

Validador de Documentos

Ejercicio práctico BootCamp by Nunsys.

Nicolás Gabarrón Blaya - 5 de abril de 2022



ÍNDICE

| | |
|---|----------|
| Descripción del problema. | 3 |
| ¿Cómo he realizado la refactorización? | 4 |
| Cambio 1 (commit bc46906). | 4 |
| Cambio 2 (commit cfd9009). | 5 |
| Cambio 3 (commit c745757). | 5 |
| Cambio 4 (commit 4cca877). | 5 |
| Cambio 5 (commits 51bf5a5 y 7899350). | 6 |
| Cambio 6 (commit ca1075c). | 6 |
| Cambio 7 (commit c4d194f). | 7 |
| Cambio 8 (commits f3e2d26 y 79dbf42). | 7 |
| Cambio 9 (commit 210b46f y 5501739). | 8 |
| Cambio 10 (commits 03dfe66 hasta b623c4e). | 8 |
| Cambio 11 (commit e0fdbda). | 8 |
| Cambio 12 (commit bc82435 y fc6fccb). | 9 |
| Comentarios personales. | 9 |

Descripción del problema.

En este ejercicio se nos proporciona un proyecto Java el cual es funcional pero contiene ciertos "Code-Smell", por lo que el código que lo compone es muy mejorable.

Se nos pide que lo refactoricemos mejorando dicho código, atendiendo a:

- Refactoring de "Code Smells".
- Principios SOLID.
- Patrones de diseño creacionales.

La función de dicho programa es validar DNI's, NIE's y CIF's, devolviendo como resultado la validez de los mismos.

La salida esperada tras refactorizar ha de ser la misma que antes de la refactorización.

```
=====
Vamos a refactorizar!
=====
DNI 11111111H es: true
DNI 24324356A es: false
NIE X0932707B es: true
NIE Z2691139Z es: false
CIF W9696294I es: true
NIE W9696294A es: false

Process finished with exit code 0
```

URL repositorio Kreitek: <https://github.com/kreitek-com/refactoring>


URL repositorio propio: <https://github.com/nicolasgabarron/refactoring>

¿Cómo he realizado la refactorización?

Antes de nada, he separado el código "malo" del código "bueno". Esta división la he hecho para poder ver el código "malo" u origen en una mitad de la pantalla y en la otra mitad trabajar la copia de dicho código para conseguir su refactorización final.

Esta separación de código la he hecho mediante paquetes:

- "com.kreitek.refactor.mal": Código no refactorizado.
- "com.nicogbdev.refactor.bien": Código refactorizado.



```

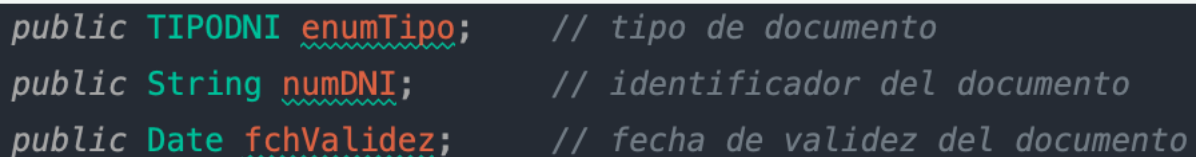
v com
  > kreitek.refactor.mal
  > nicogbdev.refactor.bien

```

Una vez hecha esta copia del código he empezado el trabajo de refactorización.

Cambio 1 (commit bc46906).

El primer cambio que he realizado ha sido en relación a los nombres de las variables. Partíamos de unas variables con nombres poco significantes, algunos acortados y todos en español.

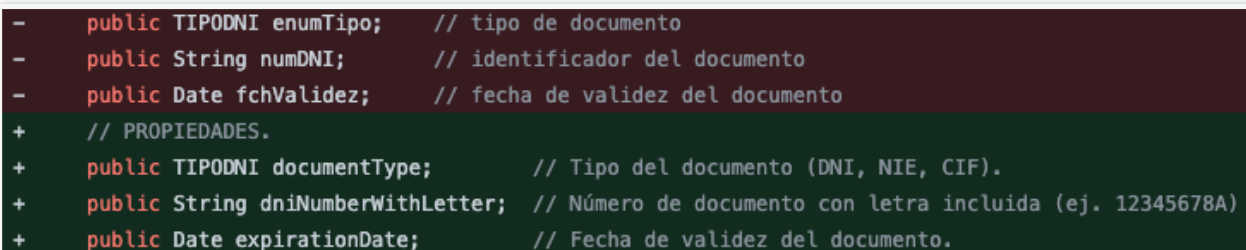


```

public TIPODNI enumTipo;    // tipo de documento
public String numDNI;      // identificador del documento
public Date fchValidez;    // fecha de validez del documento

```

Podrían no estar mal del todo, pero atendiendo a la internacionalización del código y a que éste sea auto-explicable, he traducido los nombres de todas las variables (incluidas las creadas en la lógica del programa) al inglés, haciendo su nombre mucho más descriptivo.



```

- public TIPODNI enumTipo;    // tipo de documento
- public String numDNI;      // identificador del documento
- public Date fchValidez;    // fecha de validez del documento
+
+ // PROPIEDADES.
+ public TIPODNI documentType;    // Tipo del documento (DNI, NIE, CIF).
+ public String dniNumberWithLetter; // Número de documento con letra incluida (ej. 12345678A)
+ public Date expirationDate;    // Fecha de validez del documento.

```

Cambio 2 (commit cfd9009).

El segundo cambio que he realizado ha sido un cambio de nombre a los enumerados con el fin de hacerlos más descriptivos e intuitivos en el sentido de saber a qué hacen referencia.

```
- public enum TIPODNI {  
+ public enum DniType {
```

```
- public enum TipoUltCaracter  
+ public enum CifLastCharType
```

Cambio 3 (commit c745757).

El tercer cambio que he realizado ha sido una pequeña re-estructuración con las clases existentes en paquetes para poder visualizar de una forma mucho más sencilla qué hace cada clase y a qué parte de proyecto hacen referencia. Este cambio es incremental, es decir: con el paso de los commits y el avance de la refactorización se observará cómo va creciendo su importancia y relevancia.

```
src/com/nicogbdev/refactor/bien/{ → enums}/CifLastCharType.java  
  
src/com/nicogbdev/refactor/bien/{ → enums}/DniType.java  
  
src/com/nicogbdev/refactor/bien/{ → models}/DNI.java
```

Cambio 4 (commit 4cca877).

El cuarto cambio ha consistido en empezar a aplicar algo de patrones de diseño y corrección de *Code-Smells*, creando una interfaz "*IdentificationValidate*" la cual hace referencia al principio de Responsabilidad Única (SRP) de la clase DNI.

He creado esta interfaz para crear junto a ella clases de tipo "*Validator*" las cuales implementarán dicha interfaz y contendrán toda la lógica de validación de cada tipo de documento.

```
public interface IdentificationValidate {  
    void validate();  
}
```

```
public class DniValidator implements IdentificationValidate {
```

```
public class NieValidator implements IdentificationValidate {
```

```
public class CifValidator implements IdentificationValidate {
```

Cambio 5 (commits 51bf5a5 y 7899350).

En el quinto cambio he introducido nuevas clases modelo, para CIF y NIF. Además, he creado también una *super-clase* o clase padre llamada "*IdentificationDocument*".

Estas clases harán referencia a documentos de identificación propios, sin tener que pasar por la clase DNI y hacer referencia a un enumerado para saber con qué tipo de documento estamos trabajando, ya que por ejemplo, si estamos creando un CIF, no es correcto que hagamos un "*new DNI...*", ya que no es un DNI, sino un CIF.

Además de esta creación de clases, he encapsulado todos los campos de DNI.

```
public DniType documentType;    // Tipo del documento (DNI, NIE, CIF).
public String dniNumberWithLetter; // Número de documento con letra incluida (ej. 12345678A)
public Date expirationDate;      // Fecha de validez del documento.
private DniType documentType;    // Tipo del documento (DNI, NIE, CIF).
private String dniNumberWithLetter; // Número de documento con letra incluida (ej. 12345678A)
private Date expirationDate;      // Fecha de validez del documento.
```

```
public class IdentificationDocument {
```

```
public class CIF extends IdentificationDocument{
```

```
public class NIF extends IdentificationDocument{
```

Cambio 6 (commit ca1075c).

En el sexto cambio he comenzado la refactorización propia. He sacado toda la lógica de validación de DNI's a su clase validadora "*DniValidator*".

He modularizado todo lo que he podido el código y extraído numerosos métodos que componen las operaciones que el programa ha de seguir para comprobar si un DNI es válido o no.

Además he retirado comentarios innecesarios y añadido comentarios de tipo *JavaDoc* para describir cuál es la función general de cada método, así como los parámetros que recibe y aquellos valores que devuelve.

También he creado una excepción propia para devolver de una manera más correcta los códigos de error.

| | |
|--|---------|
| src/com/nicogbdev/refactor/bien/Main.java | +6 -6 |
| src/com/nicogbdev/refactor/bien/exceptions/InvalidDniException.java | +11 -0 |
| src/com/nicogbdev/refactor/bien/interfaces/IdentificationValidate.java | +3 -1 |
| src/com/nicogbdev/refactor/bien/models/DNI.java | +2 -2 |
| src/com/nicogbdev/refactor/bien/utills/CifValidator.java | +4 -2 |
| src/com/nicogbdev/refactor/bien/utills/DniValidator.java | +132 -1 |
| src/com/nicogbdev/refactor/bien/utills/NieValidator.java | +4 -2 |

Cambio 7 (commit c4d194f).

En el séptimo cambio he introducido la utilización del patrón de diseño *Singleton* con la clase validadora DNI.

Esta decisión ha sido tomada debido a que realmente un programa no debe tener más de una instancia de cada tipo de validado, ya que dicho validador (mediante esa propia instancia) podrá encargarse de validar numerosos documentos de identidad al trabajar con las instancias modelo.

```
private static DniValidator dniValidator;


/**
 * Método que crea una instancia de clase en caso de no existir y la devuelve
 * allá donde haya sido llamado el método.
 *
 * PATRÓN SINGLETON.
 *
 * @return Instancia de clase.
 */
public static DniValidator getInstance(){
    if(dniValidator == null){
        dniValidator = new DniValidator();
    }

    return dniValidator;
}
```

Cambio 8 (commits f3e2d26 y 79dbf42).

En el octavo cambio he terminado de refactorizar todos los tipos de documentos de identificación que nuestro programa puede validar. Para ello he llevado el código de NIF y CIF a sus respectivas clases validadoras.

En este caso, ya he aplicado el patrón *Singleton* de forma directa.



- src/com/nicogbdev/refactor/bien/Main.java
- src/com/nicogbdev/refactor/bien/enums/CIFLastCharType.java
- src/com/nicogbdev/refactor/bien/enums/CifLastCharType.java
- src/com/nicogbdev/refactor/bien/models/CIF.java
- src/com/nicogbdev/refactor/bien/models/DNI.java
- src/com/nicogbdev/refactor/bien/utils/CifValidator.java
- src/com/nicogbdev/refactor/bien/utils/NieValidator.java

Cambio 9 (commit 210b46f y 5501739).

El noveno cambio más relevante que he realizado ha sido refactorizar por completo la clase modelo DNI, la cual contenía antes toda la lógica de negocio. Este era el principal punto de refactorización.

Y en estos commits he eliminado todo el código que no le corresponde para respetar el principio de Responsabilidad Única (SRP) y dejar el modelo de DNI listo para su utilización tal y como es, como modelo.

| | |
|---|---------|
| src/com/nicogbdev/refactor/bien/Main.java | +2 -2 |
| src/com/nicogbdev/refactor/bien/models/DNI.java | +2 -211 |
| src/com/nicogbdev/refactor/bien/models/DNI.java | +20 -1 |
| src/com/nicogbdev/refactor/bien/utils/DniValidator.java | +20 -43 |

Cambio 10 (commits 03dfe66 hasta b623c4e).

En el décimo cambio he introducido numerosas refactorizaciones al código en lo que a legibilidad y mantenibilidad se refiere.

Las clases afectadas han sido los modelos y clases validadoras.

| | |
|--|---------|
| mod: Cambiar ArrayList's por List. | b623c4e |
| add: Constructores por defecto privados para clases Validator. | 8d93bab |
| add: Propiedades al modelo CIF. | 8a72fb2 |
| mod: Constructores modelos. | 03dfe66 |

Cambio 11 (commit e0fdbda).

En el onceavo cambio he introducido un patrón de diseño que facilita la mantenibilidad del código y facilita su legibilidad y escalabilidad.

He introducido el patrón *Factory* debido a que realmente nosotros sabemos que tenemos unos validadores los cuales validan documentos, pero tienen una función común que es validar documentos.

¿Por qué no hacer una *factoría* que nos devuelva esas instancias de validadores que además siguen el patrón *Singleton*?

Esto nos permitiría crear objetos personalizados de cada validador, pero en nuestro caso no es necesario ya que es un ejemplo sencillo.

```
- // Instancias de Validadores.
- DniValidator dniValidator = DniValidator.getInstance();
- CifValidator cifValidator = CifValidator.getInstance();
- NieValidator nieValidator = NieValidator.getInstance();
+ // Instancia de la factoría de Validadores.
+ IDValidateFactory validateFactory = new IDValidateFactory();
+
+ DniValidator dniValidator = (DniValidator) validateFactory.getDniValidator();
+ CifValidator cifValidator = (CifValidator) validateFactory.getCifValidator();
+ NieValidator nieValidator = (NieValidator) validateFactory.getNieValidator();
```



```

public class IDValidateFactory {
    public IdentificationValidate getDniValidator(){
        return DniValidator.getInstance();
    }

    public IdentificationValidate getNieValidator(){
        return NieValidator.getInstance();
    }

    public IdentificationValidate getCifValidator(){
        return CifValidator.getInstance();
    }
}

```

Cambio 12 (commit bc82435 y fc6fccb).

Como último cambio destacable he realizado una refactorización a las propiedades de todos los modelos, creando una jerarquía mucho más correcta.

Esta refactorización ha surgido de un nuevo repaso al principio SRP en todos los modelos y el querer separar aquellas propiedades comunes en la clase padre "*IdentificationDocument*".

Además se ha eliminado el último "*dead-code*" que quedaba en el código.

| | |
|--|--------|
| src/com/nicogbdev/refactor/bien/Main.java | +8 -9 |
| src/com/nicogbdev/refactor/bien/models/CIF.java | +5 -23 |
| src/com/nicogbdev/refactor/bien/models/DNI.java | +4 -29 |
| src/com/nicogbdev/refactor/bien/models/IdentificationDocument.java | +25 -0 |
| src/com/nicogbdev/refactor/bien/models/NIE.java | +5 -24 |
| src/com/nicogbdev/refactor/bien/utills/CifValidator.java | +1 -2 |
| src/com/nicogbdev/refactor/bien/utills/DniValidator.java | +1 -2 |
| src/com/nicogbdev/refactor/bien/utills/NieValidator.java | +1 -2 |

Comentarios personales.

Con la realización de esta práctica he aprendido muchas lecciones de refactorización y código limpio, ya que tras ver los vídeos algunos conceptos no me quedaban del todo claros, pero tras ponerlos en práctica me han quedado mucho más claras.

Seguro que me dejo en "el tintero" numerosos patrones de diseño que se podrían aplicar, o hay maneras de hacerlo mejor, pero con los conocimientos actuales que tengo, es lo máximo que he podido realizar.

Quizás se me queda el capricho de comprender más la lógica de negocio del programa para poder refactorizar mejor cada validador (sobre todo con el CIF), pero no me he querido detener en demasía con ese tema.