Práctica 7

***Objetivos*: Practicar recorridos y búsquedas en tablas**

***Descripción*: Escribir un programa que permita dibujar en una ventana la trayectoria de un vehículo en un plano**

Se dispone de la clase ya realizada `Vehiculo` que simula la cinemática de un vehículo que se mueve sin rozamiento en un plano

Consultar la documentación de la clase para ver una descripción de los métodos

Vehiculo
<atributos privados>
Vehiculo() avanza(real acelX, real acelY) real tiempo() real posX() real posY() real velX() real velY()

Práctica 7

Se pide realizar estas dos clases:

- **`Trayectoria`: Tiene una lista de puntos por los que pasa el vehiculo simulado, cada uno de ellos con sus coordenadas *x* e *y***
  - **las coordenadas se expresan en metros**
  - **tiene una operación para simular el movimiento horizontal y vertical de un vehículo que, partiendo del reposo, dispone de un motor que actúa por un tiempo y luego se para**
- **`Simulacion`: Tiene el programa principal**

Práctica 7: Clase Trayectoria

La clase tiene el diagrama de clase que se muestra. Sus contenidos son:

- dos arrays de números reales llamados `posX` e `posY`, para guardar una lista de puntos en el plano (el punto `i` es el de coordenadas `posX[i]`, `posY[i]`)
- un objeto de la clase `Vehiculo`
- constructor: se le pasa el tiempo que está el motor encendido y lo usa para crear los arrays `posX` y `posY` de tamaño igual al *doble* del `tiempoMotor`; además inicializa `v` con un nuevo objeto de la clase `Vehiculo`

Trayectoria
real posX[0..N-1] real posY[0..N-1] Vehiculo v
Trayectoria(real tiempoMotor) simula() pinta(real xOrigen, real yOrigen, real escala) booleano cercaDe(real cx, real cy) real longitud()

Práctica 7: Clase Trayectoria

- `simula()`: debe llamar múltiples veces al método `avanza()` del vehículo y rellenar a cada paso los arrays `posX` y `posY` con la posición del vehículo alcanzada (obtenida con sus métodos `posX()` y `posY()`); se simularán dos etapas:
  - En la primera etapa se simula que el motor está encendido y para ello se llama `N/2` veces a `avanza()` con `acelX=0.12*v.tiempo(), acelY=0.12*v.tiempo()-9.8`
  - En la segunda etapa el motor está apagado y se llama `N/2` veces a `avanza()` con `acelX=0, acelY=-9.8` (sólo influye la gravedad)
    - Nota: llamamos `N` a la longitud de los arrays `posX` y `posY`

Práctica 7: Clase Trayectoria

- **pinta()**: dibuja en una ventana de la clase **Dibujo** la trayectoria seguida por el vehículo; para ello:
  - crea la ventana para dibujar de tamaño **600x600**
  - dibuja cada punto **posX[i]**, **posY[i]**, **0<=i<N**, haciendo esta transformación de coordenadas

$$coordXdibujo = posX[i]/escala + xOrigen$$

$$coordYdibujo = 600 - (posY[i]/escala + yOrigen)$$

  - cada punto transformado se dibuja con el método **dibujaPunto()**
  - al final de todo se llama al método **espera()** para pintar el dibujo
  - usar el esquema de recorrido en tablas
    - Nota: la transformación de coordenadas tiene en cuenta una posición del origen (xOrigen,yOrigen) y un factor de escala.

Práctica 7: Clase Trayectoria

- **cercaDe()**: retorna un booleano que indica si alguno de los puntos de la trayectoria está cerca (en un radio de 10 metros) del punto de coordenadas **cx**,**cy**
  - usar para este método el esquema de búsqueda en tablas de teoría
  - la distancia entre dos puntos se calcula como

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- **longitud()**: retorna la longitud del recorrido consistente en ir desde el primer punto de la trayectoria hasta el último, pasando por los intermedios en el orden en que aparecen en las tablas
  - usar el esquema de recorrido (parcial) en tablas

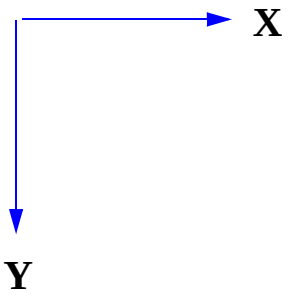
Práctica 7: Clase `Simulacion`

Esta clase tiene el programa principal, que hace lo siguiente:

- **Crea un objeto de la clase `Trayectoria`, con un tiempo de motor de 500 s**
- **Llama al método `simula()`**
- **Llama al método `pinta()`, poniendo el origen en (0,300) y con un factor de escala de 2500**
- **Lee de teclado las coordenadas de un punto y comprueba si la trayectoria tiene algún punto cerca del punto indicado, usando `cercaDe()`; muestra este resultado en pantalla**
- **Llama al método `longitud()` y muestra el resultado en pantalla**

Práctica 7 (cont.)

El origen de coordenadas está en la esquina superior izquierda de la pantalla



En la transformación de coordenadas las coordenadas `Y` se invierten, ya que en el dibujo las coordenadas son positivas hacia abajo (por eso hacemos `600-coordY`)

Práctica 7 (cont.)

***Informe*. Entregar:**

- la especificación y el diseño de los métodos de la clase `Trayectoria`
- el código de las clases `Trayectoria` y `Simulacion`
- los resultados obtenidos y una captura de pantalla del dibujo

UNIVERSIDAD DE CANTABRIA