<codoa codo/>

Codo a Codo

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

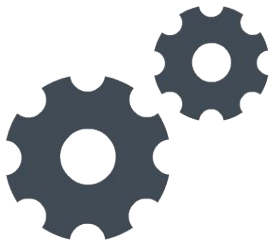
Write Once, Run Anywhere

(Escríbelo una vez, ejecútalo en cualquier lugar)

Funciones return

Funciones

- En términos generales, **una función es un “subprograma”** que puede ser **llamado desde dentro del mismo programa** como se verá más adelante, será el **caso de las recursiones** o **desde afuera** como típicamente se viene trabajando.
- Una función **se compone de una secuencia de declaraciones**, que conforman el llamado **cuerpo de la función**.
- **Se pueden pasar valores o “parámetros”** a una función, y la función **devolverá un valor,** mediante la cláusula **return**.



Sintaxis

```
static <tipo_funcion> [nombre](<tipo> argumento1, <tipo> argumento2...])
{

    // Bloque de instrucciones

    return <algo>;
}
```

¿Qué son los parámetros y los argumentos?

- **Los parámetros** En Java, un parámetro **es una variable declarada en la firma de un método**. Un parámetro se materializa en la **declaración de la función.**
- **Los argumentos**, En Java, un argumento **es el valor real que se pasa a un método cuando se llama.** Los argumentos **son los valores concretos que se suministran al método** y que se asignan a los parámetros del método en el orden en el que se especifican.

Acerca de los argumentos o parámetros

Hay algunos detalles respecto a los **argumentos** de una función, veamos:

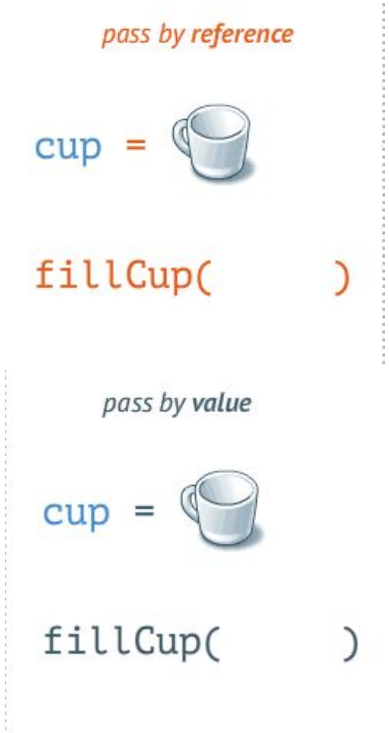
- Una función, un método o un procedimiento pueden tener una **cantidad cualquiera de parámetros**, es decir pueden tener cero, uno, tres, diez, cien o más parámetros. Aunque **habitualmente no suelen tener más de 3 y hasta 4 idealmente**.
- Si una función tiene **más de un parámetro** cada uno de ellos debe ir **separado por una coma**.

Acerca de los argumentos o parámetros

- Los **argumentos de una función** también **tienen un tipo y un nombre** que los identifica.
- El **tipo del argumento** puede ser **cualquiera** y **no tiene relación con el tipo de la función,** lo veremos en detalle cuando hablemos *acerca de return*.
- En Java **los parámetros** que podemos recibir pueden ser **por valor o por referencia**. Lo vemos a continuación.

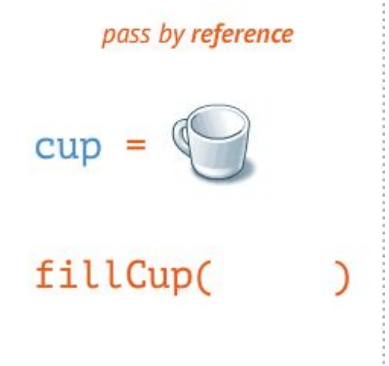
¿Qué es el paso por referencia y valor?

- Los conceptos de **paso por valor y por referencia** nacen de acuerdo a **como trata una función a los parámetros que se le pasan como entrada.**
- Para un parámetro **pasado por valor**, se creará una **copia local de la variable**, lo que implica que **cualquier modificación** sobre la misma **no tendrá efecto sobre la original.**
- Una variable **pasada como referencia,** significa que **se actuará directamente sobre la variable pasada**, por lo que **las modificaciones afectarán a la variable original.**
- En el **paso por referencia** el argumento contiene un puntero con la dirección de memoria de la variable.



Paso por valor en Java

- **En Java todos los argumentos se pasan por valor.**
- **El paso por valor significa** que al método **le llega una copia** del valor de la variable del argumento **en el caso de un tipo primitivo** de datos o una **copia del puntero** a la dirección de memoria **del objeto**.
- **En el paso por valor** al asignar un valor a la variable del argumento **no modifica el valor de la variable** original, esto ocurre tanto para argumentos de tipo primitivo como para objetos.



Acerca de la instrucción return

- **La sentencia return** se utiliza para **cerrar un bloque de código** en el cuerpo de una función, si está **seguida de una cláusula o valor o sintaxis**, **las devolverá como parte del método.**
- **Cualquier instrucción** que se encuentre **después de la cláusula return NO será ejecutada**.
- Es común encontrar funciones con **múltiples sentencias return** en el interior de condicionales, pero una vez que el código ejecuta una sentencia return **lo que existiera de allí hacia abajo no se ejecutará**.

```
static double suma(int num1, int num2){
    //Variables locales
    int suma = num1 + num2;
    //Valor de retorno
    return suma;

    //Todo lo que se encuentre luego de return
    // No se ejecuta.

}
```

```
Static <tipo> nombre (<tipo>nombre, <tipo>nombre){
    if (condicion1){
        //bloque;
        return <algo>;
    }
    if (condicion2){
            //bloque;
            return <algo2>;
        }
    if (condicion3){
            //bloque;
            return <algo3>;
        }
    }
```

Acerca de return

- **El tipo del valor** que se retorna en una función **debe coincidir con el del tipo declarado a la función**, es decir si se declara la función como int, el valor retornado debe ser un número entero, **lo cual no limita a que la función reciba parámetros de tipos diferentes**, esta es la explicación de lo dicho en la placa de argumentos y parámetros.
- **En el caso de los procedimientos (void)** podemos usar la sentencia return pero sin ningún tipo de valor, **sólo la usaríamos como una manera de terminar la ejecución del procedimiento**.

Ejemplos de función sin parámetro

```
static int sumaEntero()//Función sin parámetros
{
    int suma = 5+5;
    return suma; //Acá termina la ejecución del método
        //return 5+5+5 ; // Si colocaramos algo luego del
                         // return como por ejemplo otro
                         // return, este  nunca se ejecutará
}
```

Explicacion del ejemplo anterior

- El ejemplo sencillo anterior, es un método llamado:

```
sumaEntero( );
```

- Al ejecutarlo, la función retornará el valor de suma que es 10 (5+5).
- **Luego del return** toda línea posterior no se ejecutarán nunca, aunque no generan error alguno, no tienen utilidad.
- Para este caso es lo mismo haber escrito `return suma` que escribir `return 5+5`. Ambas líneas funcionan equivalentemente.

Ejemplos de función con parámetros

```
import java.lang.Math.*;

static double superficieCirculo(double radio){

    double sup = Math.PI*radio*radio;

    return sup;

}

static double perimetroCirculo(double radio){

    double perim = 2*Math.PI*radio;

    return perim;

}
```

Ejemplos de función con parámetros

```
import java.lang.Math.*;


static double superficieRectangulo(double base, double altura){
    double sup = base*altura;
    return sup;
}
static double perimetroRectangulo(double base, double altura){
    double perim = 2*base+2*altura;
    return perim;
}
```

Explicación del ejemplo anterior

- El ejemplo anterior, son funciones llamadas:

```
superficieCirculo(double radio);
perimetroCirculo(double radio);
superficieRectangulo(double base, double altura);
perimetroRectangulo(double base, double altura)
```

- Todas las funciones aceptan parámetros y devuelven valores.
- El valor de retorno es del tipo declarado en la funcion.

Detalles para invocar métodos funciones y procedimientos

- No importa si se trata de un método en Java o de una función o de un procedimiento, **siempre se deberá de enviar los parámetros de la forma correcta para invocarlos**.
- El **nombre debe coincidir exactamente al momento de invocar**, con el nombre **con el cual se la declaró**, pues es la única forma de identificarlo.
- **El orden de los parámetros y el tipo debe coincidir**. Hay que ser cuidadosos al momento de enviar los parámetros, debemos hacerlo en el mismo orden en el que fueron declarados y deben ser del mismo tipo (número, texto u otros).

Detalles para invocar métodos funciones y procedimientos

- Cada parámetro enviado también va **separado por comas.**
- Si una función **no recibe parámetros**, simplemente no ponemos **nada al interior** de los paréntesis, pero **SIEMPRE debemos poner los paréntesis**.
- **Invocar una función** sigue siendo una sentencia común y corriente en Java, así que esta **debe finalizar con ';'** como siempre.

Detalles para invocar métodos funciones y procedimientos

- El **valor retornado por un método o función** puede ser asignado a una **variable del mismo tipo**, pero no podemos hacer esto con un **procedimiento**, pues **no retornan valor alguno**.
- **Una función puede llamar a otra dentro de sí misma o incluso puede ser enviada como parámetro a otra.**

Desafío de clases

- Una función que halle el perímetro y la superficie de un círculo.
- Una función que halle el perímetro y la superficie de un rectángulo.
- Una función que imprima números
- Una función que imprima Strings.
- Un método main donde ejecutar todas las funciones.

Desafío de clase

Desafío de clase

Repasar la teoría y ejercicios de clase.
Clase que viene repaso general.

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizá los ejercicios obligatorios.

Todo en el Aula Virtual.