# UD03.EXAME Práctico

DAM1-Programación 2024-25
2024/12/13



- Puedes utilizar apuntes y materiales que consideres pero deberás realizar los programas individualmente. En caso contrario se retirará el examen.
- Realiza programas bien estructurados, legibles, con comentarios, líneas en blanco, identificadores adecuados, etc.
- Cuida la interacción con el usuario, presentando la información de forma clara y ordenada.
- Utiliza los casos de prueba de ejemplo para probar tu programa y definir la salida por pantalla.

- Crea un **paquete** de nombre **ud3.xxxexamen** donde agrupar los ficheros de código fuente, donde **xxx** son las iniciales de tu nombre y apellidos. Si es necesario crea un nuevo proyecto/carpeta Java.
- Crea dentro del paquete ficheros **.java** por cada ejercicio con el nombre apropiado.
- **Incluye al inicio de cada programa un comentario con tu nombre y apellidos.**
- **Entrega la carpeta xxxexamen comprimida con los archivos de código fuente .java.**

- **Tiempo estimado:** 1:30 horas

# 1. Coche de Carreras (5)

**CocheCarreras.java**

Diseña una clase para modelar coches de carreras para un juego por turnos.

Cada coche tendrá los siguientes atributos:

- nombre: una cadena que lo identificará.
- velocidad máxima, entre 1 y 100, en metros por segundo.
- número máximo de "turbos" que podrá consumir el coche antes de repostar.
- autonomía: indica el número de metros que puede recorrer el coche con el depósito lleno (o la batería totalmente cargada).
- Otros atributos almacenarán la autonomía restante y el número de turbos restantes, y la distancia recorrida en metros.

Los atributos serán privados y la clase dispondrá de los siguentes métodos para acceder a ellos:

- `getNombre()`
- `getAutonomiaRestante()`
- `getTurbosRestantes()`
- `getDistanciaRecorrida()`
- Podrán añadirse los métodos *getters* y *setters* que se consideren necesarios, siempre que su implementación mantenga la coherencia de los atributos (por ejemplo, la velocidad máxima no puede ser menor que 1 ni mayor que 100, la autonomía restante no puede ser negativa, etc.).

Un coche podrá construirse:

- indicando su nombre, la velocidad máxima, el número máximo de turbos y la autonomía máxima como parámetros en su constructor. La distancia recorrida inicial al crear el coche será cero. Asimismo el coche dispondrá inicialmente de toda su autonomía y sus turbos.
- indicando sólo su nombre, su velocidad máxima y su autonomía, en cuyo caso dispondrá de 3 turbos por defecto y los mismos valores iniciales que en el constructor anterior.

Otros métodos a implementar:

- `avanzar()`: intentará hacer avanzar el coche un número aleatorio de metros entre 1 y la velocidad máxima. Sin embargo, el número real de metros recorridos estará limitado por la autonomía restante del coche, que se decrementará adecuadamente y no podrá ser inferior a cero. El método incrementará también la distancia recorrida del coche y devolverá el valor real de metros recorridos.
- `usarTurbo()`: Al usar un turbo, si es que quedan turbos disponibles, el coche avanzará un número de metros igual al 150% de su velocidad máxima. Al igual que en el método anterior, los metros reales recorridos estarán limitados por la autonomía restante, que se decrementará al igual que se incrementará la distancia recorrida del coche. También se decrementará el número de turbos restante. El método devolverá el valor real de metros recorridos.
- `repostar()`: restablecerá la autonomía y el número de turbos a sus valores máximos.
- `reiniciarDistancia()`: reiniciará la distancia recorrida a cero.
- `mostrar()`: Imprimirá las características del coche con el siguiente formato:

```
Coche: <Nombre>
Velocidad Máxima: <velocidadMaxima> mps
Turbos: <turbosRestantes>/<maxTurbos>
Autonomía: <autonomiaRestante>/<autonomiaMaxima>
```

Incluye en la clase un método **main()** de prueba que deberá funcionar correctamente con el siguiente código:

```
System.out.println("RAYO");
System.out.println("====");
CocheCarreras rayo = new CocheCarreras("Rayo", 90, 400);
rayo.mostrar();
rayo.avanzar();
rayo.usarTurbo();
rayo.mostrar()

System.out.println("TRUENO");
System.out.println("======");
CocheCarreras trueno = new CocheCarreras("Trueno", 60, 5, 300);
trueno.mostrar();
while (trueno.getAutonomiaRestante() != 0)
    trueno.usarTurbo();
trueno.mostrar();
trueno.repostar();
trueno.mostrar();
```

# 2. Juego de Carreras (5)

**JuegoCarreras01.java**

Crea una clase principal llamada JuegoCarreras01 en la que instancies al menos dos coches (por ejemplo Rayo y Trueno) con diferentes configuraciones y simules una carrera entre ambos.

**Dinámica de la carrera:**

- La carrera se desarollará por turnos hasta que uno de ellos recorra un número de metros establecido (por ejemplo 500).
- Comenzará el coche que tenga mayor velocidad máxima
- En cada turno un coche:
    - si no tiene autonomía deberá perder el turno repostando.
    - si tiene turbos disponibles, tendrá una probabilidad del 25% de usar un turbo.
    - si no reposta ni usa un turbo avanzará normalmente.

**Salida por pantalla:**

- Antes de empezar la carrera muestra las características de ambos coches.
- Muestra los avances de cada coche en cada turno indicando su nombre, el número de metros avanzados en ese turno y el total recorridos hasta el momento. Usa el siguiente formato de salida de ejemplo:

```
Rayo avanza 53 metros (Total recorrido: 123 metros).
```

- Si el coche utiliza un turbo muestra los metros recorridos y el número de turbos restantes con el siguiente formato:

```
Trueno usa un turbo y avanza 135 metros. Quedan 2 turbos.
(Total recorrido: 423 metros).
```

- Muestra si el coche *para* a repostar.

```
Rayo para a repostar.
```

- Al finalizar la carrera, muestra el nombre del coche ganador .

```
¡Trueno ha cruzado la meta y es el ganador!
```