Tarea UT6: "Estructuras de almacenamiento de información"

Enunciado.

En la unidad anterior implementamos una aplicación capaz de gestionar un vehículo. Evidentemente se trata de una aplicación muy limitada porque lo más lógico es gestionar un conjunto de vehículos, cada uno de ellos representado por una instancia de la clase Vehículo. Efectivamente, ese es el objetivo de la primera parte de la tarea de esta unidad: implementar la clase Concesionario, que representará un concesionario de venta de coches de segunda mano. El concesionario será capaz de gestionar un máximo de 50 coches. El proyecto se llamará PROG06_Tarea.

Al iniciar nuestra aplicación, se mostrará por pantalla un pequeño menú con las siguientes opciones:

- Nuevo Vehículo.
- Listar Vehiculos.
- Buscar Vehiculo.
- Modificar kms Vehículo.
- Salir.

El funcionamiento de cada opción será el siguiente:

- **Nuevo Vehículo**: Se solicitarán al usuario por teclado los datos de un nuevo vehículo y, si los datos son correctos, se almacenará en el concesionario. En caso de error en algún dato se volverá a solicitar hasta introducir un valor correcto.
- **Listar Vehículos**: Mostrará por pantalla un listado de los vehículos del concesionario, mostrando marca, matrícula, precio, kilómetros y descripción por cada vehículo.
- **Buscar Vehículo**: Se solicitará al usuario una matrícula por teclado (no será necesario validarla) y se buscará en el concesionario un vehículo cuya matrícula coincida con la introducida. Si existe se mostrarán su marca, matrícula y precio por pantalla y en caso contrario el mensaje "No existe vehículo con la matrícula introducida".
- **Modificar kms Vehículo**: Se solicitará al usuario por teclado una matrícula y un número de kilómetros. Si el vehículo con esa matrícula existe, se actualizará su número de kms al valor introducido. Si no existe, se mostrará un mensaje por pantalla.
- **Salir**: la aplicación finalizará.

Después de la ejecución de cada opción, se mostrará de nuevo el menú.

Para la implementación TEN EN CUENTA:

- Puedes reutilizar la clase Vehículo que ya tienes implementada en la unidad anterior.
- Para añadir un nuevo vehículo se deberá validar:
    - Mediante expresiones regulares que:
        - El DNI del propietario es correcto (tan solo el formato).
        - La matrícula del vehículo es correcta (tan solo el formato), es decir, tiene el formato NNNNLLL, donde NNNN es un número entre 0000 y 9999 y LLL son letras mayúsculas del abecedario.
    - Sin expresiones regulares (utilizando métodos de la clase String):
        - Que el nombre del propietario contenga al menos un nombre y dos apellidos (no tratar nombres compuestos) y su longitud no excede de 40 caracteres.

    - Habrá que comprobar que no existe en el concesionario un vehículo con la matrícula introducida. En caso afirmativo, se mostrará un mensaje por pantalla y mostrará el menú.
- Debes implementar la clase Principal que se encargue de:
    - Instanciar un objeto Concesionario.
    - Pintar el menú y solicitar datos por teclado al usuario.
    - Realizar las validaciones de datos de entrada.
    - Mostrar datos por pantalla.
- Debes evitar el uso de la clase Vehiculo desde la clase Principal. Solo se debe utilizar la clase Concesionario.
- Debes implementar la clase Concesionario para aportar la funcionalidad que se especifica. Esta clase deberá contener la estructura de datos necesaria para almacenar los vehículos, con un tamaño máximo de 50. Por otro lado, ten en cuenta que para saber el número de vehículo existentes en la citada estructura, deberás utilizar un atributo de tipo entero. Este atributo te permitirá conocer la posición de inserción de un nuevo vehículo o hasta qué posición debes recorrer la estructura. Sus métodos serán:
    - Constructor o constructores.
    - buscaVehículo: Recibe como parámetro una matrícula, buscar el vehículo en el concesionario y devuelve una cadena con los datos del vehículo o null si el vehículo buscado no existe.
    - insertarVehiculo: Recibe todos los datos de un vehículo y trata de insertarlo en el concesionario. Devuelve 0 si se hizo con éxito, -1 si el concesionario esta vacio y -2 si la matrícula ya existe.
    - listaVehículos: Lista por pantalla los datos de todos los vehículos del concesionario.
    - actualizaKms: Recibe por parámetro una matrícula y un número de kilómetros, busca el vehículo con la cuya matrícula coincida y actualiza sus kilómetros. Devuelve true si se hizo con éxito y false en caso contrario.
- Utiliza los paquetes que consideres oportuno para organizar las clases.
- Ten en cuenta que los métodos de la clase Concesionario no deben mostrar datos por pantalla, a excepción del método que liste los vehículos. Estos métodos deben devolver un valor indicando si la operación se realizó correctamente o no.

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.

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.

La tarea tiene una puntuación total de 10 puntos repartidos de la siguiente forma:

- El concesionario inserta vehículos nuevos: 2 puntos.
- El concesionario lista vehículos por pantalla: 1 punto.
- El concesionario busca un vehículo: 1 punto.
- El concesionario modifica los kilómetros de un vehículo: 2 puntos.
- Funcionamiento de la clase principal: 3 puntos.
- Estructura general de la aplicación: 1 punto.

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.

Te recomendamos, que para llegar a realizar con éxito la tarea, sigas las siguientes indicaciones:

1. Realizar pruebas a medida que escribes código.
2. Comienza con la clase Principal y ve avanzando en la implementación de las clases y métodos que vayas necesitando.
3. Crea expresiones regulares para detectar cuando se trata del DNI o NIE (esta la puedes encontrar en los contenidos). No busques expresiones regulares complicadas, sino aquellas que de forma básica detecten de que se trata. En el siguiente enlace tienes contenidos que te pueden servir de ayuda con las expresiones regulares: https://puntocomnoesunlenguaje.blogspot.com/2013/07/ejemplos-expresiones-regulares-java-split.html

Indicaciones de entrega.

Una vez que tengas terminado el programa (**fichero .zip de Netbeans y el documento explicativo en formato pdf**), comprime ambos en un único archivo zip. El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:

apellido1_apellido2_nombre_ProgUT6.zip

Así por ejemplo la alumna Begoña Sánchez Mañas debería nombrar esta tarea como:

Sanchez_manas_begona_ProgUT6.zip