Aimee Valladares

CEN4072

27 February 2022

# **Assignment #3**

Q1) (30 Points) Derive input space partitioning test inputs for the GenericStack class with the following method signatures:

* public void push (Object X);
* public Object pop();
* public boolean isEmpty();

Assume the usual semantics for the GenericStack. Try to keep your partitioning simple and choose a small number of partitions and blocks. (For each method list inputs, characteristics, blocks, and sample values to test)

* 2 Parameters: Stack, Element
* Interface-based Characteristics
  + Characteristics for the stack:
    - Stack is empty
      * Block1 = true
        + Sample Values: []
      * Block2 = false
        + Sample Values: [“one”, “two”, “three”]
    - Stack is null
      * Block1 = true
        + Sample Values: [null], [null, “two”]
      * Block2 = false
        + Sample Values: [“bun”], [“one”, “two”, “three”]
    - Stack size
      * Block1 = 0
        + Sample Values: []
      * Block2 = 1
        + Sample Values: [“one”], [null]
      * Block3 = Greater than 1
        + Sample Values: [“one”, “two”, “three”]
  + Characteristics for Object X
    - X is null
      * Block1 = true
        + Sample Values: x = null
      * Block2 = false
        + Sample Values: x = “dog”
    - X is empty
      * Block1 = true
        + Sample Values: x = “”
      * Block2 = false
        + Sample Value: x = “one”
  + Characteristics that is a combination of Object X and the stack state
    - Object X is in the stack
      * Block1 = true
        + Sample Values: (null, [“one”, null])
      * Block2 = false
        + Sample Values: (“one”, [“two”, “three”])
* Functionality-based Characteristics:
  + Stack create (Max.Size)
  + element push into stack
    - (Element) Returns value
  + element pop from stack
    - (Element) returns value
  + element IsEmpty px stack
    - (True, False)

Q2) (25 points) Write down all 16 tests to satisfy the Multiple Base Choice Coverage (MBCC) for the second categorization of triang()’s inputs in the following table: (‘2’ and ‘1’ are base choices for side 1, other sides still have 1 base choice of ‘2’). Utilize every possible coverage criterion.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Param | b1 | b2 | b3 | b4 |
| Side 1 | 2 | 1 | 0 | -1 |
| Side 2 | 2 | 1 | 0 | -1 |
| Side 3 | 2 | 1 | 0 | -1 |

**Multiple Base Choice Coverage (MBCC)**

Base Choice(s): (2, 2, 2)

(2, 1, 2) (2, 0, 2) (2, -1, 2)

(2, 2, 1) (2, 2, 0) (2, 2, -1)

Base Choice(s): (1, 2, 2)

(1, 2, 1) (1, 2, 0) (1, 2, -1)

(1, 1, 2) (1, 0, 2) (1, -1, 0)

(0, 2, 2) (-1, 2, 2)

Base Choice Coverage (BCC)

Base Choice: (2, 2, 2)

(1, 2, 2) (2, 1, 2) (2, 2, 1)

(0, 2, 2) (2, 0, 2) (2, 2, 0)

(-1, 2, 2) (2, -1, 2) (2, 2, -1)

All Combinations (ACoC)

(2, 2, 2) (1, 2, 2) (2, 1, 2) (2, 2, 1) (0, 2, 2) (2, 0, 2)

(1, 1, 1) (0, 2, 2) (2, 0, 2) (2, 2, 0) (0, 0, 1) (0, 1, 0)

(0, 0, 0) (-1, 2, 2) (2, -1, 2) (2, 2, -1) (2, 1, 0) (2, -1, 0)

(-1, -1, -1) (1, 2, 1) (0, 1, -1) (1, 0, 2) (2, -1, -1) (0, 0, 2)

….. There are a total of 64 tests!

Each Choice Coverage (ECC)

(2, 2, 2) (1, 1, 1) (0, 0, 0) (-1, -1, -1)

Pair-Wise Coverage (PWC)

**(**-1, -1) (-1, 0) (-1, 1) (-1, 2)

(0, -1) (0, 0) (0, 1) (0, 2)

(1, -1) (1, 0) (1, 1) (1, 2)

(2, -1) (2, 0) (2, 1) (2, 2)

(-1, 0, 1) (0, -1, 2) (-1, 1, -1) (0, 1, 0)

(0, 1, 1) ` (0, 2, 2) (1, 2, -1) (2, 1, 0)

Q3) (30 Points) Answer the following questions for the method intersection() below:

public Set intersection (Set s1, Set s2)

// Effects: If s1 or s2 is null throw NullPointerException

// else return a (non null) Set equal to the intersection

// of Sets s1 and s2

Characteristic: Validity of s1

- s1 = null

- s1 = {}

- s1 has at least one element

Characteristic: Relation between s1 and s2

- s1 and s2 represent the same set

- s1 is a subset of s2

- s2 is a subset of s1

- s1 and s2 do not have any elements in common

1. Does the partition “Validity of s1” satisfy the completeness property? If not, give a value for s1 that does not fit in any block.

The partition “Validity of s1” does satisfy the completeness property because s1 can be null, empty, or have at least one element. Any value placed in s1 can fit into any of characteristics and there is not any value that would not satisfy or fit into those conditions.

1. Does the partition “Validity of s1” satisfy the disjointness property? If not, give a value for s1 that does not fit in any block.

The partition “Validity of s1” does satisfy the disjointness property because there cannot be any overlap between the characteristics of s1. Specifically, no given value can qualify for more than one condition as it can only be either null, empty, or have at least one element.

1. Does the partition “Relation between s1 and s2” satisfy the completeness property? If not, give a pair of values for s1 and s2 that does not fit in any block.

The partition “Relation between s1 and s2” does not satisfy the completeness property because the final characteristic “s1 and s2 do not have any elements in common” could be violated. For instance, s1 = {1, 2, 3} and s2 = {3, 4, 5} would violate that characteristic

1. Does the partition “Relation between s1 and s2” satisfy the disjointness property? If not, give a pair of values for s1 and s2 that does not fit in any block.

The partition “Relation between s1 and s2” does not satisfy the disjointness property because the two characteristics “s1 and s2 represent the same set” and “s1 and s2 do not have any elements in common” can be violated. For instance, if s1 and s2 are both empty sets would violate those characteristics.

1. If the “Base Choice” criterion were applied to the two partitions (exactly as written), how many test requirements would result?

There would be 6 test requirements

Q4) (15 Points) Based on Q2, add two test cases to the file <https://cs.gmu.edu/~offutt/softwaretest/java/TriangleTypeTest.java> under the collection method. Just show the lines that you have added not the full file.

list.add(new Object[] {1, 1, 1, Triangle.EQUILATERAL}); - Based on the table seen in Q2, block 1, 2, and 3 (b1, b2, b3) are 1. Since all the sides are equal size and valid as an equilateral triangle.

list.add(new Object[] {1, 1, 0, Triangle.INVALID}); - Based on the table seen in Q2, block 1 (b1) = 1, block 2 (b2) = 1, and block 3 (b3) = 0. Since the value for block 3 is not valid for a triangle.