Programación 4

EXAMEN FEBRERO 2024

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 número de examen
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el examen

Problema 1 (30 puntos)

Se desea crear un sistema para gestionar un taller mecánico de autos. Los clientes son registrados con su email (único), nombre y teléfono. Un cliente puede tener varios vehículos. Para cada vehículo se conoce su matrícula para identificarlo, marca, modelo y color. Además, de cada vehículo se mantiene un registro de todas las órdenes de trabajo, identificadas por un número. Para cada orden además se registra la fecha, estado (Pendiente, En Curso, Finalizada), descripción y costo total, que es calculado automáticamente dependiendo del tipo de orden. Las órdenes pueden ser de mantenimiento o reparación. Para las de mantenimiento se registra la cantidad de horas de trabajo, mientras que para las de reparación se lleva cuenta de todas las partes y cantidad necesaria de cada una de ellas. El sistema cuenta con un catálogo de partes, que se identifican por un código y se conoce su costo actual. Las partes pueden estar compuestas por otras partes que interesa conocer.

Además, se cuenta con la descripción del siguiente Caso de Uso:

Caso de Uso	Alta de orden
Actor	Usuario
Descripción	El caso de uso comienza cuando un usuario desea agregar una orden para un cliente. Para eso, primero se ingresa el email del cliente para que el sistema liste los vehículos disponibles. El usuario selecciona un vehículo e ingresa los datos correspondientes de la orden. Si es una orden de reparación, el sistema lista las partes disponibles y el usuario selecciona aquellas involucradas en la reparación, junto con la cantidad de cada una. Finalmente, el usuario puede elegir cancelar o confirmar la orden para darla de alta.

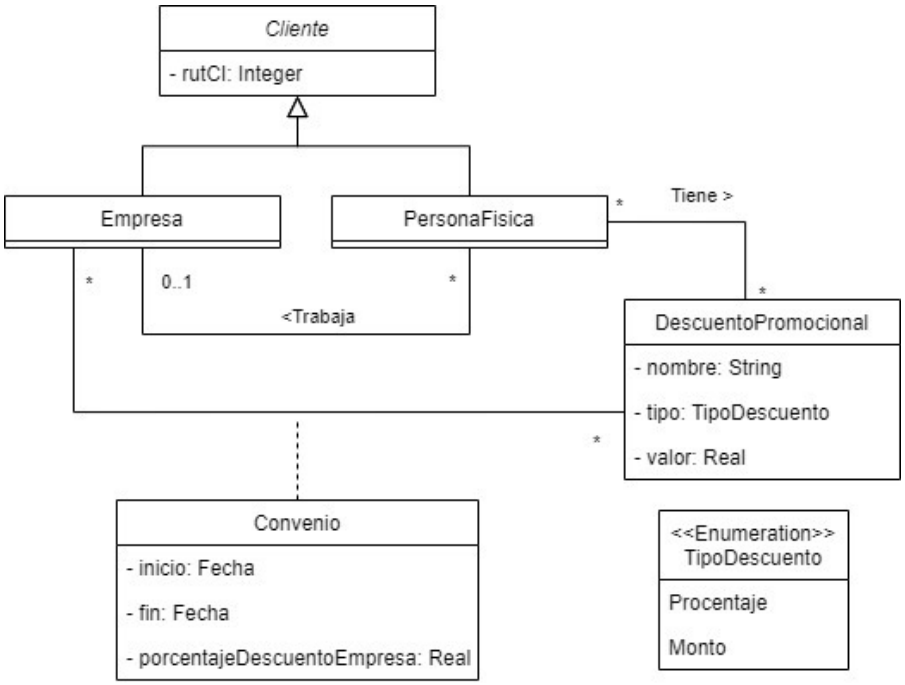
Se pide:

a) Realizar el Modelo de Dominio de la realidad planteada, incluyendo todas las restricciones que considere necesarias en lenguaje natural.
b) Realizar un Diagrama de Secuencia del Sistema para el Caso de Uso. Indique el uso de memoria del Sistema y de datatypes, si corresponde.

Problema 2 (35 puntos)

Le encomendaron el desarrollo de un sistema para manejar información de clientes y descuentos que se aplican a sus compras. A continuación, se presenta el modelo de dominio realizado en la etapa de análisis. Algunas observaciones respecto al modelo:

- El atributo rutCI (identificador) denota el RUT si el cliente es una empresa, o la cédula si es una persona física.
- La asociación Trabaja indica que una persona física trabaja en una empresa.
- Los descuentos pueden ser de montos o porcentajes, y son no acumulables. En caso de poseer más de un tipo de descuento siempre se deberá tomar el más favorable para el cliente, es decir, el de mayor monto.
- Las empresas tienen convenios que tienen una cierta validez (sólo puede haber uno activo para determinada fecha), que permiten obtener un porcentaje de descuento (que aplica exclusivamente la empresa).
- Las personas físicas tienen descuentos promocionales, ya sea directamente o través de una empresa. En este último caso, se trata del descuento promocional asociado al convenio vigente.
- Los descuentos promocionales se identifican por el nombre. El valor es un porcentaje o un monto, dependiendo del tipo de descuento.



Se definieron las siguientes operaciones del sistema:

`cotizarMejorDescuento(rutCI:Integer, precio:Real) : Real`	
Descripción	Retorna el monto del mayor descuento para una compra de un cliente. Si no hay ningún descuento se retorna cero.
Parámetros	`rutCI` : identificador del cliente `precio` : valor al cual se aplica el descuento
Precondiciones	• Existe un cliente con identificador `rutCI`. • Toda empresa tiene a lo sumo un convenio asociado, tal que la fecha actual se encuentra entre el inicio y el fin del convenio.
Postcondiciones	• Se devuelve el monto del mayor descuento para el valor `precio`, para el cliente `c` identificado por `rutCI`. • Si `c` es una `Empresa`, el resultado se calcula aplicando a `precio` el valor `porcentajeDescuentoEmpresa` de su único convenio vigente (si hay alguno).

	• Si `c` es una `PersonaFisica`, el resultado es el mayor monto resultante de aplicar los descuentos de todas las instancias de `DescuentoPromocional` asociadas a la persona y la asociada al convenio vigente de la empresa para la que trabaja la persona (si es que la persona trabaja para una empresa, y esta tiene un convenio vigente).

`altaConvenio(rutCI:Integer, inicio:Fecha, fin:Fecha, porcentajeDescuentoEmpresa:Real, nombreDescuentoPromocional:String)`	
Descripción	Da de alta un convenio vigente para una empresa.
Parámetros	`rutCI` : identificador del cliente `inicio/fin` : fechas de inicio/fin del convenio `porcentajeDescuentoEmpresa` : porcentaje de descuento aplicable a la empresa `nombreDescuentoPromocional` : identificador del descuento promocional asociado al convenio, que se aplica a las personas físicas que trabajan en la empresa
Precondiciones	• Existe un cliente con identificador `rutCI` • La fecha actual se encuentra entre `inicio` y `fin` • Existe una instancia de `DescuentoPromocional` identificada por `nombreDescuentoPromocional` • No existe un convenio asociado al cliente identificado por `rutCI`, tal que la fecha actual se encuentra entre `inicio` y `fin` del convenio
Postcondiciones	• Existe una nueva instancia `c` de `Convenio` con los datos `inicio`, `fin` y `porcentajeDescuentoEmpresa` • Existen nuevos links entre `c` y el `Cliente` identificado por `rutCI` y el `DescuentoPromocional` identificado por `nombreDescuentoPromocional`

Se pide:

a) Realizar los Diagramas de Comunicación correspondientes a las dos operaciones especificadas.
b) Realizar el Diagrama de Clases de Diseño resultante.

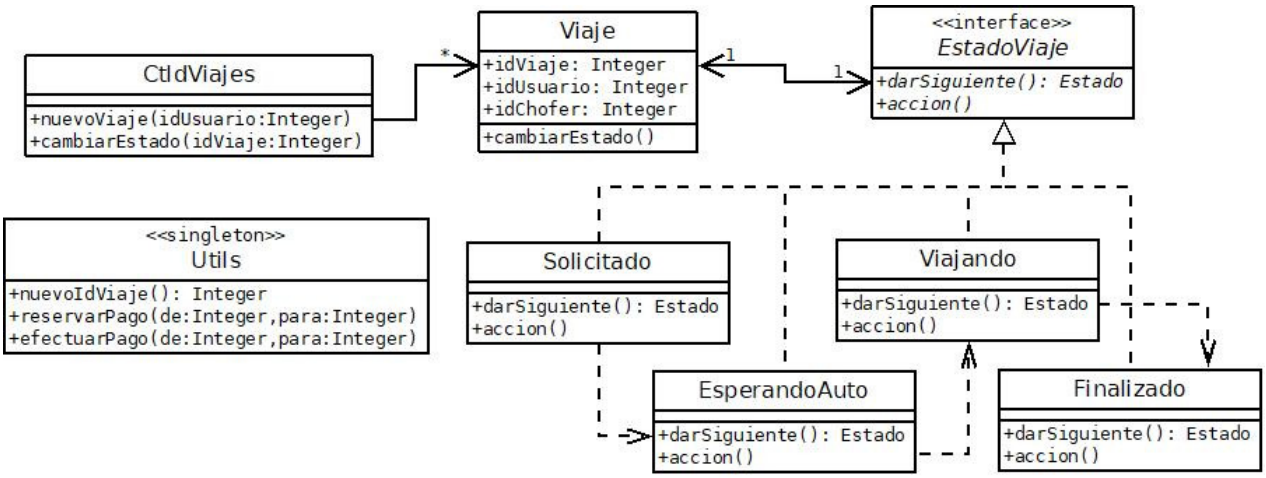
Nota: Las fechas pueden ser comparadas con los operadores básicos (<, >, =, >=, <=, etc.) como cualquier tipo primitivo. Asuma que cuenta con un identificador que denota la fecha actual del sistema.

Problema 3 (35 puntos)

Se está desarrollando una plataforma de soporte a un servicio de viajes urbanos, mediante el cual usuarios registrados pueden desplazarse entre dos puntos de una ciudad, utilizando vehículos que son proporcionados por choferes también registrados. La figura presenta una parte del diseño de la plataforma. El controlador mantiene una colección de todos los viajes actuales. Cada viaje tiene un estado que cambia según el siguiente esquema:

- Solicitado: cuando el usuario solicita el viaje y espera a que algún chofer disponible le responda.

- EsperandoAuto: cuando el chofer confirmó disponibilidad y se está dirigiendo hacia el lugar de origen donde se encuentra el usuario.
- Viajando: cuando el usuario subió al auto y se encuentra viajando hacia el destino del viaje.
- Finalizado: cuando el usuario llegó al destino y descendió del auto.



A continuación, se describe el comportamiento de las operaciones definidas:

`CtldViajes::nuevoViaje(idUsuario:Integer):Integer`

Genera un nuevo viaje con el identificador del usuario (el del chofer se asignará posteriormente), con estado Solicitado, y lo almacena en la colección de viajes de sistema. El identificador del nuevo viaje se genera con la opreación `Utils::nuevoIdViaje()`.

`CtldViajes::cambiarEstado(idViaje:Integer)`

Cambia el estado del viaje identificado por el parámetro, al siguiente estado.

`Viaje::cambiarEstado()`

Cambia al siguiente estado del viaje.

`Estado::darSiguiente():Estado`

Devuelve una instancia del siguiente estado, según el diseño de la figura.

`Estado::accion()`

En Solicitado y Finalizado, no hace nada. En EsperandoAuto invoca la operación `Utils::reservarPago()` desde el usuario para el chofer, utilizando la información almacenada en los datos del viaje. En Viajando invoca la operación `Utils::efectuarPago()` de forma similar a la anterior.

Se pide:

a) Indicar si el diseño planteado utiliza algún patrón de diseño. En caso afirmativo, decir el nombre del patrón e identificar los roles de las clases participantes.
b) Implementar en C++ el .h y .cpp de las clases CtldViajes y Viaje.
c) Implementar en C++ el .h de la interfaz Estado.
d) Implementar en C++ el .h y .cpp de las clases EsperandoAuto y Viajando.

Observaciones:

- No se pueden agregar ni utilizar más operaciones que las definidas en el diseño anterior.
- Puede asumir que existen operaciones get y set para los atributos, y utilizarlas.
- Implementar los constructores que sean necesarios. No implementar destructores.