JPass Category Partition

Verificação e Validação de Software 2020–2021

Manuel Tomás 51054
Tiago Varela 51017

2021/03/08

Contents

Selected Functions

The documentation of all the functions in the project was analysed. We had particular difficulty in identifying a lot of functions with documented boundaries. As a result, we focused on the inputs and their types, as well as paying attention to the function names in order to reach conclusions through some semblance of context.
One function that appeared to contain a number of boundaries was $makeGrid$ from $SpringUtilities$, as it involved a number of inputs who together were limited by the Container. Number of rows and columns, as well as starting indexes result in a number of boundaries.
Another function that was chosen was $stripNonValidXMLCharacters$ from $StringUtils$, which has a series of ranges that condition the behaviour, and so boundary testing was performed on all the ranges.
Finally, the last chosen function was $encrypt$ from $Aes256$, which was chosen for containing multiple input arrays each with their respective indexes, resulting in multiple regions and all the restrictions relating to them.
As such, the chosen functions were:

- java.util.StringUtils class
  - public static String stripNonValidXMLCharacters(String in)
- jpass.cript.Aes256 class
  - public void encrypt(byte[] inBlock, int inIndex, byte[] outBlock, int outIndex)
- java.util.SpringUtilities class
  - public static void makeGrid(Container parent, int rows, int cols, int initialX, int initialY, int xPad, int yPad)

One thing to note is that $Aes256$ already has tests made by the project developers. In fact, we are borrowing from these existing tests the encryption that we are testing with, since the purpose of these tests is not to test the encryption itself but the boundaries of the input space. So as to not mix our own tests with theirs, we have named our testing class $Aes256VVSTest$. This is purely so as to not cause confusion and we are aware that it should be named $Aes256Test$, as we did with the other examples.

stripNonValidXMLCharacters(String in)

Purpose

This method ensures that the output String has only valid XML unicode characters. The values being used for this implementation can be found in the function documentation. [1]

Boundary Value Analysis

Parameters

- in

Characteristics

- in -> String with N number of Y chars (N: number from 0 to +infinity; Y: any char)

As a string, the in parameter may have any number of characters. A reasonable limit for potentially special behaviour would be an empty string. Therefore, two tests will be generated for length 0 and length 0+1. (It is not possible for a string object to have length -1.)

Boundaries

In accordance with the reference for XML char values, the permitted ranges for any char are as follows:

- 0x9 -> Test 0x9, 0x9 + 1, 0x9 - 1
- 0xA -> Test 0xA, 0xA + 1, 0xA - 1
- 0xD -> Test 0xD, 0xD + 1, 0xD - 1
- $$0x20, 0xD7FF$$

  -> Test 0x20, 0x20 - 1 AND Test 0xD7FF, 0xD7FF + 1
- $$0xE000, 0xFFFD$$

  -> Test 0xE000, 0xE000 - 1 AND Test 0xFFFD, 0xFFFD + 1
- $$0x10000, 0x10FFFF$$

  -> Test 0x10000, 0x10000 - 1 AND Test 0x10FFFF, 0x10FFFF + 1

The length of the string is independent from which chars it has, and there is also no specification on the amount of valid vs invalid chars, so we will assume any number of valid or invalid chars is accepted.

Combinations

Generated Tests:

- in.length == 0
- in.length == 1
- in contains 0x9 (equivalent to 0xA - 1)
- in contains 0x9 - 1
- in contains 0xA (equivalent to 0x9 + 1)
- in contains 0xA + 1
- in contains 0xD
- in contains 0xD + 1
- in contains 0xD - 1
- in contains 0x20 (on-point)
- in contains 0x20 - 1 (off-point)
- in contains 0xD7FF (on-point)
- in contains 0xD7FF + 1 (off-point)
- in contains 0xE000 (on-point)
- in contains 0xE000 - 1 (off-point)
- in contains 0xFFFD (on-point)
- in contains 0xFFFD + 1 (off-point)
- in contains 0x10000 (on-point)
- in contains 0x10000 - 1 (off-point)
- in contains 0x10FFFF (on-point)
- in contains 0x10FFFF + 1 (off-point)

For a total of 21 test cases.
Note: For the purposes of the tests, we discovered that "strip" as used by the function name refers to replacing the given char with the char '?'.

Unit tests

shouldStripLengthZeroTest()	
What is being done?	Attempts to strip a empty string which, according to the method documentation, should return an empty string. (Same as should strip empty from Category-Partition)
What is being tested?	Tests if the returned value really is an empty string.
Boundary tested?	in.length == 0
Result	Pass

shouldStripLengthOneTest()	
What is being done?	Attempts to strip a non-empty string (for convenience, without invalid characters).
What is being tested?	Tests if the string is returned.
Boundary tested?	in.length == 1
Result	Pass

shouldStripChar0x9Test()	
What is being done?	Attempts to strip a string with a 0x9 character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0x9
Result	Pass

shouldStripChar0x9MinusOneTest()	
What is being done?	Attempts to strip a string with a 0x9-1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0x9 - 1
Result	Pass

shouldStripChar0xATest()	
What is being done?	Attempts to strip a string with a 0xA character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0xA
Result	Pass

shouldStripChar0xAPlusOneTest()	
What is being done?	Attempts to strip a string with a 0xA+1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0xA + 1
Result	Pass

shouldStripChar0xDTest()	
What is being done?	Attempts to strip a string with a 0xD character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0xD
Result	Pass

shouldStripChar0xDMinusOneTest()	
What is being done?	Attempts to strip a string with a 0xD-1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0xD - 1
Result	Pass

shouldStripChar0xDPlusOneTest()	
What is being done?	Attempts to strip a string with a 0xD+1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0xD + 1
Result	Pass

shouldStripChar0x20Test()	
What is being done?	Attempts to strip a string with a 0x20 character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0x20
Result	Pass

shouldStripChar0x20MinusOneTest()	
What is being done?	Attempts to strip a string with a 0x20-1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0x20 - 1
Result	Pass

shouldStripChar0xD7FFTest()	
What is being done?	Attempts to strip a string with a 0xD7FF character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0xD7FF
Result	Pass

shouldStripChar0xD7FFPlusOneTest()	
What is being done?	Attempts to strip a string with a 0xD7FF+1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0xD7FF + 1
Result	Pass

shouldStripChar0xE000Test()	
What is being done?	Attempts to strip a string with a 0xE000 character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0xE000
Result	Pass

shouldStripChar0xE000MinusOneTest()	
What is being done?	Attempts to strip a string with a 0xE000-1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0xE000 - 1
Result	Pass

shouldStripChar0xFFFDTest()	
What is being done?	Attempts to strip a string with a 0xFFFD character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0xFFFD
Result	Pass

shouldStripChar0xFFFDPlusOneTest()	
What is being done?	Attempts to strip a string with a 0xFFFD+1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0xFFFD + 1
Result	Pass

shouldStripChar0x10000Test()	
What is being done?	Attempts to strip a string with a 0x10000 character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0x10000
Result	Failed - The given char value is stripped because Java reads it as 0xFFFF, the maximum value for a char.(*)

shouldStripChar0x10000MinusOneTest()	
What is being done?	Attempts to strip a string with a 0x10000-1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0x10000 - 1
Result	Pass

shouldStripChar0x10FFFFTest()	
What is being done?	Attempts to strip a string with a 0x10FFFF character, which should return the very same string.
What is being tested?	Tests if the returned string is the same as the input.
Boundary tested?	in contains 0x10FFFF
Result	Failed - The given char value is stripped because Java reads it as 0xFFFF, the maximum value for a char.(*)

shouldStripChar0x10FFFFPlusOneTest()	
What is being done?	Attempts to strip a string with a 0x10FFFF+1 character, which should return the very same string with the character replaced by a '?'.
What is being tested?	Tests if the returned string is the same as the input but with the char replaced by a '?'.
Boundary tested?	in contains 0x10FFFF + 1
Result	Pass

(*) Regarding the failed tests: these are notable because they were detected by SpotBugs during the Static Testing assignment. At the time, we concluded that we could

not do anything to fix the situation as it would either be the case that SpotBugs was incorrect, or that it would not be possible to fix the issue as Java does not accept chars in the given range in normal circumstances.

encrypt(byte[] inBlock, int inIndex, byte[] outBlock, int outIndex)

Purpose

This method encrypts a block of bytes with the Class' given encryption. The block to be encrypted comes from inBlock starting at index inIndex and the encrypted block is stored in outBlock starting at outBlock.

Boundary Value Analysis

Parameters

- inBlock
- inIndex
- outBlock
- outIndex

Characteristics

- inBlock -> N number of Y bytes (N: number from 0 to +infinity. Y: any byte or null)
- inIndex -> Number from -infinity to +infinity
- outBlock -> N number of Y bytes (N: number from 0 to +infinity. Y: any byte or null)
- outIndex -> Number from -infinity to +infinity

Boundaries

For char[], some boundaries that can reasonably be defined are:

- array.length == 0
- array.length == 0 + 1

Length of -1 is impossible and a byte array would not reasonably contain null values because the default byte initialisation in Java is 0. This generates two tests for each array.

For the int values, their restrictions depend on the arrays:

- index == array.length
- index == array.length + 1
- index == array.length - 1
- index == 0
- index == 0 + 1
- index == 0 - 1

For each pair of arrayBlock-index these test cases can be performed. In particular, one can consider that the tests index > 0 and index < array.length are equivalent and test an equivalent partition. Therefore, the total number of test cases generated here are 2*5=10 (index < array.length will be ommited and tested by index > 0).

Potentially, depending on the amount of time and money, tests could be designed to test how the index works. Given index i, does the char with index i get encrypted, or does it encrypt at i+1 or i-1? "Starting of" in the documentation refers to the char at i inclusive or not? Boundary analysis of these cases is possible, but for this project these tests will be deemed not worth the cost and will not be implemented, since the reasonable implementation would be to access the array like inBlock[inIndex].

Finally, the index values have further constraints.The values must be such that copying the entirety of the inBlock region into outBlock is possible, like so:

- outBlock.length - outIndex == (inBlock.length - inIndex)
- outBlock.length - outIndex == (inBlock.length - inIndex) - 1 -> IndexOutOf-BoundsException

These restrictions restrict the outBlock length further than the previous tests which tested them at length 0 and 1, so those are replaced by these cases. They also replace the tests which tested the indexes larger than their respective blocks.

Combinations

The generated tests are the sum of all the ones presented up till now, for a total of 8 test cases.

- inBlock.length == 0 -> IndexOutOfBoundsException
- inBlock.length == 1
- outBlock.length - outIndex == (inBlock.length - inIndex)

- outBlock.length - outIndex == (inBlock.length - inIndex) - 1 -> IndexOutOfBoundsException
- inIndex == 0 (*)
- inIndex == 0 - 1 -> IndexOutOfBoundsException
- outIndex == 0 (*)
- outIndex == 0 - 1 -> IndexOutOfBoundsException

The development of the given tests furthermore reveals some new information: Each "block" as referred to by the documentation expects a length of 16. This changes the tests about the inBlock array lengths. Starting from the given index, 16 bytes are used.

- inBlock.length - inIndex == 16 (*)
- inBlock.length - inIndex == 16 + 1 -> Undefined (Should not interfer with the function's behaviour.)
- inBlock.length - inIndex == 16 - 1 -> IndexOutOfBoundsException

This changes to 9 total test cases.

(*) It's notable that all these test cases use the exact same code, as any valid values may be used for all the other parameters, and the chosen ones were such that all three test cases are contemplated by it. We could consider removing the repeated tests and replacing them by a single one that covers all these partitions.

Unit tests

shouldNotEncryptLength16MinusOneInBlockTest()	
What is being done?	Attempts to encrypt a block of size 15, which is too small according to the implementation. (This isn't documented aside from using the term "block", but since we discovered it, we've adapted the designed tests to contemplate it.)
What is being tested?	Tests if an ArrayIndexOutOfBoundsException is thrown.
Boundary tested?	inBlock.length - inIndex == 16 - 1
Result	Pass

shouldEncryptLength16InBlockTest()	
What is being done?	Attempts to encrypt a block of size 16, the correct size.
What is being tested?	Tests if the returned encrypted outBlock is the correct encryption for the given inBlock.
Boundary tested?	inBlock.length - inIndex == 16
Result	Pass

shouldEncryptLength16PlusOneInBlockTest()	
What is being done?	Attempts to encrypt a block of size 17. The behaviour in this situation is undefined, so the test was made to understand the behaviour. We have discovered that only the first 16 bytes in the array are considered.
What is being tested?	Tests if the returned encrypted outBlock is the correct encryption for the given inBlock.
Boundary tested?	inBlock.length - inIndex == 16 + 1
Result	Pass

shouldNotEncryptOutBlockTooSmallTest()	
What is being done?	Attempts to encrypt a block of size 16 into an outBlock smaller than this, which would result in an Exception because the contents would not fit.
What is being tested?	Tests if an ArrayIndexOutOfBoundsException is thrown.
Boundary tested?	outBlock.length - outIndex == (inBlock.length - inIndex) - 1
Result	Pass

shouldEncryptOutBlockSameSizeAsInBlockTest()	
What is being done?	Attempts to encrypt a block of size 16 into an outBlock of the same size, which should work.
What is being tested?	Tests if the returned encrypted outBlock is the correct encryption for the given inBlock.
Boundary tested?	outBlock.length - outIndex == (inBlock.length - inIndex)
Result	Pass

shouldNotEncryptNegativeInIndexTest()	
What is being done?	Attempts to use a negative inIndex, which should naturally result in an exception.
What is being tested?	Tests if an ArrayIndexOutOfBoundsException is thrown.
Boundary tested?	inIndex == 0 - 1
Result	Pass

shouldEncryptInIndex0Test()	
What is being done?	Attempts to use inIndex 0, which should work.
What is being tested?	Tests if the returned encrypted outBlock is the correct encryption for the given inBlock.
Boundary tested?	inIndex == 0
Result	Pass

shouldNotEncryptNegativeOutIndexTest()	
What is being done?	Attempts to use a negative outIndex, which should naturally result in an exception.
What is being tested?	Tests if an ArrayIndexOutOfBoundsException is thrown.
Boundary tested?	outIndex == 0 - 1
Result	Pass

shouldEncryptOutIndex0Test()	
What is being done?	Attempts to use outIndex 0, which should work.
What is being tested?	Tests if the returned encrypted outBlock is the correct encryption for the given inBlock.
Boundary tested?	outIndex == 0
Result	Pass

makeGrid(Container parent, int rows, int cols, int initialX, int initialY, int xPad, int yPad)

Purpose

This method aligns the first rows * cols components of parent in a grid. Each component is as big as the maximum preferred width and height of the components. The parent is made just big enough to fit them all.

Boundary Value Analysis

Parameters

- parent
- rows
- cols
- initialX
- initialY
- xPad
- yPad

Characteristics

- parent -> Container
- rows -> number from -infinity to +infinity
- cols -> number from -infinity to +infinity
- initialX -> number from -infinity to +infinity
- initialY -> number from -infinity to +infinity
- xPad -> number from -infinity to +infinity
- yPad -> number from -infinity to +infinity

Boundaries

There are some boundaries that can reasonably be defined:

- rows * cols == 0 -> this means that the area cannot be defined as it does not exists (area = 0)
- rows * cols == 0 + 1 -> this means that the area can be defined as it does exists (area = 1)
- nº of elements in the container == rows * cols - 1 -> this means that the area cannot be defined as there are not enough elements to occupy the grid formed
- nº of elements in the container == rows * cols -> this means that the area can be defined as there are enough elements to occupy the grid formed
- container has no elements -> in this case, the area can never be defined as there will be no elements to occupy the grid formed

As for initialX and initialY, which represent the initial point for the grid created, they can have any value, the only thing that will happen is that the position of the grid changes. Although it is not defined by the documentation, when developing the tests we noticed that they can even be negative, in which case, we assumed, it starts off screen, analogously to what happens when applying negative padding to an element in html.
Finally, the xPad and yPad refer to the padding to be added between the cells of the grid, so in does not influence the ability to create the grid.
Moreover, with the development of the test we discovered what type of Exception is thrown in each error case, so below we refer to the correct Exception.

Combinations

The generated tests are the sum of all the ones presented up till now, for a total of 5 test cases.

- container as no elements -> ArrayIndexOutOfBoundsException
- rows * cols == 0 -> NullPointerException
- rows * cols == 1
- nº of elements in the container == rows * cols - 1 -> ArrayIndexOutOfBoundsException
- nº of elements in the container == rows * cols

Unit tests

It should be noted that in the cases that the test should pass, we verify nothing besides that the function did not throw an Exception. We do this because we are not familiar with the way that Spring works, so we do not know what the function's exact effects should be.

shouldNotWorkEmptyContainerTest()	
What is being done?	Attempts to make a grid on an empty Container object which cannot be done as there are no elements to add to the grid.
What is being tested?	Tests if the function really throws an ArrayIndexOutOfBoundsException.
Boundary tested?	container as no elements
Result	Pass

shouldNotWorkZeroAreaSizeTest()	
What is being done?	Attempts to make a grid with size 0 which cannot be done as the grid as to have at least a cell to exist.
What is being tested?	Tests if the function really throws an NullPointerException.
Boundary tested?	rows * cols == 0
Result	Pass

shouldWorkPositiveAreaSizeTest()	
What is being done?	Attempts to make a grid with size 1 which can be done as the grid will have a cell.
What is being tested?	Tests if the function runs without throwing an Exception.
Boundary tested?	rows * cols == 0 + 1
Result	Pass

shouldNotPositiveAreaSizeGreaterThanComponentsTest()	
What is being done?	Attempts to make a grid giving a Container with less elements (in the example 5) than the size of grid intended (int the example rows * cols == 6) which cannot be done has there are not enough elements to put in the grid.
What is being tested?	Tests if the function really throws an ArrayIndexOutOfBoundsException.
Boundary tested?	nº of elements in the container == rows * cols - 1
Result	Pass

shouldWorkPositiveAreaSizeGreaterThanComponentsTest()	
What is being done?	Attempts to make a grid giving a Container with as much elements (in the example 5) as the size of grid intended (int the example rows * cols == 5) which can be done has there are enough elements to put in the grid.
What is being tested?	Tests if the function runs without throwing an Exception.
Boundary tested?	nº of elements in the container == rows * cols
Result	Pass

References

[1] W3C. Extensible markup language (xml) 1.0. [Online]. Available: http://www.w3.org/TR/2000/REC-xml-20001006#NT-Char