JPass Dataflow Coverage
Verificação e Validação de Software 2020–2021

Manuel Tomás 51054
Tiago Varela 51017

2021/04/26

Contents

Initial Mutation Score

As can be seen above, the area of the code with the most surviving mutants is jpass.util. This makes sense, as it is the area of the code with most code outside of the crypt package.
The crypt tests more thoroughly kill the mutants because the algorithms involved deal with exact values, and any deviations will quickly derrail the very precise results of the operations on the bit level. Furthermore, these classes already hahd tests developed by the developers themselves, increasing coverage.
The util package, however, is not as precise, and the tests that exercise them are similarly not as thorough, allowing many mutants to survive.

Name	Line Coverage	Mutation Coverage
jpass.crypt	99% 209/211	95% 211/221
jpass.crypt.io	83% 52/63	76% 22/29
jpass.data	88% 57/65	78% 18/23
jpass.util	79% 159/201	65% 79/122
jpass.xml.bind	47% 14/30	33% 3/9
jpass.xml.converter	100% 11/11	67% 2/3
Total	86% 502/508	67% 335/407

Mutants Investigated

We investigated all of the mutants that were covered by our tests. Mutants who simply weren't covered at all were ignored.
We have written analysis for most types of mutants investigated.

jpass.util.StringUtils

stripString(String text, int length)
Status: KILLED
A new test was added where the length to strip is as large as text.length itself.

stripNonValidXMLCharacters(final String in)
Status: SURVIVED - Equivalent Mutant
Three mutants survive due to a condition that is never exercised. As mentioned in previous reports, it is not possible to test the conditions in particular because it is not possible to store the values in the condition within a char character in Java.

jpass.util.ClipboardUtils

isDataFlavorSupported(DataFlavor flavor)
Status: SURVIVED - Equivalent Mutant
This function is part of a class that is protected. This is a class with the sole purpose of representing Empty String Content from the Clipboard. For this reason the function always returns false. There is a mutant which changes the function to return true. However, the part of the code that uses it doesn't actually need it for anything. It contains both an if and a try, where if true is returned, and exception is thrown and caught, then it simply continues. For this reason, whatever the isDataFlavorSupported function returns is irrelevant, and does not reflect on test results. Finally, we cannot test the isDataFlavorSupported function itself because it's within a protected class.

jpass.util.Configuration

getInstance()
Status: SURVIVED
This is a singleton class, and this is its getInstance function. The mutant consists of changing the condition where it checks if INSTANCE doesn't exist, denying the condition. The result is that, if INSTANCE doesn't exist yet, then it will not create a new instance, thereby always returning null. However, PIT Testing claims this mutant is not detected. However, when we change the condition ourselves, our tests very much detect it, because a new INSTANCE is never created, therefore the condition always evaluates to false, and the getInstance() function always returns null, failing the tests. We conclude that the PIT Testing implementation does not deal well with this static function, meaning the condition is only mutated *after* the INSTANCE is already created, as otherwise it would return null.

jpass.util.CryptUtils

getSha256Hash(final char[] text, final int iteration)
Status: SURVIVED - Equivalent Mutant
This function uses a MessageDigest class with an md.reset() function and this md.reset() is removed by the mutations. However, removing it does not affect the encryption algorithm. This happens because the function md.digest() already calls md.reset() on its own. The developer uses the resets for safety, as it's not guaranteed the md.digest implementation does this on its own. But if it does, then md.reset() is unnecessary, which is what the mutation testing has poined out: Removing md.reset() does not affect the results.

jpass.util.DateUtils

This class is not covered by our tests.

jpass.util.SpringUtilities

This class contained tests to test the exception cases of the makeGrid and MakeCompactGrid functions, but not to test their behaviour was correct. As such, the majority of mutants in this class survive, as whether the function executed corretly or not, the tests would always pass regardless. In order to kill these mutants, tests were added to verify that the girds were correctly created. Previously, tests only checked that no exceptions were thrown.

jpass.data.DataModel

clear()
Status: KILLED
The initial test did not check if the contents of the list were cleared, so a mutant which didn't clear the list survived. The test was fixed so as to check that the list remains clear.

jpass.crypt.Cbc

In general contains various conditions within the algorithms whose boundaries are tested. However, these conditions check for very specific cases, and we were unable to discover which specific cases lead to the conditions activating. For the same reason, we could also not discover the the boundaries for these conditions.

encrypt(byte[] data, int length)
Status: KILLED
The boundaries in the conditions weren't being tested, so tests were developed to test these boundaries.

decrypt(byte[] data, int length)
Status: KILLED
The boundaries in the conditions weren't being tested, so tests were developed to test these boundaries.

Unit tests

shouldStripStringSameLenStringTest()	
What is being done?	Attempts to strip a string passing the same length as argument, which means that if the condition is altered, the result should be different.
What is being tested?	Tests if the returned value is the same as the given one.
Mutant Killed	• change conditional boundary of line 78 • removed call to "setX()" of line 108 • removed call to "setY()" of line 114
Result	Pass
Class	jpass.util.StringUtils

shouldWorkPositiveAreaSizeGreaterThanComponentsTest()	
What is being done?	Attempts call "makeGrid()" with a container whose elements have the same size.
What is being tested?	Tests if the container is formatted with the expected size.
Mutant Killed	• replaced integer multiplication with division of line 76 • negated conditional of line 91
Result	Pass
Class	jpass.util.SpringUtilities

shouldWorkDifferentComponentSizesTest()	
What is being done?	Attempts call "makeGrid()" with a container whose elements have different sizes.
What is being tested?	Tests if the container is formatted with the expected size.
Mutant Killed	• removed call to "setWidth()" of line 94 • removed call to "setHeight()" of line 95
Result	Pass
Class	jpass.util.SpringUtilities

shouldWorkDifferentComponentSizesWithPaddingTest()	
What is being done?	Attempts call "makeGrid()" with a container whose elements have different sizes and with padding applied.
What is being tested?	Tests if the container is formatted with the expected size.
Mutant Killed	• removed call to "setConstraint()" of line 121 • removed call to "setConstraint()" of line 123
Result	Pass
Class	jpass.util.SpringUtilities

shouldWorkPositiveAreaSizeGreaterThanComponentsCompactTest()	
What is being done?	Attempts call "makeGridCompact()" with a container whose elements have different sizes and with padding applied.
What is being tested?	Tests if the container and his elements are formatted with the expected size.
Mutant Killed	• replaced integer multiplication with division of line 161 • negated conditional of line 164 • negated conditional of line 166 • negated conditional of line 169 • removed call to "setWidth()" of line 172 • negated conditional of line 179 • negated conditional of line 181 • negated conditional of line 184 • removed call to "setHeight()" of line 187 • removed call to "setConstraint()" of line 194 • removed call to "setConstraint()" of line 195
Result	Pass
Class	jpass.util.SpringUtilities

Final Mutation Score

Name	Line Coverage	Mutation Coverage
jpass.crypt	99% 209/211	95% 211/221
jpass.crypt.io	83% 52/63	76% 22/29
jpass.data	88% 57/65	83% 19/23
jpass.util	83% 164/197	84% 102/122
jpass.xml.bind	47% 14/30	33% 3/9
jpass.xml.converter	100% 11/11	67% 2/3
Total	88% 507/577	88% 358/407