***Trabajo final (Subí que te llevo)***

***Primera parte***

***Grupo 3***

***Cátedra:***

Programación 3

***Integrantes:***

* Bondoni juan
* Giardino Tomás
* Bengoa Sebastian
* Saladino Juan cruz

***URL del repositorio GitHub:***

https://github.com/SaladinoJC/TP\_Progra3\_2024.git

Índice

Páginas:

1. *Introducción*  3
2. *Diseño UML* 4
3. *Clases* 5
4. *Patrones de diseño*  7
5. *Funcionamiento del sistema* 10

*Introducción*

El trabajo realizado consiste en desarrollar un sistema informático, realizado en el lenguaje Java, que modela el funcionamiento de una empresa de traslado de pasajeros.

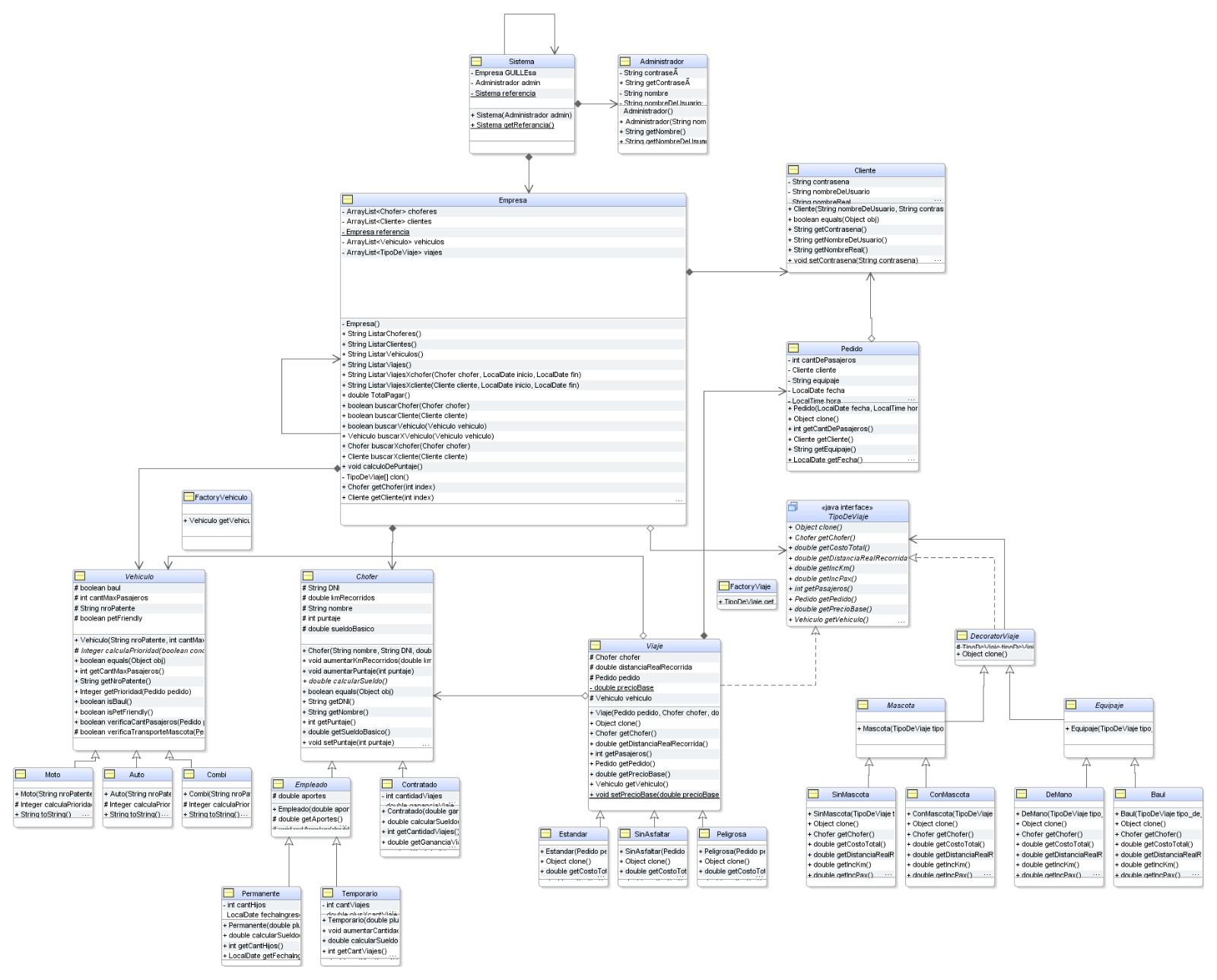
La Empresa cuenta con una flota de Vehículos de diferentes características, un conjunto de Choferes que manejan cualquiera de los Vehículos y un conjunto de Clientes registrados con los cuales opera. Estos actores y recursos son dinámicos en el tiempo, o sea, pueden aumentar.

Para su confección, se modelaron objetos representativos de una empresa de viajes, y se utilizaron diferentes patrones de diseño para la implementación de su diverso comportamiento.

A continuación se detallan las partes más importantes del código, junto con un diagrama UML, para comprender las diferentes partes y el funcionamiento de este sistema.

***Diseño UML***

Este UML es una versión simplificada con tal de que se aprecie la información más importante del sistema informático propuesto. El grupo en el trabajo entregará el UML completo.



***Clases***

* ***Empresa:*** Modela la empresa de transporte. Contendrá las listas de clientes, choferes, vehículos y viajes realizados; esta clase tendrá una única instancia y posee las funcionalidades tanto del administrador como del cliente.
* ***Cliente:*** Modela el cliente que quiere realizar un viaje. Cuenta con sus datos tanto para iniciar sesión como también para realizar pedidos.
* ***Pedido:*** Es el pedido que realiza el cliente. Cuenta con toda la información necesaria para gestionar un posible viaje, con los vehículos y choferes que tenga la empresa.
* ***Viaje:*** Modela el viaje que realiza un cliente. Contará con un pedido, un chofer, un vehículo y el precio a pagar por el cliente. Es una clase abstracta.
* ***Chofer:*** Modela al chofer que se elige para cada viaje, y posee algunos datos personales. Es una clase abstracta.
* ***Vehículo:*** Modela un vehículo de la empresa. Cuenta con todos sus datos, como patente, cantidad máxima de pasajeros, etc; es una clase abstracta.
* ***Moto:*** Clase concreta que modela una moto, y hereda de la clase vehículo todos sus métodos y atributos.
* ***Auto:*** Clase concreta que modela un auto, y hereda de la clase vehículo todos sus métodos y atributos.
* ***Combi:*** Clase concreta que modela una combi, y hereda de la clase vehículo todos sus métodos y atributos.
* ***Empleado:*** Es una clase intermedia entre Chofer y Temporal, Permanente. Es una clase abstracta que hereda de la clase Chofer todos sus métodos y atributos; pero agrega los atributos aportes y sueldoBásico para un mejorar cálculo de los sueldos finales de choferes Temporales y Permanentes
* ***Contratado:*** Es una clase concreta que hereda de chofer. Cuenta con la cantidad de viajes que realizó, así como la ganancia que se lleva por realizarlos.
* ***Temporario:*** Modela a los choferes con contratos temporales de la empresa y cuenta con la cantidad de viajes y el porcentaje que se lleva de cada viaje. Es una clase concreta que hereda de Empleado.
* ***Permanente:*** Modela los choferes que son permanentes, y es una clase concreta que hereda de Empleado.
* ***Estándar:*** Es una clase concreta que hereda los atributos y métodos de la clase Viaje, modela un TipoDeViaje realizado en una zona estándar.
* ***SinAfaltar:*** Es una clase concreta que hereda los atributos y métodos de la clase Viaje, modela un TipoDeViaje realizado en una zona sin asfaltar.
* ***Peligrosa:*** Es una clase concreta que hereda los atributos y métodos de la clase Viaje, modela un TipoDeViaje realizado en una zona peligrosa.
* ***DecoratorDeViaje:*** Es una clase Abstracta que funciona como el decorador del TipoDeViaje. Posee un único atributo que es del tipo de la interfaz TipoDeViaje.
* ***Mascota:*** Clase abstracta que hereda de DecoratorDeViaje. Representa un decorador de TipoDeViaje que distingue si el viaje es con o sin mascota.
* ***ConMascota:*** Clase concreta que hereda de Mascota y modela un TipoDeViaje con una mascota.
* ***SinMascota:*** Clase concreta que hereda de mascota y modela un TipoDeViaje sin una mascota.
* ***Equipaje:*** Clase abstracta que hereda de DecoratorDeViaje. Representa un decorador del TipoDeViaje que distingue si el viaje es con un equipaje de mano o se debe utilizar el baúl.
* ***Demano:*** Clase concreta que hereda de Equipaje y modela un TipoDeViajecon un equipaje de mano.
* ***Baul:*** Clase concreta que hereda de Equipaje y modela un TipoDeViaje que utiliza baúl.
* ***TipoDeViaje:*** Interfaz que conecta la clase abstracta Viaje con el DecoratorDeViaje.
* ***Sistema:*** Es la clase encargada de delegar a la empresa los diferentes pedidos que se realizan desde la clase Prueba. (aplicación del patrón FACADE)
* ***Administrador:*** Usuario que se encargará de algunas de las funciones que realiza la clase Sistema.
* ***FactoryViaje:*** Modela el patrón factory para crear un TipoDeViaje.
* ***FactoryVehiculo:*** Modela el patrón factory para crear algún tipo de vehículo.

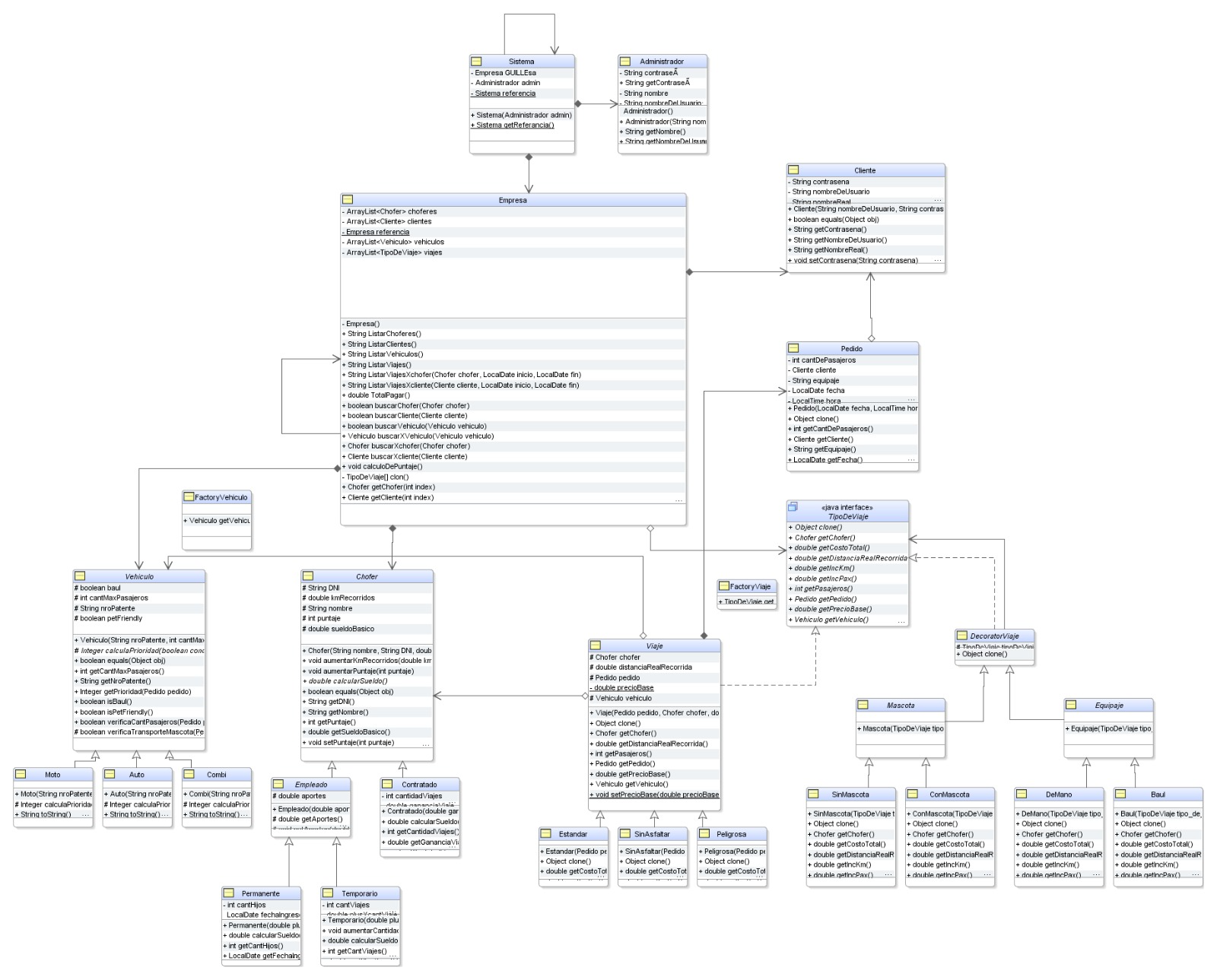
***Patrones de diseño***

En este sistema informático se utilizaron diferentes patrones de diseño. A continuación se comenta la implementación y el porqué del uso de cada uno:

* **Patrón Decorator:**

Este patrón se utiliza para poder gestionar las características del viaje. El cliente especifica en el pedido si lleva equipaje de mano o debe depositar cosas en el baúl, si lleva mascota o no, y también especifica la zona por donde se realizará el viaje, es decir, si es estándar, sin asfaltar o peligrosa.

Se utilizó este patrón ya que permite que el código sea expansible para el futuro; por ejemplo: Si se quiere agregar, un tipo de animal como lo es un perro, se debe aumentar el costo por km recorrido un 15% del básico. Únicamente se deberá incluir un clase Perro que herede de ConMascota y que tenga un nuevo comportamiento para adicionar dicho costo al total.

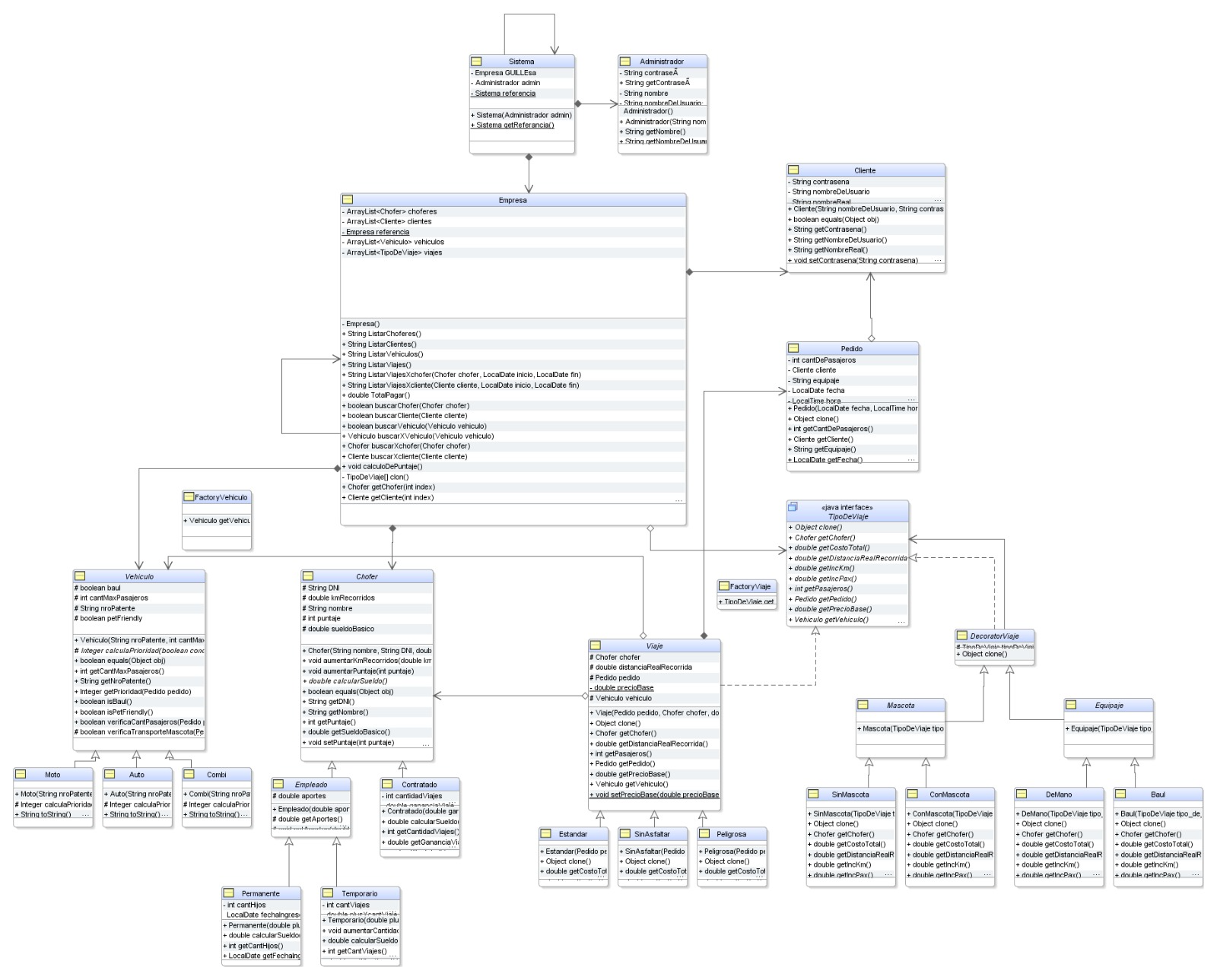


* **Patrón singleton:**

Este patrón se utiliza para que objetos como Empresa o Sistema tengan una única instancia proporcionando acceso global a ella.

La razón de su uso fue que la empresa de traslado de pasajeros es única, ya que debe contar con listas de todos los clientes, vehículos, choferes y viajes.

El sistema también es único y manejado por el administrador.

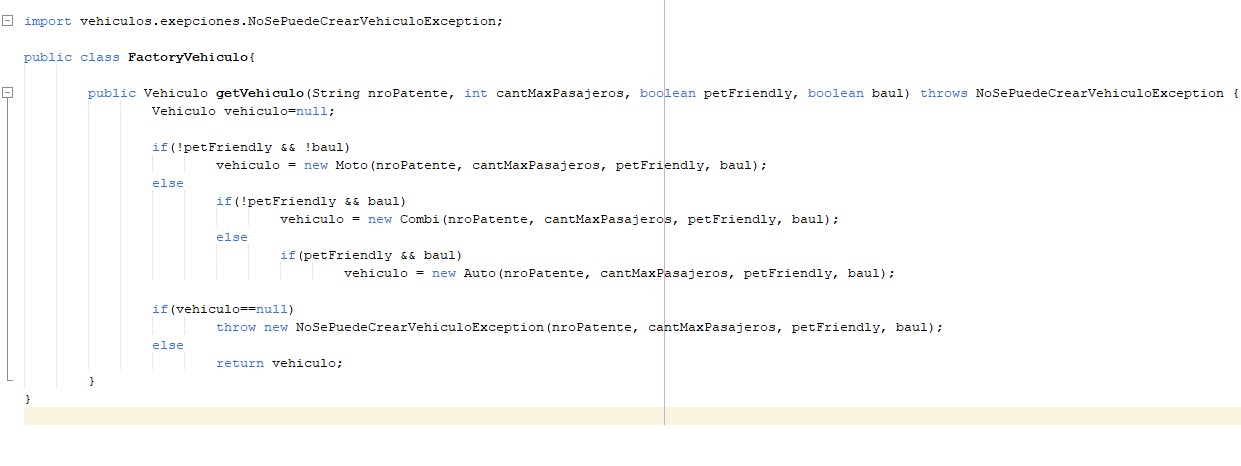


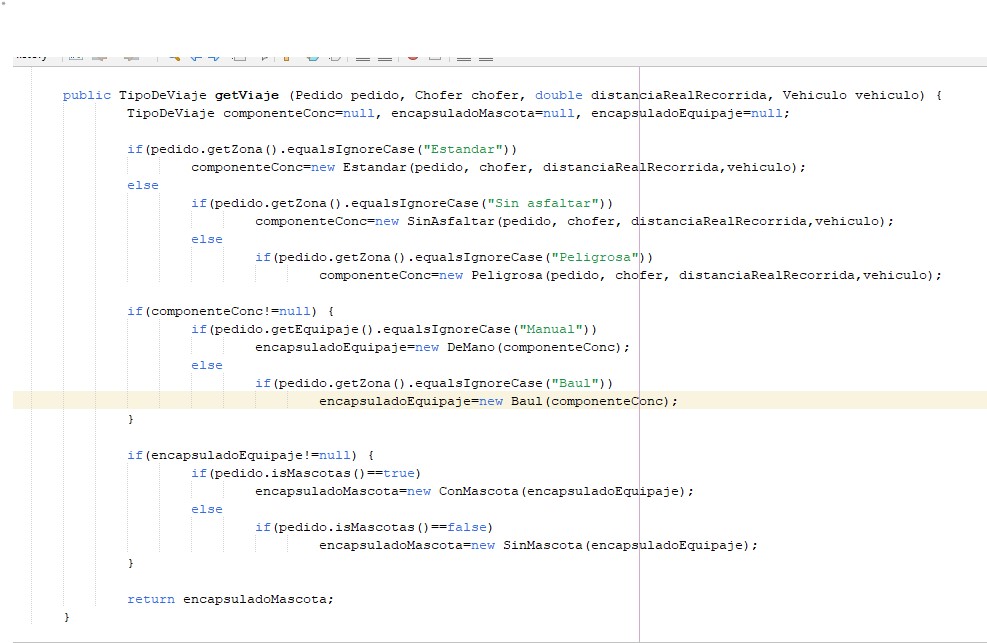
* **Patrón factory:**

El patrón factory se utiliza para poder instanciar objetos de tipo Vehículo y TipoDeViaje.

Cuando llega un pedido validado, se llamará al FactoryViaje que se encarga de crear un TipoDeViaje con los encapsulados del DecoratorDeViaje.

El FactoryVehiculo se encarga de la creación de una nuevos vehículos y de esta manera instanciar vehículos de manera más sencilla, de acuerdo a los parámetros que reciba. Por lo cual agregar otros tipos de vehículos a futuro solo requiere expandir el FactoryVehiculo con una clase hija que herede de FactoryVehiculo y agregue un nuevo tipo de vehículo.

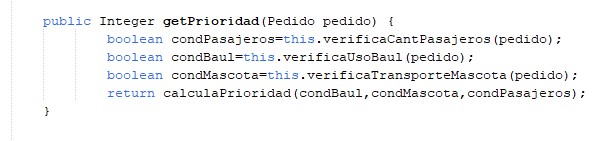




* **Patrón template:**

Este patrón se utiliza para asignarle el mejor vehículo a un viaje solicitado por un cliente. Esto depende de las características del pedido, es decir, si lleva mascota o no, si requiere del uso de un baúl para transportar su equipaje y si cumple con la capacidad máxima de pasajeros que admiten los vehículos de la empresa.

El método que indica que tipo de vehículo se asigna es getPrioridad().



* **Patrón FACADE:**

Este patrón está implementado de forma implícita por la clase Sistema, que se encarga de delegarle a la clase Empresa la implementación de las funcionalidades tanto del administrador como de los diferentes clientes.

***Funcionamiento del sistema***

La clase Sistema es la que implementa el patrón FACADE, con lo cual su función es delegarle a la clase Empresa todas las funciones pedidas desde el main presente en la clase Prueba.

Algunas de estas son:

* La inserción de un nuevo cliente, chofer, viaje o vehículo.(Tanto los choferes, como los clientes o vehículos no pueden repetirse).
* La consulta y/o modificación de algún dato que posea un objeto cliente, chofer, viaje o vehículo.
* El listado de los Viajes realizados por un Chofer en particular en un período de días.
* El listado de los Viajes realizados por un Cliente en particular en un período de días.
* Mostrar el total de dinero necesario para pagar salarios de los choferes, así como también el cálculo de salarios mensuales de cada chofer.
* Cálculo del puntaje de cada chofer.
* Manejar la evolución del viaje desde que nace como pedido, mediante diferentes funciones de carácter secuencial.