

Programación II

Guía de ejercicios

Alejandro Mostovoi

Año: 2024



Índice

Programación orientada a objetos	1
¿Qué es?	1
Ejercicios	2
Ejercicios adicionales	19



Programación orientada a objetos

¿Qué es?

Conocida por su sigla POO, la programación orientada a objetos se trata de un paradigma que permite modelar los problemas y sus soluciones estableciendo un paralelismo con elementos de la realidad, facilitando de este modo la tarea de diseño e implementación de las instrucciones de una pieza de software.

Así como la programación estructurada comienza a hacer uso de funciones para permitir la reutilización del código de un modo mucho más prolijo y entendible por sobre las técnicas usadas en la programación secuencial, la programación orientada a objetos propone modularizar aún más el código mediante el uso de entidades llamadas clases, como así también proveer estrategias que permiten realizar tareas que en otros paradigmas requerían muchas líneas de código e intrincadas implementaciones.

En esencia, la filosofía de la programación orientada a objetos apunta a facilitar el análisis de un problema y el diseño e implementación de la solución, tratando de hacerlo lo más práctico posible, legible y mantenible en el tiempo; todo esto se alcanza haciendo hincapié en la reutilización del código, entendiendo el concepto de abstracción, encapsulamiento, refactorización, herencia, polimorfismo, y otras cuestiones intrínsecas del paradigma.

Ejercicios

1. Crear una clase **Persona** que tenga los atributos públicos **nombre** y **apellido**.
 - Crear una instancia y asignarle valores.
 - Mostrar por pantalla los valores asignados.
2. Crear una clase **Vehiculo** que tenga los atributos públicos **marca**, **modelo** y un atributo privado **patente**.
 - Crear una instancia y asignarle valores; notar que el atributo privado no está disponible para la asignación de valores.
 - Mostrar por pantalla los valores asignados.
3. Crear una clase **Articulo** que tenga los atributos privados **marca** y **modelo**.
 - Crear métodos públicos para la asignación de valores.
 - Crear una instancia y asignarle valores.
 - Notar que no es posible mostrar los valores por pantalla y analizar el motivo por lo que esto ocurre.
4. Crear una clase **Cine** que tenga los atributos privados **película** y **horario**:
 - Crear métodos públicos para la asignación y recuperación de valores.
 - Crear un método público **obtenerCartelera()** que devuelva el nombre de la película y el horario.
 - Crear una instancia y asignarle valores.
 - Mostrar por pantalla los valores.
5. Crear una clase **Cine** que tenga los atributos privados **película** y **horario**.
 - Crear métodos públicos para la asignación y recuperación de valores.
 - Crear una instancia y asignarle valores.
 - Mostrar por pantalla los valores.
 - Crear una segunda instancia y asignarle valores.
 - Mostrar por pantalla los valores.
 - Cambiar los valores de la primera instancia.
 - Mostrar en pantalla los valores de ambas instancias; concluir que la instrucción **new** crea objetos distintos.

6. Crear una clase **Fruta** con variables privadas **color**, **peso**, **esEstacional**.
 - Crear setters y getters.
 - Escribir una función llamada **esComestible()** que devuelva verdadero cuando la fruta pesa menos de 200 gr y es de estación, y falso en cualquier otro caso.
 - Definir dos constructores de modo tal que la fruta pueda crearse con los valores color, peso y estacional al momento de instanciarse, o bien crearse sin valores iniciales.
7. Crear una clase **Ninja** con las variables privadas **arteMarcial**, **arma**, **fuerza** (entero) y **salto** (entero).
 - Crear setters y getters manualmente.
 - Crear una función **saltar()** que reciba un parámetro multiplicador (entero); imprimir por consola salto x parámetro.
 - Crear la función **ataque()** que imprima por consola el arma que usa el ninja y el arte marcial.
 - Crear dos instancias de **Ninja**, asignar distintos valores para cada uno de los atributos e invocar las **funciones saltar()** y **ataque()**.
8. Crear una clase **Persona** que tenga los atributos privados **nombre** y **apellido**, con sus setters y getters.
 - Crear una clase llamada **Visitante** con los mismos atributos.
 - Crear una clase **Guardia con los mismos atributos**.
 - Crear una instancia de cada una de las clases y asignarle valores.
 - Mostrar por pantalla los valores.
9. Crear una clase **Persona** que tenga los atributos privados **nombre** y **apellido**, con sus setters y getters.
 - Crear una clase llamada **Visitante** que extienda de **Persona**.
 - Crear una clase **Guardia** que extienda de **Persona**.
 - Crear una instancia de cada una de las clases y asignarle valores.
 - Mostrar por pantalla los valores; estudiar las ventajas del uso de la **herencia**.
10. Continuando con el ejemplo anterior, realizar las siguientes modificaciones:
 - Agregar en **Persona** el método **presentarse()** que devuelva nombre y apellido de la persona.
 - Crear una instancia de cada una de las clases y asignarle valores.
 - Mostrar por pantalla los valores.
 - Sobreescribir el método **presentarse()** en la clase **Guardia** de modo tal que devuelva el siguiente mensaje "Hola, mi nombre es <nombre y apellido> y soy el guardia." Donde <nombre y apellido> debe ser reemplazado por el nombre y apellido del guardia.

- Mostrar por pantalla el resultado de invocar el método **presentarse()** y advertir que la implementación en la clase **Guardia** tiene precedencia sobre la de su padre.

11. Continuando con el ejemplo anterior, realizar las siguientes modificaciones:

- Agregar en **Visitante** el atributo privado **dni** (numérico) con sus setters y getters correspondientes.
- Agregar en **Guardia** el método público **controlarDocumento()** que reciba como parámetro el dni de la persona y devuelva el mensaje “Adelante persona con dni <dni>” donde <dni> es el valor recibido por parámetro.
- Crear una instancia de cada una de las clases y asignarle valores.
- Mostrar por pantalla los valores.

12. Continuando con el ejemplo anterior, realizar las siguientes modificaciones:

- Modificar la clase **Guardia** para que el método público **controlarDocumento()** devuelva el mensaje “Adelante <nombre completo del visitante> con dni <dni>” reemplazando respectivamente con el nombre completo del visitante y su dni.
- Crear una instancia de cada una de las clases y asignarle valores.
- Mostrar por pantalla los valores.
- Analizar si es posible pasar un único parámetro al método **controlarDocumento()** y estudiar las ventajas y desventajas que tendría asociado.

13. El laboratorio Kokumo Technologies está desarrollando el prototipo de un robot explorador cuyo sistema de tracción puede ser personalizado para que se adapte mejor al terreno.

El robot, llamado KT-2020, tiene las siguientes características:

- Número de serie: KT-2020-P
- Potencia de tracción base (PTB): 10 hp
- Tracción: cualquiera de las dos opciones desarrolladas.

Los sistemas de tracción disponibles son:

- Rueda de caucho: ideal para entornos urbanos, su uso le resta 1 hp al PTB y permite el rodado de hasta 100 km; cuando se gasta, debe reemplazarse.
- Oruga: para todo tipo de terreno, le permite avanzar hasta 400 km antes de requerir reemplazo y resta 3 hp al PTB. Incorpora sensores Meke-M0 que le permiten conocer la temperatura.

Analizar, diseñar, diagramar las relaciones e implementar el código.

Crear instancias de cada una de las clases y asignarle al robot los distintos sistemas de tracción, procurando mostrar por pantalla los siguientes datos entre las distintas asignaciones: Número de serie, potencia de tracción final, tipo de tracción, cuanto puede avanzar y datos sobre cualquier característica adicional que posea.

14. Una empresa de seguridad que se dedica a la vigilancia mediante el empleo de drones, ha desarrollado un sistema de montaje que permitirá que los drones puedan cargar, además de la cámara de vigilancia, una herramienta accionable a distancia.

Actualmente el sistema de anclaje admite:

- Sensor infrarrojo: pesa 250 gramos
- Taser: pesa 300 gramos
- Brazo robótico: pesa 500 gramos

El dron puede soportar hasta 200 gramos sin sufrir penalizaciones de velocidad (5 mts/s) ni altura (100 mts); luego, por cada 50 gramos extras, el dron reduce su velocidad en 2% y la altura en 5%.

Analizar, diseñar, diagramar las relaciones e implementar el código.

Crear instancias de cada una de las clases y asignarle al dron las distintas herramientas, procurando mostrar por pantalla los siguientes datos entre las distintas asignaciones: velocidad, altura y tipo de herramienta que lleva.

15. La Marina del reino de Caballito quiere desarrollar un sistema que le permita gestionar su flota de navíos; por el momento únicamente se requiere presentar ante las autoridades un posible diseño en el que se expongan las relaciones entre las entidades que modelarán los datos.

- De los acorazados se requiere saber la flotabilidad, la solidez, la estabilidad, blindaje y potencia de fuego, además de la velocidad crucero y el nombre con el que fue bautizado.
- Los destructores se caracterizan por la potencia de fuego y altos índices de maniobrabilidad y estabilidad cuando alcanza su velocidad máxima, aunque también se necesita registrar la flotabilidad, solidez, la velocidad crucero y nombre.
- Hay únicamente un barco hospital, llamada “Sibelancia”, con excelente flotabilidad y una estabilidad extrema que la hace ideal para su trabajo; posee una capacidad de carga que le permite brindar servicios a setenta y cinco pacientes.
- “La gaucha” y “El gaucho” son dos lanchas destinadas a brindar servicio médico que se emplean para salvatajes rápidos; poseen motor fuera de borda, una elevada flotabilidad que le permite ir muy rápido, aunque debido a que la estabilidad no es muy buena, la maniobrabilidad se ve afectada; ambas poseen una grúa pequeña que les permite subir y/o arriar objetos de hasta trescientos kilos.

Analizar, diseñar, diagramar las relaciones e implementar el código.

Crear instancias de los distintos barcos, asignar valores y mostrar por pantalla.

16. Una empresa de logística que se encarga de realizar envíos, pone a disposición de sus clientes dos tipos de vehículos:

- Una camioneta con capacidad para llevar cómodas, heladeras y lavarropas.
- Un auto con espacio suficiente como para llevar televisores, bicicletas plegables y cajas pequeñas.

Modelar las entidades teniendo en cuenta las siguientes consideraciones:

- Los vehículos deben ofrecer el método **cargar()** para ir incrementando su carga, razón por la cual la función debe recibir el dato por parámetro.
- Mediante el empleo del método **listarItems()** el vehículo deberá facilitar el listado que compone la carga.

- Todos los elementos poseen una descripción, dimensiones y un número que los identifica pero, además, resulta de interés:
 - i. **Cómodas:** superficie y cantidad de cajones.
 - ii. **Heladeras:** voltaje al que trabaja y si posee freezer.
 - iii. **Lavarropas:** voltaje al que trabaja, carga y revoluciones de centrifugado.
 - iv. **Televisores:** voltaje al que trabaja, si es de led o lcd y si es inteligente.
 - v. **Bicicletas:** tamaño de rodado, si son eléctricas y cantidad de cambios.

Analizar, diseñar, diagramar las relaciones e implementar el código considerando que la capacidad máxima de carga del auto es de 5 elementos, mientras que para la camioneta es de 10.

17. Juancito Jaquer ha fabricado un scanner que es capaz de analizar el objeto, obtener información básica y advertir si ese objeto, a su vez, oficia como contenedor de otro objeto.
La idea de Juancito Jaquer es poder vender el dispositivo a los departamentos aduaneros fronterizos ya que siempre requieren analizar el contenido de los equipajes.

El scanner detecta las características de los objetos y los modela de la siguiente forma:

- Material: metal, cuero, madera, vidrio, plástico, líquido, textil, goma, "otro".
- Volumen: en centímetros cúbicos.
- Contenido: lista de objetos que contiene.

Juancito Jaquer necesita desarrollar la pieza de software que se encarga de estudiar lo que el scanner le provee y mostrar en pantalla las características de los objetos, mencionando si se trata de un objeto:

- Simple: no contiene ni se encuentra contenido.
- Contenedor: contiene al menos un objeto pero no se encuentra contenido.
- Contenido: se encuentra dentro de un objeto pero no contiene nada.
- Sambuchito: está contenido y contiene.

Analizar, diseñar, diagramar las relaciones e implementar el código.

Hacer pruebas con una Mamushka de madera de varios niveles, un portafolios vacío de cuero, un botiquín de plástico que contiene gasa y agua oxigenada, una bolsa de cuero que contenga un peine y un botiquín.

18. Una inmobiliaria barrial de alcance nacional, decide informatizarse e incluir como herramienta para el trabajo diario, un sistema que le permita la gestión de los inmuebles que administra.
Sabendo que la solución será compleja, piensa que es oportuno iniciar por las partes del sistema que le permitan dar de alta los inmuebles.

Se trate de un departamento o de una casa, es indispensable contar con los datos catastrales (provincia, barrio, nombre de la calle, altura y código postal), información sobre los ambientes

(cantidad, tipo y dimensiones), conocer sobre el contacto (nombre, apellido, teléfono y correo electrónico), y poder incluir alguna observación.

Otros aspectos comunes por los que los clientes preguntan giran en torno a si los inmuebles pertenecen a un barrio privado o no, si los ambientes son luminosos, si está conectado al suministro de gas, y si están emplazados en lugares con infraestructura cloacal.

En el caso de los departamentos, también es de interés saber el piso y el número o letra, además de conocer si el edificio admite mascota o no; por las casas, en cambio, la inquietud suele girar en torno a si tienen quincho o pileta.

Las condiciones para poder dar de alta un inmueble en el sistema es que todos los campos estén completos (salvo las observaciones, que son opcionales); en el caso del contacto, además del nombre completo, debe haberse ingresado alguno de los dos medios de contacto.

Los objetos deben ser persistidos en memoria, pudiendo emplear cualquier colección que resulte de conveniencia.

Se solicita pensar la estrategia, realizar el diagrama y codificar la solución, pensando en una estrategia por capas.

Desde el main, crear inmuebles con varias características diferentes, almacenarlas en memoria, y luego recuperarlas para mostrar por pantalla las cualidades.

19. Una empresa que se dedica a la venta de muebles necesita un sistema que le permita controlar los gastos y las ventas de los distintos artículos que gestionan.

Debido a que pretenden informatizarse de a poco, por el momento se conforman con una solución que les permita dar de alta los artículos, asignarle un costo de producción y un precio de venta.

Sin embargo, como venden al por mayor y al por menor, es fundamental que puedan manejar al menos dos listas de precio.

De los artículos simplemente se necesita saber el nombre, el costo, aunque a veces es importante contar con la posibilidad de agregar una observación.

Una lista de precio siempre es identificada por un nombre, la fecha tope de vigencia, un marcador que identifica si es mayorista o minorista, y el detalle de los artículos con su precio.

Se pide analizar el problema, modelar las entidades involucradas, realizar el diagrama, e implementar la solución.

Crear un conjunto acotado de artículos y mostrar los detalles por pantalla; adicionalmente crear una lista de cada tipo y también mostrar por pantalla el resultado.

20. Habiendo implementado con éxito la primera parte del sistema, deciden ir un paso más allá e incluir un conjunto de mejoras.

La empresa necesita que se implemente un mecanismo de validación que controle -al momento de dar de alta un artículo- que el nombre no supere los 15 caracteres y que el costo sea mayor a cero; la observación es opcional pero, si se incluye, que no supere los 30 caracteres.

Cuando un artículo no cumple con las especificaciones, debe mostrarse en pantalla qué concepto falló.

También quieren que la diferencia entre el precio más alto y el más bajo no supere el 30%, de modo tal que el sistema debe aumentar el precio más bajo para respetar ese límite (aplica a todas las listas de precio).

Se pide analizar el problema, modelar las entidades involucradas, realizar el diagrama, e implementar la solución.

Crear un conjunto acotado de artículos, algunos con errores y mostrar el resultado por pantalla; adicionalmente crear una lista de cada tipo y también mostrar por pantalla el resultado con los ajustes.

21. El banco Kokumo Bank inaugura una nueva sucursal en el reino de Caballito.

Como parte de la estrategia para captar clientes, decide lanzar un nuevo plan de préstamos personales.

De las personas se necesita conocer el dni, el sueldo bruto y la antigüedad del empleo actual.

Cuando el sistema recibe dichos datos, inmediatamente después realiza una consulta al servicio web del Ministerio del interior para validar el dni; a los efectos se debe instanciar un objeto `PersonIdentity` e invocar a la función `getInfo()` pasándole el dni.

El servicio devolverá un objeto `Person`, con los siguientes atributos: `name` (`String`), `lastname` (`String`), `birthDate` (`Date`).

Si la persona no llega a los 21 años, debe mostrarse un mensaje que indique que no es posible extender el préstamo por no reunir la edad mínima; caso contrario, el criterio de préstamo es el siguiente:

- Salario entre 30000 y 40000: 80000 con un 30% de interés.
- Salario hasta los 60000: 120000 con un 35% de interés.
- Salario hasta los 80000: 140000 con un 39% de interés.
- No se extiende créditos para salarios mayores a 80000.

Asumir la existencia de la clase `PersonIdentity`; eventualmente, para permitir la compilación del código, implementarla.

22. Un sistema de gestión automatizada necesita ser capaz de analizar documentos digitales de distintos organismos nacionales; su objetivo es aceptarlos o rechazarlos, según cumplan con las condiciones descritas más abajo.

Cada cierto tiempo el sistema en cuestión recupera los datos mediante comunicación directa con cada uno de los subsistemas, modelando los valores en clases que presentan la estructura que los del origen.

Desarrollar una solución que permita normalizar los objetos y analizar los siguientes datos:

- id (numérico): es el número de legajo, folio o expediente, del documento a analizar.
- issueDate (date): es la fecha de emisión, lanzamiento o erogación.
- body (String): se trata del cuerpo, contenido, desarrollo o caso.
- responsable (String): nombre, autor o firmante, del documento.

La solución debe poder contar con una función que reciba por parámetro una lista de Document con los atributos a priori mencionados y mostrar por pantalla "aprobado" en caso de estar aprobado y "rechazado" en caso contrario; el criterio de aprobación es que estén todos los datos completos y que el cuerpo contenga al menos 100 caracteres y la fecha no sea mayor a la de hoy.

Considerar que el sistema tiene las siguientes clases y atributos:

Escrito	Documento	Ley
- legajo	- expediente	- folio
- erogación	- lanzamiento	- emisión
- cuerpo	- contenido	- desarrollo
- autor	- nombre	- firmante

Desarrollar una solución que permita adecuar estos documentos al formato de Document, incluyendo diagramas, etc.

23. La empresa tecnológica Kokumo Tech Inc se lanza al mercado de la seguridad mediante la oferta de un sistema integral de vigilancia capaz de cubrir aire y tierra; la solución apunta a barrios privados, estancias y regiones de decenas de hectáreas.

KTI necesita conciliar los datos de los distintos dispositivos para poder brindar una solución homogénea, de modo tal que se debe pensar en una pieza de software sabiendo que:

- a. GrON3: se trata del dispositivo volador que capta objetos clasificables en grande, pequeños o medianos, y es capaz de reconocer si se trata de un humano o un animal.

Para obtener los datos del drone, debe ejecutarse el método estático `getData()` de la clase `FlyingGr0n3`, pasándole por parámetro el número de id (rango del 1 al 10). Si no existe el id del dispositivo, el método lanza una excepción; en caso contrario devuelve una lista (que puede estar vacía) de `Gr0n3Object`, que presenta los siguientes atributos:

- `size (string)`: admite los valores `big`, `small`, `medium`.
- `type (string)`: admite los valores `human`, `animal`.
- `kind (string)`: si se trata de un animal, los valores posibles son `dog`, `cat`, `cow`, `sheep`, `horse`, `undefined`.

- b. `FalderPerr177`: se trata de la unidad terrestre que sólo distingue entre humanos y no humanos, con la capacidad de identificar el género biológico de los primeros.

Mediante la instancia de la clase `FalderP`, es posible recuperar los datos si se invoca `getData()`; la función devuelve una lista que contiene a todas las unidades activas y que se modelan a través de la clase `Perr177`, cuyos atributos son:

- `id (entero)`: un id numérico
- `List<Perr177Data>`: es una lista que contiene elementos del tipo `Perr177Data`, que presenta los siguientes atributos:
 - `type (string)`: admite los valores `HUMAN`, `NO_HUMAN`, de un enum llamado `FPTType`.
 - `size (string)`: admite los valores `rande`, `pichitito`, `niranipi`.
 - `gender (string)`: admite los valores `MALE`, `FEMALE`, de un enum llamado `FPGender`.

La idea es poder implementar una clase que contenga la función `fetchSurveillanceData()` encargada de recuperar los datos de los distintos dispositivos para, luego, transformarlos en un formato homogéneo de modo tal que sea posible pasárselo a la función `printInfo()`, buscando de esta manera imprimir la siguiente información por cada objeto:

Dispositivo: <id dispositivo>

Tipo: <Humano, No humano>

Tamaño: <Grande, mediano, pequeño>

Observaciones: <los valores de `kind` o de `gender`, dependiendo el dispositivo>

24. Una empresa decide monitorear el ingreso y egreso de su personal, razón por la que ha instalado molinetes en sus distintos puntos de acceso.

Debido a la escasez de insumos, se han comprado dos modelos distintos, obligando a los desarrolladores del sistema de monitoreo a trabajar con dos drivers distintos para obtener los datos de interés:

- GiraMax: a través del método `getInboundAccess()` y `getOutboundAccess()` obtienen un número entero que indica la cantidad de personas que han ingresado y egresado.
- Molinit: similar al anterior, excepto que las funciones son `obtenerEntrada()` y `obtenerSalida()`.

Diseñar una solución que permita obtener los ingresos y egresos de manera homogénea, sin tener que considerar el tipo de clases a la que pertenece el objeto para poder recuperar los valores. Plantear el diagrama de clases e implementar.

25. Un año más tarde, la misma empresa del punto anterior adquiere molinetes de otra empresa que, a diferencia de los anteriores, posee un driver que únicamente expone una función pública:

- Rotativity: la función `getUsage()` devuelve un array numérico con dos elementos, siendo el primero de ellos destinado para los ingresos y, el segundo, para los egresos.

Diseñar una solución que permita obtener los ingresos y egresos, adaptándose a la solución anterior; de no ser posible, plantear una nueva solución para que los tres drivers puedan tratarse homogéneamente. Plantear el diagrama de clases e implementar.

26. Una empresa de correo está expandiendo sus dominios y ha decidido abrir una sucursal en el reino de Caballito.

La empresa en cuestión posee una **sede central** y un conjunto de dependencias que, a su vez, nuclean hasta tres sucursales cada una.

Para mantener el balance de carga laboral, siempre que se abre una sucursal se evalúa si las dependencias existentes ya están nucleando a tres; en caso, de ser así, se elige arbitrariamente a una y se le asigna la nueva sucursal; sin embargo, si existe alguna que nuclea a menos de tres, ésta recibirá la nueva sucursal bajo su control.

Si resulta que ya existe una dependencia con cuatro sucursales, entonces ésta cuarta debe transformarse a dependencia y la nueva sucursal quedar bajo su supervisión.

Plantear el diseño, diagrama e implementar la estructura que permita armar la estructura sabiendo que ambas tienen un id y nombre que las identifica, y que la única diferencia entre dependencia y sucursal, es que la sucursal no nuclea.

27. Crear un singleton con un atributo del tipo entero, otro cadena de caracteres y un tercero que sea una colección de objetos (cualquiera).

Desde el main, asignarle valores a los atributos y, una vez cargados, recuperarlos y mostrar en pantalla los valores.

28. Crear un singleton con un atributo del tipo entero, con sus correspondientes setters y getters; luego crear la clase SingletonCaller con el método calling(), sin parámetros, que haga uso del system output para mostrar en pantalla el atributo del singleton.

Desde el main, invocar al singleton, asignarle valor al atributo, y luego crear una instancia de SingletonCaller para ejecutar la función calling.

Concluir que el singleton siempre opera sobre la misma instancia.

29. Una fábrica automotriz es famosa por sus dos modelos de automóvil, "El deportivo" y "el familiar", el primero fabricado con piezas importadas y el segundo, con piezas nacionales.

Ambos cuentan con aire acondicionado y reproductor mp3.

El familiar cuenta con portaequipaje y un baúl espacioso, cosa que no ocurre con el otro automóvil.

- Aire acondicionado: el nacional admite frío y caliente, mientras que el chino únicamente frío.
- Mp3: el nacional tiene entrada miniplug, mientras que el alemán -además- conecta por bluetooth.

Modelar el problema e implementar una solución que permita generar cómodamente, 10 vehículos de cada tipo.

30. Un maestro repostero de ultratumba es conocido por sus tres especialidades:

- La torta de la muerte: 10 huevos de lagarto, 300 gr de harina, 3 gotas de amonio.
- El budín diabólico: 100 cm² de bilis, 200 gr harina, 10 cm³ de lavandina.
- El pionono gangrenoso: recorte de goma eva, 1 ojo daltónico, 200 gr de harina.

Modelar el problema e implementar una solución que permita crear cada uno de los manjares asesinos; mostrar en pantalla los ingredientes.

31. El equipo de marketing de una empresa dedicada al retail, quiere implementar el uso de promociones con el fin de incentivar al público objetivo.

Una lista de precios se caracteriza por un nombre, fecha de la puesta en vigencia, y la nómina de artículos con sus respectivos precios.

Un artículo presenta nombre, código de barra y costo, mientras que las promociones -que se consideran artículos también-, no posee código de barras por estar compuesta por al menos dos artículos, pero si tiene un nombre y la cantidad de cada uno de los artículos que componen la promoción.

Adicionalmente, y por motivos que no vienen al caso mencionar, los artículos que forman parte de una promoción podrían presentar distinto costo con respecto al del artículo de venta directa, para lo cual se le debe aplicar un porcentaje que es informado al momento de confeccionar la promoción.

Modelar el problema, diseñar y diagramarlo, e implementar el código que permita confeccionar una lista de precios con artículos sueltos y promociones.

Mostrar por pantalla la lista de precio.

32. Construir un árbol genealógico familiar; incluir nombre, apellido y año aproximado de nacimiento.

Utilizar como id los últimos dos dígitos del año de nacimiento y las iniciales del nombre y apellido.


33. Digital Horse, empresa de desarrollo de juegos, quiere incursionar en los juegos masivos multiplayer de rol, razón por la que están preparando un prototipo, decidiendo iniciar por la construcción de los personajes.

Los personajes, que pueden ser de la clase guerrero o mago, se caracterizan por poseer nivel, puntos de experiencia, un árbol de habilidades, un arma, una armadura, inventario, unidades de desplazamiento, puntos de vitalidad y un nombre.

Las armas poseen nombre, cantidad de daño que efectúan, y tipo de ataque (CORTANTE, PERFORANTE, APLASTANTE, MÁGICO).

Las armaduras se caracterizan por el nombre, los puntos de defensa, la restricción que le imponen a los puntos de desplazamiento, y el tipo de ataque del que protege (CORTANTE, PERFORANTE, APLASTANTE, MÁGICO).

Las habilidades poseen nombre, nivel requerido, puntos de efecto que se agregan a los existentes y tipo (DEFENSA, ATAQUE).



Todos los personajes deben iniciar con 0 puntos de experiencia, 20 de energía y tienen la capacidad de atacar, defenderse y avanzar.

Cuando se ataca, el daño potencial que se produce es igual a los puntos del arma sumado todas las habilidades de ATAQUE, aunque el atacado puede reducir el daño según la protección de la armadura y las habilidades de DEFENSA que posea.

Más precisamente, si el ataque se produce por un arma que no es del tipo para la que fue creada la armadura, el daño causado es del 100%; si los tipos coinciden, entran en juego los puntos de defensa y habilidades.

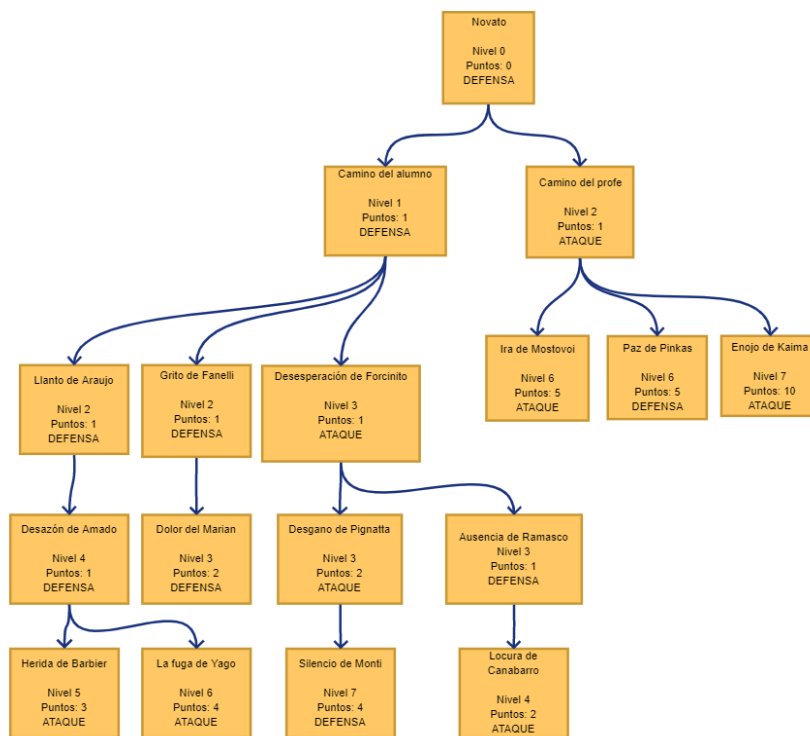
Las armaduras mágicas son capaces de repeler 100% los ataques mágicos, y son capaces de defender cualquier tipo de ataque, es decir, aplican los puntos de defensa de la armadura y habilidades defensivas.

Los personajes pueden avanzar sin detenerse, tantos metros como unidades de desplazamiento posean, menos los puntos de penalización indicados por la armadura.

Cuando un personaje muere, pierde su arma y armadura, la que pasa automáticamente al inventario del ganador; adicionalmente, el ganador adquiere tantos puntos de experiencia como energía haya quitado.

Los personajes adquieren nuevos niveles cada vez que los puntos de experiencia se igualan a su próximo nivel al cuadrado.

Cada vez que se sube de nivel, se desbloquean todas las habilidades cuyo nivel requerido coincida con el del personaje; adicionalmente, la cantidad de puntos de energía total resulta en la multiplicación de 20 por el nuevo nivel adquirido.



34. Considerar un banco que posea una jerarquía constituida por una jerarquía de tres niveles: empleado, supervisor y gerente.

Mientras el primero puede aprobar préstamos hasta \$AR 10000, el supervisor hasta \$40000 y el gerente hasta \$AR 120000; cualquier monto que exceda el máximo, no puede ser aprobado.

Plantear una estrategia que permita ilustrar los distintos casos de aprobación.

Como siempre, diseñar, diagramar e implementar el código.

35. Un mozo con poca memoria siempre llega a la mesa con los platos que los comensales solicitaron y, en vez de entregarlos en mano y a quien corresponda, se los va pasando al primero de la mesa para que, oportunamente, cada quien tome el que le corresponde.

Diseñar, plantear el diagrama de clases e implementar la solución que satisfaga la necesidad; ejemplificar considerando cuatro platos distintos y cuatro clientes a los que les corresponde uno de ellos.

36. El Reino de Caballito tiene intenciones de habilitar, el año próximo, una plataforma web para que todos sus residentes puedan buscar empleo.

Con el fin de acelerar la puesta en marcha, se pretende implementar una landing page que le permita a los postulantes suscribirse para recibir ofertas.

Los visitantes deberán informar la dirección de correo electrónico y las categorías de los empleos que les resulten de interés para que, luego, el sistema les envíe notificaciones cada vez que una empresa ingrese algún puesto laboral que se ajuste a dichas categorías.

Adicionalmente, la landing page debe permitir desuscribirse utilizando el correo del postulante como identificador.

Implementar la solución que permita la suscripción, desuscripción y notificación, sin olvidar plantear el diagrama de clases.

Categorías:

- Informática
- Administración
- Gastronomía

Mensajes:

- Gracias por suscribirte a las categorías <categorías>.
- Sentimos que te desuscribas :(
- La empresa <Nombre de la empresa> busca cubrir puestos de <Rubros>:
<mensaje de búsqueda>

37. Modelar una mascota virtual, semejante a un Tamagotchi, que incluya las funciones correspondientes a las acciones comer y jugar, y funciones que permitan saber si puede jugar o no, y una que permita devolver un valor numérico comprendido entre 0 o 10 cuyo significado es el nivel de felicidad.

Una mascota, que puede encontrarse aburrida, hambrienta o contenta, se comporta en base a su estado de ánimo.

Cuando come, esto es lo que ocurre:

- Si está hambrienta, se pone contenta.
- Si está contenta, su nivel se incrementa en una unidad.
- Si está aburrida, y hace más de 80 minutos que está aburrida, entonces se pone contenta.
- Si está aburrida desde hace 80 minutos o menos, entonces no le pasa nada, no cambia nada.

Cuando juega, esto es lo que ocurre:

- Si está contenta, su nivel se incrementa en dos unidades.
- Si está aburrida, se pone contenta.
- No produce ningún efecto jugar con la mascota si está hambrienta.

Una mascota puede jugar si está contenta o aburrida, si está hambrienta no.

Si no juega cada 15 segundos, se aburre; si estaba aburrida, pierde un punto de felicidad.
Si en 20 segundos no come, pasa a estar con hambre.

NO SE PUEDE CONSULTAR DE NINGUNA MANERA EL ESTADO ACTUAL DE LA MASCOTA.

Esto quiere decir que está prohibido hacer comparaciones del tipo estado = 'contento' o cualquiera similar utilizando mensajes especiales.

38. Una empresa de logística registra durante el día, peticiones de envío de mercadería; éstas se caracterizan por un id de pedido, código de ruta y un código alfanumérico que se emplea para identificar el artículo a enviar.

Por otro lado, a la tarde, los repartidores nocturnos se loguean a la plataforma de la empresa mediante su usuario y un password; el sistema emplea una biblioteca que permite usar la clase Security que expone el método loginUser() que admite user name y password, devolviendo true si válido y false en caso contrario; excepciones podrían ocurrir aunque no se especifican cuáles.

Cerca de las 23 hs, el sistema ejecuta una tarea que asigna a todos los repartidores logueados, hasta 5 envíos; cada repartidor está asociado a un único código de ruta y para saber cuál le corresponde, se debe invocar el método getRoute() de la clase Logistic, perteneciente a una biblioteca, que espera como parámetro el user name del repartidor.

La asignación del envío consiste en invocar el método setDelivery() de la clase Logistic, pasándole como parámetros el user name y la petición de envío; todo envío delegado debe marcarse como asignado a un responsable.

Una vez ejecutada la tarea de asignación, se emite un informe con el listado de los artículos pendientes.

39. Un satélite que orbita al planeta Tierra, toma imágenes de la superficie; cada vez que puede envía las fotos a la estación terrena que está sobrevolando para descargar todos los datos y poder seguir operando; el satélite puede capturar hasta 20 imágenes en FMZR (Full Mega Zarpad Resolution) ya que luego la memoria se llena.

Las estaciones reciben las imágenes como si fueran una cadena de texto (no menor a 24 caracteres y con un largo múltiplo de 8) y las transforman del siguiente modo:

- Los primeros 8 elementos son el código de identificación
- Los siguientes 8 elementos refieren a la zona en la que fue tomada (codificada)
- El resto es la imagen per sé.

Una vez modeladas, tienen que ser procesadas por 3 algoritmos; el primero convierte a mayúsculas todas las letras y suma 1 a cada número (si el número resultante es 10, se considera 0), el segundo de los procesos trabaja con los elementos entre la posición 9 y 16 de la imagen invirtiendo el orden, y el tercero de los procesos elimina los tres primeros caracteres de la cadena si el primer número encontrado (recorriendo desde el inicio) es par.

No puede procederse al procesamiento si el paso anterior no se ejecutó exitosamente.

Más procesos podrían incluirse en un futuro.

Ejercicios adicionales

- a. Crear un enumerado con los literales: ALTO, BAJO
 - Mostrar por pantalla los valores y el orden de los literales.
- b. Crear otro enumerado, cuyos valores sean los nombres de los días de la semana y asignarle valores numéricos.
 - Mostrar por pantalla los valores literales, los valores numéricos, y el orden de los literales.
- c. Crear una clase **Menu** cuyo constructor requiera que se le suministre una parámetro del tipo **Comida**, un enumerado con los valores POLLO, ASADO, PESCADO y PASTA.
 - Almacenar el valor pasado por parámetro en el atributo **comida** y crear el getter que lo devuelva.
 - Desde el main, crear una instancia de Menu y mostrar por pantalla el valor del atributo.
 - Concluir que los enumerados son prácticos para cuando se requieren tipar datos.
- d. Crear una clase **Persona** que presente los atributos **nombre** y **apellido**.
 - Crear setters y getters.
 - Crear un **ArrayList** de **Persona**.
 - Crear varias instancias de **Persona**, asignarle valores a los atributos, y agregarlas a la lista.
 - Imprimir los nombres y apellidos de todas las personas, recorriendo la lista con un **foreach** y con un **for** convencional.
- e. Crear una clase **Cabecera** con los atributos **número** (entero), **fecha** (fecha) y detalle (lista de **Articulo**); también crear la clase **Articulo** con los atributos **nombre** (String) y **precio** (float).
 - Implementar setters y getters en ambas clases.
 - Crear en el main una instancia de **Cabecera** y asignarle valores.
 - Crear en el main dos o tres instancias de **Articulo** y asignarle valores.
 - Agregar a **Cabecera** los artículos creados.
 - Recorrer la lista detalle del objeto **Cabecera** y mostrar en pantalla los valores de sus atributos.

- f. Implementar la clase "ListaDeArreglo" que funcione -conceptualmente- como un ArrayList.

Debe ofrecer los siguientes métodos:

- add: debe permitir agregar un elemento.
- addAll: debe permitir agregar el contenido de otra ListaDeArreglo.
- get: debe recibir un número entero que representa la posición del objeto que debe recuperar.
- clear: debe vaciar la lista.

Para evitar problemas, ListaDeArreglo debe admitir objetos del tipo Object.

Recordar que ListaDeArreglo debe autodimensionarse; la cantidad de elementos iniciales es 4 y luego debe ir incrementándose siguiendo la siguiente fórmula: $\text{capacidad_actual} \times 2$ (ej: 4, 8, 16, 32, 64, etc).

- g. Una abuelita decide involucrarse de lleno en el ámbito del desarrollo, razón por la que se propone como objetivo diagramar un conjunto de clases que le permitan modelar una receta.

A la abuelita 2.0 le interesa poder asignarle un nombre a la receta e incluir los ingredientes.

Ayudar a la abuelita proponiendo una posible solución.

- h. La abuelita quiere ir más allá de los límites autoimpuestos, así que ahora necesita incluir las cantidades de cada ingrediente.

Proponer los cambios necesarios como para poder reutilizar el código anterior, favoreciendo la mantenibilidad y legibilidad del código.

- i. Un empresario contrata a un "encendedor y apagador" de objetos debido a que tiene varios elementos cuya capacidad de puesta en marcha y detención debe testear:
- Lavarropas: enciende y apaga.
 - Tostadora: tuesta y no tuesta.
 - Moto: brm y no brm.

Encontrar el modo común de poder encender y apagar los elementos.

- j. Una compañía aseguradora abre su primer estación de control para que puedan acercarse los dueños con las motos o autos que quieran asegurar (y en un futuro otros vehículos motorizados).

El costo de la póliza depende de la política evaluatoria vigente; por el momento y debido a la inmadurez de la compañía, los criterios de evaluación cambian con cierta regularidad e, inclusive, durante la jornada de atención al cliente.

Por ejemplo, una de las políticas indica que el beneficiario debe pagar mensualmente lo que resulte de la división entre el kilometraje y el año de fabricación del vehículo.

Otra, por ejemplo, considera la cilindrada del vehículo y la divide por dos a fin de obtener el arancel a pagar.

Analizar, diseñar, diagramar las relaciones entre las clases e implementar el código para obtener por pantalla el arancel a pagar, cambiando la política para un mismo vehículo.

- k. Little Horse Robotics, reconocida fábrica de robots a nivel intergaláctico, utiliza una estrategia basada en estaciones para el ensamblado de sus productos (lo que sale de una estación, entra en la siguiente).

Debido a la pluralidad de artículos que confecciona, la cantidad de estaciones requeridas puede variar y es necesario saber qué modelo de robots se manufacturará para prepararlas, es decir, ordenarlas en secuencia y asignarles los materiales que necesitarán (tipo y cantidad).

Si en medio proceso de manufactura alguna estación se queda sin insumos, la construcción debe abortarse y dar aviso.

Todos los robots se construyen sobre el mismo esqueleto base, el que tiene slots para incorporar:

- GPU
- Brazos y piernas
- Torso
- Box-e-tracio (para sensores y otros elementos)

La cadena productiva más común está compuesta por la estación de ensamblado de GPU, la de extremidades y la de torso; sin embargo, algunos modelos podrían requerir elementos adicionales (como sensores), los que se incorporan en sus respectivas estaciones de nombre homónimo y siempre antes de instalar el torso.

Los R54 requieren 1 GPU M1, 2 brazos S1, 2 piernas S1 y 1 torso S1.

Los Z17 requieren 1 GPU M1, 2 brazos S1, 2 piernas S1, 1 pierna S2, 1 torso F1 y 1 sensor K23.

Los K11m3 requieren 1 GPU A1, 2 piernas S2 y 3 microantenas A7.



Implementar una solución que modele la dinámica de trabajo; crear diagrama e implementar código.

Probar construyendo varios sets de robots.