Examen de Programación
(Grados en Física y Matemáticas)

Junio 2012

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

1) Escribir en Java el método `posicion()`, que retorna el valor correspondiente a la expresión de la posición x que aparece abajo y que será acorde al diagrama de clases.

Esta posición corresponde al movimiento de oscilación no lineal de una partícula de masa m cargada con una carga negativa $-q$, en las proximidades de una placa indefinida cargada con una densidad de carga positiva σ. La posición es función del tiempo t que se pasa como parámetro.

En el diagrama de clases aparecen los atributos y métodos de la clase. Los atributos `q`, `sigma`, `m` y `posIni` se corresponden con los valores q, σ, m y A de las fórmulas. Los valores retornados por los métodos `fuerza()`, `periodo()` y `velocidadOrigen()` se corresponden con los valores F, T y v_0 de las fórmulas.

ParticulaCargada
-double q -double sigma -double m -double posIni
+ParticulaCargada (double q, double sigma, double m, double posIni) +double periodo() +double velocidadOrigen() +double posicion(double t) +double fuerza()

Expresión del valor de la posición: se puede suponer que siempre se cumple $0 \le t \le T$

$$x = \begin{cases} A - \frac{1}{2}\frac{F}{m}t^2, & 0 \le t \le \frac{T}{4} \\ -v_0\left(t - \frac{T}{4}\right) + \frac{1}{2}\frac{F}{m}\left(t - \frac{T}{4}\right)^2, & \frac{T}{4} < t \le \frac{3T}{4} \\ v_0\left(t - \frac{3T}{4}\right) - \frac{1}{2}\frac{F}{m}\left(t - \frac{3T}{4}\right)^2, & \frac{3T}{4} < t \le T \end{cases}$$

2) Escribir el *pseudocódigo* de un método que calcula y retorna el siguiente desarrollo en serie de la función $(1+x)e^x$, dados los valores de x y n como parámetros. Por eficiencia, usar variables separadas para el numerador y el denominador, e ir calculando el valor de estas variables a cada paso del bucle, a partir de sus valores anteriores.

$$\sum_{i=0}^{n} (i+1)\frac{x^i}{i!}$$

Observar que si conocemos el denominador del término anterior $den=(i-1)!$, el siguiente denominador, $i!$, se calcula como $den*i$. Asimismo, si conocemos el numerador del término anterior $num=x^{i-1}$, el siguiente numerador, x^i, se calcula como $num*x$.

3) Se dispone de un método declarado según la cabecera de abajo, perteneciente a la clase `Experimento`, que puede lanzar dos excepciones.

```
public static double medida() throws MedidaInestable, MedidaErronea
```

Escribir otro método estático situado en otra clase que llame al anterior y devuelva el valor retornado por él, pero que además realice los siguientes tratamientos de excepción:

- Si se lanza `MedidaInestable`, reintentar la operación hasta que se pueda retornar un valor
- si se lanza `MedidaErronea` retornar `Double.NaN`

4) Escribir un método con la cabecera indicada abajo que retorne un array de números reales conteniendo todos los números reales contenidos en el `ArrayList` que se pasa como parámetro. Es decir, el método convierte un `ArrayList` a array.

```
public static double[] convierte(ArrayList<Double> lista)
```

5) Dentro del directorio del usuario se dispone de un directorio llamado `copias_seguridad` y de otro directorio llamado `practica_3` que es un proyecto de *bluej* y contiene ficheros java (con extensión "`.java`"), clases compiladas (con extensión "`.class`"), un directorio llamado `doc` con los documentos del proyecto, y ficheros de texto con la extensión "`.txt`".

Suponiendo que el directorio de trabajo inicial es el del usuario, escribir las órdenes linux/unix necesarias para:

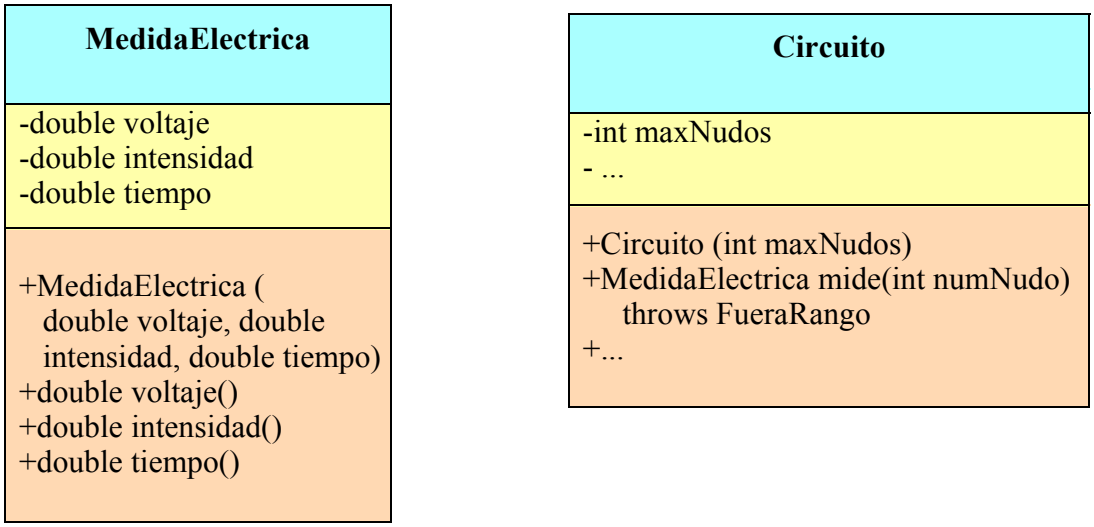
- crear dentro de `copias_seguridad` un nuevo directorio vacío llamado `copia_practica_3`
- copiar todos los ficheros java de `practica_3` a `copia_practica_3`
- mover el directorio `doc` de `practica_3` a `copia_practica_3`
- borrar los ficheros de texto de `practica_3`

Examen de Programación
(Grados en Física y Matemáticas)

Junio 2012

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

Se desea realizar parte del software de un sistema de medida de señales eléctricas en un circuito. Para ello se dispone de las clases `MedidaElectrica` y `Circuito`, ya realizadas, cuyo diagrama de clases aparece abajo:



La clase `MedidaElectrica` contiene los datos de una medida de parámetros eléctricos. Su constructor recibe los datos de la medida. Se dispone de tres métodos observadores, uno para cada dato de la medida: voltaje, intensidad y tiempo.

La clase `Circuito` representa un circuito eléctrico con un sistema de medida. Se puede medir la tensión e intensidad de cada nudo del circuito. Los nudos se identifican por un número entero entre 1 y el máximo número de nudos. Al constructor se le pasa el número máximo de nudos. El método `mide()` realiza una medida de tensión e intensidad en el nudo indicado, y retorna ambos valores junto con el instante actual en un objeto de la clase `MedidaElectrica`.

Se pide realizar la clase `ListaMedidas`, que debe obedecer al siguiente código Java parcial:

```
import java.util.*;

/**
 * Esta clase almacena una lista de medidas eléctricas de un circuito y
 * dispone de métodos para interpretar estas medidas
 */
public class ListaMedidas
{
    // atributo que contiene la lista de medidas
    private ArrayList<MedidaElectrica> lista;
```

```java
/**
 * Constructor que crea la lista de medidas y, leyendo con el método
 * mide(), le añade 10000 medidas del circuito y nudo indicados en
 * los parámetros.
 * Las medidas que lancen FueraRango se ignoran y no
 * se almacenan en la lista, pero se sigue leyendo hasta completar
 * las 10000 llamadas al método mide(). Por tanto, el tamaño final de la
 * lista es igual a 10000 si no hay errores, o menor si hay errores.
 */
public ListaMedidas(Circuito c, int numNudo) {
   ...
}

/**
 * Escribe la lista de medidas en el fichero de texto cuyo nombre se
 * indica. Se escribe cada medida (compuesta por tiempo, voltaje e
 * intensidad) en una línea, usando columnas de 15 caracteres de ancho
 * y 3 decimales para cada valor.
 * Se escribe al principio del fichero una cabecera con el texto
 * tiempo         voltaje        intensidad
 */
public void escribe(String nombreFichero) {
   ...
}

// Retorna un array de reales que contiene la derivada del array x, con
// respecto al array t. Los arrays x y t deben ser del mismo tamaño.
// Si no lo son, se lanza LongitudIncorrecta
// La solución a retornar es un array de una casilla menos que x y t.
// El contenido de este array, que es la derivada, se calcula así:
// Para cada casilla i de la derivada desde la primera a la última,
// la derivada es el incremento de x entre las casillas i e i+1
// dividido por el incremento de t entre las casillas i e i+1:
// (x[i+1]-x[i])/(t[i+1]-t[i])
private double[] derivada(double[] x, double[] t)
   throws LongitudIncorrecta
{
    ...
}


/**
 * Retorna el número de medidas para las que la intensidad o el voltaje
 * tiene cambios muy bruscos, contando los puntos en que la derivada sea
 * muy grande (mayor que el límite indicado en los parámetros, en valor
 * absoluto)
 */
public int numCambiosRapidos(double limiteDerivVolt,
                             double limiteDerivInten)
{
    ...
}
```

```
    /**
     * Valores en rango: retorna true si todos los valores de voltaje e
     * intensidad están dentro de los rangos indicados (incluidos los valores
     * extremos), y false en caso contrario
     */
    public boolean valoresEnRango(double voltajeMin, double voltajeMax,
    double intensidadMin, double intensidadMax)
    {
        ...
    }
}
```

Para el método `numCambiosRapidos()` utilizar el siguiente pseudocódigo:

```
// obtener el voltaje, intensidad y tiempo en arrays separados
volt=nuevo array de reales de tamaño=tamaño de lista
inten=nuevo array de reales de tamaño=tamaño de lista
t=nuevo array de reales de tamaño=tamaño de lista
para cada i desde 0 hasta tamaño de lista-1 hacer
  volt[i]=voltaje del elemento i de lista
  inten[i]=intensidad del elemento i de lista
  t[i]=tiempo del elemento i de lista
fin para
// inicializar el contador de puntos con cambios bruscos
entero contador=0
// calcular las derivadas del voltaje y la intensidad usando el método derivada
array-de-reales derivVolt=derivada(volt,t);
array-de-reales derivInten=derivada(inten,t);
// contar puntos con cambios bruscos de voltaje o intensidad
para cada i desde 0 hasta tamaño de derivVolt-1 hacer
  si |derivVolt[i]|>limiteDerivVolt  o  |derivInten[i]|>limiteDerivInten entonces
     contador++
  fin si
fin para
// retornar la respuesta
retorna contador
```

La notación `|valor|` significa valor absoluto. Además de lo indicado en el pseudocódigo, el método `numCambiosRapidos()` debe tratar la excepción `LongitudIncorrecta`, pues es obligatorio. El tratamiento consistirá en poner un mensaje de error y retornar `-1`.

Observar que el método `valoresEnRango` consiste en una búsqueda de alguna casilla que incumpla las limitaciones de voltaje o intensidad. Si se encuentra esta casilla retornar `false`. Si no hay ninguna, retornar `true`. Seguir por tanto el algoritmo de búsqueda visto en clase.

Notas: copiar en el texto del examen las cabeceras de los métodos a realizar, para poner el ejercicio en contexto, pero no es preciso copiar los comentarios de documentación (ni el comentario que explica el funcionamiento del método `derivada()`), para evitar repeticiones. Las cabeceras no pueden modificarse.

Valoración: Constructor y resto de los métodos: 1 punto cada uno