GUÍA PRÁCTICA DE COLECCIONES DE OBJETOS

Resolvé los ejercicios utilizando diagramas de clases UML y el lenguaje Java. Asegurate de leer al menos dos veces los enunciados antes de intentar confeccionar las soluciones.

ENUNCIADOS

1) Generá la clase `Persona` con, al menos, los atributos DNI (`String`), nombre, apellido y domicilio (calle, altura y barrio). Debe tener un constructor parametrizado que inicialice todos sus atributos.

A continuación, definí la clase `Agenda`, que contendrá una colección de personas (con `ArrayList`) y los siguientes métodos (debés deducir los parámetros y los valores de retorno de cada uno):

- **A)** `listarPersonas`, que muestre en la consola un listado con todos los datos de cada persona de la colección.
- **B)** `devolverUltimo`, que **retorne** (no muestre por consola) a la última persona de la colección (o retorne `null` si está vacía la colección).
- **C)** `buscarPersona`, que busca en la colección a la persona que posea el DNI recibido por parámetro y la retorne (o retorna `null` si no se encuentra).
- **D)** `agregarPersona`, que recibe por parámetro todos los datos necesarios para construir una `Persona`. Se construirá y agregará una persona a la colección con esos datos, siempre y cuando no exista ya alguien con ese DNI en tal colección. Retorna un `boolean` indicando si se pudo agregar o no.
- **E)** `removerPersona`, que recibe por parámetro un DNI. Debe borrar de la colección a la persona con ese DNI y retornarla (o retornar `null` si no existe tal persona).
- **F)** `modificarDomicilio`, que recibe por parámetro un DNI y un domicilio. Debe modificar el domicilio de la persona encontrada con ese DNI por el recibido por parámetro. Retorna un `boolean` indicando la operación fue exitosa o no.
- **G)** `obtenerPorBarrio`, que **retorne** (no muestre por consola) a todas las personas que vivan en un barrio determinado por parámetro.
- **H)** `vaciar`, que elimina a todas las personas de la colección, una por una. (Sin usar el método `clear` de la clase `ArrayList`).

Crear una clase con el método `main` para testear todos y cada uno de los métodos de la agenda.

2) Una inmobiliaria, exclusiva de los barrios de Recoleta, Palermo y Belgrano, tiene a la venta propiedades, de las que se sabe su domicilio (`String`), su precio y su tipo (Depto, Casa o PH).

Basado en el enunciado descripto, realizá:

- **A)** El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.
- **B)** La implementación del método `promedioDePrecio` de la clase `Inmobiliaria`, el cual debe devolver cuánto cuestan en promedio todas las propiedades.
- **C)** La implementación del método `propiedadesMasEconomicas` de la clase `Inmobiliaria`, el cual debe devolver una lista de propiedades cuyo precio esté por debajo del promedio general.
- **D)** La implementación del método `cantidadDePropiedadesSegunTipo` de la clase `Inmobiliaria`, que recibe como parámetro un tipo de propiedad y debe devolver la cantidad de propiedades de tal tipo.

3) Una academia ofrece una serie de cursos de diversas disciplinas. De cada curso se sabe su código alfanumérico que lo identifica unívocamente, su título, la cantidad de horas de duración y los alumnos que posee inscriptos. De cada alumno se sabe su nombre y su DNI. Así mismo, cada alumno posee una lista con las evaluaciones rendidas. De cada evaluación se sabe su tipo (Primer examen, Segundo examen o Final) y la nota obtenida.

Basado en el enunciado descripto, realizá:

- **A)** El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.
- **B)** La implementación del método `calcularPromedio` de la clase `Alumno`, que debe calcular el promedio de dicho alumno entre todas las evaluaciones que haya rendido.
- **C)** La implementación del método `buscarAlumnosPorEncimaDe` de la clase `Curso`, que recibe como parámetro un valor que representa un promedio, debiendo crear y devolver una lista de aquellos alumnos que hayan realizado al menos 2 evaluaciones y cuyo promedio de notas supere al promedio enviado por parámetro.

4) Una conocida marca de automóviles posee una lista con todas las concesionarias en donde posee vehículos a la venta. Cada concesionaria tiene su nombre y una lista con todos los vehículos a la venta. De cada vehículo se sabe su patente, precio y tipo (auto, camioneta o utilitario).

Basado en el enunciado descripto, realizá:

- **A)** El diagrama de clases que lo modelice, con sus relaciones, atributos y métodos.
- **B)** La implementación del método `mostrarVehiculos` de la clase `Concesionaria`, que reciba un tipo de vehículo y muestre las patentes y precios de todos los vehículos de ese tipo.
- **C)** La implementación del método `mostrarVehiculos` de la clase `Marca`, el cual debe mostrar las patentes y precios de todos los vehículos que tenga a la venta.
- **D)** La implementación del método `obtenerConcesionariaMaxVeh` de la clase `Marca`, que devuelva una lista de la o las concesionarias con la mayor cantidad de vehículos a la venta.
- **E)** La implementación del método `borrarVehiculo` de la clase `Marca`, que reciba una patente y elimine el vehículo asociado a esa patente de la concesionaria donde se encuentre.
- **F)** La implementación del método `cambiarVehiculoDeConcesionaria` de la clase `Marca`, que reciba una patente y una concesionaria. Debe mover el vehículo asociado a esa patente a la concesionaria indicada.

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

E) La implementación 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).

6) ClonFlix es un nuevo sitio en el cual los clientes pueden ver películas online. Para diferenciarse de su competencia, sus clientes pueden optar por dos tipos de servicios. El servicio Standard y el servicio Premium. El servicio Premium da acceso a películas más nuevas (generalmente estrenos de ese mismo año).

Para ver cualquier película el cliente (del cual se sabe su DNI y su nombre) no debe poseer deuda. Para ver contenido Premium, el cliente debe estar suscripto al servicio Premium.

Debe existir un método en la clase `Cliente` llamado `obtenerSaldo` que devuelva el saldo que adeuda ese cliente (el retorno de dicho método puede ser inventado). Si el saldo es positivo, entonces el cliente es deudor.

Por último, al sitio le interesa llevar un registro de las películas que miró cada cliente.

Basado en el enunciado descripto, realizá:

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

B) La implementación del método `verPelicula` perteneciente a la clase `ClonFlix`, el cual recibe como parámetros el DNI de un cliente y el nombre de la película a ver. Debe devolver alguno de los siguientes resultados:

- `CLIENTE_INEXISTENTE`: El cliente no existe.
- `CONTENIDO_INEXISTENTE`: La película no existe.
- `CLIENTE_DEUDOR`: El cliente posee deuda.
- `CONTENIDO_NO_DISPONIBLE`: La película es Premium y el cliente no está abonado a dicho servicio.
- `ALQUILER_OK`: La película puede verse sin inconvenientes. La película debería quedar registrada en el historial de películas vistas del cliente.

C) La implementación del método `darDeBaja` de la clase `ClonFlix`, que reciba un DNI y elimine de la lista al cliente asociado a tal DNI.

7) Las empresas suelen tener a una carrera tan prestigiosa como la del INSPT 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.

En ningún momento la lista de posibles candidatos que se proponen puede superar los 20 alumnos.

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

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 implementación 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.

8) 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 implementación 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:

- **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.

9) La empresa **FooParking** 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`.

10) La famosa aplicación de música online **Clonify** 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 implementación 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 implementación 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 implementación del método `primeras5Canciones` que devuelva una lista con las primeras `5` canciones de un artista. Si no llega a `5`, devolver las que haya, en una nueva lista.

E) La implementación 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.