



Algoritmos y Estructura de Datos

FUNCIONES



DOCENTE: MARX DANLY LEÓN TRUJILLO
FACULTAD DE ING. INDUSTRIAL Y SISTEMAS
E.P. INGENIERÍA DE SISTEMAS

FUNCIONES

Las funciones son un conjunto de líneas de código (instrucciones), encapsulados en un bloque, usualmente **reciben parámetros**, cuyos valores utilizan para efectuar operaciones y adicionalmente retornan un valor con la ejecución **return**.

La visibilidad de una función viene determinada por la declaración de la palabra reservada **private, public o protected**. Por defecto si no se indica nada se entiende que es public. A esta acción se le llama modificador de acceso.



FUNCIONES

Ámbito y tipo de dato del valor que retornará la función

Parámetros de entrada

Instrucciones

Nombre de la función

Valor de retorno

```
private int sumar(int numero1, int numero2)
{
    int suma = numero1 + numero2;
    return suma;
}
```



MÉTODO

Un método también puede recibir valores, efectuar operaciones con estos y retornar valores, sin embargo **un método está asociado a un objeto, SIEMPRE**, básicamente un método es una función que pertenece a un objeto o clase.



PROCEDIMIENTO

Los procedimientos son básicamente un conjunto de instrucciones que se ejecutan sin retornar ningún valor.

Diagram illustrating the components of a Java procedure:

```
private void limpiar ()  
{  
    txtNumero1.setText(null);  
}
```

Annotations:

- Ámbito de la declaración** (Scope of declaration) points to `private`.
- Nombre del procedimiento** (Name of the procedure) points to `limpiar`.
- Instrucciones** (Instructions) points to the code block inside the curly braces.



LLAMADA A UNA FUNCIÓN Y PROCEDIMIENTO

Para llamar a una función en java se utiliza la siguiente línea de código:

```
public static int suma (int num1, int num2){  
    int suma = num1 + num2;  
    return suma;  
}
```

```
public static void main(String[] args) {  
    int resultado = suma (20,15);  
    System.out.println(" El resultado de la suma es: "+ resultado);  
}
```



LLAMADA A UNA FUNCIÓN Y PROCEDIMIENTO

Para llamar a una función en java se utiliza la siguiente línea de código:

```
public static void saludo (String nom)
{
    System.out.println("Bienvenido "+nom);
}

public static void main(String[] args) {
    Scanner leer = new Scanner(System.in);
    System.out.println("Ingresar la persona a saludar");
    String nombre = leer.next();
    saludo(nombre);
}
```



MODIFICADORES DE ACCESO

Modificadores de acceso (public, private, protected)

Los modificadores de acceso determinan la visibilidad de una función en relación con otras clases y objetos.

- public:** La función es accesible desde cualquier clase.
- private:** La función solo es accesible dentro de la misma clase en la que se define.
- protected:** La función es accesible dentro de la misma clase y sus subclases.

Sin modificador (por defecto): La función es accesible dentro del mismo paquete.



MODIFICADORES DE ACCESO

```
1
2 public class EjemploModificadores {
3     public void funcionPublica() {
4         // Accesible desde cualquier clase
5     }
6
7     private void funcionPrivada() {
8         // Accesible solo dentro de la clase EjemploModificadores
9     }
10
11    protected void funcionProtegida() {
12        // Accesible dentro de la clase EjemploModificadores y sus subclases
13    }
14
15    void funcionPorDefecto() {
16        // Accesible dentro del mismo paquete
17    }
18 }
19
```



TIPOS DE RETORNO

El tipo de retorno indica el tipo de dato que la función devolverá. Puede ser un tipo primitivo (int, float, etc.), una clase, una interfaz o void si la función no devuelve ningún valor.

```
1
2 public class EjemploTiposRetorno {
3     public int obtenerEntero() {
4         return 42;
5     }
6
7     public String obtenerCadena() {
8         return "Hola, mundo!";
9     }
10
11     public List<String> obtenerListaCadenas() {
12         return Arrays.asList("uno", "dos", "tres");
13     }
14
15     public void funcionSinRetorno() {
16         System.out.println("Esta función no devuelve ningún valor.");
17     }
18 }
```



TIPOS DE RETORNO

El tipo de retorno indica el tipo de dato que la función devolverá. Puede ser un tipo primitivo (int, float, etc.), una clase, una interfaz o void si la función no devuelve ningún valor.

```
1
2 public class EjemploTiposRetorno {
3     public int obtenerEntero() {
4         return 42;
5     }
6
7     public String obtenerCadena() {
8         return "Hola, mundo!";
9     }
10
11     public List<String> obtenerListaCadenas() {
12         return Arrays.asList("uno", "dos", "tres");
13     }
14
15     public void funcionSinRetorno() {
16         System.out.println("Esta función no devuelve ningún valor.");
17     }
18 }
```



EJERCICIOS

1. Realizar una función a la que se le pasan dos enteros y muestra todos los números comprendidos entre ellos, incluyendo los mismos números.
2. Realizar una función que muestra en pantalla el doble del valor que se le pasa como parámetro.
3. Realizar una función que recorra una matriz y encuentre el mayor número.
4. Realizar una función que calcule (muestre en pantalla) el área o el volumen de un cilindro, según se especifique. Para distinguir un caso de otro se le pasará el carácter 'a' (para área) o 'v' (para el volumen). Además hemos de pasarle a la función el radio y la altura.



EJERCICIOS

5. Realizar una función que ingresando datos como parámetros me permita seleccionar las siguientes opciones:

Calcular el área de un rectángulo.

Calcular el área de un cuadrado.

Calcular el área de un trapecio.

Calcular el área de un triángulo.

Ingresar por teclado todos los parámetros necesarios.



EJERCICIOS

```
package bo016ej05;

public class Main {

    static void mostrar(int a,int b){
        int mayor, menor;
        // desconocemos el orden en el que vienen a y b.
        // Lo que haremos es poner los valores correctos en mayor, menor.

        if(a>b){ // a es el mayor. Se podría utilizar la función maximo() implementada anteriormente
            mayor=a;
            menor=b;
        }
        else{ // en este caso b será el mayor

            mayor=b;
            menor=a;
        }
    }
}
```



EJERCICIOS

```
        for (int i=menor;i<=mayor;i++)
            System.out.print(i+" ");

        System.out.println();
    }

    public static void main(String[] args) {
        int a,b;

        System.out.print("Introduzca primer numero: ");
        a=Entrada.entero();
        System.out.print("Introduzca segundo numero: ");
        b=Entrada.entero();

        mostrar(a,b);
    }
}
```

