

UNIVERSIDAD POLITÉCNICA DE TLAXCALA

Ingeniería en tecnologías de la información

5° “H”

Fundamentos de la Programación

-Practica 4 Clase Carro-

Eric Jhonathan Anaya Márquez

Jazmín Portillo Michicol

Eder López Villarreal

Karen García Leticia Vásquez

18/01/2024

Profesor Saúl Olaf Loaiza



Diagrama UML clasificador del carro

Carro

-marca: string
-modelo: string
-anoFabricacion: int
-consumoGas: double
-kmRecorridos: double

+Carro(marca:string,modelo: string,anoFabricacion:int)
+getMarca(): String
+getModelo(): string
+getAñoFabricacion(): int
+getConsumoGas(): double
+getKmRecorridos(): double
+setConsumoGas(consumo:double)void
+setKmRecorridos(km:double)void

Código de la clase Carro

```
1 package P00;
2
3 /**
4  * La clase Carro representa un vehículo con atributos como marca, modelo, año de fabricación,
5  * consumo de gas y kilómetros recorridos.
6  */
7 public class Carro {
8     2 usages
9     private final int anoFabricacion; // Año de fabricación del automóvil (atributo final que no puede ser modificado una vez establecido)
10    3 usages
11    private double consumoGas; // Consumo de gas del automóvil en km/l
12    3 usages
13    private double kmRecorridos; // Kilómetros recorridos por el automóvil
14    2 usages
15    private final String marca; // Marca del automóvil
16    2 usages
17    private final String modelo; // Modelo del automóvil
18
19    /**
20     * Constructor de la clase Carro que inicializa los atributos básicos.
21     *
22     * @param marca La marca del automóvil.
23     * @param modelo El modelo del automóvil.
24     * @param anoFabricacion El año de fabricación del automóvil.
25     */
26 }
```

```
21 public Carro(String marca, String modelo, int anoFabricacion) {
22     this.marca = marca;
23     this.modelo = modelo;
24     this.anoFabricacion = anoFabricacion;
25     this.consumoGas = 0.0; // Se inicializa en 0.0 por defecto
26     this.kmRecorridos = 0.0; // Se inicializa en 0.0 por defecto
27 }
28
29 /**
30  * Obtiene la marca del automóvil.
31  *
32  * @return La marca del automóvil.
33  */
34 3 usages
35 public String getMarca() {
36     return marca;
37 }
38
39 /**
40  * Obtiene el modelo del automóvil.
41  *
42  * @return El modelo del automóvil.
43  */
44 3 usages
45 public String getModelo() {
46     return modelo;
47 }
```

```

45     }
46
47     /**
48      * Obtiene el año de fabricación del automóvil.
49      *
50      * @return El año de fabricación del automóvil.
51      */
52     3 usages
53     public int getAnoFabricacion() {
54         return anoFabricacion;
55     }
56
57     /**
58      * Obtiene el consumo de gas del automóvil.
59      *
60      * @return El consumo de gas del automóvil en km/l.
61      */
62     3 usages
63     public double getConsumoGas() {
64         return consumoGas;
65     }
66
67     /**
68      * Obtiene los kilómetros recorridos por el automóvil.
69      *
70      * @return Los kilómetros recorridos por el automóvil.

```

The screenshot shows an IDE window with the following components:

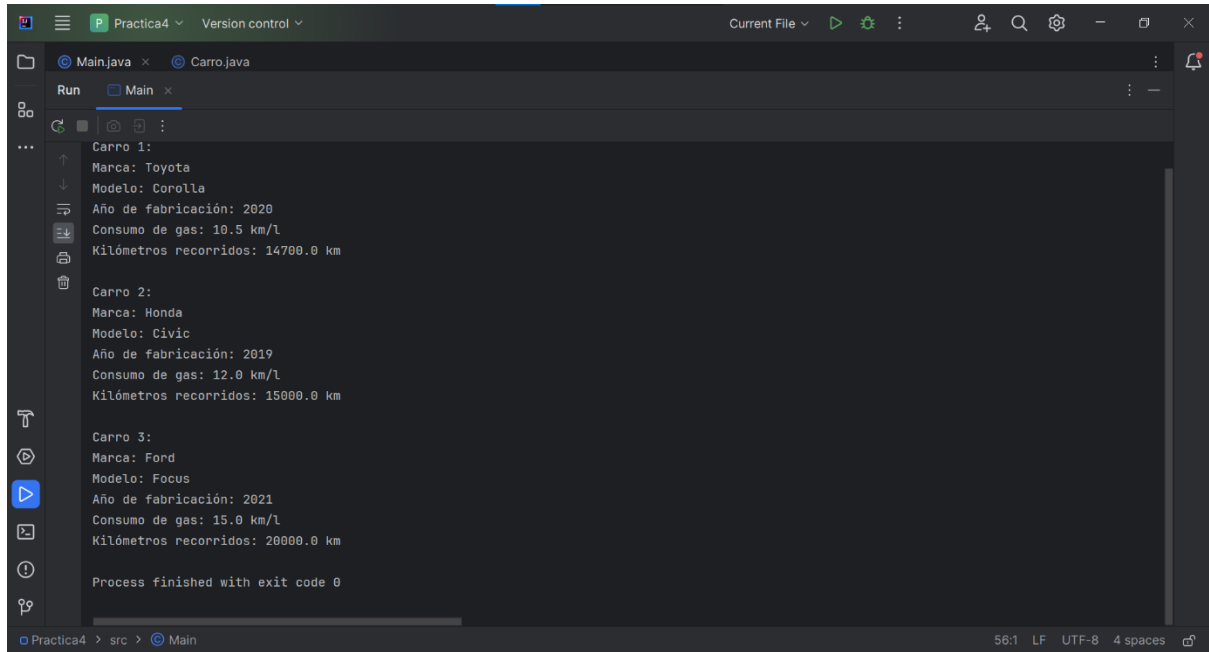
- Top Bar:** Includes a menu icon, a project icon, and tabs for "Practica4" and "Version control". On the right, there are icons for "Current File", "Run", "Settings", "Search", "Help", and window management buttons.
- Left Sidebar:** Contains icons for "Files", "Run and Debug", "Source", "Test", "Run and Coverage", and "Help".
- Editor:** Displays the code for `Carro.java`. The code includes:
 - Line 69: A Javadoc comment `*/`.
 - Line 70: A Javadoc comment `3 usages`.
 - Line 71: The start of the `getKmRecorridos()` method.
 - Line 72: The return statement `return kmRecorridos;`.
 - Line 73: The end of the `getKmRecorridos()` method.
 - Line 74: A Javadoc comment `/**`.
 - Line 75: A Javadoc comment `* Establece el consumo de gas del automóvil.`.
 - Line 76: A Javadoc comment `*`.
 - Line 77: A Javadoc comment `* @param consumo El nuevo valor del consumo de gas en km/l.`.
 - Line 78: A Javadoc comment `*/`.
 - Line 79: A Javadoc comment `3 usages`.
 - Line 80: The start of the `setConsumoGas(double consumo)` method.
 - Line 81: The assignment `this.consumoGas = consumo;`.
 - Line 82: The end of the `setConsumoGas(double consumo)` method.
 - Line 83: A Javadoc comment `/**`.
 - Line 84: A Javadoc comment `* Establece los kilómetros recorridos por el automóvil.`.
 - Line 85: A Javadoc comment `*`.
 - Line 86: A Javadoc comment `* @param km Los nuevos kilómetros recorridos.`.
 - Line 87: A Javadoc comment `*/`.
 - Line 88: A Javadoc comment `3 usages`.
 - Line 89: The start of the `setKmRecorridos(double km)` method.
 - Line 90: The assignment `this.kmRecorridos = km;`.
 - Line 91: The end of the `setKmRecorridos(double km)` method.
- Bottom Bar:** Shows the file path `Practica4 > src > POO > Carro`, the line number `101:1`, and the encoding `CRLF UTF-8 4 spaces`.

Código de clase principal con tres objetos

```
1  /**
2   * UPTx - Fundamentos de Programación Orientada a Objetos
3   * Practica No. 4      Grupo: SH
4   * Tema: Clase Carro
5   * Integrantes: Eder López Villarreal, Jazmin Portillo Michicol, Karen Leticia García Vázquez, Eric Jhonathan Anaya Márquez
6   * Objetivo: Realizar una aplicación donde se realice los registros de un coche
7   *               para establecer su mantenimiento, consumo de gas y km recorrido.
8   * Modificación: 18/ene/2024
9   */
10
11 import P00.Carro;
12 public class Main {
13     public static void main(String[] args) {
14
15         // Crea tres objetos Carro
16         Carro carro1 = new Carro("Toyota", "Corolla", 2020);
17         Carro carro2 = new Carro("Honda", "Civic", 2019);
18         Carro carro3 = new Carro("Ford", "Focus", 2021);
19
20         // Actualiza atributos de los carros
21         carro1.setConsumoGas(10.5);
22         carro1.setKmRecorridos(14700.0);
23         carro2.setKmRecorridos(15000.0);
24         carro2.setConsumoGas(12.0); // Establecer el consumo de gas antes de los kilómetros recorridos
25         carro3.setConsumoGas(15.0);
26         carro3.setKmRecorridos(20000.0);
27     }
28 }
```

```
27
28 // Muestra valores en consola
29 System.out.println("Carro 1:");
30 System.out.println("Marca: " + carro1.getMarca());
31 System.out.println("Modelo: " + carro1.getModelo());
32 System.out.println("Año de fabricación: " + carro1.getAñoFabricacion());
33 System.out.println("Consumo de gas: " + carro1.getConsumoGas() + " km/l");
34 System.out.println("Kilómetros recorridos: " + carro1.getKmRecorridos() + " km\n");
35
36 System.out.println("Carro 2:");
37 System.out.println("Marca: " + carro2.getMarca());
38 System.out.println("Modelo: " + carro2.getModelo());
39 System.out.println("Año de fabricación: " + carro2.getAñoFabricacion());
40 System.out.println("Consumo de gas: " + carro2.getConsumoGas() + " km/l");
41 System.out.println("Kilómetros recorridos: " + carro2.getKmRecorridos() + " km\n");
42
43 System.out.println("Carro 3:");
44 System.out.println("Marca: " + carro3.getMarca());
45 System.out.println("Modelo: " + carro3.getModelo());
46 System.out.println("Año de fabricación: " + carro3.getAñoFabricacion());
47 System.out.println("Consumo de gas: " + carro3.getConsumoGas() + " km/l");
48 System.out.println("Kilómetros recorridos: " + carro3.getKmRecorridos() + " km");
49 }
50 }
```

Funcionalidad



```
Carro 1:
Marca: Toyota
Modelo: Corolla
Año de fabricación: 2020
Consumo de gas: 10.5 km/l
Kilómetros recorridos: 14700.0 km

Carro 2:
Marca: Honda
Modelo: Civic
Año de fabricación: 2019
Consumo de gas: 12.0 km/l
Kilómetros recorridos: 15000.0 km

Carro 3:
Marca: Ford
Modelo: Focus
Año de fabricación: 2021
Consumo de gas: 15.0 km/l
Kilómetros recorridos: 20000.0 km

Process finished with exit code 0
```