TITLE: SECURE SOCKET PROGRAMMING

OBJECTIVES:

→ To understand and implement a secure client-side socket connection to a remote server using SSL/TLS protocol.

→ To use the SSLSocketFactory class to generate SSLSocket and exchange data over port 443.

THEORY:

Secure Socket (SSL/TLS)

→ Secure socket is an extension of the standard TCP socket that adds a layer of security using cryptographic protocols known as SSL (Secure Sockets Layer) or TLS (Transport Layer Security).

→ "Handsake" occurs before data exchange.

SSLSocketFactory

→ It is an abstract class (javax.net.ssl) that acts as a factory for creating SSLSocket objects.

Methods:

getDefault() → returns the default SSL Socket factory.

createSocket() → returns the socket connected to specific port and host (auto handling SSL layering over TCP connection).

SOURCE CODE:

```
package lab6;
import java.io.*;
import javax.net.ssl.*;
public class Solution{
    public static void main (String[] args){
        String hostname = "www.google.com";
        int port = 443;
        try{
            System.out.println ("starting SSL Handshake");
            SSLFactory
            SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory
                                                .getDefault();
            SSLSocket socket = factory.createSocket (hostname, port);
            socket.startHandshake();
            System.out.println ("Secure connection established with:"
                                        + hostname);
            System.out.println ("Cipher suite:" + socket.getSession().
                                                getCipherSuite());
            System.out.println ("Protocol:" + socket.getSession().
                                                getProtocol());
            PrintWriter out = new PrintWriter (new OutputStreamWriter
                                        (socket.getOutputStream()));
            out.println ("GET / HTTP/1.1");
            out.println ("Host:" + hostname);
```

```
            out.println(" ");
            out.flush();

            BufferedReader in = new BufferedReader(
                    new InputStreamReader(socket.getInputStream()));

            String line;
            int count = 0;
            System.out.println("Server Response");
            while ((line = in.readLine()) != null && count < 5){
                System.out.println(line);
                count++;
            }
            in.close();
            out.close();
            socket.close();
        } catch (IOException e){
            System.out.println(e.getMessage());
        }
    }
}
```

Output

CONCLUSION:

→ In this lab, we successfully created a program that establishes a secure connection with google.com at on port 443. We also demonstrated the use of SSLSocketFactory.getDefault(). We exchanged application data over the encrypted channel.