North South University

CSE427

Assignment-1

Course Instructor:
Shaikh Shawon Arefin Shimon
Lecturer
Department of Electrical and Computer Engineering,
North South University

Prepared by:
Sadia Islam Nisa, ID: 1030545042

List all of the input variables, including the state variables.

So here in my code, list of the input variable is given in the main function ob.push() method 5, 10, 15 and in each case the file output result output reverse that is 15,10,5.
My state variables object array[ ] and size is equal to 0.

```
protected Stack s;
@Test
public void testNewStackIsEmpty() {
assertTrue(s.isEmpty());
assertEquals(s.size(), 0);
}
@Test
public void testPushesToEmptyStack() {
int numberOfPushes = 6;
for (int i = 0; i < numberOfPushes; i++) {
s.push("zzz");
}
assertFalse(s.isEmpty());
assertEquals(s.size(), numberOfPushes);
}
```

Define characteristics of the input variables. . Make sure you cover
all input variables.

Characteristics of the input variable might be any type of object item like int, float , double or char type of. So in that case code can be implemented using
stack push method which is considered to be the right way for doing calculation.

```
@Test
public void testPushThenPop() {
String message = "hello";
```

```
s.push(message);
assertEquals(s.pop(), message);
}
```

Define characteristics of inputs.

The input domain of a program consists of all possible inputs that could be taken by the program. Ideally, the test selection problem is to select a subset
T of the input domain such that the execution of T will reveal all errors. In practice, the test selection problem is to select a subset of T within budget such that it reveals as many errors as possible.

```
@Test
public void testPushThenPeek() {
String message = "hello";
s.push(message);
int size = s.size();
assertEquals(s.peek(), message);
assertEquals(s.size(), size);
}
```

Partition the characteristics into blocks.

Partition the input domain into a relatively small number of groups, and then
select one representative from each group. A partition defines a set of equivalent classes, or blocks. A partition must satisfy two properties: completeness and disjoint. A partition is usually based on certain characteristic.

```
@Test
public void testPoppingDownToEmpty() {
int numberOfPushes = (int)(Math.random() * 20
+ 1);
for (int i = 0; i < numberOfPushes; i++) {
```

```
s.push("zzz");
}
for (int i = 0; i < numberOfPushes; i++) {
s.pop();
}
assertTrue(s.isEmpty());
assertEquals(s.size(), 0);
}
```

Define values for each block

A stack is a container of elements with last in, first out access policy.
Sometimes it also called LIFO.
The stack is accessed through its top.
The basic stack operations are:

- push stores a new element onto the stack top;
- pop returns the last pushed stack element, while removing it from the stack;
- empty tests if the stack contains no elements.

```
@Test(expected=NoSuchElementException.class)
public void testPopOnEmptyStack() {
assertTrue(s.isEmpty());
s.pop();
}
@Test(expected=NoSuchElementException.class)
public void testPeekIntoEmptyStack() {
assertTrue(s.isEmpty());
s.peek();
}
```

Repository Link:

https://github.com/SadiaIslamNisa/Assignment-1