**Trabajo Práctico Integrador**

**(Reentrega)**

**1er Cuatrimestre 2020 – Programación 2 (Com04)**

**Tema: Empresa alimenticia “La Santafesina”.**

***Alumna:*** *Galiano, Lucrecia Solange,*

***Legajo:*** 34643570/2015,

***Email:*** lucrecia.lv@gmail.com

***Nuevo informe:***

En esta Reentrega lo que comencé fue a corregir en primer lugar los TAD de Tranporte y los vehículos, como: en el TAD transporte lo realice fue cambiarla de abstracta a una public y a los TAD Flete, TAD CamionTrailerComun y TAD CamionMegaTrailer lo que les hice fue dejarles solo los setters, getters y métodos propios de su TAD. Pero los métodos generales (osea que uso en todos) los uni en el TAD Transporte y luego para comodidad los TAD de cada vehículo los uni en una sola clase(la clase TAD Vehiculo).

Como se puede ver:

En el TAD Transporte:

TENEMOS: los atributos Identificacion, CargaMaxima (peso), CargaMaximaOriginal (peso), CapacidadMaxima (volumen), CapacidadMaximaOriginal (volumen), Refrigeracion, CostoPorKmDeViaje, seguroCarga, Destino, tipo, kmDestino, enViaje y paquete.

* En cada caso se inicia en el contructor.
* Luego con lo setters lo que se hace es verificar que cada atributo al ser iniciado cumpla con los IREP.
* En el caso de los String: que se carguen datos distintos de null y que en el caso de la identificación que respeten sean realmente 7 caracteres,
* En el caso de los double que sean positivos,
* En el caso de los booleanos no los realice ya que no son necesario, y
* y en el caso de los paquetes utilice LinkedList ya que me pareció mas practico para cargar los paquetes por llos métodos que trae ya en su biblioteca.
* Los getters los realice porque son necesarios para usar estos datos en los TAD de Vehiculo ya que como los atributos los genere en private (que como ya sabemos este solo nos permite modificar las variables solo dentro de la clase).
* En su gran mayoría los getters están relacionados con los setter pero hay otros como Tipo()que fueron solo generados en el constructor y no necesita set ya que lo genere porque no supe como utilizar el getClass() y que me supiera decir realmente el vehiculo de que tipo era y viajando() es otro que lo genere para saber si un vehuculo esta o no en viaje.
* En los métodos solucione el que no se cargaban los paquetes cuando no hay mas lugar, inicie los transportes (Flete y los camiones como me dijo con los new en el constructor)
* En el método agregarPaquete(String destino, double volumen, double peso, boolean refrigeracionP) {}

Lo que hace es tomar los datos de un paquete, lo inicia como tal y lo agrega o no según si hay espacio y el transporte tiene la refrigeración que el paquete necesita. Y retorna un booleano confirmando la carga.

* En el método quitarPaquete(double peso, double volumen, String destino, boolean refrigeracionP) {}

Lo que se realiza es verificar que el paquete exista, de ser así el primero en aparecer se quita. Y con un booleano garantiza la eliminación.

* En el método cantPaquetes() {} Lo que hace es con el mismo método del LinkedList tomar en número la cantidad de paquetes que se tienen en el momento de llmado el método
* En el asignarDestino(String destino, Integer km) {} lo que se hace es verificar que no haya paquete en el transporte ni destino, sino lanza exception. Y si no ocurre eso se carga con normalidad.
* Este método smplemente luego dependiendo de que vehiculo sea dique se gasto en el vehiculo.
* En el método activarViaje() {} Lo que se hace es verificar que el transporte no este en viaje que tenga paquetes y que no tenga destino.
* En el método desactivarViaje() {} lo que hace es vericar en caso que el micro este ya activo lanza exception y sino lo activa, y restablese el transporte.
* Luego tenemos PaquetesIguales(LinkedList <Paquete> paquetes2) {) que es utiizado en el toString para verificar mas eficientemente la igualgad.
* En el método equals(Object obj) {} De este caso esta acorder a lo pedido en el tp, solo se compara destino, tipo, paquetesy refrigeración.
* Y finalmente realice el toString() {} que lo que hace es ir tomando todos los dato de cada transporte y lo convierte en un string y lo hace

En el TAD Vehiculo encontraremos

# TAD CamionTrailerComun

* Donde en el constructor CamionTrailerComun(String id, double carga,double capacidad, boolean refri,double costoXkm,double seguro) {} Se llama al super del contructor de Transporte y se inicializa.
* Los métodos son mismos los del TAD Transporte, al igual que el to String.

# TAD CamionMegaTrailer

* Los atributos propios de esta clase son pasajeros, CostoFijo y comidaCamionero.
* En el constructor iniciamos los atributos propios del CamionMegaTrailer y se llama al super de transporte.
* Los setters: en este caso son iguales pero cada variable tiene el suyo y en ambas que son doubles lo que se hace es verificar que se ingresen número positivos.
* Los gettters solos se estos mismos setter del TAD
* Los métodos en su mayoría los mismos los del TAD Transporte, a excepción del método CostoFinalal que en este caso se le suma CostoFijo + comidaCamionero

# TAD Flete

* Los atributos propios de esta clase son pasajeros(LinkedList) y costoPasaje.
* El constructor Flete(String id, double carga, double capacidad, double costoXkm, int acompaniantes, double costoPasaje) {} Lo que se hace es inicializar con el super y generar la cantidad de pasajeros y inicializar el costo del pasaje.
* Los setters simplemente generan
* La cantidad de pasajeros y crea lo lugares en la lista.
* El setCostoPasaje(double costoP){} lo que hace es controlas que el costo sea positivo no negativo.
* Los getters serian los que tiene en el TAD Transporte y los propios de este TAD:
* getPasajeros(){} que devuelve un LinkedList de los pasajeros del Flete.
* getCostoPasaje() {} representa el costo por pasaje individual.
* Los métodos también tiene los propios del TAD Transporte y los de este mismo TAD que son:
* cantAcompañantes() {} que retorna la cantidad de acompañantes.
* Y el método costoFinal() {} que toma del super el método costoFinal y le suma (cantAcompañantes() por getCostoPasaje()).
* Y en el toString() {} toma el toString() de super y le agrega los pasajeros y el costo de cada pasaje

TAD Paquete {

* Los atributos de este TAD son Peso, Volumen, Destino y RefrigeracionP.
* El constructor del Paquete(double peso, double volumen, String dest, boolean refrigeracionP) {} Inicia todos los atributos utilizando como hice con todos los TAD usando los setters exceptuando los boolean que no los usan a los setters.
* Los setters de peso y volumen verifican que los atributos sean iniciados con double que sean números positivos.
* El set de Destino verifica que se ingrese realmente un dato del tipo string con dato no vacio.
* Y finalmente cada setter tiene su getter correspondiente.
* El equals en este caso compara ambos paquetes uno u otro son iguales a null automáticamente no son iguales. Y después compara el destino, peso, refrigeración y volumen (todos los datos) si son iguales por ende son idénticos.
* En el toString lo que se hace es recopilar ordenada y detalladamente cada dato para que quede mas bonitos a la vista del cliente.

TAD Deposito

* Los atributos de este TAD son CapacidadMaximaOriginal, CapacidadMaxima(volumenMaximo ), Propio, Refrigeracion, paquete(LinkedList ), CostoPorTonelada, destino y carga.
* El este caso tenemos 2 constructores de
* El primero es Deposito(double capacidad, boolean tieneRefrigeracion, boolean propio) {} Inicializa cada atributo.
* El segundo es Deposito(double capacidad, boolean tieneRefrigeracion, boolean propio,double costoPorTonelada) {}
* En cada contructor en este caso se inicia para un deposito frio o uno que se debe generar totalmente desde si es frio o no, su capacidad, si es propio o no y la cantidad de paquetes los genere con LinkedList, carga(es un dato que genere para que se guarde la carga de paquetes que se genera por cada carga de transporte)
* Los setters de costoXtonelada y CapacidadMaxima lo que hacen es como la gran mayoría de los setters de doubles verificar que se carguen datos de numero positivos.
* Los getters de todos muestran los atributos para poder ser utilizados en los otros TAD.
* Los métodos
* Tenemos cargaCostoPorTonelada() {} Lo que hace es que al ser llamada verifica la carga que se le hizo a un determinado transporte y lo multiplica por el costo por tonelada CostoFinal() lo que se hace es llamar a la función cargaCostoPorTonelada que seria lo el costo que se genera en los depósitos.
* cantPaquetes() Dice la cantidad de paquetes que tiene el deposito.
* AgregarPaquete(String destino, double volumen, double peso, boolean refrigeracionP) {} verifica que haya espacio para cargarlo y que tenga la misma refrigeración que el paquete a ingresar luego de no ser asi se devuelve un boolean false y no pasa nada. Pero si si hay espacio se crea el paquete y se lo carga a los paquetes.
* QuitarPaquete(double peso, double volumen, String destino, boolean refrigeracionP) {} lo que hace es devolver un boolean dependiendo si lo pudo o no borrar.
* CargarPaquetes(Transporte t) {} Lo que hace es tomar un transporte y con el deposito ir cargándole paquetes hasta que no se puedan quitar mas del deposito o que el transporte se llene, depende de lo que suceda primero. Y a medida que se va cargand el transporte se va acumulando en dos variables, una es la que devuelve el método y la otra es la que es usada para verificar cuanto se gano por tonelada cargaada a los depósitos.
* PaquetesIguales(LinkedList <Paquete> paquetes2) {} Se utiliza en el toString para verificar que todos los paquetes de ambos depósitos sean idénticos.
* El equals(Object obj){} verifica que sean totalmente iguales 2 depositos en vace a todos los atributos que generamos.
* El toString() {} Muestra prolijamente al cliente el deposito.

En los TAD TestTP y TAD empresaSantaFe colo que lo que estaba en los enunciados 1era parte y 2da parte.

En el TAD Empresa que es el que manipula y conecta todos los TADs anteriores de la siguiente forma:

* Este tiene los siguientes atributos CUIT, Nombre, depósitos(LinkedList), camionesTrailer(Map), camionMegaTrailer(Map), fletes(Map), transportes(Map) y destinos(Map). En estos casos use Map i no LinkedList porque me servia para que fuera mas practica la búsqueda por id o en el caso de los destino por destinos y no kms.
* El constructor Empresa(String cuit,String nombre){} inicia todos los atributos.
* Los setters garantizan que los datos del CUIT y Nombre sean realmente datos que no sean nulos.
* Los getters muestran el CUIT y el Nombre
* Los métodos
* incorporarPaquete(String destino, double peso, double volumen, boolean frio) {} Lo que hace es tomar todos los depósitos y verificar uno a uno los depósitos su espacio libre y su refrigeración el que coincida con lo que necesita el paquete lo carga y se retorna un true de lo contrario se sigue recorriendo hasta el final de los depósitos y de lo contrario retorna un false.
* agregarDestino(String destino, int km) {} Verifica que no se haya cargado antes ya el destino, en ese caso de ser asi lo carga a un deposito, de lo contrario no o carga a ningún lado.
* agregarDepTercerizFrio(double capacidad, double costoPorTonelada) {} es agregar un deposito sin importar si hay o no otro igual cumpliendo todo las variables del metodo.
* agregarDeposito(double capacidad, boolean frigorifico, boolean propio) {} En este caso el deposito se genera ya en modo frio y refrigerado.
* agregarTrailer(String idTransp, double cargaMax, double capacidad, boolean frigorifico, double costoKm, double segCarga) {} Lo que hace es generar un camión tráiler. Y lo carga a un listado de vehículos que esta en el inicio del contructor de este TAD, segCarga.
* agregarMegaTrailer(String idTransp, double cargaMax, double capacidad, boolean frigorifico, double costoKm, double segCarga, double costoFijo, double comida) {} hace como el agregar tráiler pero obiamente considerando los datos de vehiculo.
* agregarFlete(String idTransp, double cargaMax, double capacidad, double costoKm, int acomp, double costoPorAcom) {} Lo mismo como se generan los otros vehículos.
* asignarDestino(String vehiculo, String destino) {} verifica que el destinoeste contenido entre los destinos y los transportes, de no sucedes alguna de las 2 dara false,
* EN EL CASO DE obtenerTransporteIgual(String id) {} ES UN METODO EN EL QUE N SERA LA CANTIDAD DE VECES QUE DEBERE RECORRER Y COMPARAR EL TRANSPORTE INGRESADO CON EL QUE ESTOY PARADA POR LO TANTO COMO DEBO RECORRER TODO EL TRANSPORTE PARA HALLARLO EN EL PEOR DE LO CASOS ESTE METODO ES LINEA O(N) YA QUE ESTA BASADO EN LISTA LOS TRANSPORTES, LOS PAQUETES, DEPOSITOS,ETC.
* cargarTransporte(String id) {} Lo que hace es verificar que realmente el exista el transporte pedido de lo contrario lanza una exception, y se fija que tenga espacio, la misma refrigeración y va tomando de cada deposito una paquete que respete lo que solicita el transporte de ser asi se van cargango del deposito al transportes los paquetes hasta llenar el vehiculo o deposito lo que seceda primero. Y retorna lo que se cargo en el transporte.
* obtenerCostoViaje(String id) {} Lo que hace es verificar el tipo de transporte que solicita es costo del viaje y llama a ese castiando la llamada al método. De no existir retorna 0.0.
* iniciarViaje(String idTransp) {} lo que hace es verificar que el transporte ingresado si no se contiene que no haga nada y de lo contrario si se contiene llama al método activarViaje(); y si el transporte ya estaba en viaje lanza excepción de lo contrario inicia el viaje, a teniendo en cuenta que este además tenga destino y paquetes que eso lo ve en el método del transporte.
* finalizarViaje(String idTransp) {} Similar al método anterior de ser que el transporte ya haya finalizado lanza excepción sino vacia cada dato eliminando el transporte y creando uno nuevo en su lugar con mismo destino, capacidadMaxima, cargaMaxima yrefrigeracion.
* El equals como la mayoría lo genere usando las mismas herramientas de Eclipse(Java)
* Y lo mismo hice con los toString. //pero como también van o pudieron ver use eso como base y en la mayoría de los TAD y los modifique un poco.

**Problemas y soluciones:**

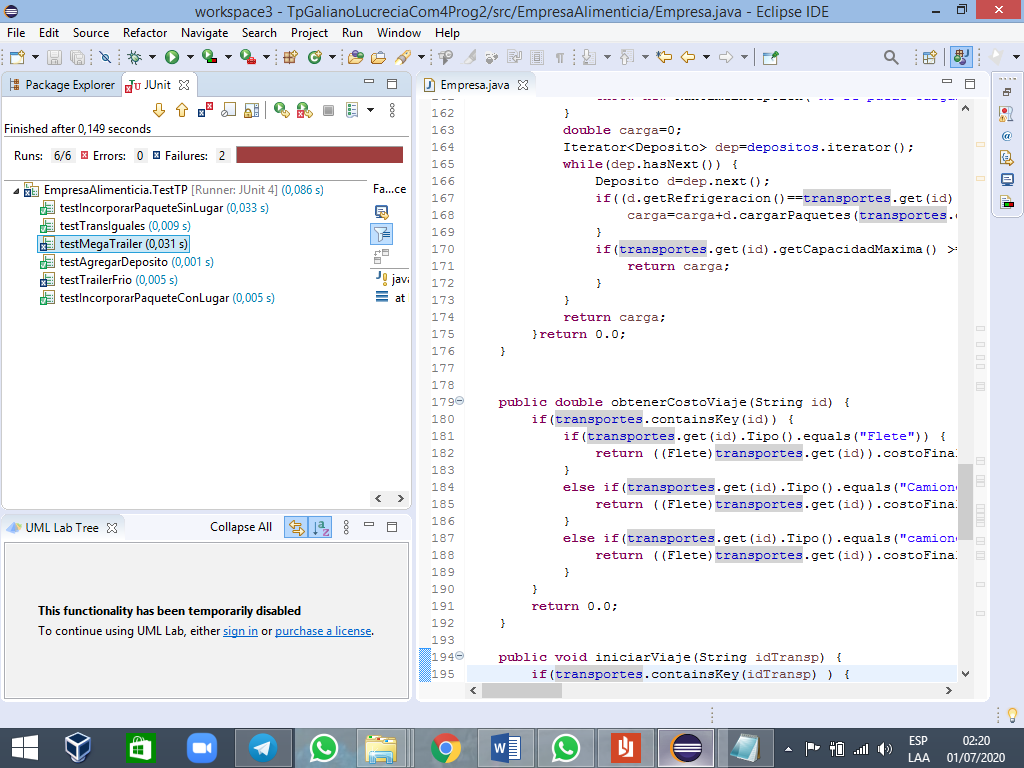
En esta entrega el mayor problema que tuve es que 3 dias antes de que me dieran la corrección logre hacerlo a la perfeccion y eclipse me lanzo 3 carteles que rechaze y en 5 minutos luego me trabo la maquina y lo perdi por completo y creo que parte de esa frustración me hizo que todavía no lo haya logrado ahora( pero a su vez note que en ese momento me deje en el camino varios métodos sin realizar sin iniciar.

Luego cuando logre tener el jUnit que cumpliera en su mayoría que solo me quedaban 3 azules, controlaba el main y me tiraba errores del tipo como que las excepciones estaban mal colocada o que le faltaban cosas a algún if. Y después lo revise nose cuantas veces con los 2 enunciados y verificaba que cumplieran con todo y yo hacen pero ya no se que hacerle. Pobre la profe nose cuantas veces le habre dicho “yo hacen pero ya no se que hacerle”.

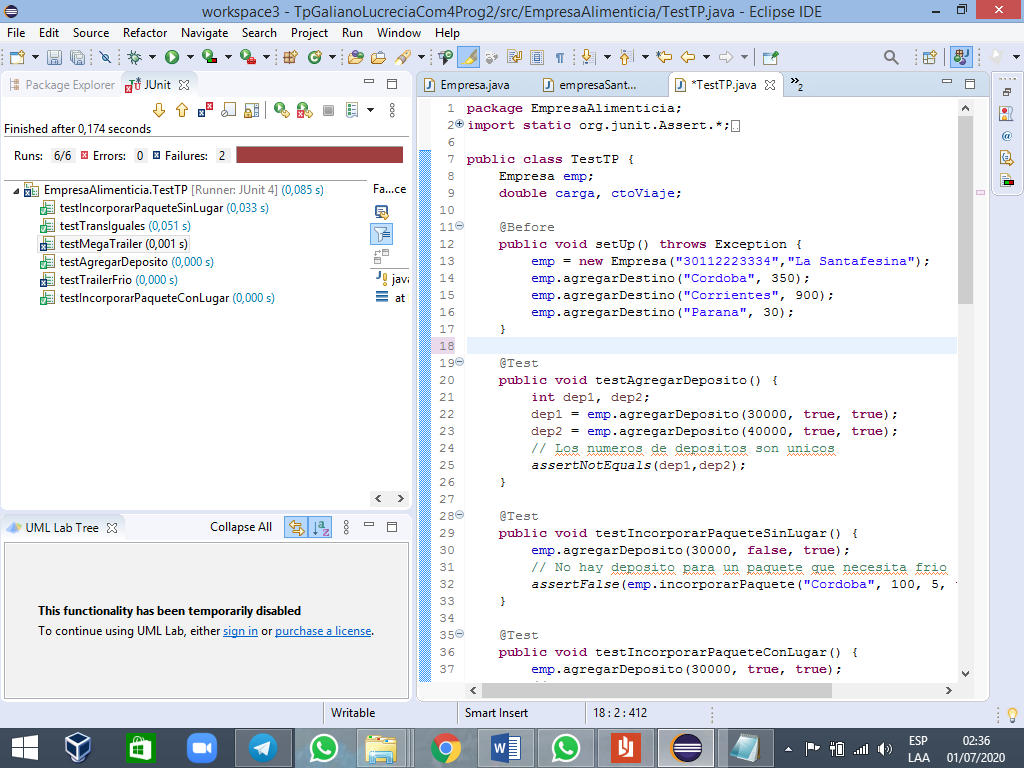
La cuestión es que el problema esta en los mega trailes en un principio creía que cargaba mal los paquetes en el deposito o que no se cargaban lo verifique pero eso anda bien. Intente eliminar lo paquetes sin el remove llamando las mimas funciones de quitar los paquetes y empeoraba.

Y después me enfoque en modificar las funciones de cargas de paquetes en el transporte y que la carga fuera acumulativa utilizando foreach y me saltaban errores, usaba uno si y el otro no y otra vez error.Hasta que lo deje como lo estoy enviando hoy y note que en algunas pruebas da todo con 0.0, si bien sigue dando azul 2 y todos los demás verdes.

(Yo creo esta bien todo.) Además es en el testMegaTrailer el problema creo ya porque en su gran mayoría me dan resultados similares a:



En ocaciones me daban resultados como estos y me desconsertaban.



Luego de horas de programar y probar el test y finalmente al terminar de escribir este informe note donde un error en donde desactivaba los transportes lo hacia mal y lo arregle.

Finalmente gracias.