```java
import java.util.Scanner;

class MatrixChainMultiplication {

    // Driver code
    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);

        int count = 0;

        while (count < 5) {
            System.out.println("Enter the number of matrices you wish to multiply:");
            int number_of_matrices = input.nextInt();

            // Creating an array of dimensions to be entered which will be always 1 greater than the number of matrices
            // Refer to the example
            int[] dimensions = new int[number_of_matrices + 1];

            // Entering the dimensions
            // If there are 3 matrices with dimensions 10 X 20, 20 X 30, 30 X 15
            // The dimension will be entered as 10, 20, 30, 15 - Hence there are 1 more dimensions than there are matrices

            int total_scalar_multiplications = 1;
            for (int i = 0; i <= number_of_matrices; i++) {
                System.out.print("Please enter dimension " + i + " : ");
                dimensions[i] = input.nextInt();
                total_scalar_multiplications = total_scalar_multiplications * dimensions[i];
            }


            // Initialize the cost matrix (M)
            int[][] m = new int[number_of_matrices + 1][number_of_matrices + 1];

            // Initialize the parenthesis matrix (S)
            int[][] s = new int[number_of_matrices + 1][number_of_matrices + 1];

            for (int i = 0; i < number_of_matrices; i++) {
                m[i][i] = 0;
            }

            int total_number_of_multiplications, min;

            // Here dimension refers to the max number of outputs that need to be calculated in each row
            // For example for dimension = 1 i.e. row 1 and number of matrices = 4 the number of outputs = 3
            for (int dimension = 1; dimension < number_of_matrices; dimension++) {

                for (int i = 1; i < number_of_matrices + 1 - dimension; i++) {

                    int j = i + dimension;

                    min = Integer.MAX_VALUE;

                    // Here 'k' refers to the k in Matrix mutiplication formula
                    for (int k = i; k <= j - 1; k++) {

                        total_number_of_multiplications = m[i][k] + m[k+1][j] + dimensions[i-1] * dimensions[k] * dimensions[j];

                        if (total_number_of_multiplications < min) {
                            min = total_number_of_multiplications;
                            s[i][j] = k;
                        }
                    }
                    m[i][j] = min;
                }
            }

            System.out.println("\n\n\tM Matrix\n");
            for(int i = 1; i <= number_of_matrices; i++ ) {
                for(int j = 1; j <= number_of_matrices; j++ ){
                    if(i <= j) {
                        System.out.print( m[ i ][ j ]+ "\t" );
                    }
                    else {
                        System.out.print( "\t" );
                    }
                }
                System.out.println("\n");
            }

            System.out.println("\tS Matrix\n");
                    for(int i = 1; i <= number_of_matrices; i++ ) {
                for(int j = 1; j <= number_of_matrices; j++ ) {
                    if( i<=j ) {
                        System.out.print( s[ i ][ j ]+ "\t" );
                    }
                                        else {
                        System.out.print( "\t" );
                    }
                        }
                            System.out.println("\n");
                    }

            System.out.println("\n\nMinimum number of matrix multiplication using dynamic programming: " + m[1][number_of_matrices]);
            System.out.println("\nTotal number of scalar multiplication without dynamic programming: " + total_scalar_multiplications);

            count++;
        }
    }
}

```