

CFGS Desarrollo de aplicaciones multiplataforma

Módulo profesional: Programación



**GENERALITAT
VALENCIANA**

Conselleria d'Educació,
Investigació, Cultura i Esport



Unió Europea

Fons Social Europeu

L'FSE inverteix en el teu futur



Material elaborado por:

Edu Torregrosa Llácer

(aulaenlanube.com)

Esta obra está licenciada bajo la licencia **Creative Commons**
Atribución-NoComercial-CompartirIgual 4.0 internacional. Para ver una
copia de esta licencia visita:
<https://creativecommons.org/licenses/by-nc-sa/4.0/>



Attribution-NonCommercial-ShareAlike
4.0 International (CC BY-NC-SA 4.0)

Métodos en JAVA



1. Introducción
2. Declaración de métodos.
3. Parámetros.
4. Paso por valor y el método main.
5. Variables locales y globales.
6. Sobrecarga de métodos
7. La pila de llamadas.
8. Recursión.
9. Ejemplos de algoritmos recursivos.

Introducción

- En muchas ocasiones, se tiene que usar un mismo bloque de código para resolver un mismo problema con datos **diferentes**. Escribir el mismo código varias veces es un trabajo repetitivo, improductivo, y difícil de mantener y modificar. Por todo ello, este tipo de código tiene una gran tendencia a provocar errores.
- La mejor forma de crear y mantener un programa grande es construirlo a partir de piezas más pequeñas(subprogramas). Cada una de estas piezas es más manejable que el programa en su totalidad.

Los **métodos**(subprogramas) son utilizados para evitar la repetición de un bloque de código en un programa. Se pueden ejecutar desde varios puntos de un programa con solo invocarlos.

Introducción

- Un **método** es un bloque de código que:
 - Se usa (se **invoca** su ejecución) desde algún punto del código para resolver un problema dado.
 - Este **método** puede emplear datos externos(**parámetros**), dichos datos deben ser proporcionados cuando llamamos(**invocamos**) al método. Si proporcionamos variables como parámetros, siempre se proporcionará una copia de la variable original(**paso por valor**).
 - Devolverá(**return**) un resultado(**valor de retorno**) en base a los parámetros que le hayamos proporcionado. Hay métodos que no devuelven nada, simplemente ejecutan el código incluido dentro del método.

Declaración de métodos en JAVA

Declaración de métodos

El método es el mecanismo de subprogramación que ofrece el lenguaje JAVA.

Un **método** se declara definiendo su **cabecera** y su **cuerpo**, dentro de una clase JAVA.

La **cabecera** de un método se define, en este orden:

- **Acceso**(modificadores de visibilidad y directiva static).
- **Tipo de retorno**, o tipo de sus resultados (valores de retorno).
- **Nombre** (identificador).
- **Lista de parámetros**, o los tipos y nombres de sus datos (parámetros).

El **cuerpo** es la secuencia de instrucciones que se ejecutan al invocarlo, puede incluir:

- Declaraciones de variables.
- Cualquier tipo de instrucción o bloque de instrucciones: asignación, condicional, bucle, e incluso, **llamadas o invocaciones a otros métodos**.

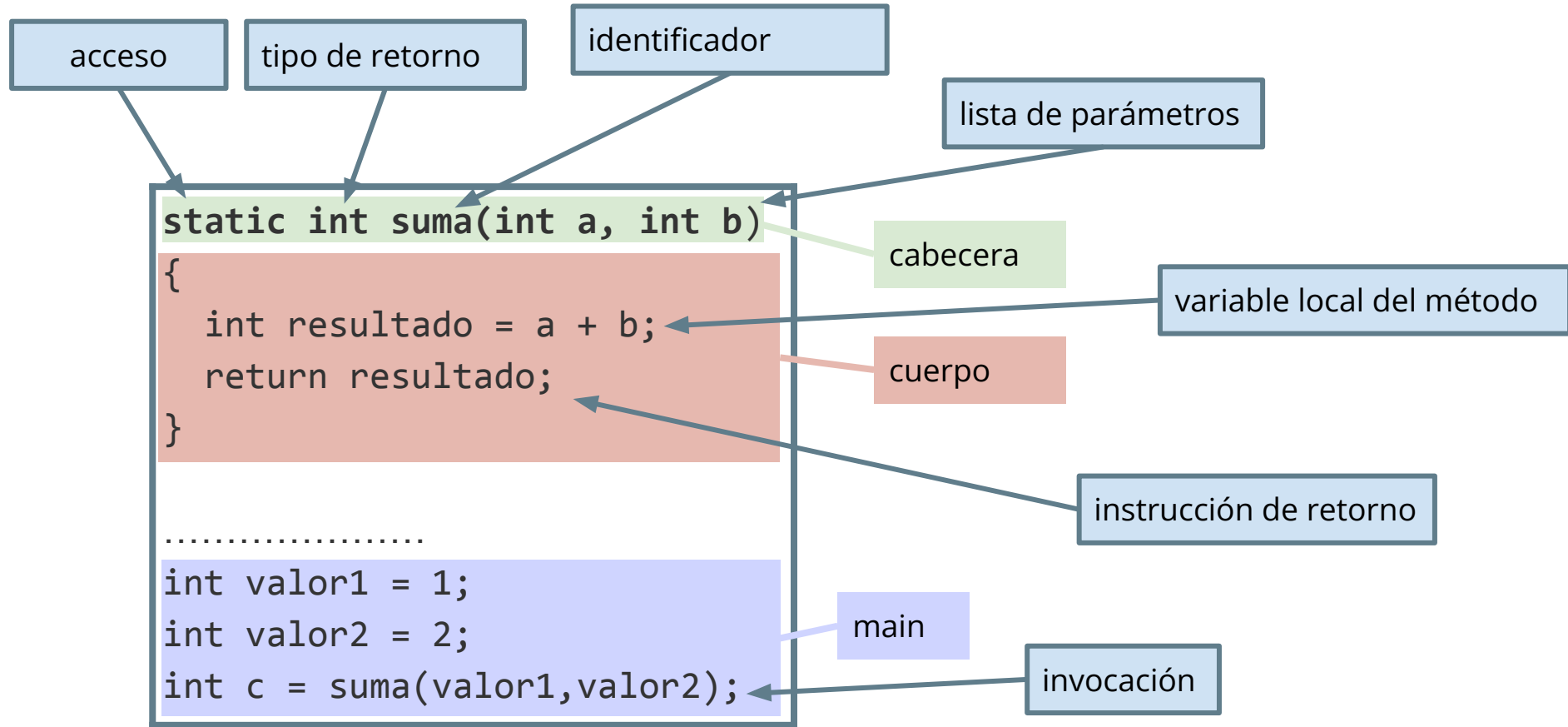
Declaración de métodos

Aquí se muestra un ejemplo de método en JAVA con su correspondiente invocación:

```
static int suma(int a, int b)
{
    int resultado = a + b;
    return resultado;
}
```

```
.....
int valor1 = 1;
int valor2 = 2;
int c = suma(valor1,valor2);
```


Declaración de métodos



Declaración de métodos

- Identificación de los componentes de la **cabecera** de un método:

acceso	tipo de retorno	identificador	lista de parámetros
static	double	areaRectangulo	(double base, double altura)

- Por ahora, normalmente se usará el modificador **static** para describir el acceso de un método, en temas posteriores se darán más detalles sobre esto.
- El nombre de un método y el de sus parámetros siguen las reglas habituales de los identificadores.
- En JAVA se debe indicar el tipo de dato de retorno antes del identificador.

```
static int suma(int a, int b) { ... } /*declaramos un método con  
identificador suma y que retorna un valor entero(int) */
```

Declaración de métodos - return

- Como resultado de su invocación, un método devuelve unos resultados, los denominados valores de retorno.
- El tipo de retorno de un método puede ser cualquier tipo de dato en JAVA (int, double, char, String, boolean, etc.). También pueden devolver objetos.
- Para devolver un valor, dentro del método se debe utilizar la instrucción **return**. En concreto, el valor de retorno debe aparecer después de la palabra reservada return. Además, se deben tener en cuenta las siguientes consideraciones:
 - La ejecución de un método **finaliza tras la ejecución de cualquiera de los return** que aparecen en su cuerpo
 - **Si un método no devuelve explícitamente ningún valor**, entonces el tipo de su resultado es **void**. En este caso, implícitamente, se ejecuta “return” como última instrucción del método

Declaración de métodos - return

Ejemplos de métodos con y sin valores de retorno:

```
static double areaCuadrado(double lado)
{
    return lado*lado;
}
static double perimetroCuadrado(double lado)
{
    return lado*4;
}
static void mostrarMenu()
{
    System.out.println("Esto es un método que muestra un menú");
    System.out.println("1 - Obtener área");
    System.out.println("2 - Obtener perímetro");
}
```

Declaración de métodos - Parámetros

- Es al invocar a un método cuando se le proporcionan los valores de los datos (parámetros) requeridos para ejecutar su código
- Los parámetros definidos en la cabecera de un método, al declararlo, se denominan parámetros formales pues no tienen asociado un valor real
 - **Sintaxis: tipo1 nomPar1, tipo2 nomPar2 ... tipoN nomParN**
- Los parámetros, a través de sus identificadores pueden usarse como variables en el cuerpo del método, **NO deben volver a declararse**.
- **Se pueden declarar métodos sin parámetros, en este caso se utilizarán paréntesis vacíos después del identificador.**

```
static void mostrarMenu( ) { ... }
```

Uso de métodos - Invocaciones

- **Un método no se ejecuta por sí mismo**, debe ser invocado para ejecutarse.
- Para invocar o llamar a un método se necesita escribir:
 - Su identificador.
 - Los datos con los que trabajará, es decir, la lista de sus parámetros actuales. siempre y cuando reciba parámetros.

Veamos las siguientes invocaciones, supongamos que los métodos existen:

```
int base = 10, altura = 5, lado = 4; String nombre = "nom";  
areaRectangulo(base, altura); //invocamos utilizando variables base y altura  
areaCuadrado(lado); //invocamos utilizando la variable lado  
tablonDeAnuncios(2*1); //invocamos utilizando un valor numérico como parámetro  
saludo("nombre"); //invocamos utilizando la palabra nombre como parámetro  
saludo(nombre); //invocamos utilizando la variable nombre como parámetro
```

Uso de métodos - Invocaciones

- Un parámetro puede ser cualquier expresión que se evalúe a un tipo de datos compatible con el del correspondiente parámetro formal de la cabecera del método.
- La llamada a un método se evalúa al resultado que éste devuelve. **Se puede invocar a un método y utilizar el retorno en cualquier expresión que tenga un tipo compatible.**

```
static double areaCuadrado(double lado)
{
    return lado*lado;
}
public static void main (String args[])
{
    int a = 10;
    double r = areaCuadrado(a);
    System.out.println("El área de un cuadrado de lado "+a+" es "+r);
    System.out.println("El área de un cuadrado de lado 5 es "+areaCuadrado(5));
}
```

Ejercicios métodos en JAVA - Iniciación

Ejercicios métodos en JAVA - Iniciación

1. Implementa un método que, dado un nombre, muestre un saludo.
2. Implementa un método que, dado un número entero, calcule el cubo.
3. Implementa un método que, dados dos números, los multiplique.
4. Utiliza el método anterior para que, dado un número, calcule su tabla de multiplicar del 1 al 10.
5. Implementa un método que dado un número diga si éste es par.
6. Implementa un método para mostrar un menú de N opciones, con su número correspondiente, la última opción será para salir.

Paso por valor en JAVA

Paso por valor en JAVA

En JAVA los punteros no existen de la misma forma que como en otros lenguajes como C o C++. En JAVA los punteros son vistos simplemente como variables, no es posible obtener un puntero a la dirección de memoria de una variable. Únicamente se puede copiar el valor de una variable en otra variable. A este proceso se le denomina '**paso por valor**'.

En JAVA los parámetros de los métodos siempre se pasan por valor, es decir, pasamos una copia al método, la variable original nunca se modifica.

Esto hace a JAVA más sencillo y menos propenso a errores. Aunque por otro lado, la manipulación de punteros en lenguajes como C y C++ es útil en la programación de sistemas operativos, debido a que es un modelo que está más cerca de la máquina y permite la manipulación de la memoria de una forma mucho más avanzada.

Paso por valor en JAVA

En JAVA todos los parámetros se pasan por valor. El paso por valor de los argumentos de un método en JAVA tiene varias consecuencias. Si al método le pasamos una variable como parámetro, al método le llega una copia del valor, y al modificar ese valor sobre el parámetro, no modificaremos el valor de la variable usada para invocar al método.

```
public class Ejemplo {  
    public static void main (String args[])  
    {  
        int num = 5;  
        int numDoble = doble(num);  
    } //numDoble vale 10, pero num vale 5  
    static int doble(int a)  
    {  
        a = a*2;  
        return a;  
    }  
}
```

```
public class Ejemplo {  
    public static void main (String args[])  
    {  
        int num = 5;  
        int doble = doble(num);  
    } //doble vale 10, pero num vale 5  
    static int doble(int num)  
    {  
        num = num*2;  
        return num;  
    }  
}
```

El método main

- El **main** es el método que aparecerá en la mayoría de las clases de JAVA que veremos a lo largo de las próximas unidades.
- Cuando se pide la ejecución de una clase, **main es el método que la JVM** invoca por defecto
- Cabecera: **public static void main (String[] args)**
- Su tipo de retorno es **void**, es decir, no retorna ningún valor.
- Su único parámetro es un **Array de Strings**. Esta lista se recibe al ejecutar la aplicación con el comando java por consola, lo veremos en detalle en la siguiente unidad:

```
public class Ejemplo {  
    public static void main (String args[])  
    {  
        for(int i = 0; i<args.length; i++) {  
            System.out.println("Parámetro" + (i+1) + ":" + args[i]);  
        } //Mostrará cada uno de los parámetros en una línea distinta  
    }  
}
```

Variables locales y globales en JAVA

Variables locales y globales

- **En el cuerpo de un método** se pueden declarar variables, que se denominan **variables locales**.
- Las variables locales **sólo se pueden usar en el método en el que se declaran**, por lo que su ámbito(scope), es dicho método.
- Los parámetros de un método se pueden considerar variables locales del mismo.
- En una clase, se pueden declarar variables globales en cualquier punto fuera del cuerpo de un método, dentro del **'class'**. De momento, para crear estas variables globales utilizaremos la directiva **'static'**. Más adelante, en OO, ya veremos en detalle el significado real del **'static'** y todas sus implicaciones.
- Los métodos de una clase pueden usar las variables globales de ésta.
- Una variable local puede tener el mismo nombre que una variable global. En este caso la variable local siempre sobrescribe a la variable global.

Variables locales y globales

Una variable local (o un parámetro) puede sobrescribir a una variable global:

```
public class Ejemplo {
    static int x; //es una variable global
    static int cubo (int z)
    {
        int x; //variable local del método cubo
        x = z*z*z; //x es la variable local, no la global
        return x;
    }
    public static void main (String args[])
    {
        int p; //variable local del método main
        p = 10; //correcto: p es local en este ámbito
        z = 100; //error: z no está en este ámbito
        x = cubo(p); //correcto: x es global, x vale 1000
    }
}
```


Variables locales y globales

Una variable local (o un parámetro) puede sobrescribir a una variable global:

```
public class Ejemplo {  
    static int num = 5;  
    static int cubo (int num)  
    {  
        int x = num*num*num;  
        return x;  
    }  
    public static void main (String args[])  
    {  
        num = cubo(num-1);  
        System.out.println(num);  
    }  
}
```

Variables locales y globales

Una variable local (o un parámetro) puede sobrescribir a una variable global:

```
public class Ejemplo {  
    static int num = 5; //es una variable global  
    static int cubo (int num)  
    {  
        int x = num*num*num; //num es la variable local del método cubo  
        return x;  
    }  
    public static void main (String args[])  
    {  
        num = cubo(num-1); // se utiliza la variable global → en num se guarda 64  
        System.out.println(num); // se imprime 64(variable local)  
    }  
}
```

Variables locales y globales

Una variable local (o un parámetro) puede sobrescribir a una variable global:

```
public class Ejemplo {  
    static int num = 5;  
    static int cubo (int num)  
    {  
        int x = num*num*num;  
        return x;  
    }  
    public static void main (String args[])  
    {  
        int num = 10;  
        num = cubo(num);  
        System.out.println(num);  
    }  
}
```

Variables locales y globales

Una variable local (o un parámetro) puede sobrescribir a una variable global:

```
public class Ejemplo {
    static int num = 5; //es una variable global
    static int cubo (int num)
    {
        int x = num*num*num; //num es la variable local del método cubo
        return x;
    }
    public static void main (String args[])
    {
        int num = 10; // es una variable local, sobrescribe la global
        num = cubo(num); // se utiliza la variable local, en num se guarda 1000
        System.out.println(num); // se imprime 1000(variable local)
    }
}
```

Sobrecarga de métodos en JAVA

Sobrecarga de métodos

En JAVA podemos tener métodos con el mismo nombre, pero se debe tener en cuenta que:

- Si el método tiene un parámetro, deben ser de distintos tipos en cada método.
- Si tiene más de un parámetro, al menos uno debe ser distinto.
- Si no tiene parámetros, no podemos tener métodos con el mismo nombre.

```
static String saludar()  
{  
    return "Hola, cómo estás?";  
}  
static String saludar(String nombre)  
{  
    return "Hola " + nombre + ", cómo estás?";  
}  
static String saludar(String nombre, String ciudad)  
{  
    return "Hola" + nombre + ", qué tal por "+ciudad+"?";  
}
```

Sobrecarga de métodos

En JAVA podemos tener métodos con el mismo nombre, pero se debe tener en cuenta que:

- Aunque el tipo de retorno sea distinto, si los parámetros son iguales, se producirá un error de compilación. Esto se debe a que JAVA realmente no sabe el método que debe utilizar. Veamos un ejemplo:

```
static String saludar()  
{  
    return "Hola, cómo estás?";  
}  
static void saludar()  
{  
    System.out.println("Hola, cómo estás?");  
}  
public static void main (String args[])  
{  
    saludar(); //ERROR: JAVA no sabe el método que debe utilizar  
}
```

Sobrecarga de métodos

En JAVA podemos tener métodos con el mismo nombre, pero se debe tener en cuenta que:

- Aunque el tipo de retorno sea distinto, si los parámetros son iguales, se producirá un error de compilación. Esto se debe a que JAVA realmente no sabe el método que debe utilizar. Veamos un ejemplo:

```
static int suma(int a, int b)
{
    return a + b;
}
static void suma(int a, int b)
{
    System.out.println(a + b);
}
public static void main (String args[])
{
    suma(2,4); //ERROR: JAVA no sabe el método que debe utilizar
}
```


La pila de llamadas en JAVA

La pila de llamadas

- Para implementar la ejecución de la llamada a un método **la JVM utiliza la pila de llamadas.**
- Esta pila de llamadas se aloja en la memoria RAM y está compuesta por los denominados registros de activación. Cada uno de estos registros de activación:
 - Tiene asociada una llamada dada a un método, la del main incluida.
 - Contiene las variables locales del método y sus parámetros, junto con el valor que se les ha asignado a éstos últimos al realizar la llamada (parámetros actuales)
- Un método solo puede usar:
 - La información contenida en su correspondiente registro de activación de la pila de llamadas.
 - Los datos contenidos en las variables globales.

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
num = 3 sum = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
n = 3 i = 0 s = 0	proceso
num = 3 sum = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
n = 3 i = 0 s = 0 + 0	proceso
num = 3 sum = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
n = 3 i = 1 s = 0	proceso
num = 3 sum = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
n = 3 i = 1 s = 0 + 1	proceso
num = 3 sum = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
n = 3 i = 2 s = 1	proceso
num = 3 sum = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
n = 3 i = 2 s = 1 + 2	proceso
num = 3 sum = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
n = 3 i = 3 s = 3	proceso
num = 3 sum = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso (int n)  
    {  
        int i, s = 0;  
        for(int i = 0; i < n; i++) s = s+i;  
        return s;  
    }  
    public static void main (String args[])  
    {  
        int num, sum;  
        num = 3;  
        sum = proceso(num);  
    }  
}
```

PILA DE LLAMADAS	
num = 3 sum = 3	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {
    static int proceso (int n)
    {
        int i, s = 0;
        for(int i = 0; i < n; i++) s = s+i;
        return s;
    }
    public static void main (String args[])
    {
        int num, sum;
        num = 3;
        sum = proceso(num);
    }
}
```

[illegible]

Pila de llamadas con 3 métodos: Ejemplo 1

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n1 = 3 p1 = 1 i = 0	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 0 p2 = 1 i = 0	proceso2
n1 = 3 p1 = 1 + ? i = 0	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n1 = 3 p1 = 1 + 1 i = 0	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n1 = 3 p1 = 2 i = 0	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n1 = 3 p1 = 2 i = 1	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 1 p2 = 1 i = 0	proceso2
n1 = 3 p1 = 2 + ? i = 1	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 1 p2 = 1 + 1 i = 0	proceso2
n1 = 3 p1 = 2 + ? i = 1	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 1 p2 = 2 i = 1	proceso2
n1 = 3 p1 = 2 + ? i = 1	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n1 = 3 p1 = 2 + 2 i = 1	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n1 = 3 p1 = 4 i = 2	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 2 p2 = 1 i = 0	proceso2
n1 = 3 p1 = 4 + ? i = 2	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 2 p2 = 1 + 2 i = 0	proceso2
n1 = 3 p1 = 4 + ? i = 2	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 2 p2 = 3 i = 1	proceso2
n1 = 3 p1 = 4 + ? i = 2	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 2 p2 = 3 + 2 i = 1	proceso2
n1 = 3 p1 = 4 + ? i = 2	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n2 = 2 p2 = 5 i = 2	proceso2
n1 = 3 p1 = 4 + ? i = 2	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n1 = 3 p1 = 4 + 5 i = 2	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
n1 = 3 p1 = 9 i = 3	proceso1
num = 3 res = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	
num = 3 res = 9	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int n1)  
    { int p1 = 1;  
      for(int i = 0; i < n1; i++) p1 += proceso2(i);  
      return p1;  
    }  
    static int proceso2(int n2)  
    {  
        int p2 = 1;  
        for(int i = 0; i < n2; i++) p2 += n2;  
        return p2;  
    }  
    public static void main (String args[])  
    {  
        int num = 3;  
        int res = proceso1(num);  
    }  
}
```

PILA DE LLAMADAS	

Pila de llamadas con 3 métodos: Ejemplo 2

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS

x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
        int y = 1;  
        for(int j = 0; j < a+b; j++) y++;  
        return y;  
    }  
    public static void main (String args[])  
    {  
        int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS

a = 4 x = 0 i = 0	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
        int y = 1;  
        for(int j = 0; j < a+b; j++) y++;  
        return y;  
    }  
    public static void main (String args[])  
    {  
        int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 4 x = 0 + ? i = 0	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 0 b = 1 y = 1 j = 0	proceso2
a = 4 x = 0 + ? i = 0	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 0 b = 1 y = 2 j = 0	proceso2
a = 4 x = 0 + ? i = 0	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 0 b = 1 y = 2 j = 1	proceso2
a = 4 x = 0 + ? i = 0	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 4 x = 0 + 2 i = 0	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
        int y = 1;  
        for(int j = 0; j < a+b; j++) y++;  
        return y;  
    }  
    public static void main (String args[])  
    {  
        int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 4 x = 2 i = 2	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 2 b = 3 y = 1 j = 0	proceso2
a = 4 x = 2 + ? i = 2	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 2 b = 3 y = 6 j = 5	proceso2
a = 4 x = 2 + ? i = 2	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
        int y = 1;  
        for(int j = 0; j < a+b; j++) y++;  
        return y;  
    }  
    public static void main (String args[])  
    {  
        int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 4 x = 2 + 6 i = 2	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
        int y = 1;  
        for(int j = 0; j < a+b; j++) y++;  
        return y;  
    }  
    public static void main (String args[])  
    {  
        int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 4 x = 8 i = 4	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 4 b = 5 y = 1 j = 0	proceso2
a = 4 x = 8 + ? i = 4	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    {  
        int x = 0;  
        for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
        return x;  
    }  
    static int proceso2(int a, int b)  
    {  
        int y = 1;  
        for(int j = 0; j < a+b; j++) y++;  
        return y;  
    }  
    public static void main (String args[])  
    {  
        int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 4 b = 5 y = 10 j = 9	proceso2
a = 4 x = 8 + ? i = 4	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
        int y = 1;  
        for(int j = 0; j < a+b; j++) y++;  
        return y;  
    }  
    public static void main (String args[])  
    {  
        int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS

a = 4 x = 8 + 10 i = 4	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
        int y = 1;  
        for(int j = 0; j < a+b; j++) y++;  
        return y;  
    }  
    public static void main (String args[])  
    {  
        int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
a = 4 x = 18 i = 6	proceso1
x = ?	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS	
x = 18	main

La pila de llamadas

Ejemplo de funcionamiento de la pila de llamadas:

```
public class Ejemplo {  
    static int proceso1(int a)  
    { int x = 0;  
      for(int i = 0; i <= a; i+=2) x += proceso2(i,i+1);  
      return x;  
    }  
    static int proceso2(int a, int b)  
    {  
      int y = 1;  
      for(int j = 0; j < a+b; j++) y++;  
      return y;  
    }  
    public static void main (String args[])  
    {  
      int x = proceso1(4);  
    }  
}
```

PILA DE LLAMADAS

Ejercicios métodos en JAVA

Ejercicios métodos

En estos ejercicios vamos a reutilizar código que realizamos en la unidad anterior para crear distintos tipos de figuras:

1. Implementa un método que a partir del lado, dibuje en cuadrado formado por asteriscos. Crea un método adicional para dibujar dicho cuadrado sin relleno.
2. Implementa un método que a partir de la altura, dibuje un triángulo rectángulo formado por asteriscos. Crea un método adicional para dibujar dicho triángulo sin relleno.
3. Crear un menú con 5 opciones, una por cada tipo de figura, y una opción para salir. Se deberá elegir una opción y realizar la acción correspondiente. El programa no terminará hasta que seleccionemos la opción SALIR del menú principal.

Recursividad en JAVA

Métodos recursivos

- Se denomina método recursivo a cualquier método que **tiene la capacidad de llamarse a sí mismo**.
- Un algoritmo es recursivo si obtiene la solución de un problema en base a los resultados que él mismo proporciona para casos más sencillos del mismo problema. Es decir, es un método que se invoca a sí mismo:
- El **caso base** de un problema es el que se puede resolver trivialmente.
- El **caso general** de un problema es el que resuelve recursivamente, empleando el mismo algoritmo para casos más sencillos
- Las funciones matemáticas recursivas se implementan fácilmente usando algoritmos recursivos, como por ejemplo la función factorial:

$$n! = \left\{ \begin{array}{ll} 1 & \text{si } n = 0 \\ n*(n-1)! & \text{si } n > 0 \end{array} \right\}$$

Recursión

- El caso base de un método recursivo permite detener la recursión (no se hace ninguna llamada en él), por lo que resulta imprescindible.
- De hecho, en las llamadas recursivas del caso general alguno de los parámetros debe modificar su valor para que finalmente, tras un número finito de llamadas, se alcance el caso base.
- La recursión hace un uso intensivo de la pila de llamadas.
- Casi siempre, los métodos recursivos permiten expresar la resolución de ciertos problemas complejos de forma mucho más sencilla que sus equivalentes iterativos.
- El cuerpo de un método recursivo debe tener una instrucción condicional.

```
static int factorial(int n) {  
    if(n == 0) return 1;           //caso base  
    else return n*factorial(n-1);  //caso general  
}
```

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {

    int r;
    if(n == 0) r = 1;
    else r = n*factorial(n-1);
    return r;
}

public static void main(String args[]) {

    int f = factorial(4);
    System.out.println(f);
}
```

[illegible]

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS

n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS

n = 3 r = 3 * ?	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS

n = 2 r = 2 * ?	fact(2)
n = 3 r = 3 * ?	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 1 r = 1 * ?	fact(1)
n = 2 r = 2 * ?	fact(2)
n = 3 r = 3 * ?	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 0 r = 1	fact(0)
n = 1 r = 1 * ?	fact(1)
n = 2 r = 2 * ?	fact(2)
n = 3 r = 3 * ?	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 0 r = 1	fact(0)
n = 1 r = 1*1	fact(1)
n = 2 r = 2 * ?	fact(2)
n = 3 r = 3 * ?	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 1 r = 1*1	fact(1)
n = 2 r = 2 * ?	fact(2)
n = 3 r = 3 * ?	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 1 r = 1*1	fact(1)
n = 2 r = 2*1	fact(2)
n = 3 r = 3 * ?	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS

n = 2 r = 2*1	fact(2)
n = 3 r = 3 * ?	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 2 r = 2*1	fact(2)
n = 3 r = 3*2	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS

n = 3 r = 3*2	fact(3)
n = 4 r = 4 * ?	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 3 r = 3*2	fact(3)
n = 4 r = 4*6	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 4 r = 4*6	fact(4)
f = ?	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
n = 4 r = 4*6	fact(4)
f = 24	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

PILA DE LLAMADAS	
f = 24	main

Recursión

Ejecución método recursivo usando la pila de llamadas:

```
static int factorial(int n) {  
  
    int r;  
    if(n == 0) r = 1;  
    else r = n*factorial(n-1);  
    return r;  
}  
  
public static void main(String args[]) {  
  
    int f = factorial(4);  
    System.out.println(f);  
}
```

[illegible]

Triángulo recursivo en JAVA: Solución 1

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void tri(int contB, int contA, int n)
{
    if (contB < contA) {
        System.out.print("* ");
        tri(contB + 1, contA, n);
    } else {
        System.out.println();
        if (contA < n)
            tri(0, contA + 1, n);
    }
}

public static void main(String[] args)
{
    int altura = 3;
    tri(0, 0, altura);
}
```

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

PILA DE LLAMADAS

tri(3,3,3)

tri(2,3,3)

tri(1,3,3)

tri(0,3,3)

tri(2,2,3)

tri(1,2,3)

tri(0,2,3)

tri(1,1,3)

tri(0,1,3)

tri(0,0,3)

main

Triángulo recursivo en JAVA: Solución 2

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

PILA DE LLAMADAS

main

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

PILA DE LLAMADAS

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

PILA DE LLAMADAS

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

PILA DE LLAMADAS

triangulo(1)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

PILA DE LLAMADAS

triangulo(0)

triangulo(1)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

PILA DE LLAMADAS

triangulo(1)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*

PILA DE LLAMADAS

filaTriangulo(1)

triangulo(1)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*

PILA DE LLAMADAS

filaTriangulo(0)

filaTriangulo(1)

triangulo(1)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*

PILA DE LLAMADAS

filaTriangulo(1)

triangulo(1)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*

PILA DE LLAMADAS

triangulo(1)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*

PILA DE LLAMADAS

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*

*

PILA DE LLAMADAS

filaTriangulo(2)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*

* *

PILA DE LLAMADAS

filaTriangulo(1)

filaTriangulo(2)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

```
*
* *
```

PILA DE LLAMADAS

filaTriangulo(0)

filaTriangulo(1)

filaTriangulo(2)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*
* *

PILA DE LLAMADAS

filaTriangulo(1)

filaTriangulo(2)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

*

* *

PILA DE LLAMADAS

filaTriangulo(2)

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```



*
* *

PILA DE LLAMADAS

triangulo(2)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

PILA DE LLAMADAS

triangulo(3)

main

*

* *

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

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

PILA DE LLAMADAS

filaTriangulo(3)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

```
*
*  *
*  *
```

PILA DE LLAMADAS

filaTriangulo(2)

filaTriangulo(3)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

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

PILA DE LLAMADAS

filaTriangulo(1)

filaTriangulo(2)

filaTriangulo(3)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

```

*
*  *
*  *  *
```

PILA DE LLAMADAS

filaTriangulo(0)
filaTriangulo(1)
filaTriangulo(2)
filaTriangulo(3)
triangulo(3)
main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

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

PILA DE LLAMADAS

filaTriangulo(1)

filaTriangulo(2)

filaTriangulo(3)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

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

PILA DE LLAMADAS

filaTriangulo(2)

filaTriangulo(3)

triangulo(3)

main

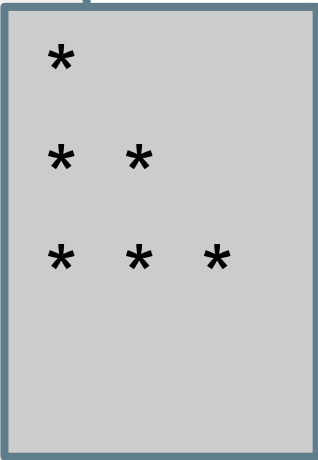
Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```



```

*
```

```

*  *
```

```

*  *  *
```

PILA DE LLAMADAS

filaTriangulo(3)

triangulo(3)

main

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

PILA DE LLAMADAS

triangulo(3)

main

*

* *

* * *

Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

PILA DE LLAMADAS

main

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

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


Triángulo recursivo en JAVA

Ejecución figura recursiva usando la pila de llamadas:

PILA DE LLAMADAS

```
static void filaTriangulo(int n)
{
    if(n>0) {
        System.out.print("* ");
        filaTriangulo(n - 1);
    }
    else System.out.println();
}

static void triangulo(int n)
{
    if(n>0) {
        triangulo(n - 1);
        filaTriangulo(n);
    }
}

public static void main(String[] args)
{
    triangulo(3);
}
```

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

Ejercicios recursividad en JAVA

Ejercicios recursividad en JAVA

1. Crea un método que obtenga la cantidad de dígitos de un número N mayor que cero. Se debe pasar como parámetro el número N.
2. Crea un método que obtenga el resultado de elevar un número a otro. Ambos números se deben pasar como parámetros. Los números deben ser positivos.
3. Crea un método que dado un número positivo, lo imprima invertido por pantalla.
4. Crea un método que compruebe si un número es binario. Un número binario está formado únicamente por ceros y unos.
5. Crea un método que obtenga el número binario de un número N pasado como parámetro.
6. Crea un método que compruebe si una palabra está ordenada alfabéticamente.
7. Crea un método que obtenga la suma de los números naturales desde 1 hasta N. Se debe pasar como parámetro el número N, debe ser mayor que cero. Se debe imprimir toda la cadena por consola. Por ejemplo, para $N=4 \rightarrow (1+2+3+4 = 10)$.

Bibliografía:

Allen Weiss, M. (2007). *Estructuras de datos en JAVA*. Madrid: Pearson

Froufe Quintas, A. (2002). *JAVA 2: Manual de usuario y tutorial*. Madrid: RA-MA

J. Barnes, D. *Programación orientada a objetos en JAVA*. Madrid: Pearson

Desing Patterns. Elements of Reusable. OO Software

JAVA Limpio. Pello Altadi. Eugenia Pérez

Apuntes de Programación de Anna Sanchis Perales

Apuntes de Programación de Lionel Tarazón Alcocer



Ilustraciones:

<https://pixabay.com/>

<https://freepik.es/>

<https://lottiefiles.com/>

Preguntas

