TRABAJO PRÁCTICO Nº 1

Resolvé los ejercicios utilizando diagramas de clases UML y Nassi-Shneiderman. Asegurate de leer al menos dos veces los enunciados antes de intentar confeccionar las soluciones.

ENUNCIADOS

1) El sitio de cursos online **"ORTDemy"** nos encomienda un sistema para manejar su plataforma, la cual, en primer lugar, consta de usuarios con ID, nombre, mail y si es o no becado. A su vez, en el sitio podemos encontrar categorías, de las cuales interesa conocer su ID y su nombre. Cada categoría cuenta con diferentes cursos, de los cuales se conoce su ID, su título, su precio y su valoración en estrellas. Además, se sabe quién es el autor de dicho curso, el cual debe ser un usuario del sitio.

Dentro del curso se encuentran las diferentes lecciones, de las cuales interesa saber su nombre, su duración en minutos y su tipo (texto, video o recurso). Por último, cada curso consta de una lista de usuarios suscriptos al mismo con la limitación de que no puede haber más de 5 alumnos con condición de becados en el mismo.

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) La explotación del método `suscribirseACurso` el cual recibe un ID de usuario y un ID de curso y debe retornar alguno de los siguientes resultados:

- **CURSO_INEX**: El curso no existe.
- **USUARIO_INEX**: El usuario no existe.
- **YA_SUSCRIPTO**: El usuario ya estaba suscripto en el curso.
- **ES_AUTOR**: El usuario que intenta suscribirse es el autor del curso.
- **MAX_BECADOS**: El usuario es becado y el curso ya cuenta con el máximo de becados posible.
- **SUSCRIPTO_OK**: El usuario se suscribió exitosamente al curso. Debe guardarse en la lista de suscriptos.

2) Un prestigioso hotel, del que sabemos el nombre, la dirección y las habitaciones que tiene, nos encomienda un sistema. De cada habitación sabemos el número, el precio por día y, si está ocupada, el nombre del cliente y la cantidad de días de estadía. Cada habitación puede tener adicionales, los cuales tienen un precio, fecha de prestación y un tipo (Desayuno, Room Service o Traslado).

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) El constructor parametrizado de la clase `Habitacion`, que recibiría como parámetros el número y el precio por día.

C) La explotación del método `calcularTotal` de la clase `Habitacion`, que devuelve el total en base al valor de la habitación por los días que se hospede el cliente y los adicionales pedidos.

D) La explotación del método `obtenerHabitacionesDisponibles` de la clase `Hotel` que debe devolver una lista con aquellas habitaciones que no estén ocupadas en la actualidad.

E) La explotación del método `realizarCheckout` de la clase `Hotel`, que recibe como parámetro el número de habitación y debe hacer lo siguiente:

I. Verificar que la habitación existe y esté ocupada actualmente.

II. Devolver el monto a abonar para dicha habitación.

El método devolverá `-1` si no se pudiera realizar el checkout[1] (por no encontrar la habitación enviada como parámetro o no estar ocupada).

3) En el instituto ORT se suelen realizar informes sobre el estado de situación de las carreras.

Se cuenta con toda la lista de alumnos que están cursando cada carrera. De cada alumno se sabe su nombre, su mail y todas las materias cursadas que tiene en su historial. De cada materia cursada se sabe su nombre, el cuatrimestre en formato `String` (por ejemplo, `"2019-02"` correspondería al segundo cuatrimestre del 2019) y un estado con estas posibles opciones: EN CURSO, REPROBADA, APROBADA.

Uno de los informes tiene que ver con saber quiénes son los alumnos con mayor cantidad de materias aprobadas en la actualidad.

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) La explotación del método `getInformeMaxMaterias` de la clase `Carrera`, que debe crear y devolver un único que contenga como datos:

I. Cantidad de materias aprobadas

II. Lista con únicamente los nombres y mails de todos aquellos alumnos que sean quienes más materias tienen aprobadas.

4) Hace mucho tiempo, en una galaxia muy lejana, había muchos droides que formaban parte de la existencia cotidiana de todos. Algo particular unía a todos esos droides: la capacidad que tenían de auto repararse. Ante algún desperfecto, podían reemplazar la o las piezas que no estaban funcionando por otras (con el mismo nombre) para seguir operando.

Para esto, cada droide llevaba un registro detallado de las piezas que lo componían, manteniéndolas por separado entre piezas operativas y no operativas. Cada una de estas piezas tiene un nombre (`String`) que es el mismo en todos los droides (por ejemplo, *"Batería de litio"*, *"Sensor de proximidad"*, *"Visor nocturno"*, etc).

Siempre que un droide encontraba a otro droide fuera de servicio (luego de batallas o simplemente por el uso) lo registraba para así, de ser necesario, usar las piezas sanas de éste para repararse. Al necesitar una pieza buscaba entre los droides rotos que tenía registrados, chequeando si encontraba entre ellos las piezas sanas que necesitaba. De encontrarlas, reemplazaba sus piezas no operativas por las operativas encontradas en los otros droides.

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) La explotación del método `autoRepararse` de la clase `Droide`, que no recibe parámetros. Debe intentar reemplazar sus piezas no operativas por las piezas operativas que pudiera encontrar en alguno de los otros droides. Cada vez que una pieza se reemplaza la pieza no operativa original se descarta. Este método devuelve alguno de estos resultados:

[1] Se entiende por "checkout" al evento que ocurre cuando un cliente deja la habitación del hotel.

- **COMPLETAMENTE_OPERATIVO**: cuando todas las piezas del droide están operativas.
- **REPARACION_PARCIAL**: cuando quedan algunas piezas no operativas, pero alguna se pudo reemplazar.
- **REPARACION_IMPOSIBLE**: cuando no se logre reparar ninguna de las piezas no operativas que pudiera tener.

5) El Poder Judicial de la Nación (PJN) nos pidió desarrollar el sistema para verificar la inscripción de las personas en el Padrón Electoral Nacional. El Padrón tiene todas las escuelas donde se vota.

De cada escuela se sabe su nombre, su Domicilio (calle, código postal y provincia) y las mesas de votación que contiene. El nombre de una escuela no se repite para otras escuelas del padrón.

De cada mesa se sabe su número, las personas que votan en ella y quién es su presidente de mesa (que debe estar además dentro de sus votantes). El número de mesa es único para todo el padrón.

De cada persona se sabe su nombre, apellido, DNI, género (Masculino, Femenino u Otro), fecha de nacimiento y Domicilio (calle, código postal y Provincia).

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) La explotación del método `designarPresidenteDeMesa` de la clase `Escuela` que recibe una mesa y una persona para designarla como su presidente de mesa. Debe tenerse en cuenta que una persona no puede votar ni ser presidente en más de una mesa. En caso de estar como votante y/o presidente en otra mesa deberá quitarse de la misma. Si la mesa destino ya tenía presidente designado deberá reemplazarlo.

C) La explotación del método `obtenerInforme` de la clase `Escuela` que no recibe parámetros y devuelve (no muestra por consola) la siguiente información de todas las mesas y cada una de las personas que votan en ellas.

- Número de la Mesa: este dato quedará repetido en la lista por cada persona encontrada que vote en la misma mesa.
- Número de orden de la persona en la mesa.
- DNI de la persona.
- Apellido y Nombre de la persona.

Ejemplo a devolver:

Mesa 001	Mesa 001	Mesa 002	Mesa 002	Mesa 002	Mesa 003
1	2	1	2	3	1
12345678	87654321	12341234	11122233	3322211	65498712
Perez Juan	Torres Ana	Gomez Ricardo	Bergara Andrea	Bolaños Roberto	Tripero Carlos

6) Las empresas suelen tener a una carrera tan prestigiosa como la de ORT como referencia para encontrar posibles candidatos a contratar cada vez que necesitan nuevos empleados en su organización. Para el mecanismo de selección de los posibles candidatos, se cuenta con toda la lista de alumnos que están cursando la carrera. De cada alumno se sabe su nombre, su mail y todas las materias que tiene aprobadas. De cada materia se sabe su nombre y la nota final obtenida.

Para que un candidato sea considerado en una búsqueda debe tener un mínimo de 5 materias aprobadas. La empresa que busca candidatos establece cuál es el promedio mínimo de notas para que un alumno pueda ser considerado candidato.

Así mismo, por políticas internas de ORT, en ningún momento la lista de posibles candidatos que se proponen puede superar los 20 alumnos.

Por ejemplo, si la empresa está buscando alumnos de promedio 7 o superior, se deberán considerar solo aquellos alumnos que posean al menos 5 materias aprobadas y entre todas las materias que tengan aprobadas su promedio sea de al menos 7. Pero nunca se devolverán más de 20 candidatos, aunque los pudiera haber.

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) La explotación del método `obtenerCandidatos` de la clase `Carrera`, que recibiendo como parámetro el promedio mínimo deseado, debe crear y devolver una lista con los nombres y mails de los primeros 20 alumnos que cumplan con los requisitos pedidos. Si no hubiera ningún candidato posible deberá devolver una lista vacía. Si los posibles candidatos no llegarán a ser 20, deberá devolver los que se hayan encontrado.

Nota: notar que no se debe devolver una lista de objetos `Alumno` sino una lista que contenga únicamente el nombre y el mail de los alumnos que sean candidatos exclusivamente.

7) Un portal de venta de libros llamado **"Amazonia"** requiere un sistema para gestionar sus ventas. El portal posee un registro de todos sus clientes y también de los productos que comercializa.

De cada cliente se sabe: nombre, apellido, dirección, email, teléfono y su historial de pedidos que realizó. El historial de pedidos consta de un elemento por cada pedido que realiza un cliente. Cada uno de estos pedidos tiene la fecha y hora en la que se realizó (formato `"dd/mm/aaaa"`), un estado (PENDIENTE, CONFIRMADO, EN_CAMINO y ENTREGADO) y una lista donde cada elemento posee el producto pedido y la cantidad pedida.

La clase `Cliente` posee un método ya provisto llamado `getPedidoPendiente` que devuelve un objeto de la clase `Pedido` con el pedido que posea el cliente en curso (o `null` si no tuviera ninguno). Cada cliente puede tener un solo pedido en estado PENDIENTE a la vez.

De cada producto, sabemos el nombre, el precio unitario y la cantidad que hay en stock actualmente.

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) La explotación del método `procesarPedido` de la clase `Portal`, que recibe como parámetro una instancia de un `Cliente` y debe:

I. Obtener el pedido pendiente del cliente recibido.

II. Verificar por cada producto del pedido si posee stock suficiente para realizarse.

III. En caso de no poseer stock suficiente, removerlo del pedido.

IV. Devolver una lista de aquellos productos que no pudieron procesarse en el pedido por falta de stock.

C) La explotación del método `confirmarPedido` de la clase `Cliente` que no recibe parámetros y debe:

I. Descontar la cantidad pedida del stock de cada producto del pedido pendiente.

II. Cambiar el estado del pedido pendiente a CONFIRMADO.

8) La empresa "**OrtParking**" cuenta con varios garajes distribuidos por distintos barrios de la ciudad. Cada garaje se identifica por un código alfanumérico. Lleva dos listas independientes: una registra todos los vehículos estacionados y la otra los vehículos retirados. El garaje cuenta con un tablero donde se guardan las llaves de los vehículos que están estacionados. Para identificar a qué vehículo pertenece una llave, cada una de estas tiene pegada una etiqueta con la patente del vehículo.

El tablero cuenta con el método `devolverLlave(...)` (ya desarrollado) que devuelve la llave correcta a partir de la patente. De no encontrarla devuelve `null`.

También cada vehículo cuenta con un método ya desarrollado que permite devolver la cantidad de meses que adeuda llamado `getMesesAdeudados()`.

Cada vehículo tiene su patente y una lista con las personas autorizadas a retirarlo del garaje. De estas personas sabemos su DNI y el nombre completo.

Tanto al entrar como al salir del garaje el vehículo debe quedar en el registro que corresponda. La llave debe quedar o bien en el vehículo o bien en el tablero, según el caso.

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) El método `estacionarVehiculo` de la clase `Garaje` que recibe como parámetros la patente del vehículo a ingresar. El método debe devolver alguno de los siguientes resultados:

- `VEHICULO_NO_HABILITADO`: cuando no se encuentra al vehículo en el registro.
- `VEHICULO_YA_ESTACIONADO`: cuando ya figura previamente estacionado.
- `NO_ESTACIONA_ADEUDA`: si el vehículo tiene más de tres (`3`) meses adeudados.
- `INGRESO_OK`: cuando se cumplen todos los requisitos para ingresar al vehículo, habiendo hecho los cambios correspondientes.

C) El método `obtenerInformeEstadoGarajes` que debe devolver una lista detallando, para cada garaje, su código y la cantidad de vehículos estacionados.

D) El método `mostrarVehiculosSinLlave` que debe mostrar por pantalla, de todos los garajes, el código del garaje y las patentes de aquellos vehículos estacionados en él cuya llave no esté guardada en el tablero.

E) El método `esPersonaAutorizada`, que recibe el DNI de una persona y verifica si la misma está autorizada para retirar algún vehículo estacionado en el garaje. Devuelve un `boolean`.

9) La famosa aplicación de música online **"ORTify"** cuenta con un registro de artistas, de los cuales se conoce su ID, su nombre y su listado de canciones. De cada canción se conoce su ID, nombre y duración en segundos. La aplicación cuenta además con un listado de usuarios, de los que se sabe su nombre de usuario (unívoco) y su estado (habilitado, prueba gratis o suspendido). Cada canción contiene un listado con los usuarios que le dieron 'like' a esa canción.

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) La explotación del método `mostrarDuracionPromedio` que permita mostrar el promedio en minutos y segundos que duran las canciones de un artista. (Ejemplo: `"3 minutos, 24 segundos"`, no `"204 segundos"`).

C) La explotación del método `esFanDestacado` de la clase `Artista` que recibe como parámetro la instancia de un usuario y se devuelve si se trata de un fan destacado o no, según si el usuario ha dado 'like' en al menos la mitad de las canciones del artista.

D) La explotación del método `primeras5Canciones` que devuelva una lista con las primeras `5` canciones de un artista. Si no llega a tener `5`, devolver las que tenga, siempre en una nueva lista.

E) La explotación del método `borrarUsuario` que reciba un nombre de usuario y lo elimine tanto de la lista de usuarios de la aplicación como en todas las canciones en donde haya dado 'like'. El método retorna la instancia, si se logró borrar, o `null` en caso contrario.

10) La casa de juegos "**OrtLand"** decide informatizar su negocio y nos solicitan manejar sus mesas de juegos de cartas. De cada juego conocemos su nombre y una lista con las mesas asignadas. De cada mesa sabemos el número de mesa, cantidad máxima de participantes que admite y una lista con los participantes que están jugando. De cada participante se conoce su nombre y edad.

En cada mesa se cuenta con un mazo de cartas españolas. De cada carta del mazo conocemos su palo (ORO, BASTO, COPA, ESPADA) y su número (no hay comodines). No todas las mesas necesariamente utilizarán todas las cartas de la baraja. Cada jugador tiene una lista con las cartas que tiene en su mano en ese momento.

Basado en el enunciado descripto, realizá:

A) El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.

B) La explotación del método `obtenerDisponibilidadJuegos` que devuelve una lista de elementos que contengan el nombre del juego y la cantidad de lugares disponibles de cada uno.

C) La explotación del método `repartirCartas` que reparta una carta a cada jugador siguiendo una ronda hasta llegar a darle `4` a cada uno. Debe haber por lo menos dos jugadores en la mesa. Verificar no quedarse sin cartas para repartir. Devuelve `true` o `false` dependiendo de si pudo repartir las cartas o no.

D) La explotación del método `acomodarJugador` de la clase `OrtLand` que reciba el nombre de un jugador, su edad y el nombre del juego en el que quiere participar. Debe intentar sumarlo a la mesa que tenga la mayor cantidad de jugadores en donde aún tenga lugar disponible. Si se encuentra lugar, la mesa deberá agregarlo a su lista de jugadores.

Devolverá uno estos posibles resultados:

- `SIN_DISPONIBILIDAD`: cuando no se haya podido agregar al jugador a una mesa existente.
- `JUEGO_NO_ENCONTRADO`: cuando no se encuentre el juego con el nombre provisto.
- `ASIGNACION_OK`: cuando el jugador haya sido asignado.