PRÁCTICA 4. Recursión.

Un método recursivo es un método que, directa o indirectamente, se hace una llamada a sí mismo. La clave para que un método recursivo llegue a una solución es que se llama a sí mismo con instancias del método diferentes, más simples. Debe haber una instancia, caso base, que se resuelva sin recursión para que el algoritmo acabe.

Para el planteamiento recursivo de la solución de un problema, es necesario que se cumplan las cuatro reglas fundamentales de la recursión:

1. Caso Base.
2. Progreso.
3. Creencia funcionamiento correcto.
4. Interés compuesto.

La recursión es una potente herramienta de resolución de problemas.

Nota: Junto con esta práctica van dos archivos:

1. La excepción ElementoNoEncontrado.java. Esta excepción señalará el caso de un elemento no encontrado en la búsqueda. Deberá crear un nuevo paquete **com.mp.excepciones** e incluirla en él.

2. Juegos de prueba para varios ejercicios.

Ejercicios a realizar:

1) Diseñe e implemente los algoritmos recursivos e iterativos que implementen

   a) La suma de los n primeros enteros.

   b) El factorial de n.

   c) La sucesión de Fibonacci.

   Diseño: Para cada uno de los programas cree las clases Sumatoria con los métodos estáticos sumaIter, sumaRec y sumaGauss, Factorial con los métodos estáticos factorialIter y factorialRec, y Fibonacci con los métodos estáticos fibonacciIter y fibonacciRec.

   Nota: Incluir en el paquete: **com.mp.practica4.ejercicio1**

   Nota: Las pruebas de unidad se situaran en el paquete: **com.mp.practica4.ejercicio1.test**

2) El juego de las Torres de Hanoi. Las Torres de Hanoi es un juego cuyos elementos son tres varillas verticales y un número indeterminado de discos que determinarán la complejidad de la solución. No hay dos discos iguales, están colocados de mayor a menor en la primera varilla ascendentemente, y no se puede colocar ningún disco mayor sobre uno menor a él en ningún momento. El juego consiste en pasar todos los discos a la tercera varilla colocados de mayor a menor ascendentemente.

Diseño: A partir de la clase TorresHanoi, en una archivo .java que acompaña a la práctica, termine de completar los métodos para obtener una salida de consola similar a la imagen siguiente.

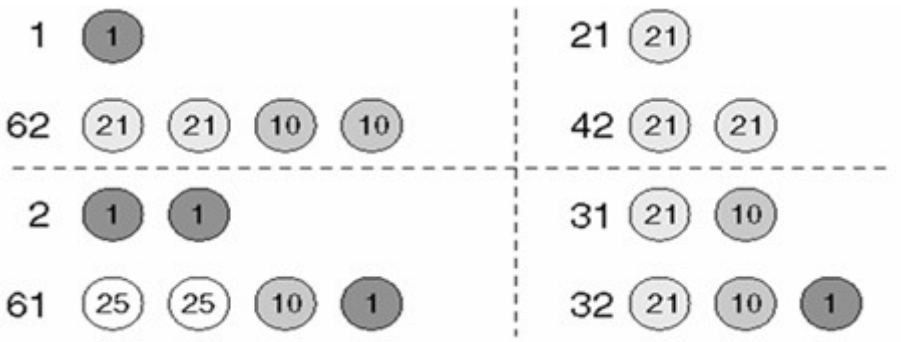
```
Movimiento: 1 Mover disco 1 desde 1 hasta 2
     ||      0     ||      0     ||      0
   **||**    2     ||      0     ||      0
  ***||***   3     ||      0     ||      0
 ****||****  4    *||*     1     ||      0
   Poste A       Poste B       Poste C

Movimiento: 2 Mover disco 2 desde 1 hasta 3
     ||      0     ||      0     ||      0
     ||      0     ||      0     ||      0
  ***||***   3     ||      0     ||      0
 ****||****  4    *||*     1    **||**    2
   Poste A       Poste B       Poste C
```

Nota: Incluir en el paquete: com.mp.practica4.ejercicio2

3) Resuelva el problema del Cambio de Monedas. "*Para una un juego de monedas arbitrario de U_1, U_2,...U_n (unidades). Una de las monedas es la unitaria. ¿Cuál es el mínimo número de monedas que se necesitan para devolver K unidades de cambio*?"

Una solución al problema se consigue utilizando un algoritmo recursivo que explore el espacio de soluciones posibles evaluando en cada caso el número de monedas necesarias. Obteniendo finalmente la combinación de monedas mínima. El planteamiento sería, para cada posible valor i calcular de forma independiente el número mínimo de monedas que se necesitan para reunir i y K-i unidades. Eligiendo el i que minimice la suma de ambos subproblemas. Esta solución no cumple la cuarta regla de la recursión y por tanto es muy ineficiente. Compruébelo.

Ejemplo de algunos de los subproblemas a resolver para cambiar 63 con el juego (1, 10, 21,25)



Nota: Incluir en el paquete: **com.mp.practica4.ejercicio3**

Nota: Las pruebas de unidad se situaran en el paquete: **com.mp.practica4.ejercicio3.test**

Los siguientes ejercicios se basan en ejercicios realizados en la práctica anterior. Deberá copiar las clases ya creadas en los paquetes correspondientes a esta práctica y modificarlas para cada ejercicio.

4) Escriba el programa que resuelve el problema de la evaluación de polinomios, utilizando técnicas recursivas para calcular la potencia.

   Nota: Incluir en el paquete: **com.mp.practica4.ejercicio4**

   Nota: Las pruebas de unidad se situaran en el paquete: **com.mp.practica4.ejercicio4.test**

5) Escriba el programa que resuelve el problema de la subsecuencia máxima, utilizando técnicas recursivas.

   Pruebe este programa con los tests desarrollados para el problema de la subsecuencia.

   Nota: Incluir en el paquete: **com.mp.practica4.ejercicio5**

   Nota: Las pruebas de unidad se situaran en el paquete: **com.mp.practica4.ejercicio5.test**

6) Escriba el programa que resuelve la búsqueda binaria recursiva. En esta solución utilizaremos la excepción ElementoNoEncontrado para avisar de que una búsqueda fracasó.

   Nota: Incluir en el paquete: **com.mp.practica4.ejercicio6**

   Nota: Las pruebas de unidad se situaran en el paquete: **com.mp.practica4.ejercicio6.test**