TRABAJO PRÁCTICO Nº 2

Resolvé los ejercicios utilizando los diagramas que correspondan. Asegurate de leer al menos dos veces los enunciados antes de intentar confeccionar las soluciones.

ENUNCIADOS

1) Arrancando despacitoooooo…….

Modelá las siguientes clases en UML, incluyendo los atributos y métodos listados. Agregá los atributos y métodos que consideres pertinentes según tu abstracción.

Clase	Atributos	Métodos

Fecha	`dia`
	`mes`
	`anio`
		`mostrarComoCadena`
		`mostrarDiasTranscurridos`

TelefonoMovil	`marca` `estaEncendido`	`prender` `llamar`
MaquinaDeCafe	`nivelAgua` `cantVasos`	`prepararInfusion` `apagar`

Por último, modelá una nueva clase, de tu preferencia, con sus atributos y sus métodos.

2) Compumundo

Realizá un diagrama de clases UML y sus respectivas relaciones de asociación (junto a la cardinalidad) a partir de la abstracción del enunciado:

> Una persona (de la que se conoce nombre, apellido y DNI) es propietaria de una computadora. La persona es capaz de trabajar y descansar, mientras que la computadora es capaz de prender, apagar y reiniciar. Una computadora consta de marca, tipo (`"Desktop"`, `"Laptop"` o `"All in One"`) y de un procesador. Algunas también traen una lectora de DVD. El procesador posee su propia marca, modelo y nivel actual de temperatura, además de poder notificar cuando llega a un nivel de temperatura crítica. De la lectora de DVD se conoce su marca y si es capaz de grabar o no.

3) Fulano y su Giat

Fulano Gómez compró un vehículo, un Giat Halio de color gris oscuro que tiene una velocidad máxima, de 180 km/h. Así como Fulano posee el Giat Halio, hay otros propietarios que poseen un vehículo al igual que él, los que pueden ser de otras marcas, modelos, colores, etc. Aún Fulano no lo puede probar ya que no sabe conducir. Realizá el diagrama que corresponda.

4) Fulano tiene registro!

En el ejercicio en dónde Fulano Gómez tiene un Giat Halio, finalmente aprendió a conducir y quiere salir a probar el auto que se compró. Para ello tiene que verificar que el vehículo esté encendido (si no lo está encenderlo), luego acelerar para ello debe indicarle al auto que velocidad incrementar en Km/h, frenar en ese caso le debe indicar la cantidad de Km/h debe hacerlo y poder girar para lo cual indicará hacia qué dirección quiere hacerlo y la cantidad de grados la cual no puede ser superior a los 90 grados. Finalmente podrá apagar el vehículo.
Modelá el diagrama correspondiente y explotá los métodos mencionados.

5) Qué complejo es el automóvil de Fulano…!

Crear la clase **Automovil** con los siguientes atributos: marca (String), modelo (String), patente (String), capacidadTanque (double), cantidadDeCombustible (double) y rendimientoPorLitro (double, indica cuantos kilometros recorre con un litro de combustible).

Implementar los siguientes métodos y constructores:

a. Constructor parametrizado: recibe marca, modelo, patente y la capacidad del tanque de combustible.
b. Método privado *calcularConsumo():* recibe la cantidad de kilómetros que se quiere recorrer y devuelve un double indicando los litros de combustible necesarios según los kilómetros requeridos.
c. Método privado *calcularDistancia():* recibe la cantidad de litros quiere consumir y devuelve un double indicando los kilómetros que puede recorrer.
d. Método privado *consumirCombustible()*: recibe la cantidad de litros que se quiere consumir y actualiza la cantidad combustible en el tanque. Devuelve un double indicando los litros faltantes si es que no le alcanza, consumirá lo que tiene.
e. Método público *viajar()*: recibe la cantidad de kilómetros que desea recorrer y devuelve la cantidad de kilómetros que efectivamente se realizaron con el combustible existente en el tanque actualizando la cantidad de combustible consumido. Ayudin….. Para realizar este método deberá valerse de los métodos anteriores.
f. Método público *cargarCombustible()*: recibe por parámetro la cantidad que se quiere cargar, que nunca debe ser menor o igual que cero o mayor al espacio disponible. Si puede, actualiza el atributo correspondiente. Devuelve true o false dependiendo del éxito de la acción.
g. Método privado *espacioDisponible()*, que devuelve la diferencia entre el tamaño del tanque y los litros de combustible almacenados.
h. Método público *pocoCombustible()*, que devuelve un valor booleano, indicando verdadero siempre que la cantidad de combustible sea menor al 15% de la capacidad del tanque.
i. El método toString() que devolverá todos los valores de estado del objeto incluyendo *espacioDisponible()* y *pocoCombustible()*.

En la clase test, crear el objeto a través del constructor. "Ford","Fiesta", "ABCD123", capacidad tanque:40. Setear el rendimiento por litro en 10 y llenar el tanque con 20 (veinte) litros de combustible.

Probar todos los métodos recorriendo 180 kilómetros primero e intentando recorrer 50 después, mostrando siempre la cantidad de kilómetros que devuelve *viajar()*.

6) Corralito corralón

Una cuenta bancaria consta de los siguientes atributos:

- La clave bancaria uniforme (CBU).
- El tipo (caja de ahorro o cuenta corriente).
- El saldo (inicialmente en 0).
- La fecha de apertura (de tipo `Fecha`).
- El titular de la cuenta (de tipo `Persona`).

La persona titular de la cuenta tiene DNI, nombre, apellido, fecha de nacimiento y domicilio (calle, altura y barrio).

Cada vez que se crea una nueva cuenta bancaria, debe establecerse la fecha de apertura automáticamente y generarse un nuevo CBU. Para ello, agregá los siguientes métodos en la clase **CuentaBancaria**, ambos sin parámetros (no es necesario que los implementes):

- `obtenerFechaDeHoy`: devuelve un objeto de tipo `Fecha` con la fecha de hoy.
- `generarCBU`: devuelve una nueva clave bancaria uniforme.

Las operaciones que debe poder hacer toda cuenta bancaria son:

- `verSaldo(): double`

Devuelve un valor double con el saldo actual de la cuenta.

- `depositar(double): void`

Actualiza el saldo de la cuenta, sumando el monto enviado por parámetro.

- `extraer(double): boolean`

Actualiza el saldo de la cuenta, restando el monto enviado por parámetro, siempre y cuando el saldo alcance, de lo contrario, no se realiza la extracción

A) Modelá la solución en UML
B) Realizá la implementación de cada método utilizando diagramas Nassi Schneiderman.
C) El constructor de `CuentaBancaria`.
D) El constructor de `Persona`.
E) El constructor de `Domicilio`.
F) El constructor de `Fecha`.

El método `main` de la clase ***Principal*** que cumpla con lo siguiente:

Una cuenta bancaria de tipo caja de ahorros le pertenece a Fulano Gómez, cuyo DNI es 12345678 y otra de tipo cuenta corriente le pertenece a Mengana Torres, con DNI 90123456. Ambos son pareja y viven juntos en Yatay 240, Almagro.

7) Que robot educado!!

Se precisa un robot que permita atender llamadas telefónicas. La compañía puede detectar algunos clientes según su número de teléfono, sin embargo, en otros casos no. Por ello, el robot debe ser capaz de procesar alguno de los siguientes métodos homónimos:

- `saludar(): void`

Emite por consola un saludo diciendo: `"Hola, mi nombre es ___. ¿En qué puedo ayudarte?"`. En donde el `"___"` debe reemplazarse por el nombre del robot.

- `saludar(Persona): void`

Emite por consola un saludo diciendo: `"Hola ___, mi nombre es ___. ¿En qué puedo ayudarte?"`. En donde en el primer `"___"` debe reemplazarse por el nombre completo de la persona y el segundo `"___"` debe reemplazarse por el nombre del robot.

Modelar la clase **Robot** en UML y realizar la implementación de cada método utilizando diagramas Nassi Schneiderman. Luego, invocar varias veces el método `saludar` a través del método `main` de la clase `Principal` con diferentes variantes para ver si saluda como corresponde. No olvides instanciar un objeto `Robot` en el `main`.

8) Dando turnos

Desarrollar la clase **Turnera**, la cual sabe dar números.
La **Turnera** posee un valor entero que representa el último número otorgado.
Esta clase debe implementar los siguientes métodos:

- otorgarProximoNumero
- verUltimoNumeroOtorgado
- resetearContador

Este último método si no recibe ningún valor reseteará el contador en cero, caso contrario resetea el atributo en el valor recibido siempre y cuando sea un número mayor o igual a cero.

9) Calculín

Desarrollar la clase **Calculadora**, la cual sabe operar valores enteros explotando todos sus métodos (en la división, si el divisor es 0, devolver 0).
La **Calculadora** tiene la siguiente definición:

Calculadora [en clase]	**Métodos**

	`+ sumar(nroA: entero, nroB: entero): entero`
	`+ restar(nroA: entero, nroB: entero): entero`
	`+ multiplicar(nroA: entero, nroB: entero): entero`
	`+ dividir(nroA: entero, nroB: entero): entero`

10) Fracciones, y más fracciones

Desarrollar la clase **CalculadoraDeFraccion**, la cual sabe operar fracciones
Cada **Fraccion** posee numerador y denominador como atributos.

La clase **Fraccion** debe responder a los siguientes métodos:

- valorReal(): double

Devuelve un valor double con el valor real de la fracción.

La clase **CalculadoraDeFraccion** debe responder a los siguientes métodos:

- sumar(Fraccion, int): Fraccion

Devuelve una nueva fracción con el resultado de la suma con el entero recibido como parámetro.

- sumar(Fraccion, Fraccion): Fraccion

Devuelve una nueva fracción con el resultado de la suma con la fracción recibida como parámetro.

11) La Liga de los programadores

Crear la clase **Superheroe** con los atributos *nombre* (String), *fuerza* (int), *resistencia* (int) y *superpoderes* (int). Todos los atributos numéricos deberán aceptar valores entre 0 y 100; en caso de que el *setter* correspondiente reciba un valor fuera de rango deberá setear el valor límite correspondiente (si recibe un valor negativo asignar cero y si recibe un valor superior a cien asignar cien).

El constructor de la clase recibirá todos los valores de sus atributos por parámetro y usará los setters (todos privados) para validar el rango correcto de los poderes del superhéroe.

También habrá el método *toString()* para devolver el estado completo de la instancia y un método *competir()*, ambos públicos. Este último recibirá otro superhéroe como parámetro y, comparando los poderes de él mismo contra el otro recibido por parámetro, devolverá "TRIUNFO", "EMPATE" o "DERROTA", dependiendo del resultado. Para triunfar un superhéroe debe superar al otro en al menos 2 de los 3 ítems.

- Escribir la clase Test que contenga el método main() para probar el correcto funcionamiento de la clase previamente realizada con el siguiente ejemplo:

superHeroe1: Nombre: "Batman", Fuerza: 90, Resistencia: 70, Superpoderes: 0

superHeroe2: Nombre: "Superman", Fuerza: 95, Resistencia: 60, Superpoderes: 70.

- Hacer jugar al superheroe1 pasándole el objeto superheroe2 y mostrar el resultado por pantalla. Chequear el resultado (debería ser "DERROTA").
- Hacer jugar al superheroe2 contra el superheroe1 y mostrar el resultado por pantalla. Chequear el resultado (debería ser "TRIUNFO").

Crear más superhéroes con distintos valores y jugar.

12) La impresora que impresiona….. Que impresionante!!

Desarrollar la clase **ImpresoraMonocromatica**. Sus atributos serán si está o no encendida, la cantidad de hojas actualmente en la bandeja y el nivel de tinta negra. Inicialmente, toda impresora está apagada, sin hojas y con nivel de tinta `100`. Debe responder a los siguientes métodos:

- `void imprimir(Documento) {…}`

Que recibe un Documento compuesto por una Fecha, un título y un cuerpo.
Imprime en consola la fecha del documento, junto a su título y cuerpo, en el siguiente formato:

```
Fecha

**Titulo**

Cuerpo
```

Al hacerlo, se descuenta `1` punto de nivel de tinta por cada `50` caracteres del cuerpo impresos y se resta `1` hoja de la cantidad en bandeja por cada `20` caracteres del cuerpo impreso. Se debe verificar antes de imprimir que se cuente con nivel de tinta suficiente y al menos una hoja en la bandeja.

- `int nivelSegunCantCaracteres(int) {…}`

Devuelve cuánta cantidad de tinta debería usarse según la cantidad de caracteres recibida por parámetro.

- `void recargarBandeja(int) {…}`

Suma a la bandeja una cantidad de hojas. El máximo de la bandeja es de `35` hojas. Se debe verificar no excederse de tal valor. Si el parámetro es negativo, la bandeja se deja como está.

13) Compumundo Reload

Retomando el ejercicio de Compumundo ahora nos piden explotar algunos métodos

`grabar(…)` de la clase **Persona** el cual recibe por parámetro un DVD y devuelve uno de los siguientes valores:

- No hay unidad de DVD
- Hay unidad pero no es capaz de grabar
- Grabación OK

Del DVD conocemos su marca y capacidad.

Desarrollar el método `mostrarComputadora(…)` de la clase **Persona** que debe mostrar todos los componentes de la computadora incluyendo sus accesorios.

14) Contador, Acumulador, Promedio

Un **Contador** sabe contar números de uno en uno, empezando en cero (valor inicial). Sólo tiene un método para incrementar su valor (en 1, sin parámetros) y otro para obtenerlo (`obtenerValor()`, devuelve un entero).
Un **Acumulador** es muy parecido, pero para incrementar su valor necesita que le pasen la cantidad a incrementar (un número real). Su método `obtenerValor()` devuelve un número real que puede ser negativo.
Un **Promedio** tiene internamente un contador y un acumulador, los que le servirán para obtener el resultado del promedio. Para cumplir con su funcionalidad tiene un método llamado `incrementar()`, que recibe un número real. Al igual que Contador y Acumulador, tiene un método `obtenerValor()` que en este caso devolverá el valor del promedio (real). Para no tener problemas con la división por cero, que produce un error de ejecución, se decidió que devuelva 0 (cero) cuando se pida el valor promedio antes de haber procesado algún dato.
Escribir un programa que cargue las notas del primer parcial de los integrantes del grupo (sólo las notas, sin nombre) y muestre la nota promedio.