Aplicação de Boundary Value Analysis no JPass

Para aplicar Boundary Value Analisys (BVA) precisamos de identificar funções que adequadas para esta estratégia. Procuramos por funções não triviais com pelo menos um número inteiro como parâmetro que pudessem ser sujeitas a testes unitários. Isto reduziu bastante o número de funções que podemos utilizar visto que muitas delas têm apenas strings como parâmetros e outras necessitavam de bastante preparação prévia, o que faria com que os testes passassem a ser de integração ou até quase de sistema. Escolhemos as funções getSha256Hash do Cryptutils e stripString do StringUtils

getSha256Hash

Esta função devolve o hash da string com que é chamada através da aplicação do algoritmo sha256 tantas vezes quantas são pedidas no parâmetro *iteration*. Embora não exista documentação para esta função, achamos razoável que fosse necessário que o valor *iteration* seja maior ou igual a zero. Tendo a nossa condição estabelecida, precisamos de definir o On-Point e o Off-Point, sendo estes, 1 e 0 respetivamente. Fizemos então dois testes, um para o On-Point e outro para o Off-Point, testando assim as duas categorias.

No teste do On-Point, nós criamos um vetor de char com o seguinte conteúdo: {'t', 'e', 's', 't'}. Seguidamente, nós chamamos a função e cifamos o vetor duas vezes. Depois, comparamos o resultado da função com a nossa cifra. Como era esperado, o valor é igual.

No teste do Off-Point, nós procedemos da mesma maneira, mas no final, ao comparar o resultado e a nossa cifra, tentámos verificar se os dois valores eram diferentes, uma vez que este valor não deveria funcionar. No entanto, a função estava escrita de tal maneira que é sempre feita uma iteração independentemente do valor.

stripString

Esta função recebe uma string *text* e um inteiro *length* e devolve uma sub-string de *text* com o comprimento *length* e três pontos no final. Aqui não faz sentido que *length* seja menor que zero, por isso, embora não exista documentação, assumimos que *length* >= 0. Os nossos valores On-Point e Off-Point são, respetivamente, 0 e -1. Foi então necessário um teste para cada um dos valores.

No teste On-Point, chamamos a função com os valores “test string” e 0. Depois comparamos o resultado com “…”. Como esperado, este teste passou.

No teste Off-Point, chamamos a função com os valores “test string” e -1. Isto provoca uma StringIndexOutOfBoundsException, por isso nós apanhamos a exceção e fizemos um assertTrue no catch, para passar o teste, uma vez que este teste é suposto falhar.