Test Expression

Objecte de la prova
El test d'aquest classe té a veure amb els casos d'ús "Alta expressió booleana" i "Modificació expressions booleanes".

A més a més els últims casos que es provaran en aquest test són d'integració amb les classes que hereden de Expression, que són Literal, Not, Or i And. Aquest petit test d'integració té a veure amb el cas d'ús "Llistar per expressió booleana".

La classe Expression és abstracta i presenta un mètode abstracte evaluate(...) que és justament el que es prova en el test d'integració. Evidentment aquest mètode no es veurà en el test unitari.

D'altra banda Expression implementa un mètode concret estàtic create(...) que retorna una instància de Expression donada una String que representa una expressió booleana. Aquesta instància representa aquesta mateixa expressió booleana però en la forma d'arbre lògic, construït a partir de les subclasse de Expression de manera que després sigui fàcil avaluar una expressió. Si la String no es correspon a una expressió booleana vàlida ha de tirar excepció. Aquest mètode és el que provarem en el test unitari, verificant si el codi llança excepcions quan és necessari.

A continuació en el test d'integració verificarem també si s'han construït correctament les instàncies d'Expression.

Altres elements integrats a la prova
Només a la segona part d'aquest test, integrem les subclasses de Expression en el test. Aquestes classes són Literal, Not, Or i And.

Drivers
No s'usa cap driver en aquest test.

Stubs

En aquest test no farem servir stubs. Les raons són les següents:

- Test unitari: Expression fa servir altres classes, concretament les seves subclasses, però **només** per cridar a les seves constructores. Aquestes constructores funcionen per valor i són trivials, només inicialitzen els atributs de les instàncies amb els paràmetres passats. No és possible, ni necessari, per tant, fer stubs d'aquestes subclasses.

- Test d'integració: com que es d'integració es fan servir les classes reals, tant per Expression com per les seves subclasses. No es requereix de l'ús de cap altra classe.

Fitxers de dades necessaris

No es requereix cap fitxer addicional.

Valors estudiats

Test unitari

Son els que anomenem CREATION TESTS en *TestExpression.java*. Aquests tests busquen provar que el codi retorna excepció quan toca. Es tracten de tests de caixa blanca.

En la documentació de ED&A expliquem quines propietats necessita verificar una String per ser candidata a ser donada d'alta com a expressió booleana al sistema. Aquest test consistirà en verificar que el codi funciona d'acord amb aquestes propietats i llança excepció quan no es compleixen.

Els casos estudiats es corresponen a més d'un test en general, els dividim en subcasos. Són els següents:

- rareCharacters: verifica que caràcters inusuals però vàlids no impedeixen que es construeixi la Expression correctament
- simpleX: verifiquen que es construeixin correctament expressions consistents en un únic operador
- noSpaces: verifica que no fer servir espais per separar operadors d'operands no impedeix que la Expression es construeixi correctament

- manySpaces: verifica que fer servir molts espais per separar operador d'operands no impedeix que la Expression es construeixi correctament
- quotesX: es verifica que es respectin les propietats que exigim a les cometes. Les trobareu a la documentació ED&A, essencialment és que n'hi hagi un nombre parell. A més els operadors entre cometes no es consideren operadors reals, i s'ha de verificar que quan capturem per error un operador entre cometes, l'expressió és invàlida i retorna excepció
- keysX: com que es tracta d'un test de caixa blanca, sabem que en la implementació les claus {} es tradueixen a l'equivalent fórmula amb &. El que es prova en aquest cas és que es retorna excepció quan s'intenten enniuar claus, o quan no tanquen
- parenthesesX: busquen verificar que es llança excepció quan els parèntesis no estan ben enniuats o no tanquen. A més també es prova que no poden faltar operands a cap operador (veure parentheses2)
- operandsAreNotWords: verifica que els operadors lògics &, |, ! no han de ser mai considerats paraules o part d'una paraula. No és correcte per tant trobar dos operadors seguits, i s'hauria de llançar excepció quan això passa.

Test d'integració

Son els que anomenem COMPLEX EVALUATE TESTS en *TestExpression.java*. En aquests tests es prova que efectivament el conjunt de classes funciona correctament per avaluar conjuntament una expressió complexa. Es proven cinc casos relativament complexos.

- exampleInStatement: l'expressió considerada és la que es troba a l'enunciat d'aquest any. En aquest cas provem una mica de tot:
    1. Que no es donin falsos positiu al trobar el que es busca en el contingut però només com a prefix
    2. Que tenir parèntesis innecessàris al voltant d'un Literal no dificulten l'avaluació i funciona correctament
    3. En general que l'avaluació és correcta per la expressió que és i els operadors que té. En particular, es proven totes les maneres de fer que retorni cert.

- prefixesInfixesSuffixesNotConsidered: explora més extensivament que no es donin falsos positius quan es troba dins del contingut a explorar les paraules de l'expressió booleana però només com a prefix, infix o sufix d'una cadena de caràcters més llarga.
- caseSensitive: hi ha un test similar a *TestLiteral.java*. Es verifica que s'ignori o es respecti la diferència entre majúscules i minúscules en funció d'un paràmetre de la funció evaluate(...), per una bateria de tests.
- naturalOrderOfOperandsRespected: en aquest test es verifica que es respecta la jerarquia natural d'operadors lògics, comproven que l'expressió s'avalua com toca.
- parenthesesNotFollowingNaturalOrder: simètricament al d'abans, es verifica que els parèntesis poden trencar la jerarquia natural d'operadors. Això es fa fent servir exactament els mateix exemple que en el test anterior i comprovant que els parèntesis causen que l'expressió avalua diferent.

Efectes estudiats

No aplica en aquest test, que no és de la capa de presentació.

Operativa

Per executar aquest test, cal introduir per consola l'ordre d'execució de java usant les llibreries de junit i hamcrest. Si es realitza des del directori arrel del projecte, és:

java -cp ./EXE/Expression:./lib/junit-4.12.jar:./lib/hamcrest-core-1.3.jar org.junit.runner.JUnitCore test.domain.expressions.TestExpression