Programación 4

EXAMEN DICIEMBRE 2014

Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas
- Escriba su nombre y número de documento en todas las hojas que entregue
- Numere las hojas e indique el total de hojas en la primera de ellas
- Recuerde entregar su numero de examen junto al examen
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial

Problema 1 (30 puntos)

Parte A:

Cuando se habla de redefinición de operaciones y de sobrecarga de operaciones, ¿se habla del mismo concepto?. Explique y ejemplifique su respuesta.

Parte B:

Se desea construir un sistema de *asistencia en ruta para conductores* (estilo GPS). El mismo contiene un conjunto muy grande de ciudades pre-cargadas, pero no así sus mapas, los cuales deben comprarse uno a uno para cada ciudad que el conductor desee.

Cada mapa tiene un nombre (el que debe coincidir con el nombre de la ciudad), una fecha de adquirido y es representado por un conjunto de segmentos de los cuales se conoce su coordenada de inicio, coordenada de fin, y una longitud (en metros). De esta forma, una calle no es más que un conjunto de segmentos a los cuales se les agrega el nombre de la calle y el tipo (Avenida, Bulevar, Ruta, Camino o Normal).

Finalmente, lo más importante del sistema: las rutas. Éstas se identifican por un nombre, se representan como un conjunto ordenado de segmentos, y también tienen la longitud total de la ruta (dada por la suma de las longitudes de sus segmentos).

Caso de Uso:	Recorrer Ruta
Actor:	Conductor
Descripción:	Este CU comienza cuando el conductor elige un mapa (mediante su nombre) de todos los mapas comprados por él. Como respuesta a la elección del mapa, el sistema devuelve una lista de todas las rutas que fueron definidas para ese mapa. Luego el conductor debe elegir una de estas rutas (la que quiera recorrer en ese momento) mediante su nombre. Esto hará que el sistema comience a guiar al conductor a través de la ruta seleccionada, segmento por segmento, debiendo el conductor solicitar pasar al siguiente segmento (esto es debido a que el sistema no cuenta con ubicación automática del vehículo) o bien cancelar la recorrida. El sistema guiará al conductor a través de todos los segmentos de la ruta elegida hasta haber terminado de recorrerla o bien hasta que el conductor decida cancelar.

Se pide:

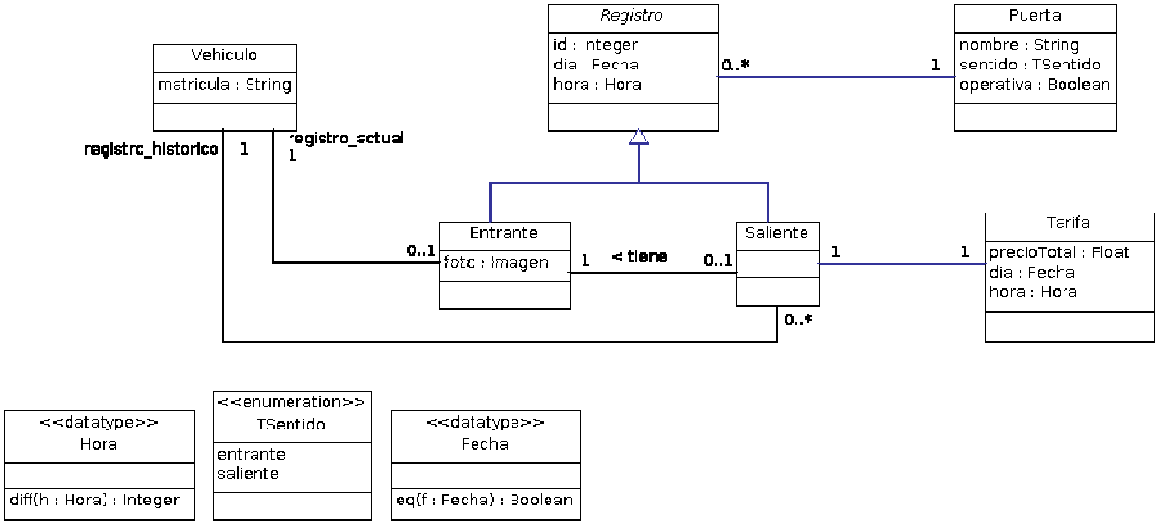
a) Modelo de dominio con restricciones en lenguaje natural.

b) DSS para el Caso de Uso Recorrer Ruta.

Problema 2 (35 puntos)

Uno de los shoppings más importantes de Montevideo le ha encargado *la gestión del acceso de vehículos*. En él se mantendrá un registro de entrada-salida, interesando ingresar un documento fotográfico tomado al ingreso. Además, se anotará la puerta por la cuál accede, registrando día y hora, ya sea de entrada como de salida.

El equipo de análisis ha construido el siguiente modelo de dominio:



Se consideran los casos de uso "Ingreso de vehículo", "Salida de vehículo" y "Obtener plazas libres", cada uno modelado con una sola operación cuyos contratos se especifican a continuación:

Operación	`ingreso_vehiculo(mat: String, foto: Imagen, nom_puerta: String)`
Descripción	Ingresa un nuevo registro entrante de un vehículo para el día y hora del sistema.
Parámetros	**mat**: Matrícula del auto **foto**: Imagen obtenida del auto **nom_puerta**: Nombre identificador de la puerta de acceso entrante
Precondiciones	- Existe en el sistema una puerta entrante de nombre **nom_puerta** - La cantidad de plazas disponibles es mayor a cero. - El sentido asociado a **nom_puerta** es entrante
Postcondiciones	- En caso de no existir, se crea una instancia Vehículo, con matricula **mat** - Se crea una nueva instancia del registro con foto igual a **foto** e identificador **id** autogenerado, asignándole el día y hora actual del sistema. - Se crea un link entre el vehiculo y el registro creado - Se crea un link entre el registro y la Puerta

Operación	`salida_vehiculo(mat: String, nom_puerta: String)`
Descripción	Ingresa un nuevo registro saliente de un vehículo para el día y hora del sistema. A su vez, registra la tarifa a asignar en base al tiempo utilizado.
Parámetros	**mat**: Matrícula del auto **nom_puerta**: Nombre identificador de la puerta de acceso saliente
Precondiciones	- Existe en el sistema una puerta saliente de nombre **nom_puerta** - Existe en el sistema un registro **Entrante** para el auto con matrícula **mat**
Postcondiciones	- Se crea una nueva instancia del registro asignándole el día y hora actual del sistema con identificador autogenerado. - Se crea un link entre el Registro y la Puerta - Se crea un link entre el vehículo y el registro creado - Se crea una nueva instancia de Tarifa asignándole el día y hora actual del sistema. El **precioTotal** se calcula como la diferencia horaria entre la entrada y la salida multplicado por la constante entera **TARIFA_HORA** - Se crea un link entre la tarifa y el registro saliente - Se crea un link entre el registro saliente y el entrante asociado al vehículo - Se destruye el link entre el vehículo y el registro entrante actual

Operación	`obtener_plazas_libres(): Integer`
Descripción	Retorna la cantidad de plazas disponibles, tomando como valor máximo de plazas **TOTAL_PLAZAS**
Postcondiciones	Retorna la cantidad de plazas disponibles, siendo este valor la diferencia entre **TOTAL_PLAZAS** y la cantidad de plazas ocupadas.

Nota:

- **TOTAL_PLAZAS** así como **TARIFA_HORA** son constantes del sistema.

Parte A:

a) Realizar los Diagramas de Comunicación de las operaciones del sistema mencionadas anteriormente, indicando tipo de visibilidad en todos los mensajes, definiendo los nuevos DataTypes que puedan haber surgido.

b) Realizar el Diagrama de Clases de Diseño resultante.

Parte B:

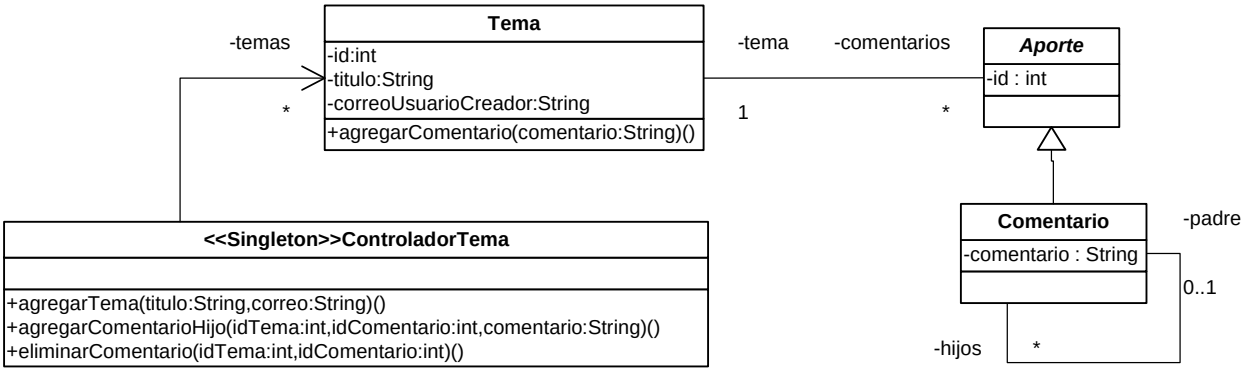
Dado que en cualquier momento pueden ocurrir fallas operacionales en las puertas de acceso tanto entrantes como salientes, el equipo de análisis le sugiere que su diseño sea capaz de incorporar un método de aviso, por ejemplo el uso de una alarma, al momento que las puertas dejen de estar operativas.
Este requerimiento debe implicar realizar la menor cantidad de modificaciones posibles a su diseño realizado.

Se pide: Indicar el/los patrones de diseño que cree son necesarios para implementar esta funcionalidad indicando los roles para cada clase. Justifique su elección.

Problema 3 (35 puntos)

Parte A:

Se dispone del siguiente modelo de una parte de un sistema donde se pueden ingresar temas de interés, y hacer comentarios sobre los mismos.



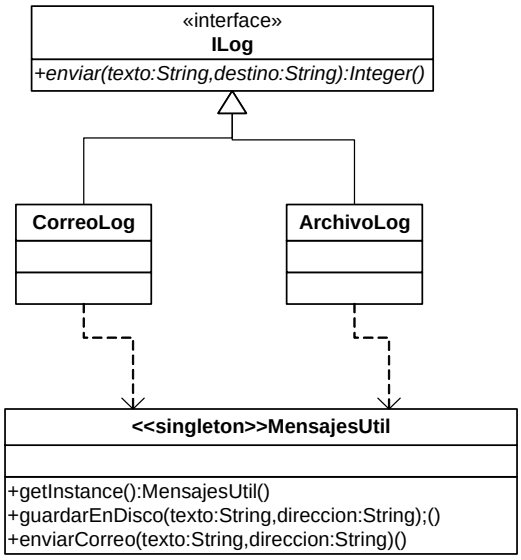
Se pide: Implemente los .h y cpp de todas las clases que aparecen en el modelo, según corresponda con sus atributos, asociaciones, operaciones, constructores y destructores. De la clase **ControladorTema**, solo implemente las operaciones:

- **eliminarComentario** que elimina la instancia de Comentario con id igual a idComentario del tema idTema
- **agregarTema** que crea un nuevo Tema con el título y correo del usuario creador pasados por parámetro (el id es autogenerado)
- **agregarComentarioHijo** que agrega un comentario a la colección de hijos del comentario con idComentario, dentro del Tema con idTema. (el id del nuevo comentario es autogenerado)

Si utiliza operaciones auxiliares debe implementarlas.

Parte B:

a) Se desea implementar un mecanismo de log para incorporar a futuro en el sistema. La implementación de este mecanismo permitirá guardar mensajes en disco y enviar por correo cierta información. Existe actualmente implementada una clase denominada **MensajesUtil** que ofrece operaciones para el envío de correo electrónico y guardado en disco, sin embargo no se desea que esta clase quede acoplada directamente a ninguna de las clases descritas en la parte A, por lo que se decidió crear una interfaz **ILog** con la operación **int enviar(String texto,String destino)** cuyas implementaciones (**CorreoLog** y **ArchivoLog**) encapsulen las llamadas a **MensajesUtil**. Las operaciones **enviarCorreo** y **guardarEnDisco** de dicha clase, si ocurre algún error lanzan la excepción: **ExcepcionDeEnvioCorreo**, y **ExcepcionGuardadoEnDisco**.



Se pide: Implemente la Interfaz **ILog**, y la clase **CorreoLog**. El parámetro destino en la operación enviar se utiliza para especificar la dirección de correo electrónico. Debe implementar el manejo de excepciones en dicha operación, que en caso de ocurrir un error debe retornar -1 y en caso de éxito 0.

b) Se desea incorporar al Sistema un nuevo tipo de clase de tipo **Aporte**, que será denominada **Pregunta**. Un **Tema** podrá tener asociado un conjunto de Preguntas, y una Pregunta estará asociada a un solo Tema. Una Pregunta podrá tener muchos comentarios pero ninguna Pregunta asociada a si misma. Además del identificador, esta nueva clase contará con un atributo de tipo String denominado pregunta, que se utilizara para describir la misma.

Se pide:

i) Extienda del diagrama de la **Parte A** con ésta nueva clase y las características detalladas anteriormente.

ii) Implemente la clase Pregunta.

iii) Implemente la operación
void AgregarPregunta(idTema:int,pregunta:String) en la clase ***ControladorTema***, que agrega al Tema con id id*Tema* una nueva Pregunta, y notifica al correo electrónico asociado al Tema, un mensaje que contiene la pregunta realizada. Si utiliza operaciones auxiliares deben ser implementadas. Para enviar un correo debe utilizar las clases implementadas en la **parte B a)** y no acceder directamente a **MensajesUtil.**

Observaciones generales:
- No incluir directivas al precompilador.
- Puede suponer la existencia de implementaciones de IDictionary, ICollection, IIterator y KeyInt según sea necesario.
- Es posible utilizar las clases set<T> y map<K,V> de la STL.

- Todos los atributo denominados como id son autogenerados, asuma que existe una clase denominada **Util**, que puede utilizar y que dispone de una operación statica: ***int getId()***, que devuelve un entero distinto cada vez que es invocada.
- En la eliminación de elementos, todos los borrados son en cascada, es decir, para un tema se eliminan todos sus comentarios, para un comentario, todos los comentarios sobre si mismos.