Asignatura: Programación 3.

Estudiantes: Joaquin Tomas Esposito - Criado Facundo - iavicoli nicolas - Eugenia Toledo

fecha: 05/05/2024

Prog. III – TPE – Primera Parte

1. **Introducción**:

Este es el informe correspondiente a la primera parte del trabajo práctico final de la asignatura indicada, consiste en la gestión de un sistema simple de transporte de mensajería y viaje de clientes con el propósito de integrar los contenidos vistos en el cuatrimestre, a continuación se detallaran la documentación del proyecto, los patrones de diseño implementados para organizar las clases, los contratos de las mismas y el funcionamiento de la aplicación.

2. **Diagrama UML**:

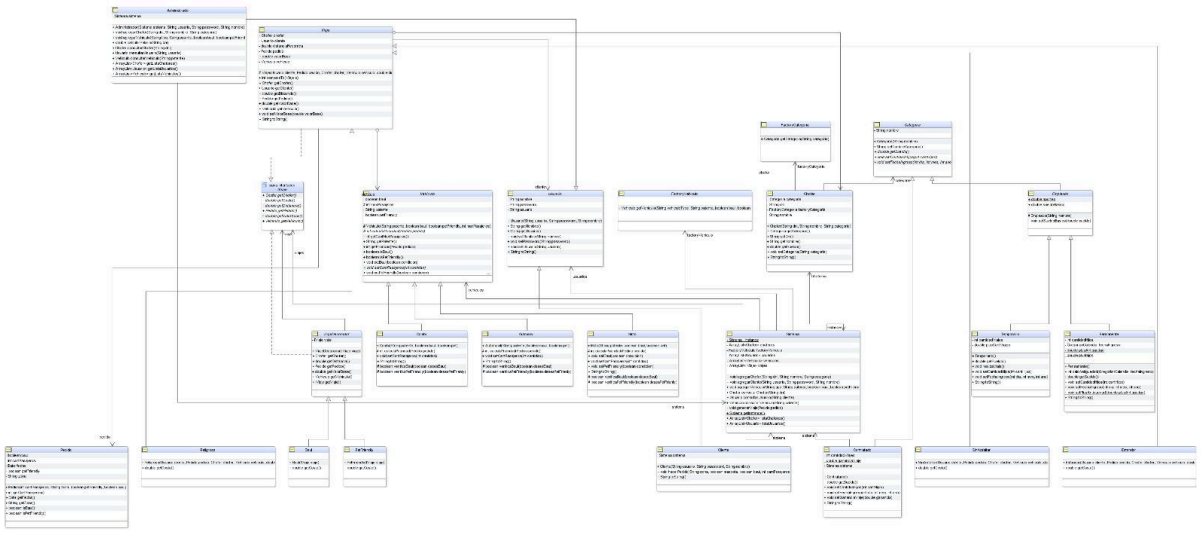

Diagrama con la relación de clases y métodos del sistema.

3. **Organización: Patrones de Diseño y Arquitectura**:

Arquitectura de 3 capas: dividida en 3, la capa de datos donde están las clases que solamente guardan información, la capa de negocio donde está toda la lógica de funcionamiento de la aplicación, y la capa de presentación donde se produce la entrada de datos de cara al usuario del programa.

Patrón Facade: Uso de la clase Sistema para crear una “interfaz” donde desde la capa de presentación se hacen peticiones para ejecutar las funcionalidades del programa.

Patron Singleton: Implementado en la clase Sistema para asegurar una sola instancia y asi afianzar la utilidad en el patrón Facade.

Patrón Factory: Aplicado a las clases que luego serían extendidas para independizar su creación de su tipo, en este caso las clases hijas de empleado y categoría estarían organizadas mediante factories de objetos de sus subtipos.

Patrón decorator: la clase viaje se somete a patrón decorator mediante una interfaz y una clase Decorator que implementa la interfaz con el fin de añadir propiedades al programa como la capacidad de llevar equipaje y mascotas sin la necesidad de modificar toda la clase viaje y mantener coherencia con la clase pedido.

4. **Clases y responsabilidades:**

Administrador

La clase Administrador representa a un usuario con rol de administrador en el sistema.

Automóvil

La clase Automóvil representa un vehículo tipo automóvil.

Baúl

La clase Baul representa un tipo de viaje que requiere espacio en el baÃºl del vehÃculo.

Categoria

La clase Categoría representa una categoría de empleados en la empresa.

Chofer

La clase Chofer representa a un conductor con su información personal y categoria

Cliente

La clase Cliente representa a un usuario cliente en el sistema.

Combi

La clase Combi representa un vehÃculo tipo combi.

Contratado

La clase Contratado representa una categorÃa de empleados contratados en la empresa.

Empleado

La clase Empleado representa a los empleados pertenecientes a la empresa, ya sean permanentes o temporarios Es una clase abstracta que define un sueldo basico y el porcentaje de aportes jubialtorios.

Estandar

La clase Estandar representa un tipo de viaje estándar.

FactoryCategoria

La clase FactoryCategoria es una fábrica que crea objetos de tipo Categoría según el tipo especificado.

FactoryVehiculo

La clase FactoryVehiculo es una fábrica que crea objetos de tipo Vehículo según el tipo especificado.

IViaje

La interfaz IViaje define los métodos necesarios para representar un viaje.

Moto

La clase Moto representa un vehÃculo tipo motocicleta.

Pedido

La clase Pedido representa un pedido de viaje realizado por un cliente.

Peligrosa

La clase Peligrosa representa un tipo de viaje que se realiza en zonas peligrosas.

Permanente

La clase Permanente representa a un empleado permanente en la empresa.

PetFriendly

La clase PetFriendly representa un tipo de viaje que permite llevar mascotas.

SinAsfaltar

La clase SinAsfaltar representa un tipo de viaje que se realiza en terrenos sin asfaltar.

Sistema

Clase Sistema representa la gestion de peticiones del usuario, informes, viajes, etc.

Temporario

La clase Temporario representa a un empleado temporal en la empresa.

Vehiculo

Esta clase abstracta representa un vehÃculo genÃ©rico utilizado en viajes.

Viaje

Esta clase abstracta representa un viaje genérico y proporciona información sobre el pedido, el chofer, el vehículo y la distancia recorrida.

ViajeDecorator
Clase abstracta que representa un decorador para un viaje.

5. **Conclusiones**:

Durante el desarrollo de este proyecto, hemos aplicado una variedad de conceptos y técnicas fundamentales en el ámbito de la programación en Java. La implementación de patrones de diseño, como Singleton, Factory, facade y decorator, ha mejorado la capacidad de mantener un código organizado. La documentación Javadoc exhaustiva proporciona claridad y guía para la futura extensión del código. En términos generales fue posible elaborar una aplicación bajo las pautas de la cátedra.