Programación 4

EXAMEN FEBRERO 2025

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 de gestión de tareas para proyectos de software. Los proyectos identificados por un código se organizan en iteraciones, las cuales se definen con un rango de tiempo [inicio, fin], no pudiendo superponerse entre ellas para un mismo proyecto. Las iteraciones se identifican con números relativos al proyecto (por ejemplo, Proyecto A23, iteraciones 1,2,3; Proyecto B12 iteraciones 1,2) y agrupan un conjunto de tareas. De cada iteración se conoce el estimado total de esfuerzo en horas de las tareas que la integran. Las tareas pueden ser de tipo Historia, Funcionalidad o Bug. Todas se identifican con un código compuesto por el prefijo del código del proyecto y un número relativo a este, por ejemplo, A23-2, B12-45. Las tareas de tipo historia pueden opcionalmente componerse exclusivamente de otras historias, funcionalidades y/o bugs. De cada tarea se conoce su descripción, la iteración a la que pertenece, su estado entre {Pendiente, En Progreso, Completada}, el estimado en horas, el usuario asignado y el creador. En caso de ser un Bug se registra la fecha y hora de su detección, mientras que para las funcionalidades se registra su prioridad definida en el rango [1..5]. Los usuarios se identifican por un email y son asignados a proyectos y tareas. Los usuarios solo pueden ser asignados a tareas de sus proyectos y un proyecto puede tener hasta 10 usuarios asignados. Una tarea puede ser asignada únicamente a un usuario por vez, pero puede ser reasignada a múltiples usuarios durante su ejecución, por lo que interesa registrar cuánto tiempo (hs) trabajó cada usuario en cada tarea.

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

Caso de Uso	Alta de Historia
Actor	Usuario
Descripción	El caso de uso comienza cuando un usuario desea dar de alta una historia. Para eso el sistema lista todos los proyectos al que el usuario está asignado. El usuario elige el proyecto en el cual desea crear la historia. Se listan todas las iteraciones que actualmente componen al proyecto. En caso de querer agregar la historia a una iteración existente, el usuario la elige, en caso contrario crea una nueva. El usuario indica la descripción y el estimado de horas de la historia a crear. Si el usuario desea agregar tareas que la componen puede realizarlo, indicando los datos necesarios para crear cada una dependiendo del tipo. Las tareas componentes se agregan una a una, listando cada vez que se agrega una y pudiéndose confirmar o deshacer cada adición. Finalmente, el sistema lista todos los detalles de la historia a crear, incluyendo los de las tareas que la componen en caso de que aplique, para que el usuario confirme o cancele.

Se pide:

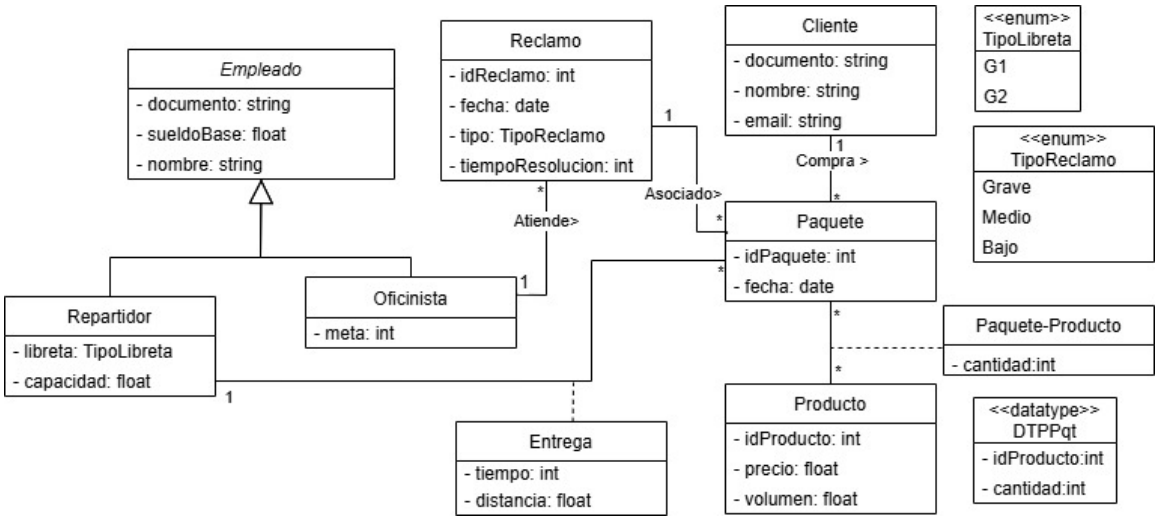
a) Realizar el Modelo de Dominio de la realidad planteada, incluyendo todas las restricciones que considere necesarias en lenguaje natural.
b) Realizar un Diagrama de Secuencia del Sistema para el Caso de Uso. Indique el uso de memoria del Sistema y de datatypes, si corresponde.

Problema 2 (35 puntos)

Luego del análisis realizado por un equipo externo para un sistema de reparto de paquetes, gestión de sueldos y reclamos, se detectaron los conceptos generales de empleados, clientes, productos, paquetes y reclamos, cuyo detalle se explica a continuación:

- Los empleados se identifican por el documento y se registra su sueldo base y su nombre. Se dividen en dos tipos, oficinistas, para los cuales se registra una meta mínima de cantidad de reclamos atendidos, y repartidores, de los cuales se conoce qué tipo de libreta poseen (G1, G2) y la capacidad máxima de carga.
- De los clientes se conoce el documento que los identifica, nombre y email.
- De los productos se registra el número identificador, el precio y el volumen que ocupan, lo que luego servirá para determinar el volumen total del paquete y si este puede ser llevado por los repartidores.
- De los paquetes se conoce un identificador único generado por el sistema, la fecha de envío y qué productos contienen, pudiendo haber más de una unidad del mismo producto asociado al paquete.
- Para los reclamos se registra qué oficinista los atiende, un identificador único generado por el sistema, la fecha, el tipo de reclamo (grave, medio, bajo), y el tiempo en minutos que se demoró en resolverlo.

Se desea que el sistema sea capaz de una vez generado el paquete y tomando en cuenta la distancia en kilómetros pasada por el cliente, asociar un repartidor con capacidad disponible para esa fecha. Una vez que el repartidor lo entregue, este indicará el tiempo que le llevó realizar la entrega. En base a esta información lograron generar el siguiente modelo de dominio.



Se definieron las siguientes operaciones del sistema:

`genPaq(doc:String, fecha:date, dist:float, prods:Set(DTPPqt)):String`	
Descripción	Crea un paquete con fecha y productos, genera la entrega asociada a un repartidor con la capacidad disponible para la fecha y retorna el nombre del repartidor
Parámetros	`String doc` identificador del cliente `Date fecha` fecha de envío del paquete `Float dist` distancia de entrega pasada por el cliente `Set(DTPPqt) prods` lista con los identificadores de productos y sus cantidades
Precondiciones	Existe un `Cliente` con identificador `doc` Para cada datavalue `dtp` de `prods` existe un `Producto p` tal que `dtp.idProducto = p.idProducto` Existe un `Repartidor` con capacidad disponible (volumen de paquete) para el día del envío indicado por `fecha`

Postcondiciones	Se genera una instancia de `Paquete pq` con `pq.idPaquete` generado por el sistema y `pq.fecha=fecha` Para cada datavalue `dtp` en `prods` se crea una instancia de `Paquete-Producto pp` con `pp.cantidad=dtp.cantidad`, un link entre `pp` y el `Producto p` tal que `p.idProducto = dtp.idProducto` y un link entre `pp` y `pq` Se crea un link entre `pq` y el `Cliente c` tal que `c.documento = doc` Se crea una instancia de `Entrega e` con `e.distancia = dist` y `e.tiempo` nulo, se crea un link entre `pq` y `e`, y se crea un link entre `pq` y un `Repartidor r` tal que la suma del volumen de los productos de los paquetes asociados a las entregas del repartidor para la fecha `fecha` más el volumen de los productos de `pq` sea menor o igual a `r.capacidad`

`calculaSueldo(doc:String, inicio:date, fin:date):float`	
Descripción	Calcula el sueldo de un empleado, tomando en cuenta que para los oficinistas se tiene el sueldo base más un bono del 10% si superan la meta de reclamos medios o graves y para los repartidores se tiene el sueldo base y 5 pesos por cada kilómetro recorrido
Parámetros	`String doc` identificador del empleado `Date inicio` fecha de inicio del mes de cálculo `Date fin` fecha de fin del mes de cálculo
Precondiciones	Existe un Empleado con identificador `doc`
Postcondiciones	Para el `Empleado e` con identificador `doc` • Si es `Oficinista`, se toma el sueldo base y se le suma un 10% si la cantidad de `Reclamos` graves o medios asociados entre `inicio` y `fin` supera `e.meta` • Si es `Repartidor`, se toma el sueldo base y se le agrega 5 pesos por cada kilómetro recorrido en las entregas asociadas entre `inicio` y `fin`

Se pide:

a) Realizar los Diagramas de Comunicación correspondientes a las operaciones especificadas. Las fechas pueden ser comparadas con los operadores básicos (<, >, =, >=, <=, etc.) como cualquier tipo primitivo. Existe un solo controlador con una colección global de clientes, empleados, productos, reclamos y paquetes.

b) Realizar el Diagrama de Clases de Diseño (DCD) resultante de los diagramas anteriores. Definir solo las relaciones y dependencias utilizadas en los diagramas de comunicación

<u>Problema 3</u> (35 puntos)

La empresa FingIA, líder en el desarrollo de sistemas de automatización basados en inteligencia artificial, ha encargado la creación de un sistema avanzado para gestionar proyectos que utilizan agentes inteligentes. Cada proyecto involucra múltiples agentes (llamados AgentesIA) que recopilan datos del entorno a través de sensores especializados y los procesan utilizando estrategias personalizadas. Estas estrategias permiten a los agentes realizar acciones complejas, como analizar patrones, activar dispositivos o generar reportes detallados. El sistema debe ser capaz de manejar grandes volúmenes de datos provenientes de sensores, asociarlos a los agentes correspondientes y ejecutar las estrategias asignadas de manera eficiente. Para lograr esto, se han definido dos componentes principales, `GestorDeColecciones`, encargado de registrar y administrar los proyectos, agentes y sensores disponibles, y `GestorDeOperaciones`, el núcleo del sistema, responsable de activar los distintos agentes. El diagrama de la siguiente página describe la estructura del sistema, destacando las relaciones entre los agentes, sensores, estrategias y los gestores encargados de la coordinación.

Se pide:

a) Como supervisor del equipo de desarrollo, ha liderado la creación del diseño preliminar. Sin embargo, algunos miembros del equipo han planteado ciertas dudas sobre la estructura propuesta. En particular, sugieren que los componentes `GestorDeColecciones` y `GestorDeOperaciones` deberían implementarse como singleton. ¿Está de acuerdo con el equipo? Justifique brevemente su respuesta.

b) ¿Qué patrón de diseño se aplica en la relación entre `AgenteIA` y las clases que implementan `Algoritmo`? Explique cómo funciona.

c) Implemente los archivos `Algoritmo.h`, `Actuador.h`, `Proyecto.h`, `AgenteIA.h` y `GestorDeColecciones.h`, teniendo en cuenta su respuesta en la parte a). Además, implemente el método `ejecutarEstrategias` de la clase `GestorDeOperaciones`, lo que además implica implementar el método `ejecutarEstrategiaAgentes` de la clase `Proyecto`, teniendo en cuenta que se debe iterar sobre todos los agentes asociados al proyecto cuya id se pasa como parámetro en `ejecutarEstrategias` y para cada agente, debe obtener los datos de los sensores asociados (utilizando el método `obtenerMediciones` que ya está implementado) y ejecutar el algoritmo correspondiente.

Observaciones: Utilice colecciones paramétricas (contenedores STL). No incluya directivas al precompilador (#include, etc). Utilice solamente las operaciones que se encuentran en el diagrama.