Soluciones al Examen de Fundamentos de Computadores y Lenguajes

Examen Parcial. Junio 2004

Cuestiones (5 cuestiones, 5 puntos en total)

1) Se desea crear una clase para hacer cálculos sobre el movimiento de un cuerpo en un plano inclinado. Crear la clase con los siguientes atributos privados:
   - ángulo del plano inclinado (en grados)
   - masa del cuerpo (kg)
   - coeficiente de rozamiento (sin unidades)
   - velocidad lineal (metros/segundo)
   - posición (en metros)

   Todos ellos serán números reales, excepto la posición que será un array con dos casillas reales, para guardar las coordenadas *x* e *y* del cuerpo.

   Crear asimismo un constructor al que se le pasen como parámetros los valores iniciales de los dos últimos atributos, para que los copie en ellos.

```
public class CuerpoEnPlanoInclinado {

  private double angulo;    // grados
  private double masa;      // Kg
  private double coef;      // coeficiente de rozamiento
  private double velLineal; // m/s
  private double[] posicion=new double[2]; // metros

  public CuerpoEnPlanoInclinado
     (double velLineal, double posx, double posy)
  {
    this.velLineal=velLineal;
    posicion[0]=posx;
    posicion[1]=posy;
  }
}
```

2) Indicar qué pasos seguirías para añadir a la clase anterior un método que escriba en un fichero de texto los tres primeros atributos del objeto, usando un formato como el indicado en el siguiente ejemplo:

```
Angulo : 3.0 grados
Masa   : 5.3 Kg
Coef   : 1.2
```

   El nombre del fichero se le pasaría como parámetro al método. No tratar errores.

Crearíamos un método que no devuelve nada y al que se le pasa como parámetro un String que contiene el nombre del fichero. Este método hará lo siguiente:

- Crear un objeto de la clase `FileWriter` para salida de texto, usando en el constructor el nombre del fichero
- Crear un objeto de la clase `PrintWriter`, para salida de texto, usando en el constructor el objeto anterior; llamemos `p` a este nuevo objeto.

continúa...

- Escribir cada uno de los tres parámetros en el fichero, usando tres veces el método `println()` del objeto `p`, y pasándole un string que será la concatenación del nombre del parámetro, el atributo que contiene su valor, y las unidades
- Cerrar el fichero con el método `close()` del objeto `p`.

3) Se dispone de una clase que contiene los siguientes atributos que almacenan 100 valores de una función real llamada *valor*(*x*):

```
public class Interpolacion
{
    public final int MAX=100;
    private double[] x= new double[MAX];
    private double[] valor= new double[MAX];
 }
```

El array `x` contiene valores reales y el array `valor` tiene en cada casilla `i` el valor de la función *valor* en el punto `x[i]`. Se pide escribir en Java un método que corresponda al siguiente pseudocódigo que realiza la interpolación lineal entre dos de los puntos almacenados, devolviendo el valor de la función que se corresponde con un valor de *x* llamado `pos`, que se le pasa como parámetro:

```
método interpola (pos : número real) retorna un valor real
    i=0;
    lazo mientras i menor que MAX-1 y pos > x[i]:
        incrementa i en 1
    fin de lazo
    si i es 0: retorna primera casilla de valor
    si i > MAX: retorna última casilla de valor
   en caso contrario:retorna interpolación entre casillas i-1 e i
fin del método
```

Siendo el valor de la interpolación:

$$interpola(i-1,i) = v_{i-1} + (v_i - v_{i-1}) \cdot \frac{pos - x_{i-1}}{x_i - x_{i-1}}$$

donde x_i es la casilla `i` del array `x` y v_i es la casilla `i` del array `valor`.

```
public double interpola(double pos) {
   int i=0;
   while (i<=(MAX-1) && pos>x[i]) {
       i++;
   }
   if (i==0) return valor[0];
   if (i>MAX) return valor[MAX-1];
   return valor[i-1]+(valor[i]-valor[i-1])*
          (pos-x[i-1])/(x[i]-x[i-1]);
}
```

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

Vamos a numerar las instrucciones para luego hacer el análisis:

```
    public double interpola(double pos) {
(1)    int i=0;
(2)    while (i<=(MAX-1) && pos>x[i]) {
(3)        i++;
       }
(4)    if (i==0) {
(5)       return valor[0];
       }
(6)    if (i>MAX) {
(7)       return valor[MAX-1];
       }
(8)    return valor[i-1]+(valor[i]-valor[i-1])*
              (pos-x[i-1])/(x[i]-x[i-1]);
    }
```

Las instrucciones (1)(3)(5)(7) y (8) son simples cálculos o asignaciones, luego son O(1). Aunque la (8) es compleja, es una sucesión de instrucciones simples que por la regla de las sumas es O(1)

El lazo while (instrucción (2)) se ejecuta en el peor caso un número de veces igual a MAX. Esto multiplicado por lo de dentro (instrucción (3)) es O(MAX) según la regla de los productos

Las instrucciones (4) y (6) tanto si se ejecuta el if, como si no, son O(1), dado que lo que se ejecuta en su interior es O(1) y la comparación también

La suma de todo es por tanto O(1)+O(MAX)+O(1)+O(1)+O1(). Por la regla de las sumas, el método es O(MAX).

---

5) Se dispone de una clase que sirve para guardar un vector de números reales, y que responde a la siguiente especificación:

```
public class Vector
{
    // cambia el tamaño del vector
    public void cambiaTamano(int nuevoTamano) {...}

    // retorna el elemento i del vector
    public double elemento(int i) throws NoExiste {...}

    // cambia el elemento i del vector al valor indicado
    public void cambia(int i, double valor) throws NoExiste {...}
}
```

Donde `NoExiste` es una excepción declarada en una clase aparte, y que se lanza si el elemento `i` no existe.

Añadir a la clase un nuevo método con los mismos parámetros que `cambia`, y que haga lo siguiente:

- llamar a `cambia`
- si se lanza `NoExiste` tratar esta excepción cambiando el tamaño del vector al

valor de `i`, con `cambiaTamano`, y luego volver a llamar a `cambia`.

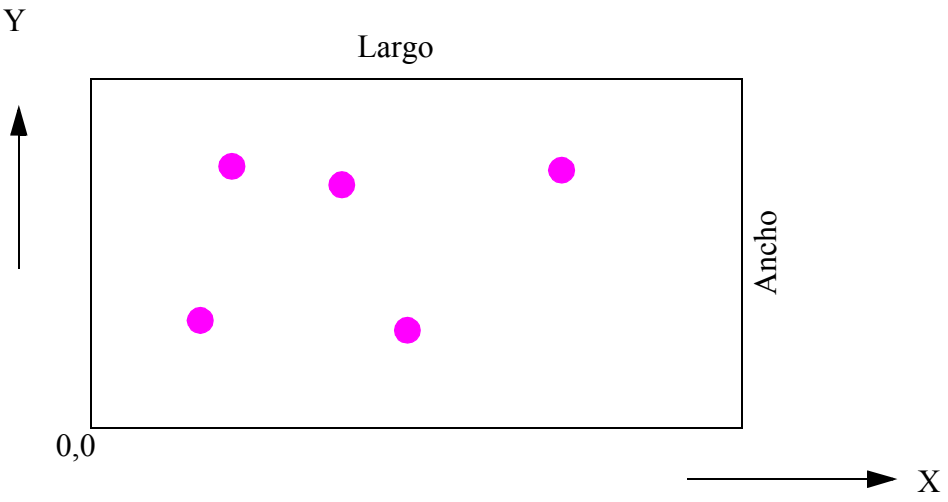
```
public void cambiaSinFallo(int i, double valor) {
   try {
      cambia(i,valor);
   } catch (NoExiste e1) {
      cambiaTamano(i);
      try {
         cambia(i,valor);
      } catch (NoExiste e2) {
         // no hacemos nada, ya que esta vez no falla
      }
   }
}
```

Examen de Fundamentos de Computadores y Lenguajes

Examen Parcial. Junio 2004

Problema (5 puntos)

Se desea implementar en Java una clase que sirva para guardar los datos del movimiento de cinco bolas de billar en una mesa como la que se muestra en la figura. El origen de coordenadas está en la esquina inferior izquierda de la mesa en el dibujo.



Para ello se dispone de una clase ya realizada que se llama `Bola` y sirve para guardar los datos de una bola. Su interfaz es:

```
public class Bola
{
    public static final int
       paredALoLargo=0,   // identifica una pared paralela al eje X
       paredALoAncho=1;   // identifica una pared paralela al eje Y

    public static final double radio=0.05; //radio de la bola (m)

    // Constructor con posiciones(m) y velocidades(m/s) iniciales
    public Bola(double posX, double posY, double velX, double velY)
    {...}

    // retorna la coordenada x de la posicion de la bola
    public double posX() {...}

    // retorna la coordenada y de la posicion de la bola
    public double posY() {...}

    // Calcula los efectos del choque con una pared
    // (el tipoPared indica si es paralela al eje X o Y u horizontal)
    public void choquePared(int tipoPared) {...}

    // Calcula los efectos de un choque con otra bola
    public void choqueBola(Bola otra) {...}
```

```
    // Distancia entre dos bolas
    public double distancia(Bola otra) {...}

    // Avanza la bola en un intervalo t (seg), en línea recta
    public void avanza (double t) throws TiempoIncorrecto {...}
}
```

La clase `TiempoIncorrecto` es una excepción declarada aparte de la forma:

```
public class TiempoIncorrecto extends Exception {}
```

Lo que se pide es crear una nueva clase llamada `MesaBillar`, con la siguiente interfaz:

```
public class MesaBillar
{
    public MesaBillar (double largo, double ancho) {...}

    public void avanza(double t) {...}

    public void dibuja() {...} // este método no se pide

}
```

La clase debe tener los siguientes atributos privados:

- largo y ancho de la mesa (en metros)
- array con cinco bolas, de la clase `Bola`

La función que hace cada método es

- *Constructor*: Se le pasa el largo de la mesa y el ancho, en metros, para que los guarde en los respectivos atributos. Luego crea las cinco bolas (con `new`) y les pasa en el constructor los siguientes datos:
  - `posX`: número aleatorio entre 0 y largo
  - `posY`: número aleatorio entre 0 y ancho
  - `velX`: número aleatorio entre 0 y 0.5
  - `velY`: número aleatorio entre 0 y 0.5

  Nota: Para obtener números aleatorios entre 0 y 1 se usa la función `random`() de la librería matemática.
- `avanza`(): Se le pasa un tiempo en segundos y hace lo siguiente:
  - Avanza el estado de cada bola en el tiempo indicado llamando al método `avanza`() de cada bola. Si se lanza `TiempoIncorrecto` se pone un único mensaje en pantalla (no uno por cada bola) y se acaba el método.
  - Detecta choques con las paredes. El choque con una pared "a lo ancho" se da si la coordenada *x* de la bola vale menos que 0 o más que el atributo `largo`. El choque con una pared "a lo largo" se da si la coordenada *y* de la bola vale menos que 0 o más que el atributo `ancho`. Si se detecta choque, llamar a `choquePared`() indicando el tipo de pared apropiado.
  - Detecta choques con otras bolas. El choque entre dos bolas ocurre si la distancia entre ellas es menor que dos veces el radio de la bola. Si se detecta choque entre dos bolas, llamar al método `choqueBola`() de una de ellas, poniendo la otra bola como argumento. Para detectar todos los posibles choques utilizar el siguiente

algoritmo:

```
lazo para i desde 1 hasta 4
    lazo para j desde i+1 hasta 5
      si (distancia entre bolas i y j < 2*radio) entonces
            llamar a choqueBola()
        fin si
    fin lazo
fin lazo
```

- `dibuja`(): Dibuja la mesa y las bolas en una ventana. Este método se especifica para dar una visión completa de la clase, pero no se pide en el examen.

```java
/**
 * Contiene los datos de una mesa de billar y sus bolas
 */

public class MesaBillar
{
    private double largo, ancho;
    private Bola[] bola=new Bola[5];

    /**
     * Al constructor se le pasa el largo de la mesa (coordenada x)
     * y el ancho (coordenada y), en metros
     */
    public MesaBillar (double largo, double ancho) {
        this.largo=largo;
        this.ancho=ancho;
        for (int i=0;i<5;i++) {
          bola[i]= new Bola(Math.random()*largo,Math.random()*ancho,
                                Math.random()*0.5,Math.random()*0.5);
        }
    }

    /**
     * Avanza las bolas un tiempo t (seg), detectando choques
     */
    public void avanza(double t) {
        try {
           // avanza el tiempo en cada bola
           for (int i=0;i<5;i++) {
               bola[i].avanza(t);
           }
        } catch (TiempoIncorrecto e) {
           System.out.println("Tiempo Incorrecto");
           return;
        }
        // detecta choques con las paredes
        for (int i=0;i<5;i++) {
            if (bola[i].posX()<0 || bola[i].posX()>largo) {
                bola[i].choquePared(Bola.paredALoAncho);
            }
            if (bola[i].posY()<0 || bola[i].posY()>ancho) {
                bola[i].choquePared(Bola.paredALoLargo);
            }
        }
        // detecta choques con otras bolas
        for (int i=0;i<4;i++) {
           for (int j=i+1;j<5;j++) {
               if (bola[i].distancia(bola[j])<2*Bola.radio){
                   bola[i].choqueBola(bola[j]);
               }
           }
        }
    }

    /**
     * Dibuja la mesa y las bolas
     */
    public void dibuja() {...}

}
```