

## Informe de la PROG06\_Tarea

### Clase Concesionario

Tenemos dos atributos de la clase y una constante.

El primero atributo es un Array de la clase Vehículo (Explicado en la tarea 5 y que en esta no he tocado nada) y el otro atributo es numVehiculos que sirve para llevar el numero de vehículos que tiene el concesionario.

La constante que he creado es el número máximo de vehículos que es 50.

La clase solo tiene un único constructor que lo que hacer es inicializar el array y definir el tamaño de este con la constante.

Y inicializar el numero de vehículos que tiene el concesionario que es cero.

```
public class Concesionario {  
    private Vehiculo[] vehiculos;  
    private int numVehiculos;  
  
    private final int MAXIMO_VEHICULOS = 50;  
}
```

```
/**  
 * CONSTRUCTOR  
 */  
public Concesionario() {  
    this.vehiculos = new Vehiculo[MAXIMO_VEHICULOS];  
    this.numVehiculos = 0;  
}
```

El método **listaVehiculos** lo único que hace es mostrar en pantalla todos los vehículos del array y en caso de que esté vacío pues imprime un mensaje que está vacío.

```
/**  
 * Imprime en pantalla la lista de todos los vehiculos del concesionario  
 */  
public void listaVehiculos() {  
    if (this.numVehiculos != 0) {  
        //Solo si tiene vehiculos  
        System.out.println("-----");  
        System.out.printf("%20s | %10s | %6s | %7s | %s\n", "Marca", "Matricula", "Precio", "Km", "Descripción");  
        System.out.println("-----");  
        for (int i=0; i<this.numVehiculos; i++) {  
            //Recorre el array hasta el numero  
            System.out.printf("%20s | %10s | %6.2f | %7d | %s\n", vehiculos[i].getMarca(),  
                vehiculos[i].getMatricula(),  
                vehiculos[i].getPrecio(),  
                vehiculos[i].getKilometros(),  
                vehiculos[i].getDescripcion());  
        }  
        System.out.println("-----");  
    } else {  
        System.out.println("No hay vehiculos almacenados\n");  
        //Muestra que no hay nada  
    }  
}
```

Método **buscaVehiculo**, al cual se le pasa por parámetro la matricula.

Recorremos el array hasta el numero de vehículos y si hay una coincidencia de matricula pues devolvemos los datos, sino NULL.

```
public String buscaVehiculo(String matricula) {  
    for (int i=0; i<this.numVehiculos; i++) {  
        if (vehiculos[i].getMatricula().equalsIgnoreCase(matricula)) {  
            return "{Marca: " + vehiculos[i].getMarca() + //Al encon  
                " Matricula: " + vehiculos[i].getMatricula() +  
                " Precio: " + vehiculos[i].getPrecio() + "€"}";  
        }  
    }  
    return null; //Si ha recorrido todo el array y no hay ninguna coincide  
}  
  
/**  
 * Imprime en pantalla la lista de todos los vehiculos del concesionario  
 */
```

El método **insertaVehiculo** se le pasa por parámetro todas la propiedades del vehículo que vamos a crear nuevo

Lo primero que hace es comprobar que el numero de vehículos del concesionario es menos al maximo, en caso que este lleno no hace nada y devuelve que está lleno con el -1.

Lo siguiente que hace es buscar si el vehiculo existe en el array, si existe termina y devuelve codigo de existencia que es el -2.

Y por ultimo si todo esta bien crea el objeto en el array y incrementa el numero de vehículo y también retorna código de correcto el 0

El método **actualizaKms** es idéntico al de buscar solo que en vez si le encuentra actualiza los nuevos Kms del objeto.

Método **borrarVehiculo**, lo primero que hacemos es crear un array auxiliar con el mismo tamaño que el array principal de la clase.

Después compruebo que tenga datos, después busque que exista dicho vehículo y posteriormente recorro el array hasta el numero de vehículos existentes, y si la matricula no coincide guardo ese objeto en el array auxiliar y si coincide no hago nada. Cuando el bucle termina tengo el array auxiliar con todos los datos menos el elemento a borrar, y lo que hago es el array auxiliar sobre escribir el array principal y decremento el numero de vehículos en el array.

```
public int insertaVehiculo(String marca, String matricula, int kilometros, LocalDate fecha, String descripcion, float precio, String nombre, int dni) {  
    int valorRetorno; //Uso esta variable para guardar los estados de retorno y solo usar un return  
  
    if (this.numVehiculos < MAXIMO_VEHICULOS) { //Si tenemos menos vehiculo que el máximo  
        if (buscaVehiculo(matricula) == null) { //Si es null es que no existe  
            vehiculos[numVehiculos] = new Vehiculo(marca, matricula, kilometros, fecha, descripcion, precio, nombre, dni); //En la posición que indica crea  
            numVehiculos++; // Incrementamos 1 el número de vehiculos  
            valorRetorno = 0; //Correcto  
        } else { // Existe ese vehiculo  
            valorRetorno = -2;  
        }  
    } else { // Esta lleno el concesionario  
        valorRetorno = -1;  
    }  
    return valorRetorno;  
}
```

```
public boolean actualizaKms(String matricula, int nuevosKms) {  
  
    for (int i=0;i<this.numVehiculos;i++) {  
        if (vehiculos[i].getMatricula().equalsIgnoreCase(matricula)) {  
            vehiculos[i].setKilometros(nuevosKms);  
            return true;  
        }  
    }  
    return false;  
}
```

```
public int borrarVehiculo(String matricula) {  
    Vehiculo[] aux = new Vehiculo[MAXIMO_VEHICULOS];  
  
    int codRetorno; //Donde se guardan los codigos para retornar al final del método  
    int contAux = 0; //Contador para el indice de aux  
  
    if (this.numVehiculos != 0) {  
        if (this.buscaVehiculo(matricula)!=null) {  
            for (int i = 0; i<this.numVehiculos; i++){  
                if (!this.vehiculos[i].getMatricula().equalsIgnoreCase(matricula)) {  
                    aux[contAux] = this.vehiculos[i]; //Guarda el vehiculo que sea distinto al  
                    contAux++; //Aumenta solo cuando se guarda un vehiculo  
                }  
            }  
            this.vehiculos = aux; //Sustitulle el Array Vehiculos por el nuevo crea  
            this.numVehiculos--; //Restamos 1 los vehiculos del concesionario.  
            codRetorno = 0;  
        } else {  
            codRetorno = -2; //El vehiculo no existe;  
        }  
    } else {  
        codRetorno = -1; // No hay vehiculos para borrar  
    }  
    return codRetorno;  
}
```

## Clase Validar

Para chequear el DNI con expresiones regulares tenemos esta clase estática **checkDNI** que se le pasa el dni en formato string.

La expresión regular le digo que es obligatorio que tenga 8 número y un única letra de las que son validas para los DNI.

```
/**
 * Método que valida el formato de un DNI
 *
 * @param dni DNI que hay que chequear el formato
 * @return
 */
public static boolean checkDNI(String dni) {
    return dni.matches("[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE]{1}$");
}
```

El método **checkMatricula** hace una validación de la matricula que le obligo a que sean 4 números seguidos y 3 letras seguidas.

```
/**
 * Método que valida el formato de una matricula 0000LLL
 *
 * @param matricula Matricula que se desea chequear el formato
 * @return
 */
public static boolean checkMatricula(String matricula) {
    return matricula.matches("[0-9]{4}[A-Z]{3}$");
}
```

El método **contarEspacios** es lo que hace contar los espacios intermedios que tiene un texto.

Por si hubiera espacios al principio o al final del texto hago una limpieza de estos espacios y después le voy contando y esa cifra es la que devuelve.

```
/**
 * Método que cuanta los espacios que tiene un string
 *
 * @param cadena Cadena de texto para contar los espacios interiores
 * @return
 */
public static int contarEspacios(String cadena) {
    int posicion, contador = 0;
    String aux = cadena.trim();

    posicion = aux.indexOf(" ");
    while (posicion != -1) {
        contador++;
        posicion = aux.indexOf(" ", posicion + 1);
    }
    return contador;
}
```

## Clase Principal

La clase Principal tiene un método **menú** que es el menú que se muestra en pantalla.

```
public static void menu() {
    System.err.println();
    System.out.println("-----");
    System.out.println("      Concesionario      ");
    System.out.println("-----");
    System.out.println("1. Nuevo Vehiculo");
    System.out.println("2. Listar Vehiculos");
    System.out.println("3. Buscar Vehiculo");
    System.out.println("4. Modificar kms Vehiculo");
    System.out.println("5. Borrar un vehiculo");
    System.out.println("6. Salir");
    System.out.println("-----");
    System.out.println("Seleccione una opción [1,2,3,4,5,6]");
}
```

<p>Esta el método MAIN, en el cual al principio defino todas las variables que utilizo para guardar los datos que solicito al usuario y también tengo variables para el control y verificación del programa,</p>	<pre>public static void main(String[] args) {     Scanner sc = new Scanner(System.in);      boolean salir = false, check = false; //Para el bucle principal     byte opcion;                          //Variable para la selección del      /**      * Variables necesarias para todos los atributos de la clase Vehi      */     String marca;     String matricula;     int kilometros = 0;     int dia, mes, anyo;     LocalDate fecha;     String descripcion;     float precio;     String nombre;     String dniString;     Dni dni; }</pre>
<p>Instancia o creación del objeto concesionario.</p>	<pre>Concesionario concesionario = new Concesionario();</pre>
<p>Si el usuario pulsa el 6 se termina el programa.</p>	<pre>case 6:     salir = true;     break;</pre>
<p>Si pulsa 5 el programa pide una matrícula y llama al método de borrar y muestra en pantalla las diferentes opciones en función de los que devuelva dicho método.</p>	<pre>case 5: //Borra un vehiculo     System.out.println("Matricula del vehiculo que desea borrar:");     matricula = sc.nextLine();      switch (concesionario.borrarVehiculo(matricula)) {         case 0:             System.out.println("Vehiculo borrado con éxito\n");             break;         case -1:             System.out.println("No hay vehiculos que borrar\n");             break;         case -2:             System.out.println("No existe vehiculo con la matricula introducida\n");             break;     }     break;</pre>
<p>Si pulsa 4 nos pide vehículo y nuevos kilómetros y llamamos al método que los cambia.</p>	<pre>case 4: //Modifica los kilometros de un vehiculo     System.out.println("Matricula del vehiculo que desea buscar:");     matricula = sc.nextLine();     System.out.println("Número de Kilometros:");     int km = sc.nextInt();     sc.nextLine();      if (!concesionario.actualizaKms(matricula, km)) {         System.out.println("No se ha podido encontrar el vehiculo que desea cambiar los kilometros\n");     }     break;</pre>
<p>Con la opción 3 lo que hace es buscar un vehículo y visualizar los datos de este sino mensaje de no existencia.</p>	<pre>case 3: //Busca por matricula     System.out.println("Matricula del vehiculo que desea buscar:");     matricula = sc.nextLine();      String result = concesionario.buscaVehiculo(matricula); //Como devuelve un string      if (result==null) {         System.out.println("No existe vehiculo con la matricula introducida\n");     } else {         System.out.println(result);     }     break;</pre>

<p>La opción 2 llama al método y lista los vehículos.</p>	<pre> break; case 2:          //Lista Vehiculos     concesionario.listaVehiculos();     break; </pre>
<p>La opción 1 es idéntica a la descrita en la tarea 5 por lo tanto solo voy a mencionar aquello que es nuevo o que ha cambiado.</p> <p>Chequeamos la matrícula llamando al método estático que lo chequea con expresión regular y se repite en bucle hasta que esta validación sea correcta.</p>	<pre> do {     System.out.println("Introduce matricula del vehiculo:");     matricula = sc.nextLine().toUpperCase();           //convierte todo en mayusculas     check = Validar.checkMatricula(matricula);         //como todo esta en mayusculas solo exigimos A-Z     if (!check) System.out.println("ERROR MATRICULA: Introduzca una matricula válida"); }while(!check); </pre>
<p>La validación del DNI está igual que estaba en la tarea 5 pero he añadido el método de validación por expresión regular, el resto es como he dicho igual que estaba.</p>	<pre> do {     System.out.println("Introducir DNI con su Letra del propietario del vehiculo [00000000T]:");     dniString = sc.nextLine();     dni = new Dni();     check = Validar.checkDNI(dniString);               //Metodo que chequea el formato del DNI     if (check) {         try {             dni.setDni(dniString);                     //Control de la excepción que genera si el DNI es incorrecto             check = true;         } catch (Exception ex) {             System.out.println(ex.getMessage());        //Muestra mensaje de la excepción diciendo que es Invalido             check = false;         }     } }while(!check); </pre>
<p>Y con el nombre usamos el otro método que cuenta los espacios intermedios que tiene un texto y si solo tiene 2 espacios interpreta que es Nombre, Apellido 1 y Apellido 2 y si tiene 1 o más de 2 pues repite hasta que sea correcto.</p>	<pre> do {     System.out.println("Introducir nombre del propietario del vehiculo:");     nombre = sc.nextLine();      check = Validar.contarEspacios(nombre)==2 &amp;&amp; nombre.length()&lt;40; //Si tiene dos espacios es que tiene     if (!check) System.out.println("ERROR Nombre: Tiene que ser asi: NOMBRE APELLIDO1 APELLIDO2"); }while (!check); </pre>
<p>Y si todas las validación son correctas al final llama al método insertar y se le pasan los datos que ha pedido por teclado y que han sido validados.</p>	<pre> //--- Se procede a insertar los datos del vehiculo en el concesionario --- switch (concesionario.insertaVehiculo(marca, matricula, kilometros, fecha, descripcion, precio, nombre, dni)) {     case 0:         System.out.println("Vehiculo insertado con éxito\n");         break;     case -1:         System.out.println("El concesionario está lleno");         break;     case -2:         System.out.println("El Vehiculo con la matricula: " + matricula + " ya existe");         break; } </pre>