UNIVERSIDAD TECNOLÓGICA NACIONAL

Materia: Programación II

Integrantes: Benevento Juan Manuel, Saederup Lucía

Proyecto : Gestor Concesionaria

Resumen de la Aplicación Desarrollada y Proceso de Desarrollo

La aplicación es un sistema de gestión para una concesionaria de vehículos que permite manejar de manera eficiente los recursos relacionados con compradores, vendedores, vehículos y ventas. Se diseñó con un enfoque modular y orientado a objetos, implementando una interfaz interactiva basada en consola.

El sistema permite las siguientes funciones principales:

- Registrar nuevos vendedores, compradores, vehículos y ventas.
- Consultar información almacenada, como listas de compradores, vehículos y vendedores.
- Mostrar vehículos disponibles según un presupuesto específico.

El proceso de desarrollo del sistema incluyó:

1. **Diseño conceptual**: Se modelaron las entidades del sistema mediante clases que reflejan el dominio del problema (Vehículo, Persona, etc.).
2. **Implementación modular**: Se dividió la lógica en módulos específicos, siguiendo el principio de responsabilidad única.
3. **Persistencia de datos**: Se utilizaron archivos JSON para almacenar datos de manera estructurada y mantener la persistencia entre ejecuciones.
4. **Pruebas funcionales**: Cada funcionalidad fue probada individualmente para asegurar su correcto funcionamiento.

Informe Técnico de la Aplicación

Funcionamiento del Sistema

El sistema se basa en una interfaz en consola que guía al usuario a través de un menú interactivo. Este menú ofrece las opciones para registrar, consultar y gestionar los datos necesarios. La lógica de negocio es gestionada por la clase `Gestor`, mientras que la interacción con el usuario recae en la clase `Menu`. La persistencia de los datos se realiza en archivos JSON mediante la biblioteca Jackson.

Decisiones de Diseño

1. **Orientación a objetos**: Se utilizó una estructura orientada a objetos para representar las entidades principales (`Vehiculo`, `Vendedor`, `Comprador`, etc.) y sus relaciones.
2. **Herencia**: Se creó la clase base `Persona` para abstraer características comunes entre `Vendedor` y `Comprador`.
3. **Persistencia flexible**: La utilización de JSON como formato de almacenamiento permite mantener los datos accesibles, legibles y fáciles de modificar.
4. **Separación de responsabilidades**:
    - El menú (`Menu`) actúa como controlador de interacción.
    - La clase `Gestor` se encarga de gestionar la lógica de persistencia.

Clases Utilizadas y sus Relaciones

1. **Clases principales**:
    - **`Vehiculo`**: Representa los vehículos disponibles para la venta, con atributos como `id`, `modelo`, `marca`, `precio` y `stock`.
    - **`Persona`**: Clase base que contiene atributos comunes a vendedores y compradores (`nombre`, `apellido`, `DNI`, etc.).
    - **`Vendedor`**: Extiende a `Persona` y agrega características específicas, como `salarioBase` y `comisión`.
    - **`Comprador`**: Extiende a `Persona` y añade el atributo `montoDisponible`.
    - **`Venta`**: Representa una transacción entre un vendedor, un comprador y un vehículo, con atributos como `fecha`, `monto`, y referencias a las entidades involucradas.
2. **Relaciones entre clases**:
    - `Persona` es la clase padre de `Vendedor` y `Comprador`.
    - `Venta` relaciona a `Vendedor`, `Comprador` y `Vehiculo`.
    - `Vehiculo` se actualiza dinámicamente según las ventas realizadas.

Estructura del Código

1. **Clases principales**:
    - **`Menu`**:
      Implementa la interfaz del sistema, mostrando opciones al usuario y delegando tareas específicas al `Gestor`.
    - **`Gestor`**:
      Centraliza las operaciones relacionadas con la persistencia, como guardar y leer datos en archivos JSON.
    - **`Persona`, `Vehiculo`, `Vendedor`, `Comprador`, `Venta`**:
      Representan las entidades del sistema, con sus atributos y métodos relacionados.
2. **Ejecución principal**:
    - El punto de entrada es la clase `App`, que instancia `Menu` y ejecuta el método `mostrarMenu`.

Fuentes de Información Consultadas

1. **Documentación oficial de Jackson**:
   Se utilizó para comprender el manejo de objetos Java y JSON, particularmente en la serialización y deserialización.
   URL: https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core
2. **Recursos sobre programación orientada a objetos (POO)**:
   Para estructurar las entidades y sus relaciones de manera coherente y eficiente.
3. **Experiencia previa**:
   Conocimiento adquirido en proyectos previos relacionados con sistemas de gestión.