Grado en Ingeniería Informática

Principios de Desarrollo de Software

Curso 2018-2019

Grupo 82

Ejercicio Guiado – 4

REFACTORING Y DISEÑO SIMPLE

Rafael Rus Rus – 100363907

Abel Rodríguez Pérez – 100363855

Grupo 1

ÍNDICE

1. INTRODUCCIÓN

El desarrollo de esta práctica va a tener como principal objetivo facilitar el aprendizaje de las técnicas de refactoring y diseño simple siguiendo los pasos facilitados por el propio enunciado. Inicialmente se definen las técnicas de prueba estructural que faltaban y que no se habían realizado en el anterior ejercicio guiado.

2. TÉCNICAS DE PRUEBA ESTRUCTURAL

Para realizar este apartado se van a desarrollar unos procesos divididos en los siguientes pasos:

2.1. GRAFO DE EJECUCIÓN

Debido a las dimensiones del grafo de ejecución, hemos decidido entregarlo en otro documento adjunto con el trabajo denominado "grafo_ejecución.pdf". No obstante, cada uno de los puntos del mismo serán explicados a continuación.

El grafo describe el funcionamiento estructural de la función VerifyLicense, dividida en 4 pasos (steps) tal y como se menciona en los comentarios del código:

- **Step 1:** try to read from the input JSON text file the different fields into a new "License" object

Así pues se comienza con la función Read_License_From_JSON donde se va a realizar el proceso de buscar/leer el fichero. Se comprueba primero si el fichero de entrada existe y si se encuentra en la ruta especificada; si no se encuentra el fichero devuelve un error, pero si lo localizamos se proceden a ejecutar otras dos comprobaciones: la primera para comprobar que es posible acceder al fichero y la segunda para verificar que es posible cerrarlo. Si no da fallo en este último if, se va a crear; si no se consigue crear se advierte de un error, pero si consigue hacerlo se comprueban que todos los campos del JSON son válidos (los campos son el StationName, el PersonInCharge, el EMail, MachineName, TypeOfLicense y License). Realizamos esta comprobación mediante un loop, del que no se sale hasta que compruebe todos los campos pedidos. A continuación, se comprueba que ninguno de estos campos salga repetido en el fichero de entrada.

Si en este último check no da fallo, se inicializa la licencia con los datos del fichero de entrada como corresponde, y se procede a comprobar si el formato de los campos de la nueva licencia inicializada se ajustan a lo pedido en el enunciado: se verifica que el campo PersonInCharge tenga una longitud entre 1 y 20; luego se comprueba que la longitud del campo MachineName esté entre 1 y 10 caracteres; también si el campo EMail tiene un formato de correo electrónico correcto; a continuación, se va a validar si el campo TypeOfLicence es "Fighter", "Starship" o "All"; y por último, se comprueba que el formato del campo License sea válido. Finalmente, se devuelve la licencia inicializada anteriormente.

- **Step 2:** instantiate a new "LicensesStore" including all generated licenses

Simplemente se crea un LicensesStore que será el mismo que habíamos creado en el ejercicio guiado 2 con todas las licencias generadas. Por este motivo, en el grafo se devuelve un LicensesStore.

- **Step 3:** try to find the license obtained from the input file

Consiste en buscar la licencia en el registro LicensesStore que habíamos creado en el ejercicio guiado 2. Para ello, primero se accede a ese LicensesStore con la función load en la que se lee el objeto JSON. Por este motivo, se comprueba que este objeto JSON se lee correctamente (si no da error), y se procede a hacer la transformación en un GSON, a lo cual también haremos otra comprobación. Si no da fallo, es que hemos conseguido acceder al fichero que generamos en el ejercicio guiado 2 con todas las licencias correctas. Luego, se recorren todas las licencias del fichero y se comprueba si el campo Signature (código HASH) coincide con alguno de los que ya hay en ese fichero (en caso de que no coincida, se devuelve null). Tras ello, se van a comprobar todos los otros campos de las licencias: si no coinciden todos se va a devolver un null; pero sí coinciden todos es que hemos dado con la licencia correcta (se ha encontrado en el fichero) y se devuelve ese License con valor lic.

- **Step 4:** check if the license found is valid at this moment

Si la licencia es encontrada en el anterior paso realizado, devuelve un True; si ocurre lo contrario, devuelve un False.

2.2. IDENTIFICAR LA CANTIDAD DE CAMINOS MÍNIMOS

A continuación se van a identificar la cantidad de caminos mínimos (componente de ciclomática) que hay presentes en el anterior grafo de ejecución, y que definirán las pruebas estructurales.

Primero calculamos el valor de la complejidad ciclomática de McCabe para saber cuánto caminos básicos se deben contemplar en los casos de prueba:

V(G) = Enlaces - Nodos + 2 = 52 - 45 + 2 = 9

Las transiciones que se cuentan son las que no están en rojo

Los caminos básicos están contenidos en la carpeta adjunta con el proyecto "caminos" cuyos casos de prueba identificados (entradas necesarias para que el método ejecute cada camino básico identificado) son:

CP-RF3-01-01:

{

"StationName": "aaab",

"PersonInCharge": "Jaimerodriguezrus",

"EMail": "jaime@uc3m.com",

"MachineName": "aabbababe",

"TypeOfLicense": "Fighter",

"License": "5034d2428f7ad441588e0e8610284ca6069b1b0e71d255fb31bcc777009428d1"

}

Salida esperada: TRUE

CP-RF3-01-02:

{

"StationName": "aaab",
"PersonInCharge": "j",
"Email": "jaime@uc3m.com",

}
Salida esperada: ERROR

CP-RF3-01-03:
{
"StationName": "aaab",
"PersonInCharge": "jsddsdsddevefvefvbefnefiedfwefofofwef",
"Email": "jaime@uc3m.com",
"MachineName": "a",
"TypeOfLicense": "Fighter",
"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"
}
Salida esperada: ERROR

CP-RF3-01-04:
{
"StationName": "aaab",
"PersonInCharge": "j",
"EMail": "jaime@uc3m.com",
"MachineName": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
"TypeOfLicense": "Fighter",
"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"
}
Salida esperada: ERROR

CP-RF3-01-05:
{
"StationName": "aaab",
"PersonInCharge": "j",
"EMail": "jaimeuc3uc3m.com",
"MachineName": "a",
"TypeOfLicense": "Fighter",
"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"

}
Salida esperada: ERROR

CP-RF3-01-06:
{
"StationName": "aaab",
"PersonInCharge": "j",
"EMail: "jaime@uc3m.com",
"MachineName": "a",
"TypeOfLicense": "Fighter",
"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c3"
}
Salida esperada: ERROR

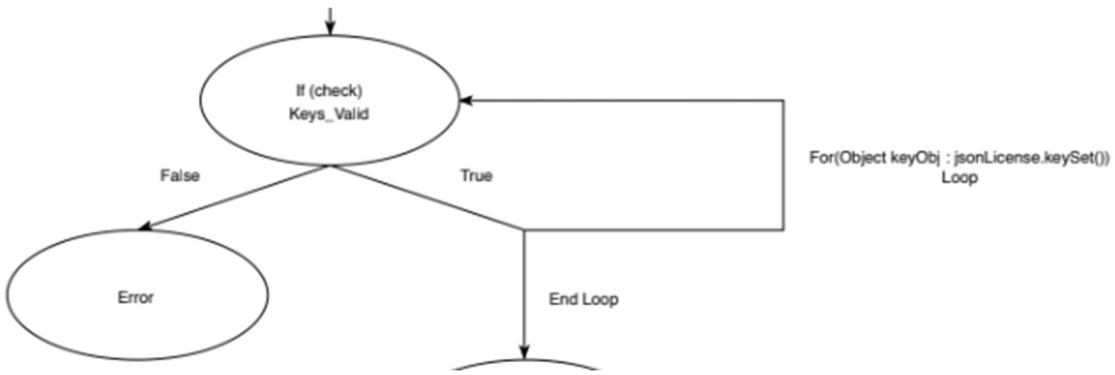
CP-RF3-01-07:
{
"StationName": "aaab",
"PersonInCharge": "j",
"EMail": "jaime@uc3m.com",
"MachineName": "a",
"TypeOfLicense": "aupaLega"
"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"
}
Salida esperada: ERROR

CP-RF3-01-08:
{

}
Salida esperada: ERROR

2.3. CASOS DE PRUEBA UTILIZANDO LA RECOMENDACIÓN PARA LA PRUEBA DE BUCLES

El primer bucle que encontramos en el grafo de ejecución es el siguiente:



CP-RF3-02-01: comprueba que pasa una sola vez por el bucle.

{

"StationName": "aaab",

"PersefonInCharge": "j",

"EMail": "jaime@uc3m.com",

"MachineName": "a",

"TypeOfLicense": "Fighter",

"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"

}

Salida esperada: ERROR

CP-RF3-02-02: comprueba que pasa 2 veces por el bucle.

{

"StationName": "aaab",

"PersonInCharge": "j",

"EMaeil": "jaime@uc3m.com",

"MachineName": "a",

"TypeOfLicense": "Fighter",

"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"

}

Salida esperada: FALSE

CP-RF3-02-03: comprueba que pasa max - 1 veces por el bucle.

{

"StationName": "aaab",

"PersonInCharge": "j",

"EMail": "jaime@uc3m.com",

"MachineName": "a",

"TypeOfLicense": "Fighter",

"Licensdfe": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"

}

Salida esperada: ERROR

CP-RF3-02-04: comprueba que pasa max veces por el bucle.

```
{
        "StationName": "aaab",
        "PersonInCharge": "Jaimerodriguezrus",
        "EMail": "jaime@uc3m.com",
        "MachineName": "aabbababe",
        "TypeOfLicense": "Fighter",
        "License": "5034d2428f7ad441588e0e8610284ca6069b1b0e71d255fb31bcc777009428d1"
}
```

Salida esperada: TRUE

CP-RF3-02-05: comprueba que pasa max + 1 veces por el bucle.

```
{
        "StationName": "aaab",
        "PersonInCharge": "j",
        "EMail": "jaime@uc3m.com",
        "MachineName": "a",
        "TypeOfLicense": "Fighter",
        "License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"
}
```

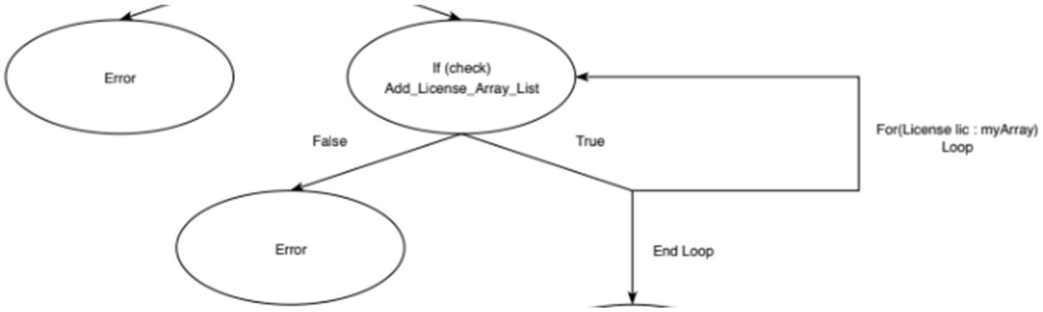
Salida esperada: FALSE

CP-RF3-02-06: comprueba que no pasa por el bucle ninguna vez.

```
{
        "StationNamfvefee": "aaab",
        "PersonInCharge": "j",
        "EMail": "jaime@uc3m.com",
        "MachineName": "a",
        "TypeOfLicense": "Fighter",
        "License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"
}
```

Salida esperada: ERROR

El segundo bucle que encontramos en el grafo de ejecución es otro bucle simple y es el siguiente:



Este segundo bucle va a comprobar que el json del store es el mismo que se le estaba introduciendo.

No es posible demostrar el correcto funcionamiento de este bucle con un json, pero para llegar hasta este bucle en el grafo de ejecución sería necesario partir desde un json del tipo:

CP-RF3-02-07:

{

"StationName": "aaab",

"PersonInCharge": "j",

"EMail": "jaime@uc3m.com",

"MachineName": "a",

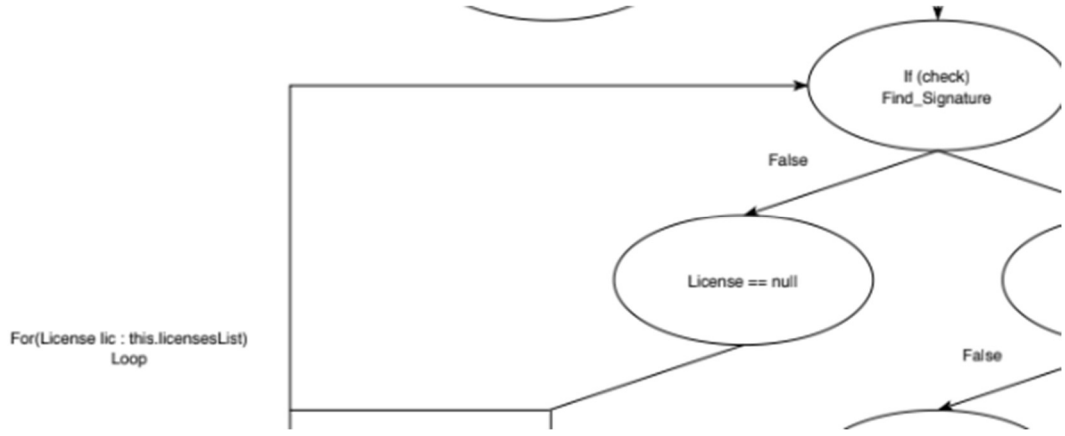
"TypeOfLicense": "Fighter",

"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"

}

La salida dependerá de si el json del store es el mismo que el que se está introduciendo (en tal caso devolverá TRUE) o de si ningún json del store coincide con el json de entrada (en este caso devolverá ERROR).

En el grafo de ejecución encontramos un tercer bucle simple:



Este tercer bucle comprueba que la licencia generada a partir del json de entrada se ha almacenado en el licensesStore y es recorrido N veces (hasta que se encuentra la licencia o hasta que se han leído todas las licencias almacenadas).

La entrada puede ser el mismo json que se ha probado en el anterior bucle:

CP-RF3-02-07:

{

“StationName”: “aaab”,

“PersonInCharge”: “j”,

“EMail”: “jaime@uc3m.com”,

“MachineName”: “a”,

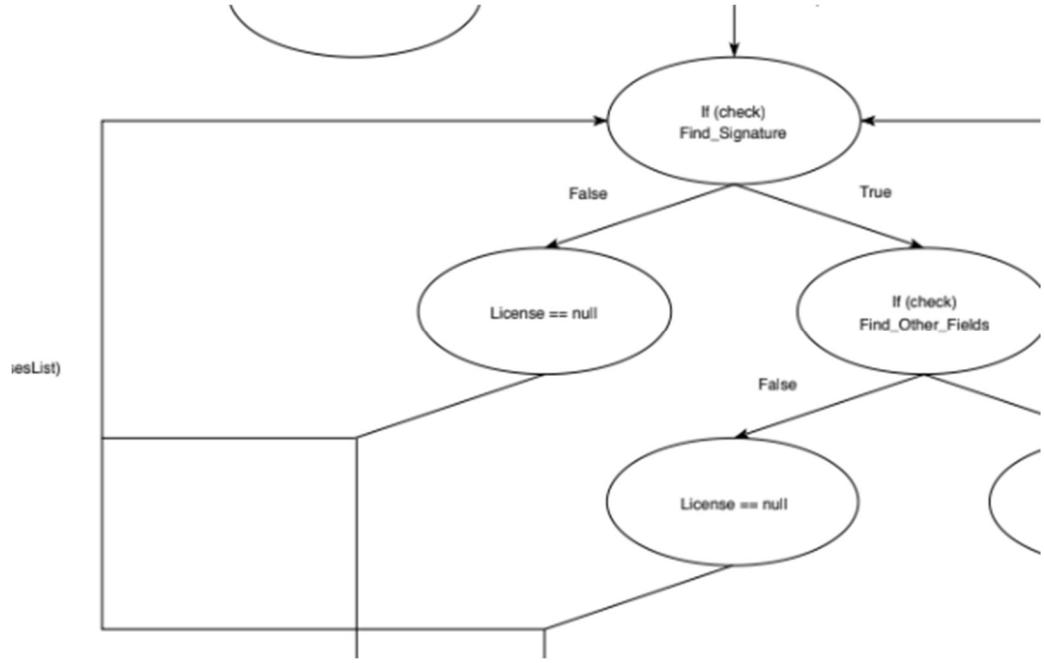
“TypeOfLicense”: “Fighter”,

“License”: “179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c”

}

Si encuentra la licencia en alguna de las iteraciones del bucle devuelve TRUE. Pero si se recorre el bucle 1 vez, 2 veces, max -1 veces, max veces o max +1 veces, y no se encuentra la licencia entonces el bucle devuelve FALSE.

Encontramos un cuarto bucle simple:



En este bucle se entra si los campos de la licencia no son los esperados, y en tal caso se vuelve a buscar la licencia en el licensesStore.

La entrada puede ser el mismo json que se ha probado en el anterior bucle:

CP-RF3-02-07:

{

“StationName”: “aaab”,

“PersonInCharge”: “j”,

“EMail”: “jaime@uc3m.com”,

“MachineName”: “a”,

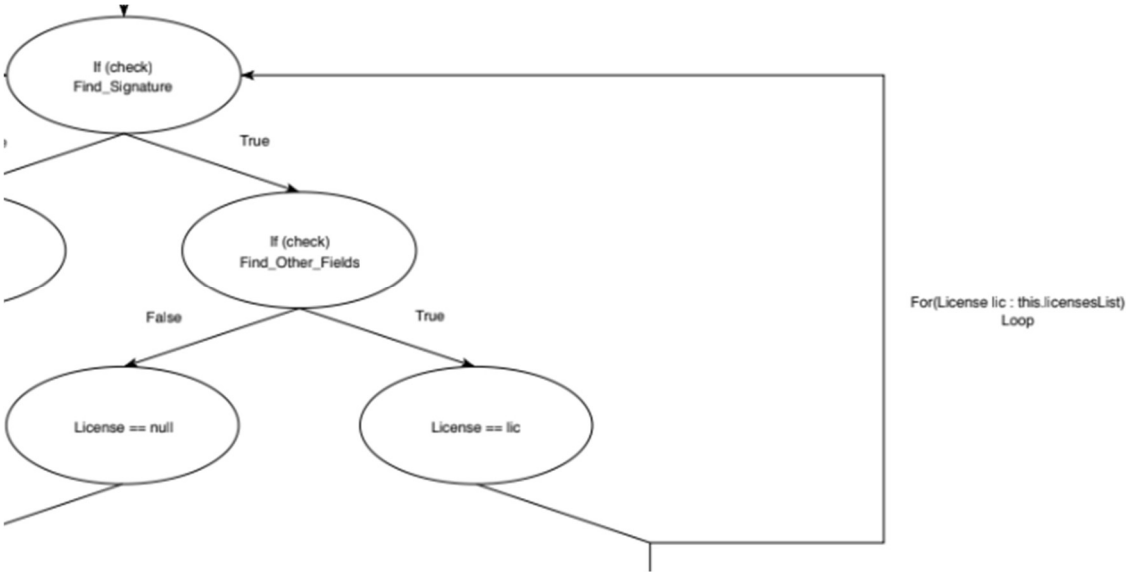
“TypeOfLicense”: “Fighter”,

“License”: “179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c”

}

Si ya se han recorrido todas las licencias devuelve FALSE.

Y por último encontramos otro quinto bucle simple:



Este bucle comprobará que los campos de la licencia son los esperados. Si se han leído todos los campos de la licencia y estos son correctos devuelve TRUE, y si no se han leído todos vuelve a leer los restantes.

La entrada puede ser el mismo json que se ha probado en el anterior bucle:

CP-RF3-02-07:

{
"StationName": "aaab",
"PersonInCharge": "j",
"EMail": "jaime@uc3m.com",
"MachineName": "a",
"TypeOfLicense": "Fighter",
"License": "179cdc7d20d147299dfcc5822fa535ef13b269b40ac9a68d4989adb23ee8342c"
}

3. REFACTORING

Todos los cambios realizados a partir del refactoring vienen descritos en el Excel adjunto con la práctica llamado "refactoring.xlsx". Cabe destacar, con respecto a las pruebas del pom.xml, que estas se han realizado utilizando el jdk 1.8.0_201, aunque este se puede cambiar en función del que tenga cada ordenador.

Para considerar el correcto funcionamiento del refactoring y que el pom.xml compile de forma correcta, se han comentado todos los casos de prueba (los del ejercicio guiado 3 y los de prueba estructural) que daban fallo o error (sólo dan error los 3 casos de prueba que devuelven FALSE dentro de la función VerifyLicense). Así pues, el pom.xml nunca da fallo a la hora de hacer refactoring.