Soluciones al Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Septiembre 2003

Cuestiones (5 cuestiones, 5 puntos en total)

1) Se dispone del siguiente array de números reales ya creado. Escribir un fragmento de programa que ponga todas las casillas pares a cero y las impares a uno.

```
double[] a=new double [1000];
```

Una solución consiste en hacer dos lazos, uno para los números pares y otro para los impares:

```
for (int i=0; i<1000; i+=2) {
  a[i]=1.0;
}
for (int i=1; i<1000; i+=2) {
  a[i]=0.0;
}
```

Otra solución es hacer un único lazo y distinguir las casillas pares de las impares con la operación módulo: `i%2`.

2) Se dispone de la siguiente clase para almacenar los datos de un cuadrilátero rectángulo; los campos `coordx` y `coordy` son, respectivamente, las coordenadas X e Y del centro del cuadrilátero:

```
public class Cuadrilatero {

  private double ancho, alto;
  private double coordx, coordy;

  public class ErrorNumerico extends Exception {}

}
```

Se pide añadirle un constructor al que se le pasen como parámetros los cuatro datos que definen al cuadrilátero (`alto`, `ancho`, `coordx`, `coordy`), para almacenarlos en los campos privados.

También se pide añadirle un método sin parámetros que retorne el área del cuadrilátero.

```
public Cuadrilatero(double ancho, double alto,
                    double coordx, double coordy)
{
  this.ancho=ancho;
  this.alto=alto;
  this.coordx=coordx;
  this.coordy=coordy;
}

public double area () {
  return ancho*alto;
}
```

3) Añadir a la clase `Cuadrilatero` del ejercicio anterior un método que cambie el tamaño (pero no el centro) del cuadrilátero, escalándolo en un factor real que se le pasará al método como parámetro. Si el factor es negativo, se deberá lanzar `ErrorNumerico`. Si no, se multiplica la altura y la anchura por el factor. El método no retorna nada.

```java
public void escala (double factor) throws ErrorNumerico {
  if (factor<0.0) {
    throw new ErrorNumerico();
  }
  ancho=ancho*factor;
  alto=alto*factor;
}
```

4) Explicar cuáles son las diferencias entre ficheros de datos y directorios dentro de un sistema operativo. Indicar por qué se dice que el sistema de ficheros es jerárquico. ¿Sería posible un sistema de ficheros sin directorios? ¿Sería útil?

   Contestar de forma razonada pero brevemente (menos de 15 líneas de texto).

   En un sistema operativo los ficheros se utilizan para almacenar información en la memoria secundaria del computador, que habitualmente es de alta capacidad y no volátil. Se distinguen dos tipos de ficheros: de datos y directorios. Los ficheros de datos contienen información tal como documentos, números, bases de datos. También pueden contener programas, descritos mediante una secuencia de instrucciones. Los directorios en cambio contienen ficheros, tanto de datos como otros directorios. Se utilizan para organizar la información, que puede ser muy voluminosa, en una estructura jerárquica. Un sistema de ficheros sin directorios sería posible, pero poco útil al mezclarse toda la información y los programas en un único lugar.

5) Se dispone del siguiente algoritmo expresado mediante pseudocódigo. Indicar brevemente (5 líneas o menos) qué hace, e indicar su tiempo de ejecución usando la notación $O(n)$. ¿Qué sería en este caso n?:

```
algoritmo edades
     entrada: p[]: array de objetos de la clase Persona;
     salidas: niños :_ entero
              jovenes : entero
              adultos : entero
              mayor_que_media : entero
     variables : suma_edades, media : reales


comienzo
    inicializa salidas y variables a cero
    lazo para i desde 1 hasta el numero de elementos de p
       suma=suma + edad de la persona i
       si edad de la persona i > 30
           adultos++
       si no, si edad de la persona i > 15
           jovenes++
       si no
           niños++
       fin de "si"
    fin de lazo;
    media=suma/numero de elementos de p
    lazo para i desde 1 hasta el numero de elementos de p
       si edad de la persona i>media
            mayor_que_media++
       fin de "si"
```

```
    fin de lazo;
fin de algoritmo
```

Al algoritmo se le pasa un conjunto de personas de las que se puede conocer la edad, y cuenta el número de adultos, jóvenes y niños. Además calcula la edad media y obtiene el número de personas cuya edad supera la media.

Al algoritmo se organiza con dos lazos ejecutados uno después de otro, cada uno de los cuales se ejecuta `n` veces, siendo `n` el número de elementos del conjunto `p` de personas. Entre los dos lazos se ejecuta una instrucción simple que es O(1). Tanto en el primer lazo como en el segundo hay instrucciones condicionales e instrucciones simples, todas ellas O(1), por lo que por la regla de las sumas el contenido del lazo es O(1). Cada lazo es por tanto O(n). Por la regla de las sumas el algoritmo total es O(n+1+n) que es O(n), donde recordamos que `n` es el número de personas de `p`.

Soluciones al Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Septiembre 2003

Problema (5 puntos)

Se desea hacer parte del software de ayuda a la navegación de un barco. El océano dispone de conjuntos de boyas capaces de medir la altura de las olas, y de un sistema de radio con el que comunicarse. El barco dispone de un equipo de radio con el que se comunica con esas boyas. El software del barco tiene algunas partes ya escritas, que se describen a continuación.

La clase `Coordenadas` almacena las coordenadas (latitud y longitud) de un punto en el globo terráqueo. Tiene un método que retorna la distancia entre dos puntos, en millas náuticas:

```
public class Coordenadas {

  public double lat,lon;

  public static double distancia(Coordenadas C1, Coordenadas C2) {...}
}
```

La clase `Boya` sirve para guardar los datos de una boya:

```
public class Boya {

  private String myId;
  private Coordenadas myPos;

  /** Constructor, al que se le pasan el identificador de la boya,
      y su posicion geografica */
  public Boya(String id, Coordenadas pos) {...}

  /** Retorna el identificador de la boya */
  public String id() {...}

  /** Retorna la posicion geografica de la boya */
  public Coordenadas pos() {...}
}
```

La clase `Radio` permite operar con la radio del barco:

```
public class Radio {

  /** Envia un mensaje a la boya b, espera la contestacion, y retorna
      la altura de las olas medida por esa boya. Si la boya no contesta,
      lanza NoContesta */
  public double alturaOlas(Boya b) throws NoContesta {...}

  /** Envia un mensaje a la boya cuyo identificador es id, espera la
      contestacion, y retorna la posicion geografica de esa boya.
      Si la boya no contesta, lanza NoContesta */
  public Coordenadas posicion(String id) throws NoContesta { ... }

  /** Devuelve el identificador de una boya que se ha situado en el
      radio de alcance de la radio del barco. Si no hay ninguno,
      retorna null */
  public String localiza() {...}

  public class NoContesta extends Exception {}
}
```

Lo que se pide es escribir la clase `SistemaBoyas`, que almacena la información sobre las boyas del sistema y tiene operaciones que facilitan el uso. La clase debe obedecer a la siguiente especificación:

```
public class SistemaBoyas {

  public static final int MAX=100;

  public SistemaBoyas(Radio radioSistema) {...}

  public void busca() throws Demasiadas{...}

  public double alturaMedia(Coordenadas posBarco) throws Pocas {...}

  public class Demasiadas extends Exception {}
  public class Pocas extends Exception {}

}
```

La clase `SistemaBoyas` tendrá los siguientes datos en campos privados;

- `num`: número de boyas almacenadas; inicialmente cero
- `boyas`: Array de boyas, en número igual a `MAX`. Se usan sólo las `num` primeras casillas
- `rad`: Objeto de la clase `Radio`, usado para la comunicación con las boyas

La función a realizar por los diferentes métodos es la siguiente:

- Constructor: almacena `radioSistema` en el campo `rad`.
- `busca()`: Busca nuevas boyas. Si el numero de boyas es `MAX` lanza `Demasiadas`. Si no, invoca al método `localiza`() de la radio del sistema. Si no devuelve ningún identificador, o si el identificador devuelto coincide con alguna de las boyas almacenadas en el sistema, no hace nada. En caso contrario, obtiene la posicion de la boya (con el método `posicion`()) y salvo que lance `NoContesta`, crea una nueva boya y la almacena en la primera casilla desocupada del array `boyas`, incrementando luego `num`.
- `alturaMedia()`: Obtiene la altura de las olas de las tres boyas mas cercanas al barco, cuyas coordenadas son `posBarco`. Retorna la media ponderada (según la distancia) de las tres. Si no hay tres o más boyas en el sistema, o si alguna no contesta, lanza `Pocas`. Para obtener las tres boyas más cercanas se usa este algoritmo:

```
d1=0; d2=0; d3=0;
lazo desde i=0 hasta num-1
   dist = distancia entre el barco y la boya i
   si (dist>d1)
      d3=d2; boya3=boya2;
      d2=d1; boya2=boya1;
      d1=dist; boya1=i;
   si no, si (dist>d2)
      d3=d2; boya3=boya2;
      d2=dist, boya2=i;
   si no, si (dist>d3)
      d3=dist; boya3=i
fin de lazo
```

  Para calcular la altura media ponderada, a, de las olas se usa la expresión:

$$a = \frac{a1 \cdot d2 \cdot d3 + a2 \cdot d1 \cdot d3 + a3 \cdot d1 \cdot d2}{d2 \cdot d3 + d1 \cdot d3 + d1 \cdot d2}$$

  donde $d1$, $d2$, y $d3$ son las distancias del barco a las boyas 1, 2 y 3 resultantes del algoritmo anterior, y $a1$, $a2$, $a3$ son las alturas de las olas de esas mismas boyas, obtenidas mediante el método `alturaOlas`().

```java
public class SistemaBoyas {

  public static final int MAX=100;

  private Boya[] boyas=new Boya[MAX];
  private Radio rad;
  private int num=0;

  /** Constructor, al que se le pasa la radio del sistema */
  public SistemaBoyas(Radio radioSistema) {
    rad=radioSistema;
  }

  /** Busca nuevas boyas. Si el numero de boyas es el maximo lanza
      demasiadas.  Si no, invoca al metodo localiza() de la radio del
      sistema. Si no devuelve ningun identificador, o si el
      identificador devuelto coincide con alguna de las boyas
      almacenadas en el sistema, no hace nada. Si no coincide, obtiene
      la posicion (con el metodo posicion()) y salvo que lance
      NoContesta, crea una nueva boya y la almacena en el sistema*/
  public void busca() throws Demasiadas{

    boolean encontrado=false;

    if (num==MAX) {
      throw new Demasiadas();
    }

    String idBoya=rad.localiza();
    if (idBoya!=null) {
      for (int i=0; i<num;i++) {
        if (idBoya.equals(boyas[i].id())) {
          encontrado=true;
          break;
        }
      }
      if (!encontrado) {
        try {
          Coordenadas pos=rad.posicion(idBoya);
          num++;
          boyas[num-1]=new Boya(idBoya,pos);
        } catch (Radio.NoContesta e) {
          // no hacemos nada
        }
      }
    }
  }

  /** Obtiene la altura de las olas de las tres boyas mas cercanas al
      barco, cuyas coordenadas son posBarco. Retorna la media ponderada
      (segun la distancia) de las tres. Si no hay tres o mas boyas en el
      sistema, o si alguna no contesta, lanza Pocas*/
  public double alturaMedia(Coordenadas posBarco) throws Pocas {

    double dist, dist1=0.0, dist2=0.0, dist3=0.0;
    double alt1, alt2, alt3;
    int boya1=0, boya2=0, boya3=0;

    if (num<3) {
      throw new Pocas();
    }
    for (int i=0; i<num; i++) {
      dist=Coordenadas.distancia(boyas[i].pos(),posBarco);
      if (dist>dist1) {
        dist3=dist2; boya3=boya2;
        dist2=dist1; boya2=boya1;
        dist1=dist;boya1=i;
      } else if (dist>dist2) {
        dist3=dist2; boya3=boya2;
```

```
          dist2=dist; boya2=i;
        } else if (dist>dist3) {
          dist3=dist; boya3=i;
          hay3=true;
        }
      }
      try {
        alt1=rad.alturaOlas(boyas[boya1]);
        alt2=rad.alturaOlas(boyas[boya2]);
        alt3=rad.alturaOlas(boyas[boya3]);
        return alt1*(dist2*dist3) +alt2*(dist1*dist3)+alt3*(dist1*dist2)/
          (dist1*dist2+dist2*dist3+dist1*dist3);
      } catch (Radio.NoContesta e) {
        throw new Pocas();
      }
    }

    public class Demasiadas extends Exception {}
    public class Pocas extends Exception {}

}
```