ESTRUCTURA DE DATOS Y ALGORITMOS **27-Enero-2005**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**

1. Ejercicio (2,5 puntos)

Diseñar y escribir en Java, un subprograma que intercambie dos nodos, *x* e *y*, en una lista *L* simplemente enlazada, dadas sólo las referencias a los nodos *x* e *y*. ¿Cuál es el tiempo de ejecución del subprograma, en función de *n* (cantidad de nodos en *L*)?

Nota: El diseño se puede realizar gráficamente o algorítmicamente, describiendo claramente los pasos a seguir. Escribir en Java la estructura de las clases utilizadas para la lista enlazada simple.

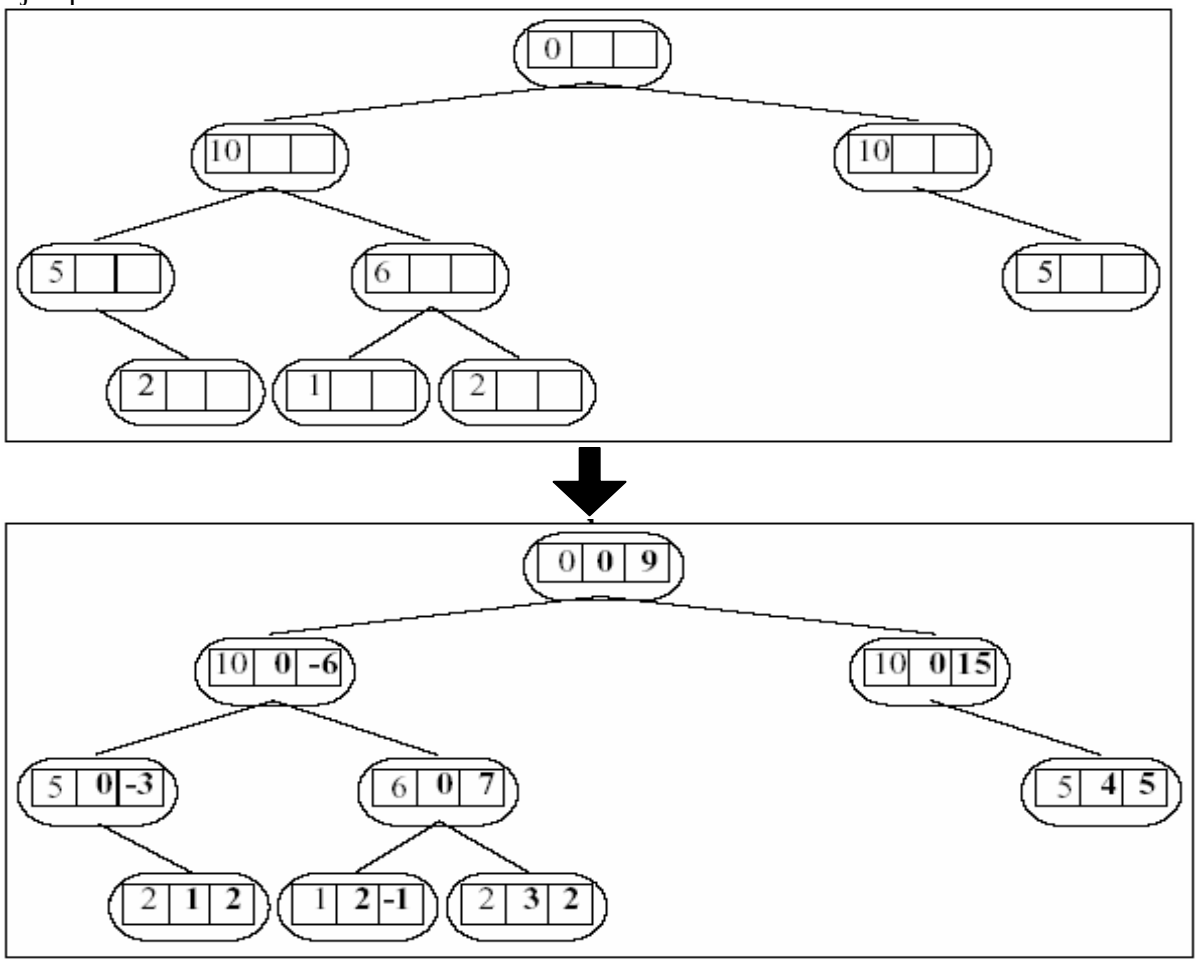
2. Ejercicio. (3 puntos)

Especificar, diseñar e implementar un subprograma tal que dado un árbol binario de elementos de tipo Nodo, devuelva el mismo árbol pero etiquetado. El subprograma etiquetará el subcampo *indice*: 1) de los nodos hoja, con el valor de la posición que ocupa el nodo hoja recorriendo todos los nodos hojas de izquierda a derecha y 2) de los nodos internos con el valor 0. En cuanto al subcampo *diferencia*, su valor indicará la diferencia de peso que tienen los nodos izquierdo y derechos del subárbol que comienza en ese nodo. El valor negativo indica que el árbol está desequilibrado hacia la izquierda (ver Anexo II)

```
Class Nodo {
     int peso;
     int indice;
     int diferencia;
}
```

```
Public void etiquetar();
```

Ejemplo:



Anexo II: Especificación de las clases BTNode, BinTree y BinTreeItr
NOTA: Todo método necesario tendrá que ser implementado.

```
public class BTNode{
        Object content;
        BTNode left;
        BTNode right;
}

public class BinTree{
        BTNode root;
}
```

```
public class BinTreeItr{
        BTNode actual;
        BinTree bTree;
}
```

3. Ejercicio (4,5 puntos)

Queremos implementar un programa para un broker de bolsa, para facilitarle la gestión de su cartera de valores y poder calcular las ganancias o perdidas que ha tenido después del cierre. Este broker actúa en diferentes bolsas del mundo. Normalmente suele tener 10 valores en cartera, aunque pueden ser más, ya que el número total de valores de todas las bolsas es más de 100.

Cuando se vende una acción común de un valor (una empresa), la **ganancia de capital** (o a veces pérdida) es la diferencia entre su precio de venta y el precio pagado originalmente para comprarla. Esta regla se puede comprender con facilidad para el caso de una sola acción, pero si se venden varias acciones compradas durante largo tiempo, se deben identificar las que realmente se venden.

Un principio normal en contabilidad, para identificar qué acciones se vendieron en este caso, es usar el siguiente **protocolo**: *las acciones que se venden son las que se han conservado el mayor tiempo* (en realidad, éste es el método predeterminado que se incorpora en varios paquetes de programas de finanzas personales).

Por ejemplo, supongamos que se compran 100 acciones a 20.00€ en el día 1, 20 a 24.00€ en el día 2, 200 a 36.00€ en el día 3, y que se venden 150 acciones a 30.00€ en el día 4. Entonces, al aplicar el protocolo muestra que de las 150 acciones vendidas, 100 fueron compradas en el día 1, 20 en el día 2 y 30 en el día 3. En este caso, la ganancia de capital sería $100 \cdot 10 + 20 \cdot 6 + 30 \cdot (-6)$, es decir, 940€.

Especificación de los métodos de la clase **CarteraDeValores**

int ganancia; //representa las ganancias o perdidas acumuladas (+:ganancia, -: pérdida)

void inicializar(); // Inicializa la cartera de valores a vacía.

boolean esVacia(); //Decide si la cartera de valores está vacía.

void apertura(*String file***);**
// Carga los datos almacenados en el fichero *file* en la cartera de valores.

void realizarTransaccion(*Transaccion transaccion***);**
// Realiza la transacción indicada por *transaccion*, actualizando la cartera de valores y las ganancias o
// pérdidas.

void verEstado();
// Escribe por pantalla el estado de las ganancias y perdidas.

void cierre(*String file***);**
// Almacena en el fichero *file* el estado de la cartera de valores.

SE PIDE:

a) **(1,5 puntos) Diseñar** una representación adecuada para gestionar la cartera de valores, utilizando las estructuras de datos vistas en clase. De forma, que todas las operaciones (menos apertura y cierre) sean de $O(1)$. **Implementar** en Java las variables miembro y clases necesarias para **CarteraDeValores**.

b) **(2 puntos)** Implementar los métodos:
*apertura(String file)***,** *realizarTransaccion(Transacción transaccion)* y *verEstado()*

c) **(1 punto)** Escribir un programa que dados los ficheros F_INICIAL.TXT (que contiene el estado inicial de la cartera) y F_TRANSACCIONES.TXT (que contiene una secuencia de transacciones realizadas en días consecutivos), calcule y escriba por pantalla cuanto es la ganancia (o pérdida) total del capital para toda la secuencia por medio del protocolo indicado para identificar las acciones.

Ejemplo:

F_INICIAL.TXT

FCC	100	20
FCC	20	24
REP	30	10
REP	50	11
GAS	30	5
GAS	200	4

F_TRANSACCIONES.TXT

C	FCC	200	36
V	FCC	150	30
C	REP	20	12
C	GAS	40	5
V	REP	20	13
V	GAS	130	4

F_CIERRE.TXT

FCC	170	36
REP	10	10
REP	50	11
REP	20	12
GAS	100	4
GAS	40	5

Pantalla: Las ganancias han sido: **970€**

Nota: C: Comprar V: Vender

Nota: En caso de necesitar alguna otra clase o método en alguna de las clases dadas, implementarlo. Para la realización de este ejercicio se dispone de las siguientes clases:

```
class Token {
  //Inicializa el fichero
  Token(String file);
  //cierto si existen más líneas en el fichero,
  //falso en caso  contrario
  bolean hasMoreTokens();
  //devuelve los datos de una línea del
  //fichero en un objeto de tipo Transaccion
   Transaccion getToken();
}
```

```
interface Entry{
  Object getKey();
  Object getValue()
}
```

```
class Transaccion{
   String operacion;
   String nombreValor;
   CompraVenta compraVenta;
  //devuelve la operación realizada
  public String getOperacion();
  //devuelve el nombre del valor
  public String getNombre();
  //devuelve la compra o venta realizada
  public CompraVenta getCompraVenta()
}
```

```
class CompraVenta{
   int numAcciones;
   int precio;
   //devuelve el numAcciones
   public int getNumAcciones();
   //devuelve el precio
   public int getPrecio();
}
```