


CARRERA	TECNOLOGÍAS DE LA INFORMACIÓN	
NOMBRE	ALEXANDER JÁCOME	
ASIGNATURA	PROGRAMACIÓN ORIENTADA A OBJETOS	
NRC	1323	
FECHA	21/11/2024	

1. INTRODUCCIÓN

En el ámbito del desarrollo de software, el modelado visual juega un papel fundamental para la comunicación de ideas y la estructuración de soluciones. En este contexto, el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) se ha establecido en un estándar ampliamente utilizado para representar de manera gráfica la arquitectura, el comportamiento y la interacción de los sistemas.

Este informe tiene como propósito analizar y ejemplificar la creación y diseño de diagramas UML, destacando su importancia en la documentación y planificación de proyectos de software.

2. Desarrollo

2.1. Marco Teórico

UML es un lenguaje de modelado gráfico que utilizan los programadores o desarrolladores de software para visualizar y especificar un programa.

Actúa como un bosquejo o plano, ayudando al programador a construir un programa mediante una representación más visual y comprensible de las líneas de código.

2.2. Tipos de Gráficos UML

- **Diagrama de caso de uso:** Este diagrama represente los procesos más importantes que son los actores y los casos que intervienen en el software
- **Diagrama de clases:** Están representando algún concepto o problema, que el programador tenga que solucionar.
- **Diagrama de secuencias:** Se utiliza para la representación objetos del software y del intercambio de los mensajes que existen entre sí.

- **Diagrama de colaboración:** Se utiliza para representar la transmisión de los mensajes que existe entre los objetos y las clases para cumplir con un objetivo.
- **Diagrama de estado:** Se utiliza para la representación de la evolución de un sistema mientras se producen determinados eventos.

2.3. Diseño de objetos

2.3.1. Plataforma Educativa

Objetos:

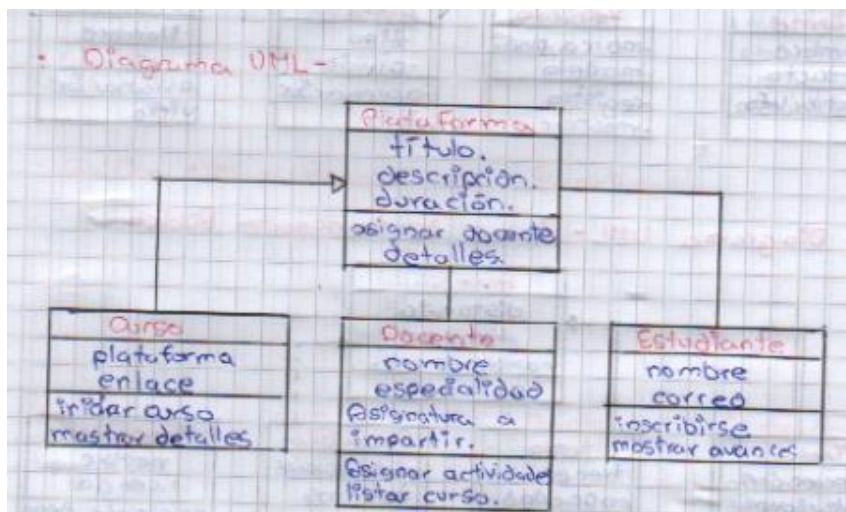
Plataforma (Superclase): Representa un curso general con atributos como el título, descripción y duración.

Curso (Subclase de Curso): Especifica detalles adicionales para cursos en línea, como plataforma y enlace de acceso.

Docente: Gestiona los cursos y las actividades. Tiene atributos como nombre, especialidad y materia que imparte.

Estudiante: Los estudiantes pueden acceder a los cursos.

Diagrama UML



Código

Clase Curso

```
package PlataformaEducativa;

public class Plataforma {
    private String titulo;
    private String descripcion;
    private int duracion; // en horas

    // Constructor
    public Plataforma(String titulo, String descripcion, int duracion) {
        this.titulo = titulo;
        this.descripcion = descripcion;
        this.duracion = duracion;
    }

    // Método para mostrar información del curso
    public void mostrarInformacion() {
        System.out.println("Curso: " + titulo);
        System.out.println("Descripción: " + descripcion);
        System.out.println("Duración: " + duracion + " meses");
    }
}
```

Clase Curso:

```
package PlataformaEducativa;

public class Curso extends Plataforma {
    private String plataforma;
    private String enlace;

    // Constructor
    public Curso(String titulo, String descripcion, int duracion, String
plataforma, String enlace) {
        super(titulo, descripcion, duracion);
        this.plataforma = plataforma;
        this.enlace = enlace;
    }

    // Método para mostrar información específica del curso
    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Plataforma: " + plataforma);
        System.out.println("Enlace: " + enlace);
    }
}
```

```
}  
}
```

Clase Docente

```
package PlataformaEducativa;  
  
public class Docente {  
    private String nombre;  
    private String especialidad;  
    private String materia;  
  
    // Constructor  
    public Docente(String nombre, String especialidad, String materia) {  
        this.nombre = nombre;  
        this.especialidad = especialidad;  
        this.materia = materia;  
    }  
  
    // Método para mostrar información del docente  
    public void mostrarInformacion() {  
        System.out.println("Docente: " + nombre);  
        System.out.println("Especialidad: " + especialidad);  
        System.out.println("Materia: " + materia);  
    }  
  
    // Método para asignar un curso  
    public void asignarCurso(Curso curso) {  
        System.out.println(nombre + " ha sido asignado al curso.");  
    }  
}
```

Clase Estudiante

```
package PlataformaEducativa;  
  
public class Estudiante {  
    private String nombre;  
    private int edad;  
    private int cursosInscritos;  
  
    // Constructor  
    public Estudiante(String nombre, int edad) {  
        this.nombre = nombre;  
        this.edad = edad;  
        this.cursosInscritos = 0;  
    }  
}
```

```

// Método para mostrar información del estudiante
public void mostrarInformacion() {
    System.out.println("Estudiante: " + nombre);
    System.out.println("Edad: " + edad);
    System.out.println("Cursos inscritos: " + cursosInscritos);
}

// Método para inscribirse en un curso
public void inscribirseCurso() {
    cursosInscritos++;
    System.out.println(nombre + " se ha inscrito en un nuevo curso.
Total de cursos: " + cursosInscritos);
}
}

```

Main Clase

```

package PlataformaEducativa;

public class Main {
    public static void main(String[] args) {
        // Crear un curso general
        Plataforma cursoGeneral = new Plataforma("Introducción a la
Programación", "Aprende las bases de programación", 30);
        cursoGeneral.mostrarInformacion();

        // Crear un curso específico
        Curso cursoOnline = new Curso("Java Avanzado", "Aprende conceptos
Java", 4, "Moodle", "www.espe.com/java");
        cursoOnline.mostrarInformacion();

        // Crear un docente
        Docente docente = new Docente("Nicolas Guerra", "Ingeniería de
Software", "Programación Orientada a Objetos");
        docente.mostrarInformacion();
        docente.asignarCurso(cursoOnline);

        // Crear un estudiante
        Estudiante estudiante = new Estudiante("Wendy Jacome", 20);
        estudiante.mostrarInformacion();
        estudiante.inscribirseCurso();
    }
}

```

Esta clase principal sirve para probar las funcionalidades del sistema. Aquí se crean instancias de las clases y se llaman a sus métodos.

2.3.2. Taller automotriz

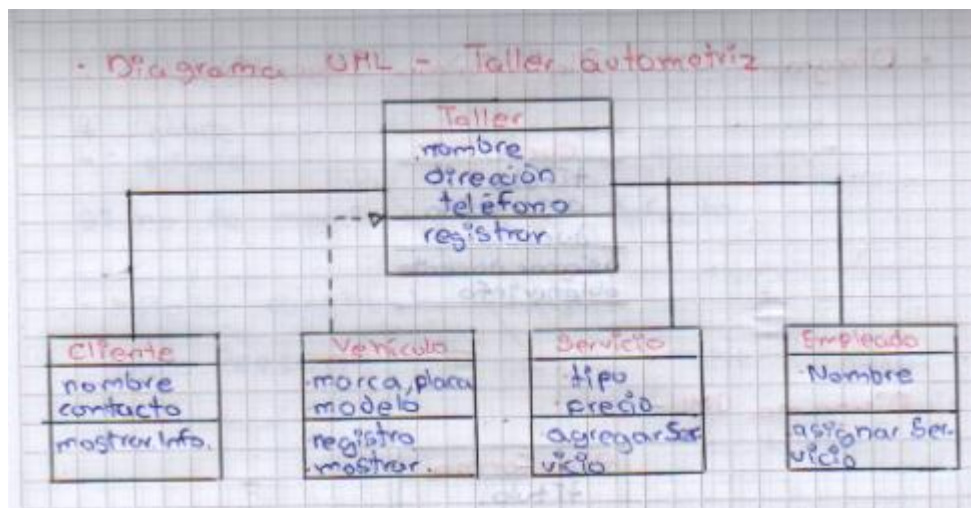
Cliente: Representa a los clientes del taller con información como el nombre.

Vehículo: Detalla las características de los vehículos registrados por los clientes, como marca, modelo y placa.

Servicio: Incluye los diferentes tipos de servicios ofrecidos con su precio.

Empleado: Los empleados del taller que realizan los servicios, con atributos como nombre y especialidad.

Gráfico UML



Código

Clase Cliente

```
package Taller;

public class Cliente {
    String nombre;
    String contacto; // Nuevo atributo

    // Constructor
    public Cliente(String nombre, String contacto) {
        this.nombre = nombre;
        this.contacto = contacto;
    }

    // Método para mostrar información del cliente
    public void mostrarInfo() {
        System.out.println("Cliente: " + nombre);
        System.out.println("Contacto: " + contacto); // Mostrar el nuevo
        atributo
    }
}
```

```
}
```

Clase Factura

```
package Taller;

public class Factura {
    int numero;
    String fecha;
    double montoTotal;

    // Constructor
    public Factura(int numero, String fecha) {
        this.numero = numero;
        this.fecha = fecha;
        this.montoTotal = 50;
    }

    // Método para mostrar la factura
    public void mostrarFactura() {
        System.out.println("Factura No: " + numero + ", Fecha: " +
fecha);
        System.out.println("Monto total: $" + montoTotal);
    }
}
```

Clase Mecanico

```
package Taller;

public class Mecanico {
    String nombre;

    // Constructor
    public Mecanico(String nombre) {
        this.nombre = nombre;
    }

    // Método para reparar un vehículo
    public void repararVehiculo(Vehiculo vehiculo) {
        System.out.println("Mecánico " + nombre + " está reparando el
vehículo con placa: " + vehiculo.placa);
    }
}
```

Clase Servicio

```
package Taller;
```

```

public class Servicio {
    String tipo;
    double costo;

    // Constructor
    public Servicio(String tipo, double costo) {
        this.tipo = tipo;
        this.costo = costo;
    }

    // Método para mostrar información del servicio
    public void mostrarServicio() {
        System.out.println("Servicio: " + tipo + ", con un costo de: $" +
costo);
    }
}

```

Clase Vehiculo

```

package Taller;

public class Vehiculo {
    String placa;
    String marca;
    String modelo;

    // Constructor
    public Vehiculo(String placa, String marca, String modelo) {
        this.placa = placa;
        this.marca = marca;
        this.modelo = modelo;
    }

    // Método para mostrar información del vehículo
    public void mostrarInfo() {
        System.out.println("Vehículo con placa: " + placa + ", Marca: " +
marca + ", Modelo: " + modelo);
    }
}

```

Clase Main (Principal)

```

package Taller;

public class Tallermain {
    public static void main(String[] args) {
        // Crear cliente
    }
}

```



```

    Cliente cliente1 = new Cliente("Flor Mar","0976588876");

    // Crear vehículos
    Vehiculo vehiculo1 = new Vehiculo("LDU2008", "Chery", "QQ3 11");

    // Crear servicio
    Servicio servicio1 = new Servicio("Cambio de aceite", 15.0);

    // Crear mecánico
    Mecanico mecanico1 = new Mecanico("Santi");

    // Crear factura
    Factura factura1 = new Factura(1, "2024-11-25");

    // Mostrar información
    cliente1.mostrarInfo();
    vehiculo1.mostrarInfo();
    servicio1.mostrarServicio();

    // Reparar vehículos
    mecanico1.repararVehiculo(vehiculo1);

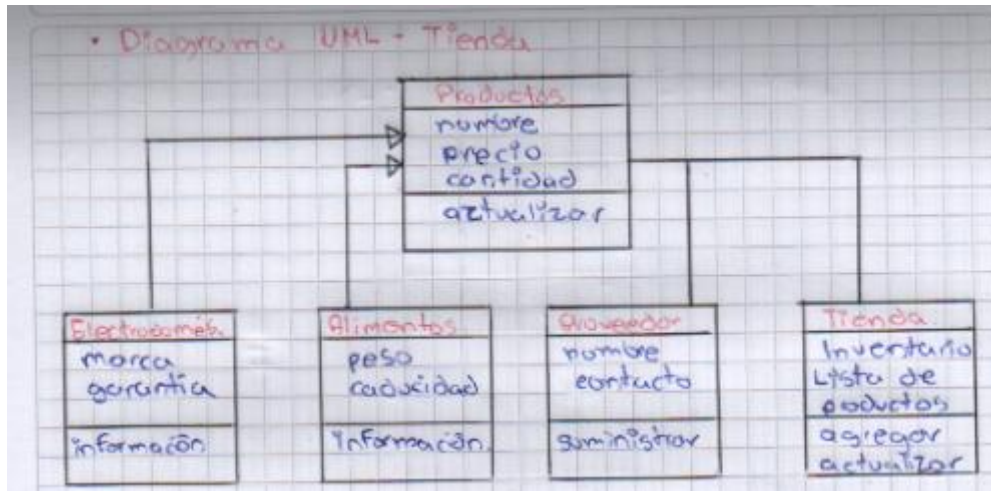
    // Mostrar factura
    factura1.mostrarFactura();
}
}

```

2.3.3. Inventario de una Tienda

- **Producto:** Representa un producto general en el inventario, con atributos como nombre, precio y cantidad disponible.
- **Electrodoméstico:** Especifica características de productos electrónicos, como marca y garantía.
- **Alimentos:** Detalla los productos alimenticios, con atributos como fecha de caducidad y peso.
- **Proveedor:** Suministra productos a la tienda y tiene atributos como nombre y contacto.
- **Tienda:** Administra el inventario, permitiendo agregar y actualizar productos.

Gráfico UML



Código:

Clase Producto:

```
package InventarioTienda;

public class Producto {
    private String nombre;
    private double precio;
    private int cantidadDisponible;

    // Constructor
    public Producto(String nombre, double precio, int cantidadDisponible)
    {
        this.nombre = nombre;
        this.precio = precio;
        this.cantidadDisponible = cantidadDisponible;
    }

    // Métodos
    public void mostrarInformacion() {
        System.out.println("Producto: " + nombre);
        System.out.println("Precio: $" + precio);
        System.out.println("Cantidad disponible: " + cantidadDisponible);
    }

    public void actualizarCantidad(int nuevaCantidad) {
        this.cantidadDisponible = nuevaCantidad;
        System.out.println("La cantidad disponible de " + nombre + " ha sido actualizada a " + cantidadDisponible);
    }
}
```

Clase Electrodomestico

```
package InventarioTienda;

public class Electrodomestico extends Producto {
    private String marca;
    private int garantia; // en meses

    // Constructor
    public Electrodomestico(String nombre, double precio, int
cantidadDisponible, String marca, int garantia) {
        super(nombre, precio, cantidadDisponible);
        this.marca = marca;
        this.garantia = garantia;
    }

    // Métodos
    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Marca: " + marca);
        System.out.println("Garantía: " + garantia + " meses");
    }
}
```

Clase Alimentos

```
package InventarioTienda;

public class Alimentos extends Producto {
    private String fechaCaducidad;
    private double peso; // en kilogramos

    // Constructor
    public Alimentos(String nombre, double precio, int
cantidadDisponible, String fechaCaducidad, double peso) {
        super(nombre, precio, cantidadDisponible);
        this.fechaCaducidad = fechaCaducidad;
        this.peso = peso;
    }

    // Métodos
    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Fecha de caducidad: " + fechaCaducidad);
        System.out.println("Peso: " + peso + " kg");
    }
}
```

Clase Proveedor

```
Clase package InventarioTienda;

public class Proveedor {
    private String nombre;
    private String contacto;

    // Constructor
    public Proveedor(String nombre, String contacto) {
        this.nombre = nombre;
        this.contacto = contacto;
    }

    // Métodos
    public void mostrarInformacion() {
        System.out.println("Proveedor: " + nombre);
        System.out.println("Contacto: " + contacto);
    }
}
```

Clase Tienda

```
package InventarioTienda;

import java.util.ArrayList;

public class Tienda {
    private String nombreTienda;
    private ArrayList<Producto> inventario;
    private ArrayList<Proveedor> proveedores;

    // Constructor
    public Tienda(String nombreTienda) {
        this.nombreTienda = nombreTienda;
        this.inventario = new ArrayList<>();
        this.proveedores = new ArrayList<>();
    }

    // Métodos
    public void agregarProducto(Producto producto) {
        inventario.add(producto);
        System.out.println("Producto agregado al inventario: " +
            producto.getClass().getSimpleName());
    }

    public void mostrarInventario() {
```

```

        System.out.println("Inventario de la tienda " + nombreTienda +
        ":");
        for (Producto producto : inventario) {
            producto.mostrarInformacion();
            System.out.println("-----");
        }
    }

    public void agregarProveedor(Proveedor proveedor) {
        proveedores.add(proveedor);
        System.out.println("Proveedor agregado: " +
        proveedor.getClass().getSimpleName());
    }

    public void mostrarProveedores() {
        System.out.println("Proveedores de la tienda " + nombreTienda +
        ":");
        for (Proveedor proveedor : proveedores) {
            proveedor.mostrarInformacion();
            System.out.println("-----");
        }
    }
}

```

Clase Main (Principal)

```

package InventarioTienda;

public class Main {
    public static void main(String[] args) {
        // Crear tienda
        Tienda tienda = new Tienda("Titan s.a.");

        // Crear productos
        Producto electrodomestico = new Electrodomestico("Refrigeradora",
        500.0, 10, "Samsung", 24);
        Producto alimento = new Alimentos("Manzana", 1.0, 100, "2024-12-
        31", 0.2);

        // Crear proveedores
        Proveedor proveedor1 = new Proveedor("Proveedor
        Electrodomésticos", "23456789");
        Proveedor proveedor2 = new Proveedor("Proveedor Alimentos",
        "0987654321");

        // Agregar productos y proveedores
        tienda.agregarProducto(electrodomestico);
    }
}

```

```

tienda.agregarProducto(alimento);
tienda.agregarProveedor(proveedor1);
tienda.agregarProveedor(proveedor2);

// Mostrar inventario y proveedores
tienda.mostrarInventario();
tienda.mostrarProveedores();
}
}

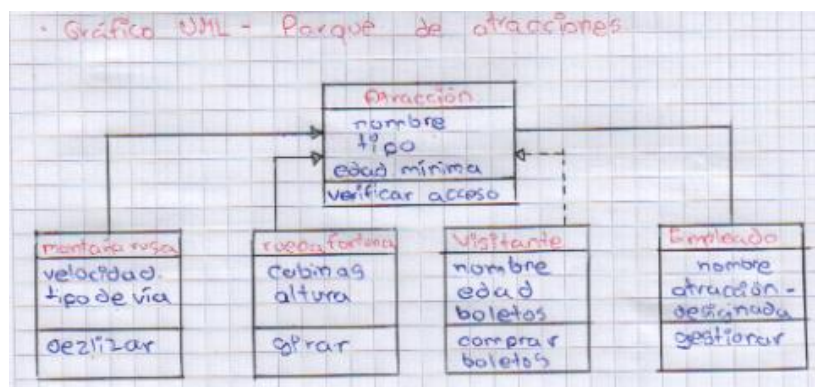
```

2.3.4. Parque de Atracciones

Objetos:

- **Atracción:** Representa una atracción en el parque, con atributos como nombre, tipo, y edad mínima requerida.
- **Montaña Rusa:** Especifica las características de una montaña rusa, como velocidad máxima y tipo de vía.
- **Rueda Fortuna:** Detalla las ruedas de la fortuna, con atributos como el número de cabinas y la altura.
- **Visitante:** Representa a los visitantes del parque, con atributos como nombre, edad y tickets comprados.
- **Empleado:** Gestiona las atracciones y la seguridad del parque, con atributos como nombre, área asignada y turno de trabajo.

Gráfico UML



Código:

Clase Atracciones

```
package ParqueAtracciones;

public class Atraccion {
    private String nombre;
    private String tipo;
    private int edadMinima;

    // Constructor
    public Atraccion(String nombre, String tipo, int edadMinima) {
        this.nombre = nombre;
        this.tipo = tipo;
        this.edadMinima = edadMinima;
    }

    // Métodos
    public void mostrarInformacion() {
        System.out.println("Atracción: " + nombre);
        System.out.println("Tipo: " + tipo);
        System.out.println("Edad mínima: " + edadMinima + " años");
    }
}
```

Clase MontañaRusa

```
package ParqueAtracciones;

public class MontañaRusa extends Atraccion {
    private double velocidadMaxima; // en km/h
    private String tipoVia;

    // Constructor
    public MontañaRusa(String nombre, String tipo, int edadMinima, double
    velocidadMaxima, String tipoVia) {
        super(nombre, tipo, edadMinima);
        this.velocidadMaxima = velocidadMaxima;
        this.tipoVia = tipoVia;
    }

    // Métodos
    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Velocidad máxima: " + velocidadMaxima + "
    km/h");
    }
}
```

```

        System.out.println("Tipo de vía: " + tipoVia);
    }
}

```

Clase RuedaFortuna

```

package ParqueAtracciones;

public class RuedaFortuna extends Atraccion {
    private int numeroCabinas;
    private double altura; // en metros

    // Constructor
    public RuedaFortuna(String nombre, String tipo, int edadMinima, int
numeroCabinas, double altura) {
        super(nombre, tipo, edadMinima);
        this.numeroCabinas = numeroCabinas;
        this.altura = altura;
    }

    // Métodos
    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Número de cabinas: " + numeroCabinas);
        System.out.println("Altura: " + altura + " metros");
    }
}

```

Clase Visitante:

```

package ParqueAtracciones;

public class Visitante {
    private String nombre;
    private int edad;
    private int ticketsComprados;

    // Constructor
    public Visitante(String nombre, int edad, int ticketsComprados) {
        this.nombre = nombre;
        this.edad = edad;
        this.ticketsComprados = ticketsComprados;
    }

    // Métodos
    public void mostrarInformacion() {

```



```

        System.out.println("Visitante: " + nombre);
        System.out.println("Edad: " + edad + " años");
        System.out.println("Tickets comprados: " + ticketsComprados);
    }
}

```

Clase Empleado:

```

public class Empleado {
    private String nombre;
    private String areaAsignada;
    private String turno;

    // Constructor
    public Empleado(String nombre, String areaAsignada, String turno) {
        this.nombre = nombre;
        this.areaAsignada = areaAsignada;
        this.turno = turno;
    }

    // Métodos
    public void mostrarInformacion() {
        System.out.println("Empleado: " + nombre);
        System.out.println("Área asignada: " + areaAsignada);
        System.out.println("Turno: " + turno);
    }
}

```

Clase Main (Principal)

```

public class Empleado {
    private String nombre;
    private String areaAsignada;
    private String turno;

    // Constructor
    public Empleado(String nombre, String areaAsignada, String turno) {
        this.nombre = nombre;
        this.areaAsignada = areaAsignada;
        this.turno = turno;
    }

    // Métodos
    public void mostrarInformacion() {
        System.out.println("Empleado: " + nombre);
        System.out.println("Área asignada: " + areaAsignada);
        System.out.println("Turno: " + turno);
    }
}

```

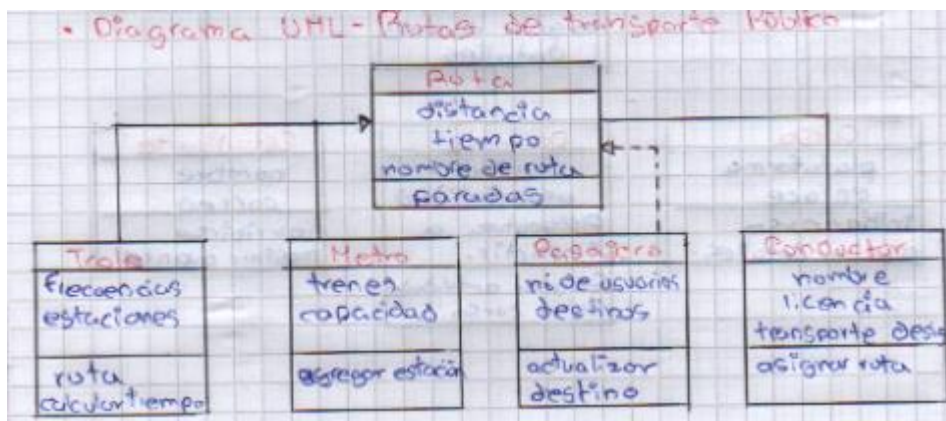
}

2.3.5. Transporte Público de la Ciudad de Quito

Objetos:

- **Ruta:** Representa una ruta de transporte público, con atributos como nombre de la ruta, distancia total y tiempo estimado de viaje.
- **Ruta Trole:** Detalla las rutas de autobús, con atributos como número de autobuses asignados y paradas principales.
- **Ruta Metro:** Especifica las rutas de tren, con información sobre estaciones, frecuencias y capacidad de los trenes.
- **Pasajero:** Representa a los pasajeros que utilizan el servicio de transporte, con atributos como número de usuarios y su destino.
- **Conductor:** Administra el transporte en las rutas, con atributos como nombre, licencia y la ruta asignada.

Gráfico UML



Código:

Clase Ruta

```
package TransportePublico;

public class Ruta {
    private String nombreRuta;
    private double distanciaTotal; // en kilómetros
}
```

```

        private double tiempoEstimado; // en minutos

        // Constructor
        public Ruta(String nombreRuta, double distanciaTotal, double
tiempoEstimado) {
            this.nombreRuta = nombreRuta;
            this.distanciaTotal = distanciaTotal;
            this.tiempoEstimado = tiempoEstimado;
        }

        // Métodos
        public void mostrarInformacion() {
            System.out.println("Ruta: " + nombreRuta);
            System.out.println("Distancia total: " + distanciaTotal + " km");
            System.out.println("Tiempo estimado: " + tiempoEstimado + "
minutos");
        }
    }
}

```

Clase RutaTrole

```

package TransportePublico;

public class RutaTrole extends Ruta {
    private int numeroAutobuses;
    private String paradasPrincipales;

    // Constructor
    public RutaTrole(String nombreRuta, double distanciaTotal, double
tiempoEstimado, int numeroAutobuses, String paradasPrincipales) {
        super(nombreRuta, distanciaTotal, tiempoEstimado);
        this.numeroAutobuses = numeroAutobuses;
        this.paradasPrincipales = paradasPrincipales;
    }

    // Métodos
    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Número de autobuses: " + numeroAutobuses);
        System.out.println("Paradas principales: " + paradasPrincipales);
    }
}

```

RutaMetro

```

package TransportePublico;

```

```

public class RutaMetro extends Ruta {
    private int estaciones;
    private int frecuencia; // en minutos
    private int capacidad; // capacidad por tren

    // Constructor
    public RutaMetro(String nombreRuta, double distanciaTotal, double
tiempoEstimado, int estaciones, int frecuencia, int capacidad) {
        super(nombreRuta, distanciaTotal, tiempoEstimado);
        this.estaciones = estaciones;
        this.frecuencia = frecuencia;
        this.capacidad = capacidad;
    }

    // Métodos
    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Número de estaciones: " + estaciones);
        System.out.println("Frecuencia de trenes: " + frecuencia + "
minutos");
        System.out.println("Capacidad por tren: " + capacidad + "
pasajeros");
    }
}

```

Clase Pasajeros

```

package TransportePublico;

public class Pasajero {
    private String nombre;
    private String destino;

    // Constructor
    public Pasajero(String nombre, String destino) {
        this.nombre = nombre;
        this.destino = destino;
    }

    // Métodos
    public void mostrarInformacion() {
        System.out.println("Pasajero: " + nombre);
        System.out.println("Destino: " + destino);
    }
}

```

Clase Conductor

```

package TransportePublico;

public class Conductor {
    private String nombre;
    private String licencia;
    private String rutaAsignada;

    // Constructor
    public Conductor(String nombre, String licencia, String rutaAsignada)
    {
        this.nombre = nombre;
        this.licencia = licencia;
        this.rutaAsignada = rutaAsignada;
    }

    // Métodos
    public void mostrarInformacion() {
        System.out.println("Conductor: " + nombre);
        System.out.println("Licencia: " + licencia);
        System.out.println("Ruta asignada: " + rutaAsignada);
    }
}

```

Clase Main(Principal)

```

package TransportePublico;

public class Main {
    public static void main(String[] args) {
        // Crear rutas
        RutaTrole rutaTrole = new RutaTrole("Trolebús Central", 15.0,
70.0, 10, "El Ejido - Quitumbe");
        RutaMetro rutaMetro = new RutaMetro("Metro Línea 1", 25.0, 32.0,
15, 5, 200);

        // Crear pasajeros
        Pasajero pasajero1 = new Pasajero("María López", "Quitumbe");
        Pasajero pasajero2 = new Pasajero("Juan Pérez", "El Ejido");

        // Crear conductores
        Conductor conductor1 = new Conductor("Carlos Ruiz", "Tipo ",
"Trolebús Central");
        Conductor conductor2 = new Conductor("Ana García", "No
requerida", "Metro Línea 1");

        // Mostrar información de rutas
        System.out.println("Información de las rutas:");
        rutaTrole.mostrarInformacion();
    }
}

```

```

        System.out.println();
        rutaMetro.mostrarInformacion();
        System.out.println();

        // Mostrar información de pasajeros
        System.out.println("Información de los pasajeros:");
        pasajero1.mostrarInformacion();
        System.out.println();
        pasajero2.mostrarInformacion();
        System.out.println();

        // Mostrar información de conductores
        System.out.println("Información de los conductores:");
        conductor1.mostrarInformacion();
        System.out.println();
        conductor2.mostrarInformacion();
    }
}

```

3. CONCLUSIONES

El desarrollo de este trabajo permitió consolidar los principios de la programación orientada a objetos (POO), recalcando la importancia del diseño modular y estructurado en la creación de software. Mediante la implementación de clases, subclases y sus relaciones, se construyeron sistemas claros y reutilizables que modelan situaciones del mundo real, tales como inventarios de tiendas, gestión de atracciones y sistemas de transporte público. Esto facilitó la comprensión de conceptos como herencia, encapsulación y polimorfismo, además de demostrar cómo estas herramientas contribuyen a mantener la coherencia y escalabilidad del código.

Asimismo, se resaltó la importancia de realizar pruebas y depuración continuas para garantizar la funcionalidad y eficiencia del sistema. Las correcciones en el código y la interacción entre las clases destacaron la necesidad de prestar atención a los detalles en la lógica y estructura del programa.

4. BIBLIOGRAFÍA

Libro - 5 Modelamiento de clases y objetos. (s. f.).

<https://luisjaramillom.github.io/POO.io/Unidades/unidad5/cap5.html>

5. ANEXOS

5.1. RESULTADO OBJETO: PLATAFORMA

```
Curso: Introducción a la Programación
Descripción: Aprende las bases de programación
Duración: 30 meses
Curso: Java Avanzado
Descripción: Aprende conceptos Java
Duración: 4 meses
Plataforma: Moodle
Enlace: www.espe.com/java
Docente: Nicolas Guerra
Especialidad: Ingeniería de Software
Materia: Programación Orientada a Objetos
Nicolas Guerra ha sido asignado al curso.
Estudiante: Wendy Jacome
Edad: 20
Cursos inscritos: 0
Wendy Jacome se ha inscrito en un nuevo curso. Total de cursos: 1
```

5.2. RESULTADO OBJETO: TALLER AUTOMOTRIZ

```
Cliente: Flor Mar
Contacto: 0976588876
Vehículo con placa: LDU2008, Marca: Chery, Modelo: QQ3 11
Servicio: Cambio de aceite, con un costo de: $15.0
Mecánico Santi está reparando el vehículo con placa: LDU2008
Factura No: 1, Fecha: 2024-11-25
Monto total: $50.0
```

5.3. RESULTADO OBJETO: INVENTARIO DE LA TIENDA

```
Producto agregado al inventario: Electrodomestico
Producto agregado al inventario: Alimentos
Proveedor agregado: Proveedor
Proveedor agregado: Proveedor
Inventario de la tienda Titan s.a.:
Producto: Refrigeradora
Precio: $500.0
Cantidad disponible: 10
Marca: Samsung
Garantía: 24 meses
-----
Producto: Manzana
Precio: $1.0
Cantidad disponible: 100
Fecha de caducidad: 2024-12-31
Peso: 0.2 kg
-----
Proveedores de la tienda Titan s.a.:
Proveedor: Indurama
Contacto: 23456789
-----
Proveedor: Frutas y Verduras Inc.
Contacto: 0987654321
-----
```

5.4. RESULTADO OBJETO: PARQUE DE ATRACCIONES

```
Información de las atracciones:
Atracción: Coloso
Tipo: Adrenalina
Edad mínima: 12 años
Velocidad máxima: 90.5 km/h
Tipo de vía: Gravedad 0

Atracción: Perla
Tipo: Familiar
Edad mínima: 0 años
Número de cabinas: 20
Altura: 45.0 metros

Información de los visitantes:
Visitante: María
Edad: 16 años
Tickets comprados: 2

Visitante: Juan
Edad: 4 años
Tickets comprados: 1

Información de los empleados:
Empleado: Carlos
Área asignada: Montaña Rusa
Turno: Mañana

Empleado: Ana
Área asignada: Rueda Fortuna
Turno: Tarde
```


5.5. RESULTADO OBJETO: TRANSPORTE PÚBLICO

Información de las rutas:

Ruta: Trolebús Central

Distancia total: 15.0 km

Tiempo estimado: 70.0 minutos

Número de autobuses: 10

Paradas principales: El Ejido - Quitumbe

Ruta: Metro Línea 1

Distancia total: 25.0 km

Tiempo estimado: 32.0 minutos

Número de estaciones: 15

Frecuencia de trenes: 5 minutos

Capacidad por tren: 200 pasajeros

Información de los pasajeros:

Pasajero: María López

Destino: Quitumbe

Pasajero: Juan Pérez

Destino: El Ejido

Información de los conductores:

Conductor: Carlos Ruiz

Licencia: Tipo

Ruta asignada: Trolebús Central

Conductor: Ana García

Licencia: No requerida

Ruta asignada: Metro Línea 1