Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Septiembre 2002

Cuestiones (5 cuestiones, 5 puntos en total)

1) Se dispone de la clase `Coche` declarada de la siguiente manera:

```
public class Coche {
    private int color;
    private String modelo;
    public static final int ROJO=1;
    public static final int VERDE=2;
    public static final int AZUL=3;
}
```

Se pide añadir a esta clase un constructor al que se le pasen como parámetros un número entero (el color del coche) y un string (el modelo). El constructor deberá comprobar si el valor del color coincide con alguna de las constantes de arriba. En caso afirmativo, almacenará el color y el modelo en los campos correspondientes. En caso negativo, lanzará una excepción `ColorErroneo`, que ha sido declarada aparte de la forma:

```
public class ColorErroneo extends Exception {}
```

2) Escribir un método para la clase `Coche` indicada arriba que muestre en una ventana (por ejemplo de la clase `Escritura`) el modelo del coche y su color en letra (rojo, verde, o azul).

3) ¿Qué diferencia hay entre un campo estático de una clase y uno normal? (contestar en menos de 8 líneas)

4) Escribir un fragmento de programa en Java que, dada la variable `N` entera, calcule el factorial de `N` almacenándolo en la variable entera `factorial`, de acuerdo con el siguiente algoritmo:

```
si N es negativo, factorial=-1 //para indicar que hay error
si no, si N es 0, factorial=1
si no:
  factorial=1
  lazo desde i=1 hasta N
      factorial=factorial*i
  fin del lazo
fin del "si"
mostrar el factorial en la pantalla
```

5) Se desea diseñar un algoritmo que escriba los $n=N>0$ primeros términos de una progresión aritmética de razón $r=R$ conociendo el primer término $a=A$. Sabiendo que el término i-ésimo se calcula por la fórmula $A+(i-1)R$, se pide especificar y diseñar en pseudocódigo un programa que efectúe dicha tarea. El programa no debe hacer multiplicaciones ni divisiones; solamente están permitidas adiciones y sustracciones.

Para escribir un número i utilizar la notación *escribir*(i) en el pseudocódigo.

Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Septiembre 2002

Problema (5 puntos)

Se desea crear una clase para gestionar los datos de avistamientos de aves en una marisma. Para ello se dispone de la clase `Especie` que representa los datos de una especie de ave:

```
public class Especie {
   public String nombreVulgar;
   public String nombreCientifico;
}
```

Y de la clase `ListaEspecies`, que almacena una lista de especies y los datos del número de ejemplares avistados de cada especie en cada mes del año. Esta clase ya está implementada y su interfaz pública es:

```
public class ListaEspecies {
   public int numEspecies() {...}
   public Especie especieNum(int numEspecie)
      throws NoEncontrada {...}
   public void avistar (Especie e, int numEjemplares, int mes)
      throws NoCabe {...}
   public int numEjemplares (int numEspecie)
      throws NoEncontrada {...}
   public int numEjemplares (int numEspecie, int mes)
      throws NoEncontrada {...}
   public int numEjemplares (Especie e)
      throws NoEncontrada {...}
   public int numEjemplares (Especie e, int mes)
      throws NoEncontrada {...}
}
```

Las operaciones de la clase `ListaEspecies` se describen a continuación:

- `numEspecies()`: retorna el número actual de especies en la lista.
- `especieNum(numEspecie)`: retorna la especie que ocupa el número `numEspecie` en la lista (empezando por cero). Si `numEspecie` es menor que cero o mayor o igual al número actual de especies, lanza `NoEncontrada`.
- `avistar(e,numEjemplares,mes)`: añade a la lista un nuevo avistamiento. Si la especie `e` es nueva y la lista está llena, lanza NoCabe. En caso contrario añade `numEjemplares` a las aves de la especie `e` avistadas en el mes indicado por `mes` (número entre 1 y 12).
- `numEjemplares()`: retorna el número de ejemplares de una especie. Para las versiones de un sólo parámetro, se retorna la suma total de todos los meses. Para las versiones de dos parámetros se retorna el dato del mes solicitado (número entre 1 y 12). En las dos primeras versiones, se retornan los datos de la especie que ocupa el número `numEspecie` en la lista (empezando por cero), aunque si `numEspecie` es menor que cero o mayor o igual al número actual de especies se lanza `NoEncontrada`. Para las dos últimas versiones la especie es la indicada por `e`, aunque si la especie no se encuentra en la lista el método lanza `NoEncontrada`.

Las excepciones `NoCabe` y `NoEncontrada` están declaradas en clases aparte ya realizadas.

Lo que se pide es crear una clase pública llamada `Marisma`, que deberá tener los siguientes campos públicos:

- `nombre`: es un string que representa el nombre de la marisma
- `lista`: es un objeto de la clase `ListaEspecies`, que contiene la lista de las especies y datos de avistamientos de esa marisma.

Asimismo, la clase `Marisma` deberá tener los siguientes métodos públicos:

- constructor: se le pasan como parámetros el nombre y la lista de especies, que se guardarán en los campos correspondientes.
- `avistar`(): método sin parámetros. Sirve para pedir por teclado los datos de un avistamiento y guardarlos en la lista. Para ello, pide por teclado el nombre vulgar de una especie, el nombre científico, el número de ejemplares avistados, y el número del mes (de 1 a 12). Con estos datos se crea un objeto de la clase `Especie` y se invoca el método `avistar`() de la lista de especies. Si se lanza `NoCabe`, se pone un mensaje de error en la pantalla.
- `maxima`(): método que retorna un objeto de la clase `Especie`, y tiene un parámetro entero que es el número de un mes. Sirve para retornar la especie que presenta el máximo número de ejemplares avistados en el mes indicado por el parámetro. Para ello busca en un lazo en el que recorre cada especie de la lista aquella especie cuyo número de ejemplares es máximo en el mes indicado. Finalizado el lazo, retorna la especie que resultó tener el máximo número de ejemplares. Si se lanzase `NoEncontrada`, el método retornará el valor `null`.
- `pertenece`(): método que retorna un booleano, y tiene un parámetro de la clase `Especie`. Este método sirve para saber si la especie indicada por el parámetro pertenece o no a la lista. Para ello comprueba el número de ejemplares de la especie indicada por el parámetro. Si el número de ejemplares es mayor que cero retorna `true`; si no, retorna `false`. Si se lanza `NoEncontrada`, entonces la especie indicada no está en la lista y se debe retornar `false`.