**Espacio de Memoria y Stack de un Proceso**

**PROCESO**: programa en ejecución dentro de la memoria central de la computadora.

La memoria de un proceso se divide (de manera lógica, no física) en varios segmentos, cuatro a saber:

* *Segmento de datos*: Parte de la memoria destinada a almacenar las variables estáticas.
* *Segmento de código*: Parte de la memoria donde se guardan las instrucciones del programa en código máquina.
* *Montículo (Heap)*: Parte de la memoria destinada a las variables dinámicas.
* *Pila ejecución del programa (Stack)*: Parte destinada a las variables locales y parámetros formales del método o función que está siendo ejecutada.



**Llamada a una función o método desde main(…):**

Cada vez que se invoca o llama a una función o método, se definen nuevamente sus parámetros formales y variables locales asignándoseles los valores iniciales correspondientes. Dichos espacios (parámetros formales y variables locales) se crean en la *Pila* (*Stack*) del programa.

En primer lugar, antes que los parámetros y variables locales, se guarda en la pila la dirección de la línea de código desde donde se ha llamado a la función o método.

Al terminar el método o la función, se libera la memoria asignada en la pila, se guarda en la pila el valor de retorno (return), si existe alguno, y se vuelve a la instrucción invocadora o del llamado.

p.e.

|  |
| --- |
| public static void main( String args[] ) {  int n;  double sum;  double [] arrd = {3.0, -10.0, 50.0, -20.0, 12.0};  n = 3;  sum = sumaD( arrd, n ); // Llamado o invocación a un método.  // Parámetros efectivos o reales: arrd, n  System.out.println(“sum= “ + sum);  { |
| public static double sumaD( double [] arr, int nelem ) {  // Parámetros formales: arr, nelem  double acum = 0.0; // variable local  for( int i = 0; i < nelem; i++ ) { // i es variable local  acum = acum + arr[i];  }  return acum; |

