

	<b>UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE</b>	<b>DEPARTAMENTO DE COMPUTACION</b>	
		<b>CÓDIGO:</b>	ESPE- DCEX-v1-2023- 011

## ACTIVIDAD AUTÓNOMO N° 2

### DATOS INFORMATIVOS

#### Apellidos y Nombres:

- Montachana Aldana Fabricio Mateo
- Jhoan David Pucó Alcívar
- Renata Alejandra Salazar Caiza
- Gabriel Alejandro Gualpa Sánchez
- Leonardo Darío Tipan Sanipatin

**Fecha:** 13-12-2024


**NRC:** 1323

**Asignatura:** Programación Orientada a Objetos

- **TEMA**
  - *Sistema de Gestión de Parking*

#### Objetivo General:

- Desarrollar un sistema integral para la gestión de un parqueadero, utilizando principios de programación orientada a objetos (POO), una interfaz gráfica intuitiva y una base de datos para almacenamiento y consulta de información. El sistema permitirá registrar, consultar y actualizar la información de los vehículos, optimizando los procesos operativos y mejorando la experiencia del usuario.

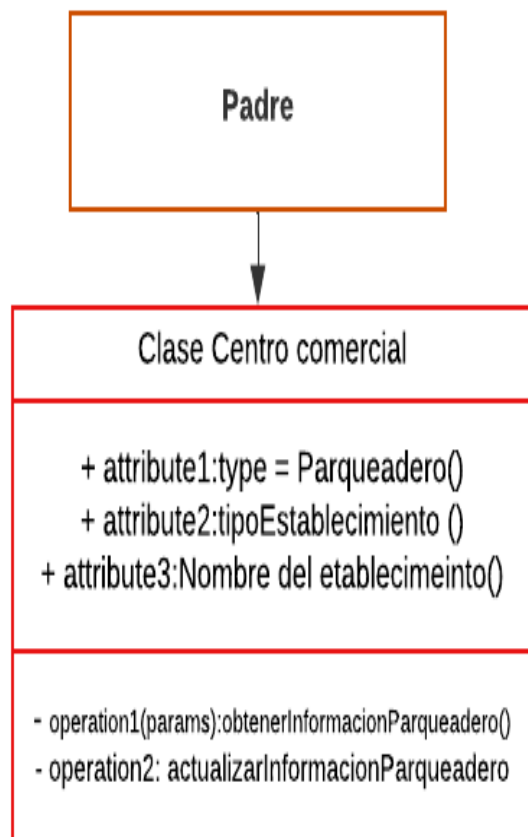
	UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE		DEPARTAMENTO DE COMPUTACION	
	CÓDIGO:		ESPE-DCEX-v1-2023-011	

## 1. DESARROLLO

Desarrolle un sistema para gestionar un parqueadero utilizando POO, una interfaz gráfica, y bases de datos. El sistema debe incluir funcionalidades de registro, consulta y actualización de vehículos.

### Main

- El código implementa un menú interactivo para gestionar un parqueadero. Permite al usuario elegir entre opciones como registrar, actualizar, consultar vehículos o salir del programa. Utiliza la clase Scanner para capturar entradas del usuario y un objeto Parqueadero para manejar las operaciones. Al seleccionar la opción 1, se solicita la placa del vehículo para su registro.



```


Main.java : Vehiculo.java : Parqueadero.java :
1 import java.util.Scanner;
2 import java.util.InputMismatchException;
3
4 public class Main {
5     public static void main(String[] args) {
6         Parqueadero parqueadero = new Parqueadero();
7         Scanner scanner = new Scanner(System.in);
8         int opcion = -1;
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Actualizar vehículo
3. Consulta vehículos
4. Salir
Ingrese una opción: 1

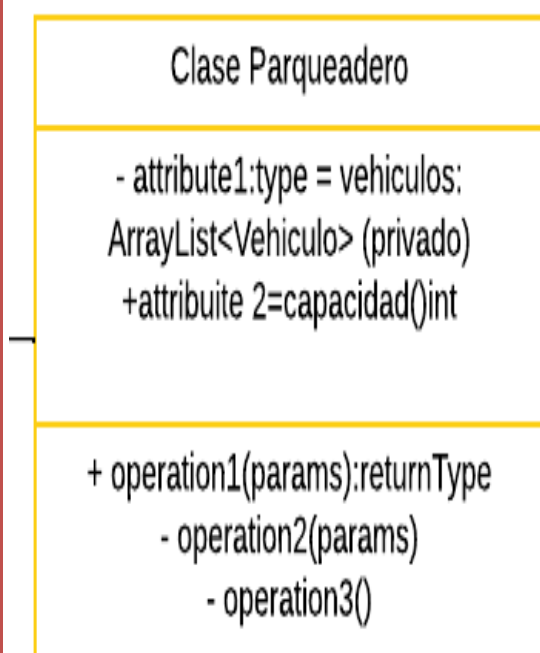
Ingrese la placa del vehículo:
  
```

	<p align="center"><b>UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE</b></p>	<p align="center"><b>DEPARTAMENTO DE COMPUTACION</b></p>	
		<p align="center"><b>CÓDIGO:</b></p>	<p align="center">ESPE- DCEX-v1-2023- 011</p>

### Parqueadero

- La clase Parqueadero gestiona una lista de vehículos utilizando un ArrayList. Permite registrar vehículos mediante su placa y tipo, actualizarlos por su puesto y mostrar la lista de vehículos registrados. La ejecución registra un nuevo vehículo, busca por puesto para actualizarlo y muestra los vehículos almacenados. Si no encuentra un vehículo por el puesto dado, informa al usuario.

La funcionalidad asegura una gestión organizada del parqueadero.



Lo que nos pide en el programa es un sistema relacionado al parqueadero sobre informacion de la clase vehiculos.

```

Parqueadero.java :
80     }
81 }
82
83 private Vehiculo buscarVehiculo(int puestoVehiculo) {
84     for (Vehiculo vehiculo : vehiculos) {
85         if (vehiculo.getPuesto() == puestoVehiculo) {
86             return vehiculo;
87         }
88     }
89     return null;
90 }
91
92 public static void main(String[] args) {
93     Parqueadero parqueadero = new Parqueadero();
94     parqueadero.registrarVehiculo("ABC123", "Automóvil", 1);
95     parqueadero.registrarVehiculo("DEF456", "Motocicleta", 2);
96     parqueadero.mostrarVehiculos();
97     parqueadero.actualizarVehiculo(1, "GHI789", "Camión");
98     parqueadero.mostrarVehiculos();
99 }
100 }

```


input

```

--- Vehículos en el parqueadero ---
Vehiculo{placa='GHI789', tipo='Camión', puesto=1}
Vehiculo{placa='DEF456', tipo='Motocicleta', puesto=2}

..Program finished with exit code 0
Press ENTER to exit console.

```

	<b>UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE</b>	<b>DEPARTAMENTO DE COMPUTACION</b>	
		<b>CÓDIGO:</b>	ESPE-DCEX-v1-2023-011

## Vehículo

La clase Vehiculo representa un vehículo con atributos: placa, tipo, y un puesto único generado automáticamente mediante el contador estático contadorPuesto. El constructor inicializa estos atributos al crear un nuevo objeto. Incluye métodos para obtener (get) y modificar (set) la placa y el tipo del vehículo, pero el puesto es inmutable. Sobrescribe el método toString para mostrar la información del vehículo de forma legible.

### Clase Vehiculo

- attribute1:placa: String ()  
+ attribute2:tipos string()  
- attribute3:puesto int()

+ operation1:getPlaca():String  
+ operation2(params):getPlaca():Void  
+ operation3():getTipo():String  
+ operation4():getPuesto():int  
+ operation5():setTipo():void  
+ operation6():setPuesto():void  
+ operation7():toString():String

```

Main.java : Vehiculo.java : Parqueadero.java :
1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 class Vehiculo {
5     private static int contadorPuesto = 1;
6     private final int puesto;
7     private String placa;
8     private String tipo;
9
10    public Vehiculo(String placa, String tipo) {
11        this.puesto = contadorPuesto++;
12        this.placa = placa;
13        this.tipo = tipo;
14    }

```

```

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Actualizar vehículo
3. Consulta vehículos
4. Salir
Ingrese una opción: 1


Ingrese la placa del vehículo: Pdt9623
Ingrese el tipo de vehículo (Carro, Moto, Camión, etc.): Camion
Vehículo registrado: Puesto: 2, Placa: Pdt9623, Tipo: Camion

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Actualizar vehículo
3. Consulta vehículos
4. Salir
Ingrese una opción: 3

--- Vehículos en el parqueadero ---
Puesto: 1, Placa: jds2415, Tipo: Auto
Puesto: 2, Placa: Pdt9623, Tipo: Camion

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Actualizar vehículo
3. Consulta vehículos
4. Salir
Ingrese una opción:

```

	<b>UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE</b>	<b>DEPARTAMENTO DE COMPUTACION</b>	
		<b>CÓDIGO:</b>	ESPE- DCEX-v1-2023- 011

### Explicación del Código:

Este código implementa un sistema de gestión de parqueadero en Java. La clase Vehículo representa cada vehículo con atributos como placa, tipo y un número único de puesto generado automáticamente. La clase Parqueadero gestiona los vehículos con métodos para registrarlos, actualizarlos y listarlos usando un ArrayList. La clase Main contiene un menú interactivo que permite al usuario realizar estas operaciones.

Al ejecutarse, el programa presenta un menú con opciones:

1. Registrar un vehículo ingresando placa y tipo.
2. Actualizar un vehículo por su puesto, modificando su placa y tipo.
3. Consultar todos los vehículos registrados.
4. Salir del programa.

El sistema captura entradas del usuario, valida los datos y actualiza la lista de vehículos en tiempo real.

Facilita la gestión del parqueadero de forma dinámica y organizada.

## UML

El diagrama UML nos muestra un sistema de parqueadero organizado en clases. Centro Comercial es una clase general que gestiona información del parqueadero. Parqueadero contiene una lista de vehículos y su capacidad. Vehículo es la clase base de la que heredan Carro, Motocicleta y Camión, cada uno con atributos y métodos específicos para su tipo. La estructura usa herencia y composición para manejar la información de los vehículos y su relación con el parqueadero.

	UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE		DEPARTAMENTO DE COMPUTACION
	CÓDIGO:		ESPE-DCEX-v1-2023-011

## *Estructura del Uml*

### *1. Clase Padre:*

- Define un concepto general que agrupa las demás clases.

### *2. Clase Centro Comercial:*

- Relacionada con el parqueadero.
- Atributos: tipo de parqueadero y nombre del establecimiento.
- Operaciones: obtener y actualizar información del parqueadero.

### *3. Clase Parqueadero:*

- Atributos: lista de vehículos (privada) y capacidad.
- Operaciones: gestionar información de los vehículos.

### *4. Clase Vehículo:*

- Atributos: placa, tipo, y puesto.
- Operaciones: getters, setters, y conversión a string.

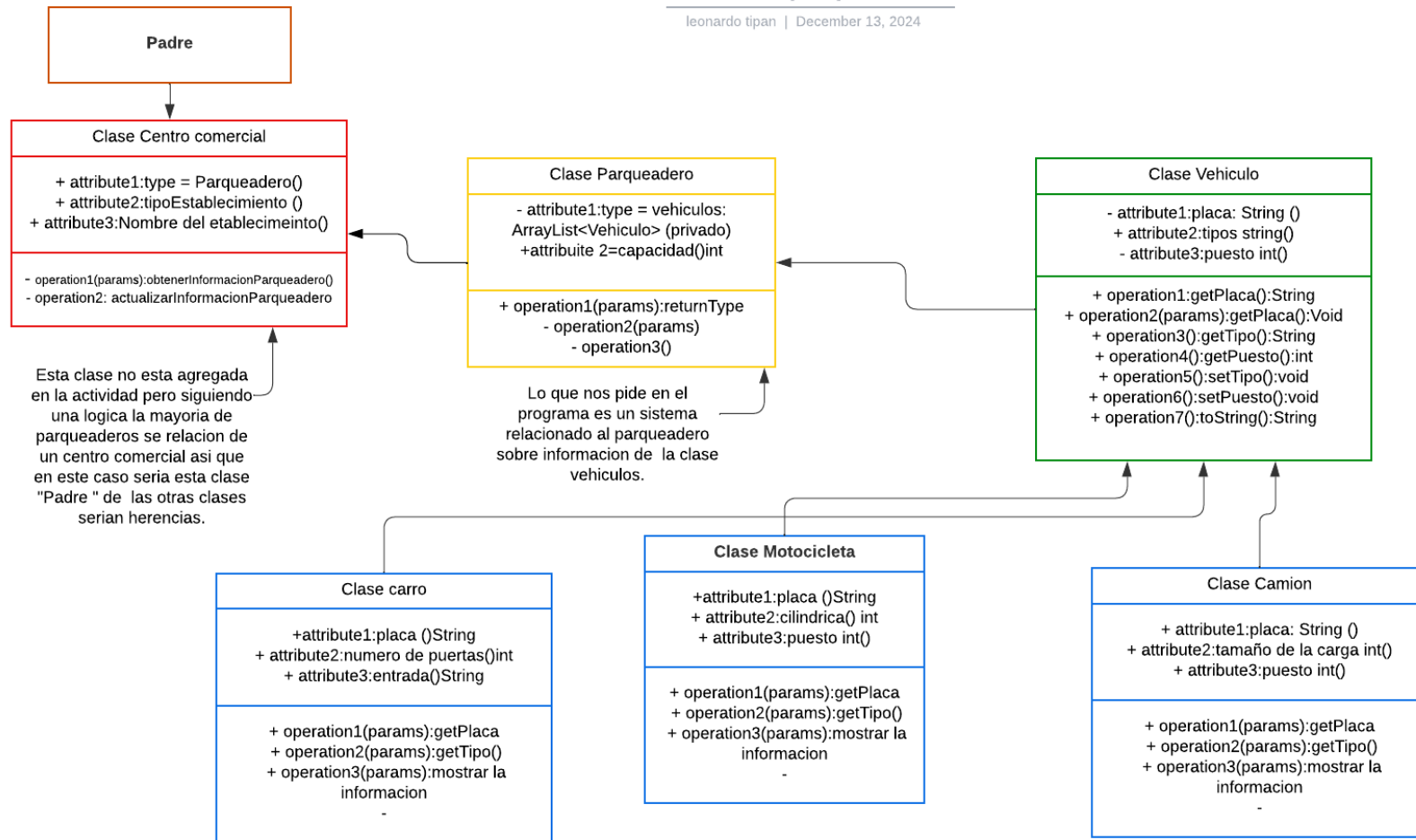
### *5. Subclases de Vehículo:*

- Carro: Tiene atributos como número de puertas y entrada.
- Motocicleta: Atributo específico: cilindraje.
- Camión: Atributo específico: tamaño de la carga.
- Cada una incluye métodos para obtener información específica.


El diseño refleja herencia (Vehículo y sus subclases) y composición (Parqueadero contiene vehículos).

## Clase de parqueadero

leonardo tipan | December 13, 2024





	<b>UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE</b>	<b>DEPARTAMENTO DE COMPUTACION</b>	
		<b>CÓDIGO:</b>	ESPE- DCEX-v1-2023- 011

## CODIGOS UTILIZADOS

### CLASE MAIN


```

Main.java : Vehiculo.java : Parqueadero.java :
1 import java.util.Scanner;
2 import java.util.InputMismatchException;
3
4 public class Main {
5     public static void main(String[] args) {
6         Parqueadero parqueadero = new Parqueadero();
7         Scanner scanner = new Scanner(System.in);
8         int opcion = -1;
9
10        do {
11            System.out.println("\n--- Menú del Parqueadero ---");
12            System.out.println("1. Registrar vehículo");
13            System.out.println("2. Actualizar vehículo");
14            System.out.println("3. Consulta vehículos");
15            System.out.println("4. Salir");
16            System.out.print("Ingrese una opción: ");
17
18            try {
19                opcion = scanner.nextInt();
20                scanner.nextLine();
21            } catch (InputMismatchException e) {
22                System.out.println("\nOpción no válida, debe ser un número.");
23                scanner.nextLine();
24                continue;
25            }
26
27            switch (opcion) {
28                case 1:
29                    System.out.print("\nIngrese la placa del vehículo: ");
30                    String placa = scanner.nextLine();
31                    System.out.print("Ingrese el tipo de vehículo (Carro, Moto, Camión, etc.): ");
32                    String tipo = scanner.nextLine();

```



```
33     parqueadero.registrarVehiculo(placa, tipo);
34     break;
35
36     case 2:
37         System.out.print("\nIngrese el puesto del vehículo a actualizar: ");
38         try {
39             int puestoVehiculo = scanner.nextInt();
40             scanner.nextLine();
41             System.out.print("Ingrese la nueva placa del vehículo: ");
42             String nuevaPlaca = scanner.nextLine();
43             System.out.print("Ingrese el nuevo tipo de vehículo: ");
44             String nuevoTipo = scanner.nextLine();
45             parqueadero.actualizarVehiculo(puestoVehiculo, nuevaPlaca, nuevoTipo);
46         } catch (InputMismatchException e) {
47             System.out.println("\nError: El puesto debe ser un número.");
48             scanner.nextLine();
49         }
50         break;
51
52     case 3:
53         parqueadero.mostrarVehiculos();
54         break;
55
56     case 4:
57         System.out.println("\nCerrando programa...");
58         break;
59
60     default:
61         System.out.println("\nOpción no válida, solo permitido los valores (1,2,3,4)");
62     }
63 } while (opcion != 4);
64
65 scanner.close();
66 }
67 }
68 }
```

	<p align="center"><b>UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE</b></p>	<p align="center"><b>DEPARTAMENTO DE COMPUTACION</b></p>	
		<p align="center"><b>CÓDIGO:</b></p>	<p align="center">ESPE- DCEX-v1-2023- 011</p>

## VEHÍCULO

```

Main.java : Vehiculo.java : Parqueadero.java :
1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 class Vehiculo {
5     private static int contadorPuesto = 1;
6     private final int puesto;
7     private String placa;
8     private String tipo;
9
10    public Vehiculo(String placa, String tipo) {
11        this.puesto = contadorPuesto++;
12        this.placa = placa;
13        this.tipo = tipo;
14    }
15
16    public int getPuesto() {
17        return puesto;
18    }
19
20    public String getPlaca() {
21        return placa;
22    }
23
24    public void setPlaca(String placa) {
25        this.placa = placa;
26    }
27
28    public String getTipo() {
29        return tipo;
30    }
31
32    public void setTipo(String tipo) {
33        this.tipo = tipo;
34    }
35
36    @Override
37    public String toString() {
38        return "Puesto: " + puesto + ", Placa: " + placa + ", Tipo: " + tipo;
39    }
40 }

```

## PARQUEADERO

```
Main.java : Vehiculo.java : Parqueadero.java :
1 import java.util.ArrayList;
2
3 class Parqueadero {
4     private ArrayList<Vehiculo> vehiculos;
5
6     public Parqueadero() {
7         this.vehiculos = new ArrayList<>();
8     }
9
10    public void registrarVehiculo(String placa, String tipo) {
11        Vehiculo vehiculo = new Vehiculo(placa, tipo);
12        vehiculos.add(vehiculo);
13        System.out.println("Vehículo registrado: " + vehiculo);
14    }
15
16    public void actualizarVehiculo(int puestoVehiculo, String nuevaPlaca, String nuevoTipo) {
17        Vehiculo vehiculo = buscarVehiculo(puestoVehiculo);
18        if (vehiculo != null) {
19            vehiculo.setPlaca(nuevaPlaca);
20            vehiculo.setTipo(nuevoTipo);
21            System.out.println("Vehículo actualizado: " + vehiculo);
22        } else {
23            System.out.println("Vehículo no encontrado.");
24        }
25    }
26
27    public void mostrarVehiculos() {
28        if (vehiculos.isEmpty()) {
29            System.out.println("No hay vehículos en el parqueadero.");
30        } else {
31            System.out.println("\n--- Vehículos en el parqueadero ---");
32            for (Vehiculo vehiculo : vehiculos) {
33                System.out.println(vehiculo);
34            }
35        }
36    }
37
38    private Vehiculo buscarVehiculo(int puestoVehiculo) {
39        for (Vehiculo vehiculo : vehiculos) {
40            if (vehiculo.getPuesto() == puestoVehiculo) {
41                return vehiculo;
42            }
43        }
44        return null;
45    }
46 }
47
```