

## Ejercicio: Sistema de Vehículos

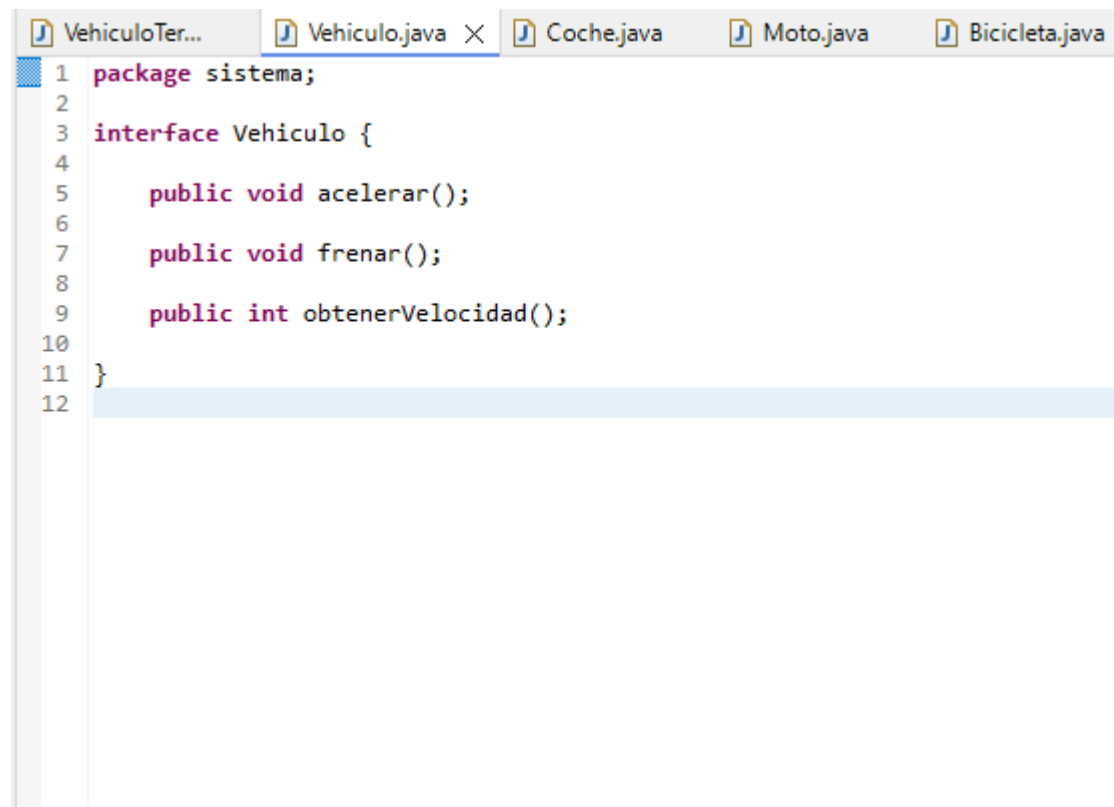
Vas a implementar un sistema básico de vehículos utilizando herencia e interfaces. La idea es crear una jerarquía de clases para representar distintos tipos de vehículos y sus características, respetando el principio de abstracción.

### Requisitos:

#### 1. Interface Vehiculo:

- Métodos:

- `void acelerar()`: Aumenta la velocidad del vehículo.
- `void frenar()`: Disminuye la velocidad del vehículo.
- `int obtenerVelocidad()`: Devuelve la velocidad actual del vehículo.



```
1 package sistema;
2
3 interface Vehiculo {
4
5     public void acelerar();
6
7     public void frenar();
8
9     public int obtenerVelocidad();
10
11 }
12
```

#### 2. Clase abstracta VehiculoTerrestre que implementa Vehiculo:

- Atributos comunes a todos los vehículos terrestres:
  - `int velocidad`: La velocidad actual del vehículo.
  - `int ruedas`: Número de ruedas del vehículo.
- Métodos implementados:
  - Implementar los métodos de la interfaz, pero dejando la lógica específica de `acelerar` y `frenar` para las subclases.

```

1 package sistema;
2
3 public class VehiculoTerrestre implements Vehiculo {
4
5     private int velocidad;
6     private int ruedas;
7
8     public VehiculoTerrestre(int velocidad, int ruedas) {
9         this.velocidad = velocidad;
10        this.ruedas = ruedas;
11    }
12
13    @Override
14    public void acelerar() {
15
16    }
17
18    @Override
19    public void frenar() {
20
21    }
22
23    @Override
24    public int obtenerVelocidad() {
25        return velocidad;
26    }
27
28    public void establecerVelocidad(int velocidad) {
29        this.velocidad = velocidad;
30    }
31
32    public int getRuedas() {
33        return ruedas;
34    }
35 }

```

### 3. Clase Coche que extiende de VehiculoTerrestre:

- Atributos:
  - boolean tieneAireAcondicionado: Indica si el coche tiene aire acondicionado.
- Métodos:
  - acelerar(): Incrementa la velocidad en 10 km/h.
  - frenar(): Reduce la velocidad en 5 km/h.
- Implementar el constructor y un método para activar/desactivar el aire acondicionado.

```
1 package sistema;
2
3 public class Coche extends VehiculoTerrestre {
4
5     public boolean tieneAireAcondicionado;
6
7     public Coche(int velocidad, int ruedas, boolean tieneAireAcondicionado) {
8         super(velocidad, ruedas);
9         this.tieneAireAcondicionado = tieneAireAcondicionado;
10    }
11
12    @Override
13    public void acelerar() {
14        int velocidad = obtenerVelocidad();
15        velocidad += 10;
16        establecerVelocidad(velocidad);
17        System.out.println("El coche ha acelerado. Velocidad actual: " + velocidad + " km/h.");
18    }
19
20    @Override
21    public void frenar() {
22        int velocidad = obtenerVelocidad();
23        velocidad -= 5;
24        if (velocidad < 0) {
25            velocidad = 0;
26        }
27        establecerVelocidad(velocidad);
28        System.out.println("El coche ha frenado. Velocidad actual: " + velocidad + " km/h.");
29    }
30
31    public void activarAireAcondicionado() {
32        if (!tieneAireAcondicionado) {
33            tieneAireAcondicionado = true;
34            System.out.println("El aire acondicionado ha sido activado.");
35        } else {
36            System.out.println("El aire acondicionado ya está activado.");
37        }
38    }
39
40    public void desactivarAireAcondicionado() {
41        if (tieneAireAcondicionado) {
42            tieneAireAcondicionado = false;
43            System.out.println("El aire acondicionado ha sido desactivado.");
44        } else {
45            System.out.println("El aire acondicionado ya está desactivado.");
46        }
47    }
48 }
```

#### 4. Clase Moto que extiende de VehiculoTerrestre:

- Atributos:
  - boolean tieneCasco: Indica si el conductor está usando casco.
- Métodos:
  - acelerar(): Incrementa la velocidad en 20 km/h.
  - frenar(): Reduce la velocidad en 10 km/h.
- Implementar el constructor y un método para verificar si el casco está puesto.

```
1 package sistema;
2
3 public class Moto extends VehiculoTerrestre {
4
5     public boolean tieneCasco;
6
7     public Moto(int velocidad, int ruedas, boolean tieneCasco) {
8         super(velocidad, ruedas);
9         this.tieneCasco = tieneCasco;
10    }
11
12    @Override
13    public void acelerar() {
14        int velocidad = obtenerVelocidad();
15        velocidad += 20;
16        establecerVelocidad(velocidad);
17        System.out.println("La moto ha acelerado. Velocidad actual: " + velocidad + " km/h.");
18    }
19
20    @Override
21    public void frenar() {
22        int velocidad = obtenerVelocidad();
23        velocidad -= 10;
24        if (velocidad < 0) {
25            velocidad = 0;
26        }
27        establecerVelocidad(velocidad);
28        System.out.println("La moto ha frenado. Velocidad actual: " + velocidad + " km/h.");
29    }
30
31    public void Cascopuesto() {
32        if (!tieneCasco) {
33            System.out.println("El casco no está puesto.");
34        } else {
35            System.out.println("El casco está puesto.");
36        }
37    }
38 }
```

##### 5. Clase Bicicleta que extiende de VehiculoTerrestre:

- Atributos:

- boolean tieneTimbre: Indica si la bicicleta tiene timbre.

- Métodos:

- acelerar(): Incrementa la velocidad en 5 km/h.
- frenar(): Reduce la velocidad en 2 km/h.

```
VehiculoTer... Vehiculo.java Coche.java *Moto.java Bicicleta.java X Main.java Main.java
1 package sistema;
2
3 public class Bicicleta extends VehiculoTerrestre {
4
5     public boolean tieneTimbre;
6
7     public Bicicleta(int velocidad, int ruedas, boolean tieneTimbre) {
8         super(velocidad, ruedas);
9         this.tieneTimbre = tieneTimbre;
10    }
11
12    @Override
13    public void acelerar() {
14        int velocidad = obtenerVelocidad();
15        velocidad += 5;
16        establecerVelocidad(velocidad);
17        System.out.println("La bicicleta ha acelerado. Velocidad actual: " + velocidad + " km/h.");
18    }
19
20    @Override
21    public void frenar() {
22        int velocidad = obtenerVelocidad();
23        velocidad -= 2;
24        if (velocidad < 0) {
25            velocidad = 0;
26        }
27        establecerVelocidad(velocidad);
28        System.out.println("La bicicleta ha frenado. Velocidad actual: " + velocidad + " km/h.");
29    }
30
31    public void usarTimbre() {
32        if (tieneTimbre) {
33            System.out.println("¡Ring Ring! El timbre ha sido usado.");
34        } else {
35            System.out.println("Esta bicicleta no tiene timbre.");
36        }
37    }
38 }
```

## MAIN

```
1 package sistema;
2
3 public class Main {
4     public static void main(String[] args) {
5         // Crear una instancia de Moto
6         Moto moto = new Moto(0, 2, true);
7         moto.acelerar();
8         moto.frenar();
9         moto.Cascopuesto();
10        System.out.println("Velocidad actual de la moto: " + moto.obtenerVelocidad() + " km/h");
11
12        System.out.println("-----");
13
14        // Crear una instancia de Coche
15        Coche coche = new Coche(0, 4, false);
16        coche.acelerar();
17        coche.frenar();
18        coche.activarAireAcondicionado();
19        coche.desactivarAireAcondicionado();
20        System.out.println("Velocidad actual del coche: " + coche.obtenerVelocidad() + " km/h");
21
22        System.out.println("-----");
23
24        // Crear una instancia de Bicicleta
25        Bicicleta bicicleta = new Bicicleta(0, 2, true);
26        bicicleta.acelerar();
27        bicicleta.frenar();
28        bicicleta.usarTimbre();
29        System.out.println("Velocidad actual de la bicicleta: " + bicicleta.obtenerVelocidad() + " km/h");
30    }
31 }
```

Console X

<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (26 sept 2024, 13:48:30 – 13:48:32) [pid: 12464]

La moto ha acelerado. Velocidad actual: 20 km/h.  
La moto ha frenado. Velocidad actual: 10 km/h.  
El casco está puesto.  
Velocidad actual de la moto: 10 km/h  
-----  
El coche ha acelerado. Velocidad actual: 10 km/h.  
El coche ha frenado. Velocidad actual: 5 km/h.  
El aire acondicionado ha sido activado.  
El aire acondicionado ha sido desactivado.  
Velocidad actual del coche: 5 km/h  
-----  
La bicicleta ha acelerado. Velocidad actual: 5 km/h.  
La bicicleta ha frenado. Velocidad actual: 3 km/h.  
¡Ring Ring! El timbre ha sido usado.  
Velocidad actual de la bicicleta: 3 km/h

MARP 2024



Repaso JAVA

○ Implementar el constructor y un método para usar el timbre.

MARP 2024