PROJECT 2

Name: Shivani Poovaiah Ajjikutira

Email ID: sajjikut@andrew.cmu.edu

1. Project2Task1Client

```
oublic class EchoClientUDP{
            int serverPort = 6789;
InputStreamReader(System.in));
aHost, serverPort);
```

```
checkString = new String(m);
buffer.length);
                    String replyString = new String(replyBytes);
                    typed = null;
                    aSocket.close();
            System.out.println("Socket: " + e.getMessage());
```

2. Project2Task1Server

```
byte[] buffer = new byte[1000];
buffer.length);
                System.out.println("Sending data to port number: " +
                DatagramPacket reply = new DatagramPacket(request.getData(),
request.getPort());
```

3. Project2Task1ClientScreen

4. Project2Task1ServerScreen

```
"C:\Program Files\Eclipse Foundation\jdk-16.0.2.7-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Intelli
The server is running.
Sending data to port number: 62343
Echoing: Submit to Canvas a single PDF file named Your_Last_Name_First_Name_Project2.pdf.
Sending data to port number: 62343
Echoing: The single PDF will contain your responses to the questions marked with a checkered flag.
Sending data to port number: 62343
Echoing: It is important that you clearly label each answer with the labels provided below.
Sending data to port number: 62343
Echoing: It is also important to be prepared to demonstrate your working code if we need to verify your submission.
Sending data to port number: 62343
Echoing: Be sure to provide your name and email address at the top of the PDF submission.
Sending data to port number: 62343
Echoing: halt!
Server side quitting.
Process finished with exit code 0
```

5. Project2Task2Client

```
System.out.println("IO: " + e.getMessage());
    if(j < Integer.toString(i).getBytes().length) {</pre>
        message[j] = Integer.toString(i).getBytes()[j];
DatagramPacket request = new DatagramPacket (message,
```

6. Project2Task2Server

```
/*
    * @author: Shivani Poovaiah Ajjikutira
    * Last Modified: 8th October 2021
    *
    * The following code is the server side for a program that adds numbers.
    * The server socket is initialized with port number 6789 and forms a
    * connection with any client socket trying to connect to port number 6789.
    * The client passes the integers to the server through DatagramPacket.
    * The addition logic happens in the server in the add method. The program
    * on the server side keeps running always.
    * */

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramPacket;
import java.net.SocketException;

public class AddingServletUDP {
    public static void main(String[] args) {
        System.out.println("Server started");
    }
}
```

```
byte[] buffer = new byte[1000];
buffer.length);
                aSocket.receive(request);
request.getPort());
                System.arraycopy(request.qetData(), 0, requestBytes, 0,
                String requestString = new String(requestBytes);
                reply.setData(replyBytes);
```

```
public static int add(int i, int j) {
    System.out.printf("Adding: %d to %d%n",i,j);
    return i+j;
}
```

7. Project2Task2ClientScreen

```
AddingServletUDP × AddingClientUDP ×

"C:\Program Files\Eclipse Foundation\jd
The client is running.

The server returned 1.

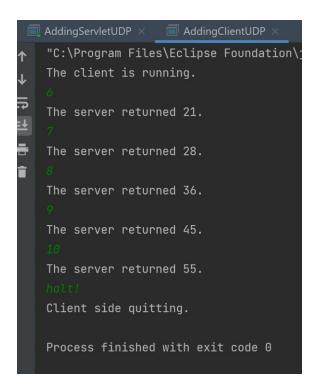
The server returned 3.

The server returned 6.

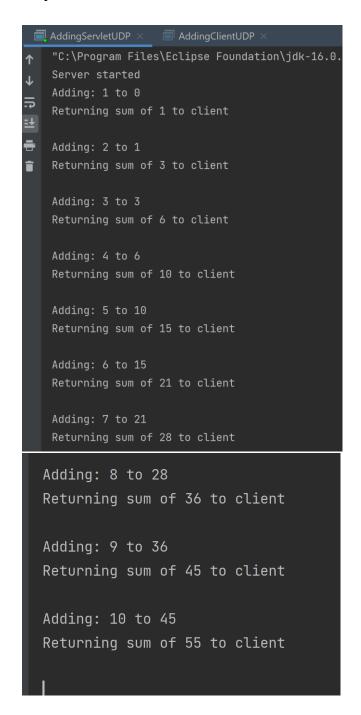
The server returned 10.

The server returned 15.

Process finished with exit code 130
```



8. Project2Task2CServerScreen



9. Project2Task3Client

```
BufferedReader typed = new BufferedReader (new
InputStreamReader(System.in));
                if ((userChoice = typed.readLine()) != null &&
```

```
value = Integer.parseInt(typed.readLine());
System.out.println(e.getMessage());
```

```
InetAddress aHost = InetAddress.getByName("localhost");
int serverPort = 6790;
   payload = id + " " + operation;
System.arraycopy(reply.getData(), 0, replyBytes, 0,
```

10. Project2Task3Server

```
byte[] buffer = new byte[1000];
buffer.length);
                aSocket.receive(request);
                System.arraycopy(request.getData(), 0, requestBytes, 0,
                String requestString = new String(requestBytes);
                byte [] replyBytes = String.valueOf(result).getBytes();
                DatagramPacket reply = new DatagramPacket(request.getData(),
            System.out.println("Socket: " + e.getMessage());
```

```
public static int performOperations(String [] requestItems) {
   String operation = requestItems[1];
    } else if(userIdSums.get(id) == null && !operation.equals("get")) {
        } else if(operation.equals("subtract")) {
```

11. Project2Task3ClientScreen

```
"C:\Program Files\Eclipse Foundation\jdk-16.0.2.7-hotspot\bin\java.exe
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum
4. Exit client
Enter value to add:
Enter your ID:
The result is 10.
2. Subtract a value from your sum.
4. Exit client
Enter value to subtract:
Enter your ID:
4. Exit client
The result is 8.
1. Add a value to your sum.
2. Subtract a value from your sum.
4. Exit client
1. Add a value to your sum.
2. Subtract a value from your sum.
4. Exit client
```

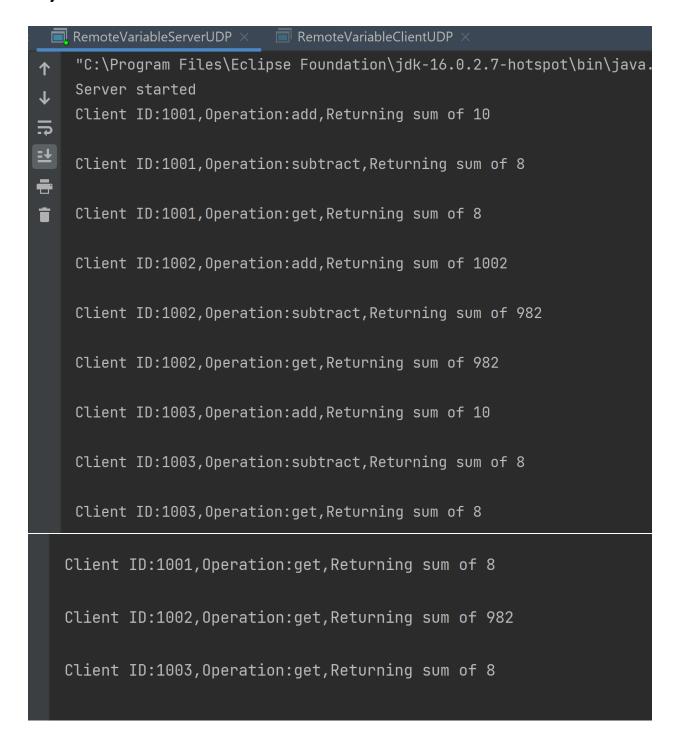
```
The result is 982.
🖶 1. Add a value to your sum.
1 2. Subtract a value from your sum.
   3. Get your sum
   4. Exit client
   The result is 982.
   1. Add a value to your sum.
   2. Subtract a value from your sum.
   Enter value to add:
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum
4. Exit client
Enter value to subtract:
Enter your ID:
The result is 8.
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum
4. Exit client
Enter your ID:
 The result is 8.
```

```
2. Subtract a value from your sum.
Client side quitting. The remote variable server is still running.
Process finished with exit code 0
↑ "C:\Program Files\Eclipse Foundation\jdk-16.0.2.7-hotspot\bin\java.e
   1. Add a value to your sum.
   2. Subtract a value from your sum.
₴

≅ Enter your ID:

    The result is 8.
    1. Add a value to your sum.
    2. Subtract a value from your sum.
    The result is 982.
    3. Get your sum
2. Subtract a value from your sum.
4. Exit client
Client side quitting. The remote variable server is still running.
Process finished with exit code 0
```

12. Project2Task3ServerScreen



13. Project2Task4Client

```
value = Integer.parseInt(typed.readLine());
System.out.println(e.getMessage());
```

```
public static int getResult(int id, int userChoice, int value) throws
        clientSocket = new Socket("localhost", serverPort);
        System.out.println("Socket: " + e.getMessage());
        if(clientSocket!=null) clientSocket.close();
private static String getOperation(int userChoice) {
```

```
// displays menu options to the user
public static void displayMenu() {
    String [] menu = {"Add a value to your sum.", "Subtract a value from
your sum.",

    "Get your sum", "Exit client"};
    for(int i =0; i<menu.length; i++) {
        System.out.printf("%d. %s%n",i+1,menu[i]);
    }
}

// New Exception created to track ID which is out of range
public static class IDOutOfRangeException extends Exception {
    public IDOutOfRangeException(String message) {
        super(message);
    }
}</pre>
```

14. Project2Task4Server

```
/*
 * @author: Shivani Poovaiah Ajjikutira
 * Last Modified: 9th October 2021
 *
 * This code follows Task 3 but uses TCP instead of UDP for the data
 * transmission between the client and server.
 * The following code is the server side for a program that returns a
 * number stored against a particular ID or adds/subtracts integers to
 * that integer. The sum/ difference is then stored against the ID.
 * The user id, operation and value(in case of add/subtract) are sent by
 * the client and output of the corresponding operation is returned to the
 * client. The server socket is initialized and continues to listen to any
 * request sent by client sockets connected to port number 7777. The server
 * receives the data from the client through Scanner "in" via the socket
 connection
 * formed. The PrintWriter "out" is used to write into the stream and send
 data
 * back to the requesting client. The performOperations method checks the
 operation
 * passed and does the required logic. A HashMap is used to store the integer
 * corresponding to each ID. In case of addition and subtraction the HashMap
 * values are updated, in case of get, the value for id as key is returned
 * as result to the client. The server is always running.
 * */

import java.io.BufferedWriter;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.io.PrintWriter;
import java.io.PrintWriter;
import java.aet.ServerSocket;
```

```
public class RemoteVariableServerTCP {
            int serverPort = 7777; // the server port number
                PrintWriter out;
OutputStreamWriter(clientSocket.getOutputStream())));
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
```

```
System.out.println("IO: " + e.getMessage());
public static int performOperations(String [] requestItems) {
   String operation = requestItems[1];
    if(userIdSums.get(id) == null && operation.equals("get")) {
        if(operation.equals("add")) {
```

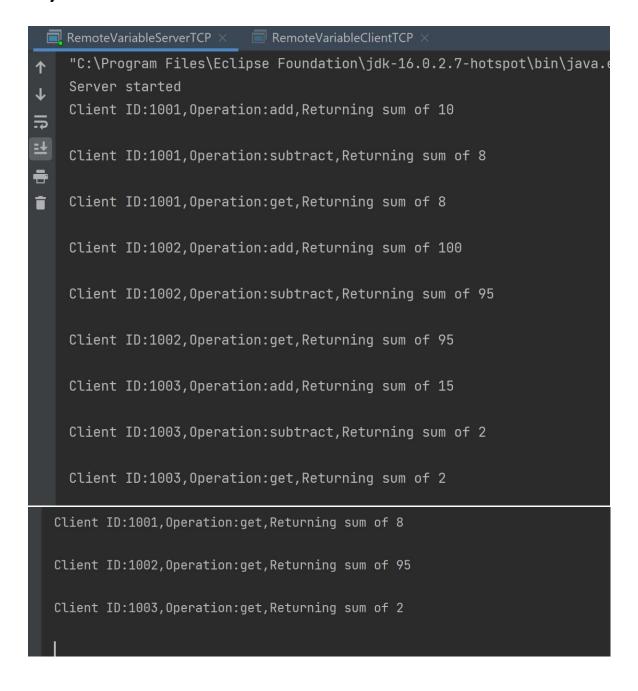
15. Project2Task4ClientScreen

```
"C:\Program Files\Eclipse Foundation\jdk-16.0.2.7-hot
1. Add a value to your sum.
2. Subtract a value from your sum.
Enter value to add:
The result is 10.
3. Get your sum
4. Exit client
Enter value to subtract:
The result is 8.
RemoteVariableServerTCP × 🔳 RemoteVariableClientTCP
1. Add a value to your sum.
2. Subtract a value from your sum.
The result is 8.
2. Subtract a value from your sum.
Enter value to add:
2. Subtract a value from your sum.
4. Exit client
```

```
Enter value to subtract:
The result is 95.
2. Subtract a value from your sum.
3. Get your sum
The result is 95.
1. Add a value to your sum.
2. Subtract a value from your sum.
Enter value to add:
The result is 15.
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum
4. Exit client
Enter value to subtract:
Enter your ID:
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
Enter your ID:
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
Client side quitting. The remote variable server is still running.
Process finished with exit code 0
RemoteVariableServerTCP × RemoteVariableClientTCP
  "C:\Program Files\Eclipse Foundation\jdk-16.0.2.7-hotspot\bin\java
  1. Add a value to your sum.
   2. Subtract a value from your sum.
   3. Get your sum
   4. Exit client
   Enter your ID:
   The result is 8.
   1. Add a value to your sum.
   2. Subtract a value from your sum.
   3. Get your sum
   4. Exit client
   Enter your ID:
   1. Add a value to your sum.
   2. Subtract a value from your sum.
   3. Get your sum
   4. Exit client
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
Client side quitting. The remote variable server is still running.
Process finished with exit code 0
```

16. Project2Task4ServerScreen



17. Project2Task5Client

```
BufferedReader typed = new BufferedReader(new
InputStreamReader(System.in));
            String id = generateID(keys);
                if ((userChoice = typed.readLine()) != null &&
```

```
value = Integer.parseInt(typed.readLine());
                            value = Integer.parseInt(typed.readLine());
Integer.parseInt(userChoice), value, keys);
                        System.out.println("The result is " + result + ".");
            System.out.println("IO: " + e.getMessage());
    private static String generateID(BigInteger [] keys) {
        BigInteger e = keys[0]; // n is the modulus for both the private and
       String pubKey = e.toString()+n.toString();
```

```
byte[] idBytes = new byte[20];
            idBytes[i] = hashedOutput[hashedOutput.length-(i)];
        return id.toString();
private static BigInteger[] generateRSAKeys() {
   d = e.modInverse(phi);
```

```
BigInteger[] keys) throws IOException {
            clientSocket = new Socket("localhost", serverPort);
InputStreamReader(clientSocket.getInputStream()));
OutputStreamWriter(clientSocket.getOutputStream())));
            String operation= getOperation(userChoice);
                tokens = id+" "+keys[0]+" "+keys[2]+" "+operation+" "+value;
```

```
out.flush();
       System.out.println("Socket: " + e.getMessage());
        if(clientSocket!=null) clientSocket.close();
   byte[] bytesOfMessage = tokens.getBytes(StandardCharsets.UTF 8);
   System.arraycopy(bigDigest, 0, messageDigest, 1, messageDigest.length
   BigInteger c = m.modPow(d, n);
    return c.toString();
private static String getOperation(int userChoice) {
```

18. Project2Task5Server

```
import java.io.BufferedWriter;
```

```
in = new Scanner(clientSocket.getInputStream());
                PrintWriter out;
                out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
                    String data = in.nextLine();
                    boolean rightSignature = checkSignature(requestItems);
```

```
System.out.printf("Signature verified: %s\n",
                    result = performOperations(requestItems);
                out.flush();
private static boolean checkSignature(String[] requestItems) throws
    StringBuilder tokens= new StringBuilder();
    for(int i=0; i<requestItems.length-1;i++) {</pre>
```

```
tokens.toString().trim().getBytes(StandardCharsets.UTF 8);
        byte[] messageToCheckDigest = md.digest(bytesOfMessageToCheck);
        System.arraycopy(messageToCheckDigest, 0, newMessage, 1,
        return bigIntegerToCheck.compareTo(decryptedHash) == 0;
            byte[] hashedPublicKey =
md.digest(pubKey.getBytes(StandardCharsets.UTF 8));
            byte[] hashedPubKey = new byte[20];
            for (int i = 1; i < hashedPubKey.length; i++) {</pre>
                hashedPubKey[i] = hashedPublicKey[hashedPublicKey.length -
            BigInteger computedId = new BigInteger(hashedPubKey);
            BigInteger bigId = new BigInteger(id);
```

```
if(bigId.equals(computedId)) return true;
        System.out.println("ID generation: "+ i.getMessage());
public static int performOperations(String [] requestItems) {
    String operation = requestItems[3];
        if(operation.equals("add")) {
```

20. Project2Task5ClientScreen

21. Project2Task5ServerScreen