

## **Universidad de las Fuerzas Armadas ESPE**



### **Programación Orientada a Objetos NRC 1322**

#### **Actividad Autónoma N°1**

#### **Sistema de Gestión de Parking**

- Pamela Inés Guapulema Pichamba
- Jesús Ismael Pichucho Montes
- Isaac Sebastian Proaño Panchi
- Karol Elizabeth Pugachi Suarez

## Contenido

I.	Resumen de Resolución .....	3
II.	Objetivo de la actividad .....	7
III.	Descripción de la actividad .....	7
IV.	Diagrama UML .....	8
V.	Código y ejecución .....	9
1.	Clase Vehiculo.java .....	9
2.	Clase Parqueadero.java .....	11
3.	Clase Registro.java .....	14
4.	Clase Main //Ejecución dentro de los Anexos .....	15
VI.	Anexos .....	17
VII.	Conclusión .....	18
VIII.	Referencias.....	18

## **I. Resumen de Resolución**

### **Diagrama de clases**

El proyecto utiliza un diagrama de clases que organiza y define las relaciones entre tres clases principales: Parqueadero, Vehiculo y Registro. Cada clase tiene atributos y métodos específicos para gestionar el funcionamiento del sistema de parqueadero:

#### **1. Parqueadero:**

##### **o Atributos:**

- capacidad: Define el número máximo de vehículos permitidos.
- vehiculos: Lista que almacena los vehículos en el parqueadero.

##### **o Métodos:**

- agregarVehiculo(Vehiculo vehiculo): Añade un vehículo al parqueadero si hay espacio disponible.
- removerVehiculo(String matricula): Remueve un vehículo del parqueadero usando su matrícula.
- listarVehiculos(): Muestra una lista de todos los vehículos en el parqueadero.
- esFull(): Verifica si el parqueadero está lleno.

## 2. Vehículo:

### o Atributos:

- matricula: Identificador único del vehículo.
- tipo: Define si el vehículo es un carro, moto, etc.
- horaEntrada: Registra la hora de entrada del vehículo.

### o Métodos:

- Constructores y métodos para acceder a los atributos (getMatricula, getTipo, getHoraEntrada).
- registrarEntrada(): Establece la hora actual como hora de entrada.

## 3. Registro:

### o Atributos:

- vehiculo: Referencia al vehículo asociado al registro.
- horaSalida: Almacena la hora de salida.

### o Métodos:

- registrarSalida(): Calcula y muestra el tiempo que el vehículo estuvo en el parqueadero.

## **Código implementado**

El programa, desarrollado en Java utilizando NetBeans, implementa estas clases para simular las operaciones del parqueadero.

### **1. Clase principal (Main):**

- o Crea un parqueadero con capacidad de dos vehículos.
- o Añade vehículos utilizando la clase Vehiculo.
- o Lista los vehículos presentes.
- o Remueve un vehículo, y registra su salida calculando el tiempo de permanencia.

### **2. Clase Parqueadero:**

- o Implementa la gestión de capacidad y el control de los vehículos presentes.
- o Muestra mensajes al usuario si el parqueadero está lleno o si un vehículo no es encontrado al intentar removerlo.

### **3. Clase Vehiculo:**

- o Establece los detalles de cada vehículo (matrícula, tipo) y su hora de entrada.
- o Permite recuperar datos del vehículo y registrar su entrada.

### **4. Clase Registro:**

- o Calcula el tiempo que un vehículo permaneció en el parqueadero al registrar su salida.

o Convierte el tiempo transcurrido a horas y minutos para mostrarlo de manera clara.

## **Relaciones**

### **Asociación entre Parqueadero y Vehículo:**

El parqueadero tiene una relación de composición con los vehículos, ya que mantiene una lista (vehículos) que contiene instancias de la clase Vehículo. Esto se indica mediante un rombo relleno en la asociación.

### **Asociación entre Registro y Vehículo:**

Existe una asociación simple entre la clase Registro y Vehículo. Un registro necesita un vehículo asociado para calcular el tiempo de permanencia, pero no depende de la existencia del vehículo fuera del contexto del registro.

### **Cardinalidades:**

En la relación entre Parqueadero y Vehículo, la cardinalidad es de 1 a \* (muchos), ya que un parqueadero puede contener múltiples vehículos.

En la relación entre Registro y Vehículo, la cardinalidad es de 1 a 1, ya que un registro corresponde a un único vehículo.

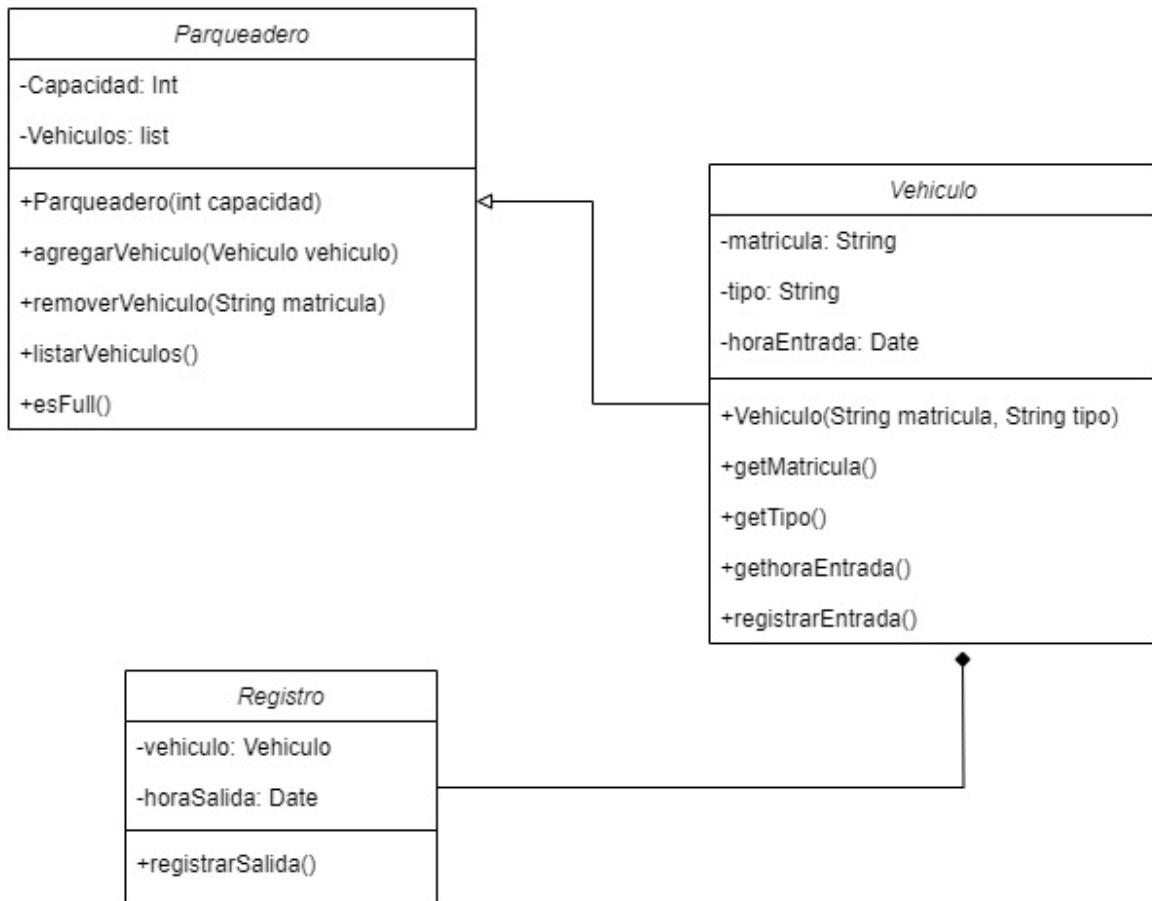
## **II. Objetivo de la actividad**

Desarrollar una simulación en Java que permita realizar las funciones de registrar, consultar y actualizar la información de los vehículos mediante el uso de una interfaz gráfica y una base de datos. Este sistema estará diseñado para gestionar un parqueadero utilizando los principios de Programación Orientada a Objetos (POO).

## **III. Descripción de la actividad**

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.

#### IV. Diagrama UML





## **V. Código y ejecución**

### **1. Clase Vehiculo.java**

```
package com.mycompany.parquadero;

import java.util.Date;

public class Vehiculo {

    private String matricula;

    private String tipo;

    private Date horaEntrada;

    public Vehiculo(String matricula, String tipo) {

        this.matricula = matricula;

        this.tipo = tipo;

        this.horaEntrada = new Date();

    }

    public String getMatricula() {

        return matricula;

    }

    public String getTipo() {

        return tipo;

    }

}
```

```
}
```

```
public Date getHoraEntrada() {
```

```
    return horaEntrada;
```

```
}
```

```
public void registrarEntrada() {
```

```
    this.horaEntrada = new Date();
```

```
}
```

```
}
```

## 2. Clase Parquadero.java

```
package com.mycompany.parquadero;

import java.util.ArrayList;

import java.util.List;

/**
 *
 * @author Admin
 */

public class Parquadero {

    private int capacidad;

    private List<Vehiculo> vehiculos;


    public Parquadero(int capacidad) {

        this.capacidad = capacidad;

        this.vehiculos = new ArrayList<>();

    }

    public void agregarVehiculo(Vehiculo vehiculo) {

        if (!esFull()) {

            vehiculos.add(vehiculo);

            vehiculo.registrarEntrada();

        }

    }

}
```

```
        System.out.println("Vehículo " + vehiculo.getMatricula() + " agregado al  
parqueadero.");
```

```
    } else {
```

```
        System.out.println("El parqueadero está lleno.");
```

```
    }
```

```
}
```

```
public Vehiculo removerVehiculo(String matricula) {
```

```
    for (Vehiculo vehiculo : vehiculos) {
```

```
        if (vehiculo.getMatricula().equals(matricula)) {
```

```
            vehiculos.remove(vehiculo);
```

```
            System.out.println("Vehículo " + matricula + " removido del  
parqueadero.");
```

```
            return vehiculo;
```

```
        }
```

```
    }
```

```
    System.out.println("Vehículo no encontrado.");
```

```
    return null;
```

```
}
```

```
public void listarVehiculos() {
```

```
    if (vehiculos.isEmpty()) {
```

```
        System.out.println("No hay vehículos en el parqueadero.");
```

```
    } else {
```

```
        for (Vehiculo vehiculo : vehiculos) {
```

```
        System.out.println("Vehículo: " + vehiculo.getMatricula() + ", Tipo: " +  
vehiculo.getTipo() + ", Hora de entrada: " + vehiculo.getHoraEntrada());  
    }  
}  
}  
public boolean esFull() {  
    return vehiculos.size() >= capacidad;  
}  
}
```

### 3. Clase Registro.java

```
package com.mycompany.parquadero;

import java.util.Date;

public class Registro {

    private Vehiculo vehiculo;

    private Date horaSalida;

    public Registro(Vehiculo vehiculo) {

        this.vehiculo = vehiculo;

    }

}
```

#### 4. Clase Main //Ejecución dentro de los Anexos

```
package com.mycompany.parqueadero;

public class Main {

    public static void main(String[] args) {

        Parqueadero parqueadero = new Parqueadero(2);

        Vehiculo vehiculo1 = new Vehiculo("ABC123", "carro");

        Vehiculo vehiculo2 = new Vehiculo("XYZ789", "moto");

        parqueadero.agregarVehiculo(vehiculo1);

        parqueadero.agregarVehiculo(vehiculo2);

        parqueadero.listarVehiculos();

        // Remover un vehículo

        Vehiculo vehiculoRemovido = parqueadero.removeVehiculo("ABC123");

        // Registrar salida

        if (vehiculoRemovido != null) {

            Registro registro = new Registro(vehiculoRemovido);

            registro.registrarSalida();

        }

    }

}
```

}

}



## VI. Anexos




### Ejecución del código

```
--- exec:3.1.0:exec (default-cli) @ Parquadero ---
Veh❖culo ABC123 agregado al parqueadero.
Veh❖culo XYZ789 agregado al parqueadero.
Veh❖culo: ABC123, Tipo: carro, Hora de entrada: Fri Dec 13 16:04:16 GMT-05:00 2024
Veh❖culo: XYZ789, Tipo: moto, Hora de entrada: Fri Dec 13 16:04:16 GMT-05:00 2024
Veh❖culo ABC123 removido del parqueadero.
Veh❖culo ABC123 sali❖. Tiempo de permanencia: 0 horas y 0 minutos.
-----
BUILD SUCCESS
```

## VII. Conclusión

En conclusión, se desarrolló un sistema de gestión para un parqueadero con un software funcional que permite registrar, consultar y actualizar el estado de los vehículos. El proyecto se llevó a cabo aplicando los principios de Programación Orientada a Objetos (POO) y superando diversos obstáculos encontrados durante su desarrollo, como en la creación del diagrama UML y la implementación del código. Finalmente, se logró demostrar la funcionalidad del sistema, cumpliendo con los objetivos planteados y consolidando los conocimientos adquiridos.

## VIII. Referencias

TodoCode. (2022, 4 mayo).  *RELACIONES entre CLASES en JAVA POO*  / *1 a 1 y 1 a N*  
/ *Explicación FÁCIL*  #18 [Vídeo]. YouTube.

<https://www.youtube.com/watch?v=XKk5k9OrAUg>

W3Schools.com. (s. f.-b). [https://www.w3schools.com/java/java\\_booleans.asp](https://www.w3schools.com/java/java_booleans.asp)

Jaramillo, L. (s. f.). *Libro*. <https://luisjaramillom.github.io/POO.io/>