FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III Curso X Primer cuatrimestre de 2021

Alumno:	GONZALEZ, Agustin
Número de padrón:	106086
Email:	aguselgonza@gmail.com

Índice

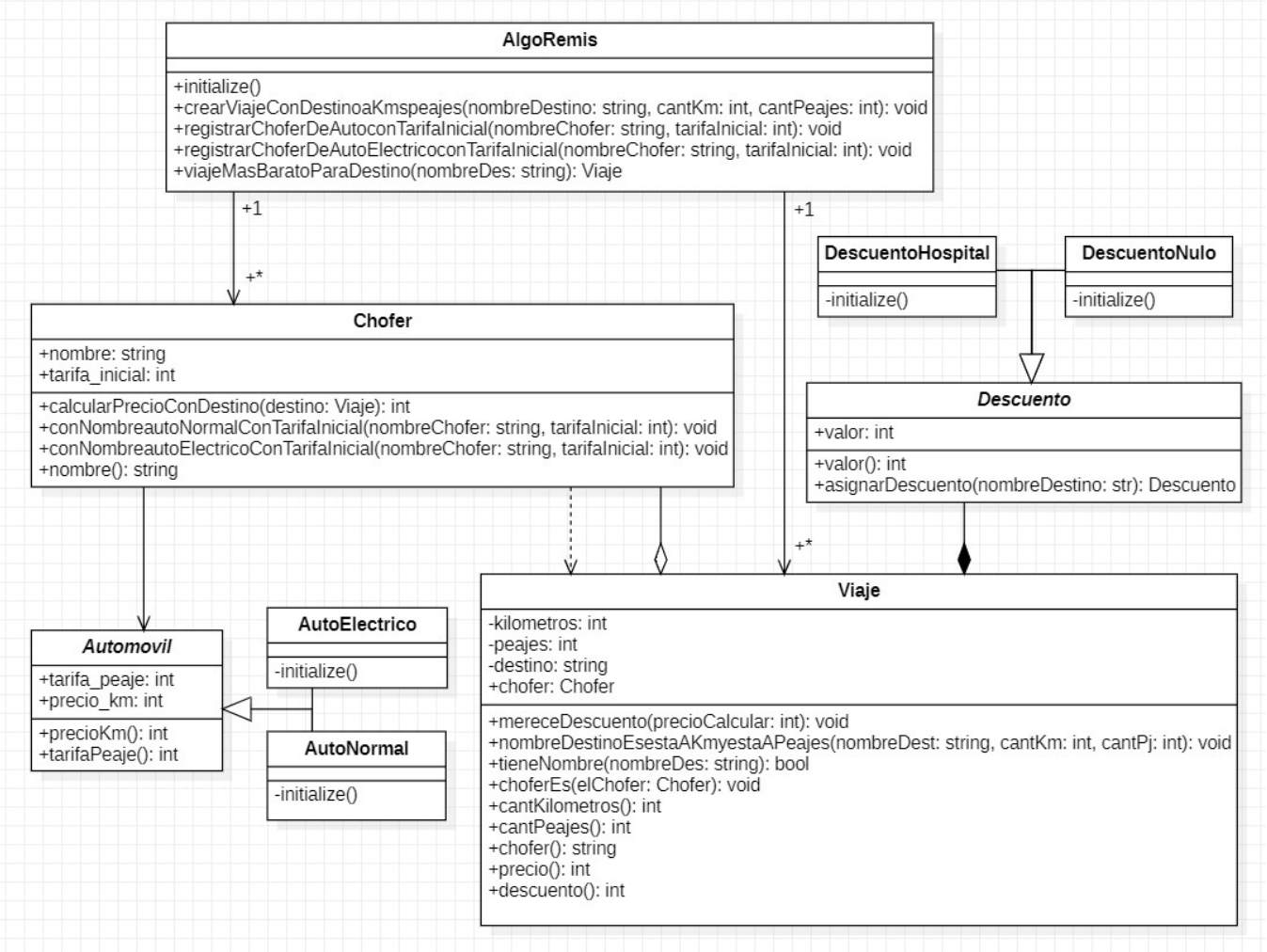
1. Introducción

El presente informe reune la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de un sistema de una agencia de viajes en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Uno de los supuestos a la hora de programar fue que, una vez creado un viaje con destino a, no se le preguntara su precio ni su chofer si todavía no se le fue asignado con el método viajeMasBarato. Otro supuesto importante es que se asume que un chofer puede estar asignado a mas de un viaje, ya que se considera que no lo hará inmediatamente, lo haría uno después de otro si sigue siendo la opción más barata. Por último, se asume que no se llamara al método viajeMasBarato sin tener ningún chofer registrado en la remiseria, lo cual carecería de sentido.

3. Diagramas de clase



Aclaraciones sobre el diagrama de clases

AlgoRemis

Se inicializa con dos ordered collection destinados a contener los distintos objetos Viaje y Chofer que vayamos creando.

Chofer

Es uno de los atributos de Viaje, pero solo es asignado cuando se AlgoRemis llama al método ViajeMasBaratoParaDestino, también es la clase a la que se le delega la responsabilidad de crear su Automóvil (esta una clase abstracta) ya que el objeto que debe crear puede ser un AutoNormal, o un AutoElectrico dependiendo de que método fue llamado por la clase AlgoRemis.

También tiene una relación de agregación con Viaje, ya que en uno de sus métodos, más específicamente en calcularPrecioConDestino, se le puede pasar un Viaje como parámetro para que calcule su precio.

Destino

La clase Viaje es dueño de esta, y lo inicializa en el método nombreDestinoEsestaAKmyestaAPeajes(), en el que se crea una instancia de la clase Descuento, y se le delega la creación del descuento pertinente dependiendo del nombre del destino.

4. Detalles de implementación

La clase madre, por llamarlo de alguna forma, será evidentemente AlgoRemis, que se encargará de crear y almacenar los objetos Chofer y Viaje.
La clase Viaje tiene las siguientes variables:

- Kilómetros (int)
- Peajes (int)
- Destino (string)
- Chofer (Chofer)

Esta es la clase instanciada por AlgoRemis cuando se llama al método crearViajeConDestinoAKmspeajes(), almacena en sus variables estos datos y es un objeto que volverá a ser modificado cuando se llame al método ViajeMasBaratoParaDestino() y será en este momento en el que se le asignara un objeto de la clase Chofer en su variable con el mismo nombre. También es en este método, a la hora de hacer un detect en el ordered collection de destinos que se llama a un método que recibe un string, y devuelve un booleano indicando si ese string es el mismo que el que se encuentra almacenado en la variable destino, esto se hace con el objetivo de no violar el encapsulamiento y el ocultamiento de la información. Cuando se llama al método precio(), este le delega la responsabilidad a la clase Chofer.
La clase Chofer tiene las siguientes variables:

- Nombre (string)
- Auto (*Automóvil*)
- Tarifa_inicial (int)

Cuando se dice automóvil, se refiere al automóvil como una abstracción, ya que se puede tratar de un objeto de clase AutoNormal, o de clase AutoElectrico, dependerá de cual se cree y se almacene en su variable según al método que llame AlgoRemis, y será este mismo método el que se encargue de delegarle la responsabilidad a Chofer.
También es acá donde se ve su relación de agregación con la clase Viaje, ya que al pasarle por parámetro un Viaje a su método calcularPrecioConDestino() devuelve un int con el valor calculado con las variables de Viaje, esta es una decisión puramente de comodidad, ya que quizá lo mas conveniente hubiera sido que este método existiera en la clase Viaje y sea calcularPrecioConChofer() y asi se le pase una variable Chofer y con eso calcule, pero considere que se podria llegar a violar el ocultamiento de información y la clase Viaje ya quedaría muy sobrecargada de mensajes.
Las clases Auto Normal y AutoElectrico son inizializadas con dos únicas variables:

- Tarifa_peaje (int)
- Precio_km (int)

Al implementar esto en una relación de herencia se tiene la ventaja de que, si en el futuro la remiseria quiere agregar choferes de otros tipos de autos, por ejemplo, un auto gasolero, solo se tendría que crear otra clase AutoGasolero con sus variables y agregar sus métodos respectivos a Chofer y AlgoRemis. La desventaja que le encuentro a esta

forma es la necesidad de repetir ciertos métodos, por ejemplo precioKm() y tarifaPeaje() son dos getters que deberán ser necesarios en todas las nuevas clases de tipo "*Automovil*" que se decidan crear en el futuro.
La clase abstracta Descuento posee una única variable, la cual se vera inicializada con un valor dependiendo de cual de las dos se vean instanciadas, esta es:

- Valor (int)

Este valor se trata del porcentaje por el que se deberá multiplicar el precio total para aplicarle el descuento, es por esto que el DescuentoNulo, posee un valor de 1. En cuanto a su implementación cabe remarcar que es importante delegarle a ésta la función de asignar los descuentos al viaje dependiendo de su destino, y asignándole al descuento de Viaje una nueva instancia de la clase Descuento, que pueden ser DescuentoNulo y DescuentoHospital cumpliendo la relación de herencia de "es un", la ventaja de hacer esto es, que si en el futuro se decide darle un descuento a, por ejemplo, la gente que va a un aeropuerto, solo se deberá crear una nueva clase DescuentoAeropuerto, y agregarle esa posibilidad al método asignarDescuento().

5. Excepciones

CantidadInvalida

El motivo de esta excepción es por si tanto los kilómetros como la cantidad de peajes ingresados en el método de la clase AlgoRemis llamado CrearViajeConDestino() son negativos, lo cual carecería de sentido.

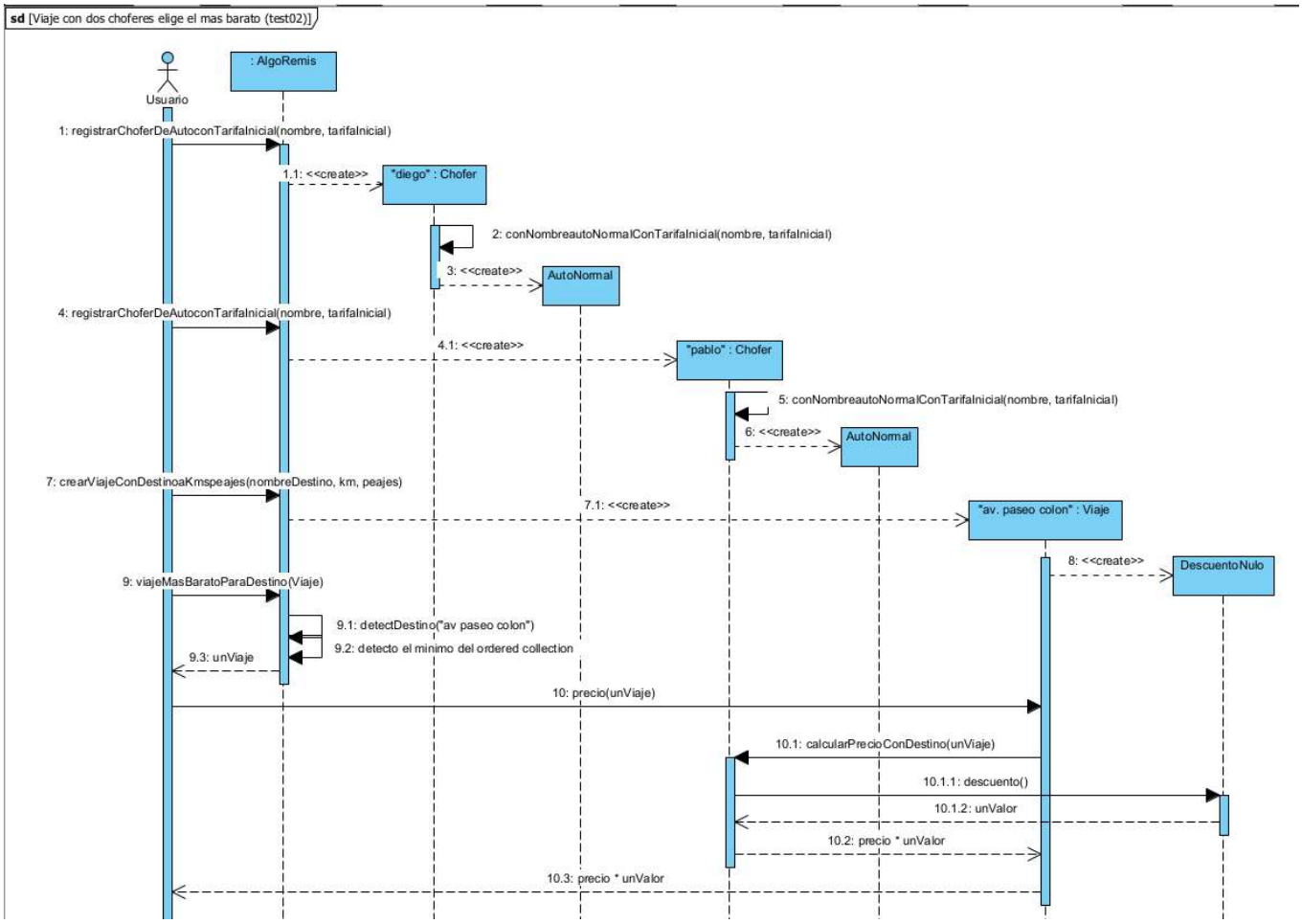
TarifaInvalida

El motivo de esta excepción es por el caso de que la tarifa inicial asignada a un nuevo chofer sea negativa, lo cual carecería de sentido. Esta excepción se encuentra en los métodos registrarChoferDeAuto() y registrarChoferDeAutoElectrico()

DestinoInexistente

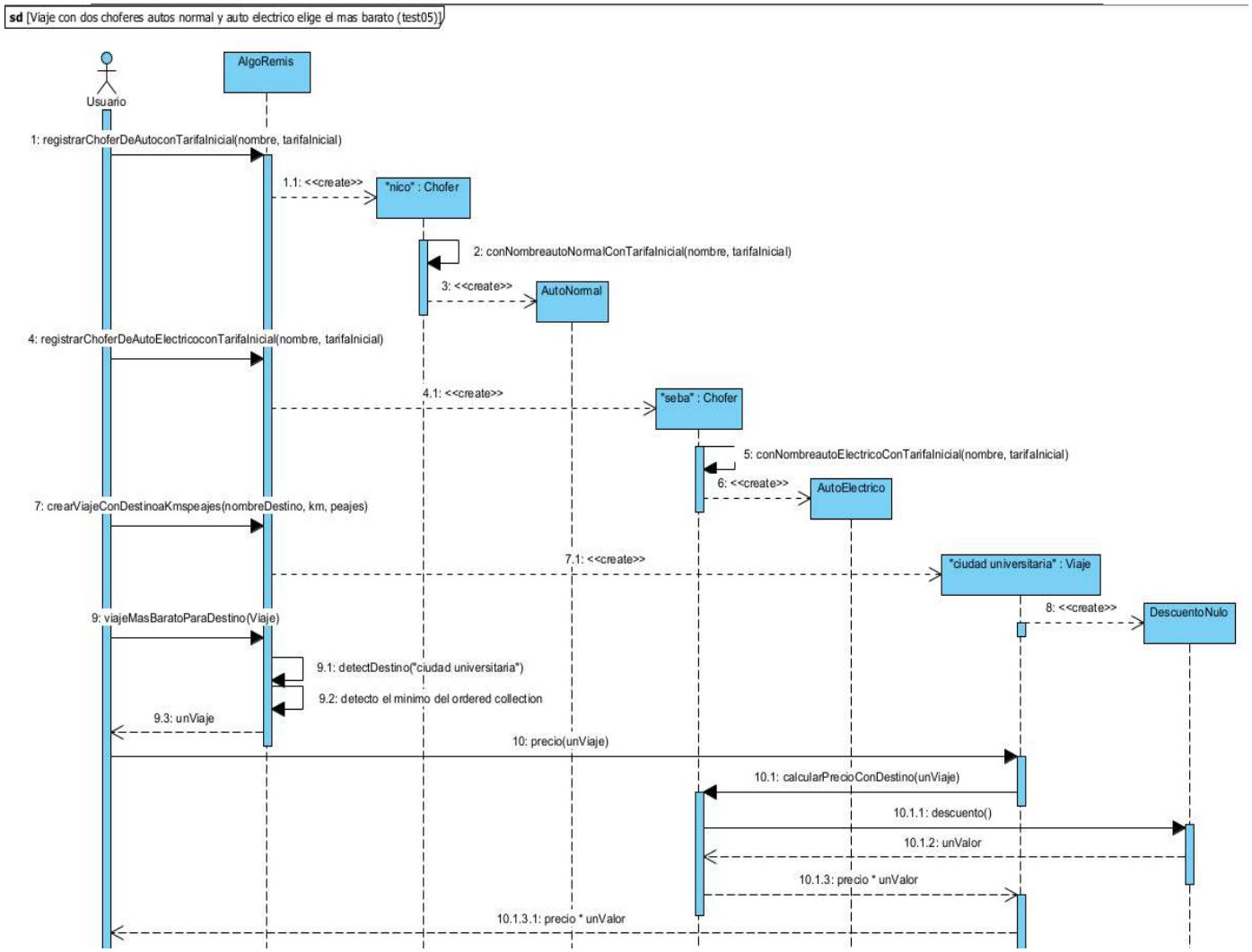
El motivo de esta excepción es para el caso de que, en el método viajeMasBaratoParaDestino() se le pase por parámetro un string que no se pueda vincular con ningún nombre de destino (parámetro de la clase Viaje) perteneciente al ordered collection destinos de AlgoRemis.

6. Diagramas de secuencia

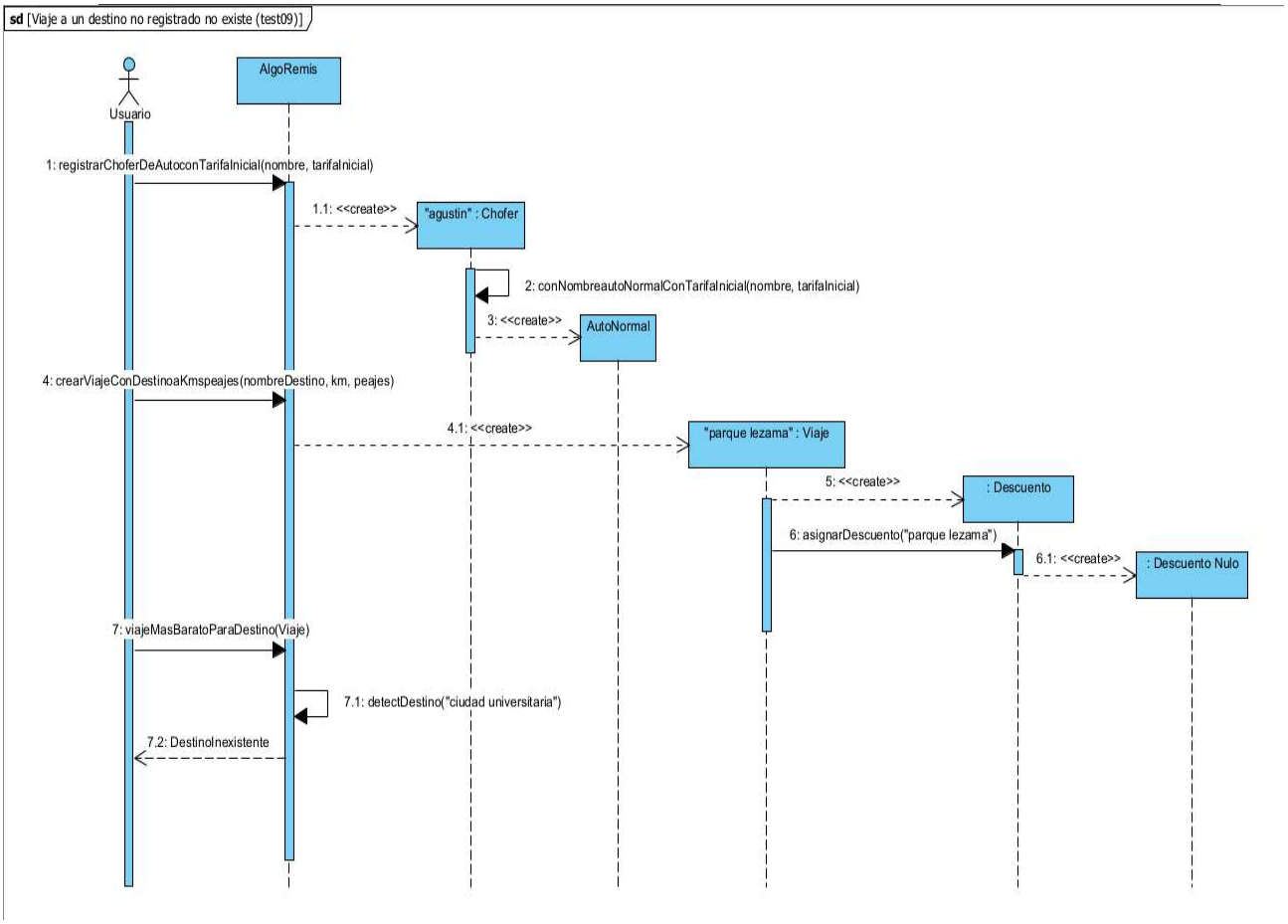


Aclaración sobre este diagrama y el siguiente, en este caso en el paso 8, cuando se crea DescuentoNulo, si se viera mas específicamente quizá se debería diagramar como en el último gráfico, el del test09 en el que incluyo el llamado al método asignarDescuento(), pero decidí obviarlo ya que no es tan relevante los pasos que se llevan a cabo desde que se crea un Descuento hasta que este se asigna como DescuentoNulo y el diagrama se ampliaba demasiado y ya no se vería nítidamente en la imagen, por esto mismo ésta secuencia decidí mostrarle en el ultimo test, que es el de menor tamaño.

En esta figura se representa el test02 pasado por la catedra en el que se crean dos choferes con diferentes tarifas, se crea un nuevo destino y se obtiene el viaje mas barato.
En este y en los siguientes diagramas se presenta una situación, en el caso que nos ocupa entre los puntos 36 y 39. Se trata de que los auto mensajes que puse con la forma detecDestino() no son un parámetro en si, sino que es la manera que se me ocurrió de expresar que estoy haciendo un detect en el ordered collection de “destinos” el cual me daría el Viaje que uso en el paso 42, lo exprese mas explicito en el ejemplo siguiente, en “detecto el mínimo del ordered collection”, en ese paso puse explícitamente que hice, y el resultado de esto me da cual es el chofer al que debo aplicarle el método 42, en este caso, con “pablo”.



En esta figura se representa el test05 pasado por la catedra, es un símil del anterior, pero en este caso, se trata de un chofer de auto eléctrico. Aquí se ve una ventaja general del algoritmo, y es que este no se ve modificado esencialmente por el hecho de que se registre un chofer de un auto diferente. Y como se explicó anteriormente, se podría agregar un nuevo tipo de auto o descuento, y la variación en el algoritmo seguiría siendo similar.



En este test además de mostrar lo que ya exprese en la aclaración anterior, mostre lo que se le pasa al método asignarDescuento() no como "nombreDestino", sino que le pase el nombre del destino en si, en el caso de este ejemplo "parque Lezama"

En este caso se trata de uno de los test realizados por el alumno, específicamente uno en el que se lanza la excepción DestinoInexistente. Se crea un destino a "parque Lezama", pero luego cuando se busca un viaje mas barato para "ciudad universitaria", al no encontrarlo el detect dentro de AlgoRemis se lanza la excepción.