Soluciones al Examen de Fundamentos de Computadores y Lenguajes

Examen Parcial. Junio 2005

Cuestiones (5 cuestiones, 5 puntos en total)

1) Escribir una clase con un atributo privado que sea un array de 1000 números reales, y un constructor que pone el valor de todas las casillas del array de índice igual a 1 o superior de manera que la casilla *i* se hace igual al logaritmo neperiano de *i*. La casilla cero no se usa.

```
public class Logs
{
    private final int MAX=1000;
    private double[] ln;

    /**
     * Constructor for objects of class Logs
     */
    public Logs()
    {
        ln= new double[MAX];
        for (int i=1; i<MAX; i++) {
            ln[i]=Math.log((double)i);
        }
    }
}
```

2) Añadir a la clase anterior un método que calcule de manera aproximada la función e^x, buscando en el array la casilla cuyo contenido es más próximo a *x*, y retornando un número real de valor igual al índice de la casilla encontrada. Si el número *x* es menor que cero o mayor que el valor de la última casilla del array, lanzar la excepción `IllegalArgumentException`.

```
    public double exp(double x) throws IllegalArgumentException
    {
        if (x<0.0 || x>ln[MAX-1]) {
            throw new IllegalArgumentException();
        }
        double diferencia=Math.abs(x-ln[1]);
        int casillaEncontrada=1;
        for (int i=2; i<MAX; i++) {
            // empezamos en 2, pues la casilla 1 ya la hemos hecho
            double difActual=Math.abs(x-ln[i]);
            if (difActual<diferencia) {
                diferencia=difActual;
                casillaEncontrada=i;
            }
        }
        return (double) casillaEncontrada;
    }
```

3) Escribir un método estático en Java que responda al siguiente pseudocódigo, que corresponde a un algoritmo de "fuerza bruta" para saber si un número es primo o no.

```
método esPrimo
   datos de entrada: número entero n
   retorna: booleano que será true si n es primo y false si no
```

```
comienzo
   divisor=2
   mientras (divisor*divisor<=n) lazo
      si (n módulo divisor)==0 entonces
         retorna false // no es primo
      fin de si
      incrementa divisor en 1
   fin de lazo
   retorna true // es primo
fin
```

```
public class Primo
{
    public static boolean EsPrimo(int n) {
        int divisor=2;
        while (divisor*divisor<=n) {
            if (n % divisor == 0) {
                return false; //no es primo
            }
            divisor++;
        }
        return true; // no es primo
    }
}
```

---

4) Indicar razonadamente el ritmo de crecimiento del tiempo de ejecución del algoritmo anterior mediante la notación $O(n)$.

La instrucción interna al "if" es simple: O(1)

El "if" tiene una condición simple y sus instrucciones son O(1); tanto si se ejecutan sus instrucciones como si no, en total es: O(1)

Las instrucciones de dentro del lazo son O(1); su suma es O(1)

El lazo se hace $\sqrt{n-1}$ veces y su contenido es O(1); en total: $O(\sqrt{n})$

El método tiene el lazo y dos instrucciones simples O(1); su suma es; $O(\sqrt{n})$

---

5) Se dispone del siguiente tipo enumerado que representa días de la semana:

```
public enum DiaSemana
{
   lunes, martes, miercoles, jueves,
   viernes, sabado, domingo;
}
```

Escribir un método estático al que se le pase un valor del tipo enumerado y nos devuelva `false` si es un día de labor, y `true` si es un fin de semana. Implementarlo con una instrucción `switch`. El método debe obedecer a la siguiente interfaz:

```
public static boolean EsFinDeSemana (DiaSemana dia) {...}
```

```
public static boolean EsFinDeSemana (DiaSemana dia) {
    switch (dia) {
        case sabado:
        case domingo:
            return true;
        default:
            return false;
    }
}
```

Examen de Fundamentos de Computadores y Lenguajes

Examen Parcial. Junio 2005

Problema (5 puntos)

Se desea hacer un programa para localizar con un radiotelescopio diversas fuentes de radio situadas en el espacio. El radiotelescopio está implementado con una clase Java que tiene la siguiente especificación:

```
public class RadioTelescopio
{
    /**
     * Inicia el movimiento del telescopio hacia las coordenadas
     * indicadas en grados
     */
    public void mueve(double acimut, double elevacion) {...}

    /**
    * Retorna true si el telescopio esta quieto, y false si esta en movimiento
    * El telescopio se para al llegar a la posicion pedida con mueve()
    */
    public boolean estaQuieto() {...}

    /**
     * Retorna la intensidad de la fuente de radio medida en vatios
     * Lanza EnMovimiento si el telescopio no esta quieto
     */
    public double intensidadRecibida() throws EnMovimiento {...}

}
```

Las coordenadas de un punto en el cielo se dan en forma de dos ángulos, *acimut* y *elevación*, que especifican la orientación del radiotelescopio en grados. El *acimut* se mide con el cero apuntando al norte, y la *elevación* con el cero apuntando al horizonte.

Se dispone también de una clase que almacena un catálogo de las fuentes de radio conocidas:

```
public class Catalogo
{
    /**
     * Constructor que lee el catalogo del fichero indicado
     */
    public Catalogo(String nombreFichero) {...}

    /**
     * Busca en el catalogo una fuente en las coordenadas indicadas
     * y retorna su nombre si la encuentra.
     * Retorna "-" si no lo encuentra.
     */
    public String busca (double acimut, double elevacion) {...}

}
```

Para guardar los datos de una fuente de radiación se utilizará la clase `FuenteRadio`. Observar que sus atributos son públicos, ya que el objetivo de esta clase es meramente el de agrupar datos.

```
public class FuenteRadio
{
    public double intensidad; // vatios
    public double acimut; // grados
    public double elevacion; // grados
    public String nombre;
```

```
    /**
     * Constructor al que se le pasan los datos de la fuente de radio
     * intensidad en vatios, angulos en grados
     */
    public FuenteRadio(double intensidad, double acimut,
                       double elevacion, String nombre)
    {...}

}
```

Utilizando estas tres clases ya realizadas, se pretende escribir otras dos clases (una completa y otra en parte) que permitan usar el radiotelescopio para hacer un barrido de un sector del cielo, y, si localiza fuentes de radio, que las almacene en una lista.

La clase `ListaFuentes` permite almacenar una lista de objetos `FuenteRadio` en un array. A continuación se muestran sus atributos y la especificación de sus métodos:

```
public class ListaFuentes
{

    private FuenteRadio[] fuente; // array para guardar las fuentes
    private int num=0;            // numero actual de fuentes almacenadas

    /**
     * Constructor al que se le pasa el tamaño maximo de la lista
     */
    public ListaFuentes(int max) {...}

    /**
     * Anade la fuente f a la lista
     * Lanza NoCabe si se ha alcanzado el maximo
     */
    public void anade(FuenteRadio f) throws NoCabe {...}

    /**
     * Escribe la lista en el fichero de texto cuyo nombre se indica
     */
    public void escribeFichero (String nombreFichero) {...}

}
```

La clase `Barrido` tiene la operación `busca()` que permite hacer el barrido del cielo:

```
public class Barrido
{
    /**
     * Constructor al que se le pasa el telescopio, el catalogo, y la
     * lista de fuentes a usar
     */
    public Barrido (RadioTelescopio tele, Catalogo cat, ListaFuentes lista)
    {...}

    /**
     * Hace una busqueda de fuentes por el sector del cielo
     */
    public void busca (double aciMax, double aciMin,
                       double eleMax, double eleMin, double intensidadMin)
                       throws RangoIncorrecto
    {...}

}
```

Se pide:

a) Escribir la clase `Barrido` (3.5 puntos). Deberá tener los siguientes elementos:

- *Atributos*: referencias a objetos de las clases `RadioTelescopio`, `Catalogo` y `ListaFuentes`, que son los objetos a usar en el método `busca()`.
- *Constructor*: Copia en los atributos de la clase las referencias a los objetos que se le pasan como parámetros.
- *Método* `busca()`: Hace una búsqueda de fuentes de radio por el sector del cielo entre el acimut máximo (`aciMax`) y acimut mínimo (`aciMin`), y la elevación maxima (`eleMax`) y elevación mínima (`eleMin`):
  - En primer lugar, el método lanza `RangoIncorrecto` si el acimut o elevación máximo es menor que el mínimo o si la elevación mínima es menor que cero o la máxima mayor que 90 grados.
  - Para buscar las fuentes el método hace un lazo en el que la elevación varía desde la mínima a la máxima, en incrementos de 0.2 grados. Internamente a ese lazo, hace otro lazo en el que el acimut varía entre el mínimo y el máximo, con incrementos de 0.6 grados. Dentro de ese nuevo lazo espera (con otro lazo) hasta que el telescopio está quieto, y luego lee la intensidad recibida (con `intensidadRecibida()`).
  - Continuando en el interior del doble lazo, si la intensidad recibida es mayor que la intensidad mínima (`intensidadMin`) busca su nombre en el catálogo (pasándole al método `busca()` del catálogo el acimut y elevación actuales), crea un objeto de la clase `FuenteRadio` para guardar los datos de la fuente de radio detectada, y almacena ese objeto en la lista con `anade()`.
  - Si durante estas últimas instrucciones se lanzase `EnMovimiento` o `NoCabe`, se pondrá un mensaje indicativo en la pantalla, pero el doble lazo continuará.

b) Escribir el método `escribeFichero()` de la clase `ListaFuentes` (1.5 puntos):

- Este método escribe en el fichero de texto cuyo nombre se le pasa como parámetro una lista de todas las fuentes de radiación cuyo nombre es igual a `"-"` (es decir, las que no estaban en el catálogo). Cada fuente va en una línea, con el siguiente formato (los datos son un ejemplo):

  ```
  acimut=30.5  elevacion=23.7   intensidad=0.00004
  ```

  Recordar que para usar ficheros es preciso tratar la excepción `IOException` del paquete `java.io`. Hacerlo poniendo un mensaje en la pantalla.

Las excepciones están definidas en tres clases escritas aparte de la forma:

```
public class EnMovimiento extends Exception {}

public class NoCabe extends Exception {}

public class RangoIncorrecto extends Exception  {}
```

a) clase Barrido

```
public class Barrido
{
    private RadioTelescopio tele;
    private Catalogo cat;
    private ListaFuentes lista;

    /**
     * Constructor al que se le pasa el telescopio, el cat·logo, y la
     * lista de fuentes a usar
     */
    public Barrido (RadioTelescopio tele, Catalogo cat,
                    ListaFuentes lista)
    {
        this.tele=tele;
        this.cat=cat;
        this.lista=lista;
    }

    /**
     * Hace una busqueda de fuentes por un sector del cielo
     */
    public void busca (double aciMax, double aciMin,
                  double eleMax, double eleMin, double intensidadMin)
                        throws RangoIncorrecto
    {
        final double saltoEle=0.2;
        final double saltoAci=0.6;

        if (aciMax<aciMin || eleMax<eleMin ||
            eleMin<0.0 || eleMax>90.0)
        {
            throw new RangoIncorrecto();
        }
        for (double ele=eleMin; ele<=eleMax; ele=ele+saltoEle) {
            for (double aci=aciMin; aci<=aciMax; aci=aci+saltoAci) {
                tele.mueve(ele,aci);
                do{
                    // no hago nada, esperando a que se pare
                } while (!tele.estaQuieto());
                try {
                   double intensidad=tele.intensidadRecibida();
                   if (intensidad>intensidadMin) {
                       String nombre=cat.busca(aci,ele);
                       FuenteRadio f=new
                          FuenteRadio(intensidad,aci,ele,nombre);
                       lista.anade(f);
                   }
                } catch (EnMovimiento e) {
                   System.out.println("Error en telescopio");
                } catch (NoCabe e) {
                   System.out.println("No cabe fuente en la lista");
                }
            }
        }
    }
}
```

b) método escribeFichero():

```
/**
  * Escribe la lista en el fichero de texto cuyo nombre se indica
  */
 public void escribeFichero (String nombreFichero) {
     try {
          FileWriter fich=new FileWriter(nombreFichero);
          PrintWriter f = new PrintWriter(fich);
          for (int i=0; i<num; i++) {
              if (fuente[i].nombre.equals("-")) {
                   f.println("Acimut="+fuente[i].acimut+
                             " Elevacion="+fuente[i].elevacion+
                             "Intensidad="+fuente[i].intensidad);
              }
          }
          f.close();
     } catch (IOException e) {
          System.out.println("Error con la entrada/salida");
     }
 }
```