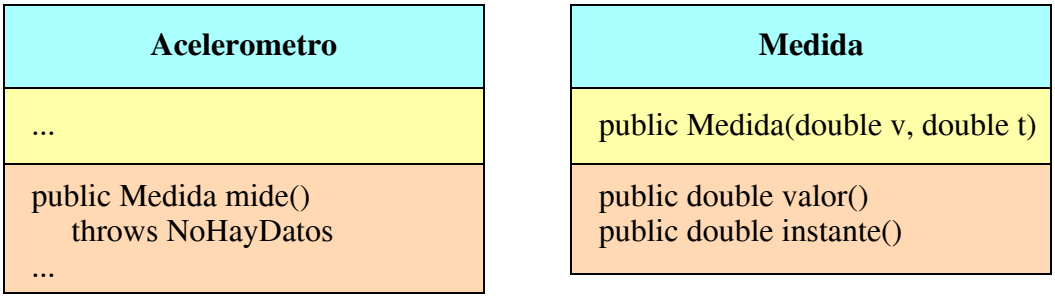
Examen Final de Fundamentos de Computadores y Lenguajes (Licenciado en Física)

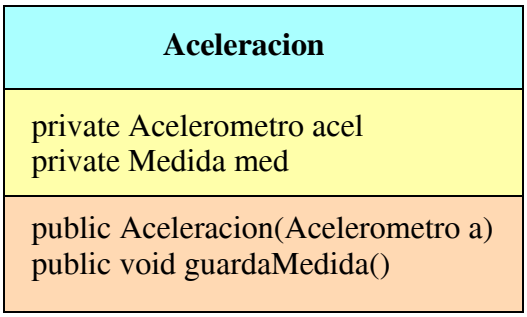
Junio 2009

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

1) Se dispone de las clases `Acelerometro` y `Medida` que responden al diagrama de clases que se muestra en la figura



Escribir el código de una clase llamada `Aceleracion` que responde al diagrama que se muestra a la derecha. Sus métodos hacen lo siguiente:



- Constructor: copia en `acel` la referencia al acelerometro `a`
- `guardaMedida()`: llama al método `mide()` del acelerómetro. Si no hay error guarda en `med` la medida retornada por el método. Si se lanza `NoHayDatos` guarda en `med` un objeto de la clase `Medida` creado con `v` y `t` iguales al valor real `NaN`.

2) Codificar en Java el siguiente algoritmo descrito mediante pseudocódigo

```
método maximaDiferencia(array de reales x) retorna real
   real maxDif=0
   para cada i desde 0 hasta el penúltimo índice de x
       dif=valor absoluto(x[i]-x[i+1])
       si dif>maxDif entonces
           maxDif=dif
       fsi
   fpara
   retorna maxDif
fin del método
```

3) Indicar utilizando la notación O(n) el tiempo de ejecución del algoritmo de la cuestión 2

4) Indicar para cada posible valor de la variable `i` que aparece en la tabla el resultado obtenido después de ejecutarse el siguiente fragmento de código Java:

i
0
1
2
3
4

```
switch (i) {
    case 0:
       j=3;
    case 1:
       j=7;
    case 2:
       j=j+3;
    case 3:
       j=j-1;
    default:
       i=0;
}
System.out.println("I="+i+", J="+j);
```

5) Escribir las órdenes que sería necesario dar a un intérprete de órdenes en Linux/Unix para crear una copia en una memoria USB montada en el directorio `/media/USB2` de los siguientes ficheros del directorio `/home/usuario`

- los acabados en `.java`
- la carpeta `doc` completa, con todos sus contenidos

Suponer que se desconoce el directorio de trabajo.

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

Junio 2009

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

Se desea implementar un sistema de cálculo de la posición de un objeto que se mueve en el plano, basado en la integración numérica de las aceleraciones tangencial y normal proporcionadas por un acelerómetro de dos dimensiones.

Se dispone para ello de la clase `Posicion`, cuyos objetos sirven para almacenar la posición de un objeto en el plano, y cuya interfaz se muestra a continuación:

```
public class Posicion {
    /**
     * Constructor al que se le pasan las coordenadas del objeto
     */
    public Posicion(double x, double y) {...}

    /**
     * Retorna la coordenada x
     */
    public double coordX() {...}

    /**
     * Retorna la coordenada y
     */
    public double coordY() {...}

    /**
     * Distancia entre dos puntos que se pasan como parámetros
     */
    public static double distancia(Posicion p, Posicion q) {...}
}
```

Se pide escribir en Java la clase `Trayectoria` que almacena la trayectoria de un objeto en el plano. La clase dispone de los atributos y métodos que se indican:

```
public class Trayectoria
{
    // Lista que contiene las posiciones de la trayectoria
    private ArrayList<Posicion> pos;
    // Instante de la ultima medida,
    // en segundos desde el inicio del experimento
    private double instanteUltimaMedida; // segundos
    // Vector de velocidad actual
    private double velX, velY;  // m/s
    // Número de errores detectados
    int numErrores;
    // Constante que indica la aceleración máxima permitida
    public static final double acelMax=10; // m/seg2
```

```
    /**
     * Constructor al que se le pasa la posicion y velocidad inicial del
     * objeto. Crea la lista pos y le añade un objeto de la clase Posicion
     * conteniendo la posición inicial dada por posX y posY.
     * Pone instanteUltimaMedida y numErrores a cero y copia la
     * velocidad inicial dada por velX y velY en los respectivos atributos
     */
    public Trayectoria(double posX, double posY, double velX, double velY)
    {...}

    /**
     * Calcula y almacena las nuevas velocidades y posiciones a partir de
     * las aceleraciones normal y tangencial dadas y del tiempo transcurrido
     * desde la ultima medida hasta el instante actual
     */
    public void registraMedida(double acelNormal, double acelTangencial,
       double instanteActual) throws DemasiadaAceleracion
    {...}

    /**
     * Llama a registraMedida y si falla incrementa el número de errores
     */
    public void anotaMedida(double acelNormal, double acelTangencial,
       double instanteActual)     {...}

    /**
     * Retorna la distancia recorrida en metros por el objeto, obtenida
     * sumando las distancias para ir desde cada punto de la trayectoria
     * al siguiente
     */
    public double distanciaTotal() {...}

    /**
     * Retorna la distancia recorrida en metros por el objeto entre los
     * puntos de las num últimas medidas; si num es mayor o igual que el
     * número total de medidas, considerarlas todas
     */
    public double distanciaRecorrida(int num) {...}
}
```

Lo que debe hacer cada método se describe a continuación:

- constructor: da valor inicial a cada atributo según lo descrito en el comentario
- `registraMedida`: Si las aceleraciones normal o tangencial exceden la aceleración máxima permitida, se lanza `DemasiadaAceleracion`. En caso contrario se realizan

los cálculos de la velocidad y posicion nuevas, de acuerdo con las siguientes expresiones:

$$v = \sqrt{v_x^2 + v_y^2}$$

$$a_x = (a_t \cdot v_x - a_n \cdot v_y)/v \qquad a_y = (a_t \cdot v_y + a_n \cdot v_x)/v$$

$$v_x' = v_x + a_x \cdot \Delta t \qquad v_y' = v_y + a_y \cdot \Delta t$$

$$p_x' = p_x + ((v_x' + v_x) \cdot \Delta t)/2 \qquad p_y' = p_y + ((v_y' + v_y) \cdot \Delta t)/2$$

siendo v la velocidad lineal, (a_x,a_y) el vector aceleración, (v_x,v_y) el vector velocidad anterior, (v_x', v_y') el vector velocidad nueva, (p_x,p_y) el vector posicion anterior, (p_x', p_y') el vector posición nueva, y Δt la diferencia entre el instante actual y el instante de la última medida. La velocidad anterior está guardada en `velX` y `velY`. La posición anterior es la posición guardada en el último elemento de la lista `pos`. Después de hacer los cálculos, la nueva velocidad calculada se guardará en `velX` y `velY`, y la nueva posición en un objeto de la clase `Posicion` que se añadirá al final de la lista `pos`. Finalmente se copiará el instante actual en el instante de la última medida.

- `anotaMedida`: llama a `registraMedida` con los datos que se le pasan como parámetros, y si se lanza `DemasiadaAceleracion` incrementa el número de errores.
- `DistanciaTotal`: Si `numErrores` es mayor que 10 retorna el valor real `NaN`. En caso contrario calcula la distancia recorrida por el objeto para ir del primer punto de la trayectoria al último, pasando sucesivamente por todos los intermedios. La distancia entre dos posiciones se obtiene con el método `distancia` de la clase `Posicion`.
- `DistanciaRecorrida`: Si `numErrores` es mayor que 10 retorna el valor real `NaN`. En caso contrario calcula la distancia recorrida en el tramo de la trayectoria situado entre los `num` últimos puntos almacenados en la lista `pos` (por ejemplo, si `pos` tiene 10 datos y `num` vale 4, hay que retornar la suma de las distancias necesarias para recorrer los tramos entre los puntos del 6 al 9, es decir de los puntos 6 al 7, del 7 al 8, y del 8 al 9). Si `num` es mayor o igual que el número total de puntos, retornar la distancia completa recorrida desde el primer punto al último. La distancia entre dos posiciones se obtiene con el método `distancia` de la clase `Posicion`.

Nota: se valorarán los métodos a desarrollar de la siguiente forma:

- constructor: 0.5 puntos
- `registraMedida`: 2 puntos
- `anotaMedida`: 1 punto
- `distanciaTotal`: 1 punto
- `distanciaRecorrida`: 1.5 puntos; si se hace este método no es necesario hacer `distanciaTotal`