INFO 6067

Testing for Development Lab Week 5

## Description – Code Coverage

Marks - 20

Instructions

Create a new project in Eclipse. Add the JUnit libraries

Import the files.

* Advice.java
* BooBoo.java
* TestAdvice.java

Run the TestCase

Run the code coverage

1. What is the starting coverage for the Advice class? Take a screenshot (1 Mark)?

|  |
| --- |
|  |

1. Provide me with the set of acceptable parameters for the methods you are testing. This helps differentiate the negative and the positive test cases.(2 Marks)

the method we are testing has the defination as `public boolean getAdvice( int i, char c) throws BooBoo` . Which means this method accept all interger values from `Integer.MIN\_VALUE` to `Integer\_MAX\_VALUE` and the second parameter accepts all character value from `Character.MIN\_VALUE` to `Character\_MAX\_VALUE`.

From the first line of getAdvice methods implementation we see if i == 1 then it throws an exception. Which means the input

( i = 1, c = “any character” ) it will be an negative test.

From the second if statement i.e. `else if` we see if i equals 1 or c equals ‘z’ then the method throws an exception, which means other negative tests would be like

( i = “any integer”, c = ‘z’ )

we cal also do some edge case tests like

( i = 0, c = ‘z’ )

( i = Integer.MAX\_VALUE, c = “any character” )

( i = Integer.MIN\_VALUE, c = “any character” )

we can also add some tests for i = negative numbers like

( i = -1, c = “any character” )

( i = -2, c = “any character” ) etc

for some positive tests we can use any inputs like `i` is not equal `1` and `c` is not equal `z`.

1. Think of the types of tests you can perform to increase your coverage. Using the @Test annotation, do 3 positive and 3 negative tests. There should be at least one test for exceptions. (4 Marks)

As the provided test has only one test method so it covers a small amout of the Advice class. As the getAdvice method which we are testing uses random numbers to generate it’s output so we need to do some more test to cover as much as we can. We know that is use a lot of test then the possibility of havin the most coverage increases. But we won’t try to do that yet.

For the positive test we can use some very simple inputs

one simple condition we need to maintain is not to provide the number 1 as the first input of the method and not to provide the character ‘z’ as the second input of the method.

The inputs we are using to do the positive tests as follows

i = 2, c = a; i = 3, c = b; i = 5, c = b, i = 6, c = x;

for the negative test we can use either i = 1 or c = ‘z’ any one of this will be an negative test. We used as follows

i = 1, c = a; i = 1, c = b; i = 0; c = z; i = 1, c = z;

for the edge case test we need to try to provide the inputs as the supported maximum and minimum for the input types. We used as follows:

i = -2147483638, c = a; i = 2147483637, c = b; i = 0, c = ?; i = 2, c = space;

for the exception test which is similar to the negative test we try to get an exception of the BooBoo class on each of the test. The input we used as follows:

i = 1, c = a; i = 1, c = z; i = 25, c = z;

1. What is the challenge with the getAdvice() method (2 Marks)?

The main challange with the getAdvice method is it uses random number to determine the output. As it uses random number value form the array so we can’t be sure when and how each case will be matched.

It is possible that only 20 test case in a single run it matches all the cases in the switch statement ( which not likely to happen )

It is also possible with a lot of tests it will not cover all the cases in the switch statement.

Another thing is the value used in the switch block it’s always filxed between 1 and 9 inclusive. So the default case will never be covered.

|  |
| --- |
|  |

1. What is the coverage percentage now (Take a screenshot) (1 Mark)?

|  |
| --- |
|  |

1. Create a new java file, and optimize the code coverage by having as many positive test cases, as many negative test cases, and exception tests as you can on the getAdvice() with as few test functions as possible. I am expecting a minimum of 5 positive and negative tests and 1 exception test. Copy and paste a snippet of the optimized test cases in this document (8 Marks).

To increase the test coverage we need to use as many test as we can so that the possibility of the random value used in the get advice methods covering all it’s branches in the test’s maximized. Which we will try to do using just by writing as many test cases as we can.

1. You should now have two test files. Use TestRunner to have both files in one test suite. (Now both test files should run with just one click) (1 mark)
2. What is the code coverage now? Take a screenshot and paste it into the document (1 Mark)?

Submit all three files separately. Your old test, the optimized test and the Test runner file.