Informe de Práctica de Recursividad

Cálculo del Bono de Desarrollo Humano (BDH)

1. Introducción

La recursión es una técnica de programación que permite resolver problemas complejos dividiéndolos en subproblemas más pequeños del mismo tipo. Para que un algoritmo recursivo sea correcto, debe contar con un caso base alcanzable y un caso recursivo que garantice el avance hacia dicho caso base. Un diseño incorrecto puede provocar errores críticos como el desbordamiento de pila.

---

2. Punto 2: Traza Manual del Árbol de Recursión

2.1. Caso de estudio

Familia con tres dependientes ubicada en zona rural. La llamada inicial es:

```
CalcularRecursivo(3, true)
```

Cada llamada recursiva reduce el número de dependientes hasta llegar al caso base.

2.2. Árbol de recursión

La Figura 1 muestra el árbol de recursión completo del método recursivo, donde cada nodo representa una llamada al método y las hojas corresponden al caso base.

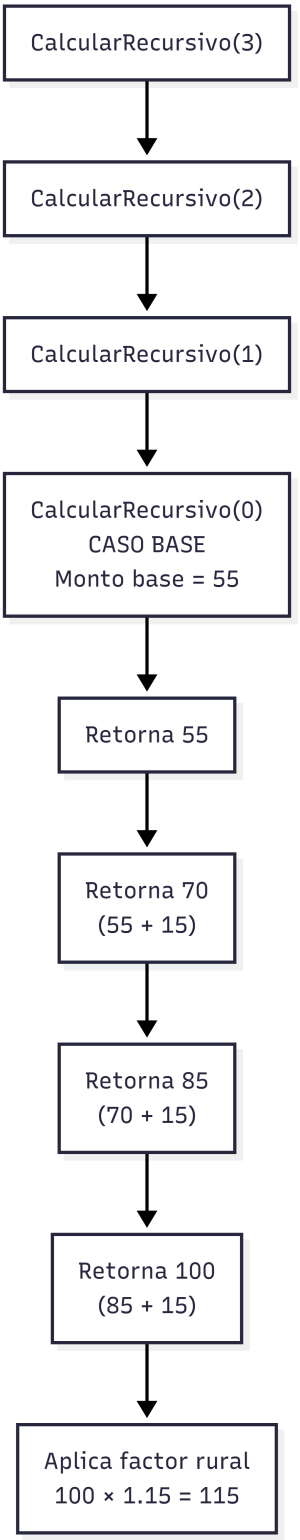

Figura 1: Árbol de recursión para el cálculo del BDH con 3 dependientes

2.3. Orden de retorno

El retorno de valores ocurre desde la llamada más profunda hacia la llamada inicial:

- `CalcularRecursivo(0)` retorna el monto base.
- `CalcularRecursivo(1)` suma un dependiente.
- `CalcularRecursivo(2)` suma dos dependientes.
- `CalcularRecursivo(3)` retorna el monto total.

---

3. Punto 3: Provocación del Error de Recursión

3.1. Código con error intencional

Para analizar el comportamiento de la pila de llamadas, se modifica el algoritmo recursivo eliminando el progreso hacia el caso base, lo que provoca una recursión infinita.

```
1  public class CalcularRecursivo {
2
3      private static double MONTO_BASE = 55.0;
4      private static double MONTO_DEPENDIENTE = 15.0;
5      private static double FACTOR_RURAL = 1.15;
6
7      private static double CalcularRecursivo_bug(int dependientes,
           boolean esRural) {
8
9          // Caso base inalcanzable
10         if (dependientes < -1) {
11             return esRural ? MONTO_BASE * FACTOR_RURAL : MONTO_BASE;
12         }
13
14         // Error: no se reduce el valor de dependientes
15         return CalcularRecursivo_bug(dependientes, esRural) +
               MONTO_DEPENDIENTE;
16     }
17
18     public static void main(String[] args) {
19         System.out.println(CalcularRecursivo_bug(1, false));
20     }
21 }
```

3.2. Resultado

Al ejecutar el programa, la máquina virtual de Java genera un error del tipo:

```
Exception in thread "main" java.lang.StackOverflowError
```

Este error indica que la pila de llamadas ha superado su límite máximo.

---

4. Reflexión Técnica y Social

El *StackOverflowError* ocurre cuando una función recursiva no alcanza su caso base, provocando que cada llamada genere un nuevo marco en la pila de ejecución. Al no existir una condición de parada efectiva, la pila crece de manera indefinida hasta agotar la memoria asignada.

En un sistema real del Ministerio de Inclusión Económica y Social, un error de este tipo podría causar la caída del servicio de cálculo del Bono de Desarrollo Humano. A nivel social, esto tendría un impacto directo en miles de familias en situación de vulnerabilidad, ya que retrasaría o impediría el acceso a recursos económicos esenciales para cubrir necesidades básicas.

---

5. Conclusiones

La práctica demuestra la importancia de diseñar algoritmos recursivos correctamente. Garantizar la convergencia hacia el caso base es esencial para evitar errores críticos de memoria y asegurar la estabilidad de sistemas que impactan directamente en la población.