

Solución al Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Junio 2006

Cuestiones (5 cuestiones, 5 puntos en total)

1) Se dispone de la siguiente clase enumerada:

```
public enum Rango
{
    positivo, cero, negativo
}
```

Escribir una clase en Java con lo siguiente

- rango: un atributo de la clase Rango
- mide: un atributo que sea un número real
- Medida(double v): un constructor al que se le pasa como parámetro el número real v; el constructor copia v en mide, y pone rango a su valor apropiado según el signo y valor de v.

```
public class Medida
{
    // atributos privados
    private double mide;
    private Rango rango;

    /**
     * Constructor
     */
    public Medida(double v)
    {
        mide=v;
        if (v<0) {
            rango=Rango.negativo;
        } else if (v>0) {
            rango=Rango.positivo;
        } else {
            rango=Rango.cero;
        }
    }
}
```

2) Escribir un método estático en Java que sirva para invertir el orden de los elementos de un array, y que responda al siguiente pseudocódigo:

```
metodo invertir(double[] a) lanza Vacio
    si (tamaño de a es cero) {
        lanza excepción Vacio
    }
    lazo para cada i desde 0 hasta (tamaño de a)/2 - 1
        intercambia a[i] con a[(tamaño de a) -(i+1)]
    fin lazo
fin metodo
```

La división (tamaño de a)/2 es entera. La excepción Vacio es una clase aparte definida así:

```
public class Vacio extends Exception {}
```

```

public static void invierte(double[] a) throws Vacio
{
    if (a.length==0) {
        throw new Vacio();
    }
    for (int i=0;i<=(a.length)/2 - 1;i++) {
        double temp=a[i];
        a[i]=a[a.length -(i+1)];
        a[a.length -(i+1)]=temp;
    }
}

```

- 3) Indicar cuál es el ritmo de crecimiento del tiempo de ejecución del algoritmo anterior utilizando la notación $O(n)$. ¿Qué es n en este caso?

En este caso, el tamaño del problema, n , es el tamaño del array a .

El método dispone de dos instrucciones en el nivel más externo: una instrucción condicional `if`, y un bucle `for`.

La instrucción `if` contiene una instrucción simple ($O(1)$) y por tanto es $O(1)$.

Para analizar el bucle `for` externo, multiplicamos el número de veces que se hace ($n/2$) por lo que tarda lo que tiene dentro. En su interior hay tres instrucciones simples, que por la regla de las sumas son $O(1)$. Por tanto, el bucle `for` externo tiene un ritmo de crecimiento que es $O(n/2)O(1)$, que por la regla de los productos es $O(n)$

El tiempo total es por tanto $O(1)+O(n)$, y por la regla de las sumas es $O(n)$.

- 4) Indicar en menos de 8 líneas las principales diferencias entre un intérprete de órdenes de un sistema operativo, y un gestor gráfico de ficheros.

El intérprete de órdenes de un sistema operativo es un programa que permite al usuario introducir órdenes por teclado con las que se pueden gestionar ficheros (por ejemplo borrar, copiar o moverlos), ejecutar programas, y administrar el sistema.

El gestor gráfico de ficheros permite hacer básicamente lo mismo, pero usando una interfaz gráfica basada en ventanas, botones, menús, y el uso del ratón.

La diferencia fundamental es que en el intérprete se requiere un aprendizaje de un conjunto de órdenes, mientras que en el gestor gráfico el manejo es más intuitivo. El gestor gráfico requiere más recursos.

- 5) Se dispone de una clase para tratamiento de imágenes con la siguiente interfaz

```

public class Imagen
{
    public void calcula() {}
    public void optimiza() throws Falla {}
    public void filtra() throws Falla {}
    public void suaviza() throws Falla {}
    public void escribeValor(boolean conFallo) {}
}

```

Escribir un método estático de una clase aparte, al que se le pasa como parámetro un objeto de la clase Imagen, y que llama sucesivamente a los métodos arriba expuestos, en el orden en que aparecen, con las siguientes particularidades:

- si se lanza Falla, al llamar a optimiza, no llamar a filtra ni a suaviza
- si se lanza Falla, al llamar a filtra, no llamar a suaviza
- en todos los casos, tanto si se lanza Falla como si no, debe invocarse a escribeValor

```
public static void procesa(Imagen img) {
    boolean conFallo=false;
    img.calcula();
    try {
        img.optimiza();
        img.filtra();
        img.suaviza();
    } catch (Falla e) {
        conFallo=true;
    }
    img.escribeValor(conFallo);
}
```

Solución al Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Junio 2006

Problema (5 puntos)

Se desea hacer una clase para almacenar los datos obtenidos en un experimento de colisiones entre partículas. Se dispone para ello de la siguiente clase ya realizada, y que permite guardar vectores de tres dimensiones, con sus coordenadas x,y,z.

```
public class Vector3D{  
  
    /**  
     * Constructor al que se le pasan las coordenadas x,y,z  
     */  
    public Vector3D(double x, double y, double z) {...}  
  
    /**  
     * retorna la coordenada x  
     */  
    public double coordX() {...}  
  
    /**  
     * retorna la coordenada y  
     */  
    public double coordY() {...}  
  
    /**  
     * retorna la coordenada z  
     */  
    public double coordZ() {...}  
}
```

Se dispone asimismo de la clase Particula, que almacena los datos del movimiento de una partícula:

```
public class Particula {  
  
    /**  
     * Constructor al que se le pasa la posicion inicial (vector en m)  
     * la velocidad inicial (vector en m/s) y el radio de curvatura  
     * de su trayectoria (m)  
     */  
    public Particula(Vector3D posInicial, Vector3D velInicial,  
                    double radioCurvatura) {...}  
  
    /**  
     * Retorna la posicion inicial  
     */  
    public Vector3D posInicial() {...}  
  
    /**  
     * Retorna la velocidad inicial  
     */  
    public Vector3D velInicial() {...}  
  
    /**  
     * Retorna el radio de curvatura de la trayectoria  
     */  
    public double radioCurvatura() {...}  
}
```

Lo que se pide es escribir la clase Resultados que responde a la siguiente interfaz:

```
/**
 * Clase que permite almacenar datos sobre movimiento de
 * partículas resultantes de un experimento de colision
 */
public class Resultados
{
    /**
     * Constructor que crea la lista de partículas con el
     * tamaño máximo especificado, y la deja vacía
     */
    public Resultados(int max) {...}

    /**
     * Añade una partícula a la lista. Lanza NoCabe si se excede
     * el tamaño máximo
     */
    public void anadeParticula(Particula p) throws NoCabe {...}

    /**
     * Retorna un objeto de la clase resultados con una lista de
     * las partículas cuya velocidad (módulo del vector velocidad
     * inicial) es superior a min. Lanza vacío si en el objeto
     * original no hay ninguna partícula
     */
    public Resultados filtraVelocidad(double min) throws Vacio {...}

    /**
     * Igual que el anterior, pero la lista retornada contiene solo
     * aquellas partículas cuya velocidad es superior a min, y el
     * radio de curvatura es inferior a radioMax
     */
    public Resultados filtraVelocidadYRadio
        (double min, double radioMax) throws Vacio {...}

    /**
     * Escribe en el fichero de texto cuyo nombre se indica
     * los radios de curvatura de todas las partículas de la lista
     * uno por línea.
     */
    public void escribeRadios(String nombreFichero) {...}
}
}
```

La clase debe tener como atributos un array de partículas (llamado part) y un entero que indica cuántas partículas hay almacenadas en el array en este momento (llamado num)

Las excepciones están declaradas en clases aparte de la forma:

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

Nota: Cada parte del problema se valorará en función de su dificultad

```
import java.io.*;
/**
 * Clase que permite almacenar datos sobre movimiento de
 * partículas resultantes de un experimento de colision
 */
public class Resultados
{
    // atributos privados
    private int num; // número de partículas almacenadas
    private Particula[] part; // lista de partículas
}
```

```

/**
 * Constructor que crea la lista de particulas con el
 * tamano maximo especificado, y la deja vacia
 */
public Resultados(int max)
{
    part= new Particula[max];
    num=0;
}

/**
 * Anade una particula a la lista. Lanza NoCabe si se excede
 * el tamano maximo
 */
public void anadeParticula(Particula p) throws NoCabe
{
    if (num==part.length) {
        throw new NoCabe();
    } else {
        part[num]=p;
        num++;
    }
}

/**
 * Retorna un objeto de la clase resultados con una lista de
 * las particulas cuya velocidad (módulo del vector velocidad
 * inicial) es superior a min. Lanza vacio si en el objeto
 * original no hay ninguna particula
 */
public Resultados filtraVelocidad(double min) throws Vacio
{
    if (num==0) {
        throw new Vacio();
    }
    // usamos como tamano num, pues la lista filtrada es
    // de tamaño menor o igual a num
    Resultados listaNueva=new Resultados(num);
    // Es obligatorio tratar NoCabe
    try {
        // Recorremos la lista original
        for(int i=0; i<num; i++) {
            // Calculamos el modulo de la velocidad inicial
            Vector3D v=part[i].velInicial();
            double vx=v.coordX();
            double vy=v.coordY();
            double vz=v.coordZ();
            // El modulo se podria calcular en un método aparte
            double modulo=Math.sqrt(vx*vx+vy*vy+vz*vz);
            // comprobamos condicion de filtrado
            if (modulo>min) {
                listaNueva.anadeParticula(part[i]);
            }
        }
    } catch (NoCabe e) {
        // no hacemos nada, pues sabemos que cabe
    }
    return listaNueva;
}

/**
 * Igual que el anterior, pero la lista retornada contiene solo
 * aquellas particulas cuya velocidad es superior a min, y el
 * radio de curvatura es inferior a radioMax
 */
public Resultados filtraVelocidadYRadio
(double min, double radioMax) throws Vacio
{
    // Es igual que el anterior pero cambia la condición de filtrado
    // Antes: if (modulo>min) {...

```

```

        // Ahora:
        if (modulo > min && part[i].radioCurvatura() < radioMax) {...
    }

    /**
     * Escribe en el fichero de texto cuyo nombre se indica
     * los radios de curvatura de todas las partículas de la lista
     * uno por línea.
     */
    public void escribeRadios(String nombreFichero)
    {
        try {
            // Abrir el fichero
            FileWriter f = new FileWriter(nombreFichero);
            PrintWriter p = new PrintWriter(f);
            // recorrer todas las partículas
            for (int i=0; i<num; i++) {
                // escribir el radio de curvatura
                p.println(+part[i].radioCurvatura());
            }
            // Cerrar el fichero
            p.close();
        } catch (IOException e) {
            System.out.println("Error inesperado: "+e);
        }
    }
}

```