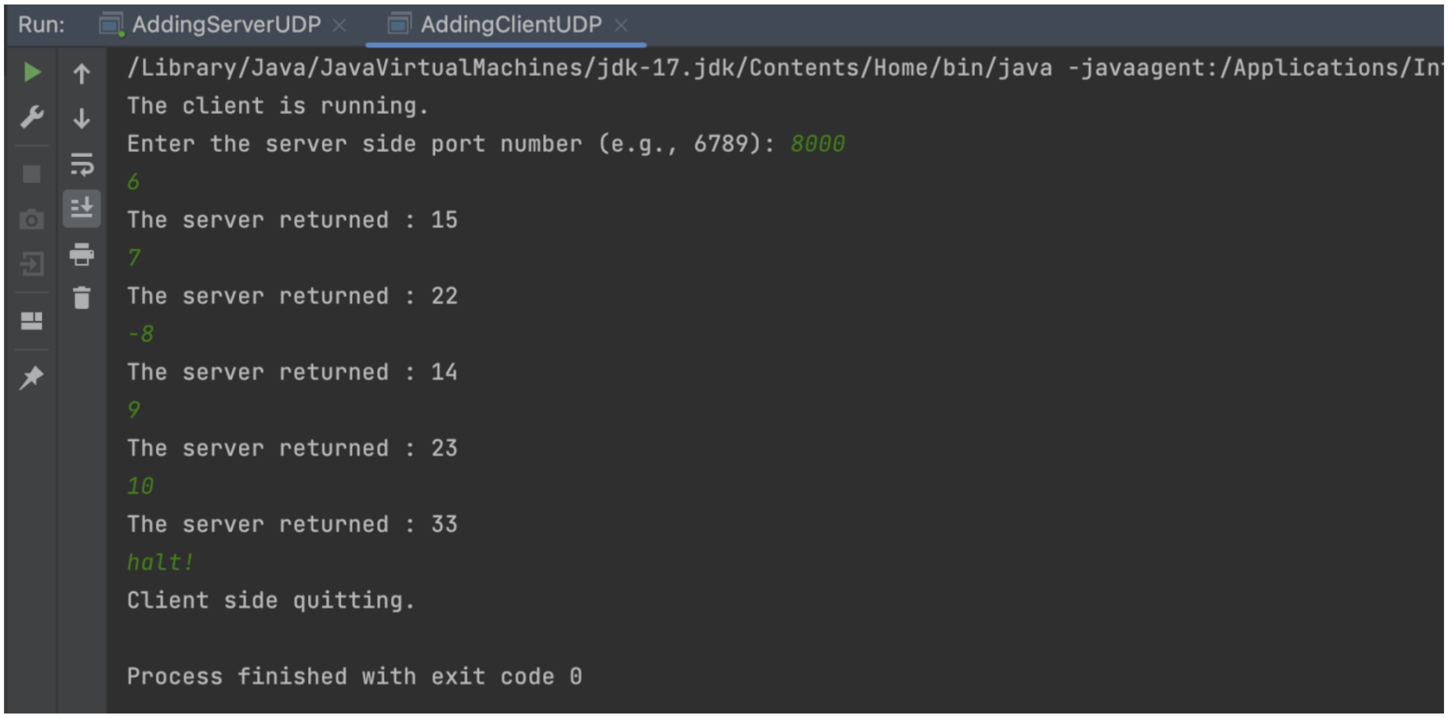
```
//Aditi Gupta - argupta@andrew.cmu.edu - Project2Task2
// Used Lab5 for separation of concerns
// Used EchoServerUDP.java from Coulouris textbook to make the changes

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.nio.ByteBuffer;

public class AddingServerUDP {
    // Static variable to store the shared integer sum
    private static int sum = 0;

    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        byte[] buffer = new byte[4]; // Use a 4-byte buffer for integers
```

```java
        try {
            // Announce that the server is running
            System.out.println("Server started");

            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            // Prompt the user for the port number to listen on
            System.out.print("Enter the port number for the server to listen on (e.g., 
6789): ");
            int serverPort = Integer.parseInt(reader.readLine());

            // Create a DatagramSocket to listen for incoming UDP packets
            aSocket = new DatagramSocket(serverPort);
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);
            while (true) {
                // Receive an incoming UDP packet (request) from a client
                aSocket.receive(request);

                // Used ChatGPT to get this line of code
                // Extract the integer value from the received packet
                int num = ByteBuffer.wrap(request.getData()).getInt();

                // Print the received request and the integer value
                System.out.println("Adding: " + num + " to "+sum);

                // Call method add with the received integer and get the updated sum
                int updatedSum = add(num);

                // Prepare and send the updated sum as a reply to the client
                // Used ChatGPT to get this line of code
                byte[] replyData = ByteBuffer.allocate(4).putInt(updatedSum).array();
                DatagramPacket reply = new DatagramPacket(replyData, replyData.length, 
request.getAddress(), request.getPort());
                aSocket.send(reply);

                // Print the new sum
                System.out.println("Returning sum of " + updatedSum + " to client");
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            if (aSocket != null)
                aSocket.close();
        }
    }

    /**
     * Adds the specified integer value to the shared sum.
     *
     * @param i The integer value to be added to the sum.
     * @return The updated sum after the addition.
     */
    public static int add(int i) {
        sum += i;
        return sum;
    }
}
```

"Project2Task2ClientConsole"

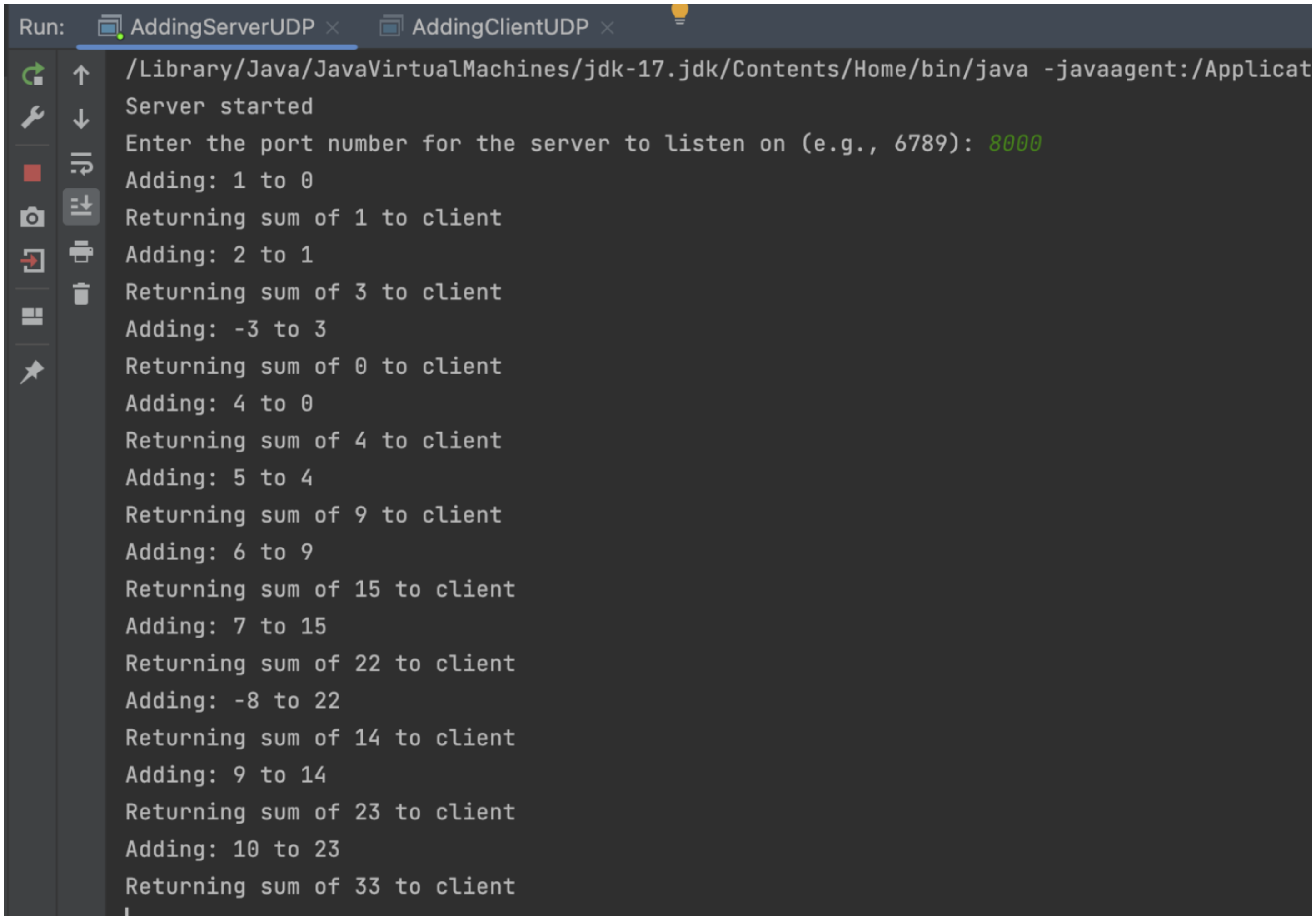


```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Ap
The client is running.
Enter the server side port number (e.g., 6789): 8000
1
The server returned : 1
2
The server returned : 3
-3
The server returned : 0
4
The server returned : 4
5
The server returned : 9
halt!
Client side quitting.

Process finished with exit code 0
```



```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/In
The client is running.
Enter the server side port number (e.g., 6789): 8000
6
The server returned : 15
7
The server returned : 22
-8
The server returned : 14
9
The server returned : 23
10
The server returned : 33
halt!
Client side quitting.

Process finished with exit code 0
```

"Project2Task2ServerConsole"



```
Run:   AddingServerUDP ×   AddingClientUDP ×
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applicat
Server started
Enter the port number for the server to listen on (e.g., 6789): 8000
Adding: 1 to 0
Returning sum of 1 to client
Adding: 2 to 1
Returning sum of 3 to client
Adding: -3 to 3
Returning sum of 0 to client
Adding: 4 to 0
Returning sum of 4 to client
Adding: 5 to 4
Returning sum of 9 to client
Adding: 6 to 9
Returning sum of 15 to client
Adding: 7 to 15
Returning sum of 22 to client
Adding: -8 to 22
Returning sum of 14 to client
Adding: 9 to 14
Returning sum of 23 to client
Adding: 10 to 23
Returning sum of 33 to client
```

Project2Task3

"Project2Task3Client"

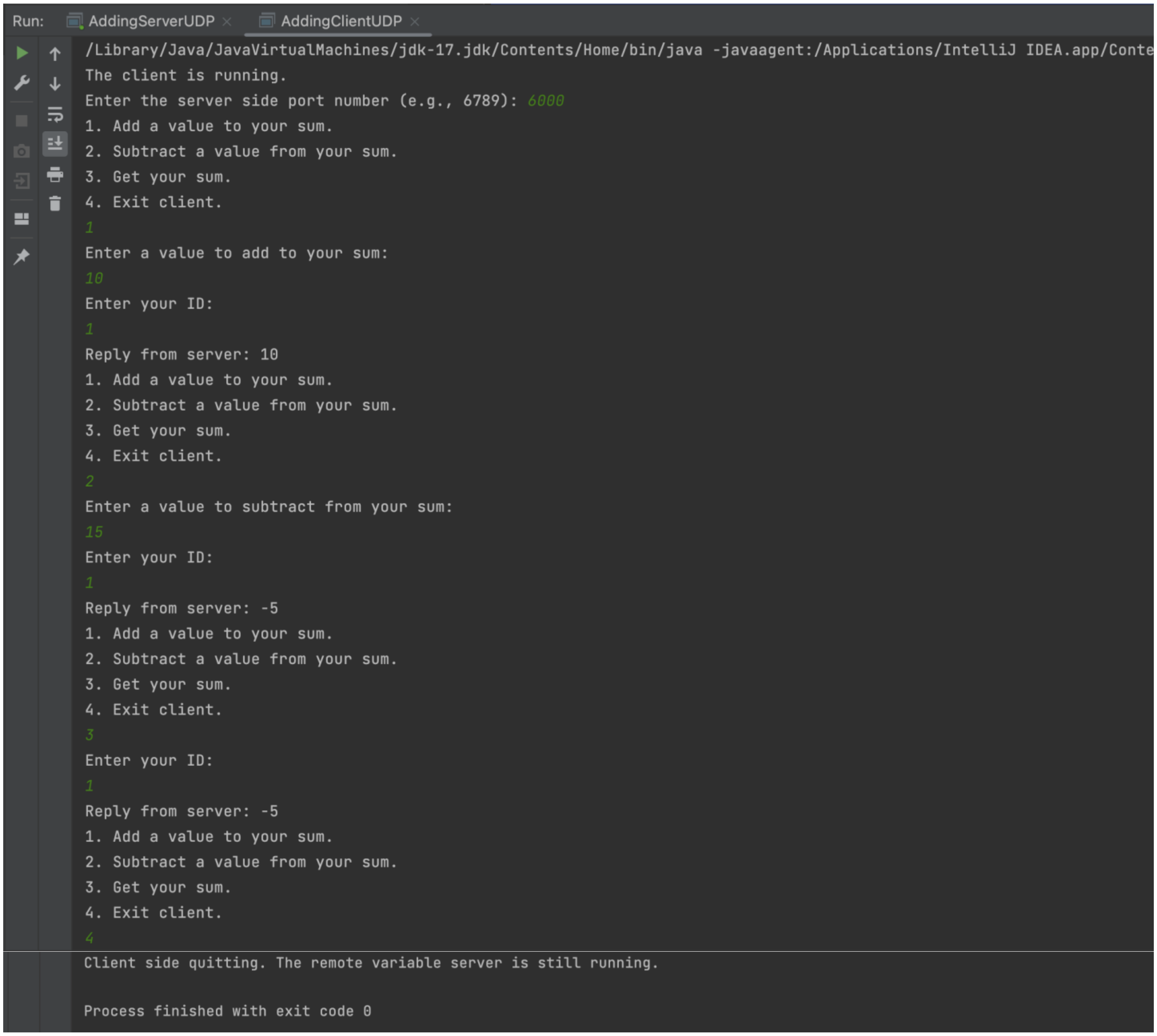
```java
//Aditi Gupta - argupta@andrew.cmu.edu - Project2Task3
// Used code from EchoClientUDP.java from Coulouris textbook to make the changes
//Used code from Lab 5 for separation of concerns

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class AddingClientUDP {
    // Variable to store the server's port number
    private static int serverPort;

    public static void main(String args[]) {
        try {
            // Announce that the client is running
            System.out.println("The client is running.");

            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            // Prompt the user for the server side port number
            System.out.print("Enter the server side port number (e.g., 6789): ");
            serverPort = Integer.parseInt(reader.readLine());

            // Display the client's main menu and capture the chosen option
            String nextLine, value, id, total;
            BufferedReader typed = new BufferedReader(new InputStreamReader(System.in));
            nextLine = menu();

            while (nextLine != null) {
                if (nextLine.trim().equalsIgnoreCase("1")) {
                    // Option 1: Add a value to the sum
                    System.out.println("Enter a value to add to your sum:");
                    value = typed.readLine();
                    System.out.println("Enter your ID:");
                    id = typed.readLine();
                    String add = "add";
                    total = id + "," + value + "," + add;
                    add(total); // Send the request to the server
                    nextLine = menu(); // Show the menu again
                } else if (nextLine.trim().equalsIgnoreCase("2")) {
                    // Option 2: Subtract a value from the sum
                    System.out.println("Enter a value to subtract from your sum:");
                    value = typed.readLine();
                    System.out.println("Enter your ID:");
                    id = typed.readLine();
                    String diff = "diff";
                    total = id + "," + value + "," + diff;
                    add(total); // Send the request to the server
                    nextLine = menu(); // Show the menu again
                } else if (nextLine.trim().equalsIgnoreCase("3")) {
                    // Option 3: Get the current sum
                    int num = 0;
                    System.out.println("Enter your ID:");
                    id = typed.readLine();
                    String get = "get";
                    total = id + "," + num + "," + get;
                    add(total); // Send the request to the server
                    nextLine = menu(); // Show the menu again
                } else if (nextLine.trim().equalsIgnoreCase("4")) {
                    // Option 4: Exit the client
                    System.out.println("Client side quitting. The remote variable server
is still running.");
                    break;
                } else {
                    System.out.println("Invalid option. Please choose a valid option (1-
4).");
                    nextLine = menu(); // Show the menu again
                }
            }
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        }
    }

    // Method to send a request to the server
    // Used Lab 5 for help for this method
    public static void add(String i) {
        DatagramSocket socket = null;
        try {
            // Change the server address to "localhost"
            InetAddress host = InetAddress.getByName("localhost");
            // Create a DatagramSocket for sending and receiving UDP packets
            socket = new DatagramSocket();

            // Convert the data to a byte array and send to the server
            byte[] m = i.getBytes();

            // Send the byte array to the server
            DatagramPacket request = new DatagramPacket(m, m.length, host, serverPort);
            socket.send(request);

            // Receive the reply from the server (containing the updated sum)
            byte[] buffer = new byte[1000];
            DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
            socket.receive(reply);

            // Display the server's reply
            System.out.println("Reply from server: " + new String(reply.getData(), 0,
reply.getLength()));

        } catch (IOException e) {
            System.out.println("Error in add method: " + e.getMessage());
        } finally {
            if (socket != null)
                socket.close();
        }
    }

    /**
     * Displays the menu of available operations to the user and captures their input.
     *
     * @return The user's menu choice.
     * @throws IOException If there's an error reading the user's input.
     */
    // Method to display the client menu and get user input
    public static String menu() throws IOException {
        System.out.println("1. Add a value to your sum.");
        System.out.println("2. Subtract a value from your sum.");
        System.out.println("3. Get your sum.");
        System.out.println("4. Exit client.");
        BufferedReader typed = new BufferedReader(new InputStreamReader(System.in));
        String nextLine = typed.readLine();
        return nextLine;
    }
}
```

"Project2Task3Server"

```
//Aditi Gupta - argupta@andrew.cmu.edu - Project2Task3
// Used code from EchoServerUDP.java from Coulouris textbook to make the changes

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.TreeMap;

public class AddingServerUDP {
    private static int sum = 0, diff=0; // Variable to store the sum/ difference of
values
```

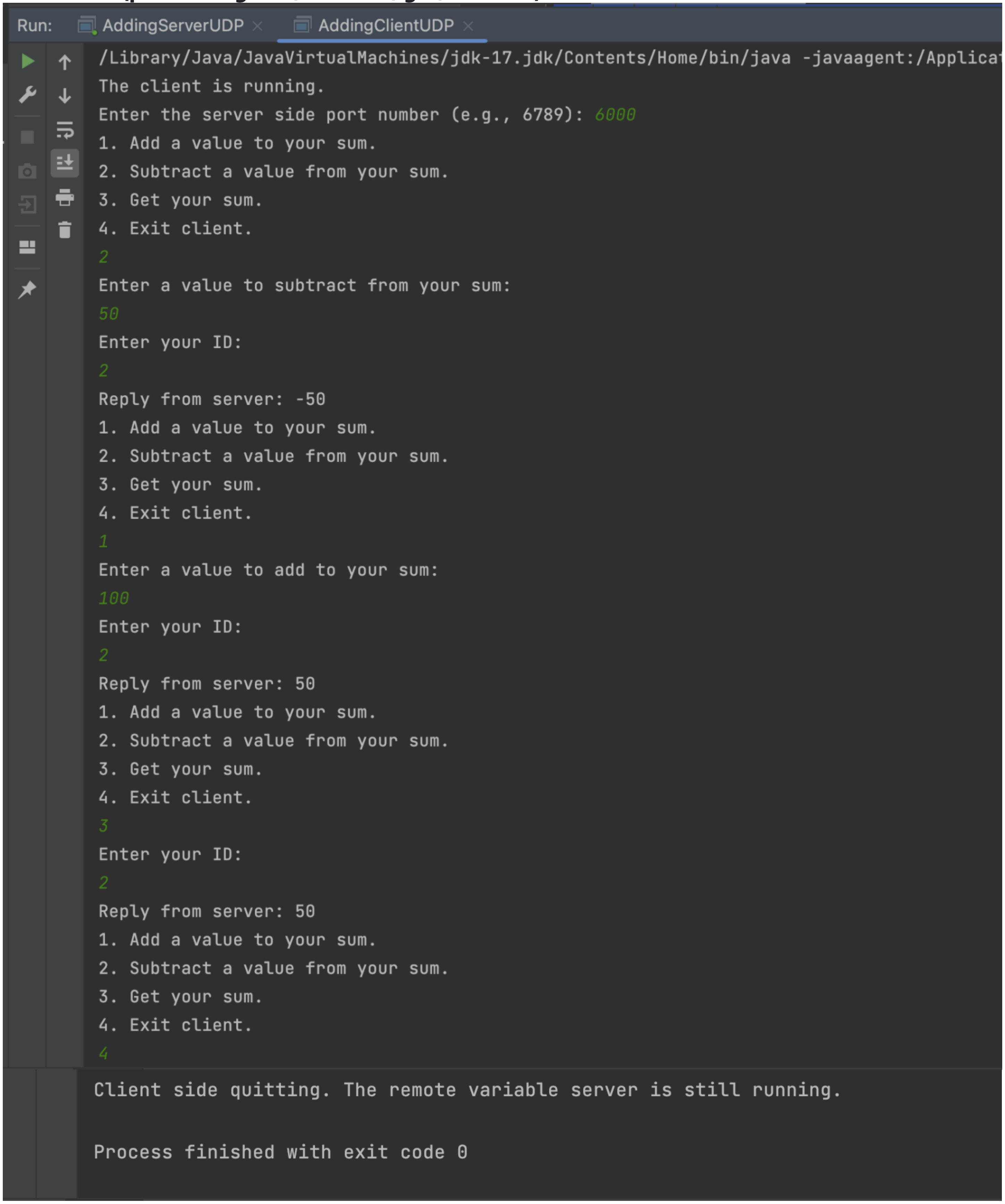
```
    // Treemap to store the shared variable for each client identified by a unique ID
    private static TreeMap<Integer, Integer> map = new TreeMap<>();
    public static void main(String args[]) {
        DatagramSocket aSocket = null; // UDP socket for communication
        byte[] buffer = new byte[2046]; // Buffer for receiving incoming UDP packets

        try {
            // Announce that the server is running
            System.out.println("Server started");

            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            // Prompt the user for the port number to listen on
            System.out.print("Enter the port number for the server to listen on (e.g.,
6789): ");
            int serverPort = Integer.parseInt(reader.readLine());

            // Create a DatagramSocket to listen for incoming UDP packets
            aSocket = new DatagramSocket(serverPort);
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);

            while (true) {
                // Receive an incoming UDP packet (request) from a client
                aSocket.receive(request);

                // Create a new byte array with length equal to the message length
                byte[] data = new byte[request.getLength()];

                // Code taken from this site:
                // https://stackoverflow.com/questions/5690954/java-how-to-read-an-
unknown-number-of-bytes-from-an-inputstream-socket-socke
                // Copy the message from request to data array
                System.arraycopy(request.getData(), request.getOffset(), data, 0,
request.getLength());

                // Split the received data into components: ID, value, and operation
                // Used ChatGPT for this line
                String[] elements = new String(data).split(",");
                int id = Integer.valueOf(elements[0]);

                // Initialize the shared variable for new clients
                if (!map.containsKey(id)) {
                    map.put(id, 0);
                }

                int value = Integer.valueOf(elements[1]);
                String operation = elements[2];
                System.out.println("Visitor's ID: " + id );
                System.out.println("Operation Requested: " + operation);

                // Perform the requested operation (addition or subtraction)
                if (operation.equalsIgnoreCase("add")||
operation.equalsIgnoreCase("get")) {
                    sum = add(map.get(id), value);
                } else {
                    sum = diff(map.get(id), value);
                }

                // Update the value associated with the client ID in the map
                map.put(id, sum);

                // Print the updated value associated with the client ID
                System.out.println("Value associated with ID " + id + ": " +
map.get(id));

                // Convert the sum to a byte array (assuming sum is an int)
                // Used ChatGPT to get this line of code
                byte[] sumBytes = String.valueOf(sum).getBytes();

                // Create a DatagramPacket with the sumBytes, client's address, and port
                DatagramPacket reply = new DatagramPacket(sumBytes, sumBytes.length,
request.getAddress(), request.getPort());

                // Send the DatagramPacket back to the client
                aSocket.send(reply);

                // Print the new sum
                System.out.println("The result is " + sum);
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            if (aSocket != null)
                aSocket.close();
        }
    }

    /**
     * Adds the given value to the initial value and returns the sum.
     *
     * @param i Initial value.
     * @param value Value to be added.
     * @return Resultant sum.
     */
    public static int add(int i, int value) {
        sum = i+value;
        return sum;
    }

    /**
     * Subtracts the given value from the initial value and returns the difference.
     *
     * @param i Initial value.
     * @param value Value to be subtracted.
     * @return Resultant difference.
     */
    public static int diff(int i, int value) {
        diff = i-value;
        return diff;
    }

}
```

"Project2Task3ClientConsole"
Client id: 1 (performing add, subtract, get, and exit)


Run: AddingServerUDP AddingClientUDP

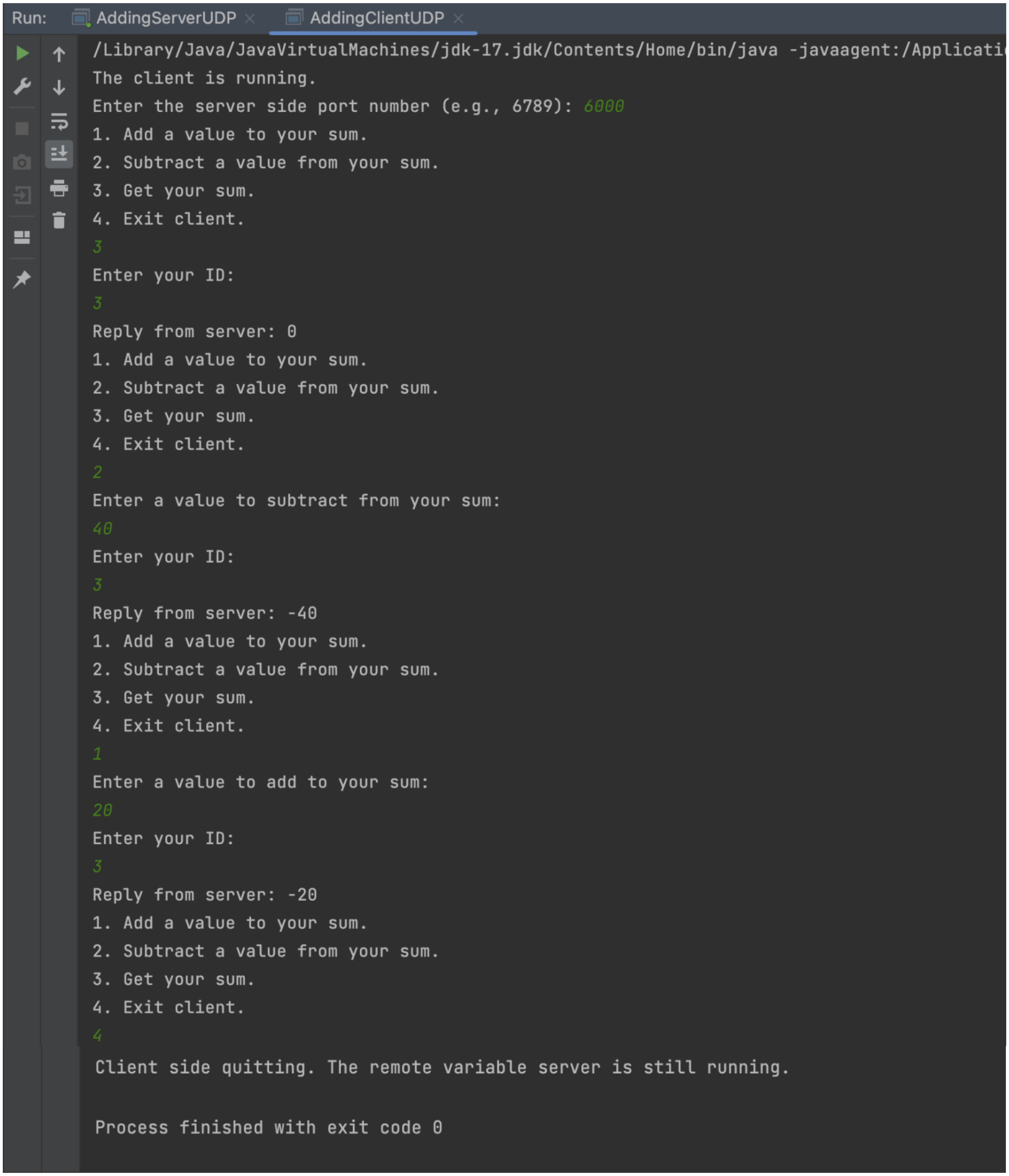
```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Conte
The client is running.
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter a value to add to your sum:
10
Enter your ID:
1
Reply from server: 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter a value to subtract from your sum:
15
Enter your ID:
1
Reply from server: -5
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
1
Reply from server: -5
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Client id: 2 (performing add, subtract, get, and exit)



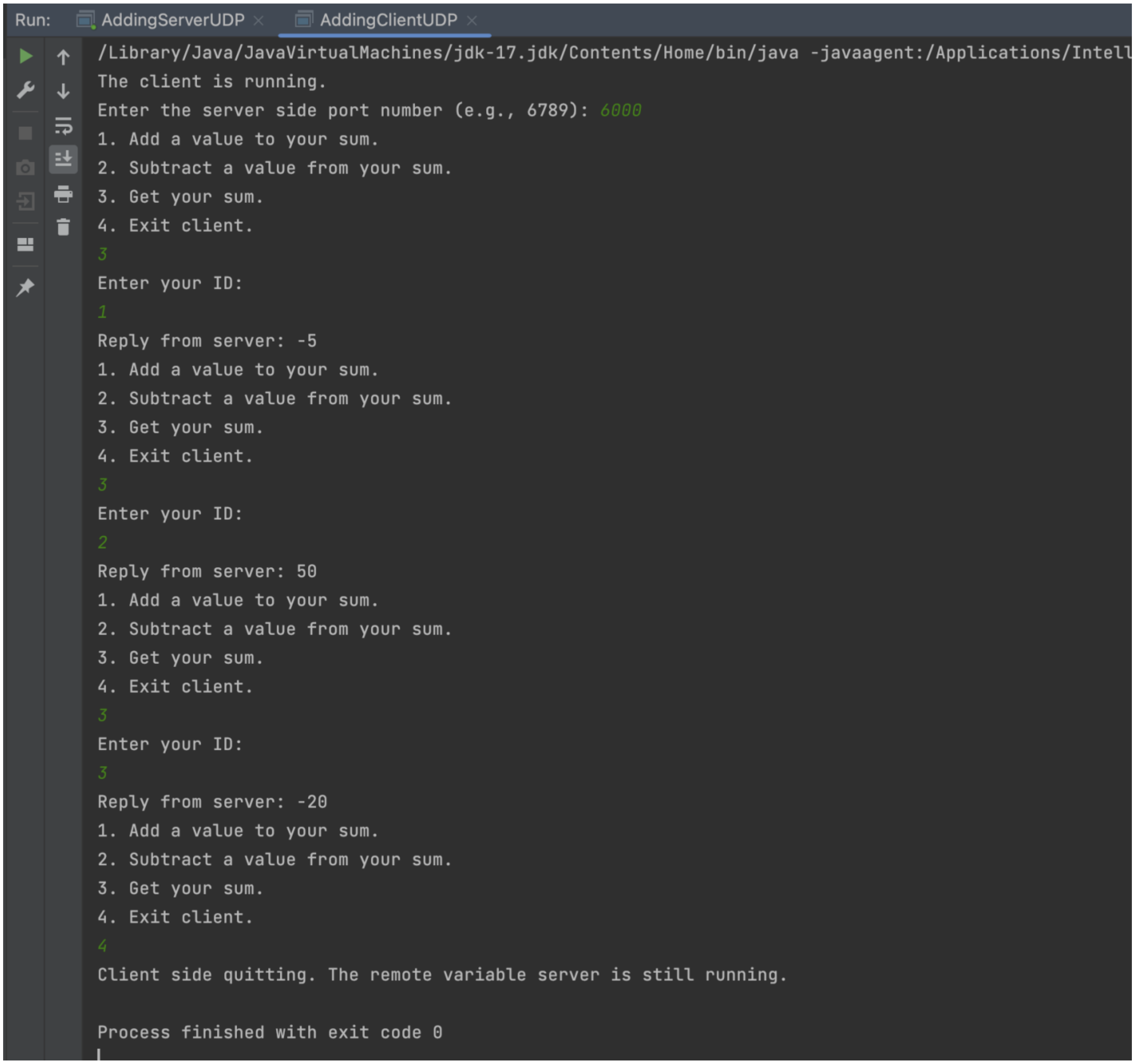
```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applicat
The client is running.
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter a value to subtract from your sum:
50
Enter your ID:
2
Reply from server: -50
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter a value to add to your sum:
100
Enter your ID:
2
Reply from server: 50
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
2
Reply from server: 50
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Client id: 3 (performing add, subtract, get, and exit)



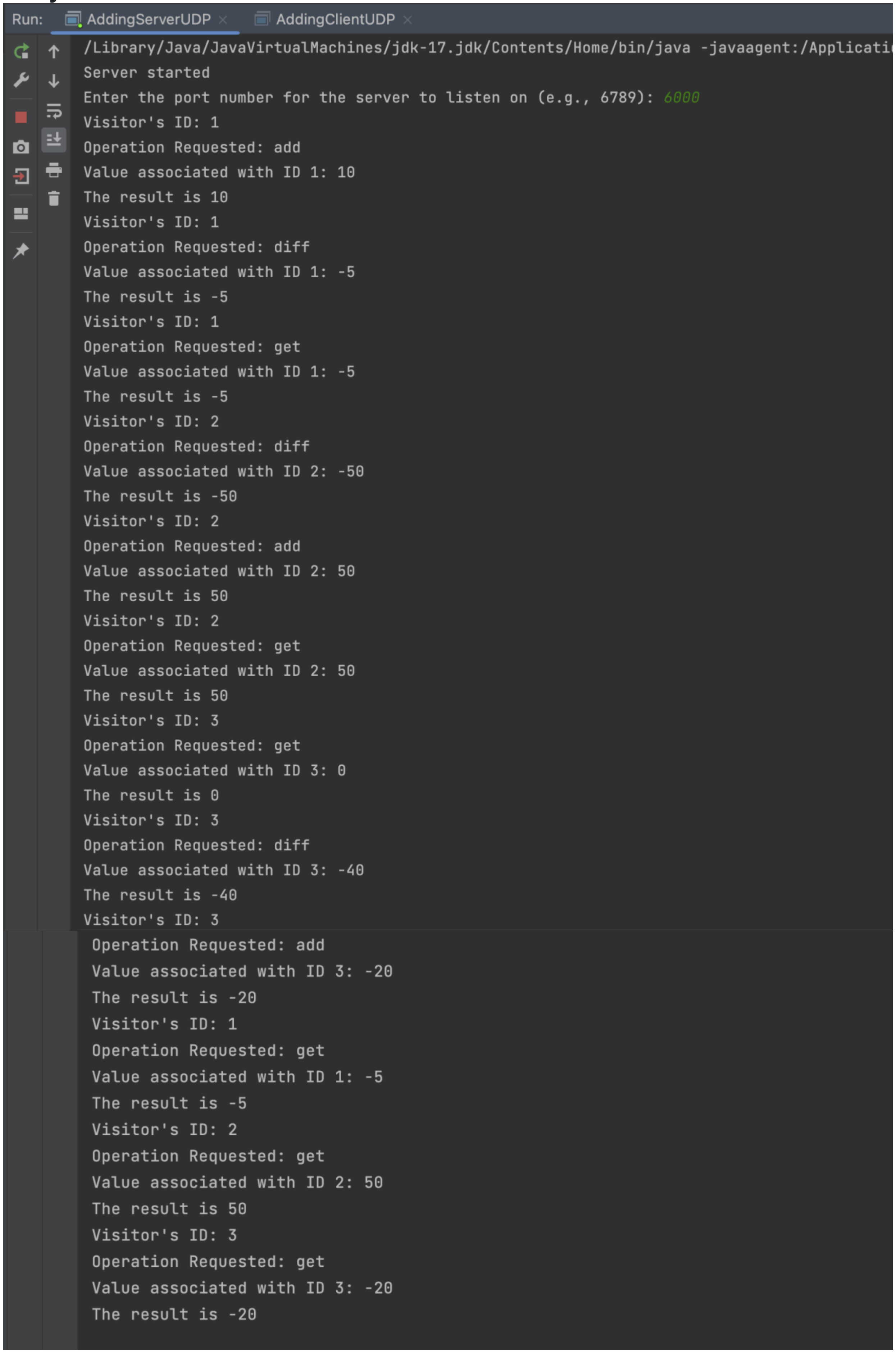
```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applicati
The client is running.
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
3
Reply from server: 0
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter a value to subtract from your sum:
40
Enter your ID:
3
Reply from server: -40
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter a value to add to your sum:
20
Enter your ID:
3
Reply from server: -20
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Client being stopped and re-run a second time with get requests from each of the three clients.



```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/Intell
The client is running.
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
1
Reply from server: -5
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
2
Reply from server: 50
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
3
Reply from server: -20
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

"Project2Task3ServerConsole"



```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applicati
Server started
Enter the port number for the server to listen on (e.g., 6789): 6000
Visitor's ID: 1
Operation Requested: add
Value associated with ID 1: 10
The result is 10
Visitor's ID: 1
Operation Requested: diff
Value associated with ID 1: -5
The result is -5
Visitor's ID: 1
Operation Requested: get
Value associated with ID 1: -5
The result is -5
Visitor's ID: 2
Operation Requested: diff
Value associated with ID 2: -50
The result is -50
Visitor's ID: 2
Operation Requested: add
Value associated with ID 2: 50
The result is 50
Visitor's ID: 2
Operation Requested: get
Value associated with ID 2: 50
The result is 50
Visitor's ID: 3
Operation Requested: get
Value associated with ID 3: 0
The result is 0
Visitor's ID: 3
Operation Requested: diff
Value associated with ID 3: -40
The result is -40
Visitor's ID: 3
Operation Requested: add
Value associated with ID 3: -20
The result is -20
Visitor's ID: 1
Operation Requested: get
Value associated with ID 1: -5
The result is -5
Visitor's ID: 2
Operation Requested: get
Value associated with ID 2: 50
The result is 50
Visitor's ID: 3
Operation Requested: get
Value associated with ID 3: -20
The result is -20
```

Project2Task4
"Project2Task4Client"

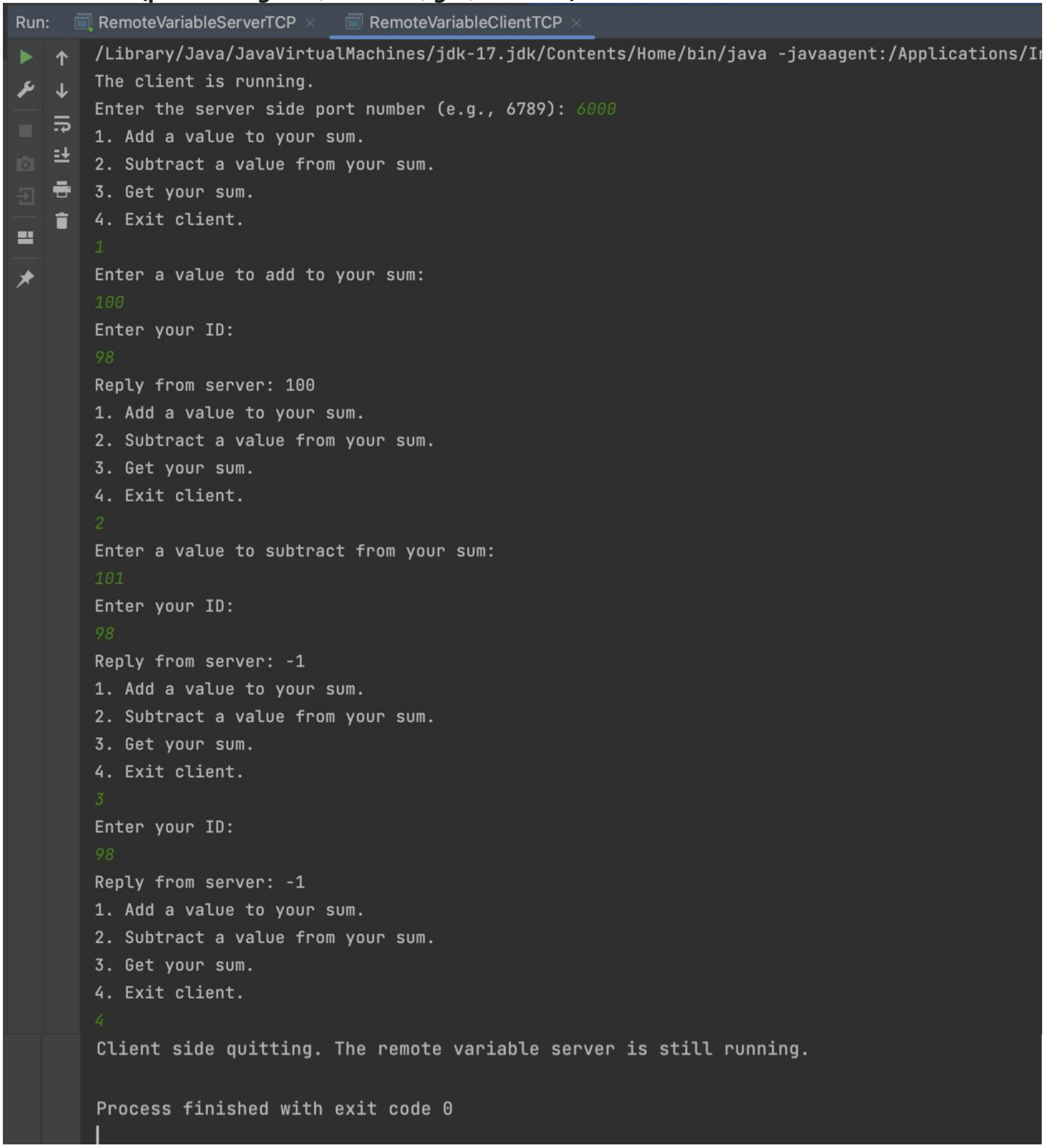
```
//Aditi Gupta - argupta@andrew.cmu.edu - Project2Task4
//Took code from EchoClientTCP.java from Coulouris textbook to make the changes
//Used code from Lab 5 for separation of concerns

import java.io.*;
import java.net.Socket;

public class RemoteVariableClientTCP {
    private static int serverPort; // Port number to connect to the server

    public static void main(String[] args) {
        try {
            // Announce that the client is running
            System.out.println("The client is running.");

            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            // Prompt the user for the server side port number
            System.out.print("Enter the server side port number (e.g., 6789): ");
            serverPort = Integer.parseInt(reader.readLine());
            String total = " "; // Stores the formatted message to be sent to the server
            while (true) {
                try {
                    // Display the menu and get user input
                    String nextLine = menu(reader);

                    // https://gist.github.com/chatton/8955d2f96f58f6082bde14e7c33f69a6

                    if (nextLine.trim().equalsIgnoreCase("1")) {
                        // Option 1: Add a value to the sum
                        System.out.println("Enter a value to add to your sum:");
                        String value = reader.readLine();
                        System.out.println("Enter your ID:");
                        String id = reader.readLine();
                        String add = "add";
                         total = id + "," + value + "," + add;
                    } else if (nextLine.trim().equalsIgnoreCase("2")) {
                        // Option 2: Subtract a value from the sum
                        System.out.println("Enter a value to subtract from your sum:");
                        String value = reader.readLine();
                        System.out.println("Enter your ID:");
                        String id = reader.readLine();
                        String diff = "diff";
                         total = id + "," + value + "," + diff;
                    } else if (nextLine.trim().equalsIgnoreCase("3")) {
                        // Option 3: Get the current sum
                        int num = 0;
                        System.out.println("Enter your ID:");
                        String id = reader.readLine();
                        String get = "get";
                         total = id + "," + num + "," + get;
                    } else if (nextLine.trim().equalsIgnoreCase("4")) {
                        // Option 4: Exit the client
                        System.out.println("Client side quitting. The remote variable
server is still running.");
                        break;
                    } else {
                        System.out.println("Invalid option. Please choose a valid option
(1-4).");
                    }

                    // Read and display the server's reply
                    String reply = communicateWithServer(total);
                    System.out.println("Reply from server: " + reply);
                } catch (IOException e) {
                    System.out.println("Error in client socket: " + e.getMessage());
                }
            }
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        }
    }

    // Method to encapsulate communication with the server
    private static String communicateWithServer(String request) {
        try (Socket socket = new Socket("localhost", serverPort);
             BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

            out.println(request); // Send the request to the server
            return in.readLine(); // Read and return the server's reply
        } catch (IOException e) {
            return "Error in client socket: " + e.getMessage();
        }
    }

    // Method to display the client menu and get user input
    public static String menu(BufferedReader reader) throws IOException {
        System.out.println("1. Add a value to your sum.");
        System.out.println("2. Subtract a value from your sum.");
        System.out.println("3. Get your sum.");
        System.out.println("4. Exit client.");
        String nextLine = reader.readLine();
        return nextLine;
    }
}
```

"Project2Task4Server"

```
//Aditi Gupta - argupta@andrew.cmu.edu - Project2Task4
//Took help from EchoServerTCP.java from Coulouris textbook to make the changes
//Used code from Lab 5 for separation of concerns

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.TreeMap;

public class RemoteVariableServerTCP {
    private static int sum = 0, diff=0; // Variable to store the sum/difference of values

    public static void main(String[] args) {

        // Create a ServerSocket for accepting incoming client connections
        ServerSocket serverSocket = null;
```

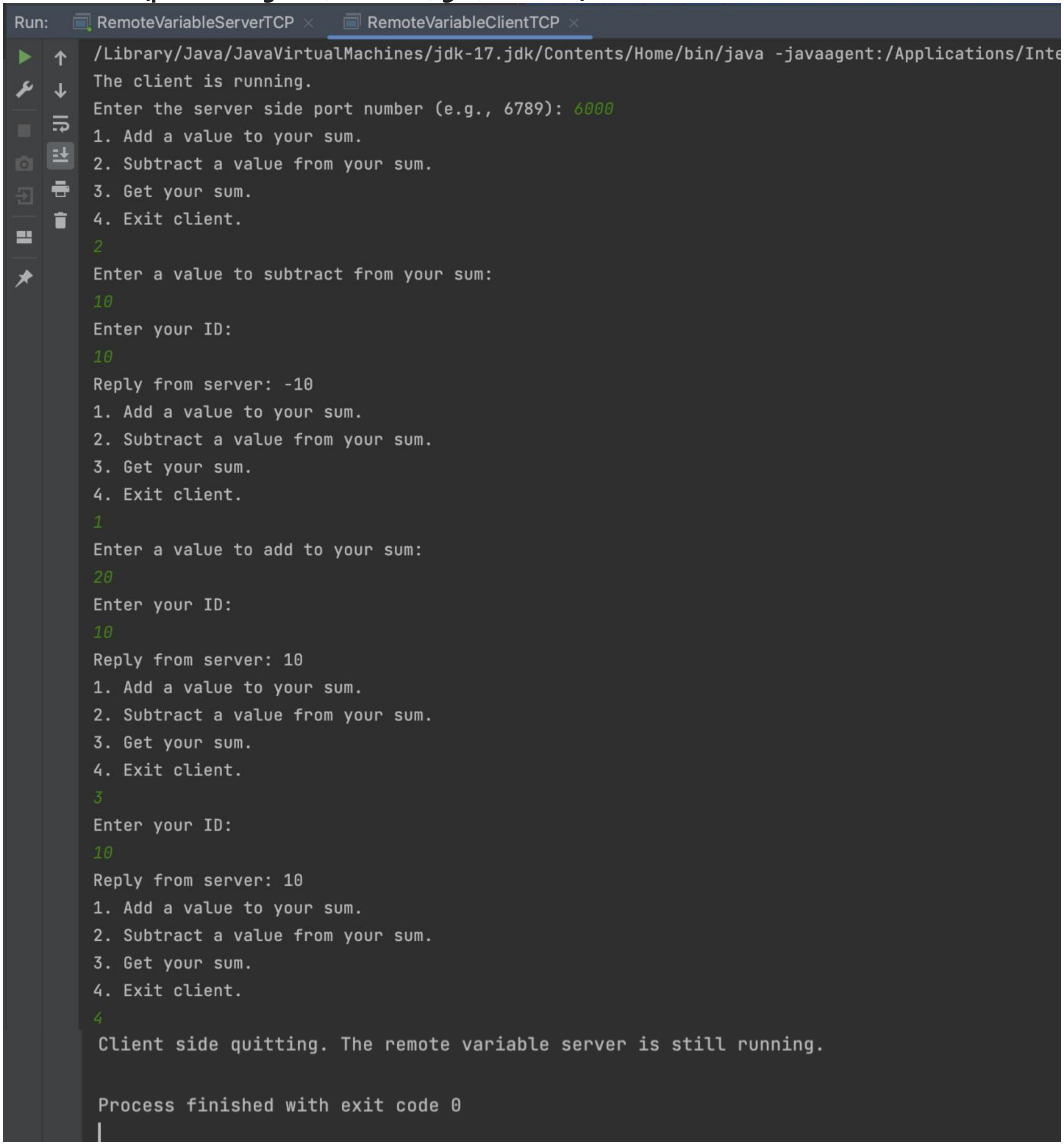
```java
        // HashMap to store the shared variable for each client identified by a unique ID
        TreeMap<Integer, Integer> map = new TreeMap<>();

        try {
            // Announce that the server is running
            System.out.println("Server started");

            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            // Prompt the user for the port number to listen on
            System.out.print("Enter the port number for the server to listen on (e.g.,
6789): ");
            int serverPort = Integer.parseInt(reader.readLine());

            // Create a ServerSocket to listen for incoming TCP connections
            serverSocket = new ServerSocket(serverPort);

            while (true) {
                // Wait for a client to connect
                Socket clientSocket = serverSocket.accept();

                // Read client request
                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
                String request = in.readLine();

                if (request == null) {
                    continue;
                }
                // Parse and process the client request
                String[] elements = request.split(",");
                int id = Integer.valueOf(elements[0]);

                // Initialize the shared variable for new clients

                if (!map.containsKey(id)) {
                    map.put(id, 0);
                }
                int value = Integer.valueOf(elements[1]);
                String operation = elements[2];
                System.out.println("Visitor's ID: " + id );
                System.out.println("Operation Requested: " + operation);
                // Perform the requested operation (addition or subtraction)
                if (operation.equalsIgnoreCase("add")||
operation.equalsIgnoreCase("get")) {
                    sum = add(map.get(id), value);
                } else {
                    sum = diff(map.get(id), value);
                }

                // Update the value associated with the client ID in the map
                map.put(id, sum);

                // Print the updated value associated with the client ID
                System.out.println("Value associated with ID " + id + ": " +
map.get(id));

                // Print the result of the operation (sum)
                System.out.println("Sum: " + sum);

                // Send the result back to the client
                out.println(sum);

                // Close the client socket when done
                clientSocket.close();
            }
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            try {
                if (serverSocket != null)
                    serverSocket.close();
            } catch (IOException e) {
                System.out.println("Error closing server socket: " + e.getMessage());
            }
        }
    }

    /**
     * Adds the provided value to the initial value and returns the sum.
     *
     * @param i Initial value.
     * @param value Value to be added.
     * @return Resultant sum.
     */

    public static int add(int i, int value) {
        sum = i+value;
        return sum;
    }
    /**
     * Subtracts the provided value from the initial value and returns the difference.
     *
     * @param i Initial value.
     * @param value Value to be subtracted.
     * @return Resultant difference.
     */
    public static int diff(int i, int value) {
        diff = i-value;
        return diff;
    }
}
```

"Project2Task4ClientConsole"
Client id: 98 (performing add, subtract, get, and exit)



```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/I
The client is running.
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter a value to add to your sum:
100
Enter your ID:
98
Reply from server: 100
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter a value to subtract from your sum:
101
Enter your ID:
98
Reply from server: -1
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
98
Reply from server: -1
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

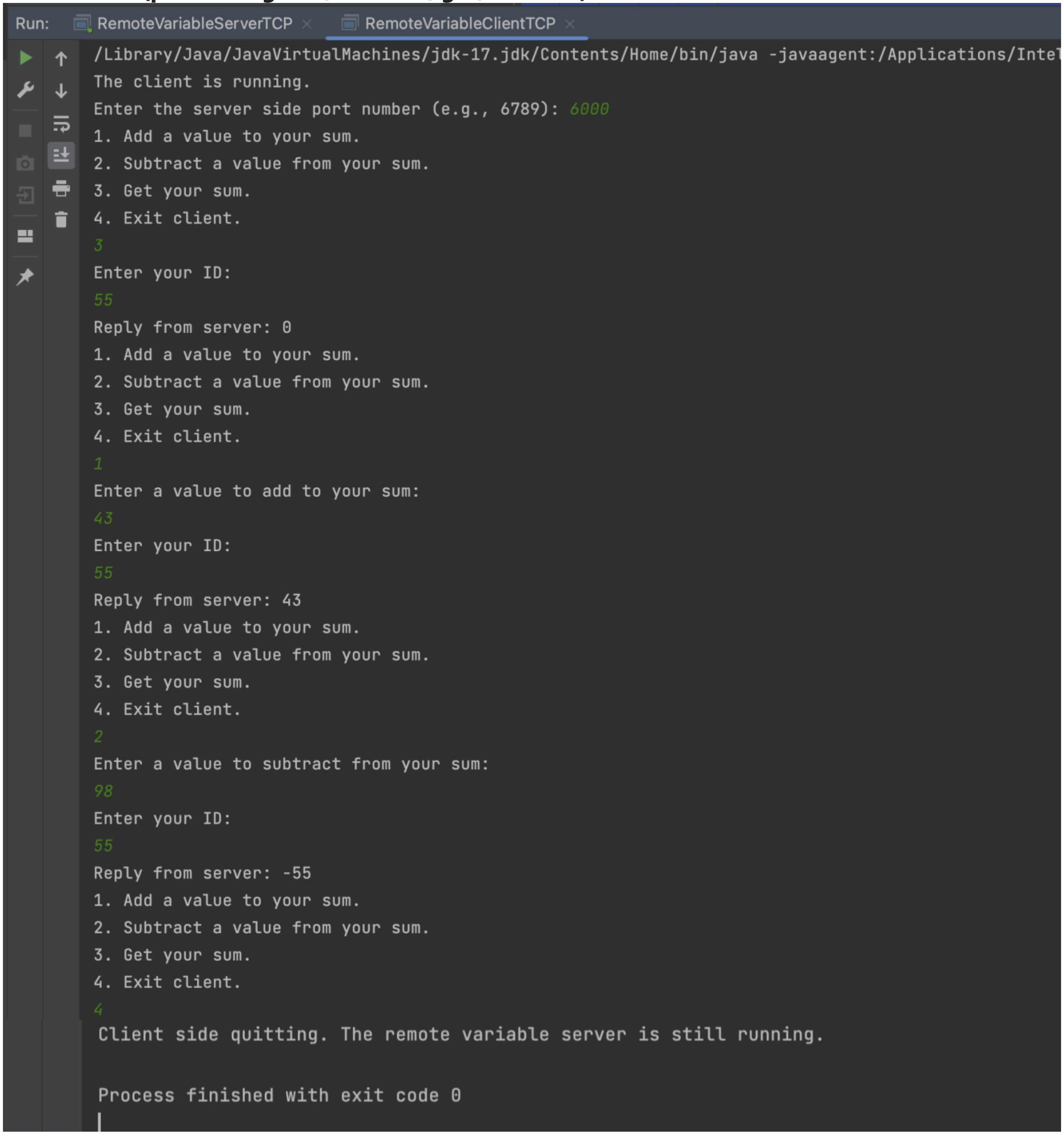
Client id: 10 (performing add, subtract, get, and exit)



Run: RemoteVariableServerTCP × RemoteVariableClientTCP ×

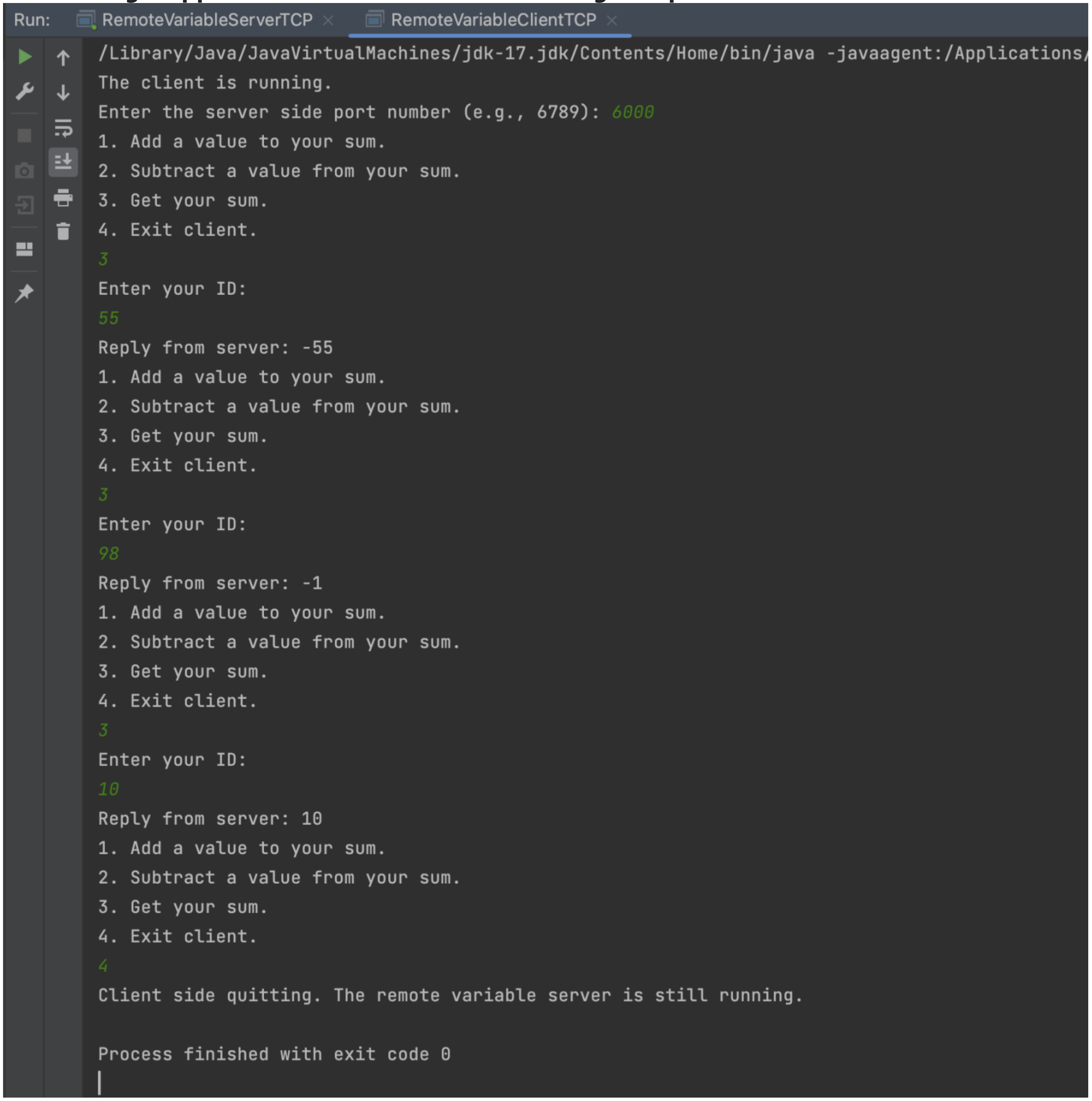
```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/Inte
The client is running.
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter a value to subtract from your sum:
10
Enter your ID:
10
Reply from server: -10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter a value to add to your sum:
20
Enter your ID:
10
Reply from server: 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
10
Reply from server: 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Client id: 55 (performing add, subtract, get, and exit)



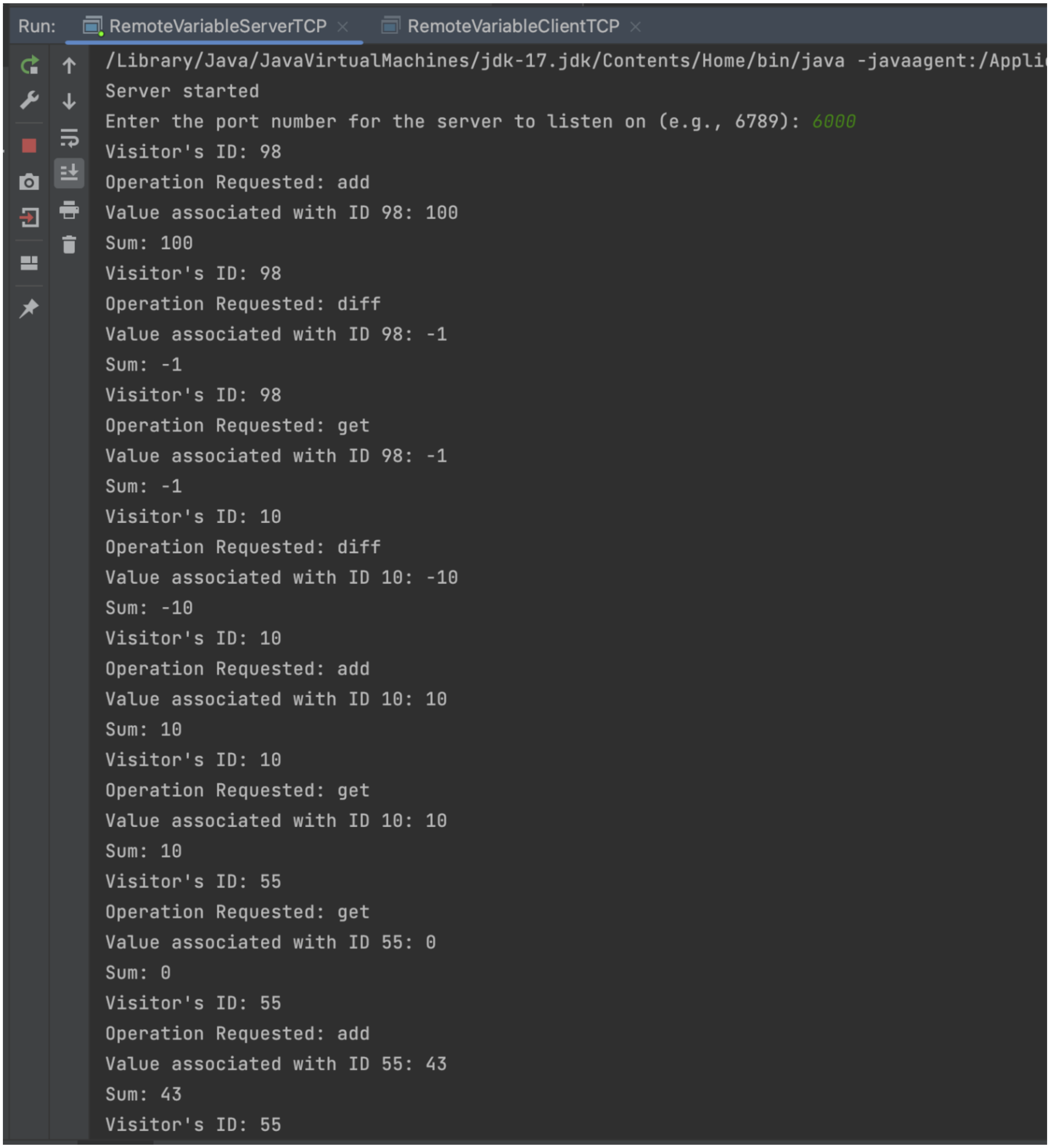
```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/Inte
The client is running.
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
55
Reply from server: 0
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter a value to add to your sum:
43
Enter your ID:
55
Reply from server: 43
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter a value to subtract from your sum:
98
Enter your ID:
55
Reply from server: -55
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Client being stopped and re-run a second time with get requests from each of the three clients.

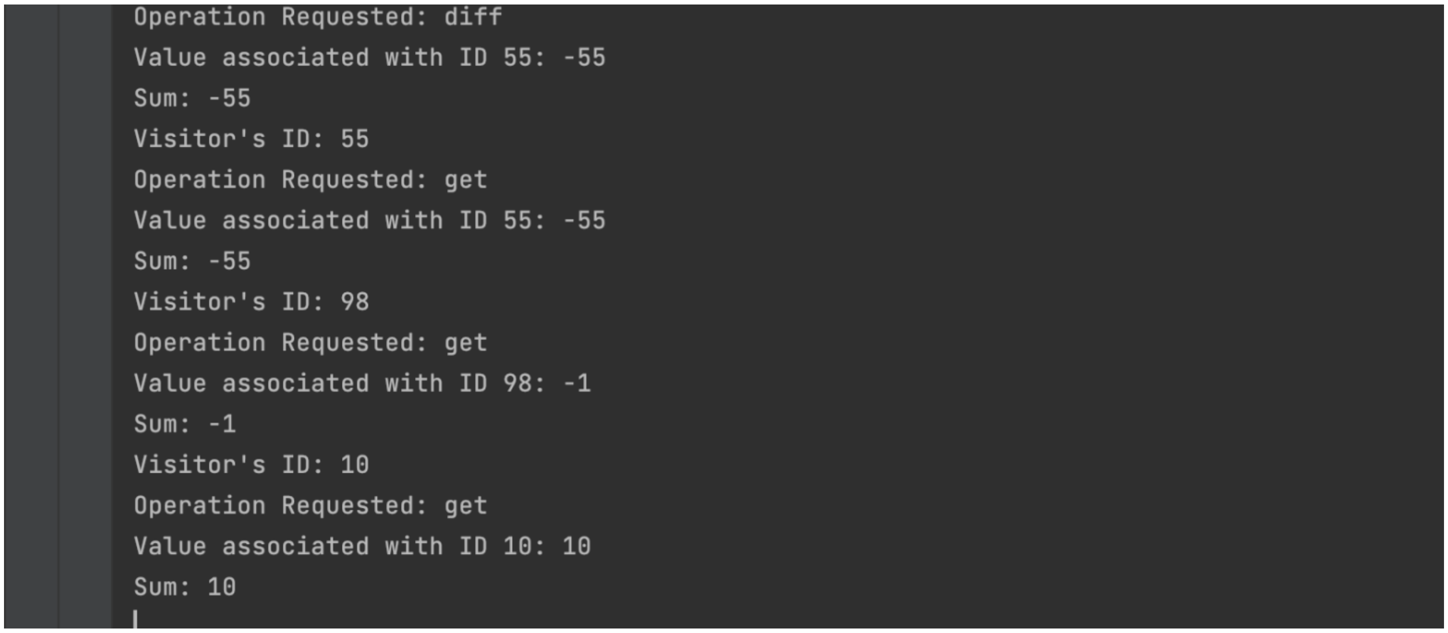


```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/
The client is running.
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
55
Reply from server: -55
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
98
Reply from server: -1
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Enter your ID:
10
Reply from server: 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

"Project2Task4ServerConsole"

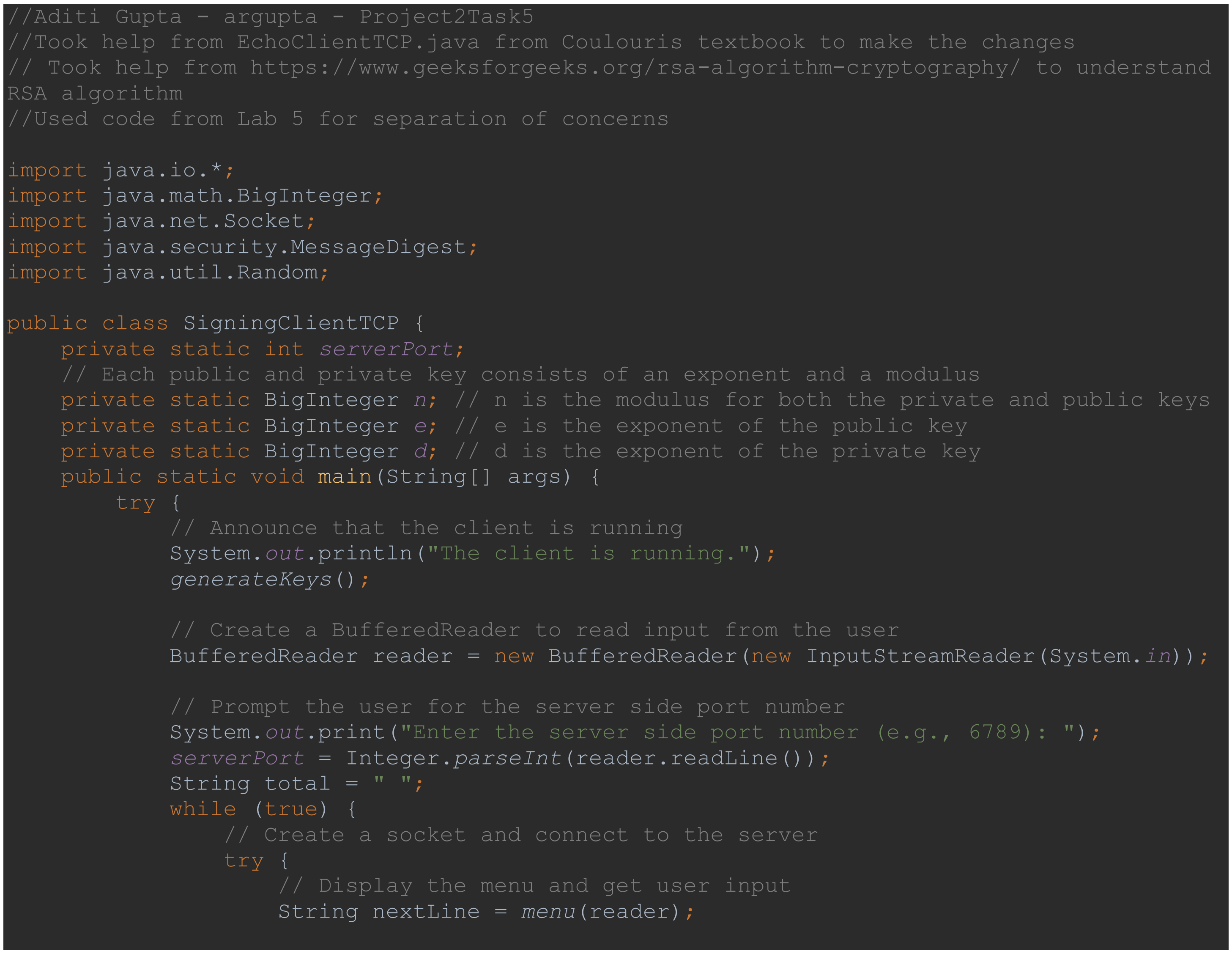


```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Appli
Server started
Enter the port number for the server to listen on (e.g., 6789): 6000
Visitor's ID: 98
Operation Requested: add
Value associated with ID 98: 100
Sum: 100
Visitor's ID: 98
Operation Requested: diff
Value associated with ID 98: -1
Sum: -1
Visitor's ID: 98
Operation Requested: get
Value associated with ID 98: -1
Sum: -1
Visitor's ID: 10
Operation Requested: diff
Value associated with ID 10: -10
Sum: -10
Visitor's ID: 10
Operation Requested: add
Value associated with ID 10: 10
Sum: 10
Visitor's ID: 10
Operation Requested: get
Value associated with ID 10: 10
Sum: 10
Visitor's ID: 55
Operation Requested: get
Value associated with ID 55: 0
Sum: 0
Visitor's ID: 55
Operation Requested: add
Value associated with ID 55: 43
Sum: 43
Visitor's ID: 55
```



```
Operation Requested: diff
Value associated with ID 55: -55
Sum: -55
Visitor's ID: 55
Operation Requested: get
Value associated with ID 55: -55
Sum: -55
Visitor's ID: 98
Operation Requested: get
Value associated with ID 98: -1
Sum: -1
Visitor's ID: 10
Operation Requested: get
Value associated with ID 10: 10
Sum: 10
```

Project2Task5
"Project2Task5Client"



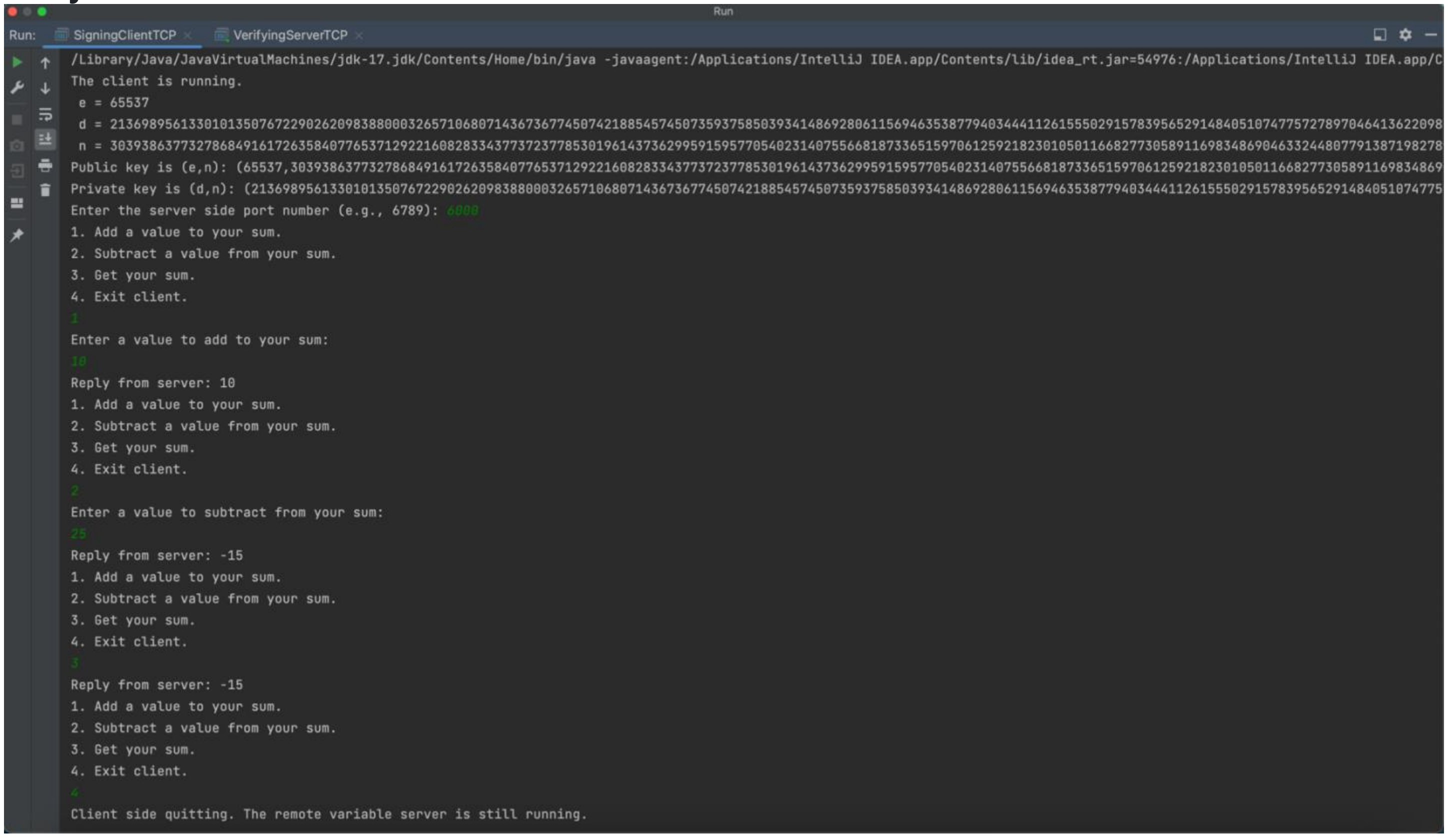
```java
//Aditi Gupta - argupta - Project2Task5
//Took help from EchoClientTCP.java from Coulouris textbook to make the changes
// Took help from https://www.geeksforgeeks.org/rsa-algorithm-cryptography/ to understand
RSA algorithm
//Used code from Lab 5 for separation of concerns

import java.io.*;
import java.math.BigInteger;
import java.net.Socket;
import java.security.MessageDigest;
import java.util.Random;

public class SigningClientTCP {
    private static int serverPort;
    // Each public and private key consists of an exponent and a modulus
    private static BigInteger n; // n is the modulus for both the private and public keys
    private static BigInteger e; // e is the exponent of the public key
    private static BigInteger d; // d is the exponent of the private key
    public static void main(String[] args) {
        try {
            // Announce that the client is running
            System.out.println("The client is running.");
            generateKeys();

            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            // Prompt the user for the server side port number
            System.out.print("Enter the server side port number (e.g., 6789): ");
            serverPort = Integer.parseInt(reader.readLine());
            String total = " ";
            while (true) {
                // Create a socket and connect to the server
                try {
                    // Display the menu and get user input
                    String nextLine = menu(reader);
                    // https://gist.github.com/chatton/8955d2f96f58f6082bde14e7c33f69a6
                    if (nextLine.trim().equalsIgnoreCase("1")) {
                        // Option 1: Add a value to the sum
                        System.out.println("Enter a value to add to your sum:");
                        String value = reader.readLine();
                        String add = "add";
                        total = hashId() + "," + e+"," +n+"," +value + "," + add;
                        String signedMessage = sign(total);
                        total = total + "," + signedMessage;
                    } else if (nextLine.trim().equalsIgnoreCase("2")) {
                        // Option 2: Subtract a value from the sum
                        System.out.println("Enter a value to subtract from your sum:");
                        String value = reader.readLine();
                        String diff = "diff";
                        total = hashId() + "," + e+"," +n+"," + value + "," + diff;
                        String signedMessage = sign(total);
                        total = total + "," + signedMessage;
                    } else if (nextLine.trim().equalsIgnoreCase("3")) {
                        // Option 3: Get the current sum
                        int num = 0;
                        String get = "get";
                        total = hashId() + "," + e +","+ n +","+ num + "," + get;
                        String signedMessage = sign(total);
                        total = total + "," + signedMessage;
                    } else if (nextLine.trim().equalsIgnoreCase("4")) {
                        // Option 4: Exit the client
                        System.out.println("Client side quitting. The remote variable
server is still running.");
                        break;
                    } else {
                        System.out.println("Invalid option. Please choose a valid option
(1-4).");
                    }

                    // Read and display the server's reply
                    //https://gist.github.com/chatton/8955d2f96f58f6082bde14e7c33f69a6
                    String reply = communicateWithServer(total);
                    System.out.println("Reply from server: " + reply);
                } catch (IOException e) {
                    System.out.println("Error in client socket: " + e.getMessage());
                } catch (Exception ex) {
                    throw new RuntimeException(ex);
                }
            }
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        }
    }

    // Method to encapsulate communication with the server
    //Taken from EchoClientTCP.java from Coulouris textbook
    private static String communicateWithServer(String request) {
        try (Socket socket = new Socket("localhost", serverPort);
             BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

            out.println(request); // Send the request to the server
            return in.readLine(); // Read and return the server's reply
        } catch (IOException e) {
            return "Error in client socket: " + e.getMessage();
        }
    }
```

```java
    // Used code from ShortMessageSign.java for this function
    public static String sign(String message) throws Exception {

        // compute the digest with SHA-256
        byte[] bytesOfMessage = message.getBytes("UTF-8");
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] bigDigest = md.digest(bytesOfMessage);

        // we add a 0 byte as the most significant byte to keep
        // the value to be signed non-negative.
        byte[] messageDigest = new byte[bigDigest.length+1];

        //code taken from ShortMessageSign.java - Signing a short message
        //https://stackoverflow.com/questions/6780395/how-can-i-convert-a-byte-to-a-
positive-biginteger-in-java
        System.arraycopy(bigDigest, 0, messageDigest, 1, bigDigest.length);

        // From the digest, create a BigInteger
        BigInteger m = new BigInteger(messageDigest);

        // encrypt the digest with the private key
        BigInteger c = m.modPow(d, n);

        // return this as a big integer string
        return c.toString();
    }

    // Method to display the client menu and get user input
    public static String menu(BufferedReader reader) throws IOException {
        System.out.println("1. Add a value to your sum.");
        System.out.println("2. Subtract a value from your sum.");
        System.out.println("3. Get your sum.");
        System.out.println("4. Exit client.");
        String nextLine = reader.readLine();
        return nextLine;
    }

    //generate private and public keys  and display this to user
    // Took code for generating public and private keys from RSAExample.java
    public static void generateKeys() {

        Random rnd = new Random();

        // Step 1: Generate two large random primes.
        // We use 400 bits here, but best practice for security is 2048 bits.
        // Change 400 to 2048, recompile, and run the program again and you will
        // notice it takes much longer to do the math with that many bits.
        BigInteger p = new BigInteger(400, 100, rnd);
        BigInteger q = new BigInteger(400, 100, rnd);

        // Step 2: Compute n by the equation n = p * q.
        n = p.multiply(q);

        // Step 3: Compute phi(n) = (p-1) * (q-1)
        BigInteger phi =
(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

        // Step 4: Select a small odd integer e that is relatively prime to phi(n).
        // By convention the prime 65537 is used as the public exponent.
        e = new BigInteger("65537");

        // Step 5: Compute d as the multiplicative inverse of e modulo phi(n).
        d = e.modInverse(phi);
        System.out.println(" e = " + e);  // Step 6: (e,n) is the RSA public key
        System.out.println(" d = " + d);  // Step 7: (d,n) is the RSA private key
        System.out.println(" n = " + n);  // Modulus for both keys
        System.out.println("Public key is (e,n): (" + e + "," + n + ")");
        System.out.println("Private key is (d,n): (" + d + "," + n + ")");
    }
    public static String hashId() throws Exception {
    String s= e.toString()+n.toString();
        // compute the digest with SHA-256
        // code taken from ShortMessageSign.java - Signing a short message
        byte[] bytesOfMessage = s.getBytes("UTF-8");
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] bigDigest = md.digest(bytesOfMessage);

        //code taken from ShortMessageSign.java - Signing a short message
        //https://stackoverflow.com/questions/6780395/how-can-i-convert-a-byte-to-a-
positive-biginteger-in-java
        BigInteger bigInteger = new BigInteger(1, bigDigest);

        //Converting big integer to string
        String hashValue = bigInteger.toString();

        //printing the last 20 characters of the hash value
        String id = hashValue.substring(hashValue.length() - 20);
        return id;
    }
}
```

“Project2Task5ServerConsole"

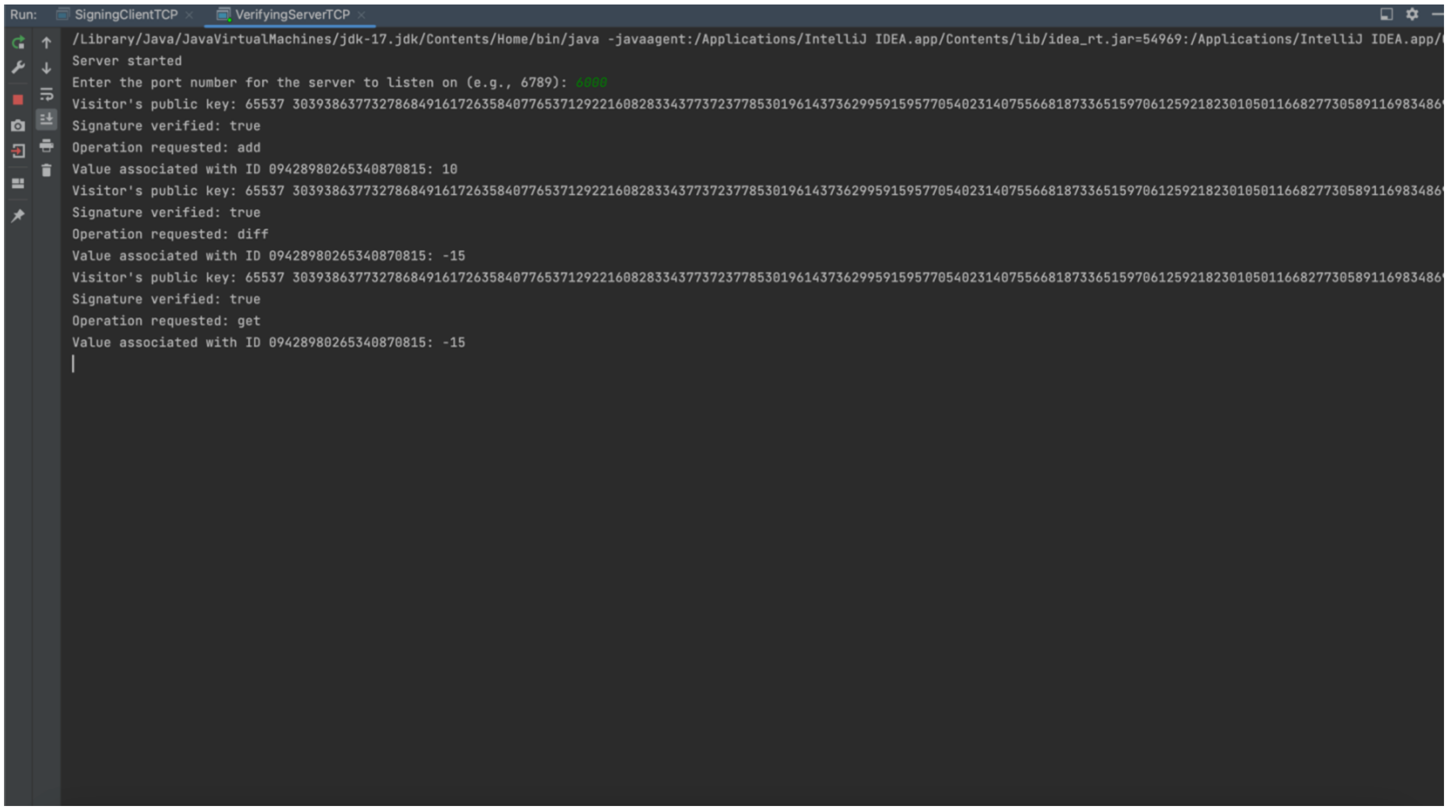
```
//Aditi Gupta - argupta - Project2Task5
//Took code from EchoServerTCP.java from Coulouris textbook to make the changes
// Took help from https://www.geeksforgeeks.org/rsa-algorithm-cryptography/ to understand
RSA algorithm
// Used ShortMessageSign.java and ShortMessageVerify.java to sign and check the signature
on very small messages.
//Used code from Lab 5 for separation of concerns

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.math.BigInteger;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.SQLOutput;
import java.util.HashMap;
import java.security.MessageDigest;
import java.util.TreeMap;

public class VerifyingServerTCP {
    private static int sum = 0, diff=0; // Variable to store the sum/difference of values
private static String id, e, n,operation, sign, operand;
private static int value;
    public static void main(String[] args) {
        // Create a ServerSocket for accepting incoming client connections
        ServerSocket serverSocket = null;

        // A hashmap to store and retrieve values associated with client IDs
        TreeMap<String, Integer> map = new TreeMap<>();
        try {
            // Announce that the server is running
            System.out.println("Server started");

            // Create a BufferedReader to read input from the user
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            // Prompt the user for the port number to listen on
            System.out.print("Enter the port number for the server to listen on (e.g.,
6789): ");
            int serverPort = Integer.parseInt(reader.readLine());

            // Create a ServerSocket to listen for incoming TCP connections
            serverSocket = new ServerSocket(serverPort);

            while (true) {
                // Wait for a client to connect
                Socket clientSocket = serverSocket.accept();

                // Read and process client request
                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
                String request = in.readLine();

                if (request == null) {
                    continue;
                }

                // Split the client request to extract details
                String[] elements = request.split(",");
                 id = elements[0];
                e= elements[1];
                n= elements[2];
                System.out.println("Visitor's public key: " + e + " " + n);

                // Compute and verify the client ID
                String computedID = hashId();
                if (!computedID.equals(id)) {
                    out.println("Error in verifying ID");
                    clientSocket.close();
                    continue;
                }

                // Check if this ID has a previous value, otherwise initialize with zero
                if (!map.containsKey(id)) {
                    map.put(id, 0);
                }

                // Extract other elements from the client request
                operand = elements[3];
                 operation = elements[4];
                 sign=elements[5];
                System.out.println("Signature verified: " + verify((id+ ","+e+ ","+n+
","+operand+ ","+operation), sign));
                System.out.println("Operation requested: " + operation);
                // Verify the client's signature
                if (!verify((id+ ","+e+ ","+n+ ","+operand+ ","+operation), sign)) {
                    out.println("Error in verifying signature");
                    clientSocket.close();
                    continue;
                }
                // Perform the requested operation (addition or subtraction or get)
                value=Integer.parseInt(operand);
                if (operation.equalsIgnoreCase("add")||
operation.equalsIgnoreCase("get")) {
                    sum = add(map.get(id), value);
                } else {
                    sum = diff(map.get(id), value);
                }

                // Update the value associated with the client ID in the map
                map.put(id, sum);

                // Print the updated value associated with the client ID
                System.out.println("Value associated with ID " + id + ": " +
map.get(id));

                // Send the result back to the client
                out.println(sum);

                // Close the client socket when done
                clientSocket.close();
            }
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } catch (Exception ex) {
            throw new RuntimeException(ex);
        } finally {
            try {
                if (serverSocket != null)
                    serverSocket.close();
            } catch (IOException e) {
                System.out.println("Error closing server socket: " + e.getMessage());
            }
        }
    }

    // Method to add two numbers
    public static int add(int i, int value) {
        sum = i+value;
        return sum;
    }

    // Method to subtract two numbers
    public static int diff(int i, int value) {
        diff = i-value;
        return diff;
    }

    // Method to compute a hash of the client's public key details
    // code taken from ShortMessageSign.java - Signing a short message
    public static String hashId() throws Exception {
        String s= e+n;
        // compute the digest with SHA-256
        // code taken from ShortMessageSign.java - Signing a short message
        byte[] bytesOfMessage = s.getBytes("UTF-8");
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] bigDigest = md.digest(bytesOfMessage);

        //code taken from ShortMessageSign.java - Signing a short message
        //https://stackoverflow.com/questions/6780395/how-can-i-convert-a-byte-to-a-
positive-biginteger-in-java
        BigInteger bigInteger = new BigInteger(1, bigDigest);
        //Converting big integer to string
        String hashValue = bigInteger.toString();

        //printing the last 20 characters of the hash value
        String id = hashValue.substring(hashValue.length() - 20);
        return id;
    }


    // Took code from ShortMessageVerify.java to check the signature on very small
messages.
    // Method to verify the client's signed message using RSA
    public static boolean verify(String messageToCheck, String encryptedHashStr)throws
Exception  {

        // Take the encrypted string and make it a big integer
        BigInteger encryptedHash = new BigInteger(encryptedHashStr);
        // Decrypt it
        BigInteger E=new BigInteger(e);
        BigInteger N=new BigInteger(n);
        BigInteger decryptedHash = encryptedHash.modPow(E,N);

        // Get the bytes from messageToCheck
        byte[] bytesOfMessageToCheck = messageToCheck.getBytes("UTF-8");

        // compute the digest of the message with SHA-256
        MessageDigest md = MessageDigest.getInstance("SHA-256");

        byte[] messageToCheckDigest = md.digest(bytesOfMessageToCheck);

        // we add a 0 byte as the most significant byte to keep
        // the value to be signed non-negative.
        byte[] messageDigest = new byte[messageToCheckDigest.length+1];

        //Took this line from https://stackoverflow.com/questions/6780395/how-can-i-
convert-a-byte-to-a-positive-biginteger-in-java
        System.arraycopy(messageToCheckDigest, 0, messageDigest, 1,
messageToCheckDigest.length);

        // Make it a big int
        BigInteger bigIntegerToCheck = new BigInteger(messageDigest);

        // inform the client on how the two compare
        if(bigIntegerToCheck.compareTo(decryptedHash) == 0) {

            return true;
        }
        else {
            return false;
        }
    }

}
```

"Project2Task5ClientConsole"



```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=54976:/Applications/IntelliJ IDEA.app/C
The client is running.
 e = 65537
 d = 21369895613301013507672290262098388000326571068071436736774507421885457450735937585039341486928061156946353877940344411261555029157839565291484051074775727897046413622098
 n = 30393863773278684916172635840776537129221608283343773723778530196143736299591595770540231407556681873365159706125921823010501166827730589116983486904633244807791387198278
Public key is (e,n): (65537,30393863773278684916172635840776537129221608283343773723778530196143736299591595770540231407556681873365159706125921823010501166827730589116983486
Private key is (d,n): (2136989561330101350767229026209838800032657106807143673677450742188545745073593758503934148692806115694635387794034441126155502915783956529148405107477
Enter the server side port number (e.g., 6789): 6000
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter a value to add to your sum:
10
Reply from server: 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter a value to subtract from your sum:
25
Reply from server: -15
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
3
Reply from server: -15
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
4
Client side quitting. The remote variable server is still running.
```

"Project2Task5ServerConsole"



```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=54969:/Applications/IntelliJ IDEA.app/
Server started
Enter the port number for the server to listen on (e.g., 6789): 6000
Visitor's public key: 65537 30393863773278684916172635840776537129221608283343773723778530196143736299591595770540231407556681873365159706125921823010501166827730589116983486
Signature verified: true
Operation requested: add
Value associated with ID 09428980265340870815: 10
Visitor's public key: 65537 30393863773278684916172635840776537129221608283343773723778530196143736299591595770540231407556681873365159706125921823010501166827730589116983486
Signature verified: true
Operation requested: diff
Value associated with ID 09428980265340870815: -15
Visitor's public key: 65537 30393863773278684916172635840776537129221608283343773723778530196143736299591595770540231407556681873365159706125921823010501166827730589116983486
Signature verified: true
Operation requested: get
Value associated with ID 09428980265340870815: -15
```