Lenguaje Java

Recursividad

José Manuel Pérez Lobato

Recursividad

- Definición de recursividad:
  Véase recursividad.

- Una definición es recursiva cuando lo que se define forma parte de la definición.
- Una función, método u operación es recursivo cuando existe una llamada así misma dentro de su código. La llamada puede ser directa:

```
metodoA() {
 metodoA();
}
```

o indirecta:

```
metodoA() {
    metodoB();
}
```

```
metodoB(){
        metodoA();
    }
```

Recursividad

- La recursividad se puede ver como un bucle en el que hay una parte de instrucciones que se repiten y una condición que define cuando se debe de terminar la repetición
- Un método recursivo consta de 2 caminos de ejecución
  - Uno que invoca al propio método (por eso es recursivo)
  - Otro que no invoca al propio método y sirve para terminar la recursividad.
- Por ejemplo en la definición de factorial tenemos:

$$n! = \begin{cases} n * (n-1)! & \text{Si } n>0 \\ 1 & \text{Si } n==0 \end{cases}$$

Ejemplo Recursividad

```
import java.io.*;

class Factorial {
int factorial (int n) {
  if (n >0) return (n* factorial (n-1));
  else return (1);
}
public static void main (String arg[]) throws IOException{
  Factorial f=new Factorial();
  Teclado t=new Teclado();
  int n;

 System.out.println("Dar un número (No mayor de 16:");
  n= t.leerInt();
  System.out.println ("factorial de "+n+" es "+ f.factorial(n));
}
}
```

Ejemplo Recursividad

```
class RecurEj {
int atrpos=1;
void procesarnums() throws IOException  {
  Teclado t= new Teclado ();
  int n,pos;
  System.out.println ("Dar un número:");
  n=t.leerInt();
  pos=atrpos;
  atrpos++;
  if (n != 0) {
   procesarnums();
   System.out.println (n);
  }
}
public static void main (String arg[]) throws IOException{
  RecurEj f=new RecurEj();
  f.procesarnums();
  System.out.println ("Fin");
}
}
```

¿Qué hace este programa?

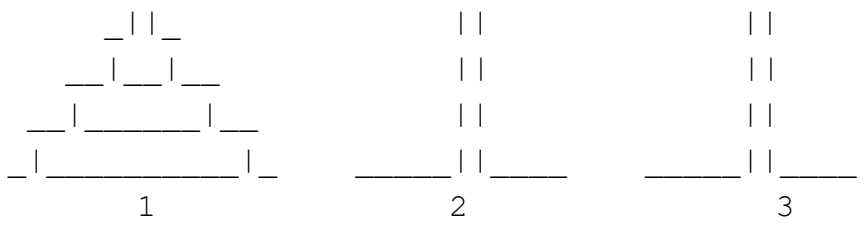
Ejercicios Recursividad

- Modificar el programa anterior para que muestre los números pares leídos de teclado (hasta dar un 0), en orden inverso al de lectura indicando la posición en la que se leyeron y el total de números leídos
- Realizar un método que calcule el término n-esimo de la serie de Fibonacci

  0, 1, 1, 2, 3, 5, 8, 13…….
- Realizar un programa que indique como colocar las Torres de Hanoi

  El problema de las torres de Hanoi consiste en mover una serie de fichas que se encuentran en una base o soporte a otra base distinta apoyándose en una base auxiliar. Todas las fichas tienen una tamaño distinto de manera que una ficha se puede apoyar en otra más grande pero nunca en otra más pequeña.

Situación inicial de las torres con tres fichas :



Situación final de las torres :