Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task0

Project2Task0Client

```
package ds.project2task0;

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

public class EchoClientUDP {
    public static void main(String[] args) {
        System.out.println("The UDP client is running.");
        DatagramSocket aSocket = null;
        Scanner scanner = new Scanner(System.in);

        try {
            // Prompt user for server port number
            System.out.print("Enter the server port number: ");
            int serverPort = scanner.nextInt();
            // Consume newline character
            scanner.nextLine();

            InetAddress aHost = InetAddress.getByName("localhost");
            aSocket = new DatagramSocket();

            System.out.println("Type a message to send. Type 'halt!' to
exit.");

            String nextLine;
            while ((nextLine = scanner.nextLine()) != null) {
                byte[] m = nextLine.getBytes();
                DatagramPacket request = new DatagramPacket(m, m.length,
aHost, serverPort);
                aSocket.send(request);

                byte[] buffer = new byte[1000];
                DatagramPacket reply = new DatagramPacket(buffer,
buffer.length);
                aSocket.receive(reply);

                // Correctly handle response size
                byte[] replyData = new byte[reply.getLength()];
                System.arraycopy(reply.getData(), 0, replyData, 0,
reply.getLength());
                String replyString = new String(replyData).trim();

                System.out.println("Reply from server: " + replyString);

                // Handle halt command
                if (replyString.equals("halt!")) {
                    System.out.println("Server received halt!");
                    System.out.println("UDP Client side quitting.");
                    break;
                }
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
            }
        } 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

```
package ds.project2task0;

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

public class EchoServerUDP {
    public static void main(String[] args) {
        System.out.println("The UDP server is running.");
        DatagramSocket aSocket = null;
        Scanner scanner = new Scanner(System.in);

        try {
            // Prompt user for port number
            System.out.print("Enter the port number to listen on: ");
            int port = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            aSocket = new DatagramSocket(port);
            System.out.println("The server is listening on port: " + port);

            while (true) {
                byte[] buffer = new byte[1000];
                DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
                aSocket.receive(request);

                // Correctly handle request string size
                byte[] requestData = new byte[request.getLength()];
                System.arraycopy(request.getData(), 0, requestData, 0,
request.getLength());
                String requestString = new String(requestData).trim();

                System.out.println("Echoing: " + requestString);
                DatagramPacket reply = new DatagramPacket(requestData,
requestData.length,
                        request.getAddress(), request.getPort());
                aSocket.send(reply);

                // Handle halt command
                if (requestString.equals("halt!")) {
                    System.out.println("Server received halt! command.");
                    System.out.println("UDP Server side quitting");
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
                    reply = new DatagramPacket("halt!".getBytes(),
"halt!".length(), request.getAddress(), request.getPort());
                    aSocket.send(reply);
                    break;
                }
            }
        } 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

```
The UDP client is running.
Enter the server port number: 6789
Type a message to send. Type 'halt!' to exit.
Hello
Reply from server: Hello
Hojoon Lee
Reply from server: Hojoon Lee
CMU
Reply from server: CMU
MISM
Reply from server: MISM
16
Reply from server: 16
halt!
Reply from server: halt!
Server received halt!
UDP Client side quitting.

Process finished with exit code 0
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task0ServerConsole

```
The UDP server is running.
Enter the port number to listen on: 6789
The server is listening on port: 6789
Echoing: Hello
Echoing: Hojoon Lee
Echoing: CMU
Echoing: MISM
Echoing: 16
Echoing: halt!
Server received halt! command.
UDP Server side quitting

Process finished with exit code 0
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task1

EavesdropperUDP.java

```
package ds.project2task1;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Scanner;

public class EavesDropperUDP {
    public static void main(String[] args) {
        System.out.println("Eavesdropper is active.");
        DatagramSocket eavesdropperSocket = null;
        Scanner scanner = new Scanner(System.in);

        try {
            // Get eavesdropper and server ports from user
            System.out.print("Enter the port for Eavesdropper to listen on:
");
            int listenPort = scanner.nextInt();
            System.out.print("Enter the masquerading server port: ");
            int serverPort = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            // Initialize the socket
            eavesdropperSocket = new DatagramSocket(listenPort);
            System.out.println("Eavesdropper listening on port " + listenPort
+ " and forwarding to server on port " + serverPort);

            byte[] buffer = new byte[1000];

            while (true) {
                DatagramPacket clientPacket = new DatagramPacket(buffer,
buffer.length);
                eavesdropperSocket.receive(clientPacket);

                // Extract and log message from client
                String interceptedMessage = new
String(clientPacket.getData(), 0, clientPacket.getLength());
                System.out.println("Intercepted from client: " +
interceptedMessage);

                // Modify first occurrence of "like" to "dislike"
                if (interceptedMessage.contains("like")) {
                    interceptedMessage =
interceptedMessage.replaceFirst("like", "dislike");
                    System.out.println("Modified message: " +
interceptedMessage);
                }

                // Forward to actual server
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
                byte[] modifiedData = interceptedMessage.getBytes();
                InetAddress serverAddress =
InetAddress.getByName("localhost");
                DatagramPacket serverPacket = new
DatagramPacket(modifiedData, modifiedData.length, serverAddress, serverPort);
                eavesdropperSocket.send(serverPacket);

                // Receive response from server
                DatagramPacket responsePacket = new DatagramPacket(buffer,
buffer.length);
                eavesdropperSocket.receive(responsePacket);

                // Forward response back to client
                DatagramPacket clientResponse = new
DatagramPacket(responsePacket.getData(), responsePacket.getLength(),
clientPacket.getAddress(), clientPacket.getPort());
                eavesdropperSocket.send(clientResponse);
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            if (eavesdropperSocket != null) eavesdropperSocket.close();
        }
    }
}
```

Project2Task1ThreeConsoles

EchoClientUDP consoles

```
UDP Client is active.
Enter the server port number: 6798
Client ready. Type a message and press Enter to send. Type 'halt!' to exit.
I like Chocolate
Server reply: I dislike Chocolate
halt!
Server reply: halt!
Server received halt request.
UDP Client terminating.
Client socket closed.
```

Hojoon Lee
hojoonle@andrew.cmu.edu

EchoServerUDP consoles

```
The UDP server is now active.
Enter the port number to listen on: 6789
Server is listening on port: 6789
Received from client: I dislike Chocolate
Echoed: I dislike Chocolate
Received from client: halt!
Server received halt command. Terminating...
Server socket closed.
```

EavesDropperUDP consoles

```
Eavesdropper is active.
Enter the port for Eavesdropper to listen on: 6798
Enter the masquerading server port: 6789
Eavesdropper listening on port 6798 and forwarding to server on port 6789
Intercepted from client: I like Chocolate
Modified message: I dislike Chocolate
Intercepted from client: halt!
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task2

Project2Task2Client

```
package ds.project2task2;

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.net.SocketException;
import java.util.Scanner;

public class AddingClientUDP {
    private static int serverPort;
    private static InetAddress serverAddress;
    private static DatagramSocket clientSocket;

    public static void main(String[] args) {
        System.out.println("Adding Client is now active.");
        Scanner scanner = new Scanner(System.in);

        try {
            // Prompt user for the server port number
            System.out.print("Enter the server port number: ");
            serverPort = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            // Set up server address and initialize the UDP socket
            serverAddress = InetAddress.getByName("localhost");
            clientSocket = new DatagramSocket();
            BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));

            System.out.println("Client ready. Enter numbers to send to the
server. Type 'halt!' to exit.");

            while (true) {
                String input = userInput.readLine().trim();

                // Handle termination command
                if (input.equals("halt!")) {
                    System.out.println("Client terminating.");
                    break;
                }

                try {
                    int number = Integer.parseInt(input);
                    int result = add(number);
                    System.out.println("The server returned " + result +
".");
                } catch (NumberFormatException e) {
                    System.out.println("Invalid input. Please enter a valid
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
integer.");
                }
            }
        } catch (SocketException e) {
            System.out.println("Socket error: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O error: " + e.getMessage());
        } finally {
            if (clientSocket != null) {
                clientSocket.close();
                System.out.println("Client socket closed.");
            }
        }
    }

    public static int add(int number) {
        try {
            // Convert integer to byte array before sending
            byte[] requestBytes = String.valueOf(number).getBytes();
            DatagramPacket requestPacket = new DatagramPacket(requestBytes,
requestBytes.length, serverAddress, serverPort);
            clientSocket.send(requestPacket);

            // Prepare buffer and receive response from server
            byte[] buffer = new byte[1000];
            DatagramPacket responsePacket = new DatagramPacket(buffer,
buffer.length);
            clientSocket.receive(responsePacket);

            // Extract and return the received integer
            return Integer.parseInt(new String(responsePacket.getData(), 0,
responsePacket.getLength()));
        } catch (IOException e) {
            System.out.println("Error communicating with server: " +
e.getMessage());
            return -1;
        }
    }
}
```

Project2Task2Server

```
package ds.project2task2;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.Scanner;

/*
 * The AddingServerUDP class implements a UDP server that maintains a running
sum.
 * It listens for client requests containing integers, adds them to the sum,
 * and returns the updated sum to the client.
 */
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
public class AddingServerUDP {
    // Stores the cumulative sum
    private static int sumValue = 0;

    public static void main(String[] args) {
        System.out.println("Adding Server is now active.");
        DatagramSocket serverSocket = null;
        Scanner scanner = new Scanner(System.in);

        try {
            // Prompt user for the port number
            System.out.print("Enter the port number to listen on: ");
            int serverPort = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            // Initialize server socket
            serverSocket = new DatagramSocket(serverPort);
            System.out.println("Server is listening on port: " + serverPort);

            byte[] buffer = new byte[1000];

            while (true) {
                DatagramPacket requestPacket = new DatagramPacket(buffer,
buffer.length);
                serverSocket.receive(requestPacket);

                // Extract client message and parse integer
                String clientMessage = new String(requestPacket.getData(), 0,
requestPacket.getLength()).trim();
                if (clientMessage.equals("halt!")) {
                    System.out.println("Server received halt command.
Terminating...");
                    break;
                }

                int number;
                try {
                    number = Integer.parseInt(clientMessage);
                } catch (NumberFormatException e) {
                    System.out.println("Invalid input received: " +
clientMessage);
                    continue;
                }

                // Perform addition operation
                sumValue = add(number);
                System.out.println("Adding: " + number + " to " + (sumValue -
number));
                System.out.println ("Returning sum of " + sumValue + " to
client");

                // Send updated sum back to client
                byte[] responseBytes = String.valueOf(sumValue).getBytes();
                DatagramPacket responsePacket = new DatagramPacket(
                        responseBytes, responseBytes.length,
requestPacket.getAddress(), requestPacket.getPort()
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
                );
                serverSocket.send(responsePacket);
            }
        } catch (SocketException e) {
            System.out.println("Socket error: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O error: " + e.getMessage());
        } finally {
            if (serverSocket != null) {
                serverSocket.close();
                System.out.println("Server socket closed.");
            }
        }
    }

    private static int add(int number) {
        return sumValue + number;
    }
}
```

Project2Task2ClientConsole

```
Adding Client is now active.
Enter the server port number: 6789
Client ready. Enter numbers to send to the server. Type 'halt!' to exit.
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.
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
Adding Client is now active.
Enter the server port number: 6789
Client ready. Enter numbers to send to the server. Type 'halt!' to exit.
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 terminating.
Client socket closed.
```

Project2Task2ServerConsole

```
Adding Server is now active.
Enter the port number to listen on: 6789
Server is listening on port: 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
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task3

Project2Task3Client

```
package ds.project2task3;

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

public class RemoteVariableClientUDP {
    private static RemoteVariableProxy proxy;

    public static void main(String[] args) {
        System.out.println("The client is running.");
        Scanner scanner = new Scanner(System.in);

        try {
            //Prompt user for the server port number
            System.out.print("Please enter server port: ");
            int serverPort = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            //Initialize the proxy to handle communication with the server
            proxy = new RemoteVariableProxy(serverPort, "localhost");

            while (true) {
                // Display menu options for the user
                System.out.println("\n1. 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.");
                System.out.print("Choose an option: ");
                int selection = scanner.nextInt();
                scanner.nextLine(); // Consume newline character

                if (selection == 4) {
                    System.out.println("Client side quitting. The remote
variable server is still running.");
                    break;
                }

                String operation = "";
                int value = 0;

                if (selection == 1 || selection == 2) {
                    System.out.print(selection == 1 ? "Enter value to add:
" : "Enter value to subtract: ");
                    value = scanner.nextInt();
                    scanner.nextLine();
                }

                System.out.print("Enter your ID: ");
                int userID = scanner.nextInt();
                scanner.nextLine(); // Consume newline character
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
                while (userID < 0 || userID > 999) {
                    System.out.println("Invalid ID. Please try again.");
                    userID = scanner.nextInt();
                    scanner.nextLine();
                }

                //Determine the operation type based on the user's selection
                switch (selection) {
                    case 1:
                        operation = "add";
                        break;
                    case 2:
                        operation = "subtract";
                        break;
                    case 3:
                        operation = "get";
                        break;
                    default:
                        System.out.println("Invalid option. Please try
again.");
                        continue;
                }

                // Send request to the server using the proxy and display the
result
                int result = proxy.sendRequestToServer(userID, operation,
value);
                System.out.println("The result is " + result + ".");
            }
        } catch (SocketException e) {
            System.out.println("Socket error: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O error: " + e.getMessage());
        } finally {
            if (proxy.getSocket() != null) {
                proxy.getSocket().close();
                System.out.println("Client socket closed.");
            }
        }
    }
}

//Implementing the proxy design pattern. It handles sending and receiving UDP
packets.
class RemoteVariableProxy {
    private int serverPort;
    private InetAddress serverAddress;
    private DatagramSocket socket;

    //Constructor
    public RemoteVariableProxy(int serverPort, String serverHost) throws
UnknownHostException, SocketException {
        this.serverPort = serverPort;
        this.serverAddress = InetAddress.getByName(serverHost);
        this.socket = new DatagramSocket();
    }
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
    //Getter
    public DatagramSocket getSocket() {
        return socket;
    }

    //Sends a request to the server and retrieves the updated sum.
    public int sendRequestToServer(int userID, String operation, int value) {
        try {
            String requestMessage = userID + " " + operation +
(operation.equals("get") ? "" : " " + value);
            byte[] requestBytes = requestMessage.getBytes();
            DatagramPacket requestPacket = new DatagramPacket(requestBytes,
requestBytes.length, serverAddress, serverPort);
            socket.send(requestPacket);

            // Prepare buffer and receive response from server
            byte[] buffer = new byte[1000];
            DatagramPacket responsePacket = new DatagramPacket(buffer,
buffer.length);
            socket.receive(responsePacket);

            return Integer.parseInt(new String(responsePacket.getData(), 0,
responsePacket.getLength()).trim());
        } catch (IOException e) {
            System.out.println("Error communicating with server: " +
e.getMessage());
            return -1;
        }
    }
}
```

Project2Task3Server

```
package ds.project2task3;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;

public class RemoteVariableServerUDP {
    // Request handler to manage user data
    private static RemoteVariableRequestHandler requestHandler = new
RemoteVariableRequestHandler();

    public static void main(String[] args) {
        System.out.println("Remote Variable Server is now active.");
        DatagramSocket serverSocket = null;
        byte[] buffer = new byte[1000];
        Scanner scanner = new Scanner(System.in);

        try {
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
            // Prompt user for server port number
            System.out.print("Enter the port number to listen on: ");
            int serverPort = scanner.nextInt();
            serverSocket = new DatagramSocket(serverPort);
            System.out.println("Server listening on port: " + serverPort);

            while (true) {
                DatagramPacket requestPacket = new DatagramPacket(buffer,
buffer.length);
                serverSocket.receive(requestPacket);

                // Extract client request message
                String clientMessage = new String(requestPacket.getData(), 0,
requestPacket.getLength()).trim();
                String[] userInputs = clientMessage.split(" ");

                // Validate input format
                if (userInputs.length < 2 || userInputs.length > 3) {
                    System.out.println("Invalid request format: " +
clientMessage);
                    continue;
                }

                int userID;
                try {
                    userID = Integer.parseInt(userInputs[0]);
                } catch (NumberFormatException e) {
                    System.out.println("Invalid user ID format: " +
userInputs[0]);
                    continue;
                }

                if (userID < 0 || userID > 999) {
                    System.out.println("Invalid user ID: " + userID);
                    continue;
                }

                String operation = userInputs[1];
                int value = (userInputs.length == 3) ?
Integer.parseInt(userInputs[2]) : 0;

                // Process client request and return result
                int result = requestHandler.handleRequest(userID, operation,
value);
                System.out.println("Processed request from User " + userID +
" | Operation: " + operation + " | Result: " + result);

                // Send response to client
                byte[] responseData = String.valueOf(result).getBytes();
                DatagramPacket responsePacket = new DatagramPacket(
                        responseData, responseData.length,
requestPacket.getAddress(), requestPacket.getPort()
                );
                serverSocket.send(responsePacket);
            }
        } catch (SocketException e) {
            System.out.println("Socket error: " + e.getMessage());
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
        } catch (IOException e) {
            System.out.println("I/O error: " + e.getMessage());
        } finally {
            if (serverSocket != null) {
                serverSocket.close();
                System.out.println("Server socket closed.");
            }
        }
    }
}

//Uses a TreeMap to store and update user-specific sums.
class RemoteVariableRequestHandler {
    private Map<Integer, Integer> userSums = new TreeMap<>();

    //Processes the client's request and updates the stored sum accordingly.
    public int handleRequest(int userID, String operation, int value) {
        userSums.putIfAbsent(userID, 0);

        switch (operation.toLowerCase()) {
            case "add":
                userSums.put(userID, userSums.get(userID) + value);
                break;
            case "subtract":
                userSums.put(userID, userSums.get(userID) - value);
                break;
            case "get":
                break; // Simply return the current sum
            default:
                System.out.println("Unknown operation: " + operation);
                return -1;
        }
        return userSums.get(userID);
    }
}
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task3ClientConsole

```
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.
Choose an option: 1
Enter value to add: 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.
Choose an option: 2
Enter value to subtract: 3
Enter your ID: 100
The result is 2.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 3
Enter your ID: 100
The result is 2.
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task3ClientConsole (Continued)

```
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 3
Enter your ID: 100
The result is 2.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 1
Enter value to add: 200
Enter your ID: 101
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.
Choose an option: 2
Enter value to subtract: 100
Enter your ID: 101
The result is 100.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 3
Enter your ID: 101
The result is 100.
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task3ClientConsole (Continued)

```
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 3
Enter your ID: 101
The result is 100.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 1
Enter value to add: 3
Enter your ID: 7
The result is 3.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 2
Enter value to subtract: 2
Enter your ID: 7
The result is 1.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 3
Enter your ID: 7
The result is 1.
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task3ClientConsole (Continued)

```
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.
Choose an option: 3
Enter your ID: 100
The result is 2.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 3
Enter your ID: 101
The result is 100.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option: 3
Enter your ID: 7
The result is 1.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Choose an option:
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task3ServerConsole

```
Remote Variable Server is now active.
Enter the port number to listen on: 6789
Server listening on port: 6789
Processed request from User 100 | Operation: add | Result: 5
Processed request from User 100 | Operation: subtract | Result: 2
Processed request from User 100 | Operation: get | Result: 2
Processed request from User 101 | Operation: add | Result: 200
Processed request from User 101 | Operation: subtract | Result: 100
Processed request from User 101 | Operation: get | Result: 100
Processed request from User 7 | Operation: add | Result: 3
Processed request from User 7 | Operation: subtract | Result: 1
Processed request from User 7 | Operation: get | Result: 1
Processed request from User 100 | Operation: get | Result: 2
Processed request from User 101 | Operation: get | Result: 100
Processed request from User 7 | Operation: get | Result: 1
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task4

Project2Task4Client

```
package ds.project2task4;

import java.io.IOException;
import java.net.*;
import java.util.*;
import com.google.gson.*;

public class NeuralNetworkClient {
    private static NeuralNetworkProxy proxy;
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("The Neural Network Client is running.");
        try {
            // Prompt user for the server port number
            System.out.print("Please enter server port: ");
            int serverPort = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            // Initialize the proxy to handle communication with the server
            proxy = new NeuralNetworkProxy(serverPort, "localhost");

            while (true) {
                int userSelection = menu();

                if (userSelection < 0 || userSelection > 5) {
                    System.out.println("Invalid selection. Please enter a
number between 0 and 5.");
                    continue;
                }

                // Create a JSON request object
                JsonObject request = new JsonObject();

                switch (userSelection) {
                    case 0:
                        request.addProperty("request", "getCurrentRange");
                        break;
                    case 1:
                        System.out.println("Enter four results for the truth
table (0 or 1 each):");
                        request.addProperty("request", "setCurrentRange");
                        request.addProperty("val1", scanner.nextDouble());
                        request.addProperty("val2", scanner.nextDouble());
                        request.addProperty("val3", scanner.nextDouble());
                        request.addProperty("val4", scanner.nextDouble());
                        scanner.nextLine(); // Consume newline character
                        break;
                    case 2:
                        request.addProperty("request", "train");
                        request.addProperty("iterations", 1);
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
                        break;
                    case 3:
                        System.out.print("Enter the number of training steps:
");
                        request.addProperty("request", "train");
                        request.addProperty("iterations", scanner.nextInt());
                        scanner.nextLine(); // Consume newline character
                        break;
                    case 4:
                        System.out.println("Enter two input values (0 or 1
each):");
                        request.addProperty("request", "test");
                        request.addProperty("val1", scanner.nextDouble());
                        request.addProperty("val2", scanner.nextDouble());
                        scanner.nextLine(); // Consume newline character
                        break;
                    case 5:
                        System.out.println("Client is shutting down...");
                        System.exit(0);
                        break;
                }

                // Send request to the server and process the response
                String response = proxy.sendJsonRequest(request);
                JsonObject jsonResponse =
JsonParser.parseString(response).getAsJsonObject();
                String status = jsonResponse.get("status").getAsString();

                System.out.println("Server response status: " + status);

                if (status.equals("OK")) {
                    switch (userSelection) {
                        case 0:
                            System.out.println("Truth Table:");
                            System.out.printf("0.0  0.0  %.1f%n",
jsonResponse.get("val1").getAsDouble());
                            System.out.printf("0.0  1.0  %.1f%n",
jsonResponse.get("val2").getAsDouble());
                            System.out.printf("1.0  0.0  %.1f%n",
jsonResponse.get("val3").getAsDouble());
                            System.out.printf("1.0  1.0  %.1f%n",
jsonResponse.get("val4").getAsDouble());
                            break;
                        case 1:
                            System.out.println("Range values successfully
updated.");
                            break;
                        case 2:
                            System.out.println("After this step, total error:
" + jsonResponse.get("val1").getAsDouble());
                            break;
                        case 3:
                            System.out.println("After training, total error:
" + jsonResponse.get("val1").getAsDouble());
                            break;
                        case 4:
                            System.out.println("The computed output is
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
approximately: " + jsonResponse.get("val1").getAsDouble());
                            break;
                    }
                }
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O Exception: " + e.getMessage());
        } finally {
            if (proxy.getSocket() != null) proxy.getSocket().close();
        }
    }

    //Displays the menu and returns the user's selection.
    public static int menu() {
        System.out.println("\nUsing a neural network to learn a truth
table.\nMain Menu");
        System.out.println("0. Display the current truth table.");
        System.out.println("1. Set new truth table values.");
        System.out.println("2. Perform a single training step.");
        System.out.println("3. Perform multiple training steps.");
        System.out.println("4. Test with input values.");
        System.out.println("5. Exit program.");
        System.out.print("Choose an option: ");
        return scanner.nextInt();
    }
}

//Sends JSON requests over UDP and receives responses from the server.
class NeuralNetworkProxy {
    private int serverPort;
    private InetAddress serverAddress;
    private DatagramSocket socket;

//Constructor
    public NeuralNetworkProxy(int serverPort, String serverHost) throws
UnknownHostException, SocketException {
        this.serverPort = serverPort;
        this.serverAddress = InetAddress.getByName(serverHost);
        this.socket = new DatagramSocket();
    }

    //Active Socket
    public DatagramSocket getSocket() {
        return socket;
    }

    //Sends a JSON request to the server and returns the response.
    public String sendJsonRequest(JsonObject request) throws IOException {
        String jsonString = request.toString();
        byte[] requestBytes = jsonString.getBytes();
        DatagramPacket requestPacket = new DatagramPacket(requestBytes,
requestBytes.length, serverAddress, serverPort);
        socket.send(requestPacket);

        byte[] responseBytes = new byte[1000];
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
        DatagramPacket responsePacket = new DatagramPacket(responseBytes,
responseBytes.length);
        socket.receive(responsePacket);

        return new String(responsePacket.getData(), 0,
responsePacket.getLength());
    }
}
```

Project2Task4Server

```
package ds.project2task4;

import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.*;

public class NeuralNetworkServer {
    private static double val1, val2, val3, val4;
    private static double totalError;
    private static NeuralNetwork neuralNetwork;
    private static ArrayList<Double[][]> userTrainingSets;
    private static Random rand = new Random();

    public static void main(String[] args) {
        System.out.println("Neural Network Server is running...");
        DatagramSocket socket = null;
        byte[] buffer = new byte[1000];
        Scanner scanner = new Scanner(System.in);

        // Initialize a truth table with default values of 0
        userTrainingSets = new ArrayList<>(Arrays.asList(
                new Double[][]{{0.0, 0.0}, {0.0}},
                new Double[][]{{0.0, 1.0}, {0.0}},
                new Double[][]{{1.0, 0.0}, {0.0}},
                new Double[][]{{1.0, 1.0}, {0.0}}
        ));

        // Initialize the neural network with random weights and biases
        neuralNetwork = new NeuralNetwork(2, 5, 1, null, null, null, null);

        try {
            // Prompt for the server port
            System.out.print("Enter the port number to listen on: ");
            int serverPort = scanner.nextInt();
            socket = new DatagramSocket(serverPort);
            System.out.println("Listening on port: " + serverPort);

            while (true) {
                DatagramPacket requestPacket = new DatagramPacket(buffer,
buffer.length);
                socket.receive(requestPacket);
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
                // Convert received data to a JSON request
                String requestData = new String(requestPacket.getData(), 0,
requestPacket.getLength());
                JsonObject jsonRequest =
JsonParser.parseString(requestData).getAsJsonObject();

                // Print the received JSON request exactly as it is
                System.out.println(jsonRequest.toString());

                JsonObject jsonResponse = processClientRequest(jsonRequest);

                // Print the JSON response before sending
                System.out.println(jsonResponse.toString());

                // Send response back to the client
                byte[] responseData = jsonResponse.toString().getBytes();
                DatagramPacket responsePacket = new
DatagramPacket(responseData, responseData.length,
                        requestPacket.getAddress(), requestPacket.getPort());
                socket.send(responsePacket);
            }
        } catch (SocketException e) {
            System.out.println("Socket error: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO error: " + e.getMessage());
        } finally {
            if (socket != null) socket.close();
        }
    }

    // Processes the incoming JSON request from the client and generates the
appropriate response.
    private static JsonObject processClientRequest(JsonObject request) {
        JsonObject response = new JsonObject();
        String requestType = request.get("request").getAsString();

        switch (requestType) {
            case "getCurrentRange":
                response.addProperty("response", "getCurrentRange");
                response.addProperty("status", "OK");
                response.addProperty("val1", val1);
                response.addProperty("val2", val2);
                response.addProperty("val3", val3);
                response.addProperty("val4", val4);
                break;

            case "setCurrentRange":
                val1 = request.get("val1").getAsDouble();
                val2 = request.get("val2").getAsDouble();
                val3 = request.get("val3").getAsDouble();
                val4 = request.get("val4").getAsDouble();

                userTrainingSets = new ArrayList<>(Arrays.asList(
                        new Double[][]{{0.0, 0.0}, {val1}},
                        new Double[][]{{0.0, 1.0}, {val2}},
                        new Double[][]{{1.0, 0.0}, {val3}},
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
                        new Double[][]{{1.0, 1.0}, {val4}}
                ));

                neuralNetwork = new NeuralNetwork(2, 5, 1, null, null, null,
null);
                response.addProperty("response", "setCurrentRange");
                response.addProperty("status", "OK");
                break;

            case "train":
                int iterations = request.get("iterations").getAsInt();
                totalError = 0.0;
                for (int i = 0; i < iterations; i++) {
                    int randomChoice = rand.nextInt(4);
                    List<Double> inputs =
Arrays.asList(userTrainingSets.get(randomChoice)[0]);
                    List<Double> outputs =
Arrays.asList(userTrainingSets.get(randomChoice)[1]);
                    neuralNetwork.train(inputs, outputs);
                    totalError =
neuralNetwork.calculateTotalError(userTrainingSets);
                }
                response.addProperty("response", "train");
                response.addProperty("status", "OK");
                response.addProperty("val1", totalError);
                break;

            case "test":
                double testVal1 = request.get("val1").getAsDouble();
                double testVal2 = request.get("val2").getAsDouble();
                List<Double> testInputs = Arrays.asList(testVal1, testVal2);
                List<Double> testResult =
neuralNetwork.feedForward(testInputs);
                response.addProperty("response", "test");
                response.addProperty("status", "OK");
                response.addProperty("val1", testResult.get(0));
                break;

            default:
                response.addProperty("response", "error");
                response.addProperty("status", "ERROR");
        }
        return response;
    }
}
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task4ClientConsole

Logical XOR Operation

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 0
Server response status: OK
Truth Table:
0.0  0.0  0.0
0.0  1.0  0.0
1.0  0.0  0.0
1.0  1.0  0.0

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 1
Enter four results for the truth table (0 or 1 each):
0 1 1 0
Server response status: OK
Range values successfully updated.
```

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 2
Server response status: OK
After this step, total error: 0.8570928299766246

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 3
Enter the number of training steps: 10000
Server response status: OK
After training, total error: 0.0085383144388307
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
0 0
Server response status: OK
The computed output is approximately: 0.07938769347505925

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
0 1
Server response status: OK
The computed output is approximately: 0.9346800657686434
```

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
1 0
Server response status: OK
The computed output is approximately: 0.9397329849629598

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
1 1
Server response status: OK
The computed output is approximately: 0.05362290641981134
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 5
```

Logical OR Operation

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 0
Server response status: OK
Truth Table:
0.0  0.0  0.0
0.0  1.0  1.0
1.0  0.0  1.0
1.0  1.0  0.0

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 1
Enter four results for the truth table (0 or 1 each):
0 1 1 1
Server response status: OK
Range values successfully updated.
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 0
Server response status: OK
Truth Table:
0.0  0.0  0.0
0.0  1.0  1.0
1.0  0.0  1.0
1.0  1.0  1.0

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 2
Server response status: OK
After this step, total error: 0.43521503487410956
```

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 3
Enter the number of training steps: 10000
Server response status: OK
After training, total error: 0.0015515827304027714

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
0 0
Server response status: OK
The computed output is approximately: 0.04129430934225464
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
1 1
Server response status: OK
The computed output is approximately: 0.9915528059901597

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
0 1
Server response status: OK
The computed output is approximately: 0.9736403770591001

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
1 0
Server response status: OK
The computed output is approximately: 0.974865150318225

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 5
Client is shutting down...

Process finished with exit code 0
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Logical AND Operation

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 0
Server response status: OK
Truth Table:
0.0  0.0  0.0
0.0  1.0  1.0
1.0  0.0  1.0
1.0  1.0  1.0

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 1
Enter four results for the truth table (0 or 1 each):
0 0 0 1
Server response status: OK
Range values successfully updated.
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 0
Server response status: OK
Truth Table:
0.0  0.0  0.0
0.0  1.0  0.0
1.0  0.0  0.0
1.0  1.0  1.0

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 2
Server response status: OK
After this step, total error: 1.0417252582792937
```

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 3
Enter the number of training steps: 10000
Server response status: OK
After training, total error: 0.0021899591785878034

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
1 0
Server response status: OK
The computed output is approximately: 0.0332282860449365
```

Hojoon Lee
hojoonle@andrew.cmu.edu

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
0 1
Server response status: OK
The computed output is approximately: 0.034790141675404944

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
0 0
Server response status: OK
The computed output is approximately: 5.463524038683752E-4
```

```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 4
Enter two input values (0 or 1 each):
1 1
Server response status: OK
The computed output is approximately: 0.9545561125678323

Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Set new truth table values.
2. Perform a single training step.
3. Perform multiple training steps.
4. Test with input values.
5. Exit program.
Choose an option: 5
Client is shutting down...
```

Hojoon Lee
hojoonle@andrew.cmu.edu

Project2Task4ServerConsole

```
Neural Network Server is running...
Enter the port number to listen on: 6789
Listening on port: 6789
{"request":"getCurrentRange"}
{"response":"getCurrentRange","status":"OK","val1":0.0,"val2":0.0,"val3":0.0,"val4":0.0}
{"request":"setCurrentRange","val1":0.0,"val2":1.0,"val3":1.0,"val4":0.0}
{"response":"setCurrentRange","status":"OK"}
{"request":"train","iterations":1}
{"response":"train","status":"OK","val1":0.8570928299766246}
{"request":"train","iterations":10000}
{"response":"train","status":"OK","val1":0.0085383144388307}
{"request":"test","val1":0.0,"val2":0.0}
{"response":"test","status":"OK","val1":0.07938769347505925}
{"request":"test","val1":0.0,"val2":1.0}
{"response":"test","status":"OK","val1":0.9346800657686434}
{"request":"test","val1":1.0,"val2":0.0}
{"response":"test","status":"OK","val1":0.9397329849629598}
{"request":"test","val1":1.0,"val2":1.0}
{"response":"test","status":"OK","val1":0.05362290641981134}
{"request":"getCurrentRange"}
{"response":"getCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":0.0}
{"request":"setCurrentRange","val1":0.0,"val2":1.0,"val3":1.0,"val4":1.0}
{"response":"setCurrentRange","status":"OK"}
{"request":"getCurrentRange"}
{"response":"getCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":1.0}
{"request":"train","iterations":1}
{"response":"train","status":"OK","val1":0.43521503487410956}
{"request":"train","iterations":10000}
{"response":"train","status":"OK","val1":0.0015515827304027714}
{"request":"test","val1":0.0,"val2":0.0}
{"response":"test","status":"OK","val1":0.04129430934225464}
{"request":"test","val1":1.0,"val2":1.0}
{"response":"test","status":"OK","val1":0.9915528059901597}
{"request":"test","val1":0.0,"val2":1.0}
{"response":"test","status":"OK","val1":0.9736403770591001}
{"request":"test","val1":1.0,"val2":0.0}
{"response":"test","status":"OK","val1":0.974865150318225}
{"request":"getCurrentRange"}
{"response":"getCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":1.0}
{"request":"setCurrentRange","val1":0.0,"val2":0.0,"val3":0.0,"val4":1.0}
```

Hojoon Lee
hojoonle@andrew.cmu.edu


```
{"request":"getCurrentRange"}
{"response":"getCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":1.0}
{"request":"setCurrentRange","val1":0.0,"val2":0.0,"val3":0.0,"val4":1.0}
{"response":"setCurrentRange","status":"OK"}
{"request":"getCurrentRange"}
{"response":"getCurrentRange","status":"OK","val1":0.0,"val2":0.0,"val3":0.0,"val4":1.0}
{"request":"train","iterations":1}
{"response":"train","status":"OK","val1":1.0417252582792937}
{"request":"train","iterations":10000}
{"response":"train","status":"OK","val1":0.0021899591785878034}
{"request":"test","val1":1.0,"val2":0.0}
{"response":"test","status":"OK","val1":0.0332282860449365}
{"request":"test","val1":0.0,"val2":1.0}
{"response":"test","status":"OK","val1":0.034790141675404944}
{"request":"test","val1":0.0,"val2":0.0}
{"response":"test","status":"OK","val1":5.463524038683752E-4}
{"request":"test","val1":1.0,"val2":1.0}
{"response":"test","status":"OK","val1":0.9545561125678323}
```