Assignment-01

Prepared by

Sohanur Rahman

Id: 1510464642

Email: rahman.sohanur@northsouth.edu

Semester: Spring 2019

CSE427 Section 1

Prepared for

Shaikh Shawon Arefin Shimon

Lecturer

Department of Electrical & Computer Engineering

North South University

Github link: https://github.com/Sohanur-Rahman642/North-South-University

Answer of the questions

(a) List of the input variables:

Since the GenericStack() class accepts generics it actually accepts a wide range of input values.. So the list of the type of input variables are:

1- Integers (positive & negative),

2- Floats (positive & negative),

3 - Doubles (positive & negative),

5- Characters

4- Strings

5- Generics or Templates

The first step in input domain modeling is to identify testable functions or methods. So the constructor & methods are used in the source code of the GenericStack class are:

1. GenerickStack();

2. Push(Object x);

3. Pop();

4. isEmpty();

5. getSize();

6. isFull();

7. peek();

8. sizeAfterPush();

The list of all input variables depends on the methods

(b) To define the characteristic we have to choose between two different approaches of input domain modeling. One is the interface-based approach and the other one is the functions-based approach.The interface-based approach considers each parameter separately. This approach is almost mechanical to follow, but the resulting tests are usually quite good.

Designing characteristics in an interface-based approach is simple. There is a mechanical translation from the parameters to characteristics. So if we consider the GenericStack class there

is a requirement of an Object type variable. An Object type variable can be anything like a integer, a double or even a String. So the input array could be a collection of all possible input variables.

Characteristic of the Stack:

Characteristic	b1	b2
Stack is null	true	false
Stack is full	true	false
Stack is full	true	false
Stack is not full	false	true

Characteristic	b1	b2	b3
Number of occurrences	0	1	More than 1
Elements occurs first	True	False	
Element occurs last	True	False	

(C) Characteristic of inputs:

Valid and invalid values: Every partition must allow all values, whether valid or invalid.

Sub-partition: A range of valid values can often be partitioned into sub-partitions, such that each sub-partition exercises a somewhat different part of the functionality.

Boundaries: Values at or close to boundaries often cause problems. This is a form of stress testing.

Normal use (happy path) : : If the operational profile focuses heavily on “normal use,” the failure rate depends on values that are not boundary conditions.

In this case, I will partition and test the Normal use ones (happy path)

(d) Partition of characteristic into blocks:

First partition of push method . The parameter of this method is an object type variable.

Characteristic	Valid of char	Valid of String	Negative	Zero	Positive
Integer			{-7,-10,-43,-577,-1754}	0	{10,32,67,159,991,1677}

Float			{-12.4,-43.2,-17.5}	0	{1.2,3.4,14.20}
Double			{-42.4,-73.12,-122.57}	0	{19.25,3.4,244.23}
Character	{a,c,h,k,x,z,y}				
String	{“Lamborghini”,”Ferrari”,”Aston Martin”,”Roles Royace”}				

(e) Defining values from each block:

I choose 1 value from each partition of the blocks(3 values from each lock) to test after implementing Test driven development on GenericStack class.

From Integer block: -1754, 0,991

From Float block: -43.2,0,3.4

From Double block: -122.57,0,244.23

From Character block: a,x,z

From String block: “Lamborghini”,”Ferrari”,”Roles Royace”