TITOLO: **Realizzazione di una calcolatrice con l'utilizzo dei Socket**
MATERIE: **Sistemi e reti, Telecomunicazioni**
OBIETTIVI: **Realizzare una calcolatrice remota**

■ ESPOSIZIONE DEL PROBLEMA DI ANALISI/PROGETTAZIONE

In questo elaborato si vuole analizzare come sia possibile eseguire operazioni su un server remoto con l'obiettivo di non gravare il client.
A tal fine si vuole implementare una calcolatrice che utilizzi a livello di trasporto i servizi offerti dal protocollo TCP e che, pertanto, non impieghi risorse del client. L'architettura deve essere client-server, e il server dovrà accettare le richieste provenienti dal client utilizzando un protocollo che preveda i seguenti comandi per effettuare le operazioni indicate:

Comando	Operazione
ADD,X,Y	Addizione: (X+Y)
SUB,X,Y	Sottrazione (X-Y)
MUL,X,Y	Moltiplicazione (X * Y)
DIV,X,Y	Divisione (X : Y)
POW,X,Y	Potenza (X^Y)

■ SVILUPPO DELLA SOLUZIONE CON SPIEGAZIONI E REALIZZAZIONE DEL CODICE SORGENTE

■ Spiegazione di dettaglio dello scenario che si vuole gestire.

Normalmente, un server in esecuzione ha un socket legato ad una specifica porta. Il server aspetta e ascolta eventuali connessioni provenienti dai client. Il client conosce l'hostname della macchina su cui il server è in esecuzione e il numero della porta sul quale il server è in ascolto.
I socket sono associati a un InputStream e a un OutputStream e quindi per scrivere e leggere si usano le normali primitive previste per gli stream.
La traduzione in linguaggio Java si esplicita in quanto segue:

■ Soluzione in Java.

CODICE SORGENTE LATO SERVER

```
package socket;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.ServerSocket;

import java.net.Socket;

public class Server {

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

        float num1, num2;

        char operazione;
```

```
Operazioni operazioni;
ServerSocket server = new ServerSocket(1235);
while(true) {
        Socket client = server.accept();
        BufferedReader input =
        new BufferedReader(new InputStreamReader(client.getInputStream()));
        DataOutputStream output = new DataOutputStream(client.getOutputStream());
        String[] nums = input.readLine().split(" ");
        num1 = Float.parseFloat(nums[0]);
        num2 = Float.parseFloat(nums[2]);
        operazione = nums[1].charAt(0);
        operazioni = new Operazioni(num1, num2);
        switch(operazione) {
                case '+' : {  //addizione
                        output.writeBytes(String.valueOf(operazioni.addizione()) +"\n");
                        break;
                        }
                case '-' : {  //sottrazione
                        output.writeBytes(String.valueOf(operazioni.sottrazione()) + "\n");
                        break;
                        }
                case '/' : {  //divisione
                        output.writeBytes(String.valueOf(operazioni.divisione()) + "\n");
                        break;
                        }
                case '*' : { //moltiplicazione
                        output.writeBytes(String.valueOf(operazioni.moltiplicazione()) + "\n");
                        break;
                        }
```

```
                                        case '^' : { //potenza

                                                    break;

                                                    }

                                        default : { //casi non previsti

                                                    output.writeBytes("Errore" + "\n");

                                                    break;

                                                    }

                                        }

                            }



                }

}
```

CODICE SORGENTE LATO CLIENT

```
package socket;

import java.io.*;

import java.net.*;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.util.Scanner;



public class Client {

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

                            String operazione;

                            Scanner scanner = new Scanner(System.in);

                            Socket client = new Socket("localhost", 1235);
```

```
                DataOutputStream output = new DataOutputStream(client.getOutputStream());
                BufferedReader input = new BufferedReader(new InputStreamReader(client.getInputStream()));
                System.out.println("Inserisci operazione usando gli spazi tra numero/operatore (Esempio: 1 + 1): ");
                operazione = scanner.nextLine();
                output.writeBytes(operazione + "\n");
                System.out.println("Risultato: " + input.readLine());
                scanner.close();
                client.close();
        }
}
```

CODICE SORGENTE PER LA GESTIONE DELLE OPERAZIONI

```
package socket;

import java.lang.Math;

public class Operazioni {

        public float num1, num2;
        Operazioni(float num1, float num2) {
                this.num1 = num1;
                this.num2 = num2;
        }
        public float addizione() {
                float somma;
                somma = this.num1 + this.num2;
                return somma;
        }


        public float sottrazione() {
                float differenza;
                differenza = this.num1 - this.num2;
```

```
                return differenza;
        }

        public float moltiplicazione() {
                float prodotto;
                prodotto = this.num1 * this.num2;
                return prodotto;
        }
        public float divisione() {
                float quoziente;

                quoziente = this.num1 / this.num2;
                return quoziente;
        }
        public float potenza() {
                float elevato_a_potenza;
                elevato_a_potenza = Math.pow(this.num1,  this.num2);
                return elevato_a_potenza;
        }
}
```

Suggerimenti di possibili percorsi alternativi per nuovi elaborati

- Implementare altre operazioni matematiche più complesse.