Detalles de la tarea de esta unidad.

Enunciado.
A lo largo de esta unidad has ido aprendiendo a crear tus propias clases así como sus distintos miembros (atributos y métodos). Has experimentando con la encapsulación y accesibilidad (modificadores de acceso a miembros), has creado miembros estáticos (de clase) y de instancia (de objeto), has escrito constructores para tus clases, has sobrecargado métodos y los has utilizado en pequeñas aplicaciones. También has tenido tu primer encuentro el concepto de herencia, que ya desarrollarás en unidades más avanzadas junto con otros conceptos avanzados de la Programación Orientada a Objetos.

Una vez finalizada la unidad se puede decir que tienes un dominio adecuado del lenguaje Java como para desarrollar tus propias clases y utilizarlas en una aplicación final que sea capaz de manipular un conjunto de datos simple. Dada esa premisa, esta tarea tendrá como objetivo escribir una pequeña aplicación en Java empleando algunos de los elementos que has aprendido a utilizar.

Se trata de desarrollar una aplicación Java denominado **PROG05_Tarea** que permita gestionar un vehículo. Mediante un menú que aparecerá en pantalla se podrán realizar determinas operaciones:

1. **Nuevo Vehiculo.**
2. **Ver Matrícula.**
3. **Ver Número de Kilómetros.**
4. **Actualizar Kilómetros.**
5. **Ver años de antigüedad.**
6. **Mostrar propietario.**
7. **Mostrar descripción.**
8. **Mostrar Precio**.
9. **Salir.**

La funcionalidad será la siguiente:

- Al iniciar la aplicación se mostrará el menú propuesto.
- Dependiendo de la opción seleccionada por el usuario:
  - **Nuevo Vehículo:** Se creará un nuevo Vehículo, si los datos introducidos por el usuario son correctos, que contendrá la siguiente información (marca, matrícula, número de kilómetros, fecha de matriculación, descripción, precio, nombre del propietario, dni del propietario). Todos los datos serán solicitados por teclado y tan solo habrá que comprobar:
    - Que la fecha de matriculación es anterior a la actual: puedes solicitar por separado dia, mes y año y construir un objeto LocalDate (tienes una referencia en el apartado Consejos y recomendaciones).
    - Que el número de kilómetros es mayor que 0.
    - Que el DNI del propietario es correcto.
    - Si no se cumple algunas de las condiciones se deberá mostrar el correspondiente mensaje de error. En ese caso habrá se mostrará de nuevo el menú principal.
  - **Ver Matrícula:** Mostrará la matrícula del vehículo por pantalla.
  - **Ver Número de Kilómetros**: Mostrará el número de kilómetros por pantalla.
  - **Actualiza Kilómetros**: Permitirá actualizar el número de kilómetros del vehículo. Habrá que tener en cuenta que solo se podrán sumar kilómetros.
  - **Ver años de antigüedad**: Mostrará por pantalla el número de años del vehículo desde que se matriculó, no la fecha de matriculación.
  - **Mostrar propietario:** Mostrará por pantalla el nombre del propietario del vehículo junto a su dni.
  - **Mostrar Descripción:** Mostrará una descripción del vehículo, incluyendo su matrícula y el número de kilómetros que tiene.
  - **Mostrar Pecio**: se mostrará el precio del vehículo.
  - **Salir**: El programa finalizará.

El proyecto de Netbeans constará de dos paquetes, donde se crearán las clases oportunas:

- **PROG05_Ejerc1**: que contendrá la clase **Vehiculo** y la clase **Principal**.
- **PROG05_Ejerc1_util**: contendrá una **clase** con un **métodos estáticos** para realizar validaciones.

La clase `Vehículo` dispondrád de los siguientes métodos:

- **Constructor o constructores.**
- **Métodos get y set para acceder a sus propiedades.**
- **Método `get_Anios()`:** Retorna un entero con el número de años del vehículo.

TEN EN CUENTA

- En la clase vehículo no debe solicitar datos por teclado ni escribir datos en pantalla. Esas operaciones se realizarán en la clase `Principal`.
- En la clase Vehículo no se deben hacer validaciones de datos. Los datos se validan en la clase `Principal` y si son correcto, se instancia el objeto `Vehículo`.
- Debes incluir **una excepción** para la validación del DNI. Es decir, cuando no sea válido, se lanzará una excepción que se gestionará en la clase Principal, desde donde se mostrará el correspondiente mensaje de error.
- La aplicación solo trabajará con un vehículo, por lo tanto, solo utilizará una referencia a un objeto de tipo **Vehículo** en la clase **Principal**.. Si existe un vehículo y el usuario selecciona **Nuevo Vehículo** en el menú, se perderá la información del vehículo existente y se guardará la del nuevo.
- No será necesario realizar comprobaciones de tipo en los datos solicitados por teclado.
- No se podrán mostrar datos de un vehículo si aún no se ha creado: en ese caso habrá que mostrar un mensaje por pantalla.

Piensa en los modificadores de acceso que debes utilizar tanto en atributos y métodos de la clase.

Además del programa deberás escribir también un informe con todas las consideraciones oportunas que se necesiten para entender cómo has realizado la tarea.

IMPORTANTE

- En la cabecera de las clases añade documentación indicando autor y descripción de la clase.
- En la cabecera de cada método añade documentación indicando la funcionalidad que implementa y el valor que devuelve.
- El código fuente Java de esta clase debería incluir **comentarios** en cada atributo (o en cada conjunto de atributos) y método (o en cada conjunto de métodos del mismo tipo) indicando su utilidad.

MEJORA

- Una mejora consistiría en, cuando un dato solicitado no es correcto, mostrar un mensaje y volver a solicitarlo hasta que se introduzca correctamente.

Se deben entregar el proyecto de Netbeans completo. Para empaquetar un proyecto en Netbeans, utiliza la opción **File - Export Project** de Netbeans: generarás un fichero .zip con el contenido completo del proyecto.

Criterios de puntuación. Total 10 puntos.

Para poder empezar a aplicar estos criterios es necesario que la aplicación compile y se ejecute correctamente en un emulador. En caso contrario la puntuación será directamente de 0,00.

Criterios de puntuación.

La clase Vehículo dispone de todos los **atributos** necesarios.	0,50
La clase Vehículo dispone de al menos un **constructor** y funciona correctamente.	0,50
La clase Vehículo dispone de los **métodos públicos de interfaz** necesarios y funcionan correctamente.	1,00
La clase Validar realiza las funciones de validación	2,00
Se utilizan excepciones para informar de errores en DNI.	1,00
La clase Principal implementa la lógica	3,00
La estructura del código es correcta	1,00
Funcionamiento general y comentarios.	1,00
Total	**10,00**

Dado que algunos criterios de puntuación son negativos, podría suceder que el balance final fuera negativo. En tal caso la puntuación final será simplemente de 0,00.

Recursos necesarios para realizar la Tarea.

- Ordenador personal.
- JDK y JRE de Java SE 11 o posterior.
- Entorno de desarrollo NetBeans 11 o posterior.

Consejos y recomendaciones.

Para realizar la aplicación te realizamos la siguiente serie de recomendaciones:

- Básate en los diferentes ejemplos que has tenido que probar durante el estudio de la unidad. Algunos de ellos te podrán servir de mucha ayuda, así que aprovéchalos.
- Puedes crear las clases que creas conveniente para llegar a una solución estructurada (por ejemplo, crear la clase DNI).
- El ejercicio resuelto de la clase DNI, en el cual se hacen comprobaciones de entrada, puede servirte de base para lanzar excepciones cuando no se cumplen ciertas condiciones. Además puedes utilizar ese código para realizar la validación el DNI del propietario del vehículo.
- En la Unidad 2, recuerda que hicimos una pequeña introducción al trabajo con cadenas de caracteres en Java.
- Si utilizas la clase Scanner, después de leer un entero lee una nueva línea para evitar errores de salto de línea. Es decir, para leer un entero siempre se deben ejecutar estas dos instrucciones:

```
valor=sca.nextInt();
sca.nextLine();
```

- Para trabajar con fechas, utiliza las clases disponibles en el API Java a partir de la versión 8 (`LocalDate` y `LocalTime`). En el siguiente enlace tienes ejemplos de uso que te servirán para desarrollar la tarea.
  - https://experto.dev/java-8-fechas-horas/
  - https://docs.oracle.com/en/java/javase/14/docs/api/java.base/java/time/LocalDate.html