HERENCIA Y POLIMORFISMO

JUAN DAVID SERRANO VACCA

UNIVERSIDAD MANUELA BELTRAN

INGENIERIA DE SOTWARE

BOGOTA, COLOMBIA

INTRODUCCION:

Los dos ejercicios trabajados en esta guía se desarrollaron con el objetivo de poner en práctica la teoría del polimorfismo y herencia, para hacer que nuestro código sea más práctico, reutilizable y legible.

El ejercicio introductorio es un sistema de registro, pago y consulta, todo esto en el ámbito universitario. Se usaron 3 clases hijas y 1 padre. En cada clase se implementaron métodos, atributos inicializados con sus debidos constructores y getters and setters. Para después implementar en la clase main lo necesario para que el programa se ejecutara correctamente.

La actividad de trabajo autónomo es un sistema simple de registro de vehículos que permite a los usuarios registrar automóviles y motocicletas, así como mostrar información sobre los vehículos registrados. El objetivo de este proyecto es mostrar cómo se pueden utilizar clases y objetos en Java para modelar objetos del mundo real y gestionarlos en un programa aplicando pilares de la programación aplicada a objetos como la herencia y polimorfismo.

DESCRIPCION DE PROCESOS:

Actividad de trabajo autónomo:

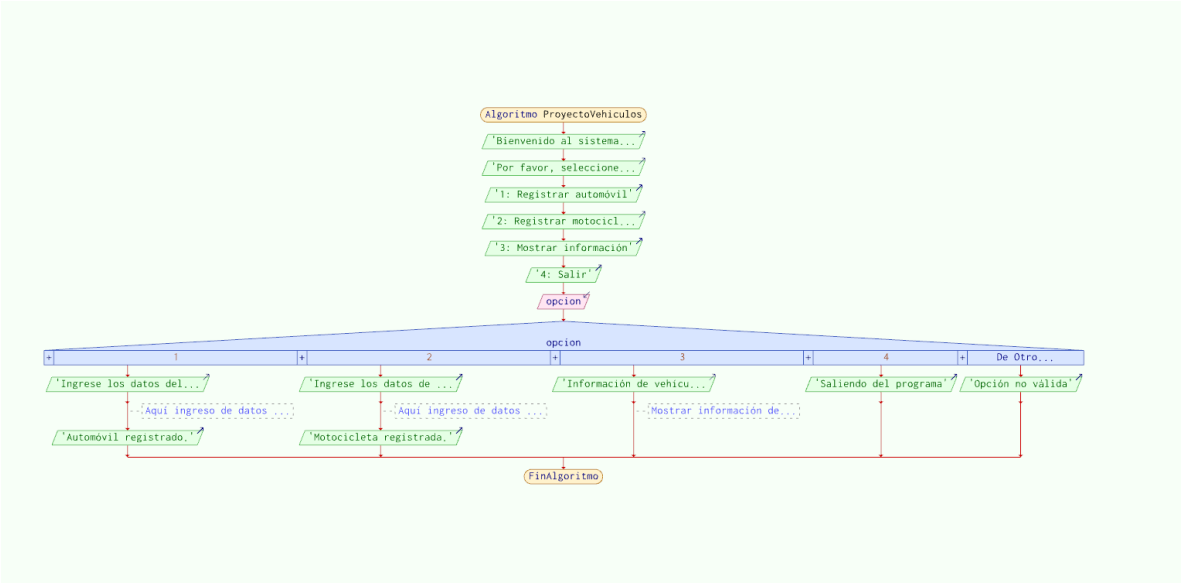
El programa comienza con un menú principal que ofrece tres opciones:

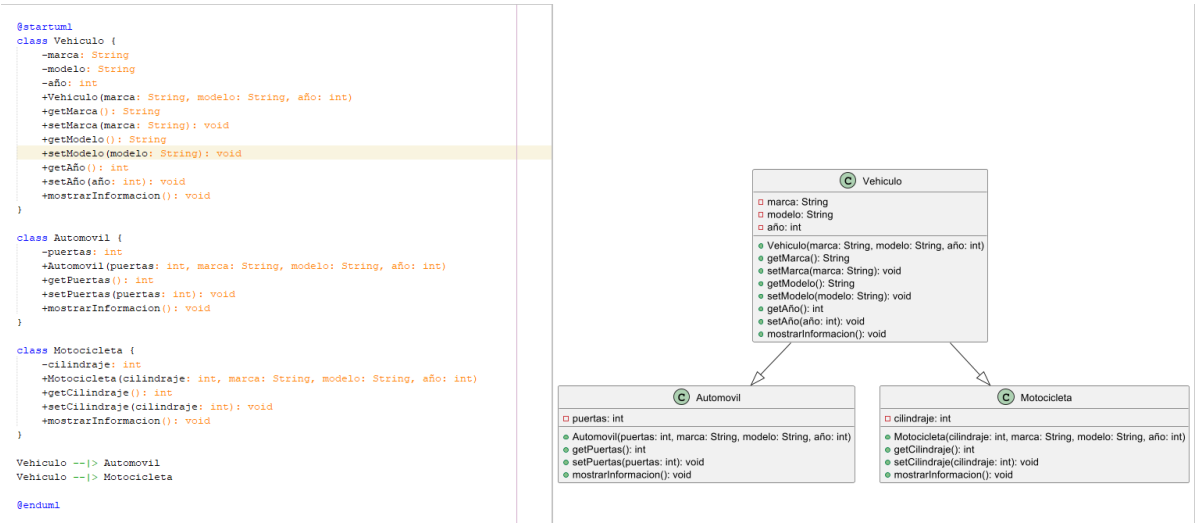
Registrar automóvil: Permite al usuario ingresar información sobre un automóvil, como marca, modelo, año y número de puertas. Luego, este automóvil se almacena en una lista.

Registrar motocicleta: Similar al caso anterior, esta opción permite al usuario ingresar información sobre una motocicleta, incluyendo marca, modelo, año y cilindraje. La motocicleta se almacena en la misma lista que los automóviles.

Mostrar información: Esta opción muestra la información de todos los vehículos (automóviles y motocicletas) registrados hasta el momento.

El usuario puede elegir cualquiera de estas opciones en cualquier momento, y el programa ejecutará la acción correspondiente. El proyecto demuestra conceptos clave de la programación orientada a objetos, como la creación de clases (Vehiculo, Automovil, Motocicleta), la herencia (Automovil y Motocicleta heredan de Vehiculo), y la gestión de listas de objetos**.**





EXPLICACION DE PROCESOS:

Menú Principal:

Cuando el programa comienza, muestra un menú principal con cuatro opciones: Registrar automóvil, Registrar motocicleta, Mostrar información y Salir.

El usuario puede seleccionar una de estas opciones ingresando un número correspondiente.

Selección de Opción: La entrada del usuario se almacena en la variable opcion.

El programa utiliza una estructura switch para determinar qué acción realizar en función de la opción seleccionada.

Registro de Automóvil:

Si el usuario elige la opción 1, el programa solicita al usuario que ingrese los datos del automóvil, como marca, modelo, año y número de puertas.

Luego, crea una instancia de la clase Automovil con los datos ingresados y la agrega a la lista lista.

Registro de Motocicleta:

Si el usuario elige la opción 2, el programa solicita al usuario que ingrese los datos de la motocicleta, como marca, modelo, año y cilindraje.

Luego, crea una instancia de la clase Motocicleta con los datos ingresados y la agrega a la lista lista.

Mostrar Información:

Si el usuario elige la opción 3, el programa recorre la lista lista y muestra la información de todos los vehículos registrados. Esto se logra utilizando el método mostrarInformacion() de cada objeto.

Salir del Programa: Si el usuario elige la opción 4, el programa finaliza y se sale.

Manejo de Clases: El proyecto utiliza tres clases principales: Vehiculo, Automovil y Motocicleta.

Vehiculo es la clase base que contiene propiedades comunes a todos los vehículos y un método mostrarInformacion().

Automovil y Motocicleta son subclases de Vehiculo que agregan propiedades específicas (puertas y cilindraje, respectivamente) y sobrescriben el método mostrarInformacion() para mostrar información adicional específica de cada tipo de vehículo.

ArrayList: Se utiliza un ArrayList llamado lista para almacenar objetos de vehículos registrados. Cada vez que se registra un automóvil o una motocicleta, se agrega a esta lista.

DESCRIPCION DEL PROGRAMA:

ACTIVIDAD DE INTRODUCCION:

Permite gestionar y registrar información personal de estudiantes, docentes y administrativos en una universidad ficticia. El programa ofrece diversas funcionalidades, como la inscripción de personas, la consulta de información personal y el pago de matrícula para estudiantes.

Características Principales

Registro de Personas: El programa permite registrar personas en tres categorías principales: estudiantes, docentes y administrativos. Para cada persona, se solicita información personal, como número de identidad, tipo de documento, nombres, apellidos, dirección, escalafón (en el caso de docentes) y salario (en el caso de administrativos).

Consulta de Información Personal: Se proporciona una opción para consultar la información personal de todas las personas registradas. Este proceso utiliza el polimorfismo para mostrar información específica de cada tipo de persona (estudiantes, docentes y administrativos) de manera uniforme.

Pago de Matrícula: Los estudiantes pueden realizar el pago de matrícula a través de la opción correspondiente en el programa. Se utiliza una implementación específica de pago de matrícula para estudiantes, y esta función también utiliza el polimorfismo para identificar y tratar solo a los estudiantes.

Polimorfismo:

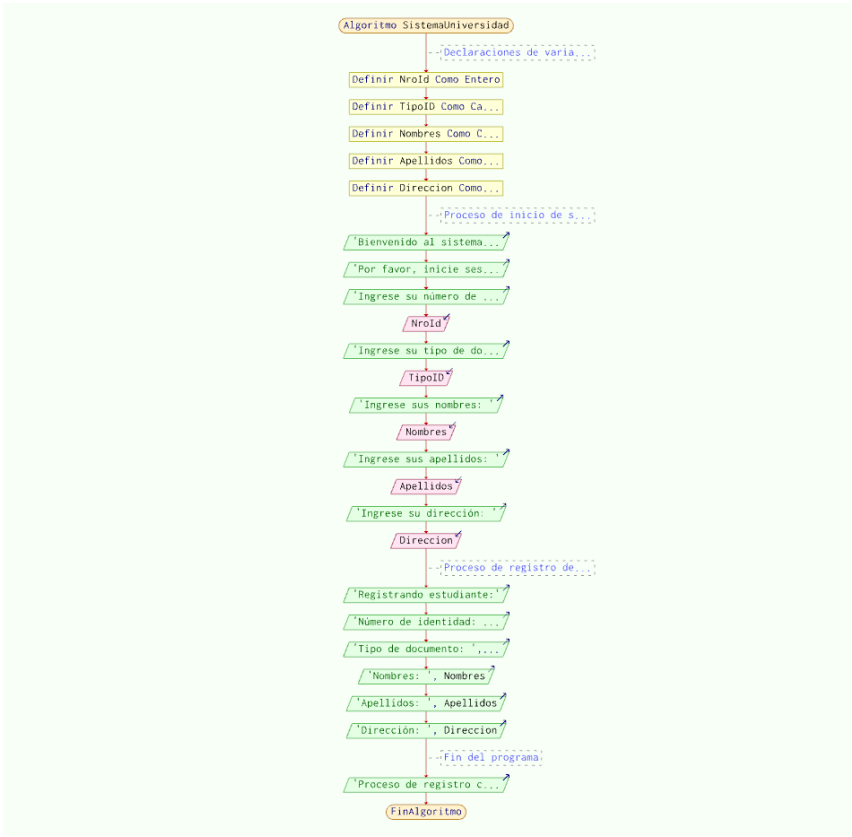
El proyecto hace un uso extensivo del polimorfismo para tratar a los diferentes tipos de personas de manera uniforme y extensible. Cada tipo de persona (estudiante, docente, administrativo) hereda de la clase base Personas y proporciona implementaciones específicas para los métodos como consultarInfoPersonal() y pagarMatricula(). Esto permite que el código principal del programa trabaje con objetos de la clase base y, al mismo tiempo, acceda a las implementaciones específicas de cada subclase.

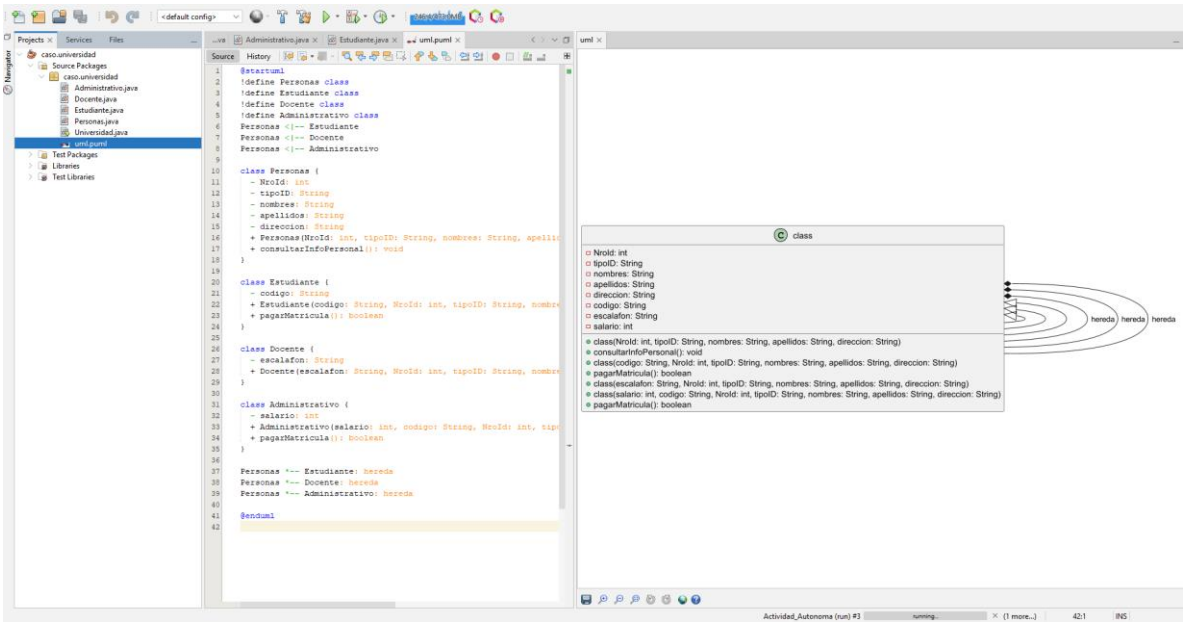
Uso de Herencia:

La herencia se utiliza para crear una jerarquía de clases donde la clase base Personas contiene los atributos y métodos comunes a todas las personas, mientras que las clases derivadas (Estudiante, Docente, Administrativo) añaden atributos y comportamientos específicos para cada tipo de persona.

Ejecución:

El programa se ejecuta en una consola interactiva, donde el usuario puede seleccionar diversas opciones del menú para interactuar con el sistema y gestionar la información de las personas en la universidad.


Algoritmo SistemaUniversidad
Declaraciones de varia...
Definir NroId Como Entero
Definir TipoID Como Ca...
Definir Nombres Como C...
Definir Apellidos Como...
Definir Direccion Como...
Proceso de inicio de s...
'Bienvenido al sistema...
'Por favor, inicie ses...
'Ingrese su número de ...
NroId
'Ingrese su tipo de do...
TipoID
'Ingrese sus nombres: '
Nombres
'Ingrese sus apellidos: '
Apellidos
'Ingrese su dirección: '
Direccion
Proceso de registro de...
'Registrando estudiante:'
'Número de identidad: ...
'Tipo de documento: ',...
'Nombres: ', Nombres
'Apellidos: ', Apellidos
'Dirección: ', Direccion
Fin del programa
'Proceso de registro c...
FinAlgoritmo


class
NroId: int
tipoID: String
nombres: String
apellidos: String
direccion: String
codigo: String
escalafon: String
salario: int
class(NroId: int, tipoID: String, nombres: String, apellidos: String, direccion: String)
consultarInfoPersonal(): void
class(codigo: String, NroId: int, tipoID: String, nombres: String, apellidos: String, direccion: String)
pagarMatricula(): boolean
class(escalafon: String, NroId: int, tipoID: String, nombres: String, apellidos: String, direccion: String)
class(salario: int, codigo: String, NroId: int, tipoID: String, nombres: String, apellidos: String, direccion: String)
pagarMatricula(): boolean
hereda
hereda
hereda

PREGUNTAS ORIENTADORAS:

En este apartado se realiza el análisis de los datos obtenidos, estos pueden ser de forma cualitativa o cuantitativa según la naturaleza

de la práctica.

¿Cuáles fueron los aprendizajes obtenidos al realizar esta guía?, liste como mínimo 3 aprendizajes y relaciónelos con su futuro que hacer profesional.

1. POO, tras trabajar con conceptos fundamentales de la programación orientada a objetos puedo decir que ya aprendí mucho más al respecto. Relacionado a mí que hacer profesional, POO es usado ampliamente por industrias de software.
2. ArrayList , aprendí a usar listas e interactuar con listas. Esto a futuro me va a servir para manejar colecciones.
3. Interacciones, Al crear el menú , estamos dando la oportunidad al usuario para interactuar con el programa.

¿Dónde presento mayor dificultad resolviendo la guía? y ¿cómo lo resolvieron? ¿cuáles fueron las estrategias de solución?

Todas las dudas fueron resueltas en clase.

CONCLUSIONES

Después de realizar satisfactoriamente esta guía, pude evidenciar varios nuevos conceptos y así mismo aprender a usarlos en el lenguaje de programación JAVA. En la investigación hecha de trabajo autónomo pude extender mis conocimientos en Herencia, Polimorfismo, Sobreescritura y manejo de datos usando listas.

El desarrollo de esta guía me ha permitido fortalecer mis habilidades técnicas en programación con Java , así mismo me ha brindado confianza y experiencia para poder abordar temas mas complejos y realizar proyectos mas complejos.