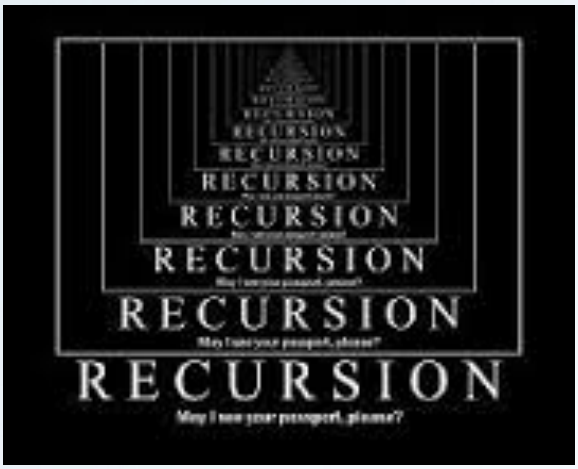
UNIVERSIDAD
DE LA FRONTERA

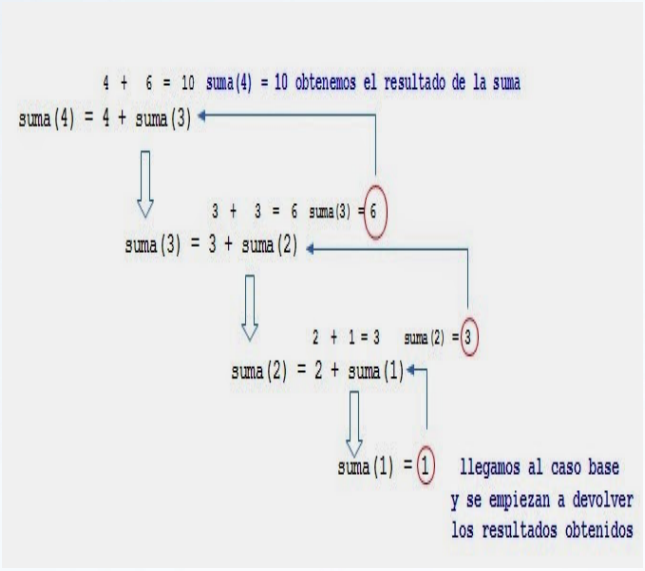
Introducción a la Programación

Taller 10. Recursividad

Octubre 2016

Departamento de
Computación e Informática





La recursividad es una técnica potente de programación que puede utilizarse en lugar de la iteración para resolver determinados tipos de problemas.

Un método es recursivo cuando entre sus instrucciones **se encuentra una llamada a sí mismo**.

La solución iterativa es fácil de entender. Utiliza una variable para acumular los productos y obtener la solución. En la solución recursiva se realizan llamadas al propio método con valores de n cada vez más pequeños para resolver el problema.

Cada vez que se produce una nueva llamada al método se crean en memoria de nuevo las variables y comienza la ejecución del nuevo método.

Para entender el funcionamiento de la recursividad, podemos pensar que cada llamada supone hacerlo a un método diferente, copia del original, que se ejecuta y devuelve el resultado a quien lo llamó.

Este taller permite realizar práctica de programación en JAVA con BlueJ, en particular se trabaja con la técnica de programación de recursividad.

Actividad 1, “Ejemplo Básico de Recursividad”

En esta actividad crearemos un programa básico para ilustrar la técnica de programación recursiva.

Paso 1.1

Copiar el siguiente código de ejemplo y ejecute para ver el funcionamiento. Como se puede verificar se calcula el factorial de un número con el uso de un ciclo for.

```java
import java.util.Scanner; // define otro paquete de libreria para usar el objeto scanner
public class Principal
{
    public Principal()
    {
    }

    public  void Main()
    {
    Scanner teclado = new Scanner(System.in); // define un nuevo objeto para ingreso por teclado en CONSOLA
    System.out.println("Ingrese un Número " +"y luego presione Enter: ");
    int numero = teclado.nextInt(); // solicita ingresar un numero por consola, solo debe ser número
    System.out.println ("El Factorial del Número es : "+this.factorial_iteracion(numero));    // usando la clase factorial con su
}

// Método Java no recursivo para calcular el factorial de un número
public double factorial_iteracion(int n){
    double fact=1;
    int i;
    if (n==0)
        fact=1;
    else
        for(i=1;i<=n;i++){
            fact=fact*i;
            System.out.println ("Ciclo For, iteración: "+ i);
        }
return fact;
}
}
```

Paso 1.2

Continuar con la creación de una clase con un método que permite el cálculo usado recursividad. Copie este código en una nueva clase llamada “Factorial”.

```java
//Calcular el factorial en modo recursivo
public class Factorial {
    public double factorial_recursivo(int n){
        System.out.println ("Llamada a factorial_recursivo con n= : "+ n);
        if (n==0)
            return 1;
        else
            return n*(factorial_recursivo(n-1));
}

}
```

Paso 1.3

Para terminar este ejemplo, se debe agregar el siguiente código al método ”Main” de la clase ”Principal”, que nos permite definir un nuevo objeto de tipo Factorial y utilizar su método de cálculo recursivo.

```java
import java.util.Scanner; // define otro paquete de libreria para usar el objeto scanner
public class Principal
{
    public Principal()
    {
    }

    public  void Main()
    {
    Scanner teclado = new Scanner(System.in); // define un nuevo objeto para ingreso por teclado en CONSOLA
    System.out.println("Ingrese un Número " +"y luego presione Enter: ");
    int numero = teclado.nextInt(); // solicita ingresar un numero por consola, solo debe ser número
    System.out.println ("El Factorial del Número es : "+this.factorial_iteracion(numero));    // usando la
    // se define un nuevo objeto de la clase factorial para usar su método factorial_recursivo
    Factorial miFactorial = new Factorial();
    System.out.println ("El Factorial del Número es : "+ miFactorial.factorial_recursivo(numero));
}
```

Nota

En el ejemplo del factorial en cada llamada recursiva se utiliza n-1
return n * (factorial(n-1)); por lo que en cada llamada el valor de n se acerca más a 0 , terminando la llamada recursiva.
La recursividad es especialmente apropiada cuando el problema tienen una clara definición recursiva.
Cuando el problema se pueda definir mejor de una forma recursiva que iterativa lo resolveremos utilizando recursividad (ejemplo: búsquedas).

Actividad 2, “Ejemplo 2 de Recursividad”

En esta actividad crearemos un nuevo programa para ilustrar la técnica de programación recursiva.

Calcular el cociente de dos números enteros.

```
//Calcular el factorial en modo recursivo
public class Factorial {
   public double factorial_recursivo(int n){
    System.out.println ("Llamada a factorial_recursivo con n= : "+ n);
    if (n==0)
       return 1;
    else
       return n*(factorial_recursivo(n-1));
}
    public  int cociente(int a, int b) {
      System.out.println ("Llamada a cociente con = : "+ a +" y"+b);
       if (a < b)
           return 0;
       else
           return 1 + cociente(a - b, b);
    }
}
```

La solución recursiva del problema se plantea de la siguiente forma:

Caso Base: Si el dividendo es menor que el divisor el cociente es cero.

Si no, se restan el dividendo y el divisor. Al resultado se le vuelve a restar el divisor. Esta resta se repite mientras se pueda realizar, o sea, mientras el resultado sea mayor o igual que el divisor.

El número de veces que se ha hecho la resta **es el cociente**. Con un ejemplo lo veremos más claro. Por ejemplo, para dividir 10 entre 3 haríamos:

10 - 3 = 7
7 - 3 = 4
4 - 3 = 1

No se puede seguir restando ya que el último valor obtenido (1) es menor que el divisor. Como se han realizado 3 restas el cociente es 3.

Paso 2.1

Copiar código destacado de la figura y luego agregar la siguientes línea en método “Main” de la clase “Principal” para verificar el funcionamiento.

System.out.println ("El cuociente de 10/3 es : "+ miFactorial.cociente(10,3));

Actividad 3, “Ejemplo de Recursividad Gráfica”

En esta actividad revisaremos un nuevo ejemplo de uso de recursividad asociada a la representación gráfica de un circulo. Reviese el funcionamiento y consulte si hay alguna instrucción de código que no se entienda. Puede cambiar parámetros de ejecución como posición, tamaño y color.

Descargar el código desde el campus virtual, taller 10, apuntes y referencias, ejemplo del circulo recursivo.

Actividad 4, “Ejercicio de Recursividad”

Trabaje en resolver el siguiente ejercicio:

Un algoritmo de búsqueda es aquel que está diseñado para localizar un elemento con ciertas propiedades dentro de una estructura de datos; por ejemplo, ubicar el registro correspondiente a cierta persona en una base de datos, o el mejor movimiento en una partida de ajedrez.

La variante más simple del problema es la búsqueda de un número en un vector.

Desarrolle un programa en Java que complete un array de 100 números enteros aleatorios, pida y valide una número entero entre 1 y 100 y luego el programa busca la primera ocurrencia y entrega un mensaje ”Número encontrado en la posición: “ (indicar el número índice de la posición).
En caso contrario entrega un mensaje diciendo que el número no fue encontrado.
Debe utilizar una clase buscar y un método de búsqueda recursiva.

Mostrar al profesor una vez terminado

UNIVERSIDAD
DE LA FRONTERA

Curso de Introducción a la Programación

Taller 10. Recursividad

UNIVERSIDAD DE LA FRONTERA
FACULTAD DE INGENIERÍA Y CIENCIAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INFORMÁTICA

Avda Francisco Salazar 01145
Temuco – Chile / casilla 54-D
Fono (56) 45 2325000 /2744219

dci.ufro.cl

DCI
DEPARTAMENTO DE COMPUTACIÓN E INFORMÁTICA

Departamento de
Computación e Informática