Lawrence Hua
LHUA
Lhua@andrew.cmu.edu
Project 2

Project2Task0Client)

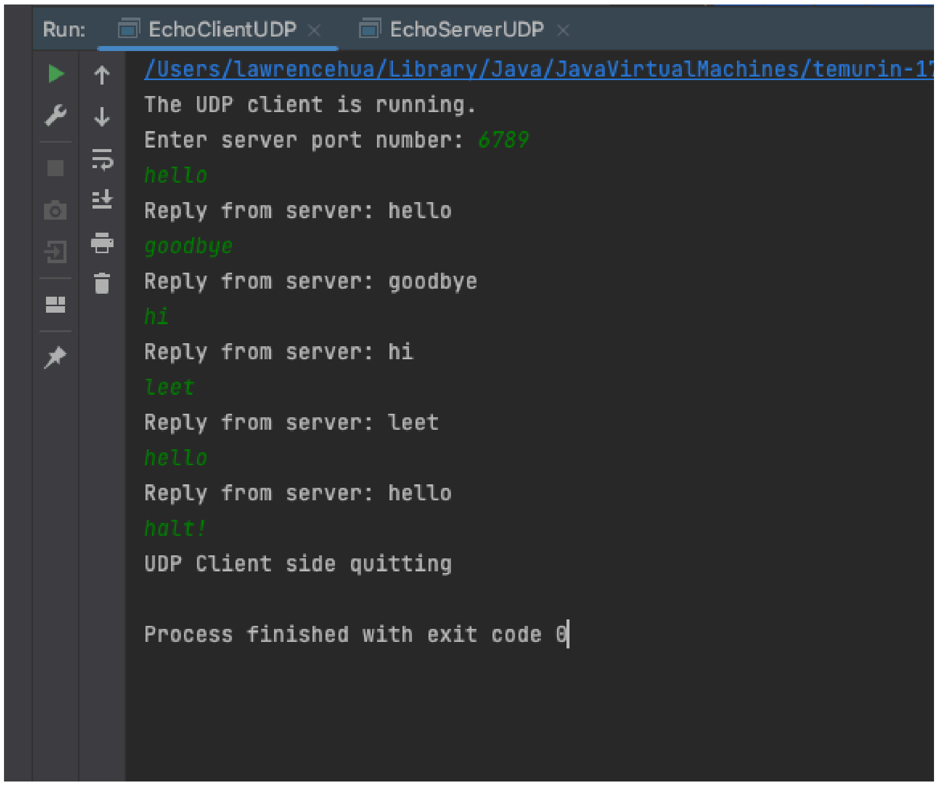
```
public class EchoClientUDP {
   public static void main(String args[]) {
       args = new String[]{"localhost"};
       // args give message contents and server hostname
       DatagramSocket aSocket = null;
       try {
           // set host name, server port for listening,
           InetAddress aHost = InetAddress.getByName(args[0]);

           // socket used for sending and receiving data.
           aSocket = new DatagramSocket();
           String nextLine;
           System.out.println("The UDP client is running.");
           System.out.print("Enter server port number: ");
           // allows user to type text into console
           BufferedReader typed = new BufferedReader(new
InputStreamReader(System.in));
           int serverPort = Integer.parseInt(typed.readLine());
           while ((nextLine = typed.readLine()) != null) {
               if (nextLine.equals("halt!")) {
                   // Send "halt!" message to server
                   byte[] m = nextLine.getBytes();
                   DatagramPacket request = new DatagramPacket(m, m.length,
aHost, serverPort);
                   aSocket.send(request);
                   System.out.println("UDP Client side quitting");
                   break; // Exit the loop
               }
               byte[] m = nextLine.getBytes();
               // sends message containing message, length, destination
address, and destination port.
               DatagramPacket request = new DatagramPacket(m, m.length, aHost,
serverPort);
               aSocket.send(request);
               byte[] buffer = new byte[1000];
               // get data from server
               DatagramPacket reply = new DatagramPacket(buffer,
buffer.length);
               aSocket.receive(reply);
               byte[] replyData = new byte[reply.getLength()];
               System.arraycopy(reply.getData(), 0, replyData, 0,
reply.getLength());
```

```
                System.out.println("Reply from server: " + new
String(replyData));
            }
        } 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();
        }
    }
}
```

Project2Task0Server)

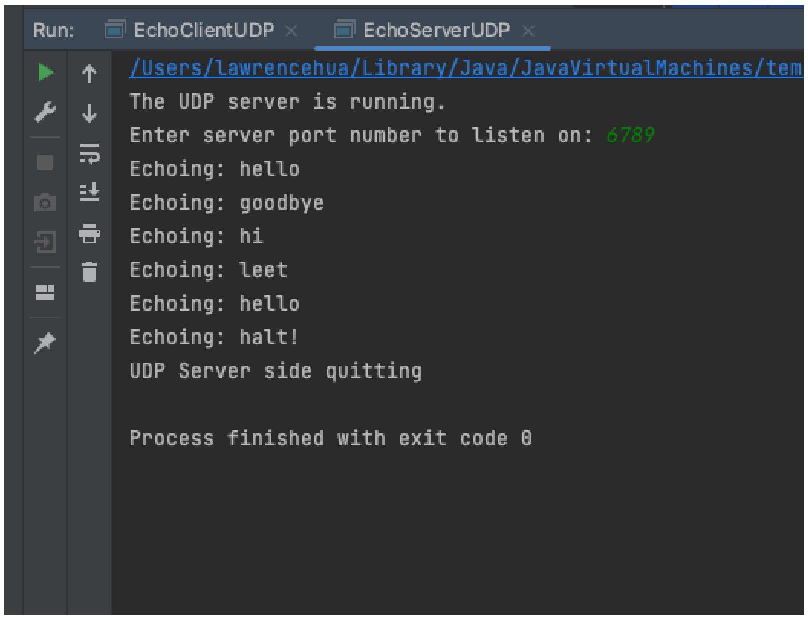
```
public class EchoServerUDP {
    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        byte[] buffer = new byte[1000];
        try {
            System.out.println("The UDP server is running.");
            System.out.print("Enter server port number to listen on: ");
            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
            int serverPort = Integer.parseInt(reader.readLine()); // Prompt the
user for the server port number
            // listen/push data on this port
            aSocket = new DatagramSocket(serverPort);
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);
            while (true) {
                aSocket.receive(request);
                // creates a reply after a receive and sends it
                byte[] requestData = new byte[request.getLength()];
                System.arraycopy(request.getData(), 0, requestData, 0,
request.getLength());
                String requestString = new String(requestData);
                // ensure that message is received from client
                System.out.println("Echoing: " + requestString);
                if (requestString.equals("halt!")) {
                    System.out.println("UDP Server side quitting");
                    // Send "halt!" message back to client
                    DatagramPacket reply = new DatagramPacket(request.getData(),
                            request.getLength(), request.getAddress(),
request.getPort());
                    aSocket.send(reply);
                    break; // Exit the loop
                }
                // sends out reply from the socket previously created
                DatagramPacket reply = new DatagramPacket(request.getData(),
                        request.getLength(), request.getAddress(),
request.getPort());
                aSocket.send(reply);
            }
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        } finally {
            if (aSocket != null) aSocket.close();
        }
    }
}
```

Project2Task0ClientConsole)



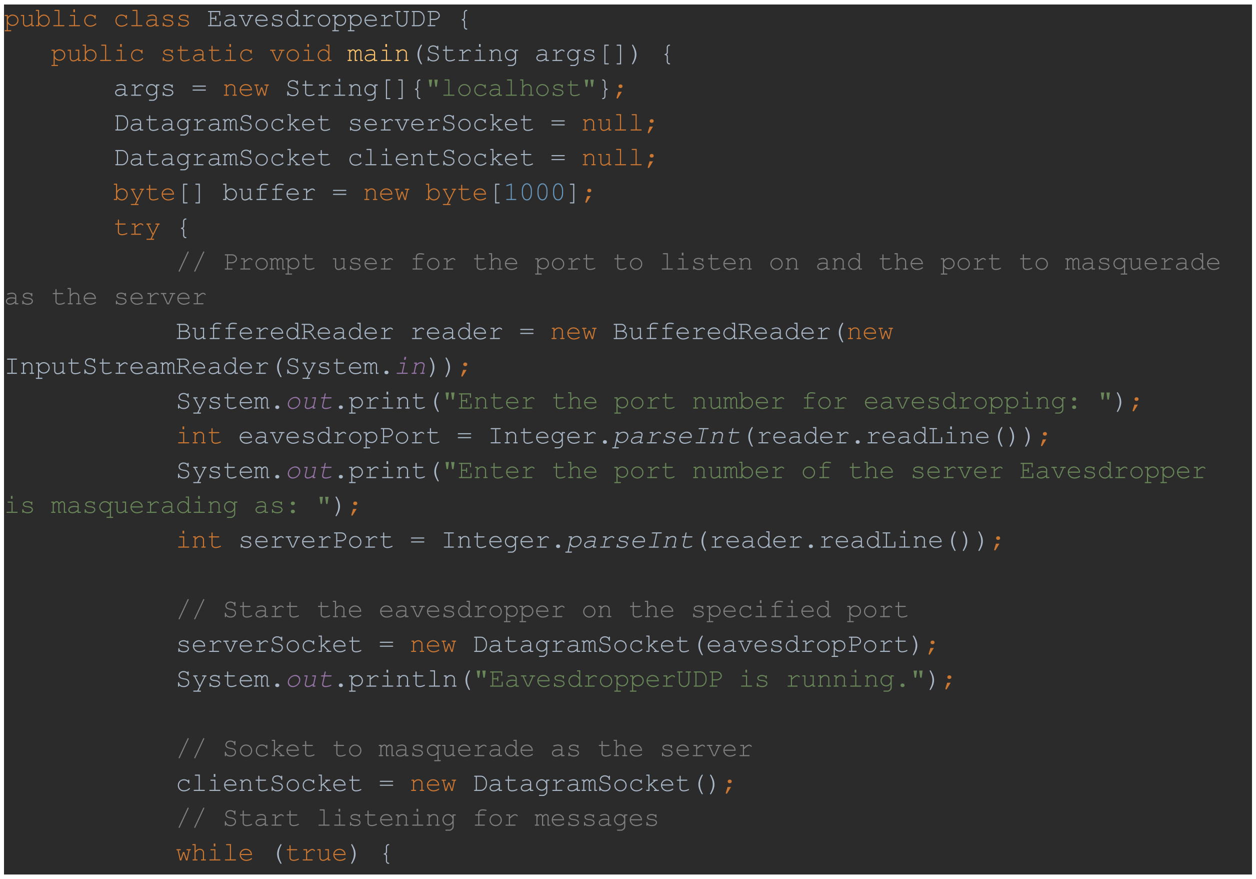
```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/temurin-17
The UDP client is running.
Enter server port number: 6789
hello
Reply from server: hello
goodbye
Reply from server: goodbye
hi
Reply from server: hi
leet
Reply from server: leet
hello
Reply from server: hello
halt!
UDP Client side quitting

Process finished with exit code 0
```

Project2Task0ServerConsole)



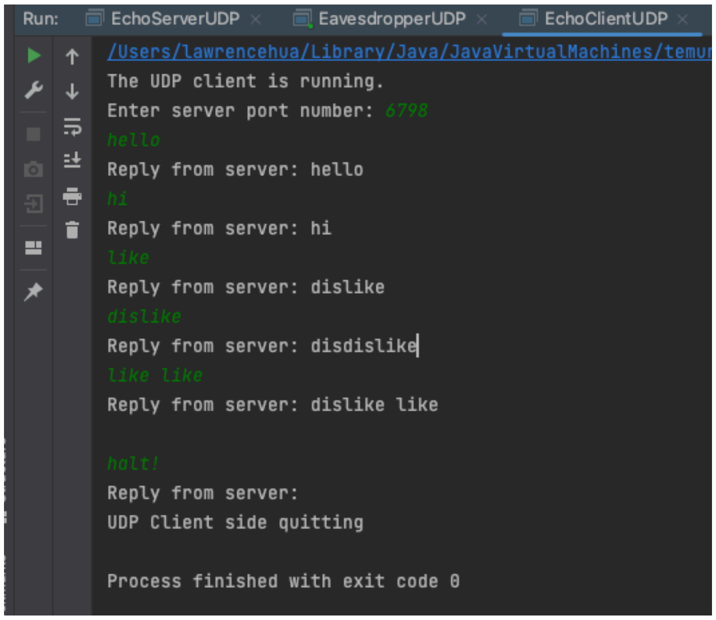
```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/tem
The UDP server is running.
Enter server port number to listen on: 6789
Echoing: hello
Echoing: goodbye
Echoing: hi
Echoing: leet
Echoing: hello
Echoing: halt!
UDP Server side quitting

Process finished with exit code 0
```

Project2Task1
EavesdropperUDP)



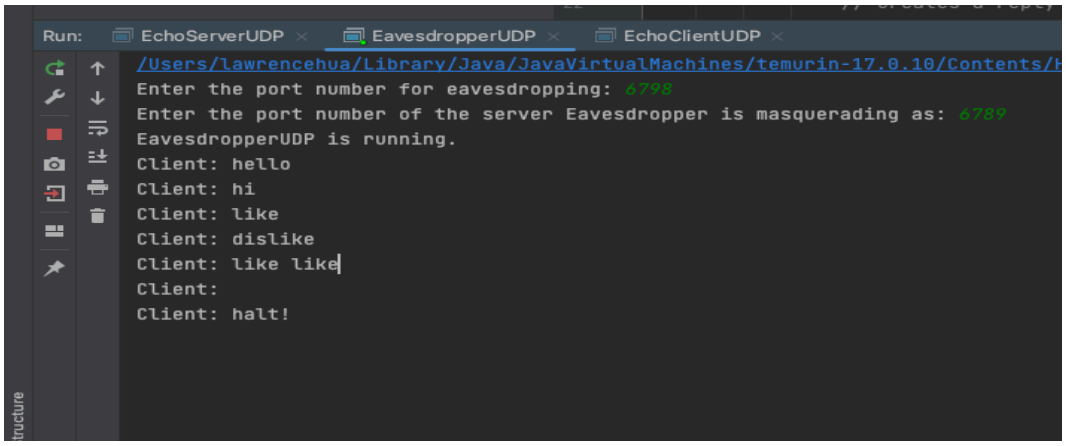
```
public class EavesdropperUDP {
   public static void main(String args[]) {
       args = new String[]{"localhost"};
       DatagramSocket serverSocket = null;
       DatagramSocket clientSocket = null;
       byte[] buffer = new byte[1000];
       try {
           // Prompt user for the port to listen on and the port to masquerade
as the server
           BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
           System.out.print("Enter the port number for eavesdropping: ");
           int eavesdropPort = Integer.parseInt(reader.readLine());
           System.out.print("Enter the port number of the server Eavesdropper
is masquerading as: ");
           int serverPort = Integer.parseInt(reader.readLine());

           // Start the eavesdropper on the specified port
           serverSocket = new DatagramSocket(eavesdropPort);
           System.out.println("EavesdropperUDP is running.");

           // Socket to masquerade as the server
           clientSocket = new DatagramSocket();
           // Start listening for messages
           while (true) {
               // Receive message from the client
               DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
               serverSocket.receive(request);
               String received = new String(request.getData(), 0,
request.getLength());

               // Display the message received from the client
               System.out.println("Client: " + received);

               // If message contains "like", replace with "dislike" only once
               if (received.contains("like")) {
                   received = received.replaceFirst("like", "dislike");
               }
               InetAddress aHost = InetAddress.getByName(args[0]);

               // Send modified message to the server
               DatagramPacket responseToServer = new
DatagramPacket(received.getBytes(), received.getBytes().length,aHost ,
serverPort);
               serverSocket.send(responseToServer);

               // Receive response from the server
               DatagramPacket serverResponse = new DatagramPacket(buffer,
buffer.length);
               serverSocket.receive(serverResponse);

               // Forward server's response back to the client
               DatagramPacket responseToClient = new
DatagramPacket(serverResponse.getData(), serverResponse.getLength(),
request.getAddress(), request.getPort());
               clientSocket.send(responseToClient);

               //clear buffer so theres no spam
               buffer = new byte[1000];
           }
//catch exceptions and finally close sockets once everything is done.
       } catch (SocketException e) {
           System.out.println("Socket Exception: " + e.getMessage());
       } catch (IOException e) {
           System.out.println("IO Exception: " + e.getMessage());
       } finally {
           if (serverSocket != null) serverSocket.close();
           if (clientSocket != null) clientSocket.close();
       }
   }
}
```

Project2Task1ThreeConsoles)



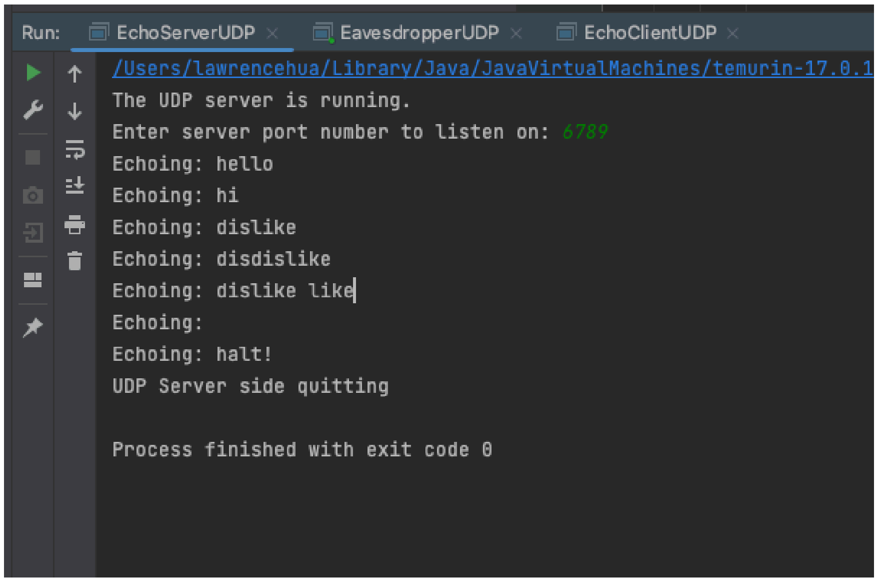
Run: EchoServerUDP EavesdropperUDP EchoClientUDP

```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/temur
The UDP client is running.
Enter server port number: 6798
hello
Reply from server: hello
hi
Reply from server: hi
like
Reply from server: dislike
dislike
Reply from server: disdislike
like like
Reply from server: dislike like

halt!
Reply from server:
UDP Client side quitting

Process finished with exit code 0
```



Run: EchoServerUDP EavesdropperUDP EchoClientUDP

```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/temurin-17.0.10/Contents/H
Enter the port number for eavesdropping: 6798
Enter the port number of the server Eavesdropper is masquerading as: 6789
EavesdropperUDP is running.
Client: hello
Client: hi
Client: like
Client: dislike
Client: like like
Client:
Client: halt!
```



Run: EchoServerUDP EavesdropperUDP EchoClientUDP

```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/temurin-17.0.1
The UDP server is running.
Enter server port number to listen on: 6789
Echoing: hello
Echoing: hi
Echoing: dislike
Echoing: disdislike
Echoing: dislike like
Echoing:
Echoing: halt!
UDP Server side quitting

Process finished with exit code 0
```

Project2Task2Client)

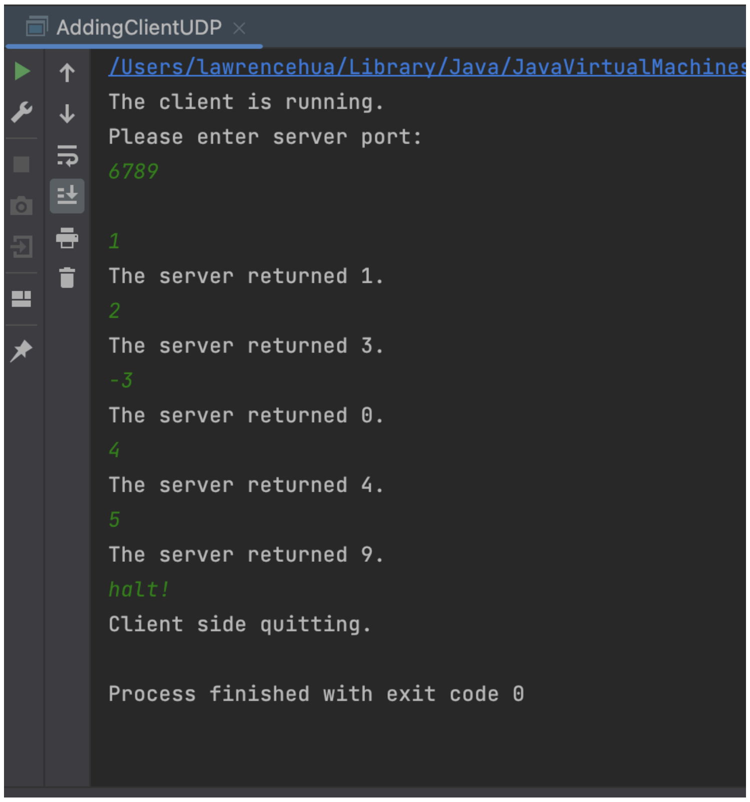
```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.*;

public class AddingClientUDP {

   public static int add(int i) {
       DatagramSocket aSocket = null;
       try {
           BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

           System.out.println();
           // Set host name, server port for listening
           InetAddress aHost = InetAddress.getLocalHost();

           // Socket used for sending and receiving data
           aSocket = new DatagramSocket();

           while (true) {
               String input = reader.readLine();

               if (input.equals("halt!")) {
                   System.out.println("Client side quitting.");
                   break;
               }

               int num = Integer.parseInt(input);

               // Convert the number to bytes and send it to the server
               String message = String.valueOf(num);
               byte[] sendData = message.getBytes();
               //i is the serverport
               DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, aHost, i);
               aSocket.send(sendPacket);

               // Receive the result from the server
               byte[] receiveData = new byte[1024];
               DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
               aSocket.receive(receivePacket);

               // Convert the received data to an integer and print it
```

```
                String receivedMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                System.out.println("The server returned " + receivedMessage +
".");
            }
        } 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();
            }
        }
        return 0;
    }

    public static void main(String args[]) throws IOException {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("The client is running.");
        System.out.println("Please enter server port: ");
        int serverPort = Integer.parseInt(reader.readLine());
        add(serverPort);
    }
}
```

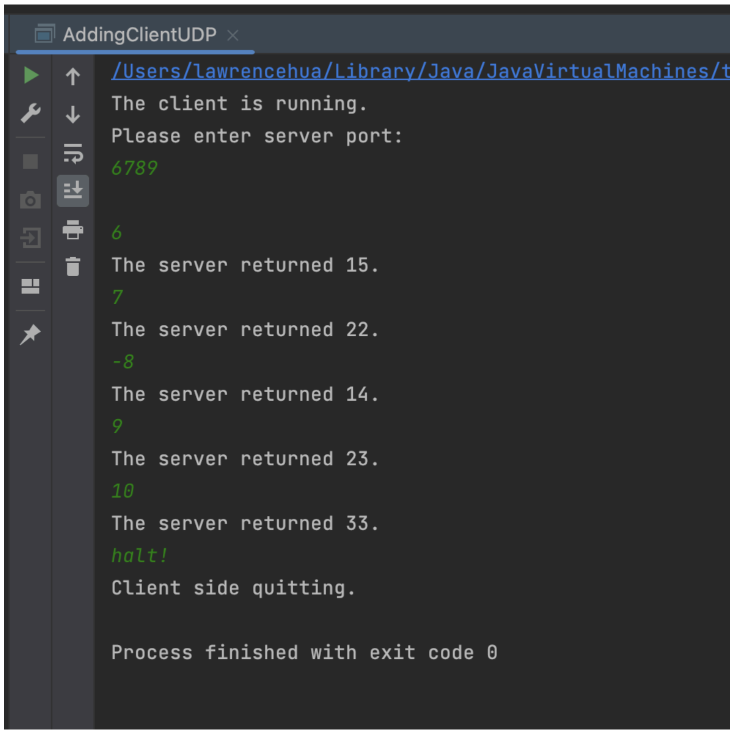
Project2Task2Server)

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

public class AddingServerUDP {
    public static int sum = 0;

    public static int add(int num) {
        sum += num;
        return sum;
    }

    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        byte[] buffer = new byte[1000];
        try {
            System.out.println("The UDP server is running.");
            // listen/push data on port 6789
            System.out.print("Enter server port number to listen on: ");
            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
            int serverPort = Integer.parseInt(reader.readLine()); // Pro
            aSocket = new DatagramSocket(serverPort);
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);
            while (true) {
                aSocket.receive(request);
                // creates a reply after a receive and sends it
                byte[] requestData = new byte[request.getLength()];
                System.arraycopy(request.getData(), 0, requestData, 0,
request.getLength());
                String requestString = new String(requestData);
                // ensure that message is received from client
                System.out.println("Adding: " + requestString + " to " + sum);
                if (requestString.equals("halt!")) {
                    System.out.println("UDP Server side quitting");
                    // Send "halt!" message back to client
                    DatagramPacket reply = new DatagramPacket(request.getData(),
                            request.getLength(), request.getAddress(),
request.getPort());
                    aSocket.send(reply);
                    break; // Exit the loop
                }
                // Perform the add operation and prepare the reply
                int num = Integer.parseInt(requestString);
                int newSum = add(num);
                System.out.println("Returning sum of " + newSum + " to client");
                System.out.println();
                String replyString = String.valueOf(newSum);
                byte[] replyData = replyString.getBytes();
                DatagramPacket reply = new DatagramPacket(replyData,
replyData.length,
                        request.getAddress(), request.getPort());
                aSocket.send(reply);
            }
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        } finally {
            if (aSocket != null) aSocket.close();
        }
    }
}
```

Project2Task2ClientConsole)

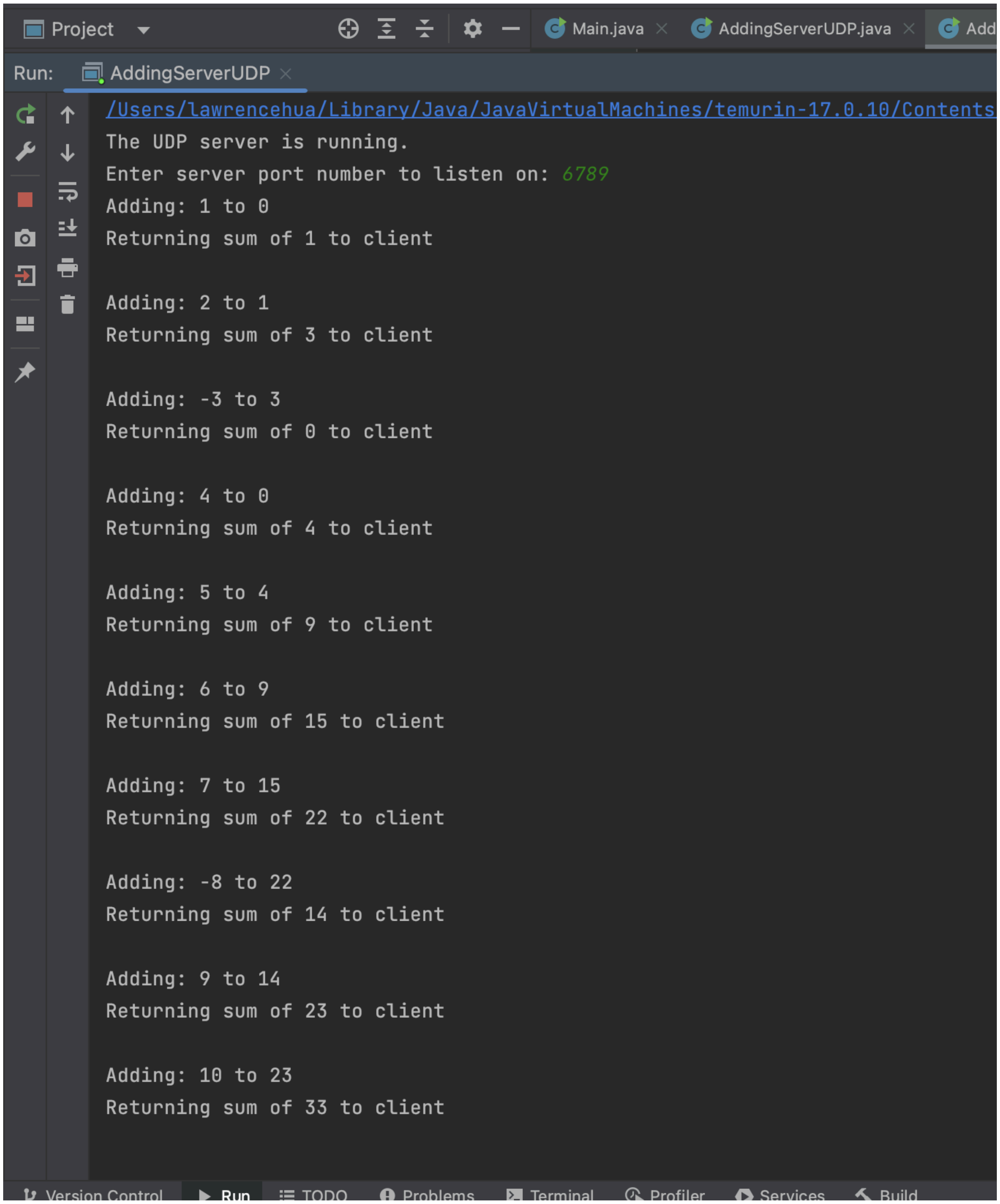


```
/Users/lawrencehua/Library/Java/JavaVirtualMachines
The client is running.
Please enter server port:
6789

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
```



```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/t
The client is running.
Please enter server port:
6789

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)



```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/temurin-17.0.10/Contents
The UDP server is running.
Enter server port number to listen on: 6789
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
```

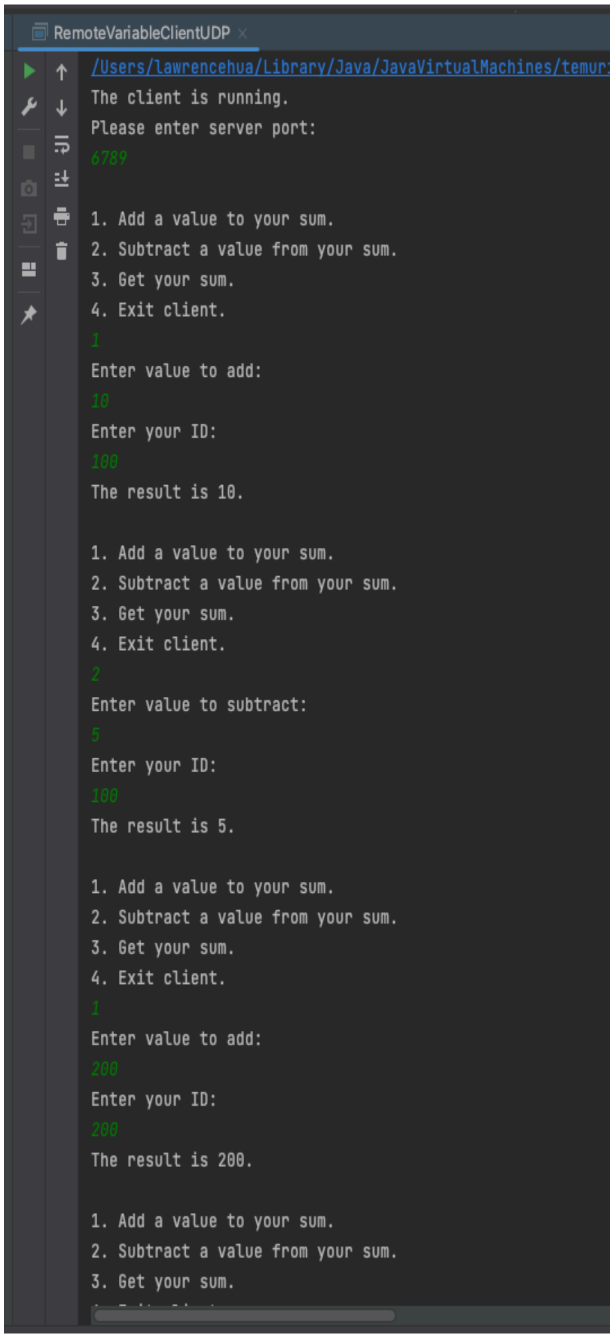
Project2Task3Client)

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.*;

public class RemoteVariableClientUDP {

    public static void main(String args[]) {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("The client is running.");
        System.out.println("Please enter server port: ");
        try {
            int serverPort = Integer.parseInt(reader.readLine());
            System.out.println();
            while (true) {
                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.");

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

                if (option == 4) {
                    System.out.println("Client side quitting.");
                    break;
                }



                String operation;
                int value = 0;

                switch (option) {
                    case 1:
                        operation = "add";
                        System.out.println("Enter value to add:");
                        value = Integer.parseInt(reader.readLine());
                        break;
                    case 2:
                        operation = "subtract";
                        System.out.println("Enter value to subtract:");
                        value = Integer.parseInt(reader.readLine());
                        break;
                    case 3:
                        operation = "get";
                        break;
                    default:
                        System.out.println("Invalid option.");
                        continue;
                }
                System.out.println("Enter your ID: ");
                int id = Integer.parseInt(reader.readLine());
                String message = id + "," + operation + "," + value;
                add(serverPort, message);
            }
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        }
    }

    public static void add(int i, String message) {
        DatagramSocket aSocket = null;
        try {
            InetAddress aHost = InetAddress.getLocalHost();
            aSocket = new DatagramSocket();

            byte[] sendData = message.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, aHost, i);
            aSocket.send(sendPacket);

            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            aSocket.receive(receivePacket);

            String receivedMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("The result is " + receivedMessage + ".\n");

        } 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();
            }
        }
    }
}
```

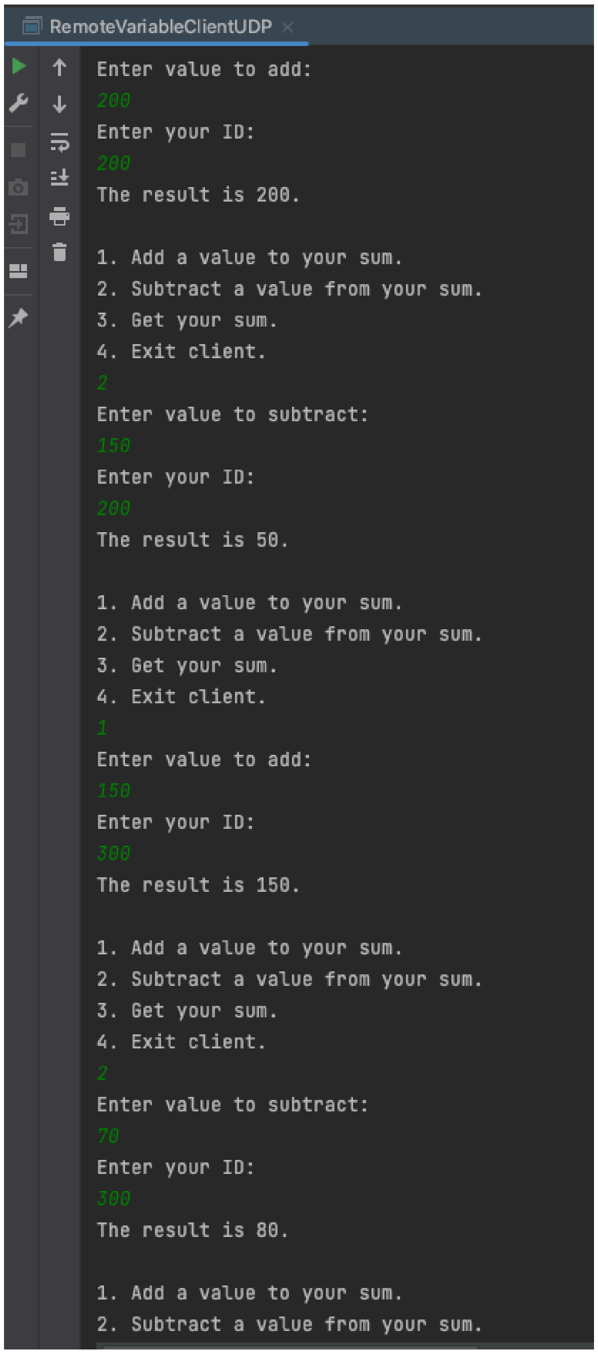
Project2Task3Server)

```java
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.util.HashMap;
import java.util.Map;

public class RemoteVariableServerUDP {
    public static Map<Integer, Integer> sumMap = new HashMap<>();

    public static void main(String args[]) {
        DatagramSocket aSocket = null;
        byte[] buffer = new byte[1000];
        try {
            System.out.println("The UDP server is running.");
            System.out.print("Enter server port number to listen on: ");
            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
            int serverPort = Integer.parseInt(reader.readLine());
            aSocket = new DatagramSocket(serverPort);
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);
            while (true) {
                aSocket.receive(request);
                byte[] requestData = new byte[request.getLength()];
                System.arraycopy(request.getData(), 0, requestData, 0,
request.getLength());
                String[] requestDataArray = new String(requestData).split(",");
                int id = Integer.parseInt(requestDataArray[0]);
                String operation = requestDataArray[1];
                int value = 0;

                if (requestDataArray.length == 3) {
                    value = Integer.parseInt(requestDataArray[2]);
                }

                int result = 0;
                switch (operation) {
                    case "add":
                        System.out.println("Adding: " + value + " to " +
get(id));
                        result = add(id, value);
                        break;
                    case "subtract":
                        System.out.println("Subtracting: " + value + " to " +
get(id));
                        result = subtract(id, value);
                        break;
                    case "get":
                        System.out.println("For ID: " + id);
                        result = get(id);
                        break;
                    default:
                        System.out.println("Invalid operation.");
                }
                System.out.println("Returning sum of " + result + " to client 
id: "+ id);
                System.out.println();


                String response = String.valueOf(result);
                DatagramPacket reply = new DatagramPacket(response.getBytes(), 
response.length(),
                        request.getAddress(), request.getPort());
                aSocket.send(reply);
            }
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            if (aSocket != null) aSocket.close();
        }
    }

    public static int add(int id, int value) {
        int sum = sumMap.getOrDefault(id, 0);
        sum += value;
        sumMap.put(id, sum);
        return sum;
    }

    public static int subtract(int id, int value) {
        int sum = sumMap.getOrDefault(id, 0);
        sum -= value;
        sumMap.put(id, sum);
        return sum;
    }

    public static int get(int id) {
        return sumMap.getOrDefault(id, 0);
    }
}
```

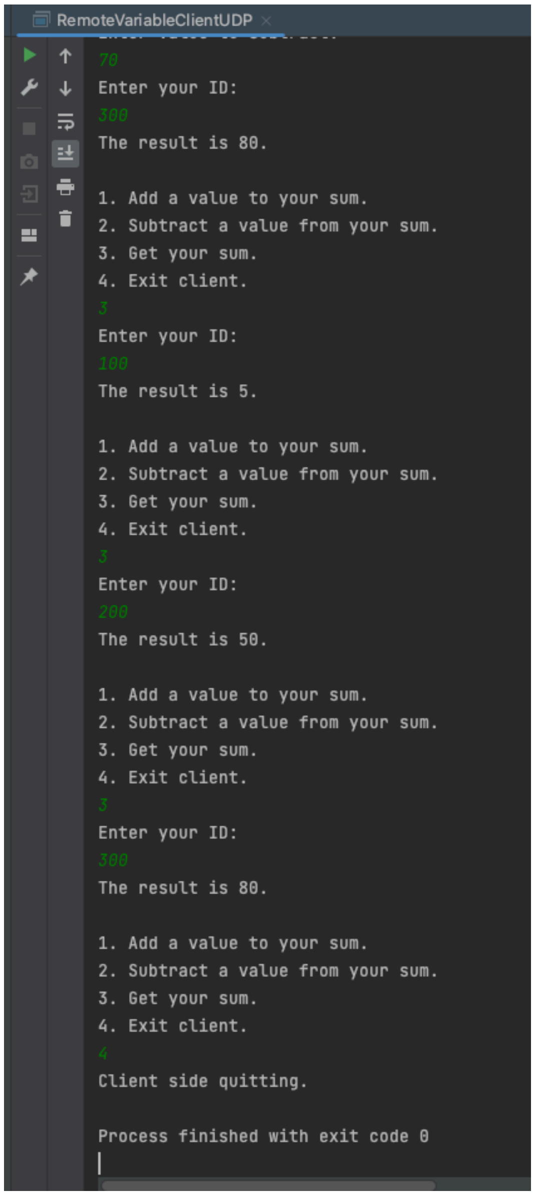
Project2Task3ClientConsole)



```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/temur
The client is running.
Please enter server port:
6789

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter value to add:
10
Enter your ID:
100
The result is 10.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter value to subtract:
5
Enter your ID:
100
The result is 5.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter value to add:
200
Enter your ID:
200
The result is 200.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
```

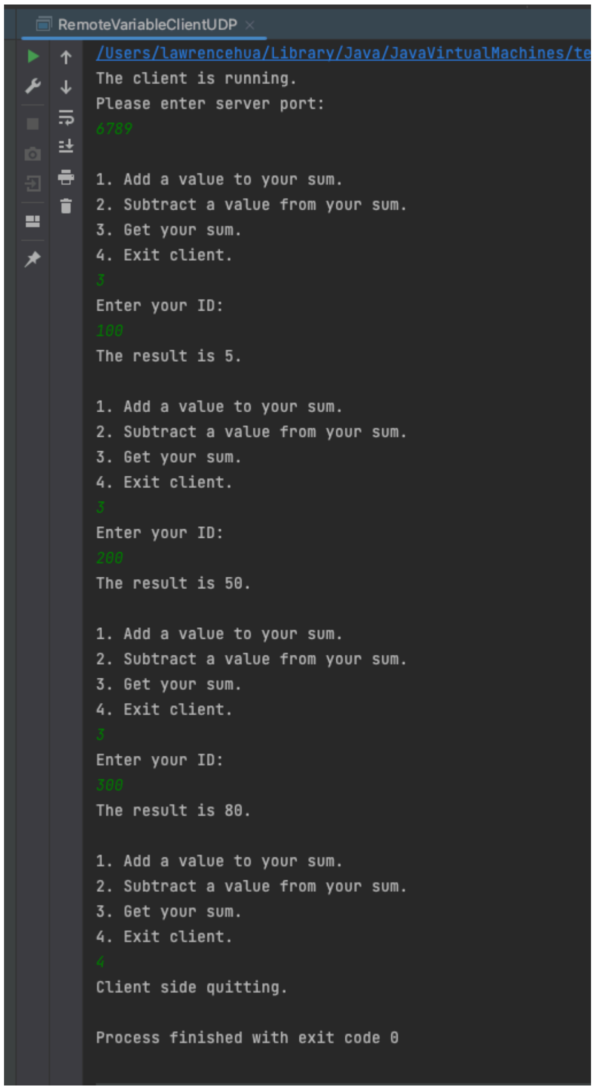


```
Enter value to add:
200
Enter your ID:
200
The result is 200.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter value to subtract:
150
Enter your ID:
200
The result is 50.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
1
Enter value to add:
150
Enter your ID:
300
The result is 150.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
2
Enter value to subtract:
70
Enter your ID:
300
The result is 80.

1. Add a value to your sum.
2. Subtract a value from your sum.
```



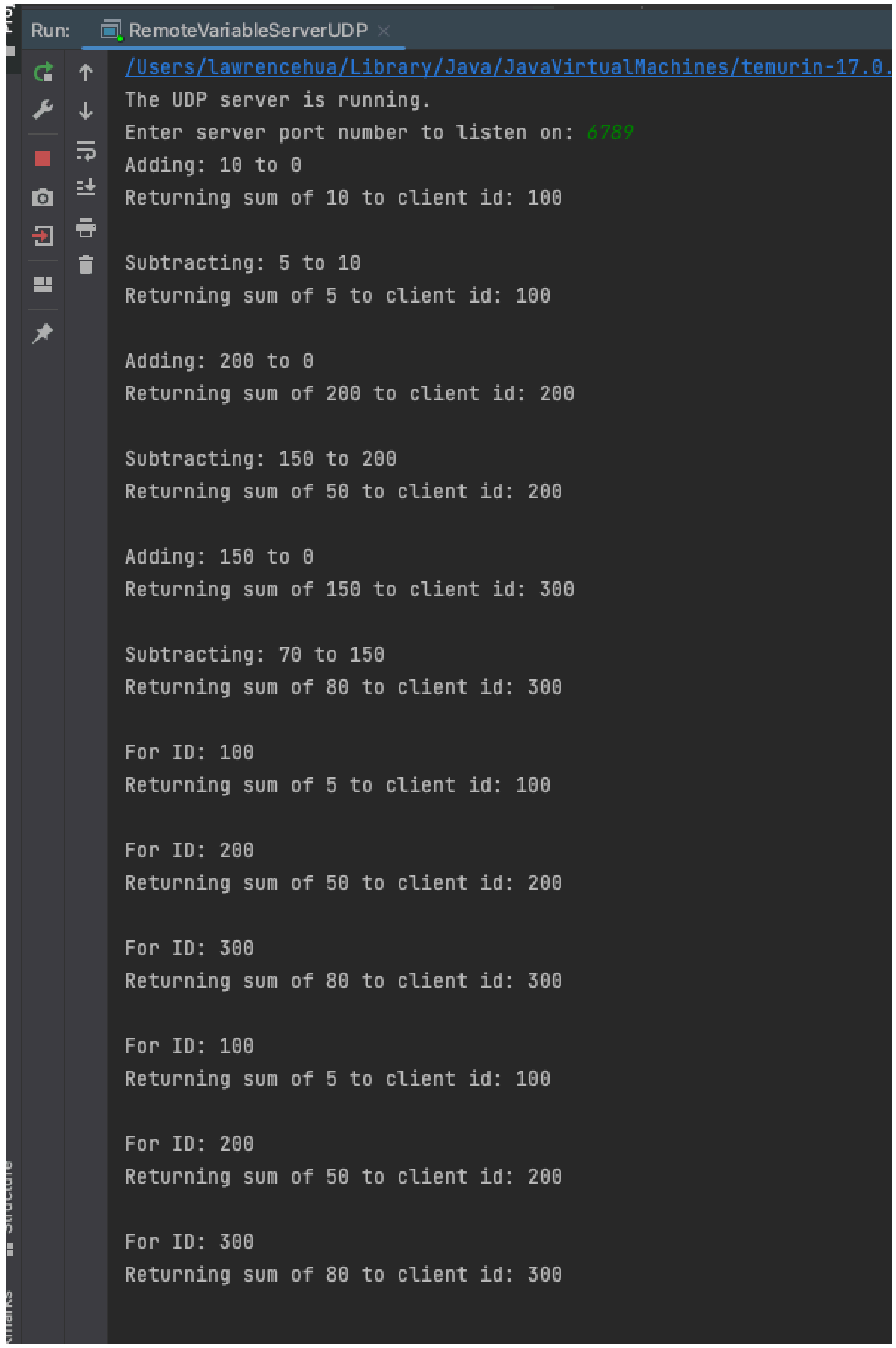
```
70
Enter your ID:
300
The result is 80.

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:
100
The result is 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:
200
The result is 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:
300
The result is 80.

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.

Process finished with exit code 0
```

THIS IS THE SECOND RUN JUST DOING GET ON ID’S 100, 200, 300*



```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/te
The client is running.
Please enter server port:
6789

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:
100
The result is 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:
200
The result is 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:
300
The result is 80.

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.

Process finished with exit code 0
```

Project2Task3ServerConsole)



```
/Users/lawrencehua/Library/Java/JavaVirtualMachines/temurin-17.0.
The UDP server is running.
Enter server port number to listen on: 6789
Adding: 10 to 0
Returning sum of 10 to client id: 100

Subtracting: 5 to 10
Returning sum of 5 to client id: 100

Adding: 200 to 0
Returning sum of 200 to client id: 200

Subtracting: 150 to 200
Returning sum of 50 to client id: 200

Adding: 150 to 0
Returning sum of 150 to client id: 300

Subtracting: 70 to 150
Returning sum of 80 to client id: 300

For ID: 100
Returning sum of 5 to client id: 100

For ID: 200
Returning sum of 50 to client id: 200

For ID: 300
Returning sum of 80 to client id: 300

For ID: 100
Returning sum of 5 to client id: 100

For ID: 200
Returning sum of 50 to client id: 200

For ID: 300
Returning sum of 80 to client id: 300
```

USED CHATGPT ON TASK 2 and 3:
https://chat.openai.com/share/82c0c9ec-420c-412d-a7f2-5d7ffa42c984