

# Examen de Fundamentos de Computadores y Lenguajes

## Examen Parcial. Junio 2006

Cuestiones (5 cuestiones, 5 puntos en total)

- 1) Escribir una clase que permita almacenar una lista de jugadores y sus correspondientes equipos. Los jugadores se representan por su nombre (un `String`), mientras que los equipos son objetos de la clase `Equipo`. La clase dispone de tres atributos privados:
  - `jugador`: array de strings que representan nombres de jugadores
  - `pertenece`: array de equipos
  - `num`: número actual de jugadores almacenados

Ambos arrays se organizarán de modo que `pertenece[i]` es el equipo al que pertenece `jugador[i]`

La clase debe tener un constructor que pone `num` a cero y crea los dos arrays del mismo tamaño, dato que se pasa como parámetro.

- 2) Añadir a la clase anterior un método que sirve para añadir un jugador y su correspondiente equipo a la lista. Ambos datos se pasan como parámetros. Si hay espacio, el nuevo jugador y equipo se insertan en la casilla `num` de ambos arrays, y luego se incrementa `num`. Si no hay espacio en los arrays para hacer esto, se lanza la excepción `NoCabe`, que está declarada en una clase aparte como

```
public class NoCabe extends Exception {}
```

- 3) Se dispone de la clase que se indica a continuación y que representa una matriz de números reales.

```
public class Matriz
{
    // atributos privados
    private double mat[][];
    ...
}
```

Escribir un método de esta clase que permita trasponer la matriz si ésta es cuadrada, intercambiando sus filas por sus columnas. Hacerlo a partir del siguiente pseudocódigo.

```
método trasponer() lanza NoCuadrada
si no coincide mat.filas con mat.columns
    lanzar excepción NoCuadrada
fin si
para todo i desde 0 hasta mat.filas-2 lazo
    para todo j desde i+1 hasta mat.columns-1 lazo
        intercambiar mat[i][j] con mat[j][i];
    fin lazo
fin lazo
fin trasponer
```

Nota: `mat.filas` representa en el pseudocódigo el número de filas de `mat`, y `mat.columns` el de columnas. Usar en Java el atributo `length` (respectivamente `mat.length` y `mat[0].length`).

- 4) Indicar razonadamente el ritmo de crecimiento del tiempo de ejecución del algoritmo anterior mediante la notación  $O(n)$ .
- 5) Se dispone de la clase enumerada que se muestra a continuación. Escribir un método estático para una clase aparte, con un parámetro del tipo `String`. El método debe invocar a `construye()` con el `String` que se le ha pasado como parámetro, y tratar la excepción `Incorrecto` de modo que se ponga en pantalla un mensaje de error en el que se incluya el valor del parámetro (ej., "El texto xxx es incorrecto", siendo xxx el valor del parámetro).

```
public enum DiaSemana
{
    monday, tuesday, wednesday, thursday, friday, saturday, sunday;

    public static DiaSemana construye(String texto) throws Incorrecto
    {
        ...
    }
}
```

la excepción `Incorrecto` está declarada en una clase aparte como

```
public class Incorrecto extends Exception {}
```

# Examen de Fundamentos de Computadores y Lenguajes

## Examen Parcial. Junio 2006

### Problema (5 puntos)

Se desea hacer un programa para gestionar los datos de la estadística de un vivero de árboles. En el vivero se plantan árboles en tres tipos de tierra, para experimentar cuál es la mejor tierra para cada especie de árbol.

El tipo de tierra se representa con un tipo enumerado:

```
public enum Tierra
{
    normal, silicea, calcarea
}
```

Para cada especie de árbol, se guardan sus datos en la clase EstadisticaArbol que tiene la siguiente interfaz

```
public class EstadisticaArbol
{
    /**
     * Constructor al que se le pasa el nombre de la especie.
     * Pone los datos a cero
     */
    public EstadisticaArbol(String especie) {...}

    /**
     * Retorna el nombre de la especie de arbol
     */
    public String especie() {...}

    /**
     * anade números de árboles plantados y crecidos,
     * para el tipo de tierra indicado
     */
    public void anade(int plantados, int crecidos, Tierra tipoTierra) {...}

    /**
     * Retorna el numero de arboles plantados para el tipo de tierra indicado
     */
    public int plantados(Tierra tipoTierra) {...}

    /**
     * Retorna el numero de arboles que han crecido,
     * para el tipo de tierra indicado
     */
    public int crecidos(Tierra tipoTierra) {...}
}
```

Los datos del vivero se guardan en la clase Vivero, cuya interfaz es:

```
public class Vivero
{
    /**
     * Busca el arbol cuyo nombre se indica y retorna sus datos
     * estadisticos. Si no lo encuentra, retorna null
     */
    public EstadisticaArbol busca (String especieArbol) {...}
}
```

Lo que se pide es escribir la clase DatosVivero, cuya interfaz se muestra a continuación

```
public class DatosVivero
{
    public DatosVivero(Vivero v) {...}

    public double conExito(String[] especie, Tierra tipoTierra)
        throws NoEncontrado
    {...}

    public Tierra mejorTierra(String especie) throws NoEncontrado
    {...}

    public void escribeEnFichero(String nombreFichero, String especie)
    {...}
}
```

La clase debe tener un atributo que es un objeto de la clase vivero. También debe tener los siguientes métodos:

- *Constructor*. Se le pasa el vivero como parámetro y lo copia en el atributo
- *conExito*: Retorna el porcentaje de arboles con éxito ( $\text{crecidos} \cdot 100 / \text{plantados}$ ) para el tipo de tierra indicado, dada una lista de nombres de especies de arboles. La lista es un array de Strings. Si alguno de los árboles no se encuentra, lanza NoEncontrado.
- *mejorTierra*: Retorna el mejor tipo de tierra (la que tiene un porcentaje de éxito mayor) para la especie indicada. Si no se encuentra, lanza NoEncontrado.
- *escribeEnFichero*: Escribe en un fichero de texto cuyo nombre se le pasa como parámetro todos los datos de la especie cuyo nombre se pasa también como parámetro. Pone un dato por línea, anteponiendo un breve texto descriptivo. En la última línea pone el mejor tipo de tierra para esa especie. Si no se encuentra la especie, pone en el fichero el nombre de la especie y un mensaje de error.

La excepción NoEncontrado está definida en una clase escrita aparte de la forma:

```
public class NoEncontrado extends Exception {}
```

*Nota*: cada parte de la clase DatosVivero se valorará según su dificultad.