

**UNIVERSIDAD DE LAS FUERZAS ARMADAS (ESPE)**

**CARRERAS TECNOLOGIA DE LA INFORMACIÓN**

**ACTIVIDAD AUTÓNOMO N.º 1 PRIMER PARCIAL**

**PROGRAMACION ORIENTADA A OBJETOS**

**“TEMA: SISTEMA DE GESTION DE PARKING**

**AUTOR:**

Celi Córdova Robert Miguel  
Guamán Ortega Cristina Alexandra  
Jiménez Quezada Pablo Andrés  
Villasis Sango William Oswaldo  
Andrés Alejandro Villacis Villarroel

**PARALELO:**

NRC-1323

**DOCENTE:**

Luis Enrique Jaramillo

**PERÍODO:**

Octubre – Marzo 2024

**FECHA ENTREGA:**

13 de diciembre del 2024

**SANGOLQUI – ECUADOR**

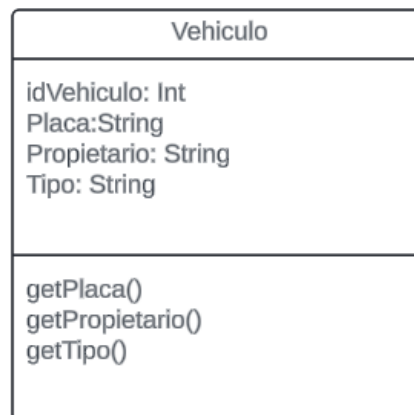
## Introducción

El presente proyecto consiste en el desarrollo de un sistema básico para gestionar un parqueadero utilizando los principios de Programación Orientada a Objetos (POO). Este sistema permite registrar, consultar y actualizar los datos de los vehículos que ingresan al parqueadero. A continuación, se presenta el diagrama de clases UML del sistema. (López, 2023)

- **Clase Vehículo:**

Representa los vehículos registrados en el parqueadero, adicional podremos obtener información de la placa, propietario y tipo de vehículo.

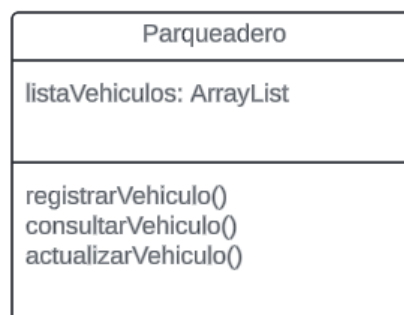
- Atributos: placa, marca, color.
- Métodos: getPlaca(), getPropietario(), getTipo().



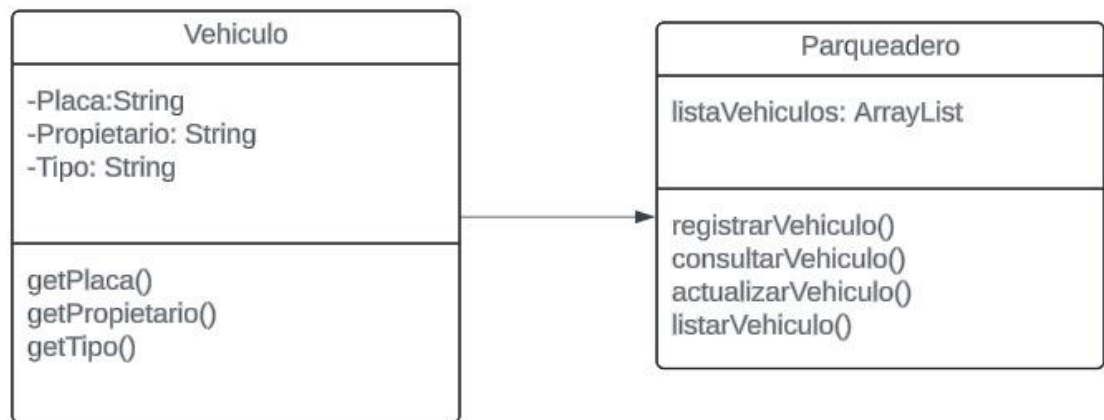
- **Clase Parqueadero:**

Gestiona el registro, consulta y actualización de vehículos, para ello usamos un array.

- Atributos: listaVehiculos (ArrayList).
- Métodos: registrarVehiculo(), consultarVehiculo(), actualizarVehiculo().



## Diagrama UML



## Codificación:

### Creación de clases.

En la creación de clases tanto de Vehículo y Parqueadero, debemos declarar los atributos, constructores y usar Getter & Setter:

### Clase Vehículo:

Atributos:

```
public class Vehiculo {
    //atributos
    //int idVehiculo;
    String placa;
    String propietario;
    String tipo;
    //Time horaEntrada;
    //Time horaSalida;
```

### Constructor

Es un método especial que se llama automáticamente cuando se crea una instancia u objeto de una clase. establece el estado inicial del sistema de gestión del parking. (Lara, 2022)

```
//Constructor:  
public Vehiculo(String placa, String propietario, String tipo) {  
    //this.idVehiculo = idVehiculo;  
    this.placa = placa;  
    this.propietario = propietario;  
    this.tipo = tipo;  
}
```

## Getters & Setters

Los getters siempre nos retornará el valor del atributo sin necesidad de pasar ningún parámetro. Mientras que el método setters siempre nos pedirá algún valor como parámetro para guardarlo al atributo de la clase y éste nunca deberá retornar algún valor.

```
//Getter y setter va antes de los métodos  
  
public String getPlaca() {  
    return placa;  
}  
  
public void setPlaca(String placa) {  
    this.placa = placa;  
}  
  
public String getPropietario() {  
    return propietario;  
}  
  
public void setPropietario(String propietario) {  
    this.propietario = propietario;  
}  
  
public String getTipo() {  
    return tipo;  
}  
  
public void setTipo(String tipo) {  
    this.tipo = tipo;  
}
```

## Clase Parquadero:

Para la clase Parquadero, tenemos la creación de array lo que permite guardar un listado de los vehículos y adicional en esta clase, tenemos la creación de métodos para manipular y leer la información almacenada en los array.

## Creación de Array

Es una estructura de datos que nos permite almacenar una ristra de datos de un mismo tipo.

```
public class Parqueadero {  
  
    private ArrayList<Vehiculo> listaVehiculos;  
  
    public Parqueadero() {  
        listaVehiculos = new ArrayList<>();  
    }  
}
```

## Métodos

### Registrar vehículo

Capturar información y al momento en que ingresa esta acción debe verificar que hay espacios disponibles antes de completar el registro.

```
public void registrarVehiculo(Vehiculo vehiculo) {  
    listaVehiculos.add(vehiculo);  
    System.out.println("Vehículo registrado con éxito.");  
}
```

### Consultar vehículo

Esta funcionalidad puede verificar si el vehículo está actualmente estacionado y proporcionar detalles como el tiempo transcurrido o el lugar que ocupa.

```
public Vehiculo consultarVehiculo(String placa) {  
    for (Vehiculo v : listaVehiculos) {  
        if (v.getPlaca().equalsIgnoreCase(placa)) {  
            return v;  
        }  
    }  
    return null;  
}
```

## Listar los vehículos registrados en el array

Necesitamos recorrer la estructura de datos donde se almacenan los vehículos y mostrar su información.

```
public void listarVehiculos() {  
    if (listaVehiculos.isEmpty()) {  
        System.out.println("No hay vehiculos registrados.");  
    } else {  
        System.out.println("\n=== Lista de Vehiculos Registrados ===");  
        for (Vehiculo v : listaVehiculos) {  
            System.out.println("Placa: " + v.getPlaca() + ", Propietario: " + v.getPropietario() + ", Tipo Vehiculo: " + v.getTipo());  
        }  
    }  
}
```

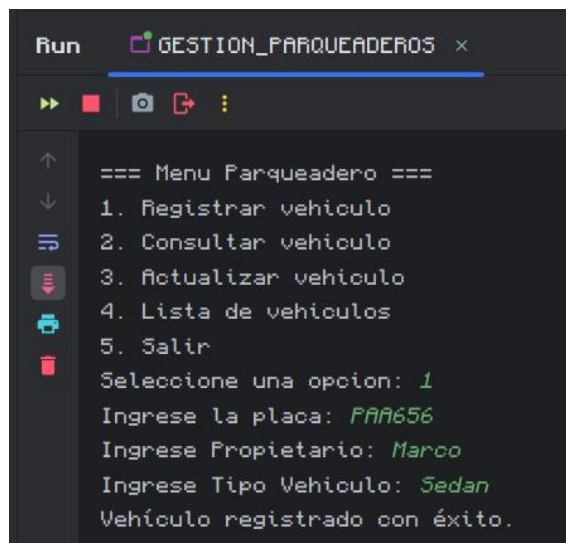
## Actualizar información de vehículo

Se puede implementar un método que permita buscar el vehículo por su placa y modificar sus datos.

```
public void actualizarVehiculo(String placa, String propietario, String tipo) {  
    Vehiculo vehiculo = consultarVehiculo(placa);  
    if (vehiculo != null) {  
        vehiculo.setPlaca(placa);  
        vehiculo.setPropietario(propietario);  
        vehiculo.setTipo(tipo);  
        System.out.println("Vehículo actualizado con éxito.");  
    } else {  
        System.out.println("Vehículo no encontrado.");  
    }  
}
```

## Ejecución del programa

### Opción 1:



```
Run  GESTION_PARQUEADEROS x  
=== Menu Parqueadero ===  
1. Registrar vehiculo  
2. Consultar vehiculo  
3. Actualizar vehiculo  
4. Lista de vehiculos  
5. Salir  
Seleccione una opcion: 1  
Ingrese la placa: PAA656  
Ingrese Propietario: Marco  
Ingrese Tipo Vehiculo: Sedan  
Vehículo registrado con éxito.
```

### Opción 2:

```
Run  GESTION_PARQUEADEROS x
>> ■ 📷 🔗 ⋮
↑
↓
≡
⏮
🖨
🗑
=== Menu Parqueadero ===
1. Registrar vehiculo
2. Consultar vehiculo
3. Actualizar vehiculo
4. Lista de vehiculos
5. Salir
Seleccione una opcion: 2
Ingrese la placa del vehículo a consultar: PAA656

INFORMACIÓN ENCONTRADA:

Placa: PAA656
Propietario: Marco
Tipo: Sedan
```

### Opción 3:

```
Run  GESTION_PARQUEADEROS x
>> ■ 📷 🔗 ⋮
↑
↓
≡
⏮
🖨
🗑
=== Menu Parqueadero ===
1. Registrar vehiculo
2. Consultar vehiculo
3. Actualizar vehiculo
4. Lista de vehiculos
5. Salir
Seleccione una opcion: 3
Ingrese la placa del vehículo a actualizar: PAA656
Ingrese nuevo propietario: Carlitos
Ingrese nuevo tipo: Convertible
Vehículo actualizado con éxito.
```

#### Opcion 4:

```
Run  GESTION_PARQUEADEROS x
>> ■ 📷 📄 ⋮
↑
↓
≡
⏮
🖨
🗑
=== Menu Parqueadero ===
1. Registrar vehiculo
2. Consultar vehiculo
3. Actualizar vehiculo
4. Lista de vehiculos
5. Salir
Seleccione una opcion: 4

=== Lista de Vehículos Registrados ===
Placa: PAA656, Propietario: Carlitos, Tipo Vehiculo: Convertible
```

#### Opción 5:

```
=== Menu Parqueadero ===
1. Registrar vehiculo
2. Consultar vehiculo
3. Actualizar vehiculo
4. Lista de vehiculos
5. Salir
Seleccione una opcion: 5
Saliendo del sistema...

Process finished with exit code 0
|
```

#### Conclusiones

- El desarrollo de un sistema de gestión para parking resalta la importancia de una estructura modular.
- La implementación del método para actualizar información de los vehículos.



## Recomendaciones

- Desarrollar un sistema de gestión de parking un proyecto que puede evolucionar desde un nivel básico hasta convertirse en una solución profesional.

## References

Lara, J. (8 de Noviembre de 2022). *hubspot*. hubspot:

<https://blog.hubspot.es/website/que-es-constructor-java#:~:text=Para%20qu%C3%A9%20sirve%20un%20constructor,y%20no%20devolver%20ning%C3%BAn%20valor.>

López, J. (3 de noviembre de 2023). *openwebinars*. openwebinars:

<https://openwebinars.net/blog/introduccion-a-poo-en-java-atributos-y-constructores/>