Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Junio 2002

Cuestiones (5 cuestiones, 5 puntos en total)

1) Escribir una clase que represente una fecha. Deberá tener:
   - tres campos privados enteros: `dia`, `mes`, `año`
   - un constructor, al que se le pasan tres parámetros enteros que representan el día, mes y año, y que se guardarán en los respectivos campos
   - un método que retorne un String conteniendo la fecha en el formato `d/m/a`, donde `d` es el día, `m` el número del mes, y `a` el año.

2) Escribir un método que llame al método estático `opera` de la clase `Ecuacion`, que se indica abajo, de modo que si se lanza la excepción `ErrorDeCalculo`, se lance a su vez la excepción `Error`, cuyo constructor no tiene parámetros. Ambas excepciones están declaradas en clases independientes.

```
static void opera() throws ErrorDeCalculo {...}
```

3) Explica brevemente (menos de 8 líneas) porqué el uso de la máquina virtual Java permite la portabilidad, pero a costa de una menor eficiencia.

4) Escribir un fragmento de programa que invoque al método estático `calcula` de la clase `Raices`, que aparece abajo, y lo haga dentro de un lazo cuyas características serán:
   - se usará una variable de control de tipo `double` cuyo valor inicial es 1.0, y que se incrementa en 0.1 unidades cada vez
   - condiciones de finalización del lazo: variable de control > 10.0, o diferencia entre el valor retornado por `calcula` en una iteración y en la anterior < 1.0e-9

   En la llamada a `calcula` se usará como parámetro la variable de control. El método `calcula` tiene la siguiente cabecera:

```
static double calcula(double i) {...}
```

5) Se dispone de dos vectores de números reales con $N>0$ componentes, $a=A$ y $b=B$. Se desea construir un programa que calcule su producto escalar depositándolo en una variable c, es decir:

$$c = \sum_{1 \le i \le N} A[i]B[i]$$

   Se pide especificar el programa y posteriormente diseñarlo manteniendo invariante la siguiente condición en el bucle (es decir, en la iteración j-ésima c debe tener el valor indicado a continuación):

$$c = \sum_{1 \le i \le j} a[i]b[i]$$

   Nótese que el rango adecuado para la variable j es $1 \le j \le N+1$. Deducir la condición de continuación adecuada para ese invariante, deducir las condiciones iniciales, y el cuerpo del bucle. Indicar el número de vueltas que da el bucle para calcular c en función de N.

Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Junio 2002

Problema (5 puntos)

Se desea hacer una parte de un programa para gestionar un catálogo de estrellas. Se dispone de una clase, `CatalogoEstelar`, cuya interfaz aparece a continuación:

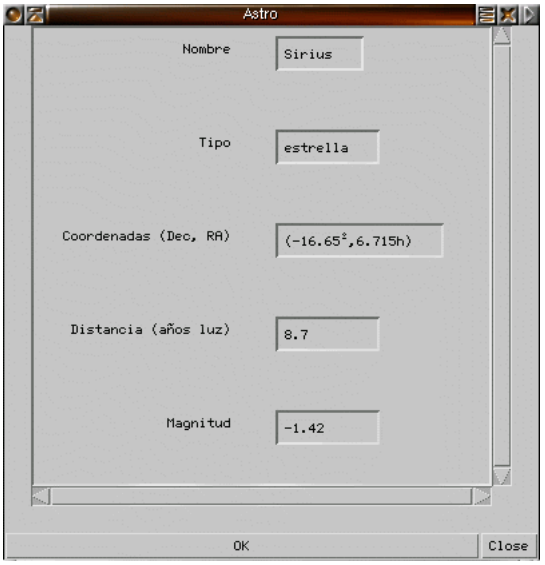
```
public class CatalogoEstelar {
   public static Astro[] cercanos
      (double ra, double dec, double angulo) throws Demasiadas {...}

   public static Astro encuentra (String nombre)
      throws NoEncontrado{...}
}
```

Como puede observarse, la clase tiene dos métodos estáticos, cuya función es:

- `cercanos`: retorna un array con las estrellas del catálogo situadas en un ángulo aparente menor o igual a `angulo`, desde las coordenadas estelares (`ra`,`dec`). Si el número de estrellas obtenidas fuese mayor que 1000, lanza `Demasiadas`.
- `encuentra`: retorna el astro cuyo nombre coincide con el parámetro `nombre`; si no lo encuentra, lanza `NoEncontrado`.

La clase `CatalogoEstelar` ya está hecha, pero usa otra clase, `Astro`, que debemos escribir, y que deberá contener lo siguiente:

- campos públicos:
  - `nombre`, de tipo `String`, representa el nombre del astro
  - `tipo`: de tipo entero, representa el tipo de astro, con un valor entre 0 y 3; definir además 4 constantes para los tipos de astros: estrella, nebulosa, galaxia, y cluster.
  - `ra`, `dec`, `mag`, `dist`: campos de tipo `double` que representarán, respectivamente, la ascensión recta (en horas), declinación (en grados), magnitud (o brillo aparente) y distancia (en años luz).
- método público: `muestra`, sin parámetros, que muestra en un objeto de la clase `fundamentos.Escritura` los datos del astro, con el formato indicado en la figura



Observar que las coordenadas estelares (`ra` y `dec`) se ponen juntas, entre paréntesis y separadas por coma. Además, el tipo de astro se pone con letra y no con número.

Por otro lado se desea escribir una clase, `Observatorio`, con los tres métodos siguientes:

```
public Astro[] filtroMagnitud (Astro[] conjunto, double magMin)
public Astro[] filtroTipo (Astro[] conjunto, boolean[] presente)
public void muestraCercanos
    (String nombre, double magMin, boolean[] presente)
```

- `filtroMagnitud()`: se le pasa un array de astros (`conjunto`) y debe crear otro array del mismo tamaño, y meter en él sólo los astros cuya magnitud sea mayor o igual que `magMin`. Posteriormente, debe crear un tercer array de tamaño igual al número de astros obtenidos y copiar en él estos astros. Finalmente retornará el nuevo array.
- `filtroTipo()`: hace lo mismo que `filtroMagnitud`, pero usando un criterio de filtrado diferente. El filtrado en este caso es por tipo: el astro sólo se añade al array final si la casilla del array `presente` correspondiente a su tipo vale `true`. Por ejemplo, si el array `presente={true, false, true, false}`, sólo se tendrán en cuenta los astros cuyo tipo sea 0 o 2.[1]
- `muestraCercanos()`: debe hacer lo siguiente:
  - Obtener mediante una llamada a `encuentra`() el astro cuyo nombre es el parámetro `nombre`. Si se lanzase `NoEncontrado` entonces pone un mensaje de error y termina
  - Definir una variable `angulo` con un valor inicial de 5 grados. Invocar al método `cercanos`() para obtener los astros cercanos al astro obtenido en el paso anterior (usando sus coordenadas `ra` y `dec`), y con el ángulo indicado por `angulo`. Si se lanza `Demasiados`, reintentar la operación reduciendo el ángulo en un 10%, sucesivamente hasta que a operación tenga éxito.
  - Aplicar al conjunto de astros obtenido el filtro de magnitud con el valor del parámetro `magMin`. Basta para ello invocar a `filtroMagnitud`()
  - Aplicar al nuevo conjunto obtenido el filtro de tipo de acuerdo al parámetro `presente`. Basta para ello invocar a `filtroTipo`()
  - Mostrar uno por uno todos los astros obtenidos en este proceso, invocando a `muestra`()

Las clases `NoEncontrado` y `Demasiadas` están definidas en clases independientes, de la manera siguiente:

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

Se pide implementar las clases `Astro` (1.5 puntos) y `Observatorio` (3.5 puntos)

1. Escribir sólo lo que sea diferente respecto al método anterior