ELENA VILLAR ORTEGA 1WEM - PROGRAMACIÓN

Métodos, Procedimientos y Funciones

Java posee muchas funciones predefinidas (por ej para calcular el seno, la raíz cuadrada, el logaritmo y otras funciones científicas). El problema es que es imposible que un lenguaje de programación suministre todas las funciones que un programador podría necesitar.

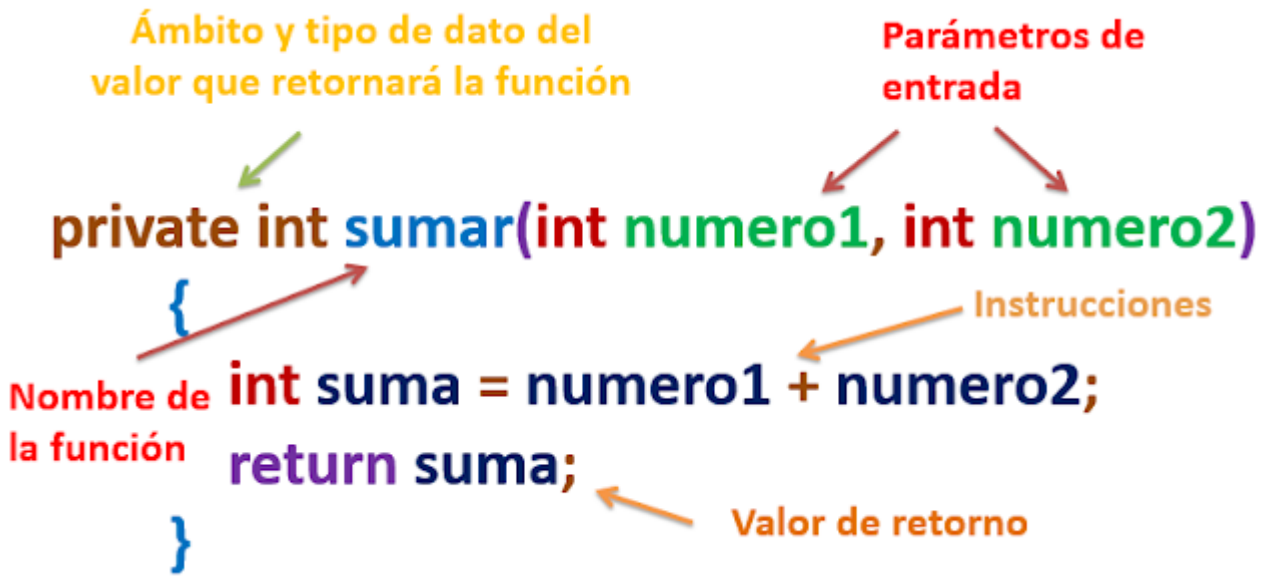
Por esta razón, prácticamente todos los lenguajes incorporan algún mecanismo para que los programadores puedan definir sus propias funciones cuando no existe la función predefinida apropiada. Para definir una nueva función, el programador debe escribir el código (programa) que calcula esa función.

Es muy común entre programadores que se hable indistintamente de estos tres términos sin embargo poseen diferencias.

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.

Sintaxis Funciones



Explicación:

- *tipo*: es el tipo del valor entregado por la función. Por ejemplo, en este caso int
- *nombre-función*: es un identificador que elige el programador para nombrar su función en este caso sumar
- *parámetros*: son los parámetros de la función, es decir de qué depende el cálculo de la función en este caso numero1(de tipo int) y numero2 (de tipo int)
- return *expresión*: señala cual es la expresión que representa el valor entregado por la función, en este caso suma, cuyo tipo debe coincidir con el tipo del valor de retorno
- Una función que no retorna ningún valor se define indicando que entrega un valor de tipo void. En este caso se prefiere hablar de *procedimiento*.
- void informa() { … }: es un procedimiento que no recibe argumentos y que no devuelve nada al entorno de la llamada

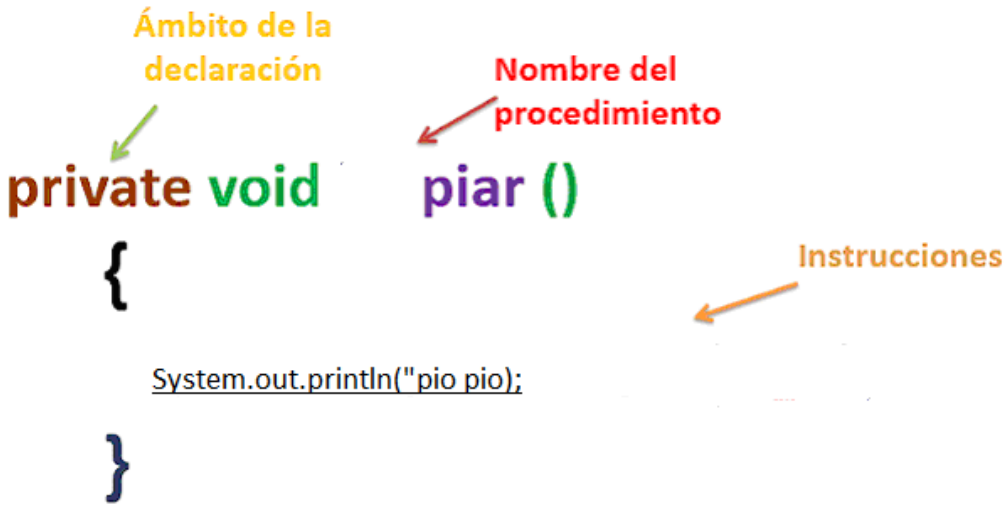
Modificador de acceso

La visibilidad de un procedimiento o función viene determinada por la declaración private, public o protected. Por defecto si no se indica nada se entiende que es public.

Métodos
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.

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

Sintasis Modificador de acceso Procedimientos



Invocación de una función

Sea la función

```
int factorial(int x) {
        // factorial recibe un argumento entero y entrega un resultado
entero.
        // El siguiente código calcula el factorial de x
        int prod= 1; //variable local prod
        int i= 1;       //variable local i
        while (i<=x) {
          prod= prod*i;
          i= i+1;
        }
        // Indica qué valor entrega esta función.
        return prod;
      }
    }
```

La firma de la función factorial es : `int factorial(int x)` que indica que para que el factorial pueda hacer su trabajo necesita un parámetro de tipo int y que devuelve al entorno de la llamada un valor de tipo también int

La forma general de una invocación o llamada de una función es:

nombre-función (argumento1,argumento2,…)

en donde *argumento1, argumento1,argumenton* son expresiones separadas por comas, son los parámetros actuales que sustituyen o hacen las veces de los parámetros formales definidos en la firma de la función . Por ejemplo:

```
factorial(n-k)
```

La semántica de una invocación de función es la siguiente:

- Se evalúan las expresiones, obteniendo así valores que corresponden a los argumentos de la función. Por ejemplo, n-k puede dar como resultado 5 en un caso particular
- Se asignan los argumentos a las variables que representan los parámetros en la definición de la función. Por ejemplo, en la función factorial
- `x= 5;`
- Se ejecutan las instrucciones señaladas en la definición de la función. Es decir:

```
int prod= 1;
int i= 1;
while (i<=x) {
  prod= prod*i;
  i= i+1;
}
```

Después de ejecutar este código, la variable prod almacena el valor 120 (5!) e i el valor 6.

- Se evalúa la expresión que acompaña return.

```
return prod;
```

Como esta expresión es simplemente la variable prod, el resultado de la evaluación es 120. Este valor es el resultado de la invocación de factorial y substituye toda la expresión de invocación:

```
factorial(n-k)
```

Término anticipado de una función

La línea return que aparece al final de las funciones es una instrucción como cualquier otra y por lo tanto puede aparecer en cualquier parte del código de la función:

```
return exp;
```

Cuando se ejecuta la instrucción return, se termina con la ejecución de la función en donde aparece y la función entrega el resultado de evaluar la expresión que acompaña return. Por ejemplo, la siguiente función calcula si un número es primo:

```
boolean esPrimo(int n) {
  int i= 2;
  while (i<n) {
    if (n%i==0) {
      return false;
    }
    i= i+1;
  }
  return true;
}
```

La instrucción return en este caso es más ``fuerte" que un break, puesto que no sólo termina el ciclo, también termina la función completa.

Ejercicios

1. Defina una función `mayor` que devuelva el valor máximo de 3 valores enteros.

```
package pruebaPyF;

public class ej1 {

	static int mayor(int num1,int num2, int num3) {
		int maximo=Integer.MIN_VALUE;
		if(num1>maximo)
			maximo=num1;
	    if (num2>maximo)
			maximo=num2;
	    if (num3>maximo)
			maximo=num3;
	    return maximo;

	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
	   System.out.println(mayor(14,5,13));
	}

}
```

Ó

```
package pruebaPyF;

public class ej1 {

	static int mayor(int num1,int num2, int num3) {
		if(num1>num2 && num1>num3) //else if: pq num1==num2 o num2>num1
ó num1==num3 o num3>num1
			return num1;
		else if (num2>num3) return num2;
	    return num3; (ESTO ES UN ELSE,como última opción)
```

```
	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
	System.out.println(mayor(5,14,9));
	}

}
```

Ó

```
package pruebaPyF;

public class ej1 {

	static int mayor(int num1,int num2, int num3) {
		if(num1>num2) {
			if(num1>num3) return num1;
			else return num3;
		}else {
			if(num2>num3) return num2;
			else return num3;
		}


	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
	System.out.println(mayor(5,14,9));
	}

}
```

2. Define una función que devuelva la media aritmética de 2 números double, con los decimales que indiquemos en la función como 3 parámetro

```
package metyproced;

public class ej2 {

	static double media(double num1,double num2, double media) {
		media=(num1 + num2)/2;
	  return media;
	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println(media(4.8,4.7,9.2));
	}

}
```

Ó

```
package metyproced;
import java.util.Scanner;
public class ej2 {

	static double mediaA(double num1,double num2, double dec) {
```

```
            double media=(num1 + num2)/2;
         double prueba = Math.pow(10, dec);
            return Math.round(media*prueba)/prueba; REDONDEA AL ENTERO MÁS
CERCANO
      }
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            Scanner sc = new Scanner(System.in);
            double num1, num2;
            int dec;
            System.out.println("Escribe una nota");
            num1 = sc.nextDouble();
            System.out.println("Escribe otra nota");
            num2 = sc.nextDouble();
            System.out.println("Escribe el número de decimales que deseas
obtener");
            dec = sc.nextInt();
            System.out.println("La media aritmética es: "+mediaA(num1, num2,
dec));
            sc.close();
      }

}
```

3. Define un procedimiento que muestre las opciones de un menú de 5 opciones (altas, bajas, modificaciones, consultas, salir)

```
package pruebaPyF;
import java.util.Scanner;
public class ej3 {

      public static void menu() {
            System.out.println("Elige una opción del menú:");
            System.out.println("1.Altas");
            System.out.println("2.Bajas");
            System.out.println("3.Modificaciones");
            System.out.println("4.Consultas");
            System.out.println("5.Salir");

      }
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            Scanner sc=new Scanner (System.in);
      int op;
            do {
                  menu();
                  op=sc.nextInt();

            }while(op!=5);
            System.out.println("Fin");
      }
}
```

4. Definir la función repite (s,n) que entrega el string s concatenado consigo mismo n veces. Es decir, se desea que `repite("hola",3)` entregue (o retorne) como resultado el string "holaholahola".

```
package pruebaPyF;
```

```
public class ej4 {

	static String repite (String s, int n) {
		String aux=" ";
		for (int i=0; i<n;i++) {
			aux+=s;
			//System.out.println(s);
		}return aux;
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
    System.out.println(repite("hola",3));
	}

}
```

5. Definir el procedimiento pintaTriangulo, con la siguiente firma `private static void pintaTriangulo(String relleno,int filas)` que produce la siguiente salida:

```
*
**
***
****
*****
******
*******
********
```

Si la llamada es `pintaTriangulo("*",8);`

SI SE PINTA TRIÁNGULO

```
package metyproced;
import java.util.Scanner;
public class ej5 {

	private static void pintaTriangulo (String relleno, int filas) {
		String aux=" ";
		if (filas<0) filas=-filas;

		for (int i=0;i<=filas;i++) {
			aux+=relleno;
			System.out.println(aux);

		}
	}


	static String repite(String  s, int n) {
		if(n<0) n=-n;
		String aux=" ";
		for (int i=0;i<n;i++) {
			aux+=s;
	}return aux;
	}
```

```
	public static void main(String[] args) {
		Scanner sc=new Scanner (System.in);
		String relleno;
		int filas;
		System.out.println("Introduce el relleno");
		relleno=sc.next();
		 System.out.println("Introduce el nº de filas");
		filas=sc.nextInt();
	   pintaTriangulo(relleno,filas);
	   sc.close();
	}

}
```

SI SE PINTA CUADRADO

```
package metyproced;
import java.util.Scanner;
public class ej5 {

	private static void pintaCuadrado (String relleno, int filas) {
		String aux=" ";
		if (filas<0) filas=-filas;
		for (int i=0;i<filas;i++) {
			for (int j = 0; j < filas; j++) {
				System.out.print(relleno);
			}
			System.out.println();

		}
	}


	static String repite(String  s, int n) {
		if(n<0) n=-n;
		String aux=" ";
		for (int i=0;i<n;i++) {
			aux+=s;
	}return aux;
	}


	public static void main(String[] args) {
	   pintaCuadrado("*",4);
	}

}
```

SI SE PINTA FLECHA

```
package metyproced;
import java.util.Scanner;
public class ej5 {

	 static void pintaFlecha(String relleno, int filas) {
		String aux=" ";
		if (filas<0) filas=-filas;
		for (int i=0;i<filas;i++) {
			aux+=relleno;
				System.out.println(aux);
```

```
                    }
                    for (int i = aux.length()-relleno.length(); i > 0; i-
=relleno.length()) {
                          System.out.println(aux.substring(0, i));
                    }

      }


      static String repite(String  s, int n) {
            if(n<0) n=-n;
            String aux=" ";
            for (int i=0;i<n;i++) {
                  aux+=s;
      }return aux;
      }


      public static void main(String[] args) {
         pintaFlecha("*",4);
      }



}
```

SI SE PINTAN TODOS A LA VEZ, ELIGIÉNDOLOS POR MEDIO DE UN MENÚ

```
package metyproced;
import java.util.Scanner;
public class ej5 {

      static String repite(String  s, int n) {
            if(n<0) n=-n;
            String aux=" ";
            for (int i=0;i<n;i++) {
                  aux+=s;
      }return aux;
      }

      private static void pintaTriangulo (String relleno, int filas) {
            String aux=" ";
            if (filas<0) filas=-filas;

            for (int i=0;i<=filas;i++) {
                  aux+=relleno;
                  System.out.println(aux);

            }
      }

      private static void pintaCuadrado (String relleno, int filas) {
            String aux=" ";
            if (filas<0) filas=-filas;
            for (int i=0;i<filas;i++) {
                  for (int j = 0; j < filas; j++) {
                        System.out.print(relleno);
                  }
                  System.out.println();
```

```
		}
	}


	 static void pintaFlecha(String relleno, int filas) {
		String aux=" ";
		if (filas<0) filas=-filas;
		for (int i=0;i<filas;i++) {
			aux+=relleno;
				System.out.println(aux);
			}
			for (int i = aux.length()-relleno.length(); i > 0; i-
=relleno.length()) {
				System.out.println(aux.substring(0, i));
			}

	}



	static void menu() {
		Scanner sc=new Scanner (System.in);
			System.out.println("Elige una opción:");
			System.out.println("1. Triángulo");
			System.out.println("2. Cuadrado");
			System.out.println("3. Flecha");
	}
	public static void main(String[] args) {
		Scanner sc=new Scanner (System.in);
		int opt, filas;
		String relleno;
		menu(); //hay que invocar el menú
		opt = sc.nextInt(); //escribe por telcado en el PUBLIC STATIC
VOID MAIN

	  switch (opt) {
	    case 1:{
		System.out.println("Introduce el relleno");
		relleno=sc.next();
		 System.out.println("Introduce el nº de filas");
		filas=sc.nextInt();
		  pintaTriangulo(relleno,filas);
		  break;
	    }
	    case 2:{
			System.out.println("Introduce el relleno");
			relleno=sc.next();
			 System.out.println("Introduce el nº de filas");
			filas=sc.nextInt();
		    pintaCuadrado(relleno,filas);
		    break;
		  }
	    case 3:{
		System.out.println("Introduce el relleno");
		relleno=sc.next();
		 System.out.println("Introduce el nº de filas");
		filas=sc.nextInt();
		  pintaFlecha(relleno,filas);
```

```
                break;
          }
            default:{
                System.out.println("Error.");
            }
            sc.close();
        }


}

}
```

Ayuda:

DIFERENCIA ENTRE ROUND, CEIL Y FLOOR

Round =return long

Ceil =return double

Floor= return int

Las funciones round, ceil y floor se usan para obtener un entero próximo a un número decimal y tienen similitudes, de hecho en algunos casos devuelven el mismo resultado. Sin embargo también tienen diferencias que es interesante conocer.

Estas tres funciones se aplican sobre valores numéricos decimales y retornan un valor numérico que en el caso de round es un entero long, mientras que en el caso de floor y ceil retornan un valor de tipo double coincidente o equivalente con un entero.

El método **round** redondea siempre al entero más próximo, por ejemplo 2.6 redondea a 3 mientras que -2.6 redondea a -3. Si el decimal está exactamente entre dos valores se redondea al entero superior más próximo (por ejemplo 2.5 redondea a 3 y -2.5 redondea a -2).

El método **floor** diremos que devuelve el entero menor, por ejemplo 2.9 quedaría en 2.0 y -2.9 quedaría en -3.0. También 2.1 quedaría en 2.0 y -2.1 quedaría en -3.0.

El método **ceil** diremos que devuelve el entero mayor, por ejemplo 2.9 quedaría en 3.0 y -2.9 quedaría en -2.0. También 2.1 quedaría a 3.0 y -2.1 quedaría en -2.0.