Soluciones del Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Junio 2003

Cuestiones (5 cuestiones, 5 puntos en total)

1) Escribir un fragmento de programa que haga lo siguiente
   - Declara una variable entera llamada `m` y almacena en ella el máximo entre `10` y otra variable que ya existe, llamada `n`
   - Declara y crea un array de `m` números reales
   - Inicializa todas las casillas del array al valor `-1.0`.

```
int m=Math.max(10,n);

double[] lista = new double [m];

for (int i=0; i<m; i++) {
  lista[i]=-1.0;
}
```

2) Crear una clase para almacenar un triángulo, guardando sus tres lados (*a*, *b*, y *c*) en campos privados. La clase tendrá los siguientes métodos, en los que no es preciso tratar ni detectar posibles errores:
   - constructor: deberá admitir como parámetros 3 números reales que representan los lados del triángulo, y almacenarlos en los campos del objeto.
   - método que retorna el área del triángulo según la expresión siguiente, en la que *p* es la mitad del perímetro:

$$A = \sqrt{p(p-a)(p-b)(p-c)}$$

```
public class Triangulo {

  private double a,b,c;

  public Triangulo(double a, double b, double c) {
    this.a=a;
    this.b=b;
    this.c=c;
  }

  public double area() {
    double p=(a+b+c)/2.0;
    return Math.sqrt(p*(p-a)*(p-b)*(p-c));
  }
}
```

3) Describir mediante pseudocódigo el siguiente algoritmo definido en lenguaje natural:

   "Se dispone de una matriz cuadrada de números reales de tamaño `n` x `n` y se desea hacer un método para intercambiar la fila `i` por la fila `j`. La matriz, `m`, así como los valores `n`, `i`, y `j` son parámetros del método. El método recorre en un lazo desde `0` hasta `n-1` las

diferentes columnas de la matriz. Para cada columna `k`, guarda el elemento de la fila `i` en una variable temporal, copia el elemento de la fila `j` en la fila `i`, y copia la variable temporal en el elemento de la fila `j`."

¿Cuál es su tiempo de ejecución en notación O(n)?

```
método intercambia:
   parámetros: m : array [n][n] de double; i, j : int
   variables : temp : double;  n :int = dimension de la matriz
   comienzo instrucciones:
      lazo desde k=0 hasta n-1
         temp=m[i][k]
         m[i][k]=m[j][k]
         m[j][k]=temp
      fin del lazo
   fin instrucciones
fin método intercambia
```

El tiempo de ejecución de las instrucciones dentro del lazo es O(1). El lazo se hace *n* veces. Por ello, el tiempo total es O(*n*)

4) Se dispone de una clase denominada `Poligono` con un método para calcular su área, y otro para borrar los datos del polígono, y que presentan la siguiente interfaz:

```
public double area() throws AristasSolapadas, FueraRango {...}
public void borrar() {...}
```

donde `AristasSolapadas` y `FueraRango` son dos excepciones declaradas dentro de la clase `Poligono`.

Se pide escribir un fragmento de programa que haga lo siguiente con el polígono `p` ya creado:

- Mostrar el área del polígono en la pantalla
- Si se lanza `AristasSolapadas`, poner en pantalla el mensaje "Las aristas se solapan"
- Si se lanza `FueraRango`, llamar al método `borrar`

```
try {
  System.out.println("Area="+p.area());
} catch (Poligono.AristasSolapadas e) {
  System.out.println("Las aristas se solapan");
} catch (Poligono.FueraRango e) {
  p.borrar();
}
```

5) Indicar cuáles de las siguientes expresiones son incorrectas en Java, de acuerdo con las reglas de construcción de expresiones y de compatibilidad de tipos. Razonar las respuestas. No considerar posibles errores debidos a variables sin inicializar.

Línea	Respuesta
`int i,j,k;` `double x,y,z;`	
`int[] lista= new int[10];`	
`double vec = new double[3];`	Incompatibilidad de tipos (debería ser double[] vec = ...)
`i=0*6;` `x=0.0;`	
`j=1.0;`	No se puede guardar double en int
`k*2=j;`	Una asignación siempre debe tener una variable sola a la izquierda
`z=i+2;` `x=j/k;`	
`lista[i]=x;`	No se puede guardar double en int
`vec[2]=1.0e6+lista[0];` `k++;`	
`lista[2]+j=lista[6];`	Una asignación siempre debe tener una variable sola a la izquierda
`vec[x]=i;`	El índice del array debe ser entero

Soluciones del Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Junio 2003

Problema (5 puntos)

Se dispone de una clase llamada `Tanque` ya implementada que representa un tanque que contiene líquido. Cada tanque tiene una válvula de apertura variable para verter ese líquido, y un sensor del nivel del líquido. La clase responde a la siguiente especificación, en la que la función de cada método se describe mediante un comentario de documentación:

```
public class Tanque {

  /** Constructor que crea el tanque y le pone su nombre y el nivel máximo */
  public Tanque(String nombre, double maximo) {...}

  /** Retorna el nombre del tanque */
  public String nombre() {...}

  /** Retorna el nivel máximo del tanque */
  public double nivelMaximo() {...}

  /** Retorna el nivel actual de líquido del tanque */
  public double nivel() {}

  /** Pone la válvula al valor de apertura indicado por porcentaje
      Si detecta que la válcula está atascada lanza Atascada
      Si el porcentaje está fuera del rango [0,100], lanza FueraRango */
  public void ponValvula (double porcentaje) throws Atascada, FueraRango {...}

  /** Retorna el valor actual de apertura de la válvula del tanque */
  public double aperturaValvula() throws Atascada {}

}
```

Las excepciones `Atascada` y `FueraRango` están declaradas en clases independientes.

Se pide hacer una nueva clase, llamada `SistemaMezcla` que permite gestionar varios tanques, para conseguir un sistema en el que se puedan mezclar líquidos en diferentes proporciones, en una planta industrial. La clase deberá responder a la siguiente interfaz:

```
public class SistemaMezcla {

  public SistemaMezcla (int maxTanques) {...}

  public void anadeTanque (Tanque t) throws Demasiados {}

  public void modificaApertura (String nombreTanque, double apertura)
    throws Atascada, NoEncontrado
  {...}

  public void informe() {...}

}
```

La clase deberá contener dos campos privados:

- `tanque`: Será un array de objetos de la clase `Tanque`
- `num`: Será un entero para almacenar el número de tanques contenidos actualmente en el sistema; inicialmente valdrá cero

Asimismo, deberá contener los métodos definidos en la interfaz, que deberán hacer lo siguiente:

- *constructor*: Debe crear el array `tanque` con un número de elementos igual a `maxTanques`; observar que de las casillas de este array sólo se utilizarán las `num` primeras casillas; además, debe poner `num` a valor cero.
- `anadeTanque`: Si `num` es igual al máximo número de tanques lanzar la excepción `Demasiados`. Si no, incrementar `num` en una unidad y hacer que la casilla `num-1` del array `tanque` sea igual al tanque `t`.
- `modificaApertura`: Debe realizar los siguientes pasos:
  - En primer lugar debe buscar entre los tanques almacenados en el array `tanque` aquel cuyo nombre coincide con `nombreTanque`. Si no lo encuentra, lanza la excepción `NoEncontrado`. Si lo encuentra, sigue al siguiente paso.
  - Limita el valor de `apertura` para que esté en el rango [0,100]. Para ello, si vale menos que cero lo pone igual a cero; si vale más que 100, lo pone igual a 100; en otros casos, lo deja como está.
  - Pone el valor de apertura (con el método `ponValvula`) del tanque encontrado en el paso uno igual al valor de apertura obtenido en el paso dos. Para llamar a este método el compilador nos obliga a tratar la excepción `FueraRango`, aunque sabemos que no va a ocurrir porque hemos limitado la apertura a valores correctos. En el tratamiento de la excepción poner en pantalla el mensaje "Error inesperado".
- `informe`: Debe mostrar un informe sobre el estado de cada tanque almacenado en el sistema. Para cada tanque debe mostrar un mensaje con su nombre, nivel actual, y valor de apertura. Si se lanza `Atascada`, entonces poner un mensaje indicando el nombre del tanque e indicando que está atascado, y seguir con los demás tanques de la misma forma.

Las excepciones `Demasiados` y `NoEncontrado` están también declaradas en clases independientes, con constructores sin parámetros.

```
public class SistemaMezcla {

  private Tanque [] tanque;
  private int num;

  public SistemaMezcla (int maxTanques) {
    tanque = new Tanque [maxTanques];
    num=0;
  }

  public void anadeTanque (Tanque t) throws Demasiados {
    if (num==tanque.length) {
      throw new Demasiados();
    }
    tanque[num]=t;
    num++;
  }

  // busca el tanque cuyo nombre coincide con nombreTanque y retorna
  // la casilla en la que esta; si no lo encuentra, lanza NoEncontrado
  private int busca (String nombreTanque) throws NoEncontrado {
    for (int i=0; i<num; i++) {
      if (nombreTanque.equals(tanque[i].nombre())) {
        return i;
      }
    }
    // Si se llega aqui es porque el tanque no se ha encontrado
    throw new NoEncontrado();
  }
```

```
  public void modificaApertura (String nombreTanque, double apertura)
    throws Atascada, NoEncontrado
  {
    int i=busca(nombreTanque);

    // limitar el valor de la apertura para que estÈ en el rango [0,100]
    if (apertura<0.0) {
      apertura=0.0;
    }
    if (apertura>100.0) {
      apertura=100.0;
    }
    // poner la apertura
    try {
      tanque[i].ponValvula(apertura);
    } catch (FueraRango e) {
      // no se debe entrar aqui ya hemos limitado el rango
      System.out.println("Error inesperado "+e);
    }
  }

  public void informe() {
    for (int i=0; i<num; i++) {
      try {
        double apertura = tanque[i].aperturaValvula();
        System.out.println("Valvula del tanque "+tanque[i].nombre()+
                           " est· al "+apertura+"%");
      } catch (Atascada e){
        System.out.println("Valvula del tanque "+tanque[i].nombre()+
                           " est· atascada");
      }
    }
  }

}
```