SIN5022_testes_software

Atividade 2: Contador de Palavras

Atividade 4 Parte A

Considere o método int countWords(String str)

- Este método recebe uma string como parâmetro e retorna a quantidade de palavras que terminam com a letra “r” ou “s”. Uma palavra precisa ter no mínimo dois caracteres.
- Exemplos: (“Ar puro”,1), (“Fazer valer a pena”, 2), (“As letras r e s são usadas sempre”, 3)

Análise de Classes de Equivalência

Entrada	Classes Válidas	Classes Inválidas
palavra	Strings tamanho > 2 [C1]	tamanho = 0 [C2]
palavra	Strings tamanho > 2 [C1]	null [C3]
palavra	Strings tamanho > 2 [C1]	tamanho = 1 [C4]
frase	Frase valida [C5]	frase com tamanho < 2 [C6]
palavra	ultima letra com r ou s	não existe

Análise de Valor Limite

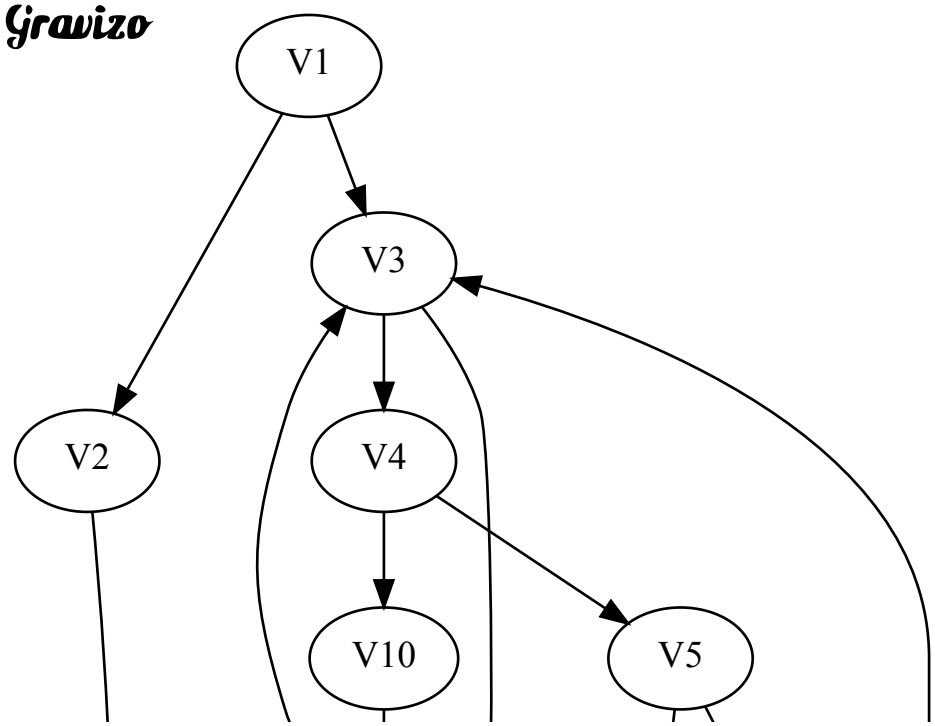
Entrada	Classes Válidas	Classes Inválidas
String	tamanho String >= 2 (V1)	Tamanho = 0 (V2), tamanho = 1 V3
String	tamanho String >= 2 (V1)	Nulo V4
String	tamanho String >= 2 (V1)	" " V5

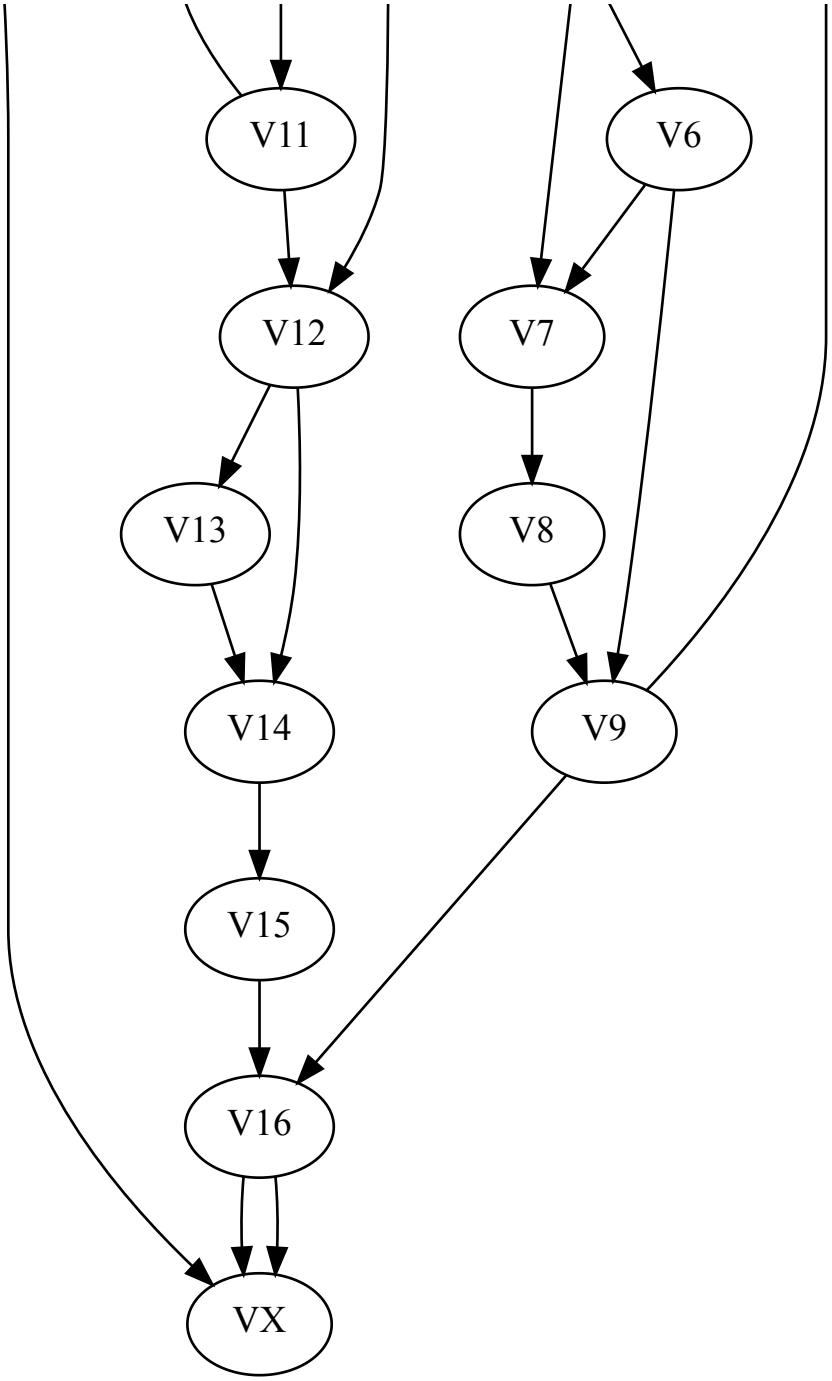
Casos de teste: Valor limite e classes de equivalência

ID	Entrada	Oráculo	Classe
1	nulo	-1	V4
2	""	0	V2
3	"ad"	0	V1
4	"a"	0	V3
5	"as"	1	C1
6	""	0	C2
7	null	0	C3
8	"s"	0	C4
9	"Testes Fortes error"	3	C5
10	"r s"	0	C4
11	" "	0	V5

Exercicio 2 - Parte B e C (Ver considerações no final do documento)

Grafo





Onde VX é a saída do programa.

ID	Entrada	Caminho Grafo
1	nulo	V1 -> V2 -> VX
2	""	V1 -> V3 -> V12 -> V13 -> V16 -> VX
3	"ad"	V1 -> V3 -> V4 -> V10 -> V11 -> V3 -> V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 -> V12 -> V13 -> V16 -> VX
4	"a"	V1 -> V3 -> V4 -> V10 -> V11 -> V3 -> V12 -> V13 -> V16 -> VX

ID	Entrada	Caminho Grafo
5	"as"	V1 -> V3 -> V4 -> V10 -> V11 -> V3 -> V4 -> V10 -> V11 -> V3 -> V12 -> V13 -> V14 -> V15 -> V16, VX
6	""	V1 -> V3 -> V12 -> V13 -> V16 -> VX
7	null	V1 -> V2 -> VX
8	"s"	V1 -> V3 -> V4 -> V10 -> V11 -> V3 -> V12 -> V13 -> V14 -> V16 -> VX
9	"Testes Fortes error"	V1 ->V3 -> V4 -> V10 -> V11 -> V3 -> V3 -> V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V5 -> V7 -> V8 -> V9 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> -> V3 ->V4 -> V10 -> V11 -> -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 -> V4 -> V5 -> V7 -> V8 -> V9 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 ->V4 -> V10 -> V11 -> V3 -> V12 -> V14 -> V15 -> V16 -> VX
10	"r s"	V1 -> V3 -> V4 -> V10 -> V11 -> V3 -> V4 -> V5 -> V6 -> V7 -> V9 -> V3 -> V4 -> V10 -> V11 -> V3 -> V12 -> V13 -> V14 -> V16 -> VX
11	" "	V1 -> V3 -> V4 -> V5 -> V6 -> V9 -> V3 -> V4 V5 -> V6 -> V9 -> V3 -> V12 -> V13 -> V16 -> VX

Considerações:

- Os testes deram 100% line coverage e branch coverage.
- Isso ocorreu na primeira tentativa. Mas há de se considerar que existem testes nos casos de teste que talvez não respeitem a técnica de classes de equivalência ou valor limite. (É uma dúvida, inclusive).
- Nesses testes em questão, eu imaginei que alguém poderia ter escrito de alguma amneira diferente o cídigo e quis testar o ID 10 e 11, por exemplo.
- A única diferença encontrada foi no ID 1, onde na especificação não consta a informação de retorno do numero -1 em caso de nulo. O teste falhou e foi adaptado para esperar -1 ao inves de zero. Essa atividade foi feita durante a etapa A da

atividade. Por isso o branch e line coverage atingiram 100% ainda nos testes funcionais.

- Neste caso então a parte B e C são praticamente a mesma coisa com exceção do ID 1, que poderia ser pego na Etapa B e C, mas foi por coincidência pego na parte A.