DAW – Programación
Recuperación 2ª Evaluación 4 de Abril de 2019
NOMBRE:_____

ENUNCIADO

Se desea realizar una aplicación que permita a los mecánicos de un garaje registrar, consultar y actualizar los trabajos (reparaciones y revisiones) que han sido realizados o que están en proceso de realización en el garaje.
Cada trabajo se identifica unívocamente por su "identificador de trabajo". El "identificador de trabajo" es un código de 8 caracteres alfanuméricos cuyo primer carácter es T seguido 7 dígitos numéricos y que se asocia con el trabajo en el momento que se registra. El primer trabajo registrado tendrá el identificador "T0000000", el segundo "T0000001" y así sucesivamente, y será asignado por el programa.
Los trabajos incluyen una pequeña descripción de la reparación o revisión a realizar.
Todos los trabajos incluyen el número de horas que van siendo necesarias para su realización. Al crear un trabajo el número de horas es 0. El número de horas irá aumentando a medida que los mecánicos van dedicando tiempo a realizar la reparación o la revisión. Cuando el trabajo se ha finalizado se marca como "finalizado" y el número de horas no puede volver a cambiarse.
Todos los trabajos deben incluir un método *incrementarHoras(int horas)* que incrementará el número de horas dedicadas a dicho trabajo, siempre y cuando el trabajo no esté en estado finalizado.
Las reparaciones incluyen el precio del material utilizado (piezas o pintura). Al registrar una reparación el precio del material es 0 y va aumentando a medida que los mecánicos van utilizando material en la reparación. Las reparaciones tendrán un método llamado *usarMaterial(double precio)* que aumentará el importe total del material utilizado en la reparación. Para asegurarnos la implementación del método *usarMaterial*, las reparaciones implementaran una *interfaz* donde se defina dicho método. Una vez que la reparación se marca como "finalizada" no se puede cambiar el precio del material utilizado.
El precio a cobrar para cada trabajo se compone de una parte fija que resulta de multiplicar el número de horas empleadas por 30€. Además, dependiendo del tipo de trabajo el coste varía de la siguiente manera:

- Reparación mecánica: su precio se calcula como fijo más el coste material multiplicado por 1.1
- Reparación de chapa y pintura: su precio se calcula como fijo más el coste material multiplicado por 1.3
- Revisión: su precio se calcula como fijo más extra independiente del número de horas de 20€.

Para calcular el precio todos los trabajos incluirán un método llamado *calcularPrecio()*.
Para poder mostrar la información de un trabajo todas las clases implementarán un método toString que generará un String con el tipo de trabajo, código, descripción, número de horas, coste del material en el caso de las reparaciones y precio final.
Debido a que es un taller pequeño que sólo dispone de 5 mecánicos, el número máximo de trabajos que puede aceptar es de 10. El programa principal deberá gestionar una única lista de trabajos, para lo que mostrará un menú con las siguientes opciones:

1. Nuevo trabajo
2. Añadir horas
3. Añadir material
4. Finalizar trabajo
5. Eliminar trabajo
6. Listar trabajos
7. Salir

- **Nuevo trabajo**: registrará un nuevo trabajo. Para ello le preguntará al usuario el tipo de trabajo (reparación mecánica, reparación de chapa y pintura o revisión) y la descripción del mismo, añadiendo el nuevo trabajo creado a la lista de trabajos. Si se ha alcanzado el número máximo de trabajos, este no se creará y se informará de ello.
- **Añadir horas:** preguntará al usuario el id del trabajo al que le va a incrementar las horas y el número de horas a incrementar. Si el id del trabajo existe y no está finalizado, incrementará sus horas.
- **Añadir material**: preguntará al usuario el id del trabajo al que le va a añadir el coste de material y el importe de dicho material. Si el id del trabajo existe, no está finalizado y además se trata de una reparación, incrementará el importe del material.
- **Finalizar trabajo**: preguntará por id de un trabajo y, si éste existe, marcará el trabajo como finalizado.
- **Eliminar trabajo**: preguntará por id de un trabajo y, si éste existe, lo eliminará de la lista.
- **Listar trabajos**: muestra por pantalla la información de todos los trabajos de la lista
- **Salir**: finaliza el programa.

Se pide:

- Diseñar el **diagrama de clases** para este sistema, agrupando elementos (atributos y métodos) comunes evitando duplicar elementos. Para ello se podrá añadir clases intermedias o abstractas si se considera oportuno. Se deberá entregar la representación de dicho diagrama por escrito.
- **Implementar las clases descritas** ajustándose al esquema diseñado en el apartado anterior. Se implementarán los métodos que se consideren necesarios y se valorará el correcto uso de los modificadores.

DIAGRAMA DE CLASES

Criterios de calificación			
Apartado	**Criterio**	**Max**	**Puntos**
Compilación	**Si el programa no compila la nota máxima del examen será de 2 puntos**		
Diseño del diagrama de clases	Elección adecuada de las clases añadiendo aquellas necesarias para evitar duplicidad	1,5	
Implementación del diagrama de clases	Correspondencia del diagrama con su implementación	0,5	
	Implementación de los métodos necesarios en cada una de las clases	0,5	
	Correcto uso de los modificadores	0,5	
	Uso correcto de clases abstractas e interfaces	0,5	
	Correcto funcionamiento de los métodos de las clases	1,5	
Programa principal: Gestor	Nuevo trabajo	1,5	
	Añadir horas y material	1	
	Finalizar trabajo	0,5	
	Eliminar trabajo	1	
	Listar trabajos	0,5	
	Funcionamiento del menú	0,5	
Total		10	