Examen de Prácticas
Programación I (Ingeniería Informática)
Informática (Licenciado en Matemáticas)

Septiembre 2006

Puntuación: 4 puntos

Disponemos de una clase ya realizada que permite almacenar los datos asociados a un mensaje de un sistema de alarma. Estos datos son la hora y minuto en el que se produjo la alarma (enteros), el tipo de alarma (entero), y un texto explicativo. La interfaz de esta clase es:

```
public class Mensaje {
   /**
    * Construye el mensaje a partir del tipo y del texto
    */
   public Mensaje(int tipo, String texto) {}

   /** Retorna la hora a la que se genero el mensaje */
   public int hora() {...}

   /** Retorna el minuto a la que se genero el mensaje */
   public int minuto() {...}

   /** Retorna el tipo de mensaje */
   public int tipo() {...}

   /** Retorna el texto del mensaje */
   public String texto() {...}
}
```

Por otro lado, se dispone de otra clase que también está realizada y que permite almacenar una secuencia de mensajes. Responde a la interfaz de la secuencia vista en clase, tomando en este caso particular la forma siguiente:

```
public class ListadoMensajes {
    /**
     * Constructor que crea la secuencia vacia
     */
    public ListadoMensajes() {...}

    /**
     * Anadir el mensaje m al final de la secuencia
     */
    public void anade(Mensaje m) {...}

    /**
     * Reiniciar la secuencia para poder recorrerla
     */
    public void reinicia() {...}

    /**
     * obtener el mensaje actual en la secuencia
     */
    public Mensaje mensajeActual() {...}

    /**
     * Avanza al siguiente elemento en la secuencia
     */
    public void avanza() {...}

    /**
     * Indica si estamos en el final de la secuencia
     */
    public boolean finalSecuencia() {...}
}
```

En tercer lugar se dispone en parte de la clase `Sensor`, que representa un sensor de alarma y obedece a la siguiente interfaz:

```
/**
 * Clase que representa un sensor de alarma que mide una magnitud real
 * y comprueba si está en un rango [min,max] entre un valor mínimo
 * y uno máximo
 */
public class Sensor
{
    // atributos privados
    private double min,max;
    private String nombre;
    private double ultimoValor;

    /**
     * Constructor al que se le pasa el mínimo y máximo valor de la
     * magnitud que mide el sensor, y el nombre. Copia estos datos
     * en los respectivos atributos
     */
    public Sensor(double min, double max, String nombre) {...}

    /**
     * Indica si hay alarma o no, comprobando si el último valor
     * de la medida, atributo ultimoValor, está entre min y max
     *
     */
    public boolean hayAlarma() {...}

    /**
     * Lee el valor actual de la magnitud del sensor y lo almacena
     * en el atributo ultimoValor
     * obtiene el valor simulando la lectura con un número
     * aleatorio que está en el rango [min-x,max+x], siendo x el 20%
     * de max-min
     */
    public double lee() {...}

    /**
     * Retorna un texto con los principales datos del sensor:
     * nombre: ultimoValor - "OK" (si esta en rango)
     * nombre: ultimoValor - "ALARMA" (si no esta en rango)
     */
    public String estado() {...}
}
```

Por último, se dispone en parte de la clase `SistemaAlarma`, que representa el sistema de alarma completo, y que obedece a la siguiente interfaz:

```
public class SistemaAlarma
{
    // atributos privados
    private ListadoMensajes lista;
    /**
     * Constructor que crea la lista de mensajes vacía
     */
    public SistemaAlarma(String nombre)  {
       lista=new ListadoMensajes();
    }

    /**
     * Procesa los sensores de alarma que se pasan en el array s,
     * recorriéndolos. Para cada uno llama a lee()
     * y luego a hayAlarma(). Si hay alarma inserta en la lista un mensaje
     * de tipo 1, cuyo texto es el devuelto por estado() para el sensor
     * con alarma. Retorna el numero de alarmas detectadas
     */
    public int procesaSensores(Sensor[] s)    {...}
```

```
    /**
     * Retorna el numero de mensajes del tipo tipoMensaje
     * que existen en la lista
     */
    public int numMensajesTipo(int tipoMensaje) {...}

    /**
     * Busca y retorna el texto del primer mensaje anterior
     * a la hora y minutos indicados
     */
    public String buscaMensajeAnteriorA(int hora, int  minuto) {...}

    /**
     * Crea una nueva secuencia de mensajes del tipo ListadoMensajes,
     * anade a ella los mensajes anteriores a la hora y minutos
     * indicados y cuyo tipo coincide con tipoMensaje, y retorna la
     * nueva secuencia
     */
    public ListadoMensajes  mensajesAnteriores DeTipo
        (int hora, int minuto, int tipoMensaje)
    {...}

   /**
    * Muestra en pantalla cada uno de los mensajes que contiene
    * la lista, uno por línea, con la hora (hh), los minutos (mm),
    * el tipo (tt) y el texto del mensaje (xxxx), con el siguiente formato:
    * hh:mm - tt = xxxx
    */
   public void muestra() {...}

}
```

Puede observarse que existe un atributo privado que es una secuencia de mensajes de la clase `ListadoMensajes`. Los comentarios de documentación explican lo que debe hacer cada método. De estos métodos, ya están realizados `numMensajesTipo`, `buscaMensajeAnteriorA`, y `mensajesAnteriores DeTipo`.

El profesor te proporcionará el software que ya está realizado.

Se pide implementar y comprobar el correcto funcionamiento de:

1) Modificar la clase `SistemaAlarma` para escribir en Java los métodos `procesaSensores` (1.5 puntos) y `muestra` (1 punto).

2) Modificar la clase `Sensor` para escribir los métodos `lee`(), `hayAlarma`() y `estado`() (1.5 puntos).