MODULARIDAD EN JAVA

1. Modularidad

En este apartado realizaremos una primera aproximación a la modularidad dentro de una clase en Java realizando descomposición funcional. La siguiente aproximación la realizaremos en la siguiente sesión descomponiendo la aplicación en clases.

1.1 Ejemplo 1

El siguiente ejemplo muestra una clase con un método auxiliar que permite escribir el mensaje "Eco…" tantas veces como le indiquemos.

```
import java.util.Scanner;

public class Eco {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Introduce un número:");
        int n = entrada.nextInt();
        eco(n); //llamada a la función
    }

    public static void eco(int veces) {
        for (int i = 0; i < veces; i++) {
            System.out.println("Eco...");
        }
    }
}
```

1.2 Ejemplo 2

El siguiente ejemplo muestra el paso de parámetros por valor y por referencia.

```
//Paso de parámetros por valor y por referencia
public class Parametros {
      static String global= "global";     //variable global a la clase

      public static void main(String args[]) {
            int num = 1;                  //por valor
            int numeros[] = {1,2,3,4,5};  //los arrays se pasan por referencia
            String cadena = "abc";        //por valor

            System.out.println("\nNúmero antes de la llamada al método:"+num);
            System.out.println("\nArray antes de la llamada al método:");
            for (int i=0; i<5; i++) {
                  System.out.print(" "+numeros[i]);
            }
            System.out.println("\nCadena antes de la llamada al método:"+cadena);
            System.out.println("\nGlobal antes de la llamada al método:"+global);

            //llamamos al método
            convierte(num, numeros, cadena);

            System.out.println("\nNúmero después de la llamada al método:"+num);
            System.out.println("\nArray después de la llamada al método:");
            for (int i=0; i<5; i++) {
                  System.out.print(" "+numeros[i]);
            }
```

```
            System.out.println("\nCadena después de la llamada al método:"+cadena);

            System.out.println("\nGlobal después de la llamada al método:"+global);
      }

      public static void convierte(int num, int numeros[], String cadena) {
            num += 1;
            //sumamos 1 al array de enteros
            for (int i=0; i<5; i++) {
                  numeros[i] += 1;
            }
            cadena = "def";
            global = "nueva";
      }
}
```

1.3 Ejemplo 3

En el siguiente ejemplo calculamos la potencia de un número. Para realizarlo nos basamos en la clase Math que define múltiples métodos matemáticos.

```
import java.io.*;

public class Potencia {
      public static void main(String args[]) throws IOException  {
            BufferedReader entrada=
                  new BufferedReader(new InputStreamReader(System.in));

            double base = 0, exponente = 0, resultado;

            System.out.println("\nIntroduce la base:");
            base = Double.parseDouble(entrada.readLine());

            System.out.println("\nIntroduce el exponente:");
            exponente = Double.parseDouble(entrada.readLine());

            //llamamos al método
            resultado = calcula(base, exponente);
            System.out.println("\n\nEl resultado es: "+resultado);
      }

      public static double calcula(double num1, double num2) {
            //Calculamos con el método pow de la clase Math
            return Math.pow(num1, num2);
      }
}
```

1.4 Ejercicios modularidad

1. Realizar un programa que imprima todos los números de un rango de valores. El programa pedirá dos números (el menor y el mayor) y llamará a un método auxiliar encargado de imprimir todos los que se encuentren en su rango.
2. Reutilizando el programa anterior escribir solo los números pares en el rango de valores. El método auxiliar encargado de imprimir todos los valores que se encuentren en su rango tiene que hacer uso de otra función llamada esPar(int num) que devolverá un booleano indicando si el número es par o no.
3. Realizar un programa que determine cuál es el mayor de dos números. El programa tendrá un método mayor que recibirá dos valores de tipo entero y devolverá cual es el mayor. Desde el método main se pedirá al usuario los valores y se llamará al método mayor.
4. Repetir el ejercicio anterior con una versión que devuelva el mayor de tres números pero reutilizando el método mayor de dos números.
5. Realizar un programa que permita simular una calculadora. El programa tendrá un método calculadora que recibirá dos valores de tipo double y un carácter que indicará la operación a realizar (+,-,*,/). Desde el método main se pedirá al usuario los valores y la operación y se llamará al método calculadora.