PROGRAMACIÓN

3ª TUTORÍA COLECTIVA

PROGRAMACIÓN MODULAR

* El resultado final de la programación modular es la **división** de un programa en un conjunto de módulos independientes que se comunican entre sí a través de llamadas.

* Un módulo representa **una tarea** determinada, identificada por un nombre (nombre del módulo) por el cual será invocada desde otros módulos.

* Siempre debe haber un módulo que describa la solución completa y que servirá de nexo de unión entre los demás, es el **programa principal**.

* Los módulos se **comunican** mediante variables de enlace a las que se denomina **parámetros** o argumentos.

* Los módulos se llaman funciones si devuelven algún argumento y procedimientos si no devuelven ningún valor. En Java hablamos de métodos.

VENTAJAS DE LA PROGRAMACIÓN MODULAR

* Los módulos se pueden desarrollar de manera independiente y paralélela por diferentes equipos, sin conocer casi unos de otros. Favorece el trabajo en equipo.
* Menor tiempo para la desarrollo de una aplicación.
* Los módulos, al ser independientes, son fácilmente modificables y/o sustituibles siempre y cuando se mantenga su interfaz con el resto, luego facilita la depuración y las pruebas, al poder hacerlo por separado cada módulo, y por tanto también facilita el mantenimiento.
* El cambio en uno de los módulos no afecta al conjunto de los demás.
* Simplifica el diseño. Disminuye la complejidad de los algoritmos y facilita su comprensión.
* Disminuye el tamaño total del programa.
* Ahorra en tiempo de **programación** porque promueve la reusabilidad del código.
* Al permitir la reutilización del código, disminuyen los costes.

VARIABLES GLOBALES, LOCALES Y DE BLOQUE

* Variables globales son aquellas que se definen para todas las funciones, y por tanto pueden ser utilizadas desde cualquiera de ellas. Aunque esto parezca una ventaja, en la práctica producen errores por ser manejadas y modificadas desde varias funciones. Un buen estilo de programación incluye una **mínima** utilización de este tipo de variables.
* Variables locales son aquellas que se definen dentro de una función y por tanto solamente pueden ser utilizadas dentro de la misma, siendo “invisibles” para el resto de las funciones. Un buen estilo de programación incluye la utilización prioritaria de este tipo de variables.
* Variables de bloque son aquellas que se definen dentro de un bloque de código, entre { y } , y por tanto solamente pueden ser utilizadas en el mismo.

PROGRAMACIÓN MODULAR con PSeInt

num es el parámetro

res1 es el valor que devuelve

```
1  // funcion que no recibe ni devuelve nada
2  SubProceso Saludar()
3    Escribir "Hola mundo!"
4  FinSubProceso
5
6  // funcion que recibe un parámetro y devuelve su doble
7  SubProceso res1 <- CalcularDoble(num)
8    res1 <- num*2 // retorna el doble
9  FinSubProceso
10
11 // funcion que recibe un parámetro y devuelve su triple
12 SubProceso res2 <-Triplicar(num)
13   res2 <- num*3
14 FinSubProceso
15
16 // proceso principal, que invoca a las funciones antes declaradas
17 Proceso PruebaFunciones
18   Escribir "Llamada a la funcion Saludar:"
19   Saludar() // como no recibe argumentos pueden omitirse los paréntesis vacios
20   Escribir "Introduce un valor numérico para x:"
21   Leer x
22   Escribir "Llamada a la función CalcularDoble"
23   Escribir "El doble de ",x," es ", CalcularDoble(x)
24   Escribir "Llamada a la función Triplicar"
25   Escribir "El tripleY de ",x," es ", Triplicar(x)
26 FinProceso
```

PROGRAMACIÓN MODULAR con Java

void: no devuelve nada

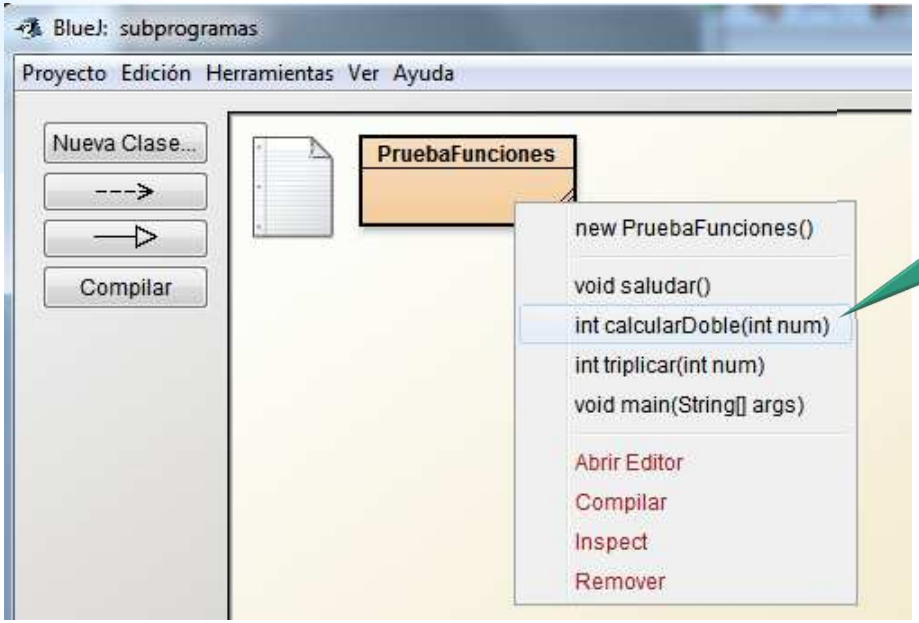
int: devuelve un int

```java
import java.util.Scanner;

class PruebaFunciones
{
  // función que no recibe ni devuelve nada
  static void saludar()
  {
    System.out.println( "Hola mundo!");
  }
  // función que recibe un parámetro (num) y devuelve su doble
  static int calcularDoble(int num)
  {
    return num*2;   // devuelve el doble de num
  }
  // funcion que recibe un parámetro y devuelve su triple
  static int triplicar(int num)
  {
    return num*3;   // devuelve el triple de num
  }
  // proceso principal, que invoca a las funciones antes declaradas
 public static void main(String [] args)
 {
    Scanner teclado = new Scanner(System.in);
    System.out.println("Llamada a la funcion Saludar");
    saludar();
    System.out.println("Introduce un valor numérico para x");
    int x=teclado.nextInt();
    System.out.println( "Llamada a la función CalcularDoble");
    System.out.println( "El doble de "+x+" es "+ calcularDoble(x));
    System.out.println( "Llamada a la función Triplicar");
    System.out.println( "El triple de "+x+" es "+ triplicar(x));
 }
}
```

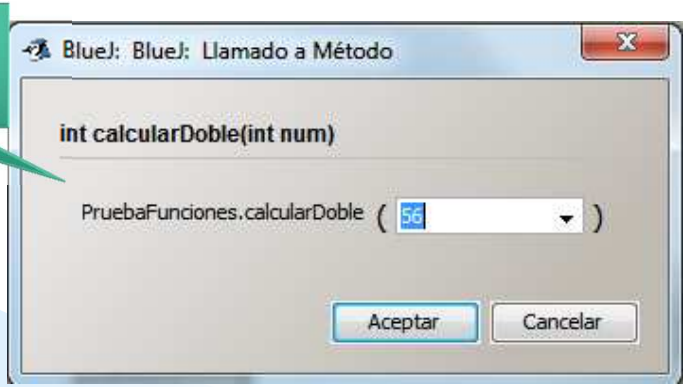
PROGRAMACIÓN MODULAR con Java

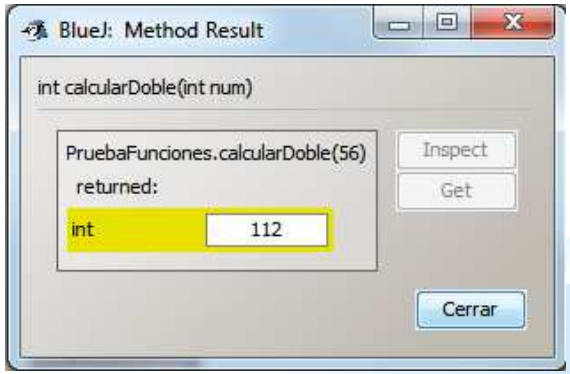
Podemos ejecutar un método con BlueJ independientemente:



Elegimos el método que queremos ejecutar

Introducimos los valores de los parámetros





MÓDULOS RECURSIVOS

* Un módulo recursivo es aquel que **se llama a sí mismo.**

* En su estructura incluye:
  * Un caso base que permite la finalización del problema.
  * Casos recursivos que hacen que la función vuelva a ser ejecutada.

Ejemplo: El factorial de un número N es:

N! = N * (N-1)!

1! = 1

2! = 2 * 1!

3! = 3 * 2!

…

```
1   SubProceso fac <- factorial(n)
2       Si n>1 Entonces
3           fac <- n* factorial(n-1)
4       Sino
5           fac <- 1
6       Fin Si
7   Fin SubProceso
8
9   Proceso Principal
10      Escribir "Introduce número: "
11      Leer num
12      res<-factorial(num)
13      Escribir "Resultado: ", res
14  FinProceso
```

MÓDULOS RECURSIVOS

```java
import java.util.Scanner;

public class FactorialRecursiva
{
  static int factorial(int n)
  {
    if (n>1)
      return n*factorial(n-1);
    else
      return 1;
  }

  public static void main(String[] args)
  {
    int numero;
    Scanner teclado = new Scanner(System.in);
    System.out.print("Introduce un número: ");
    numero=teclado.nextInt();
    System.out.println("El factorial de "+numero+" es "+factorial(numero));
  }
}
```