EJERCICIO COMPLEMENTARIO DE POO (OPCIONAL)

Enunciado

Desarrollar un programa utilizando el paradigma orientado a objetos, para controlar el taller mecánico “DH FIERROS”.

Se necesita registrar de los clientes, el nombre, apellido, numero de documento de identidad, fecha de nacimiento y más de un número de teléfono (código y número) si lo tuviera. Los clientes son los propietarios de los vehículos que el taller recibe para la reparación o diagnóstico de una falla. Cada vehículo lo atiende siempre un mismo mecánico.

Los diagnósticos cuestan $1000 y se precisa almacenar el número de identificación, fecha de emisión y precio. En cambio, para las reparaciones, se almacena un número de identificación, fecha de entrada, fecha de entrega, fecha de salida, informe y el precio que depende de la complejidad de las fallas (baja $2000, media $3500, alta $6000 y muy alta $10000). Hay que tener en cuenta, que la empresa ofrece un servicio vip para clientes en donde pagan una cuota fija mensual de $500 y obtienen un descuento especial del 30% en el costo de la reparación.

El taller atiende ciertos tipos de vehículos, como es una coupe, sedan y/o descapotable (se detalla el estado del techo). Se requiere registrar el número de patente, modelo, marca, año de fabricación, color y estado del mismo. Además, es importante especificar el motor (modelo, potencia y tipo de combustible que usa: nafta, diésel y euro diésel); las ruedas (marca, modelo y estado); los accesorios como un reproductor de música (marca, modelo y estado), butacas (material, tipo, color y estado); etc.

Los mecánicos reparan, diagnostican y prueban los vehículos. De ellos se precisa el nombre, apellido, numero de documento de identidad, fecha de nacimiento y domicilio. El sueldo bruto se calcula en relación a la antigüedad, señority, preceptismo, bono (si lo hubiera), cantidad de horas extras (tienen un valor es de $125) y el descuento del 17% del sueldo bruto que corresponde a los impuestos tributarios.

Todos los mecánicos (sin importar el señority) deben saber conducir un vehículo:

- Un ayudante es quien conduce un vehículo reparado o diagnosticado desde el circuito de prueba hasta el estacionamiento.
- Un oficial es quien conduce un vehículo desde el estacionamiento hasta la fosa o auto-elevador para ser reparado o diagnosticado.
- Un experto es quien conduce un vehículo reparado o diagnosticado por el circuito de prueba.

El señority se clasifica en ayudante (requiere de un certificado de estudio en mecánica), oficial (al menos 5 años o más de experiencia) y experto (más de 9 años). Es así que, el sueldo mensual de un ayudante es $2000, oficial $3500 y el experto $5000 más un bono de $100 por cada año extra de antigüedad).

Finalmente, se necesita hacer una demostración del funcionamiento del sistema con al menos 3 casos de prueba para cada clase. No olvidemos implementar la lógica del comportamiento.

Desarrollo:

1. Identifiquemos y marquemos en el enunciado las clases, atributos y métodos.
2. Arranquemos con el diagrama de clases (UML).
3. Diseñemos las clases, atributos y métodos.
4. Definamos y revisemos los modificadores de acceso de los atributos y métodos de cada clase.
5. Definamos que constructores son necesarios
6. Analicemos si se puede aplicar herencia con una superclase concreta o abstracta.
7. Razonemos que métodos irían en la superclase y que otros en las subclases. Hagámonos las siguientes preguntas:
    a. ¿Requerimos sobrescribir un método en una o ciertas subclases?
    b. ¿Evaluamos si un método necesita ser implementado por todas las subclases? De ser así, podemos declararlo como método abstracto dentro de una clase abstracta.
    c. ¿Necesitamos sobrecargar un método que representa una misma tarea en la superclase o subclase?
8. Analicemos cómo se relacionan las clases y determinemos qué tipos de relaciones aplican mejor para cada caso. Para esto, es clave encontrar una justificación del porqué.
9. Precisemos la multiplicidad en las relaciones.
10. Codifiquemos el diagrama en IntelliJ.
11. Puntualicemos los Getters y Setters que son necesarios.
12. Definamos los métodos toStrig(), equals() y hasCode().
13. Definamos la lógica de los métodos.
14. Por último, hagamos y definamos una clase demo para ejecutar y probar nuestro sistema.

Entregables:

- Un diagrama de clases.
- Código fuente del sistema.