Examen de Introducción al Software
(Ingeniería Informática)

Septiembre 2011

Primera parte (5 puntos, 50% nota del examen)

1) Escribir un método en Java que permita copiar todos los datos de dos arrays de números reales que se pasan como parámetros en un tercer array que será devuelto por el método. En el resultado estarán en primer lugar los datos del primer array, y a continuación los del segundo.

2) Utilizando la instrucción `switch` escribir un método Java al que se le pasa un carácter y un número real, y que devuelve el número real multiplicado por el factor de escala obtenido a partir del carácter, según los valores de la tabla:

Carácter	Factor de Escala
'd'	0.1
'c'	0.01
'm'	0.001
'D'	10.0
'H'	100.0
'K'	1000.0

Si se pasa otro carácter que no esté en la tabla, retornar `Double.NaN` para indicar el error.

La cabecera del método será:

public static double convierte (char car, double valor)

3) Escribir el *pseudocódigo* de un algoritmo iterativo que calcula la suma del siguiente desarrollo en serie de la función *arctanh(x)*:

$$\sum_{i=0}^{n} \frac{x^{2i+1}}{2i+1}$$

Para hacer más eficiente este algoritmo, se debe crear una variable para el numerador del desarrollo (x^{2i+1}). Observar que de una iteración a la siguiente, el numerador debe multiplicarse por x^2, ya que $x^{2(i+1)+1}=x^{2i+1}*x^2$. Los parámetros del algoritmo serán n y x.

4) Escribir un *método recursivo* en Java que permita obtener el cociente de dos números enteros a y b por restas sucesivas. El caso directo corresponde al caso $b>a$ y hay que retornar cero. En el caso recursivo, retornar el cociente de a-b entre b, más uno.

5) Indicar *razonadamente* si las siguientes afirmaciones son correctas o no. Utilizar un máximo de 3 líneas para cada respuesta:

   a. Podemos construir una base de datos con un archivo de papel bien organizado.

   b. La principal ventaja de una base de datos es que podemos guardar información relacionada.

   c. En una hoja de cálculo si hay una dependencia circular entre las fórmulas de diversas casillas se producirá un error.

   d. El lenguaje SQL sirve para agregar datos a una base de datos.

   e. Cuando se accede a una página web donde se nos ofrece un catálogo de productos para comprarlos, el servidor de la página Web usa una base de datos para guardar el inventario de productos a ofrecer.

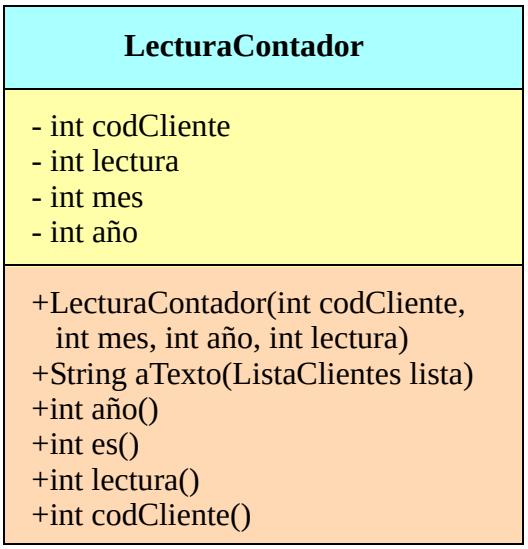
   Nota: en esta cuestión, las respuestas correctas suman 0.2 puntos y las incorrectas restan 0.1 puntos. Se valora la precisión de la respuesta.

Examen de Introducción al Software
(Ingeniería Informática)

Septiembre 2011

Segunda parte (5 puntos, 50% nota del examen)

Se dispone de una clase ya realizada llamada `LecturaContador` que permite almacenar los datos de la lectura de un contador eléctrico. El diagrama de clases se muestra en la figura[1]. El significado de los atributos es:



- `codCliente`: codigo numérico que identifica el cliente
- `lectura`: lectura del contador, en Kwh
- `mes`: mes en que se realizó la lectura (de 1 a 12)
- `año`: año en que se realizó la lectura (ej: 2011)

Y los métodos hacen lo siguiente:

- constructor: copia los parámetros en los atributos
- `aTexto`: se le pasa un objeto de la clase `ListaClientes` que es la lista de clientes donde encontrar el nombre del cliente a partir de su codigo; el método retorna un texto que contiene el nombre del cliente y todos los datos de la lectura del contador
- `codCliente`, `lectura`, `mes`, `año`: métodos observadores que retornan el atributo del mismo nombre.

Se dispone también de la clase `ListaClientes`, que dispone del método con la siguiente cabecera, que retorna el nombre del cliente cuyo identificador es `idCliente`; si ese cliente no existe, retorna un `String` de cero caracteres.

```
public String nombreCliente(int idCliente)
```

Se pide implementar en Java la clase `Consumos` que sirve para guardar una lista histórica de consumos eléctricos de diversos clientes y que dispone de métodos para obtener información a partir de esta lista.

La clase tiene una lista variable de objetos de la clase `LecturaContador`, almacenados en un atributo de tipo `ArrayList<LecturaContador>` llamado `lista`. También dispone de un atributo llamado `listaClientes`, de la clase `ListaClientes`, con el método indicado arriba que permite obtener el nombre de un cliente (*nota*: no añadir más atributos).

Además dispone de los siguientes métodos, que funcionan según sus comentarios de documentación:

1. "+" indica que el método es público y "-" indica que el atributo es privado

```java
public class Consumos {

    private ArrayList<LecturaContador> lista; // lista de consumos
    private ListaClientes listaClientes; // lista de clientes

    /**
     * Constructor, que crea la lista de consumos (llamada lista) vacía
     * y anota la lista de clientes lis que se pasa como parámetro
     * en el atributo listaClientes
     * @param lis es la lista de clientes a anotar
     */
    public Consumos (ListaClientes lis) {
       ...
    }

    /**
     * Añade la lectura de contador lec a la lista, comprobando que la
     * lectura en Kwh es mayor o igual que cero, el año está
     * comprendido entre 2000 y 2200, el mes está comprendido entre 1 y
     * 12, y que existe en listaClientes un nombre de cliente asociado
     * al código de cliente. Si estas comprobaciones son correctas,
     * añade la lectura lec a la lista y retorna true. Si alguna
     * comprobación falla, pone un mensaje en pantalla, no añade la
     * nueva lectura, y retorna false
     * @param lec es la lectura del contador que hay que añadir a la lista
     * @return un booleano que indica si se ha añadido la lectura o no
     */
    public boolean inserta(LecturaContador lec) {
        ...
    }

    /**
     *  Numero de lecturas existentes para un cliente concreto
     *  @param codCliente es el código del cliente a buscar
     *  @return el número de lecturas del cliente indicado
     */
    public int numLecturas(int codCliente) {
        ...
    }

    /**
     *  Busca y retorna la primera lectura del cliente codCliente
     *  en el mes y año indicados. Si no la encuentra, retorna null
     *  @param codCliente es el código de cliente a buscar
     *  @param mes es el mes a buscar
     *  @param año es el año a buscar
     *  @return la primera lectura del cliente que cumple las condiciones,
     *  o null si no se encuentra
     */
    public LecturaContador primeraLecturaEn
           (int codCliente, int mes, int año)
    {  ... }
```

```
    /**
     * Comprueba si la lista de lecturas es correcta. Las condiciones
     * que se comprueban son:
     * - que las lecturas estén ordenadas por fechas no decrecientes
     * - que las lecturas de cada cliente vayan siempre en orden no
     *   decreciente
     * @ return booleano que indica si la lista de lecturas es correcta o no
     */
    public boolean esCorrecta() {
       ...
    }
}
```

Para el método `numLecturas`() observar que hay que crear un contador con valor incial cero. Mediante un recorrido completo de la lista, se incrementa el contador cada vez que encontramos un objeto cuyo código de cliente coincide con `codCliente`.

Para el método `primeraLecturaEn`(), observar que se puede usar un algoritmo de búsqueda en el que la condición de búsqueda es que el código de cliente, el mes y el año coincidan con los indicados en los parámetros.

Para el método `esCorrecta` usar el siguiente pseudocódigo:

```
método esCorrecta() retorna booleano
   // recorremos todas las casillas excepto la última,
   // para comprobar si son correctas
   para cada i desde 0 hasta el tamaño de lista -2 hacer
       // tenemos que comprobar la casilla i
       LecturaContador lec= elemento i de lista
       // primero comprobamos que la fecha es creciente
       LecturaContador siguiente = elemento i+1 de lista;
       si lec.año()>siguiente.año() entonces
           // el año está desordenado
           retorna false
       fin de si
       si los años de lec y siguiente coinciden y además
          mes de lec > mes de siguiente
       entonces
           // el año es igual pero el mes está desordenado
           return false;
       fin de si
       // ahora comprobamos las lecturas crecientes
       // para ello buscamos la primera casilla posterior a i
       // del mismo cliente
       int encontrada=-1  // guardaremos aqui la casilla encontrada
       int j=i+1;
       mientras j<tamaño de lista y además encontrada==-1 hacer
           si el codCliente de la casilla j de lista coincide con
              el codCliente de lec
           entonces
               encontrada=j;
           fin de si
           j++
```

```
            fin de mientras
            si encontrada>=0 entonces
                // comprobamos si la lectura esta en orden no decreciente
                si  lectura en Kwh de la casilla encontrada de lista <
                    lectura en Kwh de lec
                entonces
                    return false;
                fin de si
            fin de si
        fin de para
        // si llegamos aqui todo está correcto
        retorna true
    fin de método
```

Valoración:

1) Constructor e `inserta`: 1.25 puntos

2) Resto de los métodos: 1.25 puntos cada uno.