Trabajo Práctico Grupal. Desarrollo en JAVA

Enunciado del problema

Una **agencia de turismo** desea implementar un sistema para manejar sus **viajes** y **transportes** asociados, gestionando la siguiente información:

- La **agencia** manejará los **viajes pendientes** como así también los **finalizados**. Contará también con un listado de **vehículos, destinos** y **responsables a bordo** para asignar a los viajes nuevos.
- Los **viajes** tendrán un nombre único, destino y cantidad de pasajeros. Incluirán también un estado, que podrá ser "pendiente", "en curso" o "finalizado" y un avance de los kms. recorridos. Se presupone que todos los viajes parten del mismo lugar. Podrán ser de 2 tipos:
    - **Corta distancia**: tendrán una base de cobro fija que luego se sumará a la parte variable.
    - **Larga distancia**: tendrán al menos un responsable a bordo, lo cual incrementará el costo del viaje.
- Los **destinos** tendrán un nombre único y una cantidad de kilómetros que determinarán la longitud del viaje.
- Los **transportes** tendrán información de patente, capacidad de pasajeros y velocidad promedio por hora. Los tipos de transporte serán:
    - **Auto**: tendrá una capacidad máxima de 4 pasajeros. Tendrá un valor base por viaje, más un valor por km a recorrer.
    - **Combi**: tendrá una capacidad máxima de 16 pasajeros. Tendrá un valor base por viaje, más un valor por pasajero por km a recorrer.
    - **Colectivo semi-cama**: tendrá una capacidad máxima de 40 pasajeros. Tendrá un valor por pasajero por km a recorrer.
    - **Colectivo coche-cama**: tendrá una capacidad máxima de 32 pasajeros. Tendrá 26 plazas de tipo "cama" y 6 plazas comunes. Tendrá un valor por pasajero por km a recorrer, más un valor adicional por las plazas ocupadas que sean de tipo "cama" por km a recorrer.
- Los **responsables a bordo** tendrán información de su nombre, DNI y sueldo por viaje (fijo independiente de la cantidad de kilómetros).

Se solicita que el sistema permita:

- **Administrar** el archivo de **vehículos** (ABM).
- **Administrar** el archivo de **responsables a bordo** (ABM).
- **Cargar** los **destinos** posibles para los viajes, como una lista fija, a partir de un **archivo** de texto o XML.
- **Crear** nuevos **viajes**, solicitando la siguiente información:
    - Destino del viaje: mediante el destino elegido se conformará el nombre único del viaje, que estará compuesto por el nombre del destino más un número que permita diferenciar este viaje del resto de los existentes con el mismo destino. Si la cantidad de kilómetros del destino supera los 100 kms. se considerará un viaje de larga distancia. De lo contrario. será de corta distancia.
    - Cantidad de pasajeros.
    - Transporte asignado al viaje: permitir asignar un transporte al viaje a partir de un listado dinámico de transportes.

  De acuerdo a los datos del viaje, se deberán contemplar las siguientes restricciones, en forma dinámica:

- Si el viaje es de larga distancia se solicitará asignar al menos un responsable a bordo, lo que incrementará el costo del viaje. No se podrán seleccionar autos este tipo de viajes.
- Si el viaje es de corta distancia no se podrán seleccionar colectivos del tipo "coche cama" para realizarlo.
- La cantidad de pasajeros no debe superar la capacidad del vehículo seleccionado.
- Si el vehículo a utilizar es "coche cama" se deberán seleccionar las plazas que serán cama del total de pasajeros del viaje. La cantidad de plazas de tipo "cama" por defecto será la máxima posible del total de pasajeros asignados.
- Un vehículo o responsable a bordo asignado a un viaje no deberá aparecer en la lista de recursos disponibles para uno nuevo, hasta que el viaje en el que estén asignados no se complete.

Para cada viaje deberá calcularse su valor a medida que se lo configura en su creación. Para su cálculo se deberá tener en cuenta el costo base si es corta distancia o bien el costo por responsables a bordo si es de larga distancia. A esto se deberá sumar el costo por el vehículo empleado para el viaje, de acuerdo a lo previamente especificado.

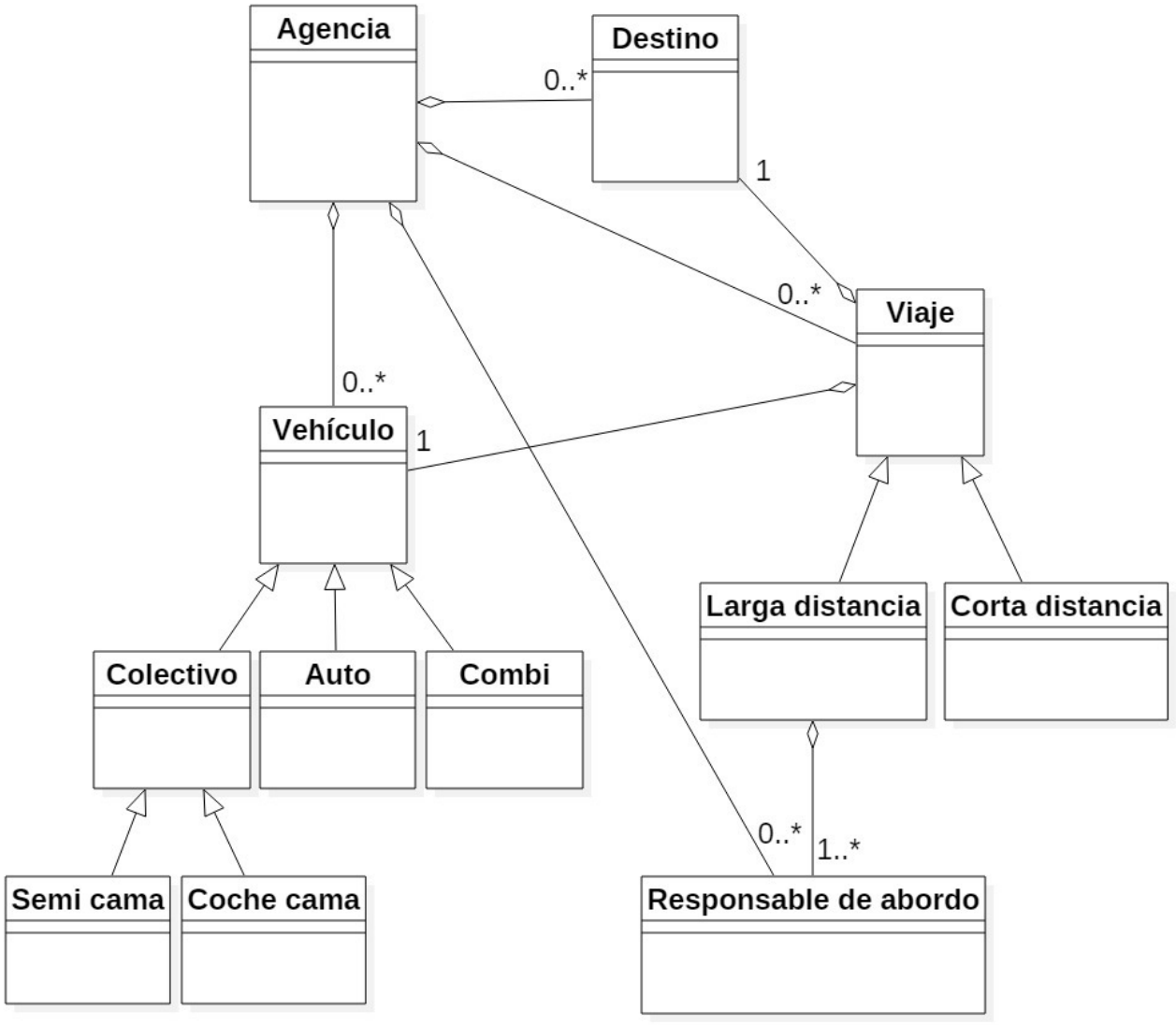
- Contar con una pantalla de **monitoreo de viajes** que permita **comenzar todos los viajes pendientes**. La interfaz proveerá una opción para iniciar, detener la ejecución de los viajes de la lista de pendientes. Cada viaje en ejecución irá mostrando información en un monitor:
  - Información del viaje al comenzar: nombre del viaje, kms. totales, cantidad de pasajeros, transporte asignado, valor.
  - Información del avance de cada viaje por hora: nombre del viaje, kms. recorridos y porcentaje cumplido para los viajes en curso, estado del viaje.

  A medida que los viajes comiencen y terminen se deberá actualizar su estado. Los viajes finalizados deberán moverse a la lista de viajes terminados. También se deberán liberar los recursos que ese viaje tenía ocupados, para poder comenzar uno nuevo.

  Para la simulación, cada segundo de ejecución será 1 hora de la vida real, medida que se deberá tener en cuenta para saber cuánto avanzó cada viaje y actualizar sus datos.
- Genere los siguientes **reportes** (por pantalla y en archivos de texto):
  - Recaudación de los viajes realizados por la empresa, permitiendo ver la información total o bien visualizarla por cada transporte y/o cada destino.
  - Ranking de responsables a bordo ordenado de mayor a menor por cantidad de kilómetros recorridos en los viajes terminados.
- Realice las **validaciones** necesarias para que la información sea consistente (ej. completitud de datos obligatorios como es el destino del viaje y la cantidad de pasajeros, validez de los valores numéricos, etc)

Sugerencias y comentarios

- **Diagrama de clases** sugerido



- La **interfaz de usuario** puede ser elegida por el grupo (caracter, gràfica (AWT y Swing), web (html + Servlets ó JSP). Está permitida la opción de desarrollar el trabajo como un proyecto de **Android**, teniendo en cuenta las variaciones de la interfaz de usuario y configuraciones adicionales.
- La **persistencia** debe implementarse mediante serialización (clásica o XML)
- Considerar:
  - el uso de las clases *Containers* provistas por Java (Tree*Set*, *ArrayList, LinkedList, TreeMap*) para administrar listas y conjuntos.
  - el uso de *Iterator* para recorrer colecciones.
  - el uso de clases específicas para el manejo de los reportes.
  - el uso de *Enum* para los valores discretos.

Condiciones de Aprobación Trabajo Práctico Java

- Conformar un **grupo** de 3 personas, Según el número de alumnos del curso se permitirán 2 grupos de 2 alumnos o un grupo de 4 alumnos.
- Implementar la totalidad de la **funcionalidad** solicitada en el enunciado del problema.
- Aplicar indefectiblemente en la solución los siguientes conceptos de la Programación orientada a Objetos: **encapsulamiento**, **polimorfismo**, **herencia**, **clases abstractas**.
- Cumplir estrictamente con el **cronograma de entregas** (parcial y final) según el siguiente detalle:

	Fecha
Entrega Parcial	25/04/2016
Entrega Final – 1ra. Fecha	02/05/2016
Entrega Final – 2da. Fecha	23/06/2016
Entrega Final – 3ra. Fecha	2da. fecha de final de julio/agosto

- En el caso de no presentar o no aprobar la Entrega Final en la 1ra. o 2da. Fecha, se deberá implementar para la 3ra. fecha de la Entrega Final, la **funcionalidad anexa** que solicitará oportunamente la cátedra. En el caso de no presentar o no aprobar la Entrega Final en ninguna de las 3 instancias, **se pierde la materia**.
- En la Entrega Final:
    - presentar copia digital (CD, por mail, etc.) de los **archivos fuentes** (.java) y la **documentación** en **HTML** generada automáticamente con *javadoc.* También se deberá entregar una copia impresa del código fuente (no es necesario imprimir la totalidad de la Interfaz de usuario (GUI), solamente la relacionada a lo funcional) **coincidente** con el que se entrega para su corrección. Todos los grupos deberán entregarla al comienzo de esa clase, el grupo que no lo haga no podrá presentar el TP en esa fecha.
    - La cátedra podrá proponer un **lote de datos** para la prueba del sistema
    - Deberán estar presentes **todos** los integrantes del grupo pues se hará una primera evaluación oral del trabajo presentado. El integrante que no lo estuviera se considerará **fuera del grupo** y deberá realizar su propio TP.
    - Los días de entrega no se permitirán modificaciones de último momento y los grupos que hagan la entrega deberán estar presentes de forma completa al horario de inicio de la clase. El resto de los grupos podrá usar el tiempo de clases para continuar con el desarrollo del TP para futuras entregas.
    - La **nota** del trabajo práctico es **individual**, basada en la participación en la resolución y defensa del trabajo práctico, y en los conocimientos conceptuales exhibidos en las entregas. Otra componente relevante en esta nota será la asistencia completa a las clases, y los avances efectuados durante las mismas.
- Otros **conceptos** que incidirán en la **aprobación** del trabajo práctico son:
    - Reutilización adecuada del código.
    - Eficiencia en los algoritmos (ej: búsquedas, ordenamientos)
    - Bajo acoplamiento entre interfaz y lógica de dominio
    - Empaquetamiento criterioso de las clases.
    - Utilización de operadores, métodos y técnicas propias del lenguaje Java (ej: manejo de errores con excepciones propias, métodos *compareTo*, *toString*)
    - Validaciones de ingresos de datos y consistencia de la información.
    - Código prolijo, claro y correctamente comentado. (ej: nombres representativos, crear variables e instancias necesarias, sobrecargar métodos)