Examen de Fundamentos de Computadores y Lenguajes
(Licenciado en Física)

Junio 2008

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

1) Escribir un método con la cabecera que se indica a continuación, que permita obtener el peso en Kg de una muestra de un material cuyo volumen en m^3 se indica en el parámetro `volumen`.

```
double peso(Material mat, double volumen)
```

El material se indica como un valor de la clase enumerada `Material`. Las densidades de los materiales considerados se muestran en la tabla.

hierro	7874 Kg/m^3
plomo	11340 Kg/m^3
cobre	8960 Kg/m^3

```
public enum Material {
    hierro, plomo, cobre
}
```

2) Escribir un método que corresponda a la cabecera que se indica abajo y que retorne un `String` que sea igual a la parte del `String original` que es posterior al primer carácter `'#'`. Por ejemplo, para el String `"1234#Esto es un texto"`, se debe retornar `"Esto es un texto"`. Si el carácter `'#'` no se encuentra, el método debe retornar un `String` de cero caracteres.

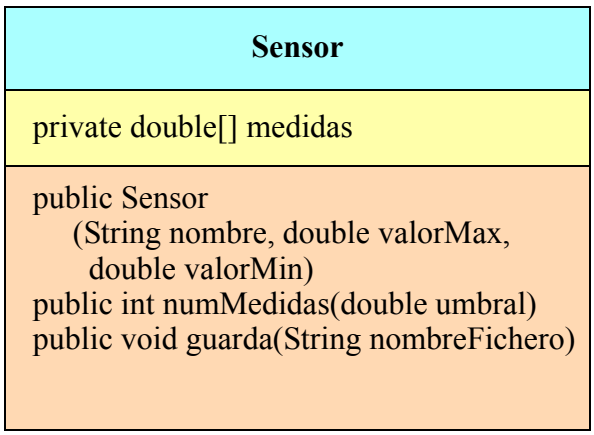
```
String parteFinal(String original)
```

3) Reescribir el siguiente fragmento de programa de modo que si se lanza la excepción `NoDisponible` en alguna de las instrucciones del interior del bucle se trata como un error leve: se pone un mensaje en pantalla y se sigue dentro del bucle; además, si se lanza la excepción `DatosIncorrectos` en esas mismas instrucciones se trata como un error grave: se pone en pantalla un mensaje y se abandona el fragmento de programa.

```
do {
  hayMasDatos=experimento.leeDatos(datos);
  if (hayMasDatos) {
      experimento.procesaDatos(datos);
  }
} while (hayMasDatos);
System.out.println("Datos procesados");
```

4) Escribir para la clase `Sensor` cuyo diagrama se muestra en la figura el método `numMedidas()` que retorna el número de valores del array `medidas` que exceden del valor indicado por umbral.

5) Escribir para la clase `Sensor` el método `guarda()` que escribe en el fichero de texto cuyo nombre se indica en `nombreFichero` todos los valores del array `medidas` que sean positivos, uno por línea.



Examen de Fundamentos de Computadores y Lenguajes
(Licenciado en Física)

Junio 2008

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

Se desea hacer parte del software del sistema de telemetría de un equipo de fórmula 1. Se dispone para ello de la clase `DatosTelemetria` que permite almacenar datos de un coche en un momento dado y tiene la siguiente parte pública:

```
public class DatosTelemetria {

        /**
         * Retorna las revoluciones por minuto
         */
        public double getRpm() {...}

        /**
         * Retorna la velocidad en metros/seg
         */
        public double getVelocidad() {...}

        /**
         * Retorna la temperatura del aceite en grados centígrados
         */
        public double getTempAceite() {}

}
```

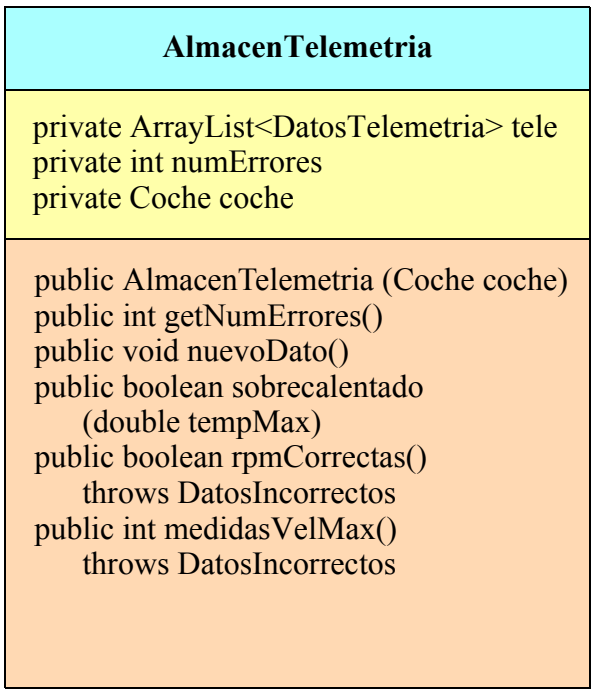
También se dispone de la clase `Coche` que dispone entre otros de un método para leer vía radio los datos actuales de la telemetría del coche:

```
public class Coche {

        /**
         * Crea un coche dado el nombre del conductor y
         * el número de chasis
         */
        public Coche(String conductor, String numChasis) {...}

        /**
         * Obtiene y retorna los datos de telemetría actuales
         */
        public DatosTelemetria leeTelemetria() throws NoDisponible {...}
}
```

Lo que se pide es escribir la clase `AlmacenTelemetria` que responde al diagrama de clases de la figura. Los métodos de esta clase deben hacer lo siguiente:



- *Constructor*: crea el almacén de datos de telemetría (`tele`), pone a cero el número de errores (`numErrores`) y copia en el atributo `coche` el parámetro del mismo nombre.
- `getNumErrores()`: Retorna el numero de errores de lectura de telemetría.
- `nuevoDato()`: Lee un nuevo dato de telemetría del coche (con `leeTelemetria()`) y lo añade al almacén de datos de telemetría, al final. Si se lanza `NoDisponible` incrementa el número de errores.
- `sobrecalentado()`: Indica si el motor está sobrecalentado. Se considera como tal si las últimas 10 medidas de telemetría tienen una temperatura del aceite por encima de `tempMax`.
- `rpmCorrectas`: Indica si las revoluciones por minuto (rpm) son correctas porque en ninguno de los datos almacenados se han superado las rpm máximas de 18000. Lanza `DatosIncorrectos` si se detecta que las rpm han sido negativas en algún momento.
- `medidasVelMax`: Retorna el numero de medidas en las que el coche ha estado entre el 90% y el 100% de la velocidad máxima. Lanza `DatosIncorrectos` si alguna velocidad es negativa.

Las excepciones `NoDisponible` y `DatosIncorrectos` están declaradas en clases aparte como:

```
public class NoDisponible extends Exception {}
public class DatosIncorrectos extends Exception {}
```

Nota: se valorará cada parte de la clase en función de su dificultad, del siguiente modo:

- `medidasVelMax`: 1.5 puntos
- `nuevoDato`, `sobrecalentado`, `rpmCorrectas`: 1 punto cada uno
- constructor y `getNumErrores`: 0.5 puntos entre los dos