Universidad de las Fuerzas Armadas

“Creación de Objetos y UML”

Integrantes:

1. Brayan Stehp Mendoza Márquez

Curso:

1323

Asignatura De Programación orientada a objetos

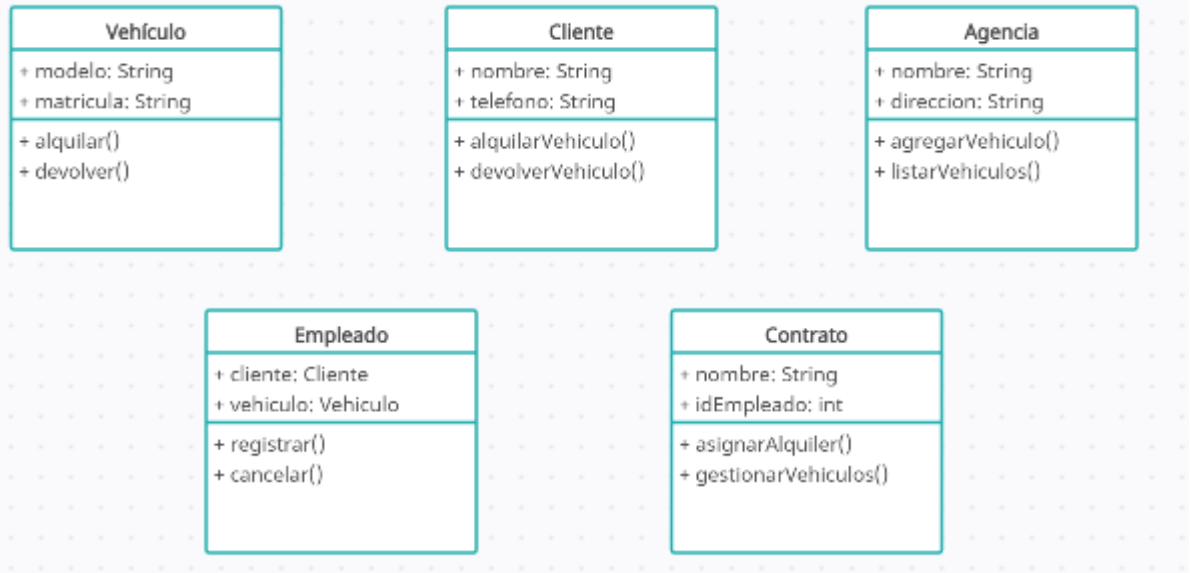
Docente:

LUIS ENRIQUE JARAMILLO MONTAÑO

05 de diciembre de 2024

1. **Diseñe 5 objetos diferentes con su correspondiente diagrama UML, asegurándose de mostrar las relaciones entre ellos.**

1.1 UML



1.2 Relaciones

Para realizar relaciones entre objetos e aprendido por diversas maneras, pero la que mas me gusto fue la forma en la que me encelaron en el instituto en que estaba estudiando ya que hay me hicieron ver que casi cualquier cosa puede tener una relación y con eso aprendí bastante rápido un ejemplo rápido aparte del que estoy haciendo en este deber seria: **Persona ↔ YouTube** como seria esta relación seria así; **una persona ve videos en YouTube y YouTube es una aplicación de videos que permite ver videos a las personas.**

Vehículo ↔ Agencia

Relación: Asociación

Una Agencia tiene una lista de Vehículos que administra.

Cliente ↔ Vehículo

Relación: Composición

Un Cliente puede alquilar uno o más Vehículos, los cuales dependen del cliente.

Empleado ↔ Cliente y Vehículo

Relación: Asociación

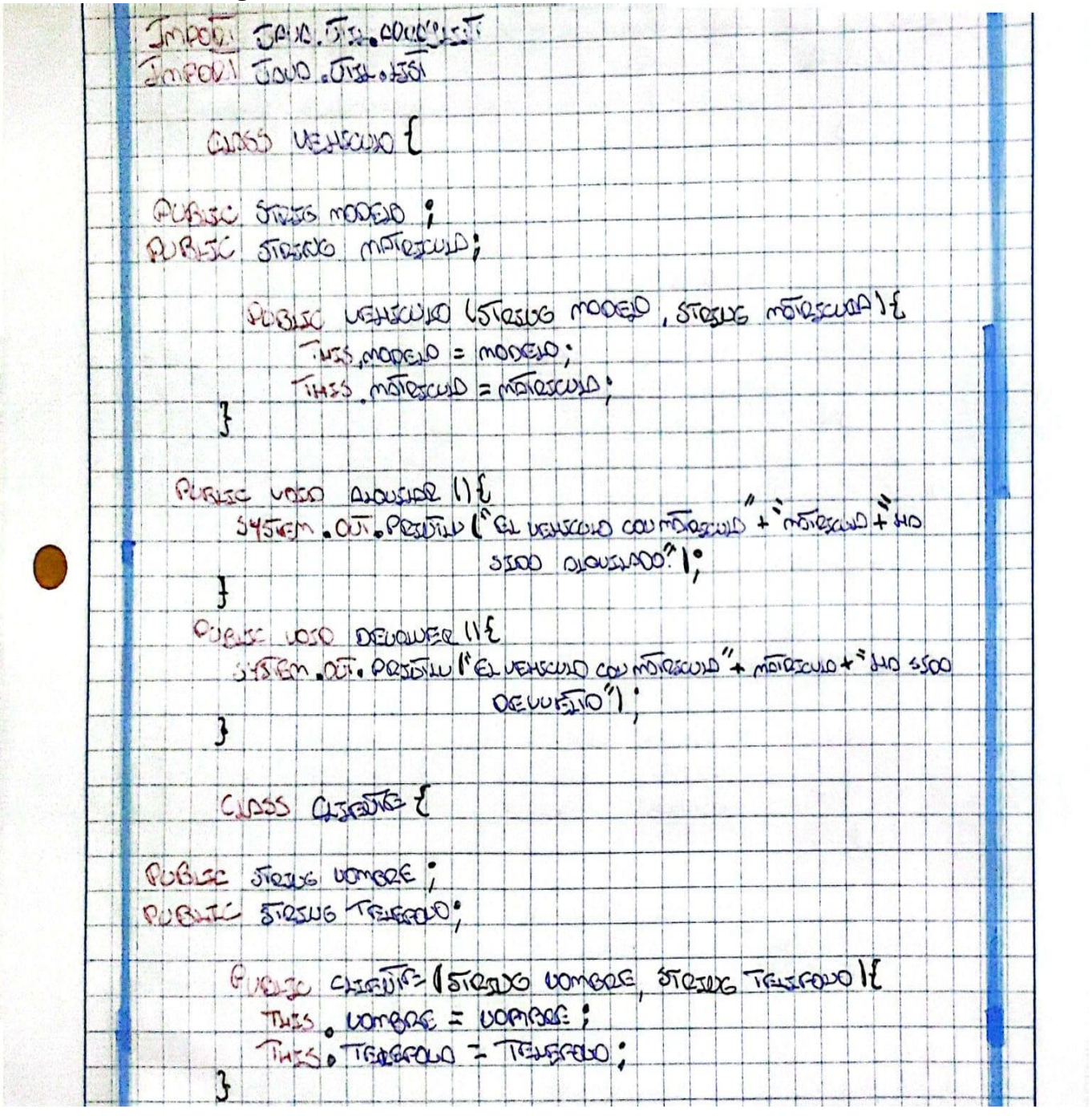
Un Empleado gestiona clientes y vehículos.

Contrato ↔ Cliente y Vehículo

Relación: Agregación

Un Contrato usa instancias de Cliente y Vehículo, pero estas existen independientemente.

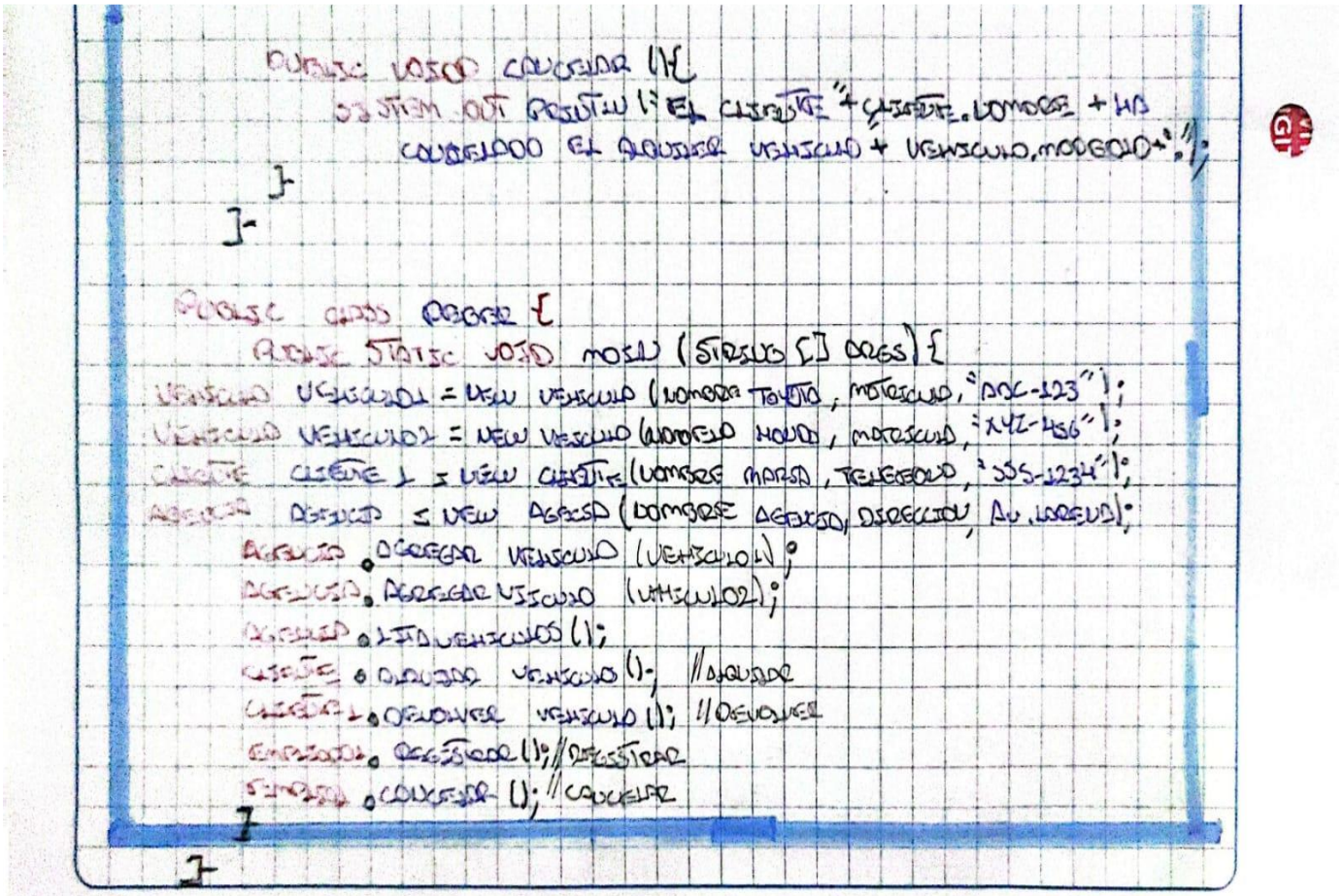
1.3 Código

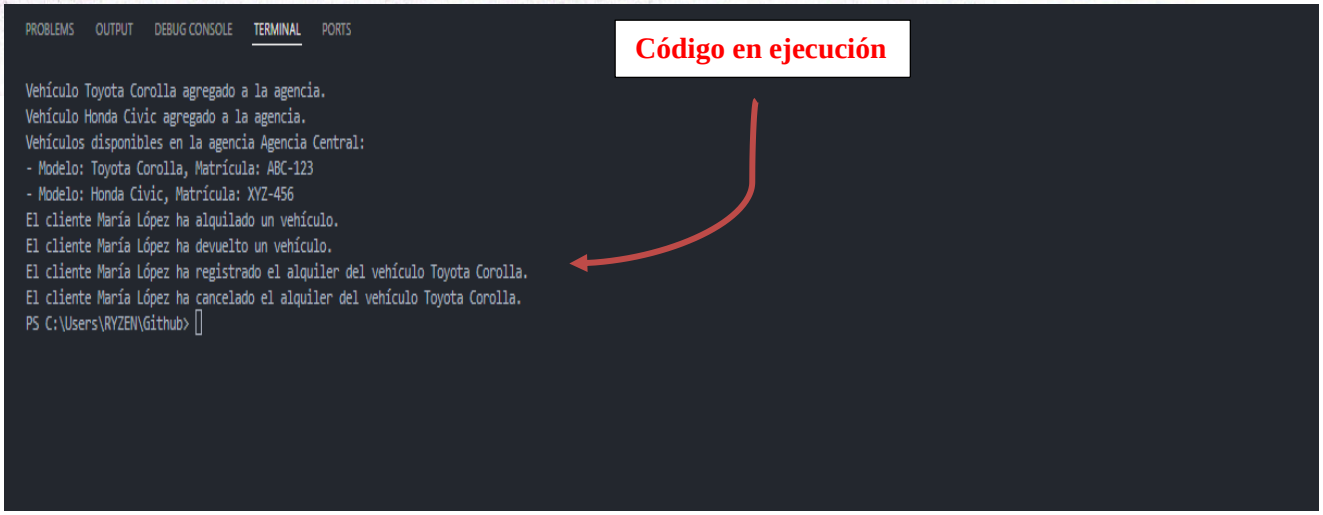


```
public void alquilarVehiculo(){
    System.out.println("El cliente " + nombre + " ha alquilado un
                        vehiculo");
}
public void devolverVehiculo(){
    System.out.println("El cliente " + nombre + " ha devuelto
                        el vehiculo.");
}
}

class Agencia{

public String nombre;
public String direccion;

    public Agencia(String nombre, String direccion){
        this.nombre = nombre;
        this.direccion = direccion;
        this.vehiculo = new Vehiculo[5];
    }
```

Edinacho

```
public void añadirVehiculo(Vehiculo vehiculo){
    vehiculo.add(vehiculo);
    System.out.println("Vehiculo" + vehiculo.modelo + " agregado");
}

public void listarVehiculos(){
    System.out.println("Vehiculos disponibles en la agencia"
            + nombre + ":");
    for(Vehiculo vehiculo : vehiculos){
    System.out.println("-modelo" + vehiculo.modelo + ", matricula" +
        vehiculo.matricula);
    }
}
}

class Empleado

public Cliente cliente;
public Vehiculo vehiculo;

    public Empleado(Cliente cliente, Vehiculo vehiculo){
        this.cliente = cliente;
        this.vehiculo = vehiculo;
    }

    public void registrar(){
        System.out.println("El cliente " + cliente.nombre + " ha
            registrado el alquiler del vehiculo" + vehiculo.modelo);
```





1.4 Resumen

Para hacer este código, seguimos varias ideas que aprendimos del libro del Ing. y aplicamos varios conceptos importantes de Java. Por ejemplo, usamos ArrayList para manejar listas dinámicas, como la flota de vehículos de la agencia, porque nos permite agregar y manejar elementos fácilmente sin preocuparnos por tamaños fijos, algo que el Ing. explicó súper claro en las clases.

Primero, diseñamos las clases según la UML (que básicamente es como un "mapa" para estructurar el programa). Creamos las clases principales (Vehiculo, Cliente, Agencia y Empleado) y les agregamos atributos y métodos que siguen

Luego, en la clase Agencia, usamos un **ArrayList** para manejar todos los vehículos que tiene la agencia. Esto nos sirvió para agregar vehículos y listar todos los disponibles, algo que es súper práctico porque podemos iterar fácilmente sobre la lista.

También trabajamos con la clase Empleado, donde definimos métodos para registrar y cancelar alquileres. Esto lo hicimos para simular cómo un empleado interactuaría con un cliente y los vehículos.

Finalmente, en el método main (), juntamos todo y simulamos un flujo completo: creamos una agencia, añadimos vehículos, un cliente alquila y devuelve un vehículo, y un empleado registra esa acción. Aquí usamos lo que el Ing. menciona mucho sobre cómo separar bien las responsabilidades entre las clases.

En resumen, usamos varias cosas del libro del Ing., como la importancia de estructurar bien el programa con la UML, el uso de ArrayList para manejar datos de forma flexible, y aplicamos POO para que cada clase tenga una responsabilidad clara. Así nos quedó un programa organizado y funcional.