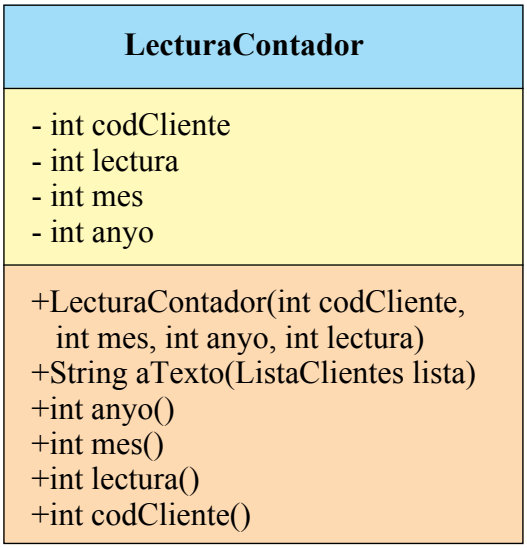
Examen de Prácticas de Introducción al Software
(Ingeniería Informática)

Septiembre 2011

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)
- `anyo`: 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`, `anyo`: métodos observadores que retornan el atributo del mismo nombre.

Se dispone también de la clase `ListaClientes`, que dispone de los métodos con las siguientes cabeceras. El primero de ellos retorna el nombre del cliente cuyo identificador es `idCliente`; si ese cliente no existe, retorna un `String` de cero caracteres. El segundo retorna el identificador del cliente cuyo nombre es `nombreCliente` y, si ese nombre no existe, retorna un numero negativo. El tercero añade un nuevo cliente, con su nombre y código de cliente, a la lista.

```
public String nombreCliente(int idCliente)
public int idCliente(String nombreCliente) {
public void anadeCliente(int idCliente, String nombre)
```

Se pide implementar en Java parte de la clase `Consumos`. Esta clase sirve para guardar una lista histórica de consumos eléctricos de diversos clientes y 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` (*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
import java.util.*;
/**
 * Clase con una lista historica de consumos electricos de diversos clientes
 */
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) vacia
     * y anota la lista de clientes lis que se pasa como parametro
     * en el atributo listaClientes
     *
     * @param lis es la lista de clientes a anotar
     */
    public Consumos (ListaClientes lis) {...}

    /**
     * Anade una lectura de contador a la lista leyendo sus datos del teclado
     * e insertandola luego mediante una llamada a inserta(). Se retorna el
     * booleano retornado por inserta
     * @return un booleano que indica si se ha anadido la lectura o no
     */
    public boolean insertaDeTeclado() {...}

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

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

    /**
     *  Busca y retorna la primera lectura del cliente codCliente
     *  en el mes y anyo indicados. Si no la encuentra, retorna null
     *  @param codCliente es el codigo de cliente a buscar
```

```
     *  @param mes es el mes a buscar
     *  @param anyo es el anyo 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 anyo)
    {...}

    /**
     *  Muestra en pantalla un listado de las lecturas del cliente indicado
     *  Si el cliente no existe, pone un mensaje de error en pantalla
     *  @param nombreCliente nombre del cliente
     */
    public void listado (String nombreCliente) {...}

     /**
     *  Retorna el consumo realizado por el cliente indicado entre el
     *  mes1-anyo1 y el mes2-anyo2. El consumo es la diferencia entre ambas
     *  lecturas
     *  Si el nombre del cliente no se encuentra en la lista, retorna -1
     *  Si no existe una lectura para el mes1-anyo1, retorna -2
     *  Lo mismo si no existe consumo para el mes2-anyo2
     *  @param nombreCliente es el nombre de cliente a usar
     *  @param mes1 es el mes de la primera fecha
     *  @param anyo1 es el anyo de la primera fecha
     *  @param mes2 es el mes de la segunda fecha
     *  @param anyo2 es el anyo de la segunda fecha
     *  @return la diferencia entre la lectura de la segunda fecha, menos la
     *  lectura de la primera fecha, o un numero negativo para indicar
     *  un error
     */
    public int consumo
    (String nombreCliente, int mes1, int anyo1, int mes2, int anyo2)
    {...}

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


}
```

Parte A

Implementar los métodos `insertaDeTeclado`(), `listado`(), y `consumo`(). El resto de los métodos de la clase `Consumos` se dan ya resueltos.

Parte B

Se pide implementar un programa de prueba que:

- Cree un objeto de la clase `ListaClientes` añadiéndole 4 clientes para poder hacer pruebas
- Cree un objeto de la clase `Consumos` y añade 10 consumos, para dos de los clientes de la lista de clientes
- Pruebe los tres métodos desarrollados en la Parte A, incluyendo casos normales y de error.

Valoración:

1) Parte A: 6 puntos

2) Parte B: 4 puntos