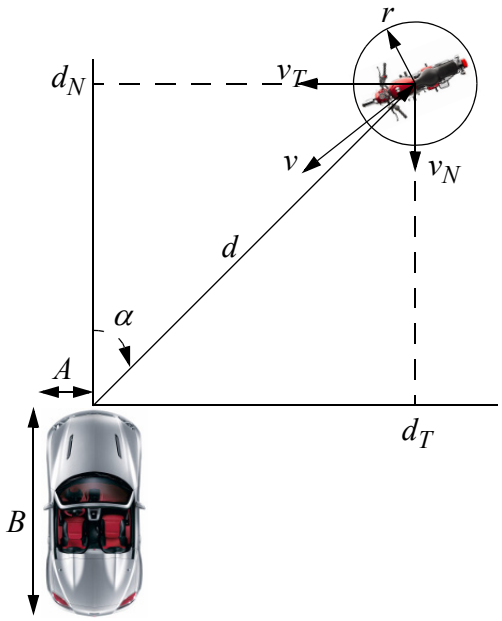
# Examen de Prácticas de Programación
# (Grados en Física y Matemáticas)

**Septiembre 2015**

El pocos años está previsto que muchos automóviles tengan capacidades de conducción automática. Se desea implementar parte del software de un sistema para evitar colisiones. El automóvil dispone de un radar que ve los obstáculos que tiene delante y mediante un software de análisis de datos determina la velocidad y posición de cada obstáculo, así como su radio.

| **Obstáculo** |
|---|
| -int id<br>-double vT<br>-double vN<br>-double d<br>-double alfa<br>-double r |
| +Obstáculo(int id, double r)<br>+set(double vT, double vN,<br>double d, double alfa)<br>+double tAlcance(double vC)<br>+double tRebase(double vC, double B)<br>+double margenAlcance(double vC)<br>+double margenRebase(<br>double vC, double B)<br>+int id()<br>+double r() |




La clase Obstáculo está ya implementada y dispone de los siguientes atributos:

- id: entero que identifica al obstáculo
- vT: velocidad tangencial ($v_T$), en m/s
- vN: velocidad normal ($v_N$), en m/s
- d: distancia al obstáculo en m
- alfa: ángulo al obstáculo (α), en grados
- r: radio del obstáculo en m

Los métodos de esta clase son:

- *Constructor*: Recibe como parámetros los valores iniciales de los atributos id y r
- set(): Recibe como parámetros los valores actualizados de los atributos vT, vN, d y alfa
- tAlcance(): Calcula el tiempo hasta alcanzar el obstáculo en la dirección del coche, dada la velocidad del coche vC ($v_C$) en m/s

$$t_{alc} = \frac{d_N - r}{v_C - v_N}$$

- tRebase(): Calcula el tiempo hasta rebasar el obstáculo en la dirección del coche, dada la velocidad del coche vC ($v_C$) en m/s y la longitud del coche B en m

$$t_{reb} = \frac{d_N + r + B}{v_C - v_N}$$

- margenAlcance(): Calcula el margen de distancia tangencial entre el coche y el obstáculo cuando transcurra el tiempo de alcance

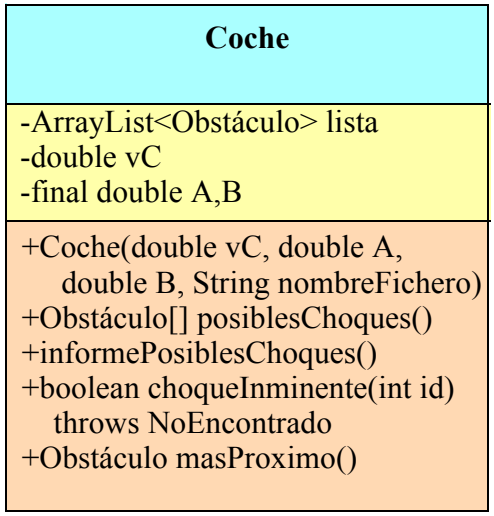
$$margen_{alc} = d_T + v_T t_{alc}$$

- margenRebase(): Calcula el margen de distancia tangencial entre el coche y el obstáculo cuando transcurra el tiempo de rebase

$$margen_{reb} = d_T + v_T t_{reb}$$

- id(): Observador del atributo id.
- r(): Observador del atributo r.

Lo que se pide es implementar en Java la clase Coche pensada para simular y probar el funcionamiento del sistema de detección de colisiones. Su diagrama de clases se muestra en la figura. La clase dispone de los siguientes atributos:




- lista: una lista de obstáculos
- vC: velocidad del coche ($v_C$) en m/s
- A, B: dimensiones del coche (semianchura y longitud) en m

Los siguientes métodos de la clase se dan ya resueltos:

- *constructor*: recibe como parámetros la velocidad del coche vC en m/s, las dimensiones A y B del coche en m y el nombre de un fichero de texto del que se leen datos para rellenar la lista de obstáculos.
- posiblesChoques(): Retorna un array conteniendo todos los obstáculos para los que se detecta un posible choque.

Se pide implementar los siguientes métodos de la clase Coche:

- informePosiblesChoques(): Obtiene un array con los posibles choques invocando al método posiblesChoques(). Posteriormente muestra en pantalla todos estos obstáculos, uno por línea, con su identificador y el tiempo de alcance.
- choqueInminente(): Busca en la lista el Obstáculo cuyo identificador coincide con el parámetro id. Si lo encuentra retorna un booleano indicando si su tiempo de alcance de ese obstáculo es menor que 2 segundos o no. Si no lo encuentra lanza NoEncontrado.
- masProximo(): Retorna el obstáculo de la lista cuyo tiempo de alcance es menor.

La excepción NoEncontrado está ya creada en una clase aparte.

Finalmente, se pide hacer un programa principal de prueba de los métodos de la clase Coche, en una clase aparte, que haga lo siguiente:

a. Crea un objeto de la clase Coche con velocidad 18.5 m/s, dimensiones A=1.3, B=4.5 y nombre de fichero "obstaculos.txt".

b. Muestra en pantalla el resultado de invocar al método choqueInminente() para el identificador 13457 y luego para el identificador 13458. Si se lanza NoEncontrado en alguna de estas llamadas se muestra un mensaje de error, se salta el paso c), pero se sigue por el paso d).

c. Muestra en pantalla el identificador del obstáculo más próximo obtenido con el método masProximo().

d. Invoca el método informePosiblesChoques() para obtener un informe de los posibles choques.

Entrega: Entregar un informe conteniendo el código de la clase Coche, el código del programa principal y una captura de pantalla de la ejecución del programa principal.

*Valoración*:

- informePosiblesChoques: 2 puntos
- choqueInminente(): 2 puntos
- masProximo(): 2 puntos
- programa principal: 2 puntos
- captura de pantalla del programa principal: 2 puntos