Universidad de las Fuerzas Armadas

Informe:

Sistema de Gestión de Parking

Nombres y Apellidos:

Jácome San Lucas Alexander Sebastian

Nathaly Stefanía Cusapaz Quiña

Henry Bolivar Borja Milán

Sheyla Daniela Bernal Correa

Programación Orientada a Obje

Docente:

LUIS ENRIQUE JARAMILLO MONTAÑO

Fecha:

13 de diciembre del 2024

CONTENIDO

OBJETIVOS

Objetivo General

Desarrollar un sistema de gestión para un parqueadero, implementando conceptos de Programación Orientada a Objetos (POO) implementando una interfaz gráfica amigable que permita realizar las funciones de registro, consulta y actualización de vehículos de manera eficiente.

Objetivos Específicos

- Diseñar un modelo UML que representa la estructura y funcionalidades del sistema, asegurando una correcta implementación de POO.
- Implementar una interfaz gráfica que facilite la interacción del usuario con el sistema.
- Aplicar principios de las buenas prácticas de programación como la reutilización de código.

INTRODUCCION

- **Descripción del Proyecto:**

  El presente proyecto describe un diseño e implementación de un sistema de gestión de proyectos utilizando Programación Orientada a Objetos (POO). El objetivo es desarrollar un sistema de gestión para un parqueadero, implementando conceptos de Programación Orientada a Objetos (POO) conuna interfaz gráfica amigable que permita realizar las funciones de registro, consulta y actualización de vehículos de manera eficiente.

- **Antecedentes:**

  - UML (o Lenguaje Unificado de Modelado, por sus siglas en inglés) es un software de lenguaje de modelado que se usa para representar el diseño de un sistema específico. UML tiene varios tipos de diagramas, que muestran diferentes aspectos de las entidades a representar.
  - Diagrama de clases: Los diagramas de clase son los más utilizados en UML y permiten representar un conjunto de clases, interfaces y su relación entre ellos. Los objetos son entidades que tienen características que los diferencian de otros y realizan diferentes acciones como borrador, lápiz, mesa. En POO, las características son variables a las que se conoce como atributos y las acciones que realiza un objeto se le conoce como métodos.
  - La serialización de objetos es el proceso mediante el cual un objeto en Java se convierte en una secuencia de bytes y ser

almacenado en un archivo, transmitido por una red o guardado en una base de datos (Oregoom.com, 2024).

- Un parqueadero automatizado tiene un sistema que permite el control de los vehículos que ingresan a la zona de parqueo, permitiendo un control sobre la ubicación y tiempo de parqueo entrada y salida, teniendo en cuenta múltiples sistemas que permiten identificar la placa vehicular (Villa et al, 2017).
- El manejo de excepciones en Java es una característica fundamental que permite gestionar errores durante la ejecución de un programa, evitando que se interrumpa abruptamente. Las excepciones representan condiciones inusuales o errores que pueden ocurrir, como intentos fallidos de abrir un archivo o la entrada de datos inválidos (Jaramillo,2024).
- GUI es una interfaz entre la persona y la máquina. El objetivo es representar el código del backend de un sistema de la forma más clara posible para el usuario para simplificar las tareas diarias. También, se puede utilizar para controlar PC, tabletas y otros dispositivos. Las GUI utilizan elementos gráficos como iconos, menús e imágenes para facilitar el manejo del usuario humano (IONOS, 2021).

METODOLOGIA

- **Herramientas y Tecnologías Utilizadas:**

  Diseño Orientado a Objetos

  Se definen las principales clases y responsabilidades.

  Clase Parqueadero

  Atributos: vehículo entrada, vehículo salida

  Métodos: parqueadero(), registrar (), mostrar (), buscar ()

  Clase vehículo

  Atributos: contador ID, ID, placa, propietario, tipo de vehículo, hora de entrada, hora de salida, estado.

  Métodos: calcular pago (), toString (), getId(), getPlaca(), setPlaca (), calcularPago (),

- **Procedimiento:**

  El sistema de gestión de parqueadero es un programa desarrollado en JAVA utilizando POO con una interfaz que permite facilitar la gestión eficiente de un parqueadero, permitiendo registrar la entrada y salida de vehículos, así como consultar y actualizar la información de los mismos.

  1.- Registrar vehículo.- se guarda la información como placa, propietario, tipo de vehículo, hora de entrada.

  2.- Registrar salida.- a qué hora sale del parqueadero y calcular el pago correspondiente.

  3.- Mostrar vehículo dentro.- proporcionar una lista de todos los vehículos que se encuentran en el parqueadero..

  4.- Salir

DESARROLLO DEL PROYECTO

- **Diseño de algoritmo:**

    - El modelo está representado por las clases Vehículo y Parqueadero, encargadas de gestionar la lógica de negocio y las operaciones relacionadas con los vehículos como el registro y salidas.
    - El usuario ingresa los datos del vehículo en “Registrar Vehículo”
    - El usuario ingresa el ID del vehículo y hace registrar la salida en“Registrar Salida”
    - Busca el vehículo por ID, verifica su estado, calcula el paso basado en el tiempo de permanencia y actualización del vehículo en la base de datos.

- **Codigo fuente:**
  - **Parqueadero**

```
package GestionParqueadero;


import java.util.ArrayList;


public class Parqueadero {
    private ArrayList<Vehiculo> vehiculos;


    public Parqueadero() {
        this.vehiculos = new ArrayList<>();
    }


    public void registrarVehiculo(String placa, String propietario, String
tipoVehiculo, long horaEntrada) {
        Vehiculo vehiculo = new Vehiculo(placa, propietario, tipoVehiculo,
horaEntrada);
        vehiculos.add(vehiculo);
        System.out.println("Vehículo registrado: " + vehiculo);
    }


    public void registrarSalida(int idVehiculo, long horaSalida) {
        Vehiculo vehiculo = buscarVehiculo(idVehiculo);
        if (vehiculo != null && vehiculo.isEstado()) {
            vehiculo.setHoraSalida(horaSalida);
            vehiculo.setEstado(false);
            double pago = vehiculo.calcularPago();
```

```
            System.out.println("El vehículo con ID " + idVehiculo + " ha
salido. Total a pagar: $" + pago);
        } else {
            System.out.println("Vehículo no encontrado o ya salió.");
        }
    }

    public void mostrarVehiculosDentro() {
        System.out.println("\n--- Vehículos dentro del parqueadero ---");
        for (Vehiculo vehiculo : vehiculos) {
            if (vehiculo.isEstado()) {
                System.out.println(vehiculo);
            }
        }
    }

    private Vehiculo buscarVehiculo(int id) {
        for (Vehiculo vehiculo : vehiculos) {
            if (vehiculo.getId() == id) {
                return vehiculo;
            }
        }
        return null;
    }
}
```

- **Vehiculo**

```
package GestionParqueadero;
import java.util.ArrayList;

public class Parqueadero {
   private ArrayList<Vehiculo> vehiculos;

   public Parqueadero() {
      this.vehiculos = new ArrayList<>();
   }

   public void registrarVehiculo(String placa, String propietario, String tipoVehiculo,
long horaEntrada) {
      Vehiculo vehiculo = new Vehiculo(placa, propietario, tipoVehiculo, horaEntrada);
      vehiculos.add(vehiculo);
      System.out.println("Vehículo registrado: " + vehiculo);
   }

   public void registrarSalida(int idVehiculo, long horaSalida) {
      Vehiculo vehiculo = buscarVehiculo(idVehiculo);
      if (vehiculo != null && vehiculo.isEstado()) {
         vehiculo.setHoraSalida(horaSalida);
         vehiculo.setEstado(false);
         double pago = vehiculo.calcularPago();
         System.out.println("El vehículo con ID " + idVehiculo + " ha salido. Total a
pagar: $" + pago);
      } else {
         System.out.println("Vehículo no encontrado o ya salió.");
      }
   }

   public void mostrarVehiculosDentro() {
      System.out.println("\n--- Vehículos dentro del parqueadero ---");
      for (Vehiculo vehiculo : vehiculos) {
         if (vehiculo.isEstado()) {
            System.out.println(vehiculo);
         }
      }
   }

   private Vehiculo buscarVehiculo(int id) {
      for (Vehiculo vehiculo : vehiculos) {
         if (vehiculo.getId() == id) {
            return vehiculo;
         }
      }
```

```
        return null;
    }
}
```

- **Main**

```
package GestionParqueadero;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Parqueadero parqueadero = new Parqueadero();
        Scanner scanner = new Scanner(System.in);
        int opcion;

        do {
            System.out.println("\n--- Menú del Parqueadero ---");
            System.out.println("1. Registrar vehículo");
            System.out.println("2. Registrar salida");
            System.out.println("3. Mostrar vehículos dentro");
            System.out.println("4. Salir");
            System.out.print("Ingrese una opción: ");
            opcion = scanner.nextInt();
            scanner.nextLine(); // Limpiar el buffer

            switch (opcion) {
                case 1:
                    System.out.print("Ingrese la placa del vehículo: ");
                    String placa = scanner.nextLine();
                    System.out.print("Ingrese el nombre del propietario: ");
                    String propietario = scanner.nextLine();
                    System.out.print("Ingrese el tipo de vehículo (Carro/Moto): ");
                    String tipoVehiculo = scanner.nextLine();
                    long horaEntrada = System.currentTimeMillis();
                    parqueadero.registrarVehiculo(placa, propietario, tipoVehiculo,
horaEntrada);
                    break;

                case 2:
                    System.out.print("Ingrese el ID del vehículo: ");
                    int idVehiculo = scanner.nextInt();
                    long horaSalida = System.currentTimeMillis();
 parqueadero.registrarSalida(idVehiculo, horaSalida);
                    break;

                case 3:
                    parqueadero.mostrarVehiculosDentro();
                    break;
```

```
            case 4:
                System.out.println("Saliendo del sistema...");
                break;

            default:
                System.out.println("Opción no válida.");
        }
    } while (opcion != 4);

    scanner.close();
  }
}
```

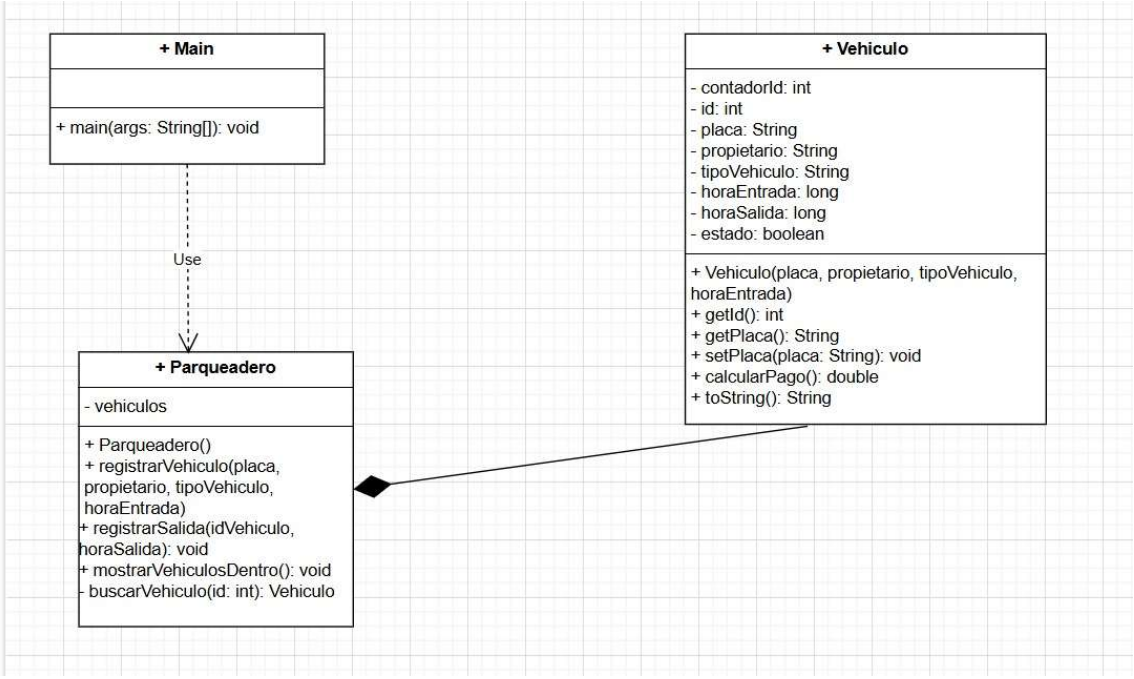
- **Interfaz Grafica**

```
package GestionParqueadero;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class InterfazGrafica extends JFrame {
    private Parqueadero parqueadero;

    private JTextField placaField;
    private JTextField propietarioField;
    private JTextField tipoVehiculoField;
    private JTextField idSalidaField;
    private JTextArea displayArea;

    public InterfazGrafica() {
        parqueadero = new Parqueadero();

        // Configuración de la ventana
        setTitle("Gestión del Parqueadero");
        setSize(600, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout(10, 10));

        // Panel de entrada de datos (registro de vehículo) en la parte superior
        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(4, 2, 10, 10));

        JLabel placaLabel = new JLabel("Placa:");
        placaField = new JTextField();
        JLabel propietarioLabel = new JLabel("Propietario:");
        propietarioField = new JTextField();
        JLabel tipoVehiculoLabel = new JLabel("Tipo Vehículo(Carro o Moto):");
        tipoVehiculoField = new JTextField();

        inputPanel.add(placaLabel);
        inputPanel.add(placaField);
        inputPanel.add(propietarioLabel);
        inputPanel.add(propietarioField);
        inputPanel.add(tipoVehiculoLabel);
        inputPanel.add(tipoVehiculoField);

        // Botón para registrar vehículo
```

```
        JButton registrarButton = new JButton("Registrar Vehículo");
        registrarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String placa = placaField.getText();
                String propietario = propietarioField.getText();
                String tipoVehiculo = tipoVehiculoField.getText();
                long horaEntrada = System.currentTimeMillis();
                parqueadero.registrarVehiculo(placa, propietario, tipoVehiculo, horaEntrada);
                updateDisplay();
                // Mostrar el ID generado para el vehículo registrado
                Vehiculo ultimoVehiculo =
parqueadero.getVehiculos().get(parqueadero.getVehiculos().size() - 1);
                JOptionPane.showMessageDialog(null, "¡Vehículo registrado con éxito! Su ID
es: " + ultimoVehiculo.getId());
            }
        });

        // Panel para mostrar vehículos dentro
        displayArea = new JTextArea();
        displayArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(displayArea);

        // Panel para registrar salida en la parte inferior
        JPanel salidaPanel = new JPanel();
        salidaPanel.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));
        JLabel idSalidaLabel = new JLabel("ID del vehículo (para salida):");
        idSalidaField = new JTextField(10);
        JButton registrarSalidaButton = new JButton("Registrar Salida");
        registrarSalidaButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    int idVehiculo = Integer.parseInt(idSalidaField.getText());
                    long horaSalida = System.currentTimeMillis();
                    parqueadero.registrarSalida(idVehiculo, horaSalida);
                    updateDisplay();
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(null, "Por favor, ingrese un ID válido.");
                }
            }
        });

        salidaPanel.add(idSalidaLabel);
        salidaPanel.add(idSalidaField);
        salidaPanel.add(registrarSalidaButton);

        // Botón para mostrar vehículos dentro
        JPanel mostrarPanel = new JPanel();
        JButton mostrarButton = new JButton("Mostrar Vehículos Dentro");
        mostrarButton.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                updateDisplay();
            }
        });

        mostrarPanel.add(mostrarButton);

        // Panel principal
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new BorderLayout(10, 10));
        mainPanel.add(inputPanel, BorderLayout.NORTH);
        mainPanel.add(registrarButton, BorderLayout.SOUTH);

        // Agregar los componentes a la ventana
        add(mainPanel, BorderLayout.NORTH);
        add(scrollPane, BorderLayout.CENTER);
        add(salidaPanel, BorderLayout.SOUTH);
        add(mostrarPanel, BorderLayout.EAST);

        setVisible(true);
    }

    private void updateDisplay() {
        displayArea.setText(""); // Limpiar el área de texto
        for (Vehiculo vehiculo : parqueadero.getVehiculosDentro()) {
            displayArea.append(vehiculo.toString() + "\n");
        }
    }

    public static void main(String[] args) {
        new InterfazGrafica();
    }
}
```

- **Diagrama UML**

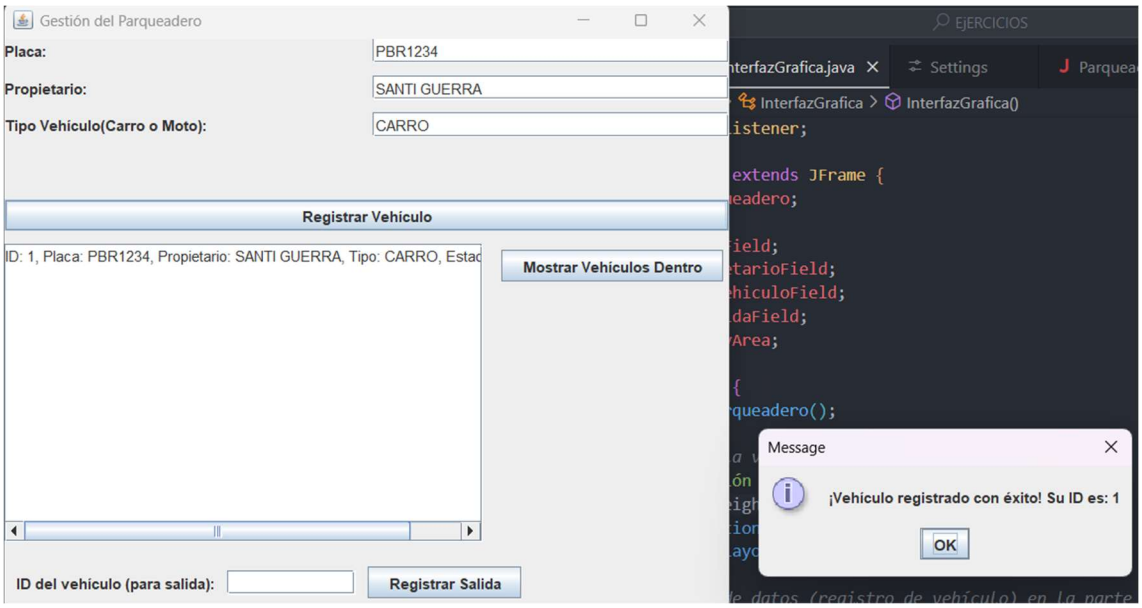


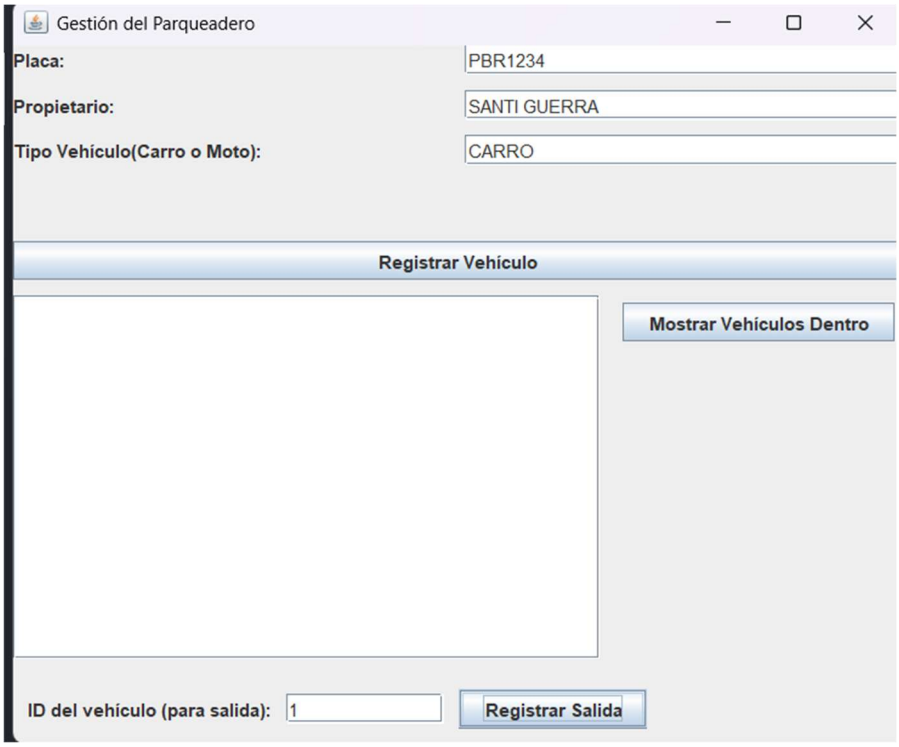
Relación entre Main y Parqueadero, la relación está representada como una dependencia porque la clase Main actúa como el punto de entrada al sistema, utilizando los métodos de la clase Parqueadero para gestionar las operaciones del parqueadero.

Relación entre Parqueadero y Vehiculo: Esta relación se define como una composición porque Parqueadero gestiona una colección de objetos Vehiculo. La composición implica que los vehículos son una parte integral de la existencia del parqueadero; si el objeto Parqueadero deja de existir, también lo hacen los vehículos asociados a él. Esto se refleja mediante un rombo en el extremo de Parqueadero, apuntando hacia Vehículo.

RESULTADOS

- Análisis de resultados



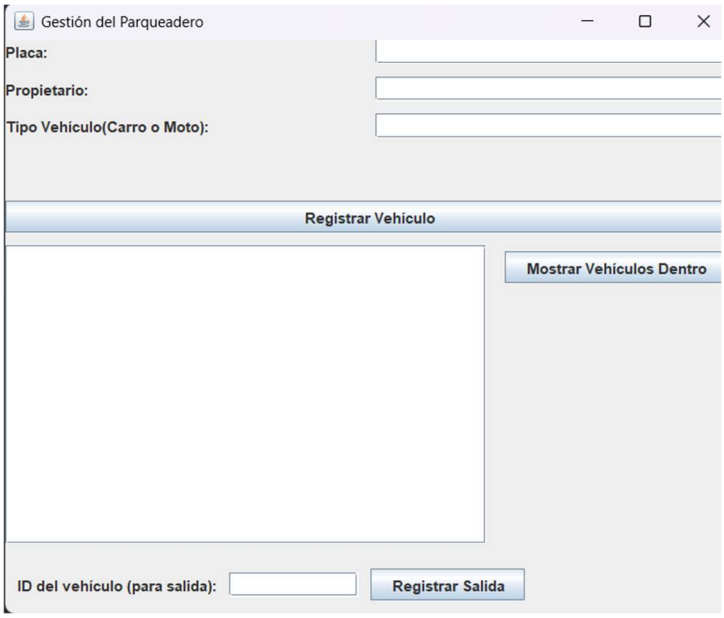


Con el proyecto de gestión de parqueadero, se logró implementar un sistema que permite registrar vehículos al ingreso, calcular pagos

basados en el tiempo de permanencia y tipo de vehículo, de igual forma gestionar la salida de estos

- Pruebas realizadas

  Interfaz



  Entrada:

  El diseño del sistema sigue los principios de Programación Orientada a Objetos (POO), utilizando encapsulación para proteger los datos y proporcionar acceso controlado a través de métodos públicos. Las clases se organizan de forma que cumplen con cada una de sus funcionalidades.

CONCLUCIONES

- El sistema desarrollado cumple con los objetivos planteados, proporcionando una herramienta funcional para la gestión de vehículos en un parqueadero mediante el uso de Programación Orientada a Objetos (POO).
- La interfaz gráfica diseñada asegura una interacción amigable y eficiente con el usuario, mejorando la experiencia de uso.
- El uso de diagramas UML durante la etapa de diseño permitió estructurar correctamente el sistema, disminuyendo errores y mejorando la planificación del proyecto.

- La aplicación de principios como la encapsulación y la reutilización de código permitió desarrollar un sistema modular, fácil de mantener y seguro en el manejo de datos.

RECOMENDACIONES

- Realizar pruebas automatizadas o manuales con diferentes escenarios ayudaría a garantizar el correcto funcionamiento del sistema en todo momento.
- Considero importante seguir mejorando la estructura del sistema para que sea más flexible y pueda adaptarse a ocasiones más complejas o con reglas específicas. También sería útil implementar reportes automatizados que muestren estadísticas o resultados en tiempo real, lo que facilita la toma de decisiones.

BIOGRAFIA

Codigofacilito. (2010). UML y diagrama de clases. https://codigofacilito.com/articulos/uml_diagramas_de_clase

IONOS. (28 enero 2021). ¿Qué es una interfaz gráfica de usuario (GUI)? GUI: qué es una interfaz gráfica de usuario - IONOS MX

Oregon.com. (2024). Serialización de Objetos en Java. https://oregoom.com/java/serializacion-de-objetos/#google_vignette

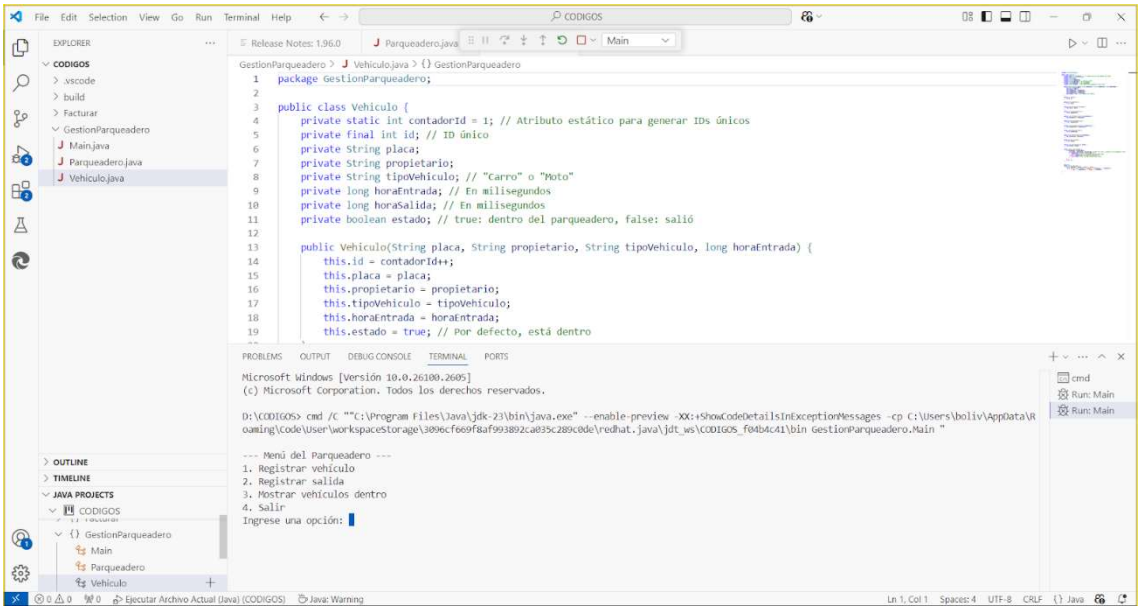
Jaramillo, L. (11 agosto 2024).Programación Orientada a Objetos. https://luisjaramillom.github.io/POO.io/

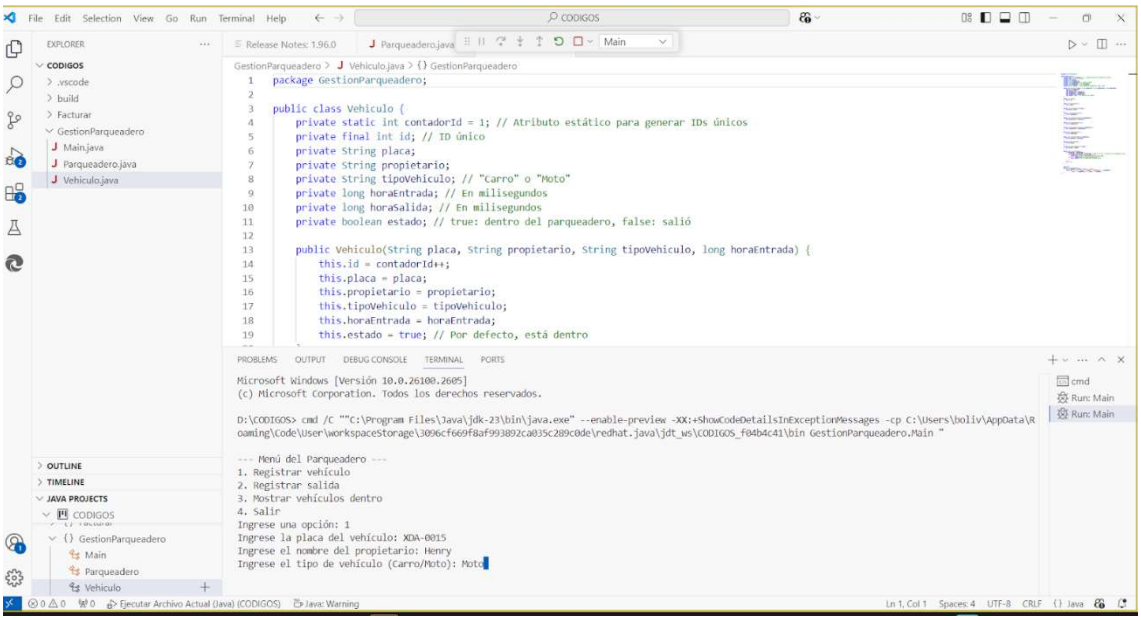
Oracle. (2023). *The Java™ Tutorials*. https://docs.oracle.com/javase/tutorial/

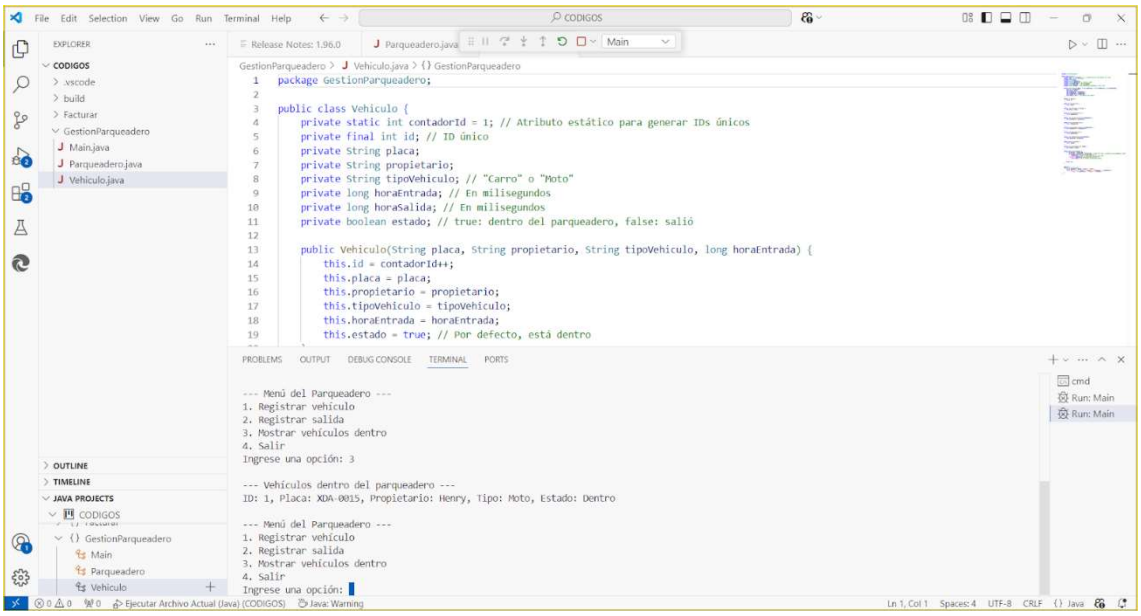
Visual Pradigm. (9 de febrero de 2022). *¿Cuáles Son Los Seis Tipos De Relaciones En Los Diagramas De Clases UML?* https://blog.visual-paradigm.com/es/what-are-the-six-types-of-relationships-in-uml-class-diagrams/

Villa, J. S., Masson Vaca, L. A (2017). Estudio de factibilidad para la implementación de un parqueadero automatizado en el centro de Guayaquil. http://repositorio.ug.edu.ec/handle/redug/22521

ANEXOS

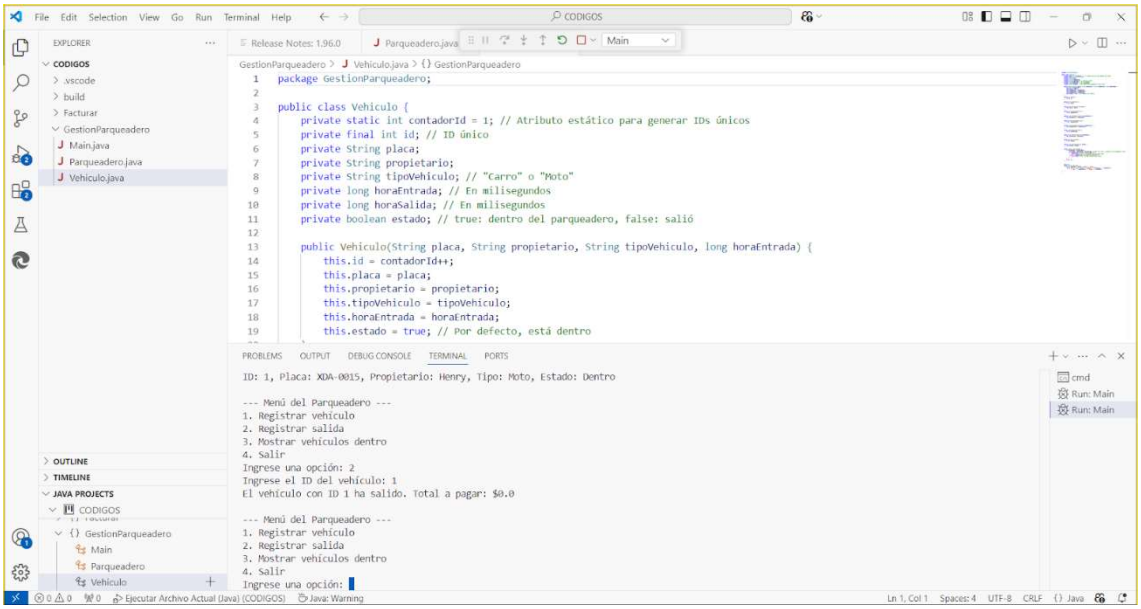

File Edit Selection View Go Run Terminal Help
CODIGOS
EXPLORER
Release Notes: 1.96.0
Parqueadero.java
Main
GestionParqueadero > Vehiculo.java > GestionParqueadero
package GestionParqueadero;
public class Vehiculo {
private static int contadorId = 1; // Atributo estático para generar IDs únicos
private final int id; // ID único
private String placa;
private String propietario;
private String tipoVehiculo; // "Carro" o "Moto"
private long horaEntrada; // En milisegundos
private long horaSalida; // En milisegundos
private boolean estado; // true: dentro del parqueadero, false: salió
public Vehiculo(String placa, String propietario, String tipoVehiculo, long horaEntrada) {
this.id = contadorId++;
this.placa = placa;
this.propietario = propietario;
this.tipoVehiculo = tipoVehiculo;
this.horaEntrada = horaEntrada;
this.estado = true; // Por defecto, está dentro
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Microsoft Windows [Versión 10.0.26100.2605]
(c) Microsoft Corporation. Todos los derechos reservados.
D:\CODIGOS> cmd /C ""C:\Program Files\Java\jdk-23\bin\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\boliv\AppData\Roaming\Code\User\workspaceStorage\3096cf669f8af993892ca035c289c0de\redhat.java\jdt_ws\CODIGOS_f04b4c41\bin GestionParqueadero.Main "
--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción:
OUTLINE
TIMELINE
JAVA PROJECTS
CODIGOS
GestionParqueadero
Main
Parqueadero
Vehiculo
Ejecutar Archivo Actual (Java) (CODIGOS)
Java: Warning
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Java


File Edit Selection View Go Run Terminal Help
CODIGOS
EXPLORER
Release Notes: 1.96.0
Parqueadero.java
Main
GestionParqueadero > Vehiculo.java > GestionParqueadero
package GestionParqueadero;
public class Vehiculo {
private static int contadorId = 1; // Atributo estático para generar IDs únicos
private final int id; // ID único
private String placa;
private String propietario;
private String tipoVehiculo; // "Carro" o "Moto"
private long horaEntrada; // En milisegundos
private long horaSalida; // En milisegundos
private boolean estado; // true: dentro del parqueadero, false: salió
public Vehiculo(String placa, String propietario, String tipoVehiculo, long horaEntrada) {
this.id = contadorId++;
this.placa = placa;
this.propietario = propietario;
this.tipoVehiculo = tipoVehiculo;
this.horaEntrada = horaEntrada;
this.estado = true; // Por defecto, está dentro
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Microsoft Windows [Versión 10.0.26100.2605]
(c) Microsoft Corporation. Todos los derechos reservados.
D:\CODIGOS> cmd /C ""C:\Program Files\Java\jdk-23\bin\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\boliv\AppData\Roaming\Code\User\workspaceStorage\3096cf669f8af993892ca035c289c0de\redhat.java\jdt_ws\CODIGOS_f04b4c41\bin GestionParqueadero.Main "
--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción: 1
Ingrese la placa del vehículo: XDA-0015
Ingrese el nombre del propietario: Henry
Ingrese el tipo de vehículo (Carro/Moto): Moto
OUTLINE
TIMELINE
JAVA PROJECTS
CODIGOS
GestionParqueadero
Main
Parqueadero
Vehiculo
Ejecutar Archivo Actual (Java) (CODIGOS)
Java: Warning
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Java



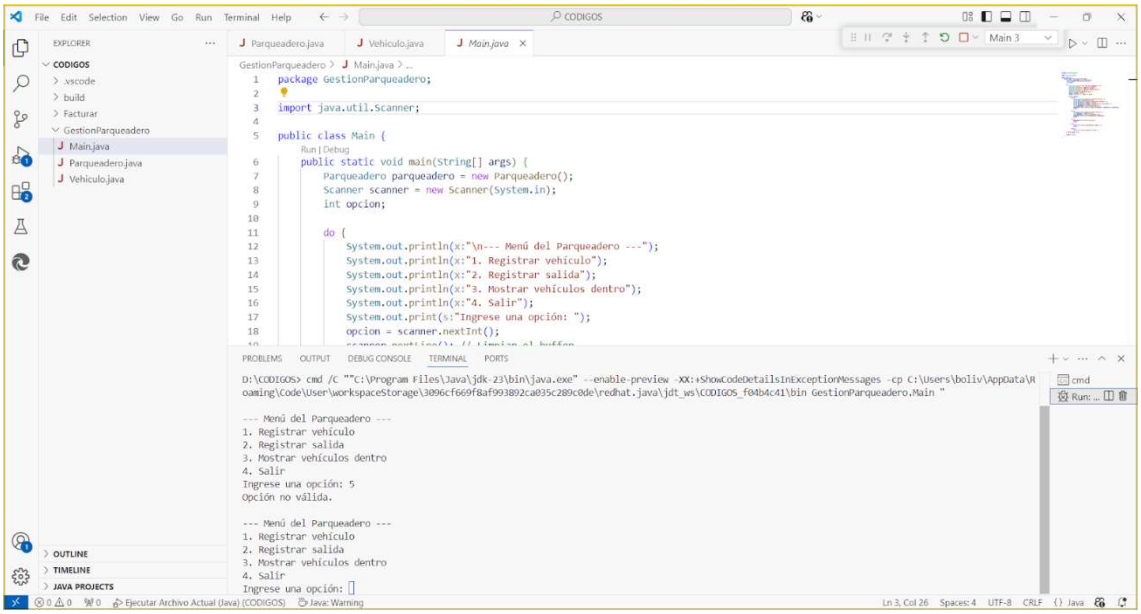
```
package GestionParqueadero;

public class Vehiculo {
    private static int contadorId = 1; // Atributo estático para generar IDs únicos
    private final int id; // ID único
    private String placa;
    private String propietario;
    private String tipoVehiculo; // "Carro" o "Moto"
    private long horaEntrada; // En milisegundos
    private long horaSalida; // En milisegundos
    private boolean estado; // true: dentro del parqueadero, false: salió

    public Vehiculo(String placa, String propietario, String tipoVehiculo, long horaEntrada) {
        this.id = contadorId++;
        this.placa = placa;
        this.propietario = propietario;
        this.tipoVehiculo = tipoVehiculo;
        this.horaEntrada = horaEntrada;
        this.estado = true; // Por defecto, está dentro
```

```
--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción: 3

--- Vehículos dentro del parqueadero ---
ID: 1, Placa: XDA-0015, Propietario: Henry, Tipo: Moto, Estado: Dentro

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción:
```



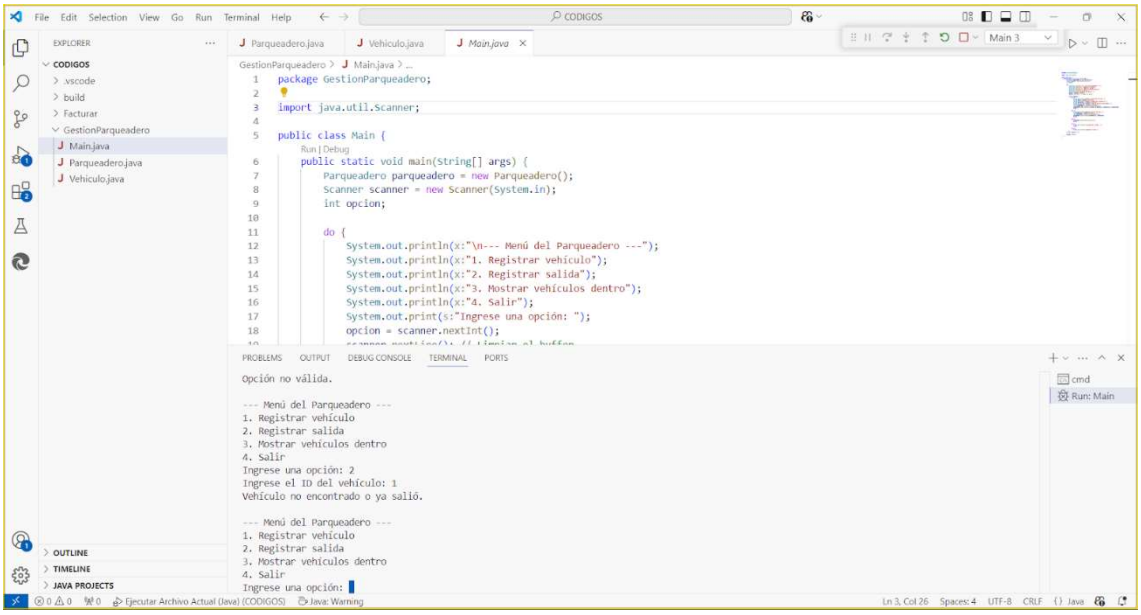
```
package GestionParqueadero;

public class Vehiculo {
    private static int contadorId = 1; // Atributo estático para generar IDs únicos
    private final int id; // ID único
    private String placa;
    private String propietario;
    private String tipoVehiculo; // "Carro" o "Moto"
    private long horaEntrada; // En milisegundos
    private long horaSalida; // En milisegundos
    private boolean estado; // true: dentro del parqueadero, false: salió

    public Vehiculo(String placa, String propietario, String tipoVehiculo, long horaEntrada) {
        this.id = contadorId++;
        this.placa = placa;
        this.propietario = propietario;
        this.tipoVehiculo = tipoVehiculo;
        this.horaEntrada = horaEntrada;
        this.estado = true; // Por defecto, está dentro
```

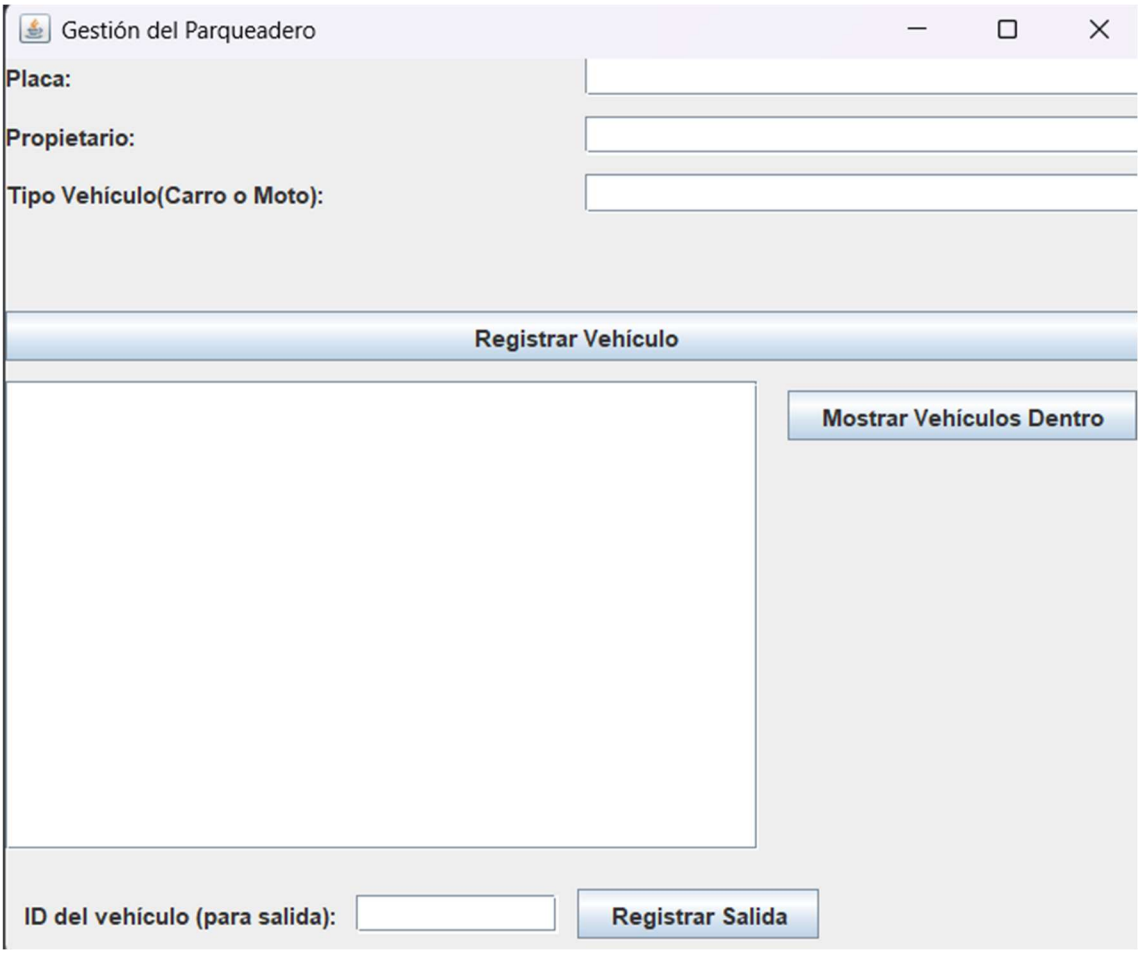
```
ID: 1, Placa: XDA-0015, Propietario: Henry, Tipo: Moto, Estado: Dentro

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción: 2
Ingrese el ID del vehículo: 1
El vehículo con ID 1 ha salido. Total a pagar: $0.0

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción:
```



```
package GestionParqueadero;

import java.util.Scanner;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Parqueadero parqueadero = new Parqueadero();
        Scanner scanner = new Scanner(System.in);
        int opcion;

        do {
            System.out.println(x:"\n--- Menú del Parqueadero ---");
            System.out.println(x:"1. Registrar vehículo");
            System.out.println(x:"2. Registrar salida");
            System.out.println(x:"3. Mostrar vehículos dentro");
            System.out.println(x:"4. Salir");
            System.out.print(s:"Ingrese una opción: ");
            opcion = scanner.nextInt();
```

```
D:\CODIGOS> cmd /C ""C:\Program Files\Java\jdk-23\bin\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\boliv\AppData\Roaming\Code\User\workspaceStorage\3096cf669f8af993892ca035c289c0de\redhat.java\jdt_ws\CODIGOS_f04b4c41\bin GestionParqueadero.Main "

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción: 5
Opción no válida.

--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción:
```


File Edit Selection View Go Run Terminal Help
CODIGOS
EXPLORER
CODIGOS
.vscode
build
Facturar
GestionParqueadero
Main.java
Parqueadero.java
Vehiculo.java
Parqueadero.java
Vehiculo.java
Main.java
GestionParqueadero > Main.java > ...
Run | Debug
package GestionParqueadero;
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
Parqueadero parqueadero = new Parqueadero();
Scanner scanner = new Scanner(System.in);
int opcion;
do {
System.out.println("\n--- Menú del Parqueadero ---");
System.out.println("1. Registrar vehículo");
System.out.println("2. Registrar salida");
System.out.println("3. Mostrar vehículos dentro");
System.out.println("4. Salir");
System.out.print("Ingrese una opción: ");
opcion = scanner.nextInt();
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Opción no válida.
--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción: 2
Ingrese el ID del vehículo: 1
Vehículo no encontrado o ya salió.
--- Menú del Parqueadero ---
1. Registrar vehículo
2. Registrar salida
3. Mostrar vehículos dentro
4. Salir
Ingrese una opción:
cmd
Run: Main
OUTLINE
TIMELINE
JAVA PROJECTS
Ejecutar Archivo Actual (Java) (CODIGOS)
Java: Warning
Ln 3, Col 26 Spaces: 4 UTF-8 CRLF Java


Gestión del Parqueadero
Placa:
Propietario:
Tipo Vehículo(Carro o Moto):
Registrar Vehículo
Mostrar Vehículos Dentro
ID del vehículo (para salida):
Registrar Salida