ESTRUCTURA DE DATOS Y ALGORITMOS **9-Septiembre-2009**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**
Nombre y apellidos: _____ Nota:

- Escribe tu **nombre y apellidos** en esta hoja e inmediatamente en todas las suplementarias, incluso las de sucio. El no hacerlo puede suponer tu expulsión
- Puedes utilizar **lápiz** para tus respuestas. No está permitido el uso de apuntes o libros. No puedes tener un **móvil** encendido ni utilizar **calculadora**.
- **Todos los alumnos implicados en una copia** de un ejercicio **tendrán una nota final de 0**. El alumno es responsable de velar por su examen. Es decir **tanto el que copia como el que se deja copiar (ya sea de manera activa o pasiva) recibirán el mismo castigo sin que exista atenuante alguno**

1. Ejercicio (2 puntos)

a) Un vector tiene la siguiente propiedad:
   1. Inserciones y borrados eficientes en medio de la estructura
   2. Inserciones y borrados eficientes en ambos extremos de la estructura
   3. Inserciones y borrados eficientes al final de la estructura
   4. Ninguna de las anteriores

b) Una lista enlazada simple tiene la siguiente propiedad:
   1. Acceso secuencial
   2. Acceso bidireccional
   3. Acceso aleatorio
   4. Todas las anteriores

c) Cual de las siguientes afirmaciones es cierta:
   1. Las estructuras de datos almacenan operaciones y las aplican sobre un objeto
   2. Las estructuras de datos restringen el uso de las variables
   3. Las estructuras de datos almacenan referencias a objetos
   4. Las estructuras de datos almacenan variables de objetos

d) Las principales diferencias entre las distintas estructuras son:
   1. La forma de organizar las operaciones y las restricciones que imponen sobre los datos
   2. La forma de organizar los datos y las restricciones que imponen sobre las operaciones
   3. La forma en que aplican la abstracción sobre variables de objetos
   4. En principio, no existe ninguna diferencia sustancial, depende del problema planteado

ESTRUCTURA DE DATOS Y ALGORITMOS **9-Septiembre-2009**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**
Nombre y apellidos: _____

- Escribe tu **nombre y apellidos** en esta hoja e inmediatamente en todas las suplementarias, incluso las de sucio. El no hacerlo puede suponer tu expulsión
- Puedes utilizar **lápiz** para tus respuestas. No está permitido el uso de apuntes o libros. No puedes tener un **móvil** encendido ni utilizar **calculadora**.
- **Todos los alumnos implicados en una copia** de un ejercicio **tendrán una nota final de 0**. El alumno es responsable de velar por su examen. Es decir **tanto el que copia como el que se deja copiar (ya sea de manera activa o pasiva) recibirán el mismo castigo sin que exista atenuante alguno**

2. *Ejercicio (4 puntos)*

Normalmente, una expresión aritmética es representada mediante un árbol binario. Dado una operación con dos operandos la podemos representar como un árbol cuya raíz sea el operador (diádico[1]), y sus subárboles sean los operandos. P.ej., la operación: (A-(4*B-3)*J+P/5)*1/23-X, equivaldría, rellenando paréntesis, a: (((A-(((4*B)-3)*J))+(P/5))*(1/23))-X, y su árbol sería el de la figura situado a la izquierda:

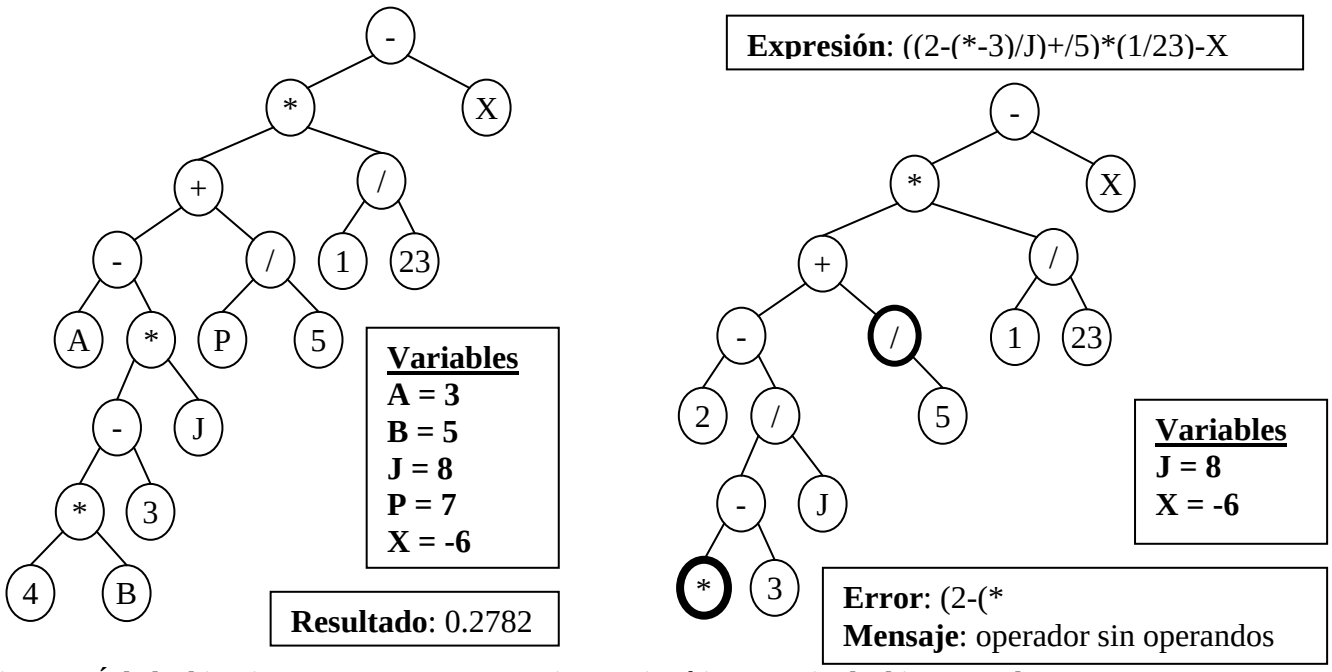

Figura 1. Árboles binarios que representan expresiones aritméticas. Izquierda: bien. Derecha: 2 errores

Para representar un árbol binario se utilizará la siguiente especificación de las clases `BTNode`, `BinTree` y `BinTreeItr`:

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

public class BinTree{
     BTNode root;
}
```

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

Nota: Cualquier otro método que actúe sobre estas clases tendrá que ser implementado.

[1] Diádico: Operador que requiere dos operandos

Para poder evaluar la expresión, además de interpretar las operaciones, necesitamos resolver los valores de las variables. Para ello, se proveen las siguientes clases:

```
public class Expr{
      public static boolean esOperardor(String op);
      public static boolean esOperando(String op);
      public static boolean esNumero(String op);
      public static float evaluar(String op, float op1, float op2);
}

public class Resolver{
      // estructura de datos para almacenar las variables y sus valores
      public void anadirVariable(String nombre, int valor);
      public boolean esVariable(String nombre);
      public int variable(String nombre);
}
```

Se pide:

a) (*1 punto*) **Completa** la definición de la clase `Resolver` definiendo sus atributos para almacenar las variables y sus valores, de forma que sus métodos sean lo más eficientes posibles. **Argumenta** tu decisión e **implementa** los métodos de la clase acorde a ella.

b) (*3 puntos*) **Especifica, diseña e implementa** un método que dado un conjunto de variables con sus correspondientes valores (una instancia de `Resolver`) y una expresión aritmética (un árbol binario), verifique y evalúe la expresión, mostrando finalmente en pantalla el resultado de la evaluación o el punto dónde se ha producido el error (es decir, muestra parte de la expresión procesada). La parte izquierda de la figura 1 muestra un ejemplo de una expresión correcta con su respectivo resultado de la evaluación. En cambio, la parte derecha de la figura 1 muestra una expresión con dos errores (círculos en negrita): son operadores a los que les falta algún operando.

   El método o modo de resolución que utilizaremos será el de “evaluación o verificación impaciente”. Es decir, en el momento en que hay una operación y dos operandos, se evalúa o en caso de detectar error se para y se muestra la parte de la expresión que ha sido procesada. Por ello, aunque haya dos errores, sólo se muestra el primero que ha sido detectado.

3. ***Ejercicio (4 puntos)***

Últimamente, y ahora que estamos en periodo de crisis, es habitual que las empresas presenten un *ERE* (*E*xpediente de *R*egulación de *E*mpleo) en los casos que tengan poco trabajo. Esta medida supone que durante un periodo (p. ej. un año) los empleados van a ser enviados al paro si la situación del trabajo así lo requiere.

La gestión de estos periodos en una empresa grande puede suponer más de un quebradero de cabeza. Normalmente, una empresa grande se organiza de forma jerárquica. Esta jerarquía se compone (como máximo) de 5 categorías distintas, siendo el 1 el de mayor categoría. Además, el número de empleados en cada categoría es proporcional al número que identifica su categoría. Por ejemplo, en la categoría 5 el número de empleados, por lo menos, es 5 veces mayor que la 4. En la 4, 4 veces mayor que la 3. Y así de forma sucesiva. Por ejemplo, en una empresa de 500 empleados, la distribución por categorías puede ser la siguiente (empezando desde la 1 categoría): 2, 4, 14, 80 y 400. (**Nota**: no puede haber categorías entremedio sin empleados, sólo las de menor categoría)

En este tipo de empresas, lo normal es que también haya grupos sindicales. Antes de presentar el ERE en la administración, la empresa tiene que negociar las condiciones del expediente con ambos: los grupos sindicales y la administración. Durante el periodo del expediente ambos, la empresa y los empleados, están obligados a cumplir las condiciones del acuerdo y en este caso, son las siguientes:

a) Todos los empleados, cada uno en su respectiva categoría, serán ordenados por su antigüedad. Cuando un número de empleados tiene que ir al paro o incorporarse al trabajo, los primeros empleados en irse o incorporarse serán los de menor categoría y antigüedad. Pero, también hay que recordar que hay que mantener la proporcionalidad requerida en cada categoría. Por ejemplo, si en la categoría 5 no hay 5 veces más empleados que en la 4, entonces hay que enviar o incorporar a alguno de la 4, y así de forma sucesiva.
b) La empresa puede mandar al paro, como máximo durante 15 días y de forma intermitente, a un número de empleados.
c) Un empleado se incorporará al trabajo después de transcurrir los 15 días, o puede que antes, si el trabajo así lo requiere.
d) Es necesario registrar el primer día de su nueva situación: trabajando o en paro.
e) Una vez que un empleado haya vuelto de nuevo a su puesto de trabajo (independientemente del modo), para que el mismo empleado pueda ser enviado de nuevo al paro se tienen que cumplir dos condiciones: *1)* que todos los empleados de la misma categoría hayan estado en el paro **el mismo numero de veces o 1 más** y *2)* que hayan pasado, como mínimo, otros 15 días.

Se quiere desarrollar un sistema informático para facilitar a la empresa la gestión de la regulación de sus empleados. Como parte del sistema se quiere implementar la clase ERE que tiene la siguiente interfaz:

• ***Constructor***: Genera un gestor de ERE vacío, sin ninguna información.
• ***inicializar (String fichero)***: Dado un fichero de texto que contiene los datos de todos los empleados de la empresa (nombre, apellido, categoría y antigüedad), se almacenan en memoria en situación de estar trabajando y ordenados por la antigüedad, cada uno en su respectiva categoría.
• ***enviarAlParo (int numero)***: Altera la situación de un número de empleados, pasando de estar trabajando a estar en paro. El listado de los empleados seleccionados se muestra en pantalla y/o el número que no se ha podido atender.
• ***incorporarAlTrabajo(int numero)***: Altera la situación de un número de empleados, pasando de estar en paro a estar trabajando. El listado de los empleados seleccionados se muestra en pantalla, en el mismo orden en que fueron enviados al paro.
• ***incorporarAlTrabajoD15()***: Altera la situación de aquellos empleados que hayan estado durante 15 días en paro, pasando a estar trabajando. El listado de los empleados seleccionados se muestra en pantalla.
• ***incorporarAlTrajadoTodos()***: Altera la situación de todos los empleados que están en paro, pasando a estar trabajando. El listado de los empleados seleccionados se muestra en pantalla.

ESTRUCTURA DE DATOS Y ALGORITMOS **9-Septiembre-2009**
Titulación: Ingeniero Técnico en Informática de Gestión **Curso: 2º**

Se dispone de la clase *Empleado*, para almacenar la información de un empleado: nombre, apellido, categoría a la que pertenece y la antigüedad (en días). Su interfaz provee los métodos de acceso (*get*), igualdad (*equals*) y orden por antigüedad (implementa la interfaz *Comparable*, es decir, el método *compareTo*). También se despone de la clase Token (ver anexo), para la manipulación de ficheros que contienen datos de empleados, y aparte, los siguientes métodos:

- public static *Date* ***hoy()***: Devuelve el día de hoy en el PC, en formato, DD-MM-AAAA.
- public static *int* **diferenciaDias***(Date fecha1, Date fecha2):* Dado dos fechas, calcula y devuelve el número de días que tienen de diferencia. El valor es positivo, si la primera fecha es menor que la segunda y negativo en caso contrario.

Se pide:

a) (1 punto) **Diseña** e **implementa** la estructura (las propiedades) más adecuada(s) para la clase *ERE*, cuya misión es gestionar la información sobre la situación (trabajando o en paro) de los empleados de una empresa que haya presentado el expediente regulador, teniendo en cuenta las condiciones arriba descritas y las estructuras de datos vistas en clase. **Nota**: no implementes ni modifiques la clase *Empleado*. En caso de necesitar algo más, implementa las clases y los métodos necesarios.

b) **Implementa** los métodos *inicializar* y *enviarAlParo* de la clase *ERE*.

Nota: No es necesario implementar los métodos típicos de cualquier estructura de datos que hayamos visto en clase, siempre que se utilicen los nombres habituales. También, se proveen los siguientes métodos típicos en una estructura de datos:

public Object[] **toArray**(); //devuelve un array con todos los datos almacenados en la estructura de datos
public void **init**(Object[] objects) // inicializa la estructura de datos con los datos almacenados en el array

Ejemplo:

empleados.txt

```
Yo        Trabajo   5      500
Yo        También   5      1000
Algunas   Veces     5      10
Yo        Siempre   1      5000
Yo        Jefe      2      3000
….
```

Tabla 1. Desglose, por categoría, del número de empleados trabajando y resumen de los que están en paro.

Operaciones/estado	1	2	3	4	5	En paro
situación inicial	2	4	14	80	400	0
enviarAlParo(160)	2	4	14	79	395	6
			…			
	2	4	14	56	280	144
	2	4	13	55	275	151
	2	4	13	54	267	157
incorporarAlTrabajo(4)	2	4	13	55	270	153

Nota: Si por ejemplo el Sr. Algunas Veces tiene la menor antigüedad entre todos los empleados, será el primero en ir al paro pero también en volver. Además, este señor no volverá a ir al paro hasta que todos los demás compañeros de su misma categoría hayan estado en el paro y hayan pasado por lo menos 15 días a partir de su última incorporación al trabajo.

Anexo:

```
class Token{
        public Token(String file); // Abre el fichero de texto indicado por el parámetro 'file'
        public boolean hasMoreTokens(); // Devuelve True si no se ha llegado al final de fichero y False, en caso contrario
        public Empleado getToken(); // Devuelve los datos de un empleado, correspondiendo a una línea en el fichero
}
```