Informe de la PROG05_Tarea

Clase Principal

Método public static void main(String[] args)

La variable boleana "**salir**" es utilizada para que se repita constantemente el bucle principal del programa mientra esta sea false.

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    boolean salir = false;
    byte opcion;
    Vehiculo vehiculo = null; // Iniciamos a null
```

La variable "**opcion**" utilizada para que en ella el usuario introduzca la opción del menú que considere.

El objeto "**vehiculo**" que es de la clase Vehiculo donde almacenaremos los datos del vehiculo.

Todo el programa está dentro de un bucle while que como he mencionado se repetirá mientra la variable salir sea false.

```
}while (!salir); //repite mientras salir sea false
```

Menu() es un método el cual muestra en pantalla el menú de opciones.

Una vez el usuario introduce la opción que desea el swich evalua cual es la introducida y ejecuta su correspondiente case.

```
do {
    menu();
    opcion = sc.nextByte();

    switch (opcion){
```

Creamos el objeto vehiculo y para ell lo hacemos con la función nuevoVehiculo

```
case 1: //1. Nuevo vehiculo
    vehiculo = nuevoVehiculo();
    System.out.println("Nuevo Vehiculo creado");
    break;
```

En los casos del 2 al 8 lo primero que hace el programa es comprobar que el objeto vehículo ha sido inicializado y eso lo hace comparando vehículo con null. En caso que se haya creado el objeto continua con lo que corresponda a la opción seleccionada.

En caso que no se hubiera creado el nuevo vehículo, muestra un mensaje que dice que se cree.

```
case 2: //2. Ver Matricula
    if (vehiculo != null) {
        System.out.println("Matricula: " + vehiculo.getMatricula());
    }else {
        System.out.println("Se tiene que crear un nuevo Vehiculo [Opción 1]");
    }
    break;
```

Las opciones 2 y 3 llaman a los métodos Getter correspondientes de la clase vehículo para obtener la información deseada, en este caso Matricula y Kilómetros.

```
case 2: //2. Ver Matrícula
    if (vehiculo != null) {
        System.out.println("Matricula: " + vehiculo.getMatricula());
    }else {
        System.out.println("Se tiene que crear un nuevo Vehiculo [Opción 1]");
    }
    break;

case 3: //3. Ver Número de Kilómetros
    if (vehiculo != null) {
        System.out.println("Kilometros: "+ vehiculo.getKilometros());
    }else {
        System.out.println("Se tiene que crear un nuevo Vehiculo [Opción 1]");
    }
    break;
```

La opción 4 es la de actualizar los kilómetros del vehículo. Donde los nuevos kilómetros que el usuario introduce por teclado les almacenamos en una variable temporal. Los nuevos kilómetros les comparamos con los kilómetros que ya tiene el objeto llamando al método getter y si el nuevo valor es mayor en llama al al método setter de kilómetros y los actualiza.

```
case 4: //4. Actualizar Kilómetros
    if (vehiculo != null) {
        System.out.println("Introduce los nuevos kilometos del vehículo:");
        int numtemp = sc.nextInt();
        sc.nextLine();
        if (numtemp>vehiculo.getKilometros()) {    //Comprueba que los kilometros qu
            vehiculo.setKilometros(numtemp);
        } else {
            System.out.println("No puede tener menos kilometros");
        }
    }else {
        System.out.println("Se tiene que crear un nuevo Vehiculo [Opción 1]");
    }
    break;
```

En los casos 5, 6, 7 y 8 llama a los diferentes métodos Get que devuelven valores de los atributos.

```
case 5: //Ver años de antiguedad
    if (vehiculo != null) {
        System.out.println("El vehiculo tiene " + vehiculo.getAnios() + " años");
    }else {
        System.out.println("Se tiene que crear un nuevo Vehiculo [Opción 1]");
    }

    break;
case 6: //6. Mostrar Propietario
    if (vehiculo != null) {
        System.out.println("Propietario " + vehiculo.getNombrePropietaro());
        System.out.println("con DNI: " + vehiculo.getDniPropietario());
    }else {
        System.out.println("Se tiene que crear un nuevo Vehiculo [Opción 1]");
    }
    break;
case 7: //7. Mostrar descripción
    if (vehiculo != null) {
        System.out.println("Descripcion: " + vehiculo.getDescripcion());
        System.out.println("Matricula: " + vehiculo.getMatricula());
        System.out.println("Kilometros: " + vehiculo.getKilometros());
    }else {
        System.out.println("Se tiene que crear un nuevo Vehiculo [Opción 1]");
    }
    break;
case 8: //8. Mostrar precio
    if (vehiculo != null) {
        System.out.printf("El precio es: %.2f €\n",vehiculo.getPrecio());

    }else {
        System.out.println("Se tiene que crear un nuevo Vehiculo [Opción 1]");
    }
    break;
```

La opción 9 cambia el valor de salir, para que termine el while principal.

```
case 9: //9. Salir
    salir=true;
    break;
```

Método public static void menu()

Un método de la clase Principal que muestra en pantalla el menú de opciones

```java
/**
 * Metodo que muestra en pantalla el menú
 */
public static void menu() {
    System.err.println();
    System.out.println("------------------------------");
    System.out.println("    Gestión de un Vehiculo    ");
    System.out.println("------------------------------");
    System.out.println("1. Nuevo Vehiculo");
    System.out.println("2. Ver Matrícula");
    System.out.println("3. Ver Número de Kilómetros");
    System.out.println("4. Actualizar Kilómetros");
    System.out.println("5. Ver años de antigüedad");
    System.out.println("6. Mostrar propietario");
    System.out.println("7. Mostrar descripción");
    System.out.println("8. Mostrar Precio");
    System.out.println("9. Salir");
    System.out.println("------------------------------");
    System.out.println("Seleccione una opción [1,2,3,4,5,6,7,8,9]");

}
```

Método public static Vehiculo nuevoVehiculo()

El método nuevoVehiculo retorna un objeto de tipo Vehículo.
Las variables locales son para almacenar los datos que le vamos pidiendo al usuario.

```java
public static Vehiculo nuevoVehiculo(){
    Scanner sc = new Scanner(System.in);
    /**
     * Variables temporales para almacenar los datos
     */
    String marca;
    String matricula;
    int kilometros = 0;
    int dia, mes, anyo;
    LocalDate fecha;
    String descripcion;
    float precio;
    String nombre;
    String dniString;
    Dni dni;
```

Como los kilómetros nos pide que tiene que ser mayor de 0 una vez que el usuario introduce el valor Validamos con un método estático de la clase validar si los kilómetros son mayores que cero.
Y todo esta dentro de un bucle do-while que se repite hasta que el usuario introduce correctamente los kilómetros.

```java
do {
    System.out.println("Introduce los kilometros que tiene el vehiculo");
    kilometros = sc.nextInt();
    sc.nextLine();

    check = Validar.checkKilometros(kilometros); //Método de la clase Util que valida si e

    if (!check) {
        System.out.println("ERROR KM: los Km del vehiculos tiene que ser mayor que 0");
    }
}while(!check); //Repite hasta que se introduzca un kilometro mayor que 0
```

Para introducir la fecha el se le pide al usuario por separado el día, mes y año, después se crea el objeto fecha con los datos introducidos y esa fecha es validada para comprobar que es anterior a la fecha actual.
Este proceso se repite hasta que se introduzca una fecha válida.

```
do {
    System.out.println("----- Fecha de Matriculación -----");
    System.out.println("Introduce el día:");
    dia = sc.nextInt();
    sc.nextLine();

    System.out.println("Introduce el mes:");
    mes = sc.nextInt();
    sc.nextLine();

    System.out.println("Introduce el año:");
    anyo = sc.nextInt();
    sc.nextLine();

    fecha = LocalDate.of(anyo, mes, dia); //Inicia el Objeto con los datos creados

    check = Validar.checkFecha(fecha);   //Método de la clase Util que valida si la fecha es ant

    if (!check) {
        System.out.println("ERROR FECHA: La fecha debe ser anterior a hoy");
    }

}while(!check); //Se repite mientras la fecha no sea anterior
```

Bucle que se repite hasta que el usuario introduce un DNI con su letra correctamente.

La validación del dni lo hacemos que en método setter del DNI que que en caso de no ser un dni valido lanza una excepción y dicha excepción la estamos controlando con el bloque try-catch

```
do {
    System.out.println("Introducir DNI con su Letra del propietario del vehiculo [00000000T]:");
    dniString = sc.nextLine();
    dni = new Dni();

    try {                          //Control de la excepción que genera si el DNI es incorrecto
        dni.setDni(dniString);
        check = true;
    } catch (Exception ex) {
        System.out.println(ex.getMessage()); //Muestra mensaje de la excepción diciendo que es Inva
        check = false;
        //Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
    }
}while(!check);
```

Clase Vehiculo

Atributos

Todos los atributos de la clase son privados y para acceder a ellos ya sea para obtener su contenido como para modificarlo se tiene que hacer con los diferentes métodos que tiene la clase.

```
public class Vehiculo {
    /**
     * Atributos de la Clase
     */
    private String marca;
    private String matricula;
    private int kilometros;
    private LocalDate fechaMatriculacion;
    private String descripcion;
    private float precio;
    private String nombrePropietaro;
    private Dni dniPropietario;
```

Constructor

Esta clase solo consta de un constructor, el cual tiene como parámetros todos los atributos que se inicializan.

Seguidamente tenemos los Getter y los Setter de los cuales solo voy a comentar los que tengan algo diferentes a modificar un atributo o devolverle.

```
public Vehiculo(String marca, String matricula, int kilometros,
                LocalDate fechaMatriculacion, String descripcion,
                float precio, String nombrePropietaro, Dni dniPropietario) {

    this.marca = marca;
    this.matricula = matricula;
    this.kilometros = kilometros;
    this.fechaMatriculacion = fechaMatriculacion;
    this.descripcion = descripcion;
    this.precio = precio;
    this.nombrePropietaro = nombrePropietaro;
    this.dniPropietario = dniPropietario;
}
```

Métodos a mencionar

Este método lo que devuelve es un entero con los años que tiene el vehículo, para hacer eso lo que hace es restar al año en curso el año de matriculación.

```
/**
 * Metodo que devuelve los años que tiene el vehiculo, restado al año actual e
 *
 * @return retorna los años que tiene el vehiculos
 */
public int get_Anios(){
    return (LocalDate.now().getYear() - this.fechaMatriculacion.getYear());
}
```

Este método getter devuelve una cadena con el numero de dni y la letra que esta almacenado en un Objeto de la clase Dni

```
    */
   public String getDniPropietario() {
       return dniPropietario.getNif();
   }
```

Del atributo “marca” solo tiene un getter para obtener la marca del vehículo, no tiene setter ya que como en el constructor es obligatorio ponerlo por lo tanto una vez establecido no se tiene que poder cambiar.

```
/**
 * Método para obtener la marca del vehiculo
 *
 * @return Retorna una cadena con la marca del vehiculo
 */
public String getMarca() {
    return marca;
}
```

Clase Validar

Método public static boolean checkKilometros(int km)

Método estático que valida si los kilómetros que se le pasan por parámetro son mayores que cero.

```
public static boolean checkKilometros(int km){
    boolean ok = false;
    if (km>0) {
        ok = true;
    }
    return ok;
}
```

Método public static boolean checkFecha(LocalDate fecha)

Método estático que valida si la fecha data en anterior a la fecha actual y para ello usa un método del objeto creado LocalDate.

```
 */
public static boolean checkFecha(LocalDate fecha){
    boolean ok = false;
    if (fecha.isBefore(LocalDate.now())) {
        ok = true;
    }
    return ok;

}
```

Clase Dni

Atributos

La clase Dni solo tiene un atributo que es el numero DNI de tipo entero.

Tambien cuenta con diferentes métodos getter y setter y varios estáticos para hacer las validaciones.

```
 */
public class Dni {
    private int numDni;
```

Método public void setDni(String numNIF) throws Exception

El método setDni es utilizado para almacenar en la variable privada el numero de DNI y hace dentro la validación de este y en caso que el DNI no sea correcto este lanza una excepción que es la que he explicado en la clase Principal

```
public void setDni(String numNIF) throws Exception {

    if (Dni.validarNIF(numNIF)) {
        this.numDni = Dni.extraerNumeroNIF(numNIF);
    } else {
        throw new Exception ("NIF Invalido: " + numNIF);
    }

}
```

Método public static boolean validarNIF(String nif)

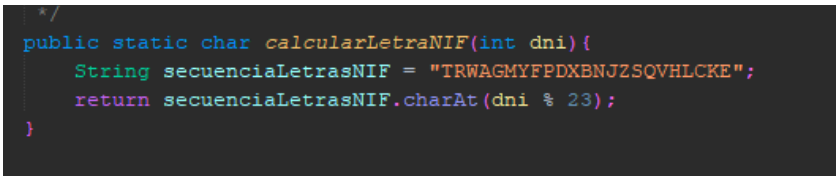
Este método estatico es el utilizado para validar si el NIF dado esta correctamente formado, lo que viene hacer es separar la letra del numero y con el numero genera la letra si esta coincide con la que venia en el NIF.

```
public static boolean validarNIF(String nif) {
    int numero;
    char letra;
    boolean valido = false;


    if (nif== null) {
        valido = false;
    } else if (nif.length()<8 || nif.length()>9) {
        valido = false;
    } else {
        numero = Dni.extraerNumeroNIF(nif);
        letra = Dni.extraerLetraNIF(nif);

        if (letra == Dni.calcularLetraNIF(numero)) {
            valido = true;
        }
    }
    return valido;
}
```

Método public String getNif()

Para obtener el número de DNI lo que hacemos es retornar en forma de cadena el numero de DNIy calcula la letra que le corresponde, ya que como dije solo almacena el número.

```
 */
public String getNif() {
    return (""+this.numDni+Dni.calcularLetraNIF(numDni));
}
```

Método public static char calcularLetraNIF(int dni)	
Este método es usado para calcular la letra del DNI y la devuelve en forma de Caracter.	

```
 */
public static char calcularLetraNIF(int dni){
    String secuenciaLetrasNIF = "TRWAGMYFPDXBNJZSQVHLCKE";
    return secuenciaLetrasNIF.charAt(dni % 23);
}
```