Tarea Integradora 3 - Segunda entrega:

Requerimientos Funcionales:

RF1: Permitir el registro de vehículos, sea un Carro (a Gasolina, Eléctrico o Híbrido) o una Motocicleta, pidiendo para cada uno su marca, modelo, cilindraje, tipo de vehículo (Nuevo o Usado), kilometraje, placa (en caso de ser Usado), precio base, precio del Documento del SOAT, año de este documento, código de este documento (en caso de que el año especificado sea el año actual en el que estamos), monto de cobertura que da este documento, precio del Documento de la Revisión Técnico Mecánica, año de este documento, código de este documento (en caso de que el año especificado sea el año actual en el que estamos) y el nivel de gases liberado que especifica este documento. Cabe recalcar que los documentos de un vehículo se almacenan en una lista simplemente enlazada.

- Para los carros en general se pide además el tipo de carro (Sedan o Van), el número de puertas, y si está polarizado o no. En el caso de los carros a Gasolina, se pide adicionalmente su capacidad del tanque de gasolina y su tipo de gasolina (Extra, Ordinaria o Diesel). En el caso de los carros Eléctricos, se pide adicionalmente su tipo de cargador (Rápido o Normal) y la duración de su batería. En el caso de los carros Híbridos, se piden adicionalmente los atributos que para los carros a Gasolina y para los Eléctricos se pidieron adicional en su caso.
- Para las motocicletas se pide además el tipo de motocicleta (Estándar, Deportiva, Scooter o Cross) y su capacidad del tanque de gasolina.

No obstante, para añadir un vehículo, se verifica antes si la lista de vehículos del concesionario está vacía para añadirlo directamente, o en el caso contrario, se determina a través de su placa (en caso de ser un vehículo Usado) o de su marca, modelo y cilindraje (en caso de ser un vehículo Nuevo) que no exista ya un vehículo en el sistema con esa misma placa o esos mismos tres atributos. Cabe recalcar que a la hora de ser añadido al sistema, se calcula automáticamente su precio total de venta (en base a su precio base, tipo de vehículo y años respectivos de sus documentos SOAT y Revisión Técnico Mecánica), su consumo de gasolina (para carros a Gasolina, Híbridos, y Motocicleta, en base a la capacidad del tanque de gasolina y al cilindraje) y su consumo de batería (para carros Eléctricos e Híbridos, en base al tipo de cargador, duración de la batería y cilindraje).

Por último, si al querer añadir un vehículo, este es un carro usado y de modelo menor a 2015, este además de ser registrado en la lista de vehículos del sistema, se traslada además a un parqueadero de sólo carros viejos usados, representado por una matriz 10 X 5. En efecto, los carros del 2014 sólo pueden guardarse en la columna 1, los del 2013 en la segunda, y así hasta la columna 4; en la última columna (5) se guardan todos los carros con modelo menor a 2011.

RF2: Permitir el registro de personas, sean Empleados o Clientes, pidiendo para cada uno su nombre, apellido y número de identificación. En el caso de los clientes, se

pide además su número de teléfono y correo electrónico. Sin embargo, para añadir una nueva persona, se verifica antes si la lista de personas del concesionario está vacía para añadirla directamente, o en el caso contrario, que no exista ya una persona en el sistema con el mismo ID de la persona que se quiere añadir.

RF3: Buscar eficientemente un cliente dado su número de teléfono. Para esto, primero se verifica si existen clientes registrados en la lista de personas del sistema. Si es así, se crea una lista local y exclusiva de clientes que están registrados en esta lista de personas y ésta se ordena por número de teléfono del cliente ascendente. Luego, utilizando búsqueda binaria se intenta encontrar este cliente en la lista con dicho número de teléfono. Finalmente, el sistema debe indicar el tiempo que tardó la búsqueda hasta encontrarlo para evidenciar la eficiencia de ésta.

RF4: Buscar eficientemente un empleado dado su ID. Para esto, primero se verifica si existen clientes registrados en la lista de personas del sistema. Si es así, se crea una lista local y exclusiva de clientes que están registrados en esta lista de personas y ésta se ordena por ID del cliente ascendente. Luego, utilizando búsqueda binaria se intenta encontrar este cliente en la lista con dicho ID. Finalmente, el sistema debe indicar el tiempo que tardó la búsqueda hasta encontrarlo para evidenciar la eficiencia de ésta.

RF5: Importar datos de un archivo csv con información, sea de vehículos o de personas. Para esto, se le pide al usuario el nombre del archivo que desea importar y qué es lo que quiere importar. Luego, el sistema debe verificar al ser:

- Vehículos: que el o los vehículos que se quieren importar no existan ya en el sistema para que de esta forma puedan ser añadidos correctamente a la lista de vehículos.
- Personas: y éstas al ser:

    ▪ **Empleados:** que el o los eventuales clientes que tengan a cargo existan en el sistema (sino se añade(n) primero este(os) con, a su vez, su(s) eventual(es) vehículo(s) de interés previamente verificado(s) que exista(n) en el sistema también, sino se añade(n) primero este(os)), y si es así, que el o los empleados que se quieren importar no existan ya en el sistema para que de esta forma puedan ser añadidos correctamente a la lista de personas.

    ▪ **Clientes:** que el o los eventuales vehículos de interés de cada cliente existan en el sistema (sino se añade(n) primero este(os)), y si es así, que el o los clientes que se quieren importar no existan ya en el sistema para que de esta forma puedan ser añadidos correctamente a la lista de personas.

RF6: Exportar datos en un archivo csv con información de una especie escogida de vehículos (carros a gasolina, eléctricos, híbridos, o motocicletas), incluyendo todos

los datos de sus documentos. Además, el sistema debe exportarlo ordenadamente por los siguientes dos criterios en este orden: *marca ascendente, y modelo descendente.* Por último, se usa un separador que el usuario especifique para delimitar los atributos entre ellos.

RF7: Guardar toda su información a través de la serialización de sus objetos en archivos binarios, cada vez que se registren vehículos o personas (guardado automático) en sus respectivas listas o estructuras propias creadas. Tiene que existir un path file que especifique el nombre del archivo en el que se almacenará la información guardada de los objetos.

RF8: Cargar toda su información a través de la deserialización de los objetos guardados en archivos binarios (sean de vehículos o personas), cada vez que se ejecuta el programa (carga automática). Cabe recalcar que el sistema verifica primero que existan los archivos binarios de vehículos y personas, para ser deserializados. Si uno de ellos no existe, no se carga la información de éste, y sólo se carga entonces la información de los archivos que existan.

RF9: Desplegar la información principal de los clientes presentes en la lista de personas del sistema, ordenados ascendentemente por su nombre y correo electrónico (si dos nombres coinciden, se "desempata" por el correo electrónico ascendente también). Sin embargo, primero se verifica antes que existan personas de tipo Cliente registradas en el sistema. Si es así, se crea una lista local y exclusiva de clientes que están registrados en esta lista de personas, se procede a ordenar esta lista de clientes por los criterios ya mencionados, para que finalmente el sistema muestre en pantalla la información de todos los clientes registrados en esta lista.

RF10: Desplegar la información principal de los clientes presentes en la lista de personas del sistema, ordenados descendentemente por su apellido y número de teléfono (si dos apellidos coinciden, se "desempata" por el número de teléfono descendente también). Sin embargo, primero se verifica antes que existan personas de tipo Cliente registradas en el sistema. Si es así, se crea una lista local y exclusiva de clientes que están registrados en esta lista de personas, se procede a ordenar esta lista de clientes por los criterios ya mencionados, para que finalmente el sistema muestre en pantalla la información de todos los clientes registrados en esta lista.

RF11: Desplegar la información principal de los empleados presentes en la lista de personas del sistema, ordenados ascendentemente por su apellido y nombre (si dos apellidos coinciden, se "desempata" por el nombre ascendente también). Para esto, primero se verifica que existan personas de tipo Empleado registradas en el sistema. Si es así, se crea una lista local y exclusiva de empleados que están registrados en esta lista de personas, y se procede a ordenar esta lista de empleados por los

criterios ya mencionados, para que finalmente el sistema muestre en pantalla la información de todos los empleados registrados en esta lista.

RF12: Desplegar la información principal de los empleados presentes en la lista de personas del sistema, ordenados descendentemente por su cantidad de ventas realizadas y ID (si dos cantidades de ventas coinciden, se "desempata" por el ID descendente también). Para esto, primero se verifica que existan personas de tipo Empleado registradas en el sistema. Si es así, se crea una lista local y exclusiva de empleados que están registrados en esta lista de personas, se procede a ordenar esta lista de empleados por los criterios ya mencionados, para que finalmente el sistema muestre en pantalla la información de todos los empleados registrados en esta lista.

RF13: Desplegar la información principal de los vehículos presentes en la lista de vehículos del sistema, por especie (Carro a Gasolina, Eléctrico, Híbrido, o Motocicleta) y tipo de vehículo (Usado, Nuevo o ambos tipos a la vez). Sin embargo, primero se verifica antes que existan vehículos registrados en el sistema, y en caso afirmativo, se procede a mostrarlos en pantalla.

RF14: Desplegar la información principal de los carros presentes en el parqueadero de carros viejos usados del concesionario, pidiendo primero el año del modelo de los carros que se quieren consultar. No obstante, primero se verifica antes que existan vehículos registrados en el sistema, y en caso afirmativo, se procede a buscar que haya al menos una especie de carro (a Gasolina, Eléctrico o Híbrido) en el parqueadero. Si es así, se muestra la información de cada uno de ellos en pantalla.

RF15: Permitir la eliminación de un vehículo presente en la lista de vehículos del sistema, sea Usado (a través de su placa) o Nuevo (a través de la coincidencia de 3 atributos: modelo, marca y cilindraje). Sin embargo, se verifica antes que existan vehículos registrados en el sistema, y en caso afirmativo, se busca el vehículo en cuestión a través del o de los atributos respectivo(s), y si se encuentra, se elimina. Aclaración: al momento de vender un vehículo, este no se elimina del sistema para que le quede constancia al usuario en el catálogo de vehículos que este vehículo fue vendido y tiene ahora un dueño.

RF16: Permitir la eliminación de una persona presente en la lista de personas del sistema, sea de tipo Empleado o Cliente. Sin embargo, se verifica antes que existan personas registradas en el sistema, y en caso afirmativo, se busca la persona en cuestión a través de su ID, y si se encuentra, se elimina.

RF17: Asignarle uno o varios vehículos de interés a un cliente, sean Usados o Nuevos. En efecto, este(os) vehículo(s) se almacena(n) en un árbol binario de búsqueda ordenado por kilometraje. No obstante, primero se verifica antes que existan vehículos registrados en la lista de vehículos del sistema; y en caso afirmativo, se

comprueba si existe el vehículo en cuestión en esta lista (para los Usados: con su placa. Para los Nuevos: con la coincidencia de 3 atributos: modelo, marca y cilindraje); y en caso afirmativo, se verifica ahora que existan personas registradas en la lista de personas del sistema; y en caso afirmativo, se comprueba si existe el cliente en cuestión en esta lista; y en caso afirmativo, se procede a verificar (buscar) recursivamente por el árbol binario de búsqueda que no exista el vehículo en cuestión (a través de su kilometraje); y si es así, se añade dicho vehículo también recursivamente a este árbol.

RF18: Asignarle uno o hasta 5 clientes a un empleado para que éste esté a cargo de ellos como potencial vendedor de los vehículos. En efecto, este(os) cliente(s) se almacena(n) en un árbol binario de búsqueda ordenado por número de teléfono. No obstante, primero se verifica antes que existan personas registradas en la lista de personas del sistema; y en caso afirmativo, se comprueba si existen tanto el cliente en cuestión como el empleado en cuestión en esta lista a través de sus ID; y en caso afirmativo, se procede a verificar (buscar) recursivamente por el árbol binario de búsqueda que no exista el cliente en cuestión (a través de su número de teléfono); y si es así, se añade dicho vehículo también recursivamente a este árbol.

RF19: Permitir la venta de un vehículo a un cliente, sea Carro (a Gasolina, Eléctrico o Híbrido) o Motocicleta. Para esto, primero se verifica antes que existan vehículos registrados en la lista de vehículos del sistema; y en caso afirmativo, se comprueba si existe el vehículo en cuestión en esta lista (para los Usados: con su placa. Para los Nuevos: con la coincidencia de 3 atributos: modelo, marca y cilindraje); y en caso afirmativo, se verifica ahora que existan personas registradas en la lista de personas del sistema; y en caso afirmativo, se comprueba si existen tanto el cliente en cuestión como el empleado en cuestión en esta lista a través de sus ID; y en caso afirmativo, se procede a verificar que dicho empleado esté a cargo de dicho cliente (en el caso de los vehículos de interés de este último, sólo si el vehículo que se quiere vender está en el en el árbol binario de búsqueda de vehículos “favoritos” [o de interés], se elimina de ahí a través de su kilometraje. Sino, no es una obligación que esté ahí para venderle el vehículo); y en caso afirmativo, a partir de una matriz 4X4 de enteros y dependiendo de si los años de los documentos SOAT y Revisión técnico mecánica son los actuales o no, se decodifican dos códigos, resultado de recorrer esta matriz en letra L (para el caso del SOAT) o en letra Z (para el caso de la Revisión técnico mecánica), que se guardan cada uno en una cadena caracteres. Finalmente, se debe poder realizar la venta después todo lo anterior; si el vehículo vendido era un carro viejo usado que entonces está presente en el parqueadero, se procede a eliminarlo de ahí a través de su placa.

<u>Adicionalmente, se le dio la posibilidad al programa de:</u>

RF20 extra: Permitir el registro de sedes pidiendo para cada una su nombre, su NIT y su dirección. Estas sedes también tienen como atributos adicionales un número de

ventas y una cantidad total de ganancias. Sin embargo, para añadir una nueva sede, se verifica antes si la lista doblemente enlazada de sedes está vacía para añadirla directamente, o en el caso contrario, que no exista ya una sede en el sistema con el mismo serial de la sede que se quiere añadir.

RF21 extra: Desplegar la información de las sedes presentes en la lista doblemente enlazada de sedes del sistema. Sin embargo, primero se verifica antes que existan sedes registradas en el sistema, y en caso afirmativo, se procede a mostrarlas en pantalla.

RF22 extra: Permitir la eliminación de una sede presente en la lista doblemente enlazada de sedes del sistema. Sin embargo, se verifica antes que existan sedes registradas en el sistema, y en caso afirmativo, se busca la sede en cuestión a través de su NIT, y si se encuentra, se elimina.

RF23 extra: Permitir la decoración de tipo "detalle" del menú de inicio con un ringlete encima del carro del fondo de pantalla, con el objetivo de dar la impresión de movimiento del carro, y por ende del ringlete girando por efecto del viento. Para esto, se utilizarán hilos adicionales al hilo principal (concurrencia).

Requerimientos No Funcionales:

RNF1: El programa debe tener una interfaz gráfica de usuario.
RNF2: El programa debe usar herencia, polimorfismo y desacoplamiento (interfaces).
RNF3: El programa debe ser persistente.
RNF4: El programa debe ser concurrente.
RNF5: El programa debe tener tests (pruebas unitarias).
RNF6: El programa debe manejar excepciones.

Diseño de los escenarios y casos de prueba

Configuración de los Escenarios

Nombre	Clase	Escenario
setup1	EmployeeTest	Un objeto de la clase Employee es creado con id = "22-11".
setup2	EmployeeTest	Un objeto de la clase Employee es creado con id = "11-22".

setup1	ClientTest	Un objeto de la clase Employee es creado con id = “397368-45456”.
setup1	SoatTest	Un objeto de la clase Soat es creado con priceDoc = 932443, year = 0, codeDoc = “”, y coverageAmount = 83921.
setup2	SoatTest	Un objeto de la clase Soat es creado con priceDoc = 932443, year = 2019, codeDoc = “94583257” y coverageAmount = 83921.
setup1	ReviewTest	Un objeto de la clase Review es creado con priceDoc = 128345, year = 0, codeDoc = “” y gasLevel = 0.
setup2	ReviewTest	Un objeto de la clase Review es creado con priceDoc = 128345, year = 2018, codeDoc = “94583257” y gasLevel = 37824.
setup1	GasolineTest ElectricTest HybridTest MotorcycleTest	Un objeto de la clase Gasoline/Electric/Hybrid/Motorcycle es creado con basePrice = 57849439, typeVehicle = ‘U’. Año del SOAT: 2019. Año de la Revisión técnico mecánica: 2019.
setup2	GasolineTest ElectricTest HybridTest MotorcycleTest	Un objeto de la clase Gasoline/Electric/Hybrid/Motorcycle es creado con basePrice = 57849439, typeVehicle = ‘U’. Año del SOAT: 2018. Año de la Revisión técnico mecánica: 2020.
setup3	GasolineTest ElectricTest HybridTest MotorcycleTest	Un objeto de la clase Gasoline/Electric/Hybrid/Motorcycle es creado con basePrice = 57849439, typeVehicle = ‘U’. Año del SOAT: 2020. Año de la Revisión técnico mecánica: 2017.
setup4	GasolineTest ElectricTest HybridTest MotorcycleTest	Un objeto de la clase Gasoline/Electric/Hybrid/Motorcycle es creado con basePrice = 57849439, typeVehicle = ‘U’. Año del SOAT: 2020. Año de la Revisión técnico mecánica: 2020.
setup5	GasolineTest ElectricTest HybridTest MotorcycleTest	Un objeto de la clase Gasoline/Electric/Hybrid/Motorcycle es creado con basePrice = 57849439, typeVehicle = ‘N’. Año del SOAT: 0. Año de la Revisión técnico mecánica: 0.
setup6	GasolineTest HybridTest MotorcycleTest	Un objeto de la clase Gasoline/Hybrid/Motorcycle específicamente con cylinder = 4736 y capacityGasoline = 34682.
setup6 (Electric) setup7 (Hybrid)	ElectricTest HybridTest	Un objeto de la clase Electric/Hybrid específicamente con cylinder = 4736, typeCharger = ‘F’ y durationBattery = 55.
setup7 (Electric) setup8	ElectricTest HybridTest	Un objeto de la clase Electric/Hybrid específicamente con cylinder = 8769, typeCharger = ‘N’ y durationBattery = 44.

(Hybrid)		
setup1	RingletTest	Un objeto de la clase Ringlet es creado.
setup2	RingletTest	Un objeto de la clase Ringlet es creado y se añade un objeto de la clase Square con rotation = 1 a la lista de cuadrados de la clase Ringlet.
setup1	ListHeadquartersTest	Un objeto de la clase ListHeadquarters es creado.
setup2	ListHeadquartersTest	Un objeto de la clase ListHeadquarters es creado y se añade un objeto de la clase Headquarter a la lista enlazada de sedes del sistema, específicamente uno con nit = "109238-32111"
setup3	ListHeadquartersTest	Un objeto de la clase ListHeadquarters es creado y se añaden dos objetos de la clase Headquarter a la lista enlazada de sedes del sistema, específicamente uno con nit = "109238-32111" y otro con nit = "767432-43551"
setup1	BSTClientsInChargeTest	Un objeto de la clase BSTClientsInCharge es creado.
setup2	BSTClientsInChargeTest	Un objeto de la clase BSTClientsInCharge y también de Client es creado (con phone = “33758217”), y éste último se añade al árbol binario de búsqueda que almacena los clientes que tiene a su cargo un empleado.
setup3	BSTClientsInChargeTest	Un objeto de la clase BSTClientsInCharge es creado y se añaden 5 objetos de Client al árbol binario de búsqueda que almacena los clientes que tiene a su cargo un empleado, con ID’s diferentes.
setup4	BSTClientsInChargeTest	Un objeto de la clase BSTClientsInCharge es creado y se añaden dos objetos de la clase Client al árbol binario de búsqueda que almacena los clientes que tiene a su cargo un empleado, específicamente uno con phone = "109238-32111" y otro con phone = "767432-43551"
setup1	BSTFavoriteVehiclesTest	Un objeto de la clase BSTFavoriteVehicles es creado.
setup2	BSTFavoriteVehiclesTest	Un objeto de la clase BSTFavoriteVehicles y también uno de Gasoline o Electric o Hybrid o Motorcycle es creado (con cylinder = 65329), y éste último se añade al árbol binario de búsqueda de vehículos de interés de un cliente.
setup3	BSTFavoriteVehiclesTest	Un objeto de la clase BSTFavoriteVehicles es creado y se añaden dos objetos de la clase Gasoline o Electric o Hybrid o Motorcycle al árbol binario de búsqueda que almacena los vehículos de interés de un cliente, específicamente uno con cylinder = 65329 y otro con cylinder = 86734.
setup1	CompanyTest	Una lista de Person se crea y se le añaden 2 objetos Employee y uno Client.

setup2	CompanyTest	Una lista de Person se crea y se le añaden 2 objetos Client y uno Employee.
setup3	CompanyTest	Una lista de Vehicle se crea y se le añaden 4 objetos Gasoline (2 usados y 2 nuevos) y un objeto Electric, Hybrid y Motorcycle, cada uno.
setup4	CompanyTest	Una lista de Vehicle se crea y se le añaden 4 objetos Electric (2 usados y 2 nuevos) y un objeto Gasoline, Hybrid y Motorcycle, cada uno.
setup5	CompanyTest	Una lista de Vehicle se crea y se le añaden 4 objetos Hybrid (2 usados y 2 nuevos) y un objeto Gasoline, Electric y Motorcycle, cada uno.
setup6	CompanyTest	Una lista de Vehicle se crea y se le añaden 4 objetos Motorcycle (2 usados y 2 nuevos) y un objeto Gasoline, Electric y Hybrid, cada uno.
setup7	CompanyTest	Un objeto de la clase Company es creado.
setup8	CompanyTest	En el sistema existe: un cliente, un vehículo, un empleado, y este último está a cargo de dicho cliente.
setup9	CompanyTest	En el sistema existe: un cliente, un vehículo, un empleado, y este último **no** está a cargo de dicho cliente.
setup10	CompanyTest	En el sistema existe: un vehículo y un empleado.
setup11	CompanyTest	En el sistema existe: un vehículo y un cliente.
setup12	CompanyTest	En el sistema existe: un empleado y un cliente.
setup13	CompanyTest	En el sistema existen 2 carros a gasolina (uno con model = 2014 y otro con model = 2013), otros 2 eléctricos (uno con model = 2012 y otro con model = 2011) y otros 2 híbridos (uno con model = 2010 y otro con model = 2009).

Diseño de los Casos de Prueba

Objetivo de la Prueba: Verificar que al comparar criterios (atributos) de tipo String (o convertidos a String para este fin) de un objeto de la clase Employee, devuelve correctamente un número positivo si es mayor lexicográficamente, negativo si es menor lexicográficamente, o igual a 0 si son iguales lexicográficamente. En el método únicamente se comparan la cantidad total de ventas y el ID del empleado. Para esto, visto que para probar esta acción da lo mismo usar cualquiera de estos atributos mencionados para verificar que devuelva el entero adecuado, sólo se usó el **ID del empleado** como ejemplo (para objetivo de ordenamiento descendente).

Clase	Método	Escenario	Valores de Entrada	Resultado

Employee	compareTo	setup1	exampleCurrentID = “11-22”	Número positivo El resultado de idEmployee del objeto de tipo Employee comparado al atributo actual exampleCurrentID dio correctamente un entero positivo al ser lexicográficamente “22-11” mayor que “11-22”.
Employee	compareTo	setup2	exampleCurrentID = “22-11”	Número negativo El resultado de idEmployee del objeto de tipo Employee comparado al atributo actual exampleCurrentID dio correctamente un entero negativo al ser lexicográficamente “11-22” menor que “22-11”.
Employee	compareTo	setup2	exampleCurrentID = “11-22”	Número 0 El resultado de idEmployee del objeto de tipo Employee comparado al atributo actual exampleCurrentID dio correctamente 0 al ser lexicográficamente “11-22” igual a “11-22”.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un objeto de la clase Employee con todo el contenido y separador definido al invocar el método toString de esta clase.

Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Employee	toString	setup2	separator = “;”	Se muestran todos los datos del objeto de la clase Employee que se creó con los valores pasados por parámetro al constructor.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un objeto de la clase Client con todo el contenido y separador definido al invocar el método toString de esta clase.

Clase	Método	Escenario	Valores de Entrada	Resultado
Client	toString	setup1	separator = “;”	Se muestran todos los datos del objeto de la clase Client que se creó con los valores pasados por parámetro al constructor.

Objetivo de la Prueba: Verificar que se decodifica aleatoriamente un código de 8 dígitos de un objeto de la clase Soat, a partir de una matriz cuadrada de tamaño 4, luego de recorrerla en L. Esto ocurre al momento de vender un vehículo nuevo, o usado pero con el documento vencido.

Clase	Método	Escenario	Valores de Entrada	Resultado
Soat	decodeCode	setup1	ninguno	Se decodifica correctamente el código al comprobar que una cadena que representa un código estando primero vacía (por ser un vehículo nuevo), ahora es de longitud 8.
Soat	decodeCode	setup2	Se crea un objeto de la clase Soat con: priceDoc = 932443 year = 2019 codeDoc = “94583257” coverageAmount = 83921	Se decodifica correctamente el código al comprobar que una cadena no vacía con un código vencido (el vehículo es usado y el año de su documento SOAT no es el actual) de longitud 8, ahora es diferente al nuevo código.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un objeto de la clase Soat con todo el contenido y separador escogido al invocar el método toString de esta clase.

Clase	Método	Escenario	Valores de Entrada	Resultado
Soat	toString	setup2	separator = “;”	Se muestran todos los datos del objeto de la clase Soat que se creó con los valores pasados por parámetro al constructor.

Objetivo de la Prueba: Verificar que se decodifica aleatoriamente un código de 8 dígitos de un objeto de la clase Review, a partir de una matriz cuadrada de tamaño 4, luego de recorrerla en Z. Esto ocurre al momento de vender un vehículo nuevo; o usado pero con el documento vencido.

Clase	Método	Escenario	Valores de Entrada	Resultado
Review	decodeCode	setup1	ninguno	Se decodifica correctamente el código al comprobar que una cadena que representa un código estando primero vacía (por ser un vehículo nuevo), ahora es de longitud 8.
Review	decodeCode	setup2	ninguno	Se decodifica correctamente el código al comprobar que una cadena no vacía con un código vencido (el vehículo es usado y el año de su documento de la Revisión técnico mecánica no es el actual) de longitud 8, ahora es diferente al nuevo código.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un objeto de la clase Review con todo el contenido y separador escogido al invocar el método toString de esta clase.

Clase	Método	Escenario	Valores de Entrada	Resultado
Review	toString	setup2	separator = ";"	Se muestran todos los datos del objeto de la clase Review que se creó con los valores pasados por parámetro al constructor.

Objetivo de la Prueba: Verificar que se calcula correctamente el precio total de un vehículo, sea carro a gasolina, eléctrico híbrido, o una motocicleta, dependiendo del tipo de vehículo (nuevo o usado) y del año de sus dos documentos (SOAT y Revisión técnico mecánica.

Clase	Método	Escenario	Valores de Entrada	Resultado
Gasoline Electric Hybrid Motorcycle	getTotalPrice	setup1	ninguno	Se calcula correctamente el precio esperado de acuerdo con los valores de entrada.

Gasoline Electric Hybrid Motorcycle	getTotalPrice	setup2	ninguno	Se calcula correctamente el precio esperado de acuerdo con los valores de entrada.
Gasoline Electric Hybrid Motorcycle	getTotalPrice	setup3	ninguno	Se calcula correctamente el precio esperado de acuerdo con los valores de entrada.
Gasoline Electric Hybrid Motorcycle	getTotalPrice	setup4	ninguno	Se calcula correctamente el precio esperado de acuerdo con los valores de entrada.
Gasoline Electric Hybrid Motorcycle	getTotalPrice	setup5	ninguno	Se calcula correctamente el precio esperado de acuerdo con los valores de entrada.

Objetivo de la Prueba: Verificar que se calcula correctamente el consumo de gasolina de un carro a gasolina, híbrido, o una motocicleta, dependiendo de su cilindraje y su capacidad del tanque de gasolina.

Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Gasoline Hybrid Motorcycle	calculateConsumeGasoline	setup6	ninguno	Se calcula correctamente el consumo de gasolina esperado de acuerdo con los valores de entrada.

Objetivo de la Prueba: Verificar que se calcula correctamente el consumo de batería de un carro eléctrico o uno híbrido dependiendo de su tipo de cargador, duración de la batería y cilindraje.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Electric Hybrid	calculateConsumeBattery	setup6 (Electric) setup7 (Hybrid)	ninguno	Se calcula correctamente el consumo de batería esperado de acuerdo con los valores de entrada.
Electric Hybrid	calculateConsumeBattery	setup7 (Electric) setup8 (Hybrid)	ninguno	Se calcula correctamente el consumo de batería esperado de acuerdo con los valores de entrada.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un objeto de la clase Gasoline con todo el contenido y separador escogido al invocar el método toString de esta clase.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Gasoline	toString	ninguno	Se crea un objeto de la clase Gasoline con atributos inventados cualesquiera. separator = “;”	Se muestran todos los datos del objeto de la clase Gasoline que se creó con los valores pasados por parámetro al constructor.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un objeto de la clase Electric con todo el contenido y separador escogido al invocar el método toString de esta clase.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Electric	toString	ninguno	Se crea un objeto de la clase Electric con atributos inventados cualesquiera. separator = “;”	Se muestran todos los datos del objeto de la clase Electric que se creó con los valores pasados por parámetro al constructor.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un objeto de la clase Hybrid con todo el contenido y separador escogido al invocar el método toString de esta clase.

Clase	Método	Escenario	Valores de Entrada	Resultado
Hybrid	toString	ninguno	Se crea un objeto de la clase Hybrid con atributos inventados cualesquiera. separator = ";"	Se muestran todos los datos del objeto de la clase Hybrid que se creó con los valores pasados por parámetro al constructor.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un objeto de la clase Motorcycle con todo el contenido y separador escogido al invocar el método toString de esta clase.

Clase	Método	Escenario	Valores de Entrada	Resultado
Motorcycle	toString	ninguno	Se crea un objeto de la clase Motorcycle con atributos inventados cualesquiera. separator = ";"	Se muestran todos los datos del objeto de la clase Motorcycle que se creó con los valores pasados por parámetro al constructor.

Objetivo de la Prueba: Verificar que se añada correctamente un cuadrado (clase Square) a la lista de cuadrados de la clase Ringlet.

Clase	Método	Escenario	Valores de Entrada	Resultado
Ringlet	addSquare	setup1	rotation = 1	Se agregó correctamente un nuevo cuadrado con rotation = 1 a la lista de cuadrados de Ringlet. La lista de cuadrados ahora tiene tamaño igual a 1.

Objetivo de la Prueba: Verificar que se rote correctamente un cuadrado (clase Square) de la lista de cuadrados de la clase Ringlet. **Aclaración:** el que rota realmente es rotate() de la clase Square; la clase Ringlet sólo lo invoca con rotateSquares() sin más.

Clases	Métodos	Escenario	Valores de Entrada	Resultado
Ringlet Square	rotateSquares rotate (llamado)	setup2	ninguno	Se rotó correctamente el cuadrado de la lista de

				cuadrados de Ringlet, al cambiar su atributo rotation.

Objetivo de la Prueba: Verificar que se añada correctamente o no una sede a la lista enlazada de sedes del sistema.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
ListHeadquarters	addHeadquarter	setup1	Se crea un nodo de la clase Headquarter con atributos inventados cualesquiera.	Se agregó correctamente una nueva sede a la lista enlazada de sedes del sistema. Esta lista de ahora tiene un primer nodo (first != null).
ListHeadquarters	addHeadquarter	setup2	Se crea un nodo de la clase Headquarter con un NIT idéntico al que se creó en el setup.	No se agregó correctamente una nueva sede a la lista enlazada de sedes del sistema porque ya existía una sede con NIT igual. Esta lista de sigue teniendo un primer nodo *first* solamente.

Objetivo de la Prueba: Verificar que la búsqueda en la lista enlazada de sedes del sistema retorne las sedes buscadas cuando están presentes en esta lista y null cuando no se encuentran.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
ListHeadquarters	searchHeadquarter	setup3	nit = "109238-32111"	Se ha encontrado correctamente un objeto de la clase Headquarter con el mismo NIT pasado por parámetro al método de búsqueda.
ListHeadquarters	searchHeadquarter	setup3	nit = "88113"	Null El NIT pasado por parámetro no existe en la lista enlazada.

Objetivo de la Prueba: Verificar que se elimine una sede de la lista enlazada de sedes del sistema, cuando dicha sede se encuentra a través de su NIT.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
ListHeadquarters	removeHeadquarter	setup3	nit = "109238-32111"	Se ha encontrado y eliminado correctamente un objeto de la clase Headquarter con el mismo NIT pasado por parámetro al método de eliminación. Ahora el único nodo presente en esta lista doblemente enlazada se encuentra en su primer nodo (first).
ListHeadquarters	removeHeadquarter	setup3	nit = "88113"	No se ha encontrado y por ende tampoco eliminado correctamente un objeto de la clase Headquarter con el NIT pasado por parámetro al método de eliminación porque éste NIT no existe en la lista enlazada

Objetivo de la Prueba: Verificar que se añada correctamente o no un cliente al árbol binario de búsqueda que almacena los clientes que tiene a su cargo un empleado. **Aclaración:** el que asigna un cliente a un empleado realmente es addAssignedClient() de la clase BSTClientsInCharge; la clase Company sólo lo invoca con toAssignClient() sin más.				
Clases	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
BSTClientsInCharge Company (llama)	addAssignedClient toAssignClient (llamado)	setup1	Se crea un nodo de la clase Client con atributos inventados cualesquiera.	Se agregó correctamente un nuevo cliente al árbol binario de búsqueda porque no existía uno con *phone* igual. Este árbol binario de búsqueda ahora tiene un primer nodo *root* (root != null).

BSTClientsInCharge Company (llama)	addAssignedClient toAssignClient (llamado)	setup2	Se crea un nodo de la clase Client con un phone idéntico al que se creó en el setup.	No se agregó correctamente un nuevo cliente al árbol binario de búsqueda porque ya existía uno con *phone* igual. Este árbol binario de búsqueda sigue teniendo un primer nodo *root* solamente.
BSTClientsInCharge Company (llama)	addAssignedClient toAssignClient (llamado)	setup3	Se crea un nodo de la clase Client con un phone diferente a los que se crearon el el setup.	No se agregó correctamente un nuevo cliente al árbol binario de búsqueda porque el empleado ya estaba atendiendo a 5 clientes (número máximo de clientes que puede atender). Le salta la excepción **WorkloadException** en el método de la clase Company. Este árbol binario de búsqueda sigue teniendo 5 nodos solamente.

Objetivo de la Prueba: Verificar que la búsqueda en el árbol binario de búsqueda que almacena los clientes que tiene a su cargo un empleado, retorne los clientes buscados cuando están presentes en este árbol y null cuando no se encuentran.

Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
BSTClientsInCharge	searchClientInCharge	setup4	phone = "109238-32111"	Se ha encontrado correctamente un objeto de la clase Client con el mismo *phone* pasado por parámetro al método de búsqueda.
BSTClientsInCharge	searchClientInCharge	setup4	phone = "88113"	Null El *phone* pasado por parámetro no existe en el árbol

				binario de búsqueda.

Objetivo de la Prueba: Verificar que se elimine un cliente del árbol binario de búsqueda que almacena los clientes que tiene a su cargo un empleado, cuando dicho cliente se encuentra a través de su número de celular.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
BSTClientsInCharge	removeClientsInCharge	setup3	phone = "(+57) 301 332-9878"	Se ha encontrado y eliminado correctamente un objeto de la clase Client con el mismo celular pasado por parámetro al método de eliminación. Ahora este BST tiene 4 nodos.
BSTClientsInCharge	removeClientsInCharge	setup3	phone = "88113"	No se ha encontrado y por ende tampoco eliminado correctamente un objeto de la clase Client con el celular pasado por parámetro al método de eliminación porque éste celular no existe en el árbol binario de búsqueda.
BSTClientsInCharge	removeClientsInCharge	setup4	phone = "109238-32111"	Se ha encontrado y eliminado correctamente un objeto de la clase Client con el mismo celular pasado por parámetro al método de eliminación. Ahora el único nodo presente en este

				BST se encuentra en la raíz.
BSTClientsInCharge	removeClientsInCharge	setup4	phone = " 88113"	No se ha encontrado y por ende tampoco eliminado correctamente un objeto de la clase Client con el celular pasado por parámetro al método de eliminación porque éste celular no existe en el árbol binario de búsqueda.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un nodo de tipo Client situado en el árbol binario de búsqueda que almacena los clientes que tiene a su cargo un empleado, con todo el contenido y separador definido.

Clase	Método	Escenario	Valores de Entrada	Resultado
BSTClientsInCharge	printClientsInCharge	setup2	ninguno	Se muestran todos los datos del nodo de tipo Client presente en el árbol, con los valores pasados por parámetro al constructor de Client.

Objetivo de la Prueba: Verificar que se añada correctamente o no un vehículo al árbol binario de búsqueda que almacena vehículos de interés de un cliente. **Aclaración:** el que asigna un vehículo de interés a un cliente realmente es addFavoriteVehicle() de la clase BSTFavoriteVehicles; la clase Company sólo lo invoca con toAssignVehicleNew() **ó** toAssignVehicleUsed() sin más.

Clase	Método	Escenario	Valores de Entrada	Resultado
BSTFavoriteVehicles Company (llama)	addFavoriteVehicle toAssignVehicleNew (lo llama si es vehículo nuevo)	setup1	Se crea un nodo de la clase Gasoline o Electric o Hybrid o	Se agregó correctamente un nuevo vehículo al árbol binario de búsqueda porque no existía uno con cylindraje igual. Este árbol binario de búsqueda ahora

	ó toAssignVehicleUsed (lo llama si es vehículo usado)		Motorcycle, con atributos inventados cualesquiera.	tiene un primer nodo *root* (root != null).
BSTFavoriteVehicles Company (llama)	addFavoriteVehicle toAssignVehicleNew (lo llama si es vehículo nuevo) **ó** toAssignVehicleUsed (lo llama si es vehículo usado)	setup2	Se crea un nodo de la clase Gasoline o Electric o Hybrid o Motorcycle, con un cilindraje idéntico al que se creó en el setup.	No se agregó correctamente un nuevo vehículo al árbol binario de búsqueda porque ya existía uno con cylindraje igual. Le salta la excepción **FavoriteVehicleException** en el método de la clase Company. Este árbol binario de búsqueda sigue teniendo un primer nodo *root* solamente.

Objetivo de la Prueba: Verificar que la búsqueda en el árbol binario de búsqueda que almacena vehículos de interés de un cliente, retorne los vehículos buscados cuando están presentes en este árbol y null cuando no se encuentran (a través de su cilindraje).

Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
BSTFavoriteVehicles	searchFavoriteVehicle	setup3	cylinder = 65329	Se ha encontrado correctamente un objeto de la clase Gasoline o Electric o Hybrid o Motorcycle, con el mismo cilindraje pasado por parámetro al método de búsqueda.
BSTFavoriteVehicles	searchFavoriteVehicle	setup3	cylinder = 21234	Null El cilindraje pasado por parámetro no existe en el árbol binario de búsqueda.

Objetivo de la Prueba: Verificar que se elimine un vehículo del árbol binario de búsqueda que almacena vehículos de interés de un cliente, cuando dicho vehículo se encuentra a través de su cilindraje.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
BSTFavoriteVehicles	removeFavoriteVehicle	setup3	cylinder = 65329	Se ha encontrado y eliminado correctamente un objeto de la clase Gasoline o Electric o Hybrid o Motorcycle, con el mismo cilindraje pasado por parámetro al método de eliminación. Ahora el único nodo presente en este BST se encuentra en la raíz.
BSTFavoriteVehicles	removeFavoriteVehicle	setup3	cylinder = 21234	No se ha encontrado y por ende tampoco eliminado correctamente un objeto de la clase Gasoline o Electric o Hybrid o Motorcycle, con el cilindraje pasado por parámetro al método de eliminación porque éste cilindraje no existe en el árbol binario de búsqueda.

Objetivo de la Prueba: Verificar que se despliega correctamente toda la información de un nodo de tipo Gasoline o Electric o Hybrid o Motorcycle, situado en el árbol binario de búsqueda que almacena vehículos de interés de un cliente, con todo el contenido y separador definido.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**

BSTFavoriteVehicles	printVehiclesOfInterest	setup2	ninguno	Se muestran todos los datos del nodo de tipo Gasoline o Electric o Hybrid o Motorcycle, presente en el árbol con los valores pasados por parámetro al constructor del vehículo respectivo.

Objetivo de la Prueba: Verificar que se retorna una lista de empleados llenada a partir de los empleados presentes en la lista de personas del sistema, ordenada descendentemente por cantidad total de ventas y ID.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	showEmployeesByTotalSalesAndID	setup1	Se crea manualmente la lista de empleados que se espera obtener.	Se retorna correctamente una lista de empleados de tamaño 2, ordenada descendentemente por cantidad total de ventas y ID del empleado.

Objetivo de la Prueba: Verificar que se retorna una lista de empleados llenada a partir de los empleados presentes en la lista de personas del sistema, ordenada ascendentemente por apellido y nombre.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	showEmployeesByLastNameAndName	setup1	Se crea manualmente la lista de empleados que se espera obtener.	Se retorna correctamente una lista de empleados de tamaño 2, ordenada ascendentemente por apellido y nombre.

Objetivo de la Prueba: Verificar que se retorna una lista de clientes llenada a partir de los clientes presentes en la lista de personas del sistema, ordenada descendentemente por apellido y celular.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	showClientsByLastNameAndPhone	setup2	Se crea manualmente la lista de clientes que se espera obtener.	Se retorna correctamente una lista de clientes de tamaño 2, ordenada descendentemente por apellido y celular.

Objetivo de la Prueba: Verificar que se retorna una lista de clientes llenada a partir de los clientes presentes en la lista de personas del sistema, ordenada ascendentemente por nombre y correo electrónico.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	showClientsByNameAndEmail	setup2	Se crea manualmente la lista de clientes que se espera obtener.	Se retorna correctamente una lista de clientes de tamaño 2, ordenada ascendentemente por nombre y correo electrónico.

Objetivo de la Prueba: Verificar que se retorna una lista de carros a gasolina por tipo de vehículo (usado o nuevo) que se especifique, llenada a partir de los vehículos presentes en la lista de vehículos del sistema.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	showCarsGasoline	setup3	Tipo de vehículo: ‘U’ (usados) Se crea manualmente la lista de carros a gasolina usados que se espera obtener.	Se retorna correctamente una lista de carros a gasolina usados de tamaño 2.
Company	showCarsGasoline	setup3	Tipo de vehículo: ‘N’ (nuevos)	Se retorna correctamente una lista de carros a

			Se crea manualmente la lista de carros a gasolina nuevos que se espera obtener.	gasolina nuevos de tamaño 2.
Company	showCarsGasoline	setup3	Tipo de vehículo: ‘B’ (ambos) Se crea manualmente la lista de carros a gasolina de ambos tipos que se espera obtener.	Se retorna correctamente una lista de carros a gasolina de ambos tipos de tamaño 4.

Objetivo de la Prueba: Verificar que se retorna una lista de carros eléctricos por tipo de vehículo (usado o nuevo) que se especifique, llenada a partir de los vehículos presentes en la lista de vehículos del sistema.

Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	showCarsElectric	setup4	Tipo de vehículo: ‘U’ (usados) Se crea manualmente la lista de carros eléctricos usados que se espera obtener.	Se retorna correctamente una lista de carros eléctricos usados de tamaño 2.
Company	showCarsElectric	setup4	Tipo de vehículo: ‘N’ (nuevos) Se crea manualmente la lista de carros eléctricos nuevos que se espera obtener.	Se retorna correctamente una lista de carros eléctricos nuevos de tamaño 2.
Company	showCarsElectric	setup4	Tipo de vehículo: ‘B’ (ambos) Se crea manualmente la lista de carros eléctricos de ambos tipos que se espera obtener.	Se retorna correctamente una lista de carros eléctricos de ambos tipos de tamaño 4.

Objetivo de la Prueba: Verificar que se retorna una lista de carros híbridos por tipo de vehículo (usado o nuevo) que se especifique, llenada a partir de los vehículos presentes en la lista de vehículos del sistema.

Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**

Company	showCarsHybrid	setup5	Tipo de vehículo: ‘U’ (usados) Se crea manualmente la lista de carros híbridos usados que se espera obtener.	Se retorna correctamente una lista de carros híbridos usados de tamaño 2.
Company	showCarsHybrid	setup5	Tipo de vehículo: ‘N’ (nuevos) Se crea manualmente la lista de carros híbridos nuevos que se espera obtener.	Se retorna correctamente una lista de carros híbridos nuevos de tamaño 2.
Company	showCarsHybrid	setup5	Tipo de vehículo: ‘B’ (ambos) Se crea manualmente la lista de carros híbridos de ambos tipos que se espera obtener.	Se retorna correctamente una lista de carros híbridos de ambos tipos de tamaño 4.

Objetivo de la Prueba: Verificar que se retorna una lista de motocicletas por tipo de vehículo (usado o nuevo) que se especifique, llenada a partir de los vehículos presentes en la lista de vehículos del sistema.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	showMotorcycles	setup6	Tipo de vehículo: ‘U’ (usados) Se crea manualmente la lista de motocicletas usadas que se espera obtener.	Se retorna correctamente una lista de motocicletas usadas de tamaño 2.
Company	showMotorcycles	setup6	Tipo de vehículo: ‘N’ (nuevos) Se crea manualmente la lista de motocicletas nuevas que se espera obtener.	Se retorna correctamente una lista de motocicletas nuevas de tamaño 2.
Company	showMotorcycles	setup6	Tipo de vehículo: ‘B’ (ambos) Se crea manualmente la lista de motocicletas de	Se retorna correctamente una lista de motocicletas de

			ambos tipos que se espera obtener.	ambos tipos de tamaño 4.

Objetivo de la Prueba: Verificar que se elimine una persona de la lista de personas del sistema, cuando se encuentra a través de su ID.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	removePerson	setup1	ID idéntico al de una persona que se haya creado en el setup.	Se ha encontrado y eliminado correctamente la persona con el mismo ID pasado por parámetro al método de eliminación. Ahora la lista es de tamaño 2.
Company	removePerson	setup1	ID que no lo tenga ninguna de las personas que se haya creado en el setup.	No se ha encontrado y por ende tampoco eliminado correctamente la persona con el ID pasado por parámetro al método de eliminación porque éste ID no existe la lista de personas. La lista sigue de tamaño 3.

Objetivo de la Prueba: Verificar que se elimine un vehículo usado de la lista de vehículos del sistema, cuando se encuentra a través de su placa.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	removeVehicleWithLicensePlate	setup3	Placa idéntica al de un vehículo que se haya creado en el setup.	Se ha encontrado y eliminado correctamente el vehículo con la misma placa pasada por parámetro al método de eliminación. La lista ahora es de tamaño 6.
Company	removeVehicleWithLicensePlate	setup3	Placa que no la tenga ninguno de los vehículos que se hayan creado en el setup.	No se ha encontrado y por ende tampoco eliminado correctamente el vehículo

				con la placa pasada por parámetro al método de eliminación porque ésta placa no existe la lista de vehículos. La lista sigue de tamaño 7.
Company	removeVehicleWithLicensePlate	setup7	Se crea y agrega un vehículo usado específicamente con model = 2014, a la lista de vehículos del sistema.	Se ha encontrado y eliminado correctamente el vehículo con la misma placa pasada por parámetro al método de eliminación. Además, como estaba en el parqueadero, también se eliminó de ahí. La lista ahora es de tamaño 0.

Objetivo de la Prueba: Verificar que se elimine un vehículo nuevo de la lista de vehículos del sistema, cuando se encuentra a través de la coincidencia de 3 atributos en un vehículo: marca modelo y cilindraje.

Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	removeVehicleWithoutLicensePlate	setup3	Marca, modelo y cilindraje idénticos al de un vehículo que se haya creado en el setup.	Se ha encontrado y eliminado correctamente el vehículo con la misma marca, modelo y cilindraje pasados por parámetro al método de eliminación. La lista ahora es de tamaño 6.

Company	removeVehicleWithoutLicensePlate	setup3	Marca, modelo y cilindraje que no los tenga ninguno de los vehículos que se hayan creado en el setup.	No se ha encontrado y por ende tampoco eliminado correctamente el vehículo con la marca, modelo y cilindraje pasados por parámetro al método de eliminación porque éstos atributos no existen en la lista de vehículos. La lista sigue de tamaño 7.

Objetivo de la Prueba: Verificar que se añada un empleado nuevo a la lista de personas del sistema, cuando no se encuentra otro con el mismo ID.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	addEmployee	setup1	Se crea un objeto Employee con ID diferente a los que se crearon en el setup.	Se agregó correctamente el empleado. La lista ahora es de tamaño 4.
Company	addEmployee	setup1	Se crea un objeto Employee con ID igual al de otro empleado que se creó en el setup.	No se agregó correctamente el empleado porque ya exisitía otro con el mismo ID. La lista sigue de tamaño 3.

Objetivo de la Prueba: Verificar que se añada un cliente nuevo a la lista de personas del sistema, cuando no se encuentra otro con el mismo ID.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	addClient	setup1	Se crea un objeto Client con ID	Se agregó correctamente el cliente. La lista ahora es de tamaño 4.

			diferente a los que se crearon en el setup.	
Company	addClient	setup1	Se crea un objeto Client con ID igual al de otro empleado que se creó en el setup.	No se agregó correctamente el cliente porque ya exisitía otro con el mismo ID. La lista sigue de tamaño 3.

Objetivo de la Prueba: Verificar que se añada un vehículo (Carro a gasolina, eléctrico o híbrido, o una Motocicleta) a la lista de vehículos del sistema, cuando no se encuentra otro con la misma placa (vehículo usados) o por la coincidencia de 3 atributos: marca, modelo y cilindraje (vehículo nuevos).

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	addVehicle	setup3	Se crea un objeto Gasoline/Electric/Hybrid/Motorcycle con placa diferente a los que se crearon en el setup, o con uno de los mencionados 3 atributos diferente a los de los que se crearon en el setup.	Se agregó correctamente el vehículo. La lista ahora es de tamaño 8.
Company	addVehicle	setup3	Se crea un objeto Gasoline/Electric/Hybrid/Motorcycle con placa igual al de otro vehículo que se creó en el setup, o con los 3 atributos mencionados, idénticos a los de uno que se creó en el setup.	No se agregó correctamente el vehículo porque ya exisitía otro con la misma placa, o con los mismos 3 atributos mencionados. La lista sigue de tamaño 7.
Company	addVehicle	setup7	Se crean y agregan 11 objetos de la clase Vehicle con model = 2014 y typeVehicle = 'U'.	Se agregan correctamente los 10 primeros vehículos pero el onceavo no porque el parqueadero se llenó para los carros viejos usados de modelo 2014. Le salta la excepción **LackOfLandException**. La lista sigue de tamaño 10.

Objetivo de la Prueba: Verificar que se importen datos de un archivo csv, sea de cualquier especie de vehículo, de clientes, o de empleados.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	importData	setup7	Nombre del archivo con la información correspondiente a importar. El archivo tiene n líneas para n objetos y éstos tienen datos diferentes a los que estén registrados en el sistema en sus respectivas listas. data = 1 o 2 o 3 o 4 o 5 o 6	Se importó correctamente la información porque no existía ningún objeto con un parámetro de identificación igual al de los importados. Ahora la respectiva lista de objetos implicada aumentó a tamaño n porque son n líneas para n objetos.
Company	importData	setup7	Nombre del archivo con la información correspondiente a importar. El archivo tiene n líneas para n objetos y éstos tienen datos iguales a los que ya estaban registrados en el sistema en sus respectivas listas. data = 1 o 2 o 3 o 4 o 5 o 6	No se importó correctamente la información porque ya existían objetos con un parámetro de identificación igual al de los importados. La respectiva lista de objetos implicada sigue con el mismo tamaño que tenía antes de importar.

Objetivo de la Prueba: Verificar que se exporten datos de cualquier especie de vehículo (carros a gasolina, eléctricos, híbridos, o motocicletas) a un archivo csv. El archivo tiene n líneas para n objetos, y están separados por un carácter escogido por el usuario.

Clase	Método	Escenario	Valores de Entrada	Resultado
Company	exportDataVehicles	setup7	Nombre del archivo donde irá la información	Se exportó la información de la especie de vehículo especificada. Ahora en la

			correspondiente a exportar. separator = “;” vehicle = ‘G’ o ‘E’ o ‘H’ o ‘M’.	carpeta “data” existe un arhivo csv con el respectivo nombre escogido.

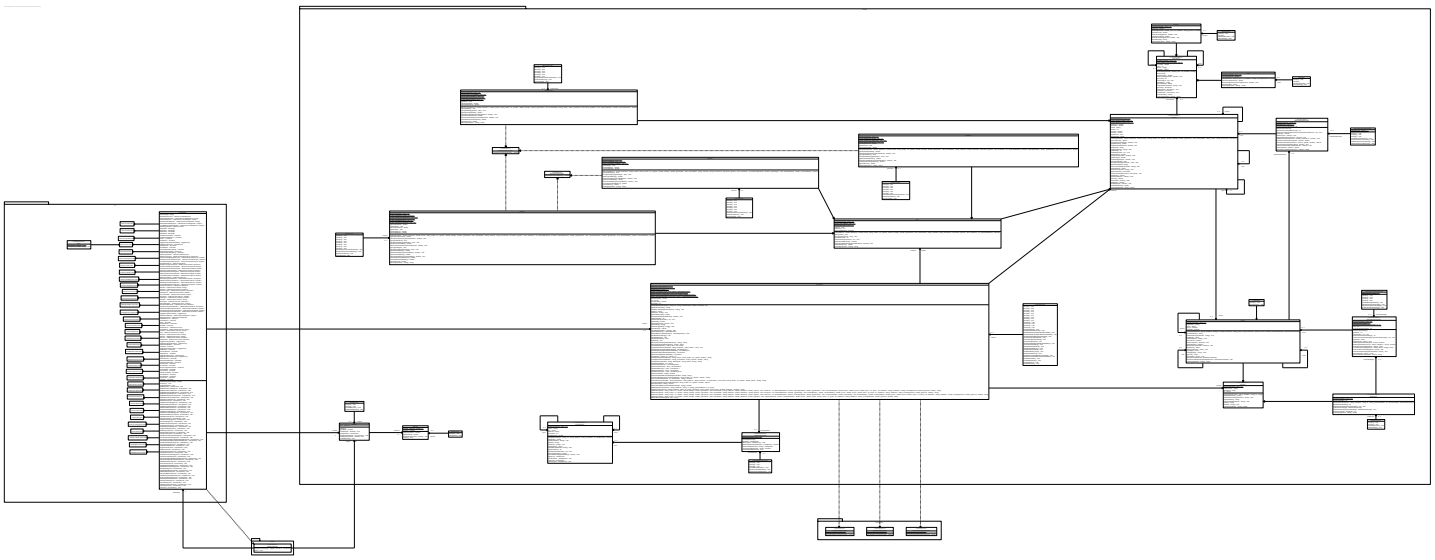
Objetivo de la Prueba: Verificar que se venda correctamente un vehículo.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	sellAVehicle	setup8	Valores inventados cualesquiera para las variables: licensePlate, idEmployee, idClient, selection, typeVehicle, brand, model, cylinder, phone.	Se vendió correctamente el vehículo al cliente. Ahora el vehículo tiene un dueño. Se actualizan: el número de ventas de la compañía, la cantidad total de ganancias de ésta, y la cantidad total de ventas del cliente, todo con respecto a los valores de los atributos inventados de los objetos: Vehicle, Employee y Client.
Company	sellAVehicle	setup9	Valores inventados cualesquiera para las variables: licensePlate, idEmployee, idClient, selection, typeVehicle, brand, model, cylinder, phone.	No se vendió correctamente el vehículo al cliente porque el empleado especificado no está a cargo del cliente especificado.
Company	sellAVehicle	setup10	Valores inventados cualesquiera para las variables: licensePlate, idEmployee, idClient, selection, typeVehicle, brand, model, cylinder, phone.	No se vendió correctamente el vehículo al cliente porque no existe el cliente especificado en el sistema.
Company	sellAVehicle	setup11	Valores inventados cualesquiera para las variables:	No se vendió correctamente el vehículo al cliente porque no existe el empleado especificado en el sistema.

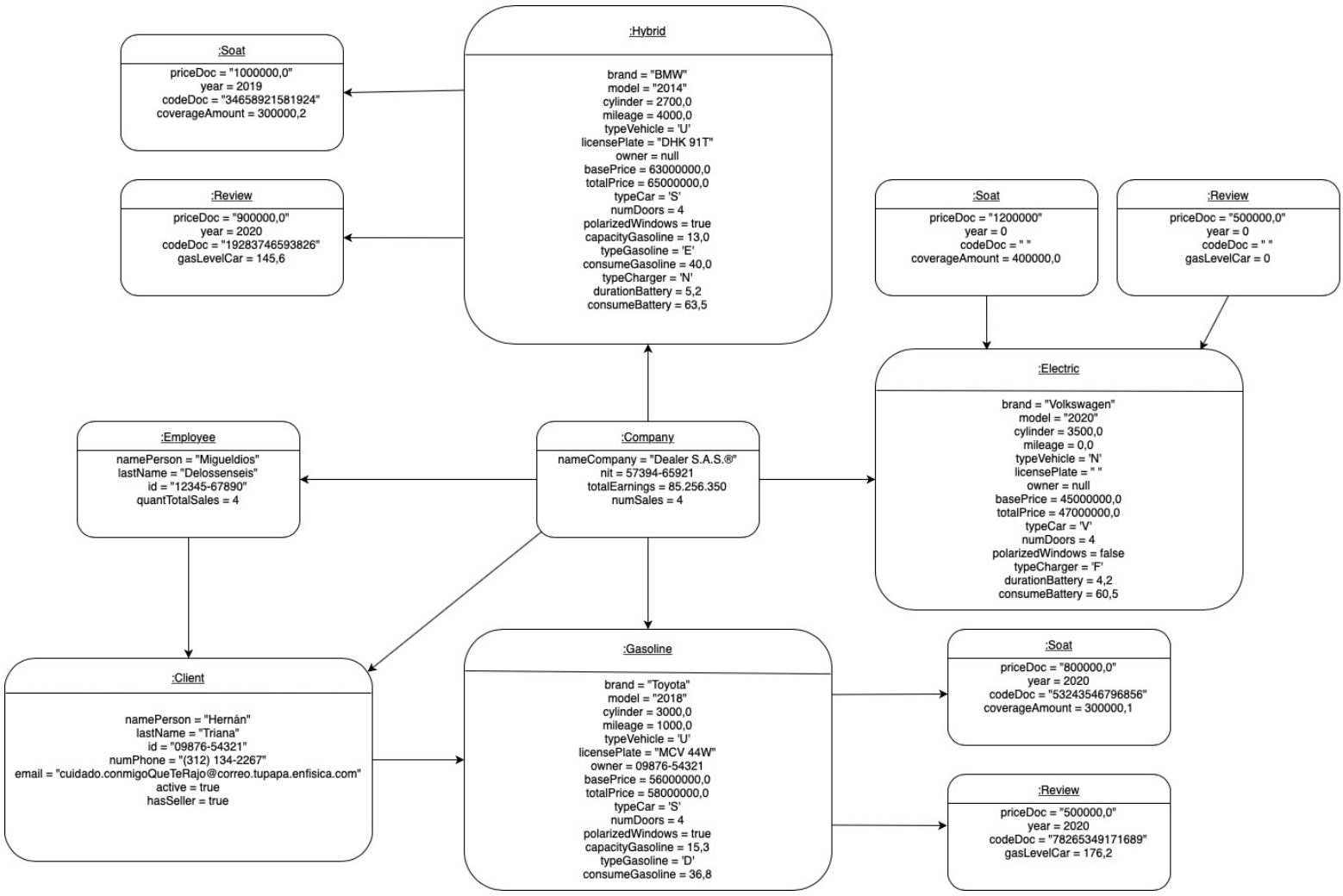
			licensePlate, idEmployee, idClient, selection, typeVehicle, brand, model, cylinder, phone.	
Company	sellAVehicle	setup12	Valores inventados cualesquiera para las variables: licensePlate, idEmployee, idClient, selection, typeVehicle, brand, model, cylinder, phone.	No se vendió correctamente el vehículo al cliente porque no existe el vehículo especificado en el sistema.

Objetivo de la Prueba: Verificar que se retorna una lista de carros viejos usados de un modelo especificado menor a 2015, llenada a partir de los carros presentes en la matriz que representa el parqueadero del concecionario.

Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Company	lookCarsParking	setup13	model = 0 Se crea manualmente la lista de carros viejos usados que se espera obtener.	Se retorna correctamente una lista de carros viejos usados de modelo 2014. Esta lista es de tamaño 1.
Company	lookCarsParking	setup13	model = 1 Se crea manualmente la lista de carros viejos usados que se espera obtener.	Se retorna correctamente una lista de carros viejos usados de modelo 2013. Esta lista es de tamaño 1.
Company	lookCarsParking	setup13	model = 2 Se crea manualmente la lista de carros viejos usados que se espera obtener.	Se retorna correctamente una lista de carros viejos usados de modelo 2012. Esta lista es de tamaño 1.
Company	lookCarsParking	setup13	model = 3	Se retorna correctamente una lista de carros

			Se crea manualmente la lista de carros viejos usados que se espera obtener.	viejos usados de modelo 2011. Esta lista es de tamaño 1.
Company	lookCarsParking	setup13	model = 4 Se crea manualmente la lista de carros viejos usados que se espera obtener.	Se retorna correctamente una lista de carros viejos usados de modelos menores a 2011. Esta lista es de tamaño 2.




:Soat
priceDoc = "1000000,0"
year = 2019
codeDoc = "34658921581924"
coverageAmount = 300000,2
:Hybrid
brand = "BMW"
model = "2014"
cylinder = 2700,0
mileage = 4000,0
typeVehicle = 'U'
licensePlate = "DHK 91T"
owner = null
basePrice = 63000000,0
totalPrice = 65000000,0
typeCar = 'S'
numDoors = 4
polarizedWindows = true
capacityGasoline = 13,0
typeGasoline = 'E'
consumeGasoline = 40,0
typeCharger = 'N'
durationBattery = 5,2
consumeBattery = 63,5
:Review
priceDoc = "900000,0"
year = 2020
codeDoc = "19283746593826"
gasLevelCar = 145,6
:Soat
priceDoc = "1200000"
year = 0
codeDoc = " "
coverageAmount = 400000,0
:Review
priceDoc = "500000,0"
year = 0
codeDoc = " "
gasLevelCar = 0
:Electric
brand = "Volkswagen"
model = "2020"
cylinder = 3500,0
mileage = 0,0
typeVehicle = 'N'
licensePlate = " "
owner = null
basePrice = 45000000,0
totalPrice = 47000000,0
typeCar = 'V'
numDoors = 4
polarizedWindows = false
typeCharger = 'F'
durationBattery = 4,2
consumeBattery = 60,5
:Employee
namePerson = "Migueldios"
lastName = "Delossenseis"
id = "12345-67890"
quantTotalSales = 4
:Company
nameCompany = "Dealer S.A.S.®"
nit = 57394-65921
totalEarnings = 85.256.350
numSales = 4
:Client
namePerson = "Hernán"
lastName = "Triana"
id = "09876-54321"
numPhone = "(312) 134-2267"
email = "cuidado.conmigoQueTeRajo@correo.tupapa.enfisica.com"
active = true
hasSeller = true
:Gasoline
brand = "Toyota"
model = "2018"
cylinder = 3000,0
mileage = 1000,0
typeVehicle = 'U'
licensePlate = "MCV 44W"
owner = 09876-54321
basePrice = 56000000,0
totalPrice = 58000000,0
typeCar = 'S'
numDoors = 4
polarizedWindows = true
capacityGasoline = 15,3
typeGasoline = 'D'
consumeGasoline = 36,8
:Soat
priceDoc = "800000,0"
year = 2020
codeDoc = "53243546796856"
coverageAmount = 300000,1
:Review
priceDoc = "500000,0"
year = 2020
codeDoc = "78265349171689"
gasLevelCar = 176,2