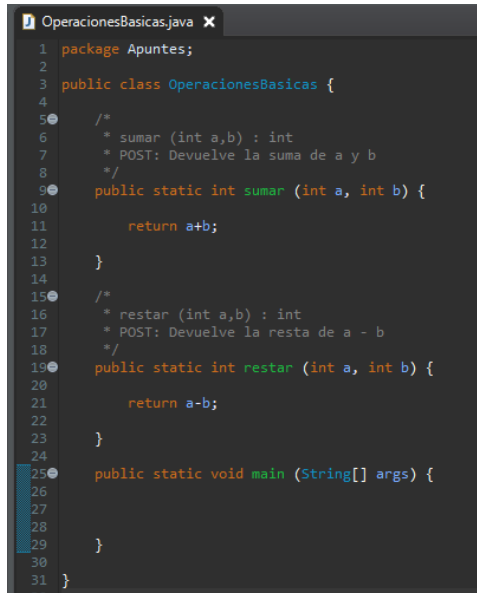


Las funciones y procedimientos son como pequeños bloques de código que podemos llamar cuando necesitemos desde cualquier lugar, ya sea desde la propia clase o desde otra (si esta cumple el requisito de visibilidad, cosa que veremos más adelante, quedémonos con que todas las clases y funciones que creamos van a ser de tipo *public*). Voy a crear una clase que tenga funciones que realicen las operaciones básicas:



```
1 package Apuntes;
2
3 public class OperacionesBasicas {
4
5     /*
6      * sumar (int a,b) : int
7      * POST: Devuelve la suma de a y b
8      */
9     public static int sumar (int a, int b) {
10
11         return a+b;
12     }
13
14
15     /*
16      * restar (int a,b) : int
17      * POST: Devuelve la resta de a - b
18      */
19     public static int restar (int a, int b) {
20
21         return a-b;
22     }
23
24     public static void main (String[] args) {
25
26
27
28
29     }
30
31 }
```

He creado dos funciones, sumar y restar, las funciones/procedimientos tendrán para nosotros la siguiente forma **de momento**:

```
public static TIPO_DE_DATO_A_DEVOLVER NOMBRE_DE_LA_FUNCION (PARAMETROS) { }
```

Es decir, ponemos public static, el tipo de dato a devolver, el nombre de nuestra función, y luego entre paréntesis los tipos de datos que va a recibir, en ambos casos yo le tengo que pasar dos variables de tipo entero, el nombre de estos parámetros no tiene que coincidir con el nombre de las variables que le pasemos, recomiendo comentar tal y como yo he hecho lo que hacen nuestras funciones/ procedimientos, con la siguiente forma:

```
NOMBRE_DE_LA_FUNCION (PARAMETROS) : TIPO_DE_DATO_A_DEVOLVER
```

```
POST: <Resumen de lo que hace nuestra función/procedimiento>
```

Si tenemos varios tipos de parámetros, por ejemplo 2 *ints* y un *boolean*, los separaremos de la siguiente forma en el comentario -> (*int a,b; boolean c*)

Mientras que en el paso real de parámetros pondremos -> (*int a, int b, boolean c*)

De momento no podemos hacer funciones complejas con lo poco que sabemos, así que en lo que queda de PDF todo será teoría.

Dos funciones no pueden tener el mismo nombre y recibir los mismos parámetros en el mismo orden. Veamos esto con un ejemplo:

```
4
5  /*
6   * sumar (int a,b) : int
7   * POST: Devuelve la suma de a y b
8   */
9  public static int sumar (int a, int b) {
10
11      return a+b;
12  }
13
14
15  /*
16   * sumar (int a,b,c) : int
17   * POST: Suma 3 numeros
18   */
19  public static int sumar (int a, int b, int c) {
20
21      return a+b+c;
22  }
```

Como veis estas dos funciones se llaman *sumar*, pero una recibe *(int, int)* y la otra *(int, int, int)*, ya que el programa puede diferenciarlas perfectamente, aun así, esta es una mala práctica de programación conocida como **sobrecarga**, ocurre cuando varias funciones tienen el mismo nombre, también no solo importa la cantidad de parámetros sino el orden, por ejemplo:

```
3  public class OperacionesBasicas {
4
5      /*
6       * sumar (int a,b) : int
7       * POST: Devuelve la suma de a y b
8       */
9      public static int sumar (int a, int b) {
10
11          return a+b;
12      }
13
14
15      /*
16       * sumar (int a,b,c) : int
17       * POST: Suma 3 numeros
18       */
19      public static int sumar (int a, int b, int c) {
20
21          return a+b+c;
22      }
23
24
25      /*
26       * sumar (int a; double b) : double
27       * POST: Suma dos elementos
28       */
29      public static double sumar (int a, double b) {
30
31          return a+b;
32      }
33
34
35      /*
36       * sumar (double b, int a) : double
37       * POST: Multiplica dos elemetos (soy consciente del nombre de la funcion)
38       */
39      public static double sumar (double b, int a) {
40
41          return b*a;
42      }
43  }
```

Como veis tengo 4 funciones que se llaman igual, pero que se consideran distintas por que reciben diferentes parámetros, fijémonos en las dos última, aunque ambas reciben un *int* y un *double*, el programa sabe que son dos funciones diferentes ya que una recibe *(int, double)* y la otra *(double, int)*, estas dos últimas realizan cosas diferentes, ya que una suma y la otra multiplica (puse ese nombre, aunque no tenga nada que ver, solo como ejemplo) pero el programa sabe a cual me refiero por el **orden**.

Espero que ahora tenga más sentido la primera frase de esta página.

DIFERENCIA ENTRE FUNCIONES Y PROCEDIMIENTOS

Es muy sencillo, las funciones devuelven un valor, mientras que los procedimientos no devuelven nada, solo realizan procesos, se pueden diferenciar fácilmente fijándonos en el **tipo de dato que devuelven**, los procedimientos nos devolverán un tipo *void*, que es el que viene en la función *main* -> `public static void main (String[] args) {`

Esto quiere decir que no devuelven nada, osea, que solo realizan **procesos**, mientras que las funciones devuelven un tipo, el que sea, mientras **no** sea *void* -> int, double, char, String...

Como llamar a estas funciones

COMO LLAMAR A NUESTRAS FUNCIONES

Desde la misma clase

```
55 public static void main (String[] args) {
56
    int a = 5;
    int nombreDiferenteAlParametro = 7;
    int c = sumar (a,nombreDiferenteAlParametro); // primera funcion -> 12
59
    int d = sumar (5,7,9); // segunda funcion -> 21
60
    double doble = 12.63;
    int numeroEntero = 25;
    double g = sumar (numeroEntero, doble); // tercera funcion -> 37.63
    double f = sumar (doble, numeroEntero); // cuarta funcion -> 315.75
67
68 }
```

También se pueden llamar funciones desde otras funciones, es como si llamas a un bloque de código que devuelve un valor, y lo puedes prácticamente donde quieras, ya que, si nos fijamos, una función no es más que un valor del tipo que tenga que devolver, si es un entero como en las líneas 59 y 61, las variables *c* y *d* se les asigna un número entero, los procedimientos sin embargo están más restringidos, ya que no representan un valor, solo realizan procesos, como escribir en pantalla, o realizar cambios en **arrays**, pero esto ya lo veremos en su apartado.

```
/*
 * sumar (int a,b) : int
 * POST: Devuelve la suma de a y b
 */
public static int sumar (int a, int b) {

    return sumarAux (a,b);

}

public static int sumarAux (int a, int b) {

    return a+b;

}
```

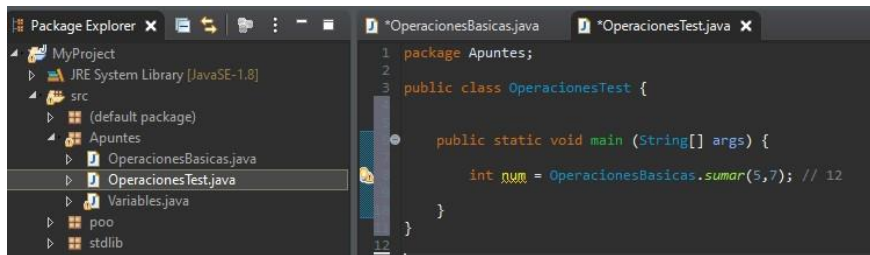
Como veis llamo a la función *sumarAux* desde otra función, la función *sumarAux* debería ser de tipo *private* en vez de *public*, pero eso ya lo veremos en temas de visibilidad.

Desde otra clase

Como ya expliqué en el anterior PDF, cuando llamamos a la función *Math.pow(BASE, EXPONENTE)*, estamos llamando a la función *pow* de la clase *Math*, y pasándoles dos parámetros, uno que será la base (el primero), y el otro el exponente. Y esta función nos devuelve el resultado, ej: *Math.pow(3,2) -> 9*

Ahora tratemos de llamar a nuestra función *suma* desde otra clase, pueden pasar dos cosas, que la otra clase este en el mismo paquete, o que no:

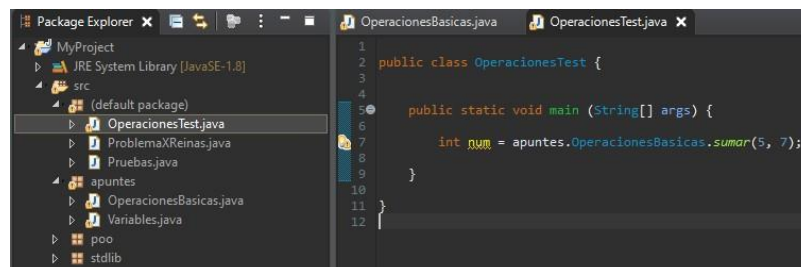
Mismo paquete



OperacionesTest está en el mismo paquete que *operaciones básicas*, por lo que para usar las funciones basta con poner el *NombreDeLaClase.funcion(PARAMETROS)*. Siempre y cuando la función no sea de tipo *private*.

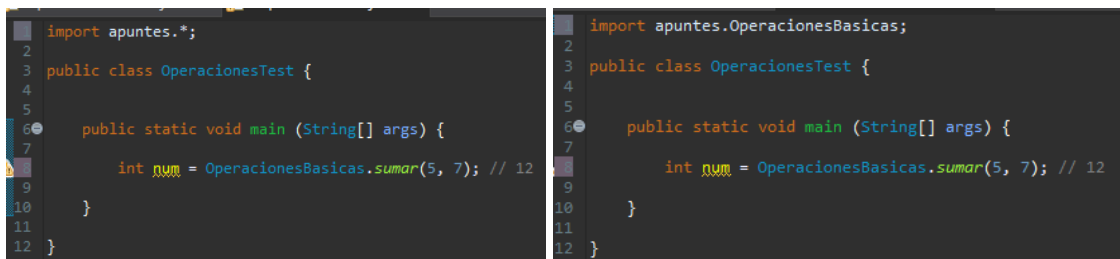
Distinto paquete

(Había llamado *Apuntes* al paquete, y no puede llevar mayúscula al principio, se lo cambié)



De esta forma llamamos a la función de la siguiente forma:

paquete.NombreDeLaClase.funcion(PARAMETROS). Siempre y cuando la función sea del tipo *public*, esto se explicará en **visibilidad**, también podemos importar el paquete entero o solo la clase poniendo esta sentencia, de las siguientes formas respectivamente:



En la primera importo todas las clases del paquete *apuntes*, mientras que en la segunda solo la clase *OperacionesBasicas*, recordad que si la clase pertenece a un paquete, la primera sentencia es la de *package nombrePaquete;* y luego ya ponemos los imports.