

Enginyeria del programari

Pràctica de proves unitàries

Grup: PraLab1

Aarón Arenas Tomás – 78098697N

Marc Cervera Rosell – 47980320C

Pau Escolà Barragán – 48253559L

Data d’entrega: 4 de gener del 2022

**Curs 2021 – 22**

Índex

[Introducció: 1](#_Toc92101524)

[Implementació de mètodes: 1](#_Toc92101525)

[Paquet data 1](#_Toc92101526)

[Excepcions 1](#_Toc92101527)

[Classe NIF 1](#_Toc92101528)

[Classe AccredNumb 1](#_Toc92101529)

[Classe DocPath 1](#_Toc92101530)

[Classe EncryptedData 1](#_Toc92101531)

[Classe EncryptingKey 1](#_Toc92101532)

[Classe PINcode 1](#_Toc92101533)

[Classe Password 1](#_Toc92101534)

[Paquet Services 1](#_Toc92101535)

[Excepcions 1](#_Toc92101536)

[Interfícies 1](#_Toc92101537)

[Classes 1](#_Toc92101538)

[Paquet publicadministration 1](#_Toc92101539)

[Classe QuotePeriodColl 1](#_Toc92101540)

[Classe QuotePeriod 1](#_Toc92101541)

[Classe PDFDocument 1](#_Toc92101542)

[Classe LaboralLifeDoc 1](#_Toc92101543)

[Classe MemberAccreditationDoc 1](#_Toc92101544)

[Classe UnifiedPlatform 1](#_Toc92101545)

[Tests: 1](#_Toc92101546)

[Paquet data 1](#_Toc92101547)

[Classe NIF 1](#_Toc92101548)

[Classe AccredNumbTest 1](#_Toc92101549)

[Classe DocPathTest 1](#_Toc92101550)

[Classe EncryptedDataTest 1](#_Toc92101551)

[Classe EncryptingKeyTest 1](#_Toc92101552)

[Classe PINcodeTest 1](#_Toc92101553)

[Classe PasswordTest 1](#_Toc92101554)

[Paquet Services 1](#_Toc92101555)

[Classe CertificationAuthorityImplTest 1](#_Toc92101556)

[Classe SSImplTest 1](#_Toc92101557)

[Paquet publicadministration 1](#_Toc92101558)

[Classe QuotePeriodCollTest 1](#_Toc92101559)

[Classe QuotePeriodTest 1](#_Toc92101560)

[Classe PDFDocumentTest 1](#_Toc92101561)

[Classe LaboralLifeDocTest 1](#_Toc92101562)

[Classe MemberAccreditationDocTest 1](#_Toc92101563)

[Classe UnifiedPlatformNormalTest 1](#_Toc92101564)

[Classe UnifiedPlatformNullTest 1](#_Toc92101565)

# Introducció:

En aquesta última activitat, es demana realitzar la implementació i les proves pertinents del cas d’ús *Obtener certificación.* En altres paraules, la creació de test i la implementació del codi que passa aquests tests.

Es començarà per formalitzar algunes classes considerades bàsiques, atès que la seva única responsabilitat és la d’emmagatzemar certs valors. Totes elles aniran en un paquet anomenat data.

# Implementació de mètodes:

## Paquet data

### Excepcions

En aquest paquet és crea una classe que conté una excepció anomenada *BatPathException* que serà llençada quan hi hagi un *DocPath* erroni o *null.* Aquesta excepció serà usada en la classe *UnifiedPlatform* en els mètodes que impliquin qualsevol tipus de gestió d’algun document.

### Classe NIF

Aquesta classe representa el NIF d’una persona. El NIF és el sistema d’identificació tributària que s’utilitza a l’estat espanyol per tota persona física o jurídica.

La classe codificada, és quasi la mateixa que es proporciona amb l’enunciat però s’han realitzat algunes modificacions. Una d’aquestes modificacions, ha estat realitzada en el mètode *getter* que retorna el NIF. Originalment (en l’enunciat), aquest mètode és un *getter* clàssic que solament retorna el NIF, però després de la modificació, és possible que es llanci una excepció informant d’un NIF invàlid.

L’última modificació d’aquesta classe, és l’afegit d’un nou mètode *boolean.* Aquest mètode, retornarà cert si el NIF és vàlid i fals en cas contrari. Per comprovar que és vàlid, s’empra la següent estratègia:

1. Es comprova que la instància *nif* no sigui nul·la. En cas de ser-ho és retornarà fals.
2. Atès que tenim un valor per a la instància *nif* (sigui o no correcte), el següent pas és crear un vector de caràcters per introduir el susdit valor de la instància per poder recórrer amb més facilitat el *String*.
3. Es comprova que el valor de *nif* tingui una longitud de 9. En cas de no tenir-la és retornarà fals.
4. Seguidament, com ja es segur que el valor de *nif* té la longitud correcta, s’ha de comprovar el format. Per fer-ho, es comprova que tot el vector de caràcters fins a la penúltima posició (inclosa) són xifres numèriques. En cas de no ser així, es retornarà fals.
5. Finalment, com totes els 8 primers caràcters tenen el format correcte (són xifres numèriques) cal comprovar que l’ultima posició del vector de caràcters sigui una lletra. Si el mètode ha arribat fins aquí significa que el valor *nif* no és null, té una longitud de 9 caràcters i que els 8 primers caràcters son nombres, per tant, si l’últim caràcter és una lletra, el mètode retornarà cert. En cas contrari, fals.

Aquest mètode s’usa per comprovar si cal o no llançar una excepció en el *getter* que retorna el NIF.

### Classe AccredNumb

Aquesta classe representa l’identificador de la SS.

En primer lloc, esmentar que aquesta és una classe totalment implementada per l’alumnat, és a dir, en l’enunciat no s’ha proporcionat res del codi d’aquesta classe.

En segon lloc, cal remarcar que és una classe bastant semblant a la classe *NIF*.

Aquesta classe *AccredNumb*, igual que la classe *NIF*, és una classe final atès que solament interessa el valor de la seva instància.

Després de definir el constructor de la classe (el qual rep el nombre de la SS d’una persona com a paràmetre), és defineix un mètode *getter* que retorna el nombre de la SS en cas de ser vàlid. En cas contrari, llança una excepció indicant que el nombre de la SS introduït no és vàlid.

Per poder saber si s’ha de llançar la excepció o no, és defineix un mètode per comprovar la correctesa del format del valor del nombre de la SS. Per fer-ho, es segueix la estratègia a continuació detallada:

1. Es comprova que la instància *ssNum* no sigui nul·la. En cas de ser-ho és retornarà fals.
2. Un cop es segur que no és nul·la, es crea un vector de caràcters que contindrà el valor de *ssNum*. D’aquesta manera serà més senzill poder recórrer el *String.*
3. Seguidament, si el *String* te una longitud de 12, es comprova tot recorrent el vector de caràcters, que tots i cadascun d’aquests caràcters siguin valors numèrics atès que en la vida real, el nombre de la SS d’una persona només conté xifres numèriques. En cas de trobar alguna xifra no numèrica, es retornarà fals.
4. Finalment, es retornarà cert si i solament si la longitud del *String* és de 12.

Aquest mètode s’usa per comprovar si cal o no llançar una excepció en el *getter* que retorna el *ssNum*.

Darrerament, es creen els mètodes *equals()* (cosa que comporta la creació del mètode *hashCode()*), i el mètode *toString()*.

### Classe DocPath

Aquesta classe, com les dues anteriors és una classe final atès que solament interessa el valor de la seva instància.

Com les dues classes anteriors, aquesta conté els mètodes *equals(), hashCode()* i *toString()* que són mètodes *@Override.*

També com en les classes anteriors, es disposa d’un mètode *getter* que retorna el *path* del document en cas sigui correcte. En cas contrari, es llança una excepció informant de la invalidesa del *path.*

Per comprovar si el *path* és vàlid o no, solament cal comprovar que aquest sigui no nul. Aquesta tasca la duu a terme el mètode booleà *compPathCode()*.

### Classe EncryptedData

De la mateixa manera que en les classes anteriors, aquesta és una classe final que conté el seu constructor, el *equals(),* el *hashCode(),* el *toString(),* el *getter* que retorna les dades encriptades en funció de si són vàlides (cosa que es comprova mitjançant una excepció en el *getter*) i el mètode que determina si les dades encriptades són vàlides o no*.*

Per comprovar si les dades encriptades són vàlides, solament cal comprovar si la instància *data* és *null* o no. En cas de ser *null,* les dades encriptades no seran vàlides.

### Classe EncryptingKey

Aquesta classe segueix la mateixa estructura que les anteriors però adaptada a variable d’instància d’aquesta classe.

En aquesta ocasió per comprovar la validesa de la clau, solament és necessari que no sigui nul·la.

### Classe PINcode

Aquesta classe segueix la mateixa estructura que les anteriors però adaptada a variable d’instància d’aquesta classe.

En aquesta classe, el mètode que comprova la validesa del PIN, segueix l’estratègia a continuació detallada:

1. Si el valor de *pin* és *null*, és retornarà fals.
2. En cas contrari, es crearà un vector de caràcters que contindrà el *pin*.
3. Seguidament, si la longitud del *String* *pin* (variable de la classe), té una longitud de 3, és comprovarà, mitjançant el vector de caràcters, que tots els caràcters siguin dígits. En cas contrari es retornarà fals.
4. Finalment, el mètode retornarà cert, és a dir, el PIN és vàlid si i solament si el *String pin* te una longitud de 3.

### Classe Password

Aquesta classe segueix la mateixa estructura que les anteriors però adaptada a variable d’instància d’aquesta classe.

El que cal remarcar d’aquesta classe, és el mètode que dona per vàlida o no la contrasenya.

Tal i com està implementat el mètode hi ha 2 coses fonamentals perquè la contrasenya sigui acceptada. Ha de tenir una longitud mínima de 7 caràcters i ha de contenir dígits alfanumèrics. Per tant, perquè la contrasenya sigui acceptada, com a mínim haurà de contenir un dígit numèric, un dígit alfabètic i una longitud mínima de 7 caràcters.

## Paquet Services

### Excepcions

### Interfícies

#### Interfície CertificationAuthority

#### Interfície SS

### Classes

#### Classe CertificationAuthorityImpl

#### Classe SSImpl

## Paquet publicadministration

### Classe QuotePeriodColl

### Classe QuotePeriod

### Classe PDFDocument

### Classe LaboralLifeDoc

### Classe MemberAccreditationDoc

### Classe UnifiedPlatform

# 

# Tests:

## Paquet data

### Classe NIF

En la realització d’aquest test en concret es comprova si es realitzen correctament les excepcions de la implementació de la classe NIF. Per a poder realitzar aquestes comprovacions es necessari un @BeforeAll, on s’inicialitzaran 6 objectes diferents. Un amb un NIF correcte, i la resta incorrectes. Amb més números del compte, sense cap caràcter al final, amb caràcters de sobra, amb l’ordre alterat de lletres i números, i per últim un NIF buit.

Una vegada inicialitzats els objectes anteriors, realitzem un @Test per a comprovar que la funció equals que hem implementat funcioni correctament.

També hem realitzat un @Test per a la comprovació de les excepcions. Per a la realització d’aquesta tasca el que fem es cridar a la funció de comparació assertTrue/False que ens comprova si el NIF es correcte (retorna Vertader o Fals depenent del cas).

### Classe AccredNumbTest

En la realització d’aquest test en concret es comprova si es realitzen correctament les excepcions de la implementació de la classe AccredNumbTest. Per a poder realitzar aquestes comprovacions es necessari un @BeforeAll, on s’inicialitzaran 5 objectes diferents. Un amb un Número d’Acreditació correcte, i la resta incorrectes. No havent número d’acreditació, amb més i amb menys números del compte, i sent els caràcters en comptes de números.

Una vegada inicialitzats els objectes anteriors, realitzem un @Test per a comprovar que la funció equals que hem implementat funcioni correctament.

També hem realitzat un @Test per a la comprovació de les excepcions. Per a la realització d’aquesta tasca el que fem es cridar a la funció de comparació assertTrue/False que ens comprova si el Número d’Acreditació es correcte (retorna Vertader o Fals depenent del cas).

### Classe DocPathTest

En la realització d’aquest test en concret es comprova si es realitzen correctament les excepcions de la implementació de la classe DocPathTest. Per a poder realitzar aquestes comprovacions es necessari un @BeforeAll, on s’inicialitzaran 2 objectes diferents. Un amb una ruta d’un document, i una altre inicialitzat a null.

Una vegada inicialitzats els objectes anteriors, realitzem un @Test per a comprovar que la funció equals que hem implementat funcioni correctament.

També hem realitzat un @Test per a la comprovació de les excepcions. Per a la realització d’aquesta tasca el que fem es cridar a la funció de comparació assertTrue/False que ens comprova si el Número d’Acreditació es correcte (retorna Vertader o Fals depenent del cas).

### Classe EncryptedDataTest

En la realització d’aquest test en concret es comprova si es realitzen correctament les excepcions de la implementació de la classe DocPathTest. Per a poder realitzar aquestes comprovacions es necessari un @BeforeAll, on s’inicialitzaran 3 objectes diferents. Dos d’aquests són correctes i l’altre es null.

Una vegada inicialitzats els objectes anteriors, realitzem un @Test per a comprovar que la funció equals que hem implementat funcioni correctament.

Un altre @Test comprova les excepcions. Per a la realització d’aquesta tasca el que fem es cridar a la funció de comparació assertTrue/False que ens comprova si el Número d’Acreditació es correcte (retorna Vertader o Fals depenent del cas).

I per últim tenim un @Test que el que fa es comprovar que la funció toString que hem implementat funcioni correctament.

### Classe EncryptingKeyTest

En la realització d’aquest test en concret es comprova si es realitzen correctament les excepcions de la implementació de la classe EncryptingKeyTest. Per a poder realitzar aquestes comprovacions es necessari un @BeforeAll, on s’inicialitzaran 2 objectes diferents. Un de correcte, sent aquest una clau d’encriptació, i l’altre es incorrecte, sent aquest null.

Una vegada inicialitzats els objectes anteriors, realitzem un @Test per a comprovar que la funció equals que hem implementat funcioni correctament.

Un altre @Test comprova les excepcions. Per a la realització d’aquesta tasca el que fem es cridar a la funció de comparació assertTrue/False que ens comprova si el Número d’Acreditació es correcte (retorna Vertader o Fals depenent del cas).

I per últim tenim un @Test que el que fa es comprovar que la funció toString que hem implementat funcioni correctament.

### Classe PINcodeTest

### Classe PasswordTest \*

## Paquet Services

### Classe CertificationAuthorityImplTest

### Classe SSImplTest

## Paquet publicadministration

### Classe QuotePeriodCollTest

### Classe QuotePeriodTest

### Classe PDFDocumentTest

### Classe LaboralLifeDocTest

### Classe MemberAccreditationDocTest

### Classe UnifiedPlatformNormalTest

### Classe UnifiedPlatformNullTest