Ejercicio 6

Requisitos funcionales

Acciones:

- Registrar nuevos pedidos de entrega.
- Seleccionar el tipo de transporte adecuado para la entrega según la distancia y el peso.
- Calcular el costo total de la entrega, aplicando descuentos si es necesario.
- Generar un reporte mensual que indique los ingresos totales por tipo de transporte.
- Guardar y cargar la información de los pedidos en archivos CSV.

Datos necesarios:

- Nombre del cliente
- Peso del artículo
- Distancia desde la oficina al lugar de entrega
- Tipo de transporte seleccionado
- Costo total del pedido

Análisis y Diseño

Identificación de clases y atributos:

Clases:

- Interfaz: Es una clase interface que tendrá los métodos que va a realizar un transporte.
- Transporte: Es una clase abstracta que tiene los atributos generales de un transporte.
- Camión: Es una clase con los atributos específicos de un camión.
- Motocicleta: Es una clase con los atributos específicos de una motocicleta.
- Drone: Es una clase con los atributos específicos de un drone.
- Pedido: Es la clase en la que se realizaran los pedidos y guardar en un csv.
- Gestion: Es la clase encargada de realizar los métodos.
- DriverProgram: Es la clase encargada de interactuar con el usuario.

Atributos:

Interfaz:

Transporte:

- Double tarifabase: Es el encargado de guardar el precio de la tarifa base por kilómetro.
- Double capacidad: Es el encargado de guardar el peso del articulo a enviar.

Camion:

- Int descuento: Es el porcentaje de descuento por cierta distancia de kilómetros recorridos.

Motocicleta:

Drone:

- Int limite: Es el encargado de guardar el límite que puede recorrer el drone para una entrega.

Pedido:

- Date fecha: Es el encargado de guardar la fecha en la que se realizo el pedido.
- Double costototal: Es el encargado de guardar el costo total del pedido

Gestion:

- ArrayList<Pedido> pedidos: Es el encargado de almacenar los pedidos realizados.

DriverProgram:

Relación:

- Interfaz: Es una clase interface.
- Transporte: Es una clase abstracta que implementa Interfaz.
- Camión: Es una clase modelo que extiende Transporte.
- Motocicleta: Es una clase modelo que extiende Transporte.
- Drone: Es una clase modelo que extiende Transporte.
- Pedido: Es una clase que está compuesta por Transporte.
- Gestión: Es la clase encargada de realizar todos los métodos necesarios.
- DriverProgram: Es la clase encargada de interactuar con el usuario.

Métodos y comportamiento polimórfico:

Métodos interfaz:

- Int calcularCosto()
- String validarEntrega()

Interfaz:

- Int calcularCosto(): Se encargará de calcular el costo total del pedido.
- String validarEntrega(): Se encargará de dar aprobación a la entrega o no.

Transporte:

- Getters y setters: Se encargarán de recibir y devolver datos.

Camion:

- Getters y setters: Se encargarán de recibir y devolver datos.

Motocicleta:

- Getters y setters: Se encargarán de recibir y devolver datos.

Drone:

- Getters y setters: Se encargarán de recibir y devolver datos.

Pedido:

- Getters y setters: Se encargarán de recibir y devolver datos.

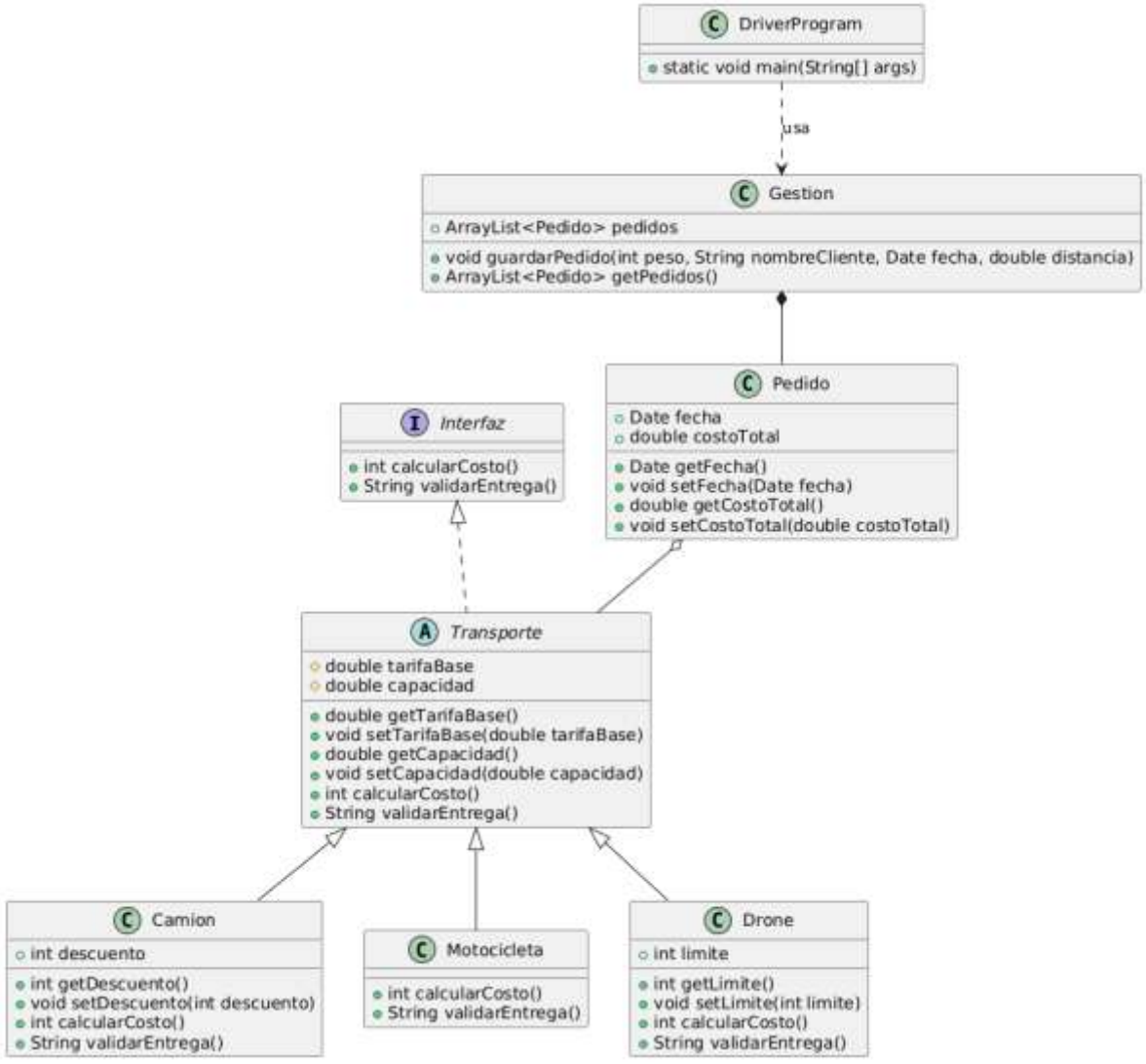
Gestion:

- String guardarPedido(int, String, Date, double): Se encargará de guardar los pedidos en el ArrayList

DriverProgram:

- public static void main(String[] args).

Diseño UML


DriverProgram
+ static void main(String[] args)
usa
Gestion
- ArrayList<Pedido> pedidos
+ void guardarPedido(int peso, String nombreCliente, Date fecha, double distancia)
+ ArrayList<Pedido> getPedidos()
Pedido
- Date fecha
- double costoTotal
+ Date getFecha()
+ void setFecha(Date fecha)
+ double getCostoTotal()
+ void setCostoTotal(double costoTotal)
Interfaz
+ int calcularCosto()
+ String validarEntrega()
Transporte
double tarifaBase
double capacidad
+ double getTarifaBase()
+ void setTarifaBase(double tarifaBase)
+ double getCapacidad()
+ void setCapacidad(double capacidad)
+ int calcularCosto()
+ String validarEntrega()
Camion
- int descuento
+ int getDescuento()
+ void setDescuento(int descuento)
+ int calcularCosto()
+ String validarEntrega()
Motocicleta
+ int calcularCosto()
+ String validarEntrega()
Drone
- int limite
+ int getLimite()
+ void setLimite(int limite)
+ int calcularCosto()
+ String validarEntrega()