Examen de Programación
(Grados en Física y Matemáticas)

Septiembre 2012

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

1) Se desea escribir un método que calcule el precio de un producto con IVA, dado el precio sin IVA y el tipo de IVA a aplicar. Los tipos a aplicar están definidos por las siguientes constantes enteras definidas en la misma clase que el método:

```
public static final int SUPERREDUCIDO=1;
public static final int REDUCIDO=2;
public static final int GENERAL=3;
```

El método tendrá la siguiente cabecera. Los precios están en céntimos de euro.

```
public static int precioConIVA(int precioSinIVA, int tipoIVA)
```

El IVA a aplicar para cada tipo corresponde a la siguiente tabla. El método debe construirse con una instrucción `switch`. El resultado se redondeará al céntimo más próximo.

Tipo	IVA a aplicar
SUPERREDUCIDO	4%
REDUCIDO	10%
GENERAL	21%

2) Escribir el *pseudocódigo* de un método iterativo que permita conocer si un número `n` que se pasa como parámetro es primo. El algoritmo consistirá en ir dividiendo el número `n` entre los números naturales comprendidos entre el 2 y el `n`-1. Si alguna división es exacta significará que el número `n` no es primo. Para saber si una división es exacta se puede comprobar si el resto de la división es cero o no. La cabecera del método será:

```
public static boolean esPrimo(int n)
```

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

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

Escribir otro método que invoque al anterior desde otra clase diferente, retorne el valor retornado por él y trate las excepciones del siguiente modo:

- si se lanza `MedidaInestable` pone un mensaje de error y retorna 0.0
- si se lanza `MedidaErronea` pone un mensaje de error y lanza la misma excepción para que sea tratada en donde se invocó al nuevo método

4) Escribir un método que escriba en un fichero de texto todos los elementos de un array de números reales que se pasa como parámetro. Se escribirán uno por línea

representándolos con 3 decimales. El nombre del fichero se pasa también como parámetro. La cabecera del método será la siguiente:

```
public static void escribeEnFichero(String nombreFichero, double[] valores)
```

Si se lanza alguna excepción de entrada/salida tratarla con un mensaje en pantalla.

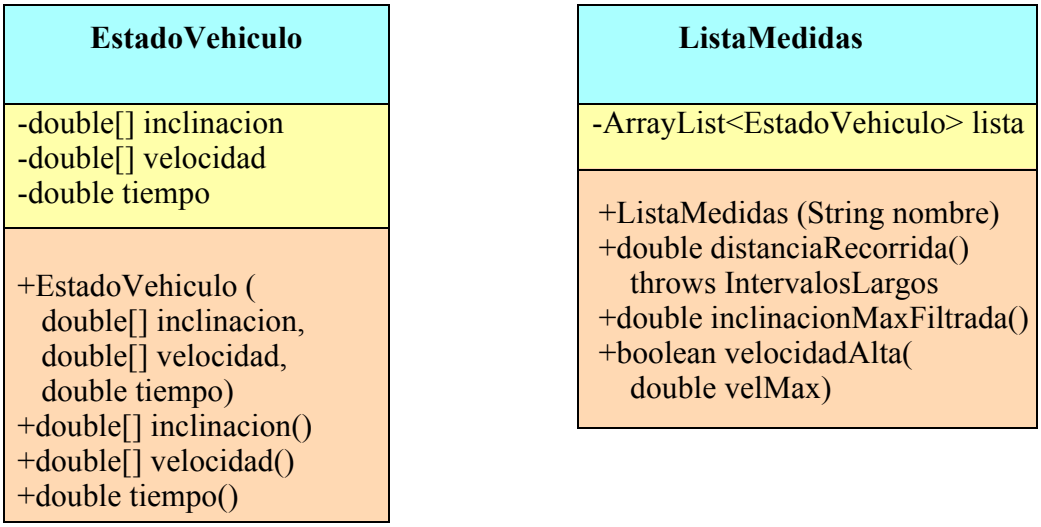
5) En un computador con sistema operativo Linux se dispone de un directorio llamado `proyectos` que está dentro del directorio del usuario. Dentro de `proyectos` hay un directorio llamado `practica12` que contiene un proyecto java. El directorio de trabajo inicial es el del usuario. Se desea escribir un *script* que haga lo siguiente:
   - crear en el directorio del usuario un nuevo directorio llamado `copias`
   - crear dentro de `copias` los siguientes directorios: `fuentes`, `clases`, `textos`
   - copiar en `fuentes` todos los ficheros acabados en `.java` de `practica12`
   - mover a `clases` todos los ficheros acabados en `.class` de `practica12`
   - mover a `textos` todos los ficheros acabados en `.txt` de `practica12`
   - borrar todos los ficheros acabados en `.ctxt` de `practica12`

Examen de Programación
(Grados en Física y Matemáticas)

Septiembre 2012

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

Se desea escribir parte del software de monitorización de un vehículo de exploración del terreno. Para ello se dispone de la clase `EstadoVehiculo`, ya realizada, cuyo diagrama de clases aparece abajo:



La clase `EstadoVehiculo` contiene los datos de una medida del estado del vehículo en un instante determinado. Sus atributos guardan:

- `inclinacion`: es un array de 3 casillas, que guardan la inclinación del vehículo con respecto al eje X, al eje Y y al eje Z, en grados
- `velocidad`: es un array de 2 casillas que guarda la velocidad del vehículo en la dirección X y en la dirección Y, en metros por segundo
- `tiempo`: es el instante al que corresponde la medida, en segundos desde el arranque del vehículo

El constructor de la clase `EstadoVehiculo` recibe los datos de la medida. Se dispone de tres métodos observadores, uno para cada dato de la medida: inclinación, velocidad y tiempo.

Se pide realizar la clase `ListaMedidas` que almacenará en un atributo llamado `lista` una lista de medidas del estado del robot en sucesivos instantes de tiempo y que dispondrá de operaciones para trabajar con esta lista. La clase debe obedecer al diagrama de clases que aparece más arriba. Los métodos deben hacer lo siguiente

- *Constructor*: lee del fichero de texto cuyo nombre se pasa como parámetro la lista de las medidas. Si se produce una excepción al leer el fichero deja la lista creada pero vacía. El fichero contendrá en cada línea los datos de una medida, en forma de números separados por espacios en blanco y con el siguiente orden:

```
inclinacion_X  inclinacion_Y  inclinacion_Z  velocidad_X  velocidad_Y tiempo
```

- `distanciaRecorrida`(): Calcula y retorna la distancia recorrida por el vehículo. Esta distancia es la suma de todas las distancias entre cada medida de la lista y la siguiente. La distancia entre dos medidas con velocidades $[vx_1,vy_1]$ y $[vx_2,vy_2]$ y tiempos t_1 y t_2, se calcula como:

$$(t_2 - t_1)\sqrt{vx_1^{\,2} + vy_1^{\,2}}$$

  Si se detecta que en alguno de los cálculos la diferencia de tiempo t_2-t_1 es mayor que 5 segundos entonces el cálculo de la distancia resultará incorrecto y para avisar de este problema se lanzará la excepción `IntervalosLargos`

- `inclinacionMaxFiltrada`(): Retorna la máxima inclinación según el eje Z en valor absoluto, filtrada de modo que los valores en los que haya variaciones muy altas con respecto a los colindantes no se tengan en cuenta (y así evitar problemas de ruido en el sensor que mide la inclinación). Para este método usar el siguiente pseudocódigo

```
/** Retorna la máxima inclinación según el eje Z, filtrada */
método inclinacionMaxFiltrada() retorna real
   real inclMaxFilt=0 // máxima inclinacion filtrada que se encontró hasta ahora
   constante real maxVariacion=5
   // recorre todas las casillas de lista desde la segunda a la penúltima
   para i desde 1 hasta tamaño de lista-2 hacer
      real inclinacion= | inclinacion[2] de la casilla i de lista |
      // si la inclinación supera la máxima encontrada hasta ahora,
      // trabajamos con ella
      si inclinacion>inclMaxFilt entonces
         // buscamos una variación grande entre la inclinación de la casilla i y
         // las dos anteriores y las dos posteriores si existen
         booleano demasiadaVariacion=false
         entero j= max(0,i-2)
         mientras j<= min (i+2,tamaño de lista-1) y no demasiadaVariacion hacer
            si  | (inclinacion[2] de la casilla j de lista) - inclinacion| > maxVariacion
               demasiadaVariacion=true;
            fin si
            j++
         fin mientras
         // si no hay demasiada variación anotamos el nuevo máximo
         si no demasiadaVariacion entonces
            inclMaxFilt=inclinacion
         fin si
      fin si
   fin para
   retorna inclMaxFilt
fin método
```

  nota: la notación `|valor|` significa valor absoluto

- `velocidadAlta`(): indica si en alguna de las medidas el módulo de la velocidad (raíz cuadrada de los cuadrados de las velocidades X e Y) es superior al parámetro `velMax`. Este método consiste en una búsqueda de alguna casilla que incumpla las limitación de velocidad máxima. Si se encuentra esta casilla retornar `true`. Si no hay ninguna, retornar `false`. Seguir por tanto el algoritmo de búsqueda visto en clase.

Valoración: Constructor y resto de los métodos: 1.25 puntos cada uno