UF4.NF1.p6.Funciones

Una función es un fragmento de código etiquetado con un nombre.

La función se puede definir antes o después del main().

Ejemplo:

```
public class Main{
  public static void main(String[] args) {
    System.out.println("suma= "+suma(3,5));
    System.out.println("suma= "+suma(2,0.5,3,0.75));
  }
  static int suma (int a, int b){
    int suma;
    suma=a + b;
    return(suma);
  }
    // funcion sobrecargada
  static double suma (int a, double pesoA, int b, double pesoB){
    double suma=a*pesoA+b*pesoB;
    return(suma);
  }

}
```

En el lenguaje C++ hay tres formas de pasar datos a una función: por valor, por dirección y por referencia.

En el lenguaje Java, solamente se pasa por valor, lo que significa que se efectúa una copia local de las entidades que se están pasando. Vamos a explicar mediante ejemplos, el significado de la expresión pasar por valor:

Vamos a ver en este ejemplo el significado de "paso por valor". La variable a toma el valor inicial de 3. Cuando se llama a la función se pasa el valor de a en su único argumento, el valor de a se copia en el parámetro x, la variable x toma el valor de 3. En el curso de la llamada a la función, el valor de x cambia a 5, pero cuando la función retorna, la variable x ha dejado de existir. La variable a no se ha modificado en el curso de la llamada a la función, y sigue valiendo 3.

Durante el curso de la llamada a la función funcion, existe la variable a y su copia x, pero son dos variables distintas, aunque inicialmente guarden el mismo valor,

Por ahora, definiremos las funciones static. Habitualmente, los nombres siguen el estilo Camel:los nombres comienzan en minúscula, distinguiendo en los nombres compuestos cada palabra mediante una mayúscula inicial, lo que recuerda las jorobas de un camello, algunos ejemplos:

```
suma(), treSaludos, calculaRaizCuadrada(), muestraTodosDatosClientes()
```

Sobrecarga de funciones.

Java permite tener dos o más funciones con el mismo nombre en el mismo programa. Esto se conoce como **sobrecarga.** La forma de distinguir entre funciones sobrecargadas es mediante su lista de parámetros, que deben ser distintas ya sean en número o en tipo.

Ejemplo: Queremos diseñar una función para calcular la suma de dos enteros, pero también es útil hacer una suma ponderada, donde cada sumando tiene un peso distinto.

```
// funcion sobrecargada
 public static void main(String[] args) {
        System.out.println("suma= "+suma(3,5));
        System.out.println("suma= "+suma(2,0.5,3,0.75));

        static int suma (int a, int b){
          int suma;
          suma=a + b;
          return(suma);
        }
       // funcion sobrecargada
        static double suma (int a, double pesoA, int b, double pesoB){
          int suma=a*pesoA+b*pesoB;
          retunr(suma);
        }
   }
```

Si llamamos a la función suma de la forma suma(2,3) se ejecuta la primera versión y devolverá 5. En cambio, si se llama con suma(2,0.5,3,0.75), se ejecutará la segunda versión y devolverá 3.25.

Es muy común encontrar en la API funciones (métodos) sobrecargadas. Por ejemplo System.out.println, se encuentra sobrecargada para poder mostrar en pantalla cualquier tipo de dato.

Sobrecarga de funciones.

Se trata de los *varargs* introducidos en Java 5.

Primero que nada, la forma de declarar un método con argumentos variables es muy parecida a cualquier otra declaración. method(String… args){…}

Nótese que hay tres puntos después del tipo de variable, es lo que caracteriza a los métodos varargs.

Estos métodos son muy parecidos a los que usan arreglos pero tienen dos diferencias básicas que es bueno conocer.

//Primero lo primero, declaramos las variables que se usarán en el ejemplo.

```
int[] num={4,7,8,2};
int cuatro=4,dos=2;
//Y los métodos también. Nótese la diferencia entre ambos.

int suma_a (int... numero){
    int resultado = 0;
    for(int i = 0; i < numero.length; i++){
        resultado += numero[i];
    }return resultado;
}
int suma_b (int[] numero){
    int resultado = 0;
    for(int i = 0; i < numero.length; i++){
        resultado += numero[i];
    }return resultado;
}
```

PRIMERA DIFERENCIA: La forma en que se llama al método.

```
//Llamadas válidas:
System.out.println(suma_a(num));
System.out.println(suma_a(num[0],num[1],num[2],num[3]));
System.out.println(suma_a(cuatro,7,8,dos));

System.out.println(suma_b(num));
```

```
//Llamadas NO válidas:
System.out.println(suma_b(num[0],num[1],num[2],num[3]));
System.out.println(suma_b(cuatro,7,8,dos));
```

SEGUNDA DIFERENCIA: En los métodos "varargs", el argumento variable debe ser siempre el último argumento mientras que los métodos con arreglos no es necesario.

```
//Declaraciones válidas:
int suma_a(String cadena, int… numero){…}
int suma_b(String cadena, int[] numero){…}
int suma_b(int[] numero, String cadena){…}
//Declaración NO válida:
int suma_a(int… numero, String cadena){…}
```

Ejercicio 1: Función a la que se le pasan tres enteros y que calcule el máximo de los tres.

Ejercicio 2: Función a la que se le pasan dos enteros y muestra todos los números comprendidos entre ellos, inclusive ellos.

Ejercicio 3: Escribir una función que decida si dos enteros son amigos. Dos número son amigos si la suma de sus divisores propios(distintos de ellos mismo) son iguales.

Ejercicio 4: Hacer un programa que defina 3 tipo de arrays. Uno de enteros, otro de double y otro de carácteres. Escribir el método imprimirLista(). Habrá tres métodos imprimirLista() y la forma de distinguirlos es por el tipo del parámetro que se le pasa: int[], double[] o character[].