MÉTODOS

1. INTRODUCCIÓN

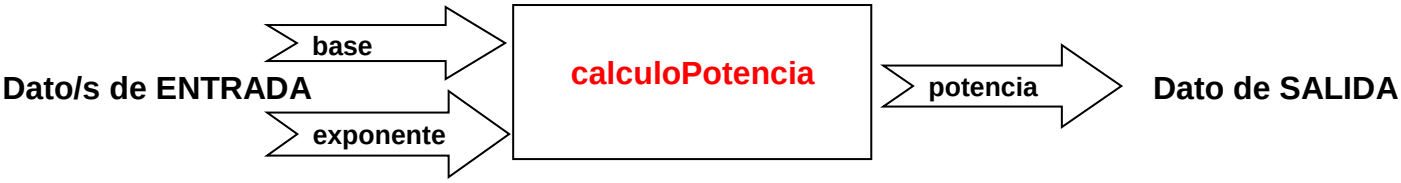
Podemos definir un método como un conjunto de instrucciones que se agrupan con un objetivo común, y ayudan a resolver parte de un problema.

En cuanto a las ventajas que aportan podemos citar:
- la **claridad** en el programa ya que se reduce el tamaño del código fuente del programa principal,
- la **partición** de un problema en partes más pequeñas (divide y vencerás),
- **reutilización** de código

En los métodos podemos destacar dos aspectos:

- la **declaración/definición** del mismo que sería definir el conjunto de sentencias, tipos de datos, parámetros necesarios, que forman parte del mismo. Es decir, indicamos como deben realizarse las operaciones o cálculos, y por otro,

```
public static  int   calculoPotencia( int base,  int exponente ) {...código...}
```



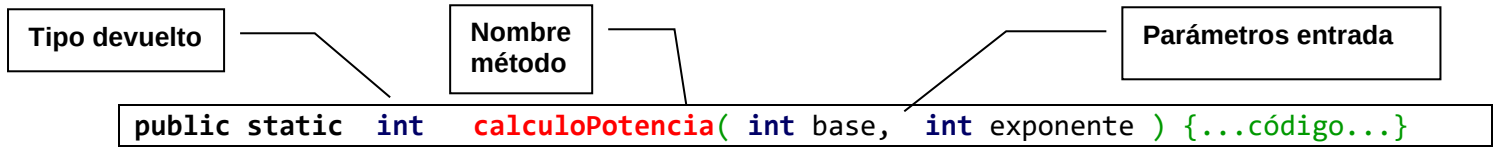
- la **invocación/llamada/ejecución** que normalmente se suele realizar desde el programa principal o también se puede realizar desde otro método. Se trata de usar el método (ejecutarlo), llamarlo pasándole la información que necesita y recogiendo su resultado.

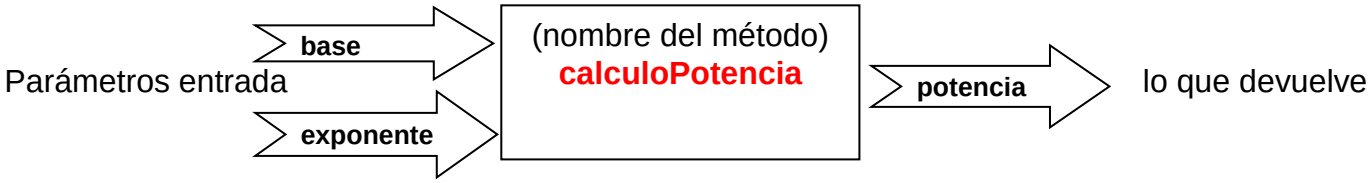
```
calculoPotencia(2, 5);  //calculará 2 elevado a 5
```

2. ELEMENTOS DE UN MÉTODO

Todo método cuenta en su **definición/declaración** con los siguientes elementos:

- **Cabecera**:
  - Tipo de datos que devuelve tras el cálculo. Si no devuelve nada se pone `void`.
  - Un nombre o identificador que nos permite identificarlo dentro del contexto del programa.
  - Una lista de parámetros entre ( ) que consiste en una lista, separada por comas, de variables con sus tipos de datos asociados, que representan a los datos externos que necesita, por tanto de entrada al método. Un método puede no necesitar ningún parámetro, entonces no se pondrá nada solo los paréntesis.





- **Cuerpo**:
  - Una sección de declaración de variables locales al método
  - El bloque de instrucciones de código asociadas al método.

Como Java es un lenguaje de Objetos los métodos siempre pertenecerán o estarán asociados a una determinada clase. Los métodos en Java son funciones definidas dentro de una clase y pueden devolver un valor o no, dependiendo de lo que nos interese. Por eso, en la práctica los métodos se pueden comportar tanto como:

- **Funciones**, cuando devuelven un valor de forma explícita
- **Procedimientos**, cuando no devuelven ningún valor de forma explícita → **void**

3. SINTAXIS DE CREACIÓN DE MÉTODOS

A continuación, se muestra la estructura que se debe seguir a la hora de crear/definir/declarar un método:

```
[<Accesibilidad>] <tipo_dato_devuelto ó void> <nombre_metodo> ( [<lista_parametros>]  ){
  <declaracion de variables locales>
  <Bloque de instrucciones>
   [return valor;]
}
```

Ejemplo:

```
public static  int   calculoPotencia( int base,  int exponente ) {
  int potencia=1;

  for(int exp=1; exp <= exponente ; exp++){
      potencia = potencia * base;
  }

  return potencia;
}
```

CABECERA

CUERPO

[<Accesibilidad>] → o cualificador de acceso, de momento pondremos siempre public static , y la veremos más adelante cuando veamos las Clases y la orientación a Objetos en Java.

<tipo_dato_devuelto> → representa el tipo de dato que devuelve el método (String, Integer, int, float, char, ...). Cuando un método no devuelve ningún dato (se comporta como un procedimiento), se utiliza la palabra reservada **void** y en este caso no debe aparecer la palabra return dentro del cuerpo de la función.

<nombre_metodo> → Deben comenzar con letra minúscula, y si está compuesta por 2 palabras, la primera letra de la segunda palabra debe comenzar con mayúscula. De preferencia que sean verbos.
Ejemplos: arrancarCoche(), sumar(), calculoRaizCuadrada()....

(<lista_parametros>) → tipo de datos de cada parámetro, como la declaración de una variable:
(**tipoDato nombreParámetro)**

Si hay 2 ó más parámetros los expresaremos separados por una coma:
(int dato1, double dato2, int dato3, float dato4)

Si no necesita ningún parámetro los expresaremos con **( )**

En los métodos que devuelven algún dato, la forma de indicar el valor que devuelven es con la palabra reservada **return** seguida del valor en cuestión. A continuación se muestran diferentes formas de indicar el valor devuelto:

return constante; → return 5;
return variable; → return num_articulo;
return expresión; → return (a+b);

La palabra **return** es recomendable que aparezca una única vez al final del código.

PASO DE PARÁMETROS

Los métodos cuentan con un mecanismo de comunicación de información con el programa principal y con otros métodos desde los que puedan ser invocados o llamados. Este mecanismo conocido como **paso de parámetros** permite que se pueda intercambiar información, entre los métodos y el programa principal.

Los métodos contarán con una lista de parámetros (variables junto a sus tipos asociados), los cuales se definen en la declaración del método, en este caso se denominan **lista de parámetros formales**. Cuando invocamos o llamamos a un método entonces hablamos de **parámetros actuales**, es decir los parámetros formales se hacen actuales (los valores concretos que toman los parámetros formales) en una determinada llamada o ejecución.

```
calculoPotencia(int numeroBase,  int exponente) → parámetros formales (en definición)


calculoPotencia(base, exp);  → parámetros actuales (en una llamada)
```

INVOCACIÓN-LLAMADA-EJECUCIÓN DE MÉTODOS

La forma de invocar o ejecutar a los métodos desde el programa principal o desde cualquier otro método, consiste en poner su nombre con la lista de parámetros actuales encerrados entre paréntesis, teniendo en cuenta que si el método devuelve algún valor, hemos de emplear el nombre del método como si se tratara de una variable. Ejemplo de invocación de un método:

```
// Invocación de un método que no devuelve ningún valor
    ...
    metodo1();
    ...
// Invocación de métodos que devuelven algún valor
    ...
    resultado = metodo2();
    ...
    if (metodo3() > 0)
    ...
```

Ejemplo de invocación:

```
public static void main(String[] args) {
        int base=5, exp=3, resultado;

        resultado = calculoPotencia(base, exp);  //calcula 5³
        System.out.println(resultado);

       //CIUDADO esto NetBeans no lo controla sería un error nuestro, calcularía 3⁵
        resultado = calculoPotencia(exp, base);

        System.out.println( calculoPotencia(2,4) );
    }
```

La secuencia de pasos que tiene lugar cuando se invoca un método es la siguiente:

1. Se crea o reserva la zona o espacio de memoria para el método, creando los objetos locales que tenga el método
2. Se transfieren la información desde los parámetros actuales a los formales (sólo en el caso de que existan parámetros)
3. Se transfiere el control al método y se ejecuta el bloque de código del mismo
4. Se borra el espacio de memoria del método y se transfiere el control al programa principal o al método que lo hubiera llamado

3.1.- VISIBILIDAD DE LOS IDENTIFICADORES EN JAVA

Podemos definir la **visibilidad o ámbito de una variable** como la zona del programa donde dicha variable es accesible y por lo tanto puede ser utilizada en cualquier expresión.

Las variables definidas dentro de un bloque de programa delimitado por las llaves { ... }, sólo serán visibles y existirán dentro del bloque definido por estas llaves.

Esto mismo también puede ser aplicado a los métodos, de manera que las variables declaradas en un método, existirán desde el lugar del método en el que son declaradas hasta el final del bloque en el que han sido definidas. Fuera del método estas variables ya no son accesibles ni utilizables.

3.2.- DATOS ESTRUCTURADOS COMO PARÁMETROS

En java existe la posibilidad de utilizar como parámetros tipos de datos estructurados tales como los arrays, colecciones u objetos de una clase. Cuando pasamos un dato de este tipo a un método, en lugar de copiar todos los elementos desde la zona de memoria del programa principal a la zona de memoria del método en cuestión, lo que se copia únicamente es **la referencia o la dirección de memoria** donde comienza el dato estructurado. Por tanto, es como si el parámetro se pasara por referencia, al modificar los elementos del parámetro en el método, automáticamente quedarán modificados en la zona de memoria del programa principal. Es decir, estaremos modificando los datos originales.

El hecho de que los datos estructurados se pasen por referencia tiene sentido, puesto que, si el número de elementos de un array es muy grande, la acción de realizar la copia de todos los elementos a la zona de memoria del método, tendría un **coste espacial y temporal muy alto**. Es decir, tendríamos que utilizar memoria y tiempo extra para realizar esta copia.

4.- RECURSIVIDAD

En general podemos decir que la recursividad consiste en **definir un concepto en términos del propio concepto que se está definiendo**.

Ejemplo:

"El factorial de un número natural n, es 1 si dicho número es 0, o n multiplicado por el factorial del número n-1, en caso contrario".

*Factorial(N) = N * Factorial(N-1)*
*Factorial(N-1) = N-1 * Factorial(N-2)*
...
Factorial(0) = 1

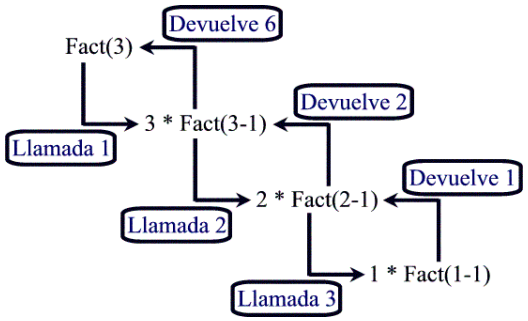
El cálculo tradicional del factorial de un número sería:

Factorial de 7 o 7! → *7 * 6!*
Factorial de 6 o 6! → *6 * 5!*
….

Trasladado a la programación, diremos que un **método es recursivo** si dentro del código del mismo, aparece una llamada así mismo, es decir el método se invoca así mismo.

```
public static void main(String[] args) {
  long numero=6, resultado;

  resultado = factorial(numero);

  System.out.println("FACTORIAL de " + numero + " es " + resultado);
}
```

```
// ***************** MÉTODO FACTORIAL RECURSIVO ***********************

public static long factorial(long numero){
   long fact;

   if (numero==0){
       fact = 1;
   }else{
       fact = numero * factorial(numero-1);
   }
   return fact;
}
```



Ejemplo de ejecución para calcular el factorial de 3 → factorial(3)

EJERCICIO RESUELTO: Método que escribe la tabla de multiplicar dado un número.

```
public class metodoTablaMultiplicar {

    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        int numero;
        String tabla;

        System.out.println("Indica el numero de la tabla de multiplicar:");
        numero = teclado.nextInt();

        tabla = generaTablaMultiplicar(numero); //LLAMADA/EJECUCIÓN DEL MÉTODO
        System.out.println("Tabla del " + numero + tabla);
    }


//DECLARACIÓN/DEFINICIÓN DEL MÉTODO
    public static String generaTablaMultiplicar(int num) {
        String texto = "";

        for (int i = 1; i <= 10; i++) {
            texto = texto + "\n" + num + "x" + i + "=" + (num * i);
        }

        return texto;
    }
}
```

FIJAROS: La recogida de información y su muestra por pantalla, se realizan en el main y los métodos solo calculan o generan información, no realizan ningun print()!!!

```
run:
Indica el numero de la tabla de multiplicar:
6
Tabla del 6
6x1=6
6x2=12
6x3=18
6x4=24
6x5=30
6x6=36
6x7=42
6x8=48
6x9=54
6x10=60
BUILD SUCCESSFUL (total time: 4 seconds)
```