Gestion de Vehiculos (Carro o Moto)

Integrantes:

Jose Antonio Mogollon

Jaider Alejandro Muñoz Ortega

El proyecto realizado está basado en un programa construido en el lenguaje de programación Java; el cual, tiene como funcionalidad principal, el manejo de un sistema de gestión de vehículos.

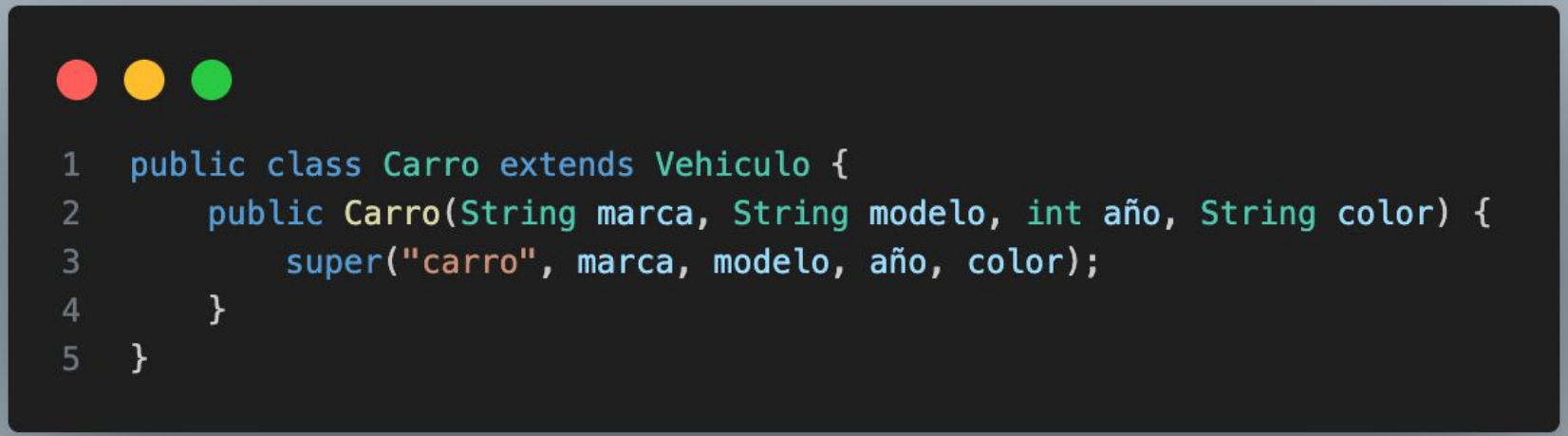
Decisiones de diseño: La estructura del código se basó esencialmente en el uso de las clases indicadas para empezar a realizar el sistema de gestión, el cual usa las excepciones para generar alertas sobre situaciones inesperadas.

El proyecto se dividió en 7 archivos de código:

- Aplicación Gestión Vehículos: Método principal
- Carro: Heredero de la clase vehículo
- Moto: Heredero de la clase vehículo
- Vehículo: Constructor de los atributos de la lista de arreglos
- Mantenimiento: Constructor opcional para definir atributos adicionales
- Gestor Vehículos: Constructor de la lista de arreglos que almacena las cadenas que definen los atributos del vehículo
- Documento: Constructor opcional para definir atributos adicionales.

Funcionalidades: Las funcionalidades principales del código son:

- **Registro de vehículos**: El código permite registrar nuevos vehículos, incluyendo información como tipo de vehículo (carro o moto), marca, modelo, año y color.
- **Gestión de mantenimientos:** El código permite registrar mantenimientos para cada vehículo, incluyendo información como fecha de mantenimiento, tipo de cambio y costo.
- **Visualización de información:** El código permite visualizar información sobre los vehículos y mantenimientos registrados, incluyendo detalles como tipo de vehículo, marca, modelo, año, color, fecha de mantenimiento, tipo de cambio y costo.

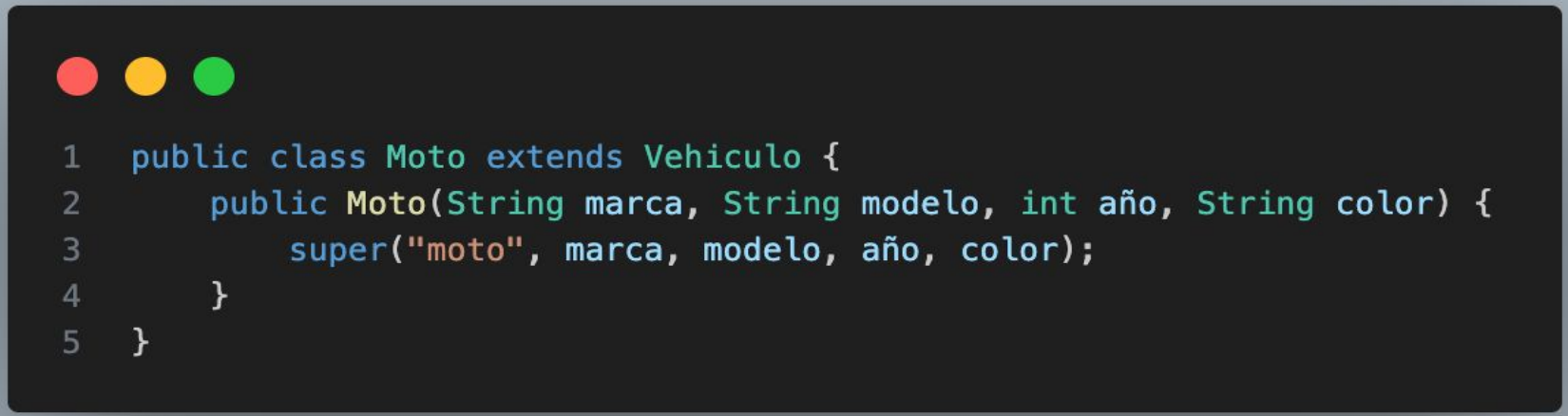


```java
public class Carro extends Vehiculo {
    public Carro(String marca, String modelo, int año, String color) {
        super("carro", marca, modelo, año, color);
    }
}
```

Clase Carro

public class Carro: Esto define una clase llamada Carro que es pública y extends Vehiculo Indica que Carro es una subclase de Vehiculo, lo que significa que hereda las propiedades y métodos de Vehiculo.

public Carro: Este es el constructor de la clase Carro, que se utiliza para crear instancias de esta clase.
super: Esta llamada al constructor de la clase padre (Vehiculo) pasa argumentos a su constructor.



```java
public class Moto extends Vehiculo {
    public Moto(String marca, String modelo, int año, String color) {
        super("moto", marca, modelo, año, color);
    }
}
```

Clase Moto

public class Moto: Esto crea una nueva clase pública llamada Moto y extends Vehiculo Indica que Moto es una subclase de Vehiculo lo que significa que hereda las propiedades y métodos de la clase Vehiculo.

public Moto: Este es el constructor de la clase Moto, que se utiliza para crear instancias de esta clase.

super: Aquí se llama al constructor de la clase padre (Vehiculo) y se le pasan los argumentos. Se envía un valor fijo ("moto") junto con los parámetros marca, modelo, año y color, que se reciben en el constructor de Moto.

```java
import java.util.Date;

public class Mantenimiento {
    private Date fechaMantenimiento;
    private String tipoCambio; // Tipo de cambio realizado
    private double costo;
    private Vehiculo vehiculo;

    public Mantenimiento(Vehiculo vehiculo){
        this.vehiculo = vehiculo;
    }

    public list<Vehiculo> getVehiculos() {
        return list.of(vehiculo);
    }

    public Mantenimiento(Date fechaMantenimiento, String tipoCambio, double costo) {
        this.fechaMantenimiento = fechaMantenimiento;
        this.tipoCambio = tipoCambio;
        this.costo = costo;
    }

    public Date getFechaMantenimiento() {
        return fechaMantenimiento;
    }

    public String getTipoCambio() {
        return tipoCambio;
    }

    public double getCosto() {
        return costo;
    }

    public void registrarMantenimiento() {
        System.out.println("Mantenimiento registrado: " + tipoCambio + " el " + fechaMantenimiento);
    }
}

```

Clase Mantenimiento

El constructor Mantenimiento permite crear un objeto Mantenimiento asignando un Vehiculo a la variable vehiculo. No inicializa los otros atributos.

El método list devuelve una lista que contiene el vehículo asociado a este mantenimiento, utilizando el método of de la clase list,

El método regisroMantenimiento imprime un mensaje en la consola indicando que el mantenimiento ha sido registrado, mostrando el tipo de cambio y la fecha.

```java
import java.util.Date;

public class Documento {
    private String tipo; // "Tecnomecanica" o "Seguro"
    private Date fechaEmision;
    private Date fechaVencimiento;

    public Documento(String tipo, Date fechaEmision, Date fechaVencimiento) {
        this.tipo = tipo;
        this.fechaEmision = fechaEmision;
        this.fechaVencimiento = fechaVencimiento;
    }

    public String getTipo() {
        return tipo;
    }

    public Date getFechaEmision() {
        return fechaEmision;
    }

    public Date getFechaVencimiento() {
        return fechaVencimiento;
    }
}
```

Clase Documento

El constructor de la clase Documento se utiliza para crear instancias de esta clase recibe tres parámetros: tipo, fechaEmision y fechaVencimiento.

this.tipo, this.fechaEmision, y this.fechaVencimiento se usan para referirse a los atributos de la clase y asignarles los valores pasados al constructor.

Los metodos **getters**. Permiten acceder a los atributos de la clase desde fuera de ella:

- getTipo(): Retorna el tipo del documento.
- getFechaEmision(): Retorna la fecha de emisión del documento.
- getFechaVencimiento(): Retorna la fecha de vencimiento del documento.

```java
public class list<T> {

    public static list<Vehiculo> of(Vehiculo vehiculo) {
        // TODO Auto-generated method stub
        throw new UnsupportedOperationException("Unimplemented method 'of'");
    }

}

```

public static list<Vehiculo> of(Vehiculo vehiculo): Este es un método estático que se espera que cree y retorne una instancia de list<Vehiculo>. El método toma como argumento un objeto de tipo Vehiculo.

Cuerpo del método: // TODO Auto-generated method stub: es un comentario que indica que el método aún no ha sido implementado y se debe completar en el futuro.

- throw new UnsupportedOperationException("Unimplemented method 'of'"): Este código lanza una excepción que indica que el método no está implementado. Esto es común durante el desarrollo, para señalar que se necesita completar el método antes de que se pueda usar

```java
import java.awt.Component;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JOptionPane;

public class GestorVehiculos {
    private List<Vehiculo> vehiculos;
    private Object parentComponent;

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

    public void agregarVehiculo(Vehiculo vehiculo) {
        vehiculos.add(vehiculo);
    }

    public void mostrarVehiculos() {
        for (Vehiculo v : vehiculos) {
            JOptionPane.showMessageDialog((Component) parentComponent, v.getTipo()
            + " - " + v.getMarca()
            + " " + v.getModelo() + "\n Año de modelo: " + v.getAño() + "\n Color: "
            + v.getColor() + "\n Kilometros: " +  v.getKilometraje() );
        }
    }

    public void alertarEventos() {
        for (Vehiculo v : vehiculos) {
            v.alertarEventos();
        }
    }
    public List<Vehiculo> getVehiculos() {
        return vehiculos;
    }
}
```

Clase GestorVehiculos

private List<Vehiculo> vehiculos: Una lista que almacena objetos de tipo Vehiculo.

private Object parentComponent: Un componente padre para mostrar los cuadros de diálogo, que se puede usar para establecer la posición del cuadro de mensaje.

Metodo agregarvehiculo Permite agregar un objeto Vehiculo a la lista vehiculos.

Metodo mostrar Vehiculos: Itera sobre la lista de vehículos y muestra un cuadro de diálogo por cada uno. Cada cuadro de diálogo incluye detalles del vehículo, como tipo, marca, modelo, año, color y kilometraje.

Llama al método alertarEventos() de cada objeto Vehiculo en la lista, lo que permite manejar cualquier notificación o alerta asociada con el vehículo.