LAB ASSESSMENT-2

COMPUTER NETWORKS

SWE-2002

Vaidya urmila suman

21MIS1098

(a) Client is sending a message to the server. The server encodes the message and returns to the client (Encoding is done by replacing the character by the ASCII value of the remainder using the formula (ASCII (chr) mod (nth prime)) n-value is sent by the client) Write the program to implement the above.

Aim:

To implement a client-server communication system where the client sends a message to the server. The server encodes the message using a formula based on the ASCII values and the nth prime number. The encoded message is then returned to the client.

Algorithm:

Client Program:

1. Create a Socket and connect it to the server's IP address and port.
2. Retrieve the input stream and output stream of the socket.
3. Send the message to the server by writing it to the output stream.
4. Send the n-value to the server by writing it to the output stream.
5. Read the encoded message from the input stream.
6. Print the encoded message received from the server.
7. Close the socket.

Server Program:

1. Create a ServerSocket and bind it to a specific port.
2. Start listening for incoming client connections using the accept() method of the

ServerSocket.

3. When a client connection is accepted:

• Retrieve the input stream and output stream of the socket.

• Read the message and n-value sent by the client from the input stream.

• Calculate the nth prime number using a helper method.

• Encode the message by iterating over each character, calculating its ASCII value,

and replacing it with the remainder of the ASCII value divided by the nth prime.

• Send the encoded message back to the client using the output stream.

• Close the client socket.

4. Continue listening for new client connections

Server.java

Code:-

```
import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.List;


public class server {

   public static void main(String[] args) {

      int port = 6666;

      try {

         ServerSocket serverSocket = new ServerSocket(port);

         System.out.println("Server started, waiting for client connection...");
```

```
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " + clientSocket.getInetAddress());

                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

                String message = in.readLine();
                int n = Integer.parseInt(in.readLine());
                String encodedMessage = encodeMessage(message, n);

                out.println(encodedMessage);
                clientSocket.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static int getNthPrime(int n) {
        List<Integer> primes = new ArrayList<>();
        int num = 2;

        while (primes.size() < n) {
            boolean isPrime = true;
```

```
        for (int prime : primes) {
            if (num % prime == 0) {
                isPrime = false;
                break;
            }
        }

        if (isPrime) {
            primes.add(num);
        }

        num++;
    }

    return primes.get(n - 1);
}

private static String encodeMessage(String message, int n) {
    int prime = getNthPrime(n);
    StringBuilder encodedMessage = new StringBuilder();

    for (char c : message.toCharArray()) {
        int asciiVal = (int) c;
        int encodedChar = asciiVal % prime;
        encodedMessage.append(encodedChar);
    }
```

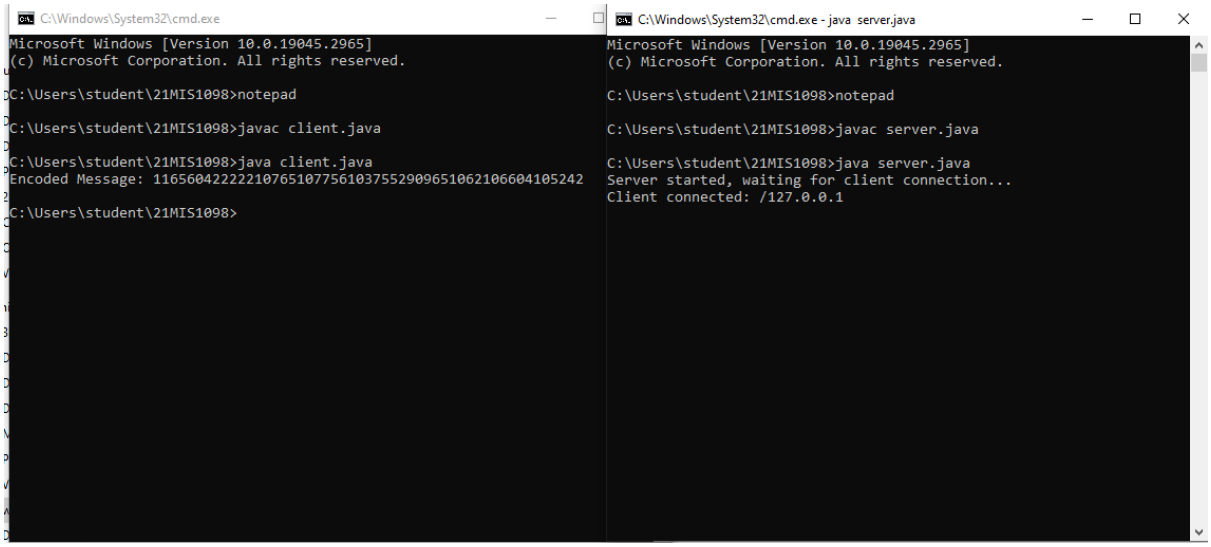
```
        return encodedMessage.toString();
    }
}
```

Client.java

Code

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class client {
    public static void main(String[] args) {
        String host = "localhost";
        int port = 6666;

        try {
            Socket socket = new Socket(host, port);
            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            String message = "Nothing..... Its just Pushpanath texting here";
            int n = 5;

            out.println(message);
            out.println(n);

            String encodedMessage = in.readLine();
```

```
            System.out.println("Encoded Message: " + encodedMessage);


            socket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



(b)Implement a TCP/IP socket-based ATM system. Server – to maintain the customer details (Name, Card_no., Pin, Balance). Client – when a customer wants to withdraw an amount, validate his login with pin and balance.

Aim:

To write a TCP/IP socket-based program to maintain customer details and when a customer wantsto withdraw an amount, validate his login with pin and balance

Algorithm:

Client Program

1. Create a Socket object with the IP address of the server and the port number.

2. Create input and output streams for reading user input and communicating with the server.

3. Prompt the user to enter their card number.

4. Send the card number to the server using the output stream.

5. Prompt the user to enter their PIN.

6. Send the PIN to the server using the output stream.

7. Read the response from the server using the input stream.

8. If the response is "welcome":

• Prompt the user to enter the withdrawal amount.

• Send the withdrawal amount to the server using the output stream.

• Read the withdrawal response from the server using the input stream.

• Print the withdrawal response.

9. Close the socket.

Server Program

1. Define customer details such as name, card number, PIN, and balance.

2. Define response messages for different scenarios, such as successful login, invalid credentials, successful withdrawal, and insufficient balance.

3. Create a ServerSocket object and bind it to a specific port.

4. Enter a continuous loop to accept client connections.

5. Accept a client connection and create input and output streams for communication with the client.

6. Read the card number from the client using the input stream.

7. Convert the card number to an integer.

8. Read the PIN from the client using the input stream.

9. Convert the PIN to an integer.

10. Check if the card number and PIN match the predefined customer details.

11. If the credentials are valid:

• Send the "welcome" response to the client using the output stream.

• Read the withdrawal amount from the client using the input stream.

• Convert the withdrawal amount to an integer.

• Check if the withdrawal amount is within the available balance.

• If the balance is sufficient:

• Deduct the withdrawal amount from the balance.

• Send the "withdraw successful" response to the client using the output stream.

• If the balance is insufficient:

• Send the "insufficient balance" response to the client using the output stream.

12. If the credentials are invalid:

• Send the "invalid credentials" response to the client using the output stream.

13. Continue to the next iteration to accept new client connections.

Server.java

Code

```
import java.io.*;

import java.util.*;

import java.net.*;


class server {

   public static void main(String[] args) throws Exception {

      String name = "Pushpanath M R - 21MIS1107";

      int cno = 7777;

      int pin = 1111;

      int bal = 700000;

      String x, y, z;

      int no, pin1, amt;

      String a1 = "welcome";

      String a2 = "Invalid Credentials";

      String a3 = "withdraw successful";

      String a4 = "insufficient balance";
```

```
    ServerSocket ss = new ServerSocket(6666);
    while (true) {
        Socket s = ss.accept();
        BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));
        DataOutputStream out = new DataOutputStream(s.getOutputStream());

        x = in.readLine();
        no = Integer.parseInt(x);
        y = in.readLine();
        pin1 = Integer.parseInt(y);

        if (cno == no && pin == pin1) {
            out.writeBytes(a1 + '\n');
            z = in.readLine();
            amt = Integer.parseInt(z);

            if (amt <= bal) {
                bal = bal - amt;
                out.writeBytes(a3 + '\n');
            } else {
                out.writeBytes(a4 + '\n');
            }
        } else {
            out.writeBytes(a2 + '\n');
        }
    }
}
```

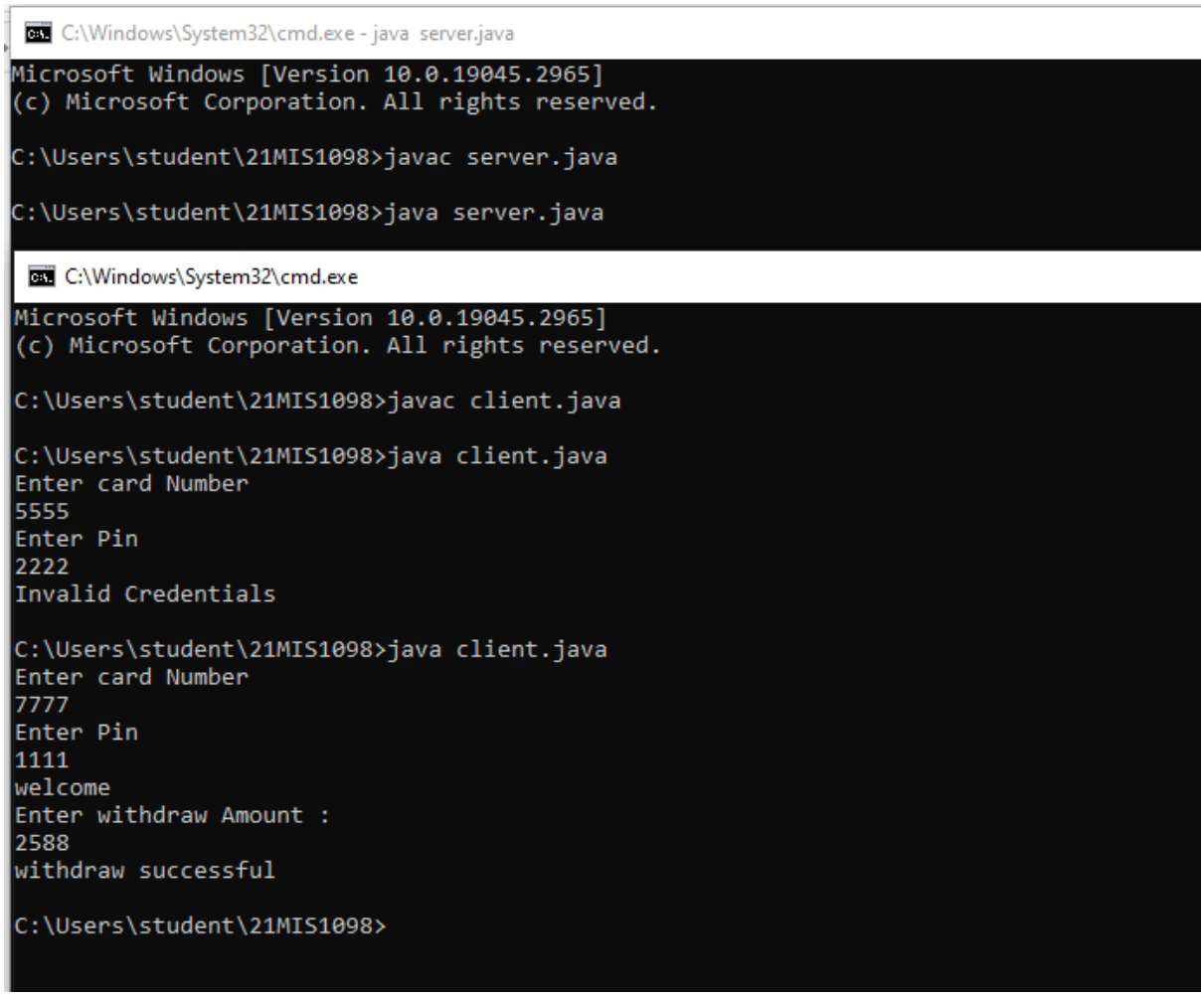
```
}
```

Client.java

Code

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

class client {
    public static void main(String[] args) throws Exception {
        String number, pin, r1, amt, r2;
        Socket s = new Socket("localhost", 6666);
        BufferedReader inc = new BufferedReader(new InputStreamReader(System.in));
        DataOutputStream out = new DataOutputStream(s.getOutputStream());
        BufferedReader ins = new BufferedReader(new InputStreamReader(s.getInputStream()));

        System.out.println("Enter card Number");
        number = inc.readLine();
        out.writeBytes(number + '\n');

        System.out.println("Enter Pin");
        pin = inc.readLine();
        out.writeBytes(pin + '\n');

        r1 = ins.readLine();
        System.out.println(r1);

        if (r1.equals("welcome")) {
```

```
        System.out.println("Enter withdraw Amount :");
        amt = inc.readLine();
        out.writeBytes(amt + '\n');

        r2 = ins.readLine();
        System.out.println(r2);
    }
  }
}
```



(c) In an IPV4 packet the value of header length is 1000 in binary. Write a code to find how many bytes of options are being carried by this packet.

Aim:

To Write a IPV4 Packet program to find the value of header length 1000, then find the

numberof bytes of options are being carried by this packet.

Algorithm:

1. Read the header length value in binary.

2. Convert the binary value to an integer.

3. Calculate the actual header length in 32-bit words by multiplying the header

length valueby 4.

4. Subtract 20 from the actual header length to determine the options length.

5. Print the options length in bytes.

Code

```
public class exp3 {
   public static void main(String[] args) { String headerLengthBinary = "1000";
   int headerLength = Integer.parseInt(headerLengthBinary, 2); int optionsLength = (headerLength - 5) * 4;
   System.out.println("Options length: " + optionsLength + " bytes");
   }
   }
```