TACKLING WITH TESTING & TEST CASES.
HTTPS://GITHUB.COM/BASANTISCITS/SOEN_6011

PROBLEM 7 REVIEW FUNCTION : F2 TAN(X) VERSION 1.0

Basant Gera (40082433)*
Dept. of Computer Science & Software Engineering
Concordia University
basantgera29@gmail.com

Submitted to : Pankaj Kamthan* & Team*
Dept. of Computer Science & Software Engineering
Concordia University
kamthan@cse.concordia.ca

August 3, 2019

1 Steps taken into account to perform testing are as follows :

- The test cases should fulfill the function range and domain requirements.
- The test case should cover all the functionalities and all possible fields as per the requirement.
- Removing the duplicate test cases created.
- Checking the test cases description.
- Checking the testing steps given in the test cases.
- Verifying the test data.
- Need to focus on the positive, negative and other integrated test cases.
- It should have inputs, actual results and expected result well described and documented.
- Ensure that there should be both negative and positive scenarios as per the project requirement.
- Ensure proper description is added in the test cases in order to understand, the reason for writing this test case.
- Ensure the information related to test environment setup, prerequisites , success and failure end conditions are mentioned.
- Ensure that the test cases should have some specific id or number so that it can easily traceable. That is the traceability should be maintained.
- Test cases should follow the Requirements which is mentioned in Requirement document in problem Number 2.

2 Computing Environment Used are as follows :

- **IDE Used** : Eclipse.
- **IDE Version Used**: Neon 4.6
- **OS used** : Widows 10.
- **Testing Environment in Java for testing**: JUnit 4.0
- **Reviewing Tool used** : Codacy
- **Test Cases Check Based on Review on properties like**:Security,Error Prone,Code Style,Compatibility,Unused Code,Performance,Naming Conventions,Indentation,Comments,spacing.
- Performing Manually testing and testing via JUnit 4.0.
- Done all work in personal computing environment.

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

3 Review Results based on requirements and test cases

Test Case *JUnit or Manually Testing*	Requirement stated	Expected Result	Actual Result	Pass or Fail
SEP-CN2-01	Understand the general expression for Taylor series of a function f(x) and In order to compute the expression tan(x), prior understanding of the function being used is critical.To understand how the Taylor series is formed and can be used on trigonometric functions. tan(289) tan(-165) tan(194) tan(110)	-2.9042108777 0.2679491924 0.2493280028 -2.7474774195	-2.9043973661178435 0.2679450486272898 0.24932545686957802 -2.7477177891011286	Fail *[JUnit]*
SEP-CN2-02	SEP-CN2-01 Copied	SEP-CN2-01 Copied	SEP-CN2-01 Copied	SEP-CN2-01 Copied
SEP-CN2-03	Check periodicity of x value and Check if the x value is less than 180 and Reduce x so that it lies in the range 0 degree and 180 degree by adding or subtracting a suitable multiple of 180 from it. tan(-165)	0.2679491924	0.2679450486272898	Fail *[JUnit]*
SEP-CN2-04	Check if x value is in Q1 and Checking if the x value is in quadrant 1 and If the value is in other quadrants - reduce it to or use symmetry. tan(40)	0.8390996312	0.8390038123135642	Fail *[JUnit]*
SEP-CN2-05	Use polynomial for tangent calculation and To convert x to radians by multiplying it by /180 and To calculate the tangent value using the polynomial Taylor series equation with the given x value. tan(289)	-2.9042108777	-2.9043973661178435	Fail *[JUnit]*

4 Review based on Code Smell by Code Review Tool on Test Cases as per requirement document : Codacy

Test Cases *JUnit or Manually Testing*	Security	Error prone	Unused Code	Performance	Code Style	Naming Conventions	Indentation	Comments	Spacing
SEP-CN2-01	✓	✓	✓	✓	✓	✓	✓	✓	✗
SEP-CN2-02	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1
SEP-CN2-03	✓	✗	✓	✗	✓	✓	✓	✓	✗
-CN2-04	✓	✓	✓	✓	✓	✓	✓	✓	✗
SEP-CN2-05	✓	✓	✓	✓	✓	✓	✓	✓	✗
Over All	✓	✓	✓	✗✓	✓	✓	✓	✓	✗

5 Review based on Code Smell by Code Review Tool on Test Cases in TangentTest.java : Codacy

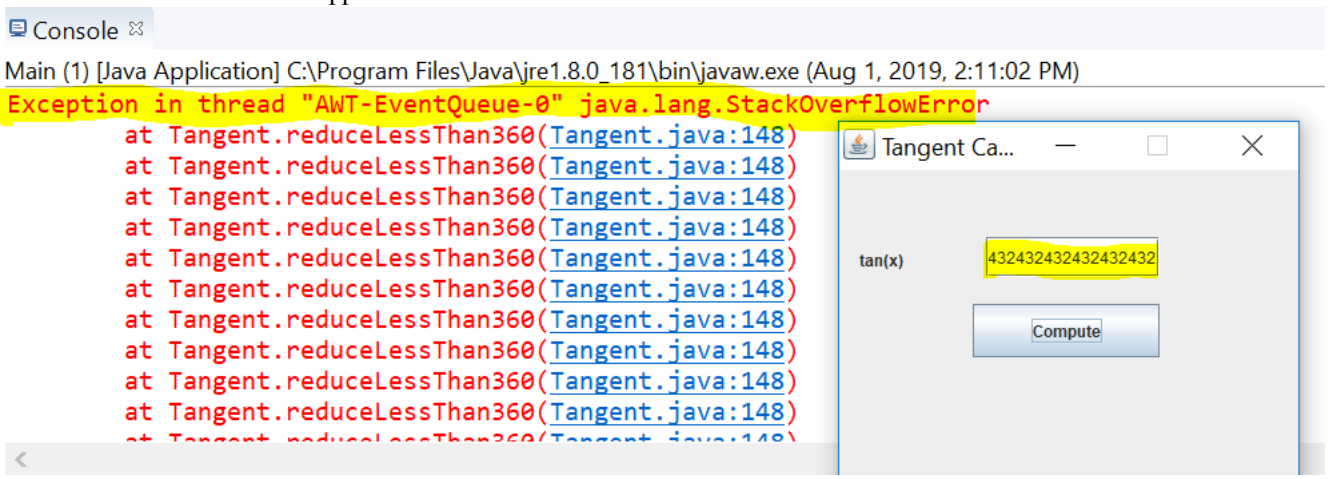
Test Cases *JUnit or Manually Testing*	Security	Error prone	Unused Code	Performance	Code Style	Naming Conventions	Indentation	Comment	Spacing
testAngleInQ1()	✓	✓	✓	✓	✓	✓	✓	✓	✗
testAngleInQ2()	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1	Same As 1
testAngleInQ3()	✓	✓	✓	✓	✓	✓	✓	✓	✗
testAngleInQ4()	✓	✓	✓	✓	✓	✓	✓	✓	✗
testNegativeAngle()	✓	✗	✓	✓	✓	✓	✓	✓	✗
getValue()	✗	✗	✗	✗	✗	✗	✗	✗	✗

6 Comments for code review which is done :

- Since Calculating precision values is important for trigonometric function that's why many test cases fails.
- Used one test case which comes under unused code.

```
@Test
public void getValue() {}
```

- Program seems to be error prone when exceeding the length of number typed in text box which can be controlled due to which this happens.



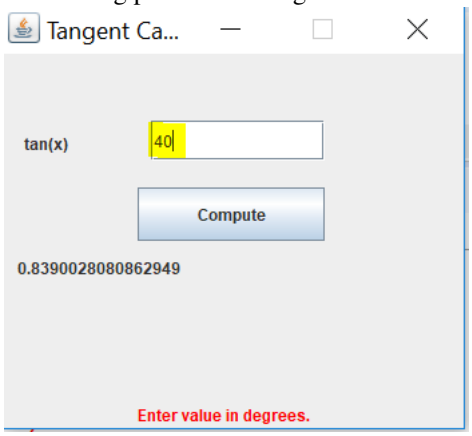
- Since we were told not to used any in build function that programmer has used Math.pow,Math.PI,Math.abs which programmer need to design by itself.

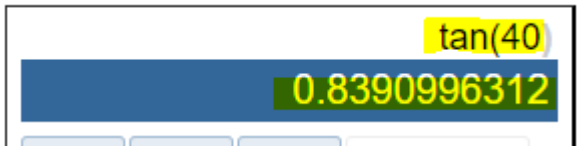
```
// return if the result has to be negative
private boolean getQuadrant() {
  double value = Math.abs(this.userInput);

  return (this.userInput)
     + (Math.pow(this.userInput, 3) / 3)
      + ((2 * Math.pow(this.userInput, 5)) / 15)
      + ((2 * Math.pow(this.userInput, 7)) / 315);
}
private double degreeToRadian(double degree) {
 return degree * (Math.PI / 180);
}
```

- Coding convention followed somewhere.
- Commenting done some where in the code and somewhere not and made java doc for few functions for understandably of code.Picture below shows no comments.

```
boolean inputNegative;
double result;

public Tangent(double userInput) {
  this.userInput = userInput;
  this.resultNegative = false;
  this.calculateReciprocal = false;
  this.inputNegative = false;
  this.process();
}
```

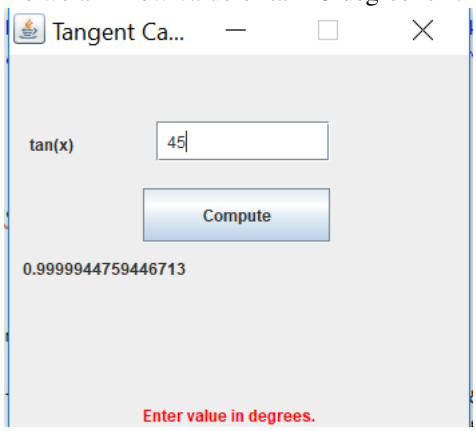
- Test cases not matching with the requirement which user written as per as per problem number 2.
- Calculating positive and negative value is correct up to 5 or sometimes 6 points but then it does not match.





Since it show value are not precise after some points

- As we all know value of tan 45 degree is 1.But programmer calculates something else.



References

[1] https://www.rapidtables.com/test/testt.html

[2] https://www.calculator.net/scientific-calculator.html

[3] http://www.softwaretestingclass.com/test-case-review-process-tips-and-tricks/

[4] https://www.codacy.com/