# CLEAN CODE

## Ejemplo1

En el primer ejemplo de codigo refleja los algunos conceptos básicos de clean codes, particularmente los bloques de nombre y comentarios.

```java
        int min = 0;
        int max = Integer.MAX_VALUE;
        int numGrande;
        int numPequeño;
        int resto;
        int maximoComunDivisor = 0;
        int minimoComunMultiplo = 0;
        Scanner teclado = new Scanner(source: System.in);
        String entrada;
        //Pedimos los numeros por teclado y realizamos comprobaciones de contorno
        do{
            System.out.println(x: "Introduzca un numero positivo: ");
            entrada = teclado.nextLine();
            numGrande = Integer.parseInt(s: entrada);
        }while(!(numGrande > min && numGrande <= max ));
        do{
            System.out.println(x: "Introduzca un numero menor que el anterior: ");
            entrada = teclado.nextLine();
            numPequeño = Integer.parseInt(s: entrada);
        }while(!( numPequeño > min && numPequeño <= numGrande));
        //Ya hemos pedido los numeros y ahora vamos a calcular el MCM y el MCD, por eso creamos otras 2 variables
        int dividendo = numGrande;
        int divisor = numPequeño;
        do{
            resto = dividendo%divisor;
            if(resto == 0){
                maximoComunDivisor = divisor;//Si el resto de la division es 0, el divisor es el MCD
                System.out.println("El maximo comun divisor de " + numGrande + " y " + numPequeño + " es: " + maximoComunDivisor);
                break;
            }
            else{
                dividendo = divisor;
                divisor = resto;
            }
        }while( resto != 0);
        minimoComunMultiplo = (numGrande*numPequeño)/maximoComunDivisor;
        System.out.println("El minimo comun multiplo de " + numGrande + "y " + numPequeño + " es: " + minimoComunMultiplo);
```

Podemos observar que los nombres de todas las variables del codigo son faciles de pronunciar y buscar, ademas de tener un significado. Lo que nos permite añadir solamente 3 comentarios concisos, en partes del codigo que podrian llevar a confusion, pero sin abusar de ellos ya que tenemos un codigo autoexplicativo. Esto a su vez hace que no tengamos comentarios desfasados.

## Ejemplo2

Este ejemplo muestra los conceptos clave para hacer un código más limpio cuando trabajamos con metodos.

```java
public class Punto2D {
    private double x;
    private double y;
    private static int contador = 0;// contador total de puntos
    private static final String NOMBRE = "Punto";//Utilizamos para ahorrar memoria
    private int id;

    public double calcularDistancia( Punto2D param ){
        double distancia_calculada =
                Math.sqrt(
                        Math.pow((param.x-x) ,b: 2) +
                                Math.pow((param.y-y) ,b: 2) );
        return distancia_calculada;
    }
    public static double calcularAreaHeron( Punto2D param1, Punto2D param2, Punto2D param3 ){
        double a = param1.calcularDistancia(param: param2);
        double b = param1.calcularDistancia(param: param3);
        double c = param2.calcularDistancia(param: param3);
        double semiPerimetro = (a+b+c)/2;
        double area = Math.sqrt(semiPerimetro*(semiPerimetro - a)*(semiPerimetro - b)*(semiPerimetro - c));
        return area;
    }
```

Los dos metodos que observamos son cortos, sus nombres son verbos, estan enfocados en una unica tarea, tienen pocos argumentos y no genera efectos colaterales.