Recuperación del Segundo Parcial de PRG - ETSInf
Fecha: 8 de junio de 2011. Duración: 1 hora y media (90 minutos)

NOTA: Se debe responder en hojas aparte. No es necesario entregar esta hoja.

Las confederaciones hidrográficas nos piden un programa en Java para llevar el control del agua embalsada en los pantanos. El programa deberá leer los datos de los pantanos del fichero `pantanos.txt`, cuyo formato es el siguiente:

```
4
    218     168  Benagéber
      6       4  El Regajo
     20       7  María Cristina
     71      52  Loriguilla
```

La primera línea contiene el número de pantanos, cada línea de las restantes contiene la descripción de un pantano: el primer valor **entero** representa la capacidad total, el segundo valor **entero** la cantidad de agua almacenada. El **resto** de la línea contiene el nombre del embalse. Los valores enteros representan hectómetros cúbicos (Hm^3). Un Hm^3 equivale a 10^9 litros de agua.

Para la implementación del programa es necesario programar tres clases: `Pantano`, `RedPantanos` y `ControlNiveles` cuyos detalles se describen a continuación.

1. (3 puntos) La clase `Pantano` debe tener tres atributos: una cadena de caracteres para el nombre y dos enteros para la capacidad y la cantidad embalsada.
   (a) (0.4 puntos) Declaración de los atributos.
   (b) (0.4 puntos) Constructor de la clase que reciba tres parámetros: nombre, capacidad y cantidad embalsada.
   (c) (0.6 puntos) Tres consultores. Uno por cada atributo.
   (d) (0.5 puntos) Un método que devuelva el porcentaje de ocupación.
   (e) (0.3 puntos) Un método para incrementar la cantidad de agua embalsada. Recibirá como argumento un entero con la cantidad en Hm^3 a aumentar.
   (f) (0.3 puntos) Un método para decrementar la cantidad de agua embalsada. Análogo al anterior, pero cuyo argumento será la cantidad a disminuir.
   (g) (0.5 puntos) Método `toString()` que muestre la información de un pantano como sigue:

   ```
   María Cristina       7      20     35%
   ```

   que corresponde, por este orden, a nombre, cantidad de agua embalsada, capacidad máxima y porcentaje de ocupación.

Solución:

```
public class Pantano
{
    /** Variable de instancia con el nombre del pantano. */
    private String nombre;
    /** Variable de instancia con la capacidad máxima del
        pantano en hectómetros cúbicos (Hm3) */
    private int capacidad;
    /** Variable de instancia con el nivel de ocupación
```

```
        actual del pantano en hectómetros cúbicos (Hm3) */
    private int nivel;

    public Pantano( String nombre, int capacidad, int nivel )
    {
        this.nombre = nombre;
        this.capacidad = capacidad;
        this.nivel = nivel;
    }
    public String getNombre() { return nombre; }
    public int getCapacidad() { return capacidad; }
    public int getNivel() { return nivel; }

    public double getPorcentajeOcupacion() { return (100.0*nivel)/capacidad; }

    public void incrementar( int hm3 )
    {
        nivel += hm3;
        nivel = Math.min( nivel, capacidad );
    }
    public void decrementar( int hm3 )
    {
        nivel -= hm3;
        nivel = Math.max( nivel, 0 );
    }

    public String toString()
    {
        return String.format("%-20.20s  %6d  %6d   %.0f%%", nombre, nivel, capacidad,
                                                             getPorcentajeOcupacion());
    }
}
```

2. (5.5 puntos) Clase `RedPantanos` con un único atributo, la lista de pantanos. *No es necesario mantener un contador de pantanos porque el número de pantanos cargados no variará a lo largo de la ejecución del programa.*
   (a) (0.3 puntos) Declaración del array que almacenará la lista de pantanos.
   (b) (0.3 puntos) Constructor de la clase, que se encargará de invocar al método privado `leeFichero()`.
   (c) (1.5 puntos) Método `leeFichero()` que no espera ningún argumento y que debe leer el contenido del fichero `pantanos.txt` para almacenarlo adecuadamente en la lista de pantanos. Este método reservará el espacio necesario para todos los pantanos que figuren en el fichero.
   (d) (0.3 puntos) Método consultor para devolver el número de pantanos.
   (e) (0.7 puntos) Método `toString()` que devuelva una `String` con la información de todos los pantanos.
   (f) (1.2 puntos) `private void repasarAlertas( double umbral )`
   Escribe en la salida estándar todos los pantanos de la red para los que su ocupación (en %) supere el umbral indicado. Un ejemplo de alerta sería:

   ```
   *** ALERTA: María Cristina      18      20     90%
   ```

   (Se puede aprovechar el método `toString()` de la clase `Pantano`).

(g) (1.2 puntos) `private void actualizarVariaciones( int V[] )`
Actualiza la cantidad de agua en los pantanos de la red. El agua embalsada en el pantano i-ésimo variará en la cantidad indicada en `V[i]`, aumentará o decrecerá según el valor de `V[i]` sea positivo o negativo.

Solución:

```
import java.util.*;
import java.io.*;

public class RedPantanos
{
    private Pantano red[];

    public RedPantanos() { leeFichero(); }

    private void leeFichero()
    {
        try {
            Scanner sf = new Scanner( new File( "pantanos.txt" ) );
            int numPantanos = sf.nextInt();
            red = new Pantano [ numPantanos ];
            int contador=0;
            while( sf.hasNext() && contador < numPantanos ) {
                int capacidad = sf.nextInt();
                int nivel = sf.nextInt();
                String nombre = sf.nextLine().trim();
                red[contador++] = new Pantano( nombre, capacidad, nivel );
            }
            sf.close();
        }
        catch( IOException ioe )
        {
            ioe.printStackTrace( System.err );
            System.exit(-1);
        }
    }
    /** Este método no se pedía */
    public void guardaEnFichero()
    {
        try {
            PrintWriter pw = new PrintWriter( "pantanos.txt" );
            pw.println( red.length );
            for( Pantano p : red ) {
              pw.printf(" %6d  %6d  %s\n", p.getCapacidad(), p.getNivel(), p.getNombre());
            }
            pw.close();
        }
        catch( IOException ioe )
        {
            ioe.printStackTrace( System.err );
            System.exit(-1);
        }
    }
```

```
    public int numPantanos() { return red.length; }

    public String toString()
    {
        String s = "";
        for( int i=0; i < red.length; i++ ) s += red[i].toString() + "\n" ;
        return s;
    }

    public void actualizarVariaciones( int V[] )
    {
        for( int i=0; i < V.length && i < red.length; i++ ) {
            if ( V[i] > 0 )
                red[i].incrementar( V[i] );
            else if ( V[i] < 0 )
                red[i].decrementar( -V[i] );
        }
    }

    public void repasarAlertas( double umbral )
    {
        System.out.println( "\n" );
        for( int i=0; i < red.length; i++ )
            if ( red[i].getPorcentajeOcupacion() > umbral )
                System.out.println( "*** ALERTA: " + red[i].toString() );
    }
}
```

3. (1.5 puntos) Clase `ControlNiveles` que contendrá el método `main()` y el método `int [] cargarVariaciones( String nombreFichero )`.

   Lo que debe hacer el programa al ejecutar esta clase figura en el código siguiente. Vosotros debéis implementar el método `int [] cargarVariaciones( String nombreFichero )`.

   El formato del fichero de variaciones es simple, la primera línea contiene un entero con el número de pantanos, que siempre coincidirá con el número de pantanos que hay en el fichero `pantanos.txt`. Después aparecerá una línea por cada pantano en que se haya detectado variación, el primer valor es el índice del pantano y el segundo valor los Hm^3. Ver el siguiente ejemplo.

```
4
0  5
3  43
```

   El método `cargarVariaciones()` abrirá el fichero, leerá el primer entero que indica el número total de pantanos y creará un array cuyo tamaño será el número total de pantanos. A continuación leerá las variaciones de cada pantano, a razón de una variación por línea, el primer entero en cada línea es el índice del pantano que se utilizará para guardar el segundo valor de la línea en la posición correspondiente del array. Finalmente este método devolverá el array.

Solución:

```
import java.util.*;
import java.io.*;
```

```
public class ControlNiveles
{
    public static void main( String args[] ) throws Exception
    {
        RedPantanos red = new RedPantanos();
        int V[] = null;
        if ( args.length > 0 ) V = cargarVariaciones( args[0] );
        if ( V != null ) { // Solo si se encuentran variaciones.
            red.actualizarVariaciones( V );
            red.guardaEnFichero();
            red.repasarAlertas( 80.0 );
        }
    }

    private static int [] cargarVariaciones( String nombreFichero )
    {
        try {
            File df = new File( nombreFichero );
            if ( !df.exists() ) return null;
            Scanner sf = new Scanner( df );
            int numPantanos = sf.nextInt();
            int V[] = new int [ numPantanos ];
            int contador = 0;
            while( sf.hasNext() && contador < numPantanos ) {
                int indice = sf.nextInt();
                V[indice] = sf.nextInt();
                contador++;
            }
            sf.close();
            return V;
        }
        catch( IOException ioe )
        {
            ioe.printStackTrace( System.err );
            System.exit(-1);
        }
        return null;
    }
}
```