GCC255 - REO 4: Exercício de aula

Mateus Carvalho Gonçalves - 201810245
Otávio de Lima Soares - 201811022
Pedro Antônio de Souza - 201810557

1) Considerando as especificações disponibilizadas do programa *series*, entendemos que o domínio das variáveis de entrada é o conjunto dos números reais com as exceções listadas abaixo.

As classes foram definidas pelo domínio das variáveis de entrada e também pela relação entre *start* e *end*, que é uma característica importante entre as entradas.

As especificações do programa não tratam os comportamentos para valores de P negativos. Por isso, em conversas entre o grupo chegou-se à conclusão de que nesses casos deveriam o resultado dependeria de I e F. Para I >= F o resultado teria que ser a contagem descendente, e para I < F seria [] (array vazio).

Condição de entrada	Classes válidas	Classes inválidas
Domínio de *start* (I)	$MIN^1 \leq start \leq MAX^2$	$start < MIN^1$ E $start > MAX^2$
Domínio de *end* (F)	$MIN^1 \leq end \leq MAX^2$	$end < MIN^1$ E $end > MAX^2$
Domínio de *step* (P)	$MIN^1 \leq step < 0$ E $0 < step \leq MAX^2$	$step == 0$
Relação entre *start* e *end* (R)	$start == end$ E $start > end$ E $start < end$	$\|end - start\| > MAX^2$

[1] Menor valor negativo possível para uma variável do tipo *float*
[2] Maior valor positivo possível para uma variável do tipo *float*

O critério AVL é aplicado quando há intervalos, e então gera-se casos de teste para os limites dos intervalos. Nesse caso as I e F têm apenas um intervalo, logo os valores limites são os limites desse único intervalo; P tem dois intervalos, que gera mais um valor limite (0); e como a R não é definido por um intervalo propriamente dito podemos considerar o valor limite como sua classe inválida.

- Limites de I: MIN, MAX => [anterior(MIN), MIN, posterior(MIN), anterior(MAX), MAX, posterior(MAX)]
- Limites de F: MIN, MAX => [anterior(MIN), MIN, posterior(MIN), anterior(MAX), MAX, posterior(MAX)]
- Limites de P: MIN, 0, MAX => [anterior(MIN), MIN, posterior(MIN), anterior(0),0, posterior(0), anterior(MAX), MAX, posterior(MAX)]

Como os valores limites apenas representam os limites dos intervalos das classes de equivalência, existem outros valores (ou combinações de valores) que são importantes de serem testados, por exemplo, *start* e *end* negativos, *start* e *end* positivos, *start* negativo e *end* positivo, o contrário, etc. Abaixo estão listados os casos de teste criados a partir das análises discutidas, considerando anterior(N) e posterior(N) como N - 0.1 e N + 0.1, respectivamente..

Casos de testes:

T = {

t1 (<MIN - 0.1, MIN, 0.1>, I inválido),

t_2 (<MIN, MIN + 0.1, 0.1>, [MIN, MIN + 0.1]),

t_3 (<MIN + 0.1, MIN, 0.1>, [MIN + 0.1, MIN]),

t_4 (<MAX - 0.1, MAX, 0.1>, [MAX - 0.1, MAX]),

t_5 (<MAX, MAX - 0.1, 0.1>, [MAX, MAX - 0.1]),

t_6 (<MAX + 0.1, MAX, 1>, I inválido),

t_7 (<MIN, MIN - 0.1, 1>, F inválido),

(<MIN + 0.1, MIN, 0.1>, [MIN + 0.1, MIN]),

(<MIN, MIN + 0.1, 0.1>, [MIN, MIN + 0.1]),

(<MAX, MAX - 0.1, 0.1>, [MAX, MAX - 0.1]),

(<MAX - 0.1, MAX, 0.1>, [MAX - 0.1, MAX]),

t_8 (<MAX, MAX + 0.1, 0.1>, F inválido)

t_9 (<0, 1, MIN - 0.1>, P inválido),

t_{10} (<0, 1, MIN>, []),

t_{11} (<0, 1, MIN + 1>, []),

t_{12} (<0, 1, -0.1>, []),

t_{13} (<0, 1, 0>, P inválido),

t_{14} (<0, 0.1, 0.1>, [0, 0.1]),

t_{15} (<0, 1, MAX - 1>, [0]),

t_{16} (<0, 1, MAX>, [0]),

t_{17} (<0, 1, MAX + 1>, P inválido),

t_{18} (<1, -1, -1>, [1, 0, -1]),

t_{19} (<0, 0, 1>, [0]),

t_{20} (<MIN, MAX, 5>, R inválido)

}

Casos de teste marcados com a mesma cor representam casos iguais que satisfazem a testes para classes de valores limites diferentes. Eles foram duplicados para que a visualização dos testes para cada critério seja mais legível.

2) A partir das análises do item anterior, foram gerados 20 casos de teste. Todos os casos foram desenvolvidos para serem testados por meio da ferramenta JUnit. Dos 20 casos de teste, 12 executaram com sucesso e 8 falharam, mostrando uma taxa de sucesso de 60%.

A lista abaixo mostra todos os casos de teste com falha, suas saídas esperadas e o que o que foi apontado pelo JUnit:

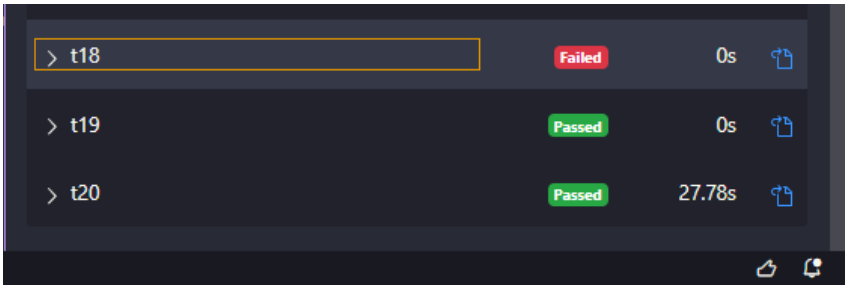
- t_3:
  - Saída esperada: Array[2];
  - Saída do JUnit: [];
- t_5:
  - Saída esperada: Array[2];
  - Saída do JUnit: Erro de memória;
- t_7:
  - Saída esperada: Lançamento de exceção;
  - Saída do JUnit: Não lançou exceção;
- t_9:
  - Saída esperada: Lançamento de exceção;
  - Saída do JUnit: Não lançou exceção;
- t_{13}:
  - Saída esperada: Lançamento de exceção;
  - Saída do JUnit: Erro de memória;
- t_{14}:
  - Saída esperada: Array[2];
  - Saída do JUnit: Array[1];
- t_{17}:
  - Saída esperada: Lançamento de exceção;
  - Saída do JUnit: Não lançou exceção;
- t_{18}:
  - Saída esperada:Array[3];
  - Saída do JUnit: Array[2].

Os problemas nos casos de teste t_7, t_9 e t_{17} podem estar relacionados ao Java tratar operações com os maiores e menores valores (do tipo *float*) possíveis a serem representados, ou seja, a linguagem faz algum tratamento e não lança exceções quando o maior valor é incrementado, ou o menor valor é subtraído.

Para t_{13}, a especificação do programa *series* aponta apenas que “the stepsize must be nonzero” (a variável *step* deve ser diferente de 0), mas observando o código do programa, vê-se que há tratamentos para alterar o valor dessa variável para 1. O tratamento feito parece ser confundido com outra especificação do programa: “if it (a variável *step*) is not specified, it is assumed to be of unit size (1)” (se a variável não for especificada, ela deve assumir o valor 1).

Nas figuras abaixo é possível observar a lista de resultados das execuções do JUnit, feitas na IDE VSCode.


Java Test Report
t1 Passed 29.89s
t2 Passed 0s
t3 Failed 0.01s
t4 Passed 0s
t5 Failed 24.3s
t6 Passed 25.93s
t7 Failed 0s
t8 Passed 27.42s
t9 Failed 0s
t10 Passed 0s
t11 Passed 0s
t12 Passed 0s
t13 Failed 25.95s
t14 Passed 0s
t15 Passed 0s
t16 Passed 0s
t17 Failed 0s


t18 Failed 0s
t19 Passed 0s
t20 Passed 27.78s