# **Actividad Autónoma N.º 1 Primer Parcial**

# **Sistema de Gestión de Parking**

# Marlon Loya

# Pablo Loya

# Universidad de las fuerzas Armadas ESPE

# 1323: Programación orientada a objetos (POO)

# Ing. Luis Enrique Jaramillo Montaño

# 13 de diciembre de 2024

**INTRODUCCION**

El objetivo de nuestro informe es desarrollar un sistema para gestionar un parqueadero utilizando Programación Orientada a Objetos (POO), una interfaz gráfica y bases de datos. Este sistema debe incluir funcionalidades de registro, consulta y actualización de vehículos.

**DESARROLLO**

Vamos a implementar un programa para gestionar un parqueadero. Este debe contener un menú que puede ubicarse en la clase principal (*main*) o implementarse mediante una clase específica que sea llamada cuando se requiera. El programa permitirá crear usuarios, registrar y gestionar los vehículos asociados a estos usuarios.

Comenzaremos aplicando los conceptos básicos de POO aprendidos, detallando los requerimientos para el modelado de clases y objetos. Definiremos dos clases principales:

* Vehículo: Representa un vehículo registrado en el parqueadero.
* Parqueadero: Administra los vehículos estacionados o registrados.

Estas clases estarán relacionadas mediante una relación de agregación, donde la clase Parqueadero contiene una colección de objetos de tipo Vehículo.

Estructura del programa  
El programa debe iniciar mostrando un menú con las siguientes opciones:

* Registrar un vehículo.
* Consultar vehículos registrados.
* Actualizar el estado de un vehículo (Estacionado/Retirado).
* Salir del programa.

Encapsulamiento  
Los atributos de las clases deben ser privados y accesibles únicamente a través de métodos *getters* y *setters*. Además, implementaremos constructores parametrizados para inicializar los objetos con datos como la placa, marca, modelo y estado del vehículo.

Gestión de errores (excepciones)  
El sistema manejará errores en las siguientes situaciones:

* Intentar registrar un vehículo duplicado (placa existente).
* Intentar actualizar el estado de un vehículo inexistente.

Persistencia de datos  
Los vehículos registrados deben guardarse en un archivo de texto. Al iniciar el programa, los datos deben cargarse automáticamente desde dicho archivo.

Colecciones y arreglos  
Utilizaremos una lista dinámica, como un *ArrayList*, para gestionar la lista de vehículos registrados.

Conceptos aplicados

El programa a desarrollar gestionará un parqueadero y contará con un menú principal, que puede implementarse en la clase principal (*Main*) o mediante una clase específica que se invoque cuando sea necesario. Este sistema permitirá:

* Crear usuarios.
* Registrar vehículos.
* Gestionar vehículos asociados a los usuarios.

Modelamiento de Clases y Objetos  
Aplicaremos los conceptos básicos de Programación Orientada a Objetos (POO) para modelar los requerimientos del sistema. Se definirán dos clases principales:

* Vehículo: Representa un vehículo registrado en el parqueadero.
* Parqueadero: Administra los vehículos estacionados o registrados.

Estas clases estarán relacionadas mediante una relación de agregación, donde el parqueadero contiene una colección de objetos del tipo Vehículo.

Menú Principal  
El programa debe iniciar mostrando un menú con las siguientes opciones:

* Registrar un vehículo.
* Consultar vehículos registrados.
* Actualizar el estado de un vehículo (Estacionado/Retirado).
* Salir del programa.

Encapsulamiento  
Los atributos de las clases serán privados y accesibles únicamente mediante métodos *getters* y *setters*. Además, implementaremos constructores parametrizados para inicializar objetos con datos como la placa, marca, modelo y estado del vehículo.

Gestión de Errores (Excepciones)  
El sistema manejará las siguientes situaciones mediante excepciones:

* Intentar registrar un vehículo duplicado (placa existente).
* Intentar actualizar el estado de un vehículo inexistente.

Persistencia de Datos  
Los vehículos registrados se almacenarán en un archivo de texto. Al iniciar el programa, los datos se cargarán automáticamente desde dicho archivo, permitiendo mantener la información entre ejecuciones.

Colecciones y Arreglos  
Se utilizará una lista dinámica, como un *ArrayList*, para gestionar la lista de vehículos registrados.

Relaciones entre Clases  
El parqueadero tiene una relación de agregación con la clase Vehículo, administrando una lista de vehículos estacionados o registrados. Si se desea gestionar propietarios, puede agregarse una clase Usuario relacionada de manera similar.

Estructura General del Programa  
La clase *Main* será responsable de orquestar la lógica del sistema mediante un menú interactivo que ofrezca las siguientes funcionalidades:

* Registrar vehículos: Solicitar al usuario los datos necesarios (placa, marca, modelo y estado).
* Consultar vehículos registrados: Mostrar todos los vehículos almacenados en el sistema.
* Actualizar el estado de un vehículo: Cambiar entre "Estacionado" y "Retirado".
* Salir del programa.

Lectura y Escritura por Consola  
El programa interactuará con el usuario mediante mensajes en la consola, para:

* Mostrar las opciones del menú.
* Solicitar datos al usuario.
* Informar sobre el estado de las operaciones realizadas (registro, consulta, actualización o errores).

Excepciones Adicionales  
Aunque inicialmente no se implementan todas las posibles excepciones, el sistema podrá ampliarse para manejar:

* Registros duplicados de vehículos.
* Intentos de modificar vehículos inexistentes.

Persistencia Extendida  
Aunque el programa base gestiona los datos en memoria mediante un *ArrayList*, puede extenderse para utilizar una base de datos o estructuras de archivo más avanzadas.

**Atributos y Métodos de las Clases**

**Clase Vehículo**

Atributos:

int id: Identificador único del vehículo.

String placa: Placa del vehículo.

String marca: Marca del vehículo.

String modelo: Modelo del vehículo.

boolean estado: Estado (true para Estacionado, false para Retirado).

Métodos:

Getters:

getId()

getPlaca()

getMarca()

getModelo()

isEstado()

**Setters**:

setEstado(boolean estado)

**Clase Parqueadero**

Atributos:

ArrayList<Vehículo> vehículos: Lista de vehículos registrados en el parqueadero.

Métodos:

registrarVehículo(String placa, String marca, String modelo, boolean estado): Agrega un nuevo vehículo al sistema.

consultarVehículos(): Lista todos los vehículos registrados.

actualizarEstado(int id, boolean nuevoEstado): Cambia el estado de un vehículo dado su ID.

cargarDatos(): Lee la información de los vehículos desde un archivo de texto al iniciar.

guardarDatos(): Escribe la información de los vehículos en un archivo de texto al cerrar.

**CONCLUSIONES**

Los Sistemas de Gestión son herramientas clave para mejorar eficiencia, calidad y sostenibilidad en las organizaciones. Su implementación requiere participación activa de todos los niveles y un enfoque de mejora continua. Promueven innovación, responsabilidad y preparan a las organizaciones para enfrentar desafíos con mayor competitividad.

**BIBLIOGRAFÍA**

Lista las fuentes consultadas siguiendo el formato adecuado (APA, MLA, etc.).