TRABAJO PRÁCTICO Nº4

Programación Orientada a Objetos – Explotación de Métodos

Esta guía de ejercicios está diseñada para practicar el diseño de clases y sus relaciones usando colecciones de objetos (que permiten relaciones de uno a muchos) y enumerados.

Los ejercicios que te presentamos te ofrecerán un panorama más amplio en las posibles relaciones que puede haber entre clases, así que observá con atención cuáles son las responsabilidades de cada clase y con quiénes se relacionan realmente.

Verás también que se hace mayor hincapié en la explotación de los métodos: todos estos ejercicios pueden llevarse a código, y si bien no es obligatorio, sí es recomendable.

Casi todos los ejercicios de esta guía fueron exámenes en cuatrimestres previos, y están al nivel de lo que esperamos conseguir para que puedan pasar a la siguiente materia.

Debés resolver obligatoriamente 5 de ellos, desde el ejercicio 3 en adelante (los anteriores los plantearemos juntos en clase).

Guía de ejercicios:

Ejercicio 1.

Una institución educativa ofrece cursos técnicos pertenecientes a diferentes disciplinas y oficios (mecánica, electricidad, plomería, etc.).

Cada curso requiere una escolaridad mínima necesaria, y puede ofrecerse en diversos momentos del año. A cada una de estas “tandas” se la llama Cursada.

Para cada cursada se indica fecha de apertura, cupo de inscriptos y quién es el profesor que lo dicta.

Para inscribirse en un curso, éste debe tener cupo disponible. Quienes quieran hacerlo deben estar primero registrados en la institución y cumplir con los requisitos de escolaridad del curso.

Tomando en cuenta el enunciado descripto:

a) Diseñá el diagrama de clases correspondiente.
b) Desarrollá el algoritmo asociado al método *registrarAlumno(...)* de la clase ***Institución***. Este método recibe como parámetros los datos de un alumno (Documento, Nombre, Apellido y Escolaridad). Debe devolver como parámetro un valor booleano con *true* si logró registrar al alumno y con *false* en caso de que el alumno ya exista registrado en la Institución.

   Pista: *Creá el método buscarAlumnoPorDocumento(...) de la clase* **Institucion**, *al cual pasándole un dni nos devolverá la instancia de* **Alumno**, *o un null si no lo encuentra.*

c) Desarrollá el algoritmo del método *inscribirAlumnoEnCurso(...)* el cual pertenece a la clase ***Institución*** y recibe como parámetros el dni del ***Alumno*** que desea inscribirse

y la instancia de la ***Cursada*** en la cual éste desea inscribirse. El método devuelve un valor booleano indicando si la inscripción fue exitosa.

Ejercicio 2.

Una biblioteca tiene una vasta colección de libros. Cada libro tiene un título, un género (novela, teatro, poesía, ensayo u otros), editorial, año y autor. De cada libro puede haber más de un ejemplar y cada ejemplar tiene un identificador, una fecha de préstamo, y un estado (en biblioteca, prestado o en reparación).

Las editoriales tienen nombre y país, y los autores tienen nombre, nacionalidad y fecha de nacimiento.

De los lectores se sabe su dni, su nombre, su dirección y su teléfono. Cada Lector puede tener hasta un máximo de 3 préstamos vigentes. Cuando se realiza el préstamo de un ejemplar se registra la fecha de retiro y el ejemplar que se está retirando. El préstamo expira a los 7 días, y por cada día de retraso se impone una "multa" de dos días sin posibilidad de retirar un nuevo libro.

Tomando en cuenta el enunciado descripto, realizar:

a) El diagrama de clases correspondiente.
b) La explotación del método *prestar(...)* de la clase ***Biblioteca***, más todos los algoritmos derivados que pudieran surgir. Este método recibe la instancia del ***Libro*** requerido y del ***Lector*** que lo pide, y devuelve un valor *enumerado* con los siguientes posibles valores y significados:
    - **PRESTAMO_EXITOSO**: El libro es prestado exitosamente (en este caso, antes de salir del método hay que marcar el ***Ejemplar*** como prestado, entregándoselo al lector , y procesar y registrar el ***Prestamo***).
    - **NO_HAY_EJEMPLARES**: No hay ejemplares disponibles del libro.
    - **TOPE_PRESTAMOS_ALCANZADO**: El lector ya posee tres libros en préstamo.
    - **MULTA_VIGENTE**: El lector posee una multa en vigencia.

Ejercicio 3.

ORTFlix 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).

Existe un método ya provisto de la clase ***Cliente*** que se llama *obtenerSaldo(...)* que devuelve el saldo que adeuda ese cliente (dicho método no debe diagramarse).

Para ver una película el cliente no debe poseer deuda.

Para ver contenido Premium, debe estar suscripto al servicio Premium.

Por último, se va llevando un historial por cliente de todas las películas que miró.

Basados en el enunciado descripto, realizá:

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

b) La explotación del método *verPelícula(...)* perteneciente a la clase ***Empresa***, el cual recibe como parámetros el dni del cliente y el nombre de la película a ver. Devuelve lo siguiente:
   - **ALQUILER_OK**: La película puede verse sin inconvenientes (en este caso la película debe quedar registrada en el historial del cliente).
   - **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.

c) La explotación del método *verSerie(...)*, también perteneciente a la clase ***Empresa***. Este método recibe como parámetros la instancia del ***Cliente*** que desea ver la serie, y una instancia de ***CapituloSerie*** cuyos atributos son el nombre de la serie, el número de temporada y el número de capítulo. El método devuelve:
   - **ALQUILER_OK:** La serie puede verse sin inconvenientes (en este caso, los datos completos del capítulo deben quedar registrados en el historial del cliente).
   - **CLIENTE_INEXISTENTE**: El cliente no existe.
   - **CONTENIDO_INEXISTENTE**: La combinación de serie / temporada / capítulo no existe.
   - **CLIENTE_DEUDOR:** El cliente posee deuda.
   - **CONTENIDO_NO_DISPONIBLE**: La serie/temporada/capítulo que se desea visualizar es Premium y el cliente no está abonado a dicho servicio.

Ejercicio 4.

Una inmobiliaria tiene una lista con todos los barrios en donde posee propiedades a la venta. Cada barrio se tiene su nombre y una lista con todas las propiedades a la venta, y de cada propiedad se sabe su dirección, su precio y su tipo (DEPARTAMENTO, CASA ó PH).

Tomando en cuenta el enunciado descripto, realizá:

a) El diagrama de clases que lo represente.

b) La explotación del método *obtenerBarrioMaxProp()* de la clase ***Inmobiliaria***, que devuelva una lista de él o los barrios con la mayor cantidad de propiedades en cartera.

c) la explotación del método *mostrarPropiedades(...)* de la clase ***Barrio***, que dado un tipo de propiedad, muestre las direcciones y precios de todas las propiedades de ese tipo que posea ese barrio.

d) La explotación del método *mostrarPropiedades()* de la clase ***Inmobiliaria***, el cual debe mostrar la dirección y el precio de todas las propiedades que posea a la venta.

Ejercicio 5.

La empresa “Abarajame la Plancha” se dedica a la venta de artículos para el hogar y cuenta con sucursales que están distribuidas por todo el país.

Cada sucursal posee un nombre y sabe qué productos comercializa y cuál es el stock actual de cada uno. Además, cada sucursal posee una lista de aquellas sucursales cercanas a ella.

Cuando un cliente decide comprar de un producto en una sucursal, ésta verifica si posee stock del producto requerido y en dicho caso se registra la venta sin problemas. Pero en caso de no poseer stock, se consulta si en alguna de las sucursales cercanas hay stock; de ser así se informa al cliente a qué sucursal debe dirigirse (pero no se registra la venta).

Registrar la venta implica simplemente incrementar un contador de ventas realizadas que habrá por cada producto de cada sucursal, descontando además la cantidad vendida del stock.

Basándote en el enunciado descripto:

a) Desarrollá el diagrama de Clases que lo represente.

b) Desarrollá el método *vender(...)* de ***Sucursal***. El método recibe como parámetro una instancia del ***Producto*** a vender y la cantidad requerida. Debe devolver un string con uno de los siguientes mensajes posibles:

   1. "Venta realizada." *(en este caso debe descontar el stock y sumar uno a la cantidad vendida de dicho producto en esa sucursal).*
   2. "El producto que desea adquirir se encuentra en la Sucursal NOMBRE_SUCURSAL."
   3. "No hay stock del producto requerido ni en esta sucursal ni en ninguna de las cercanas."

Ejercicio 6.

La agencia de viajes "Tu Viaje Soñado" se dedica a ofrecer diferentes paquetes turísticos a diversas ciudades del mundo. Cada paquete tiene un nombre de fantasía, la ciudad que visita y la duración en días concretos de estadía en esa ciudad (sin contar los tiempos de viaje aéreo). Además, el paquete está compuesto por el pasaje aéreo, el hotel y las excursiones que incluye.

Del pasaje aéreo se detalla aeropuerto de origen y destino (una ciudad puede tener más de un aeropuerto), el tipo (turista, ejecutiva o primera clase) y el costo del viaje ida y vuelta por base doble (dos personas).

Del hotel se detalla nombre, dirección, teléfono, la cantidad de estrellas que tiene (1 a 5), el tipo de servicio que se incluye en el paquete (sólo desayuno, media pensión, pensión completa o todo incluido) y el precio por día para base doble.

De cada excursión se cuenta con una breve descripción y el precio para base doble.

Tomando en cuenta el enunciado descripto, realizá:

a) El diagrama de clases, declarando las relaciones entre las clases, sus atributos y sus métodos.
b) La implementación del **constructor parametrizado** de la clase ***Hotel.***
c) La explotación del método *obtenerPaquetesPorCiudad(...)* de la clase ***Agencia***, que recibe una instancia de ***Ciudad***. Devuelve una lista con los paquetes que se ofrezcan para esa ciudad.
   Nota: *Si la lista está vacía significa que no hay paquetes para esa ciudad.*
d) La explotación del método *mostrarPaquetesPorCiudad(...)* de la clase ***Agencia***, que recibe una instancia de ***Ciudad***. Debe mostrar por pantalla el detalle completo de

todos los paquetes que se ofrezcan. De cada paquete se detallan los datos del paquete en sí, del pasaje aéreo, del hotel y de las excursiones con las que cuenta.
Nota: *Si no hubiera paquetes para esa ciudad debería aparecer la leyenda "No existen ofertas de paquetes para la ciudad deseada en la actualidad".*

e) La explotación del método *calcularValor(...)* de ***Paquete*****,** que recibe la cantidad de pasajeros (es siempre mayor a 0). Devuelve el valor total del paquete para esa cantidad de pasajeros.
Nota: *Tomar en cuenta que si la cantidad es un número impar, el pasajero restante debe abonar el 70% del valor especificado para base doble.*

Ejercicio 7.

La Elaboradora "Mi Perro Bobby" se dedica a la fabricación de productos alimenticios para mascotas y animales de granja. Fabrica distintos tipos de alimento, algunos para perros, otros para gatos, otros para equinos, etc. Estos alimentos pueden ser para cachorros o adultos, y ser alimento light o no.

Asimismo, la fábrica lleva una lista con todas las materias primas que se utilizan para elaborar cada alimento. Cada Materia Prima está compuesta por un código único, un nombre y un número que indica la cantidad actual en stock.

Además del tipo de alimento, si es para cachorros o adultos y si es light, cada Alimento tiene un código y un nombre fantasía, más la especificación detallada de los ingredientes, con el código y la cantidad de cada materia prima necesaria para fabricar 100 kilos de alimento (valor constante y tope mínimo para la producción de una tanda de cualquier alimento).

Los alimentos se fabrican a partir de una Orden de Fabricación, en la cual figuran la fecha de creación, el alimento que se va a elaborar y cuántos kilos se desean producir.

Tomando en cuenta el enunciado descripto, realizá:

a) El diagrama de clases que modelice el enunciado, declarando las relaciones entre las clases, sus atributos y sus métodos.
b) La implementación del constructor de la clase ***Elaboradora***, que recibe como parámetro la fecha del día y la guarda en el atributo *fechaDeHoy*,
c) La explotación del método *verificarIngredientes(...)* de ***Elaboradora***, que recibe una instancia del ***Alimento*** a elaborar y los kilos a producir. Devuelve una lista de ingredientes faltantes, con el código de ***MateriaPrima*** y la cantidad requerida para fabricar el alimento en cuestión.
Nota: *Si la lista está vacía significa que no hay faltantes y que el alimento se puede elaborar.*

d) La explotación del método *crearOrdenFabricacion(...)* de ***Elaboradora*****,** que recibe la instancia del alimento a elaborar y los kilos a producir (que siempre debe ser mayor o igual al mínimo). Devuelve una ***OrdenFabricacion*** con la fecha del día, la instancia del Alimento y los kilos a producir.
Nota: *Cuando la cantidad requerida sea menor al mínimo, o cuando no haya materia prima suficiente para fabricar la cantidad requerida, el método debe devolver* null.

Ejercicio 8.

La empresa "MaradonaMejorQuePelé" se dedica al alquiler por hora de canchas de fútbol.

Cuando se contrata una cancha, la empresa se encarga de asignar un árbitro para ese partido.

Las canchas se pueden alquiler de a bloques de 1 hora, desde las 7am hasta la 1am.

De cada árbitro se saben nombre y teléfono, qué días y en qué rango horario puede dirigir, siempre en bloques de una hora. De cada bloque se sabe si está libre o no.

Además se lleva un registro de qué días y en qué horarios está reservada cada Cancha. De esa manera se puede saber cuál es la disponibilidad en caso de un nuevo pedido de reserva.

Al realizarse una nueva reserva, lo que se hace es:

1) Se le pregunta al usuario el día y el horario deseado.
2) Se verifica si hay alguna cancha disponible en el día y horario solicitado.
3) Se verifica que haya algún árbitro disponible para asignar.
4) En caso de que haya disponibilidad de cancha y de árbitro se le pide al usuario a nombre de quién debe quedar registrada la reserva y cuánto abonará en concepto de seña.
5) Luego se registra la reserva, asociándola a la cancha correspondiente en el día y horario elegido, guardando también a nombre de quién está hecha la reserva y cuánto pagó en concepto de seña.
6) Por último, se asigna el árbitro elegido, marcando en su agenda que ese horario ya lo tiene ocupado.

Nota: *en caso de que no ser posible hacer la reserva (por falta de cancha y/o árbitro) se le informa al usuario y termina ahí el proceso.*

Tomando en cuenta esto:

a) Confeccioná el diagrama de clases que modelice el enunciado, con sus relaciones, atributos y métodos.
b) Explotá el método *reservarCancha()* de la clase ***Empresa*** y que no recibe nada como parámetro (se encarga internamente de obtener los valores con los que debe trabajar) y devuelve un valor booleano con *true* en caso de haber podido realizar la reserva en forma exitosa y *false* en caso contrario. Para este punto, desarrollar los métodos privados que hagan falta para cargar la información por teclado.

Ejercicio 9.

El restaurant parrilla “Los Nenes” nos contrata para que le desarrollemos el sistema con el cual llevaría adelante toda la operatoria de su funcionamiento cotidiano.

La parrilla lleva un control de cada mesa que está siendo atendida (abiertas). De estas mesas se sabe cuántos comensales se están atendiendo, y hay una lista de todos los Pedidos que la mesa realizó a fin de calcular el importe a abonar al terminar la comida *(ver más abajo cómo está compuesto un Pedido)*.

También se sabe cuáles son las mesas que están disponibles (cerradas), o sea, que en este momento no poseen comensales y cuya lista de pedidos debe estar vacía.

La parrilla lleva un stock de todos los ingredientes que posee con un código único para cada ingrediente y una cantidad.

Así mismo, hay una lista de bebidas que la parrilla ofrece, cada una con su stock.

En la parrilla se ofrecen diversos platos; cada uno tiene una descripción, un precio y la especificación de los ingredientes que utiliza. De cada ingrediente tiene su código y la cantidad necesaria del mismo para la elaboración del plato. Por lo tanto, cada vez que se debe elaborar un plato se verifica que se posea existencia suficiente de cada ingrediente. Si este requisito puede cumplirse se descuentan esos ingredientes del stock y se comienza con su elaboración; si no es posible, se rechaza el pedido y se informa a la mesa. En el caso de tratarse de bebidas, simplemente se verifica que haya en existencia y se descuenta para poder llevarla a la mesa.

Cada Pedido está compuesto por un número que identifica a la mesa que pertenece, una lista de platos y una lista de bebidas. Tené en cuenta que puede llegar a haber más de un pedido para la misma mesa (por ejemplo luego de hacer el pedido inicial, se realiza un segundo pedido con los postres, etc.)

Tomando en cuenta el enunciado descripto, realizá:

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

b) La explotación del método *realizarPedido(...)* de la clase ***Restaurant***. Este método recibe una instancia de la clase ***Pedido***, la cual está compuesta por un número de mesa, una lista de platos y una lista de bebidas (alguna de las dos listas podría estar vacía, pero nunca las dos al mismo tiempo).
   El método debe realizar lo siguiente:
   - ✓ Verificar que la mesa del pedido exista y esté efectivamente “abierta” (es decir que tiene comensales sentados).
   - ✓ Verificar que los platos que se hayan pedido posean stock suficiente de ingredientes para su elaboración. Remover de la lista del Pedido aquellos platos que no se puedan realizar por faltante de ingredientes y descontar los ingredientes correspondientes de aquellos platos que sí poseen stock suficiente.
   - ✓ Verificar que las bebidas que se hayan pedido estén en stock. Remover de la lista de bebidas del Pedido aquellas que no se posean en stock y descontar del stock aquellas que sí se poseen en stock.
   - ✓ Cargar a la mesa el Pedido actualizado con aquellos platos y bebidas que sí se pueden entregar.

El método debe devolver un valor numérico entero con los siguientes significados:

1. La mesa no existe o no está abierta.
2. El pedido pudo realizarse pero con faltante de algunos platos.
3. El pedido pudo realizarse pero con faltante de algunas bebidas.
4. El pedido pudo realizarse pero con faltante de platos y bebidas.
5. El pedido se realizó con éxito.

c) Explotación del método *cerrarMesa()* de la clase ***Restaurant***. Este método recibe un número de mesa y devuelve un valor double con el importe a abonar por dicha mesa. Para obtener el importe, el método debe buscar la ***Mesa*** y recorrer su lista de ***pedidos***. Por cada ***Pedido*** ir sumando los valores de todos los platos y bebidas que posea. Una vez hecho esto, se debe cerrar la mesa, dejando su lista de pedidos vacía.
   Nota: *Si el número de mesa enviado no correspondiese a ninguna mesa, entonces el método devolverá un -1 como resultado.*

Ejercicio 10.

La empresa de juegos de entretenimiento “Jugate Conmigo” desea implementar un nuevo sistema informático para gestionar el uso de sus máquinas de juegos mediante tarjetas de precarga o pago anticipado.

Para usar los juegos, sus clientes deben adquirir una tarjeta en el local y cargar en ella una suma de dinero de la que luego, justo antes de jugar, se descuenta el precio de cada partida o sesión.

Cada tarjeta posee un código especial, alfanumérico, formado por el código del establecimiento, la fecha de emisión en formato AAAAMMDD y un número secuencial y único que se va incrementando con cada tarjeta agregada en el día (el valor inicial de este número es 1). Tiene también otro atributo para guardar un saldo en pesos que al principio vale 0. Al ser comprada, la empresa impone un monto mínimo de carga de $100 (cien pesos) pero luego se pueden cargar con cualquier monto. Las tarjetas no tienen fecha de vencimiento.

La empresa mantiene registradas sus tarjetas en dos lugares distintos: primero, tiene una lista donde están todas las tarjetas nuevas disponibles para la venta, y se asegura de que al comenzar cada día haya dos mil (2000) tarjetas nuevas disponibles.

Cuando una tarjeta se vende es movida a otra lista, donde están todas las tarjetas vendidas y activas del día. Si un jugador trae una tarjeta comprada previamente (por ejemplo de ayer o del verano pasado) e intenta usarla en un juego, deberá comprobarse que posea, además de saldo suficiente, el código alfanumérico propio del establecimiento. Si es así debe también agregarse a la lista de tarjetas activas. Otra forma de reactivar la tarjeta es agregarle saldo.

La empresa agrupa los juegos en dos tipos: electromecánicos (calesitas, autitos, cazapremios, etc.) o electrónicos (videojuegos o *flippers*). Para simplificar su administración cada grupo tiene un precio único.

Cuando un juego se instala se conecta a la empresa que lo controla para informarle sobre cada partida y otros estados (fallas, reinicios, necesidad de mantenimiento, etc.). Además, cada juego posee un código alfanumérico que es único y lo identifica, cuyos primeros dos caracteres indican si es un juego electromecánico (EM) o electrónico (EL) y un nombre fantasía, que es el que se le muestra al jugador.

Cuando un jugador quiere comenzar una partida pasa su tarjeta por el lector del juego. Entonces, el juego le envía a la empresa su propia información y la de la tarjeta, y la empresa le devuelve un enumerado con los siguientes valores y significados:

- TARJETA_INVALIDA: La tarjeta es inválida.
- MONTO_INSUFICIENTE: El monto es insuficiente.
- TRANSACCION_VALIDA: Transacción válida, puede jugar.

Sólo cuando la transacción sea válida se habilitará el juego; en los demás casos el intento de jugar debe ser rechazado.

Además, cada vez que sea válida la partida, la empresa descontará de la tarjeta el importe del juego, según el grupo al cual éste pertenezca, y guardará la información de la transacción (código y nombre del juego y número de tarjeta) en el historial del día. Posteriormente cada jugador puede revisar en un tablero especial qué juegos usó ese día.

Al finalizar el día, la Empresa lista las tarjetas activas.

Tomando en cuenta el enunciado descripto, realizá:

a) El diagrama de clases que modelice la solución, con sus relaciones, atributos y métodos.
b) La explotación de los constructores de ***Empresa***, ***Juego***, ***Tarjeta*** y ***RegistroPartida***.
c) La explotación del método *venderTarjeta()* de ***Empresa***, que recibe un monto en pesos y devuelve una ***Tarjeta***, ya cargada con el monto indicado. Este método, si no hay tarjetas disponibles, también puede devolver *null*.
d) La explotación del método *tarjetaValida()* de ***Empresa***, que recibe una ***Tarjeta*** y controla que el código de empresa de la tarjeta coincida con el código de la empresa.
e) La explotación del método *agregarCredito()* de ***Empresa***. Este método recibe una ***Tarjeta*** y un monto en pesos. Debe verificar la validez de la ***Tarjeta*** y, si es válida y no está activa, activarla. Una vez que la tarjeta esté activada deberá sumarle el monto enviado como parámetro a su saldo original. Devolverá mostrar un mensaje con el número de tarjeta y los montos original, cargado y actualizado, o en su defecto, el mensaje de error de “Tarjeta Inválida”.
f) La explotación del método *validarPartida()* de ***Empresa***. Este método recibe los datos de un juego y una ***Tarjeta***. Verifica que tarjeta sea válida y posea saldo suficiente: si está todo bien, procesa la transacción y la registra en el historial del día. Devuelve un enumerado que indica el resultado de la transacción.
g) La explotación del método *jugar()* de ***Juego***. Este método recibe una ***Tarjeta***. Verifica con la ***Empresa*** que la ***Tarjeta*** sea válida. Si lo es, inicia la partida; si no, informa del error.
h) Explotación del método *listarPartidasDeTarjeta()* de ***Empresa***, que recibe una ***Tarjeta*** y muestra la lista con todos los juegos utilizados hoy con ella.