GUÍA PRÁCTICA DE INTRODUCCIÓN A POO

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

Modelado de clases en Java

1) Modelá la clase `Persona`, la cual posea como atributos nombre, apellido y año de nacimiento. Luego, implementá métodos que permitan:

- Devolver el nombre completo de la persona.
- Mostrar su estado en la consola.
- Obtener su edad actual.
- Saber si es mayor que una edad dada.
- Cambiar su nombre.
- Cambiar su apellido.
- Cambiar su nombre y su apellido.

Finalmente, instanciá una persona en el método `main` y probá todos sus métodos.

2) Desarrollar la clase `Empleado`, cuyos atributos sean su DNI, nombre, apellido, salario base, estado civil (soltero, casado, divorciado o viudo) y cantidad de hijos.

Se sabe que todos los empleados cobran el salario base más un plus del 3% de éste por cada hijo, con un tope de hasta 10%. Del salario resultante debe descontarse un 4% en caso de que el estado civil sea soltero.

Desarrollar el método `obtenerSalarioFinal()` que retorne el valor del salario del empleado según las reglas enunciadas.

3) Modelá la clase `CuentaBancaria`, la cual posea como atributos la clave bancaria uniforme (CBU), el tipo (caja de ahorro o cuenta corriente) y el saldo (inicialmente en `0`). Luego, implementá métodos que permitan:

- Obtener el saldo actual.
- Depositar dinero en la cuenta (actualizando el saldo).
- Extraer dinero de la cuenta (actualizando el saldo). Solo puede quedar en saldo negativo si es cuenta corriente.
- Obtener los últimos 3 dígitos del CBU.

4) Modelá la clase `Fecha`, la cual posea como atributos el día, el mes y el año. Luego, implementá métodos que permitan:

- Obtener la fecha como cadena, en formato `"dd/mm/aaaa"`.
- Saber si es Navidad.
- Sumar un mes.

Finalmente, instanciá una fecha en el método `main` y probá todos sus métodos.

5) Modelá la clase `Fraccion`, la cual posea como atributos numerador y denominador. Luego, implementá métodos que permitan:

- Mostrarse en consola, con el formato `"numerador/denominador"`.
- Obtener el valor decimal.
- Sumarle un entero.
- Sumarle una fracción.
- Simplificar la fracción.
- Saber si la fracción es propia, impropia u aparente.[1]

Finalmente, instanciá una fracción en el método `main` y probá todos sus métodos.

6) Refactoreá las clases de los cuatros ejercicios anteriores con los siguientes cambios:

- Agregar un constructor que inicialice todos sus atributos por parámetro.
- Establecer sus atributos como privados y colocar los getters/setters que correspondan.
- Implementar el método `toString()` y comprobar su funcionamiento.

7) Se precisa un robot (que tiene nombre) 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?"`.[2]

- `saludar(Persona): void`

  Emite por consola un saludo diciendo: `"Hola ___, mi nombre es ___. ¿En qué puedo ayudarte?"`.[3]

Modelá la clase `Robot` en Java. Luego, invocá varias veces el método `saludar` a través del método `main` de la clase `Principal` con diferentes variantes para ver si saluda como corresponde.[4]

8) Desarrollá la clase `Password`, que permita tener contraseñas como objetos. Su único atributo será el valor de la password (`String`). Debe responder a los siguientes métodos:

- `boolean esFuerte() {…}`

  Devuelve si la password es fuerte o no. Una password es fuerte cuando tenga al menos `8` caracteres.

- `boolean nuevoValor(String) {…}`

  Establece como nuevo valor de password el recibido como parámetro, siempre y cuando su longitud sea mayor o igual a `6`, si no, lo deja como estaba. Devuelve si se pudo o no establecer el valor.

---

[1] **Fracciones propias**: son aquellas en las que el numerador es menor que el denominador. **Fracciones aparentes**: son aquellas cuyo numerador es múltiplo del denominador. **Fracciones impropias**: son aquellas en las que el numerador es mayor que el denominador, pero no múltiplo.
[2] El `"___"` debe reemplazarse por el nombre del robot.
[3] El primer `"___"` debe reemplazarse por el nombre completo de la persona y el segundo `"___"` debe reemplazarse por el nombre del robot.
[4] No olvides instanciar un objeto `Robot` en el `main`.

- `String generarAleatorio(int) {…}`

  Devuelve una cadena que representa un valor de password aleatorio cuya longitud coincida con el parámetro recibido. Si el parámetro es menor que `6`, devuelve `null`.

Además, deben poder crearse passwords con o sin valor inicial, por ello es que la clase contará con un constructor sobrecargado:

- `Password(String) {…}`

  Crea un password cuyo valor viene dado por parámetro.

- `Password() {…}`

  Crea un password cuyo valor se crea automáticamente.

Relaciones 1 a 1 y de dependencia entre clases

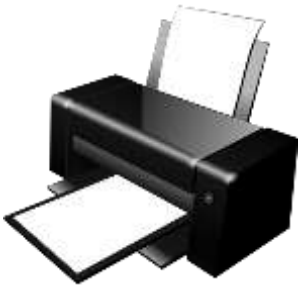
9) Refactoreá la clase `Persona` del **ejercicio 1)**, cambiando el año de nacimiento por su fecha de nacimiento y agregando el atributo `domicilio`, que contenga calle, altura y barrio.

10) Refactoreá la clase `CuentaBancaria` del **ejercicio 2)**, agregando el atributo `titular`, que representa a la persona titular de la cuenta, y el atributo `fechaDeApertura`.

11) Utilizando las clases generadas hasta ahora, codificá en Java la siguiente situación:

> *Una cuenta bancaria de tipo caja de ahorros le pertenece a Fulano Gomez, nacido el 16/04/1990 y otra de tipo cuenta corriente le pertenece a Mengana Torres, nacida el 23/11/1991. Ambos están casados y viven juntos en Av. Triunvirato 3174, Villa Ortúzar.*

Supongamos que Fulano y Mengana se mudan a nuevo hogar: **¿Hay que cambiar el domicilio de cada uno o basta con cambiar uno de los dos?**



12) Desarrollar la clase `ImpresoraMonocromatica`. Sus atributos serán si está o no encendida, la cantidad de hojas actualmente en su bandeja y el nivel de tinta negra. Inicialmente, toda impresora está apagada, sin hojas y con nivel de tinta en `100`.

Debe responder a los siguientes métodos:

- `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.

- `void imprimir(Documento) {…}`

  Imprime en consola la fecha del documento[5], junto a su título y cuerpo, en el siguiente formato:

```
Fecha                    **Titulo**

Cuerpo
```

[5] Modelá la clase correspondiente.

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 y cantidad de hojas suficientes.

13) Se desea implementar la lógica de un dispositivo POSNET que procesa pagos con tarjetas de crédito. Las tarjetas de crédito guardan el nombre de la entidad financiera a la que pertenecen (únicamente Visa, MasterCard o Maestro), el nombre de la entidad bancaria, el número de tarjeta, el saldo disponible y los datos del titular (nombre, apellido, fecha de nacimiento y domicilio). Cada vez que se cree una nueva tarjeta, deberán indicarse todos estos datos.

A la hora de abonar, el POSNET recibiría la tarjeta con la que desea hacerse el pago, junto con el monto que se desea abonar y la cantidad de cuotas (de 1 a 6).

Si el pago es en 1 cuota, no se genera ningún recargo, de lo contrario, el monto se incrementará en un 3% por cada cuota superior a 1. (Ejemplo: Pagar en 4 cuotas representará un 9% de incremento).

El POSNET debe chequear que la tarjeta tenga saldo suficiente para poder efectuar el pago junto con el recargo, si hubiese. En caso de éxito, debe generar y retornar (no mostrar) un ticket donde consten los siguientes datos:

- Nombre y apellido del cliente.
- Monto total a pagar.
- Monto de cada cuota.

Si la operación no tuvo éxito, se retornará `null`.

A) Desarrollar, en la clase `Posnet`, el método `efectuarPago()`, cuyos parámetros, lógica y valor de retorno deben deducirse según lo enunciado. Desarrollar también los métodos derivados que puedan surgir de él para conseguir el objetivo.

B) Desarrollar el método `main` del proyecto y generar las instancias necesarias para poder efectuar un pago de $10000 en 5 cuotas, usando una tarjeta de crédito con saldo disponible de $15000 (el resto de los datos, pueden inventarse a tu gusto).

14) Realizá el prototipo de una máquina que preparará café. Ella consta de una marca, si está encendida y además dos componentes principales:

- Un módulo de agua. Tiene un medidor de temperatura y un indicador para saber si requiere o no mantenimiento.
- Un módulo de leche. Tiene un medidor de líquido, de 0 a 5 y una textura para la leche que se sirva (espumosa, normal o líquida). Inicialmente, este módulo está vacío y con la textura establecida de forma líquida.

A la hora de preparar un café, se necesita decirle a la máquina el tipo de café a realizar (expresso, latte o lágrima).

Nota: Por ahora no modelamos la necesidad de colocar un filtro con café molido o una cápsula para realizar el café.

Como la cantidad de leche en el módulo es limitada, entonces cada tipo de café le representa un costo a la máquina:

- Expresso: No lleva leche.
- Latte: 2 puntos de leche. La textura de la leche debe ser espumosa.
- Lágrima: 3 puntos de leche. La textura de la leche debe ser líquida.

Para cualquier tipo de café, el módulo de agua debe estar a una temperatura superior a 70°C y no requerir mantenimiento Presuponé que el agua siempre va a alcanzar (provendría de la red). Además, para que todo pueda ocurrir, la máquina debe estar encendida.

Finalmente, cuando se le pida un café a la máquina (y dado que aún es un prototipo limitado), esta deberá simplemente responder con un mensaje en su pantalla que indique si hubo algún error (aclarar cuál), de lo contrario, mostrará una leyenda que diga: `"El café <poner_tipo_de_café> se ha servido correctamente"`.

A) Desarrollar las clases correspondientes para modelar lo enunciado.

B) Desarrollar, en la clase `MaquinaDeCafe`, el método `servirCafe()`, cuyos parámetros, lógica y valor de retorno deben deducirse según lo enunciado. Desarrollar también los métodos derivados que puedan surgir de él para conseguir el objetivo.

C) Desarrollar el método `main` del proyecto y generar las instancias necesarias para poder preparar los siguientes cafés (en este orden):

a. Un latte.
b. Otro latte.
c. Un expresso
d. Una lágrima

Si se requiriera cargar el módulo de leche, hacerlo.