5.D. Encapsulación, control de acceso y visibilidad.

1. Encapsulación, control de acceso y visibilidad.

1.3. Ejercicio resuelto.

Vamos a intentar implementar una clase que incluya todo lo que has visto hasta ahora. Se desea crear una clase que represente un **DNI español** y que tenga las siguientes características:

- La clase almacenará el número de DNI en un `int`, sin guardar la letra, pues se puede calcular a partir del número. Este atributo será privado a la clase. Formato del atributo: `private int numDNI`.
- Para acceder al DNI se dispondrá de dos métodos **obtener** (`get`), uno que proporcionará el número de DNI (sólo las cifras numéricas) y otro que devolverá el NIF completo (incluida la letra). El formato del método será:

```
* public int obtenerDNI ().
```

```
* public String obtenerNIF ().
```

- Para modificar el DNI se dispondrá de dos métodos establecer (`set`), que permitirán modificar el DNI. Uno en el que habrá que proporcionar el NIF completo (número y letra). Y otro en el que únicamente será necesario proporcionar el DNI (las siete u ocho cifras). Si el DNI/NIF es incorrecto se debería lanzar algún tipo de **excepción**. El formato de los métodos (**sobrecargados**) será:

```
* public void establecer (String nif) throws ...
```

```
* public void establecer (int dni) throws ...
```

- La clase dispondrá de algunos métodos internos privados para calcular la letra de un número de DNI cualquiera, para comprobar si un DNI con su letra es válido, para extraer la letra de un NIF, etc. Aquellos métodos que no utilicen ninguna variable de objeto podrían declararse como estáticos (pertenecientes a la clase). Formato de los métodos:

```
* private static char calcularLetraNIF (int dni).
```

```
* private boolean validarNIF (String nif).
```

```
* private static char extraerLetraNIF (String nif).
```

```
* private static int extraerNumeroNIF (String nif).
```

Para calcular la letra NIF correspondiente a un número de DNI puedes consultar el artículo sobre el NIF de la Wikipedia:

Artículo en la Wikipedia sobre el Número de Identificación Fiscal (NIF).

Solución

La clase tendrá un único atributo de objeto: **el número de DNI**.

private int numDNI;

Está claro que para poder trabajar con los DNI/NIF vas a necesitar implementar el algoritmo para calcular la letra de un número de DNI. Para ello puedes crear un método (que en principio podría ser privado) que realice ese cálculo. Para facilitar la implementación de ese método, crearemos un arrray estático y constante (final) con las letras posibles que puede tener un NIF y en el orden adecuado para la aplicación del algoritmo de cálculo de la letra (algoritmo conocido como **módulo 23**):

private static final String LETRAS_DNI= "TRWAGMYFPDXBNJZSQVHLCKE";

Con esta cadena disponible, es muy sencillo implementar el algoritmo del **módulo 23**:

private static char calcularLetraNIF (int dni) {

```
 char letra;


 // Cálculo de la letra NIF

 letra= LETRAS_DNI.charAt(dni % 23);
```

```
    // Devolución de la letra NIF
    return letra;
}
```

Este método estático ha sido definido como privado, aunque también podría haber sido definido como público para que otros objetos pudieran hacer uso de él (típico ejemplo de uso de un método estático).

Para poder manipular adecuadamente la cadena NIF, podemos crear un par de métodos para extraer el número de DNI o la letra a partir de una cadena NIF. Ambos métodos pueden declararse estáticos y privados (aunque no es la única posibilidad):

```
private static char extraerLetraNIF (String nif) {
    char letra=   nif.charAt(nif.length()-1);
    return letra;
}
private static int extraerNumeroNIF (String nif) {
    int numero= Integer.parseInt(nif.substring(0, nif.length()-1));
    return numero;
}
```

Una vez que disponemos de todos estos métodos es bastante sencillo escribir un método de comprobación de la validez de un NIF:

- Extracción del número.
- Extracción de la letra.
- Cálculo de la letra a partir del número.
- Comparación de la letra extraída con la letra calculada.

De manera que el método nos podría quedar:

```
private static boolean validarNIF (String nif) {
    boolean valido= true;   // Suponemos el NIF válido mientras no se encuentre algún fallo
    char letra_calculada;
    char letra_leida;
    int  dni_leido;
    if (nif == null) {  // El parámetro debe ser un objeto no vacío
        valido= false;
    }
    else if (nif.length()<8 || nif.length()>9) {    // La cadena debe estar entre 8(7+1) y 9(8+1) caracteres
        valido= false;
    }
    else {
        letra_leida= DNI.extraerLetraNIF (nif);    // Extraemos la letra de NIF (letra)
        dni_leido= DNI.extraerNumeroNIF (nif);  // Extraemos el número de DNI (int)
        letra_calculada= DNI.calcularLetraNIF(dni_leido);  // Calculamos la letra de NIF a partir del número extraído
        if (letra_leida == letra_calculada) {   // Comparamos la letra extraída con la calculada
            // Todas las comprobaciones han resultado válidas. El NIF es válido.
            valido= true;
        }
        else {
```

```
            valido= false;
        }
    }
        return valido;
}
```

En el código de este método puedes comprobar que se hace uso de los métodos estáticos colocando explícitamente el nombre de la clase:

* DNI.extraerLetraNIF.
* DNI.extraerNumeroNIF.
* DNI.calcularLetraNIF.

En realidad en este caso no habría sido necesario pues estamos en el interior de la clase, pero si finalmente hubiéramos decidido hacer públicos estos métodos, así es como habría que llamarlos desde fuera (usando el nombre de la clase y no el de una instancia).

Y por último tan solo quedarían por implementar los métodos públicos (la interfaz):

- Los dos métodos **obtener** (get). Obtener el NIF (String) u obtener el DNI (int).
- Los dos métodos **establecer** (set). A partir de un int y a partir de un String.

En el primer caso habrá que devolver información añadiéndole (si es necesario) información adicional calculada, y en el segundo habrá que realizar una serie de comprobaciones antes de proceder a almacenar el nuevo valor de DNI/NIF.

El código de los métodos **obtener** podría quedar así;

```
public String obtenerNIF () {
  // Variables locales
    String cadenaNIF;   // NIF con letra para devolver
    char letraNIF;      // Letra del número de NIF calculado
    // Cálculo de la letra del NIF
    letraNIF= DNI.calcularLetraNIF (numDNI);
    // Construcción de la cadena del DNI: número + letra
    cadenaNIF= Integer.toString(numDNI) + String.valueOf(letraNIF);
    // Devolución del resultado
    return cadenaNIF;
}
public int obtenerDNI () {
    return numDNI;
}
```

En el caso de los métodos establecer (**método establecer sobrecargado**) podemos lanzar una excepción básica con un mensaje de error de "NIF/DNI inválido" para que la reciba el objeto que utilice este método. De esta manera podría controlarse el error de un posible establecimiento de valores de NIF/DNI inválido.

El código de los métodos **establecer** podría quedar así;

```
public void establecer (String nif) throws Exception {
    if (validarNIF (nif)) { // Valor válido: lo almacenamos
        this.numDNI= DNI.extraerNumeroNIF(nif);
    }
    else { // Valor inválido: lanzamos una excepción
        throw new Exception ("NIF inválido: " + nif);
```

```
    }
}
public void establecer (int dni) throws Exception {
    // Comprobación de rangos
    if (dni>999999 && dni<99999999) {
        this.numDNI= dni; // Valor válido: lo almacenamos
    }
    else { // Valor inválido: lanzamos una excepción
        throw new Exception ("DNI inválido: " + String.valueOf(dni));
    }
}
```

El código completo de la clase DNI podría ser:

```
/**-----------------------------------------------------------------
 * Clase DNI
 -----------------------------------------------------------------*/

    public class DNI {
        // Atributos estáticos

        // Cadena con las letras posibles del DNI ordenados para el cálculo de DNI
        private static final String LETRAS_DNI= "TRWAGMYFPDXBNJZSQVHLCKE";

        // Atributos de objeto
        private int numDNI;

        // Métodos

        public String obtenerNIF () {
          // Variables locales
            String cadenaNIF;   // NIF con letra para devolver
            char letraNIF;      // Letra del número de NIF calculado

            // Cálculo de la letra del NIF
            letraNIF= calcularLetraNIF (numDNI);
```

```
    // Construcción de la cadena del DNI: número + letra
    cadenaNIF= Integer.toString(numDNI) + String.valueOf(letraNIF);

    // Devolución del resultado
    return cadenaNIF;
}

public int obtenerDNI () {
    return numDNI;
}

public void establecer (String nif) throws Exception {
    if (DNI.validarNIF (nif)) { // Valor válido: lo almacenamos
        this.numDNI= DNI.extraerNumeroNIF(nif);
    }
    else { // Valor inválido: lanzamos una excepción
        throw new Exception ("NIF inválido: " + nif);
    }

}

public void establecer (int dni) throws Exception {

    // Comprobación de rangos
    if (dni>999999 && dni<99999999) {
        this.numDNI= dni; // Valor válido: lo almacenamos
    }
    else { // Valor inválido: lanzamos una excepción
        throw new Exception ("DNI inválido: " + String.valueOf(dni));
    }
}

private static char calcularLetraNIF (int dni) {
    char letra;

    // Cálculo de la letra NIF
    letra= LETRAS_DNI.charAt(dni % 23);

    // Devolución de la letra NIF
```

```
    return letra;
}

private static char extraerLetraNIF (String nif) {
    char letra=   nif.charAt(nif.length()-1);
    return letra;
}

private static int extraerNumeroNIF (String nif) {
    int numero= Integer.parseInt(nif.substring(0, nif.length()-1));
    return numero;
}

private static boolean validarNIF (String nif) {
    boolean valido= true;   // Suponemos el NIF válido mientras no se encuentre algún fallo
    char letra_calculada;
    char letra_leida;
    int  dni_leido;

    if (nif == null) {  // El parámetro debe ser un objeto no vacío
       valido= false;
    }
    else if (nif.length()<8 || nif.length()>9) {    // La cadena debe estar entre 8(7+1) y 9(8+1) caracteres
       valido= false;
    }
    else {
       letra_leida= DNI.extraerLetraNIF (nif);    // Extraemos la letra de NIF (letra)
       dni_leido= DNI.extraerNumeroNIF (nif);  // Extraemos el número de DNI (int)
       letra_calculada= DNI.calcularLetraNIF(dni_leido);  // Calculamos la letra de NIF a partir del número extraído
       if (letra_leida == letra_calculada) {   // Comparamos la letra extraída con la calculada
          // Todas las comprobaciones han resultado válidas. El NIF es válido.
          valido= true;
       }
       else {
             valido= false;
       }
    }

       return valido;
```

```
    }
}
```