

F10: σ

Deliverable 1

1 Problem 1 -Description

This symbol notifies standard deviation(SD), it is a method that measures extent of variation or separation of data values. The symbol σ is taken from Greek letter sigma. If the value of standard deviation is low it indicates data points are close to the mean, while higher value indicates wider range of values. The values that goes into the function is called as domain, all possible outcome of function is co-domain and actual output from system is called Range. Things required to calculate standard deviation are mean and variance. Mean is calculated by summing up all values and dividing it by total number of values and variance is calculated by taking difference of each, squaring it and then averaging the results. Calculator will compute population and sample standard deviation. The Range for standard deviation is between negative infinity to positive infinity

Properties of Standard Deviation

1. Measures spread-out numbers.
2. It is expressed in same unit as data.
3. Used to measure statistical results such as margin of errors.
4. Using standard deviation, we can calculate normal, extra-large and extra small values.

Population Standard Deviation

This is used when an entire population can be measured, and where every member of a population can be sampled. The following is the equation:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$$

Where x_i is one individual value, μ is the mean/expected value, N is the total number of values

Sample Standard Deviation

In this it is not possible to sample every member within a population, so above equation must be modified such that the deviation can be measured through random samples of the population.

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

Where x_i is one sample value, \bar{x} is the sample mean and N is number of sample value.

2 Problem 2 -Requirements And Assumptions

2.1 Functional Requirements

1: Low priority , 5:High Priority

2.1.1 Input Requirement

When the user enters zero input/input=null, the function shall pop out an error stating "Input cannot be Null". As number divided by zero leads to infinity. (length !=0)

Priority of this requirement: 5

2.1.2 Length Requirement

When the user enters less than two inputs, the function shall display error message "Enter more than one input". This is because minimum length required to calculate standard deviation is 2.(min length=2)

Priority of this requirement: 5

2.1.3 Multiple inputs

User shall be able to enter 'n' inputs from the console provided by function. There is no limit for inputs. Function has to handle inputs.

Priority of this requirement: 3

2.1.4 Handling real numbers

As user enters real numbers, the function shall be able to accept, process it and output's as real numbers.

Priority of this requirement: 3

2.1.5 Calculate Mean

When calculating standard deviation function, the function shall automatically call mean and retrieve result without notice to user.

Priority of this requirement: 4

2.1.6 Calculate variance

When calculating standard deviation function, the function shall calculate variance explicitly or within the function and retrieve result without getting to user notice.

Priority of this requirement: 4

2.1.7 Display Output

User shall be able to see only final output of function which is Standard deviation value, and output will be displayed on console. User shall not have any problem while viewing output.

Priority of this requirement: 5

2.2 Non-Functional Requirements

2.2.1 Performance

It is analysed on how the function responds to given input provided at certain time.

2.2.2 Correctness

Correctness of the function is measurable by checking input-output behaviour. The generated output is verified by comparing results computed manually.

2.2.3 Consistency

The consistency of the function remains same throughout all calculators, as math definition for standard deviation is unchangeable. And consistency of output for all input also remains same.

2.2.4 Accessibility

Defines how easily the function is accessible by all kinds of stakeholders, in different platforms and with integration of hardware.

2.2.5 Usability

The function is easily usable by all stakeholders and also learn-able to achieve specific needs.

2.3 Constraints

1. Interfaces for calculating standard has already been defined and is not bounded to change.
2. Some calculators are region specific.
3. Mode for selecting standard deviation may vary from different calculators.
4. Users from non-mathematical background will have difficulties in accessing function through calculator.

2.4 Assumptions

1. All inputs provided by users are real numbers.
2. Inputs are of population standard deviation.
3. Users will be familiar with accessing functions in calculator.
4. All calculators that supports Math and Statistics contains standard deviation function.
5. Value of standard deviation directly proportional to data points or mean value.

3 Problem 3 - Pseudocode And Algorithm

Algorithm 1 Squareroot(number)- common for both Iterative and Recursive

begin:

1. SET Sqrt=number/2
 2. Do
 3. temp=sqrt
 4. Add temp value with (number/temp) and divide whole by 2
 5. CONTINUE WHILE ((temp - result) != 0)
 6. RETURN result
- end
-

3.1 Using Iterative Approach

Advantages:

1. Easier to understand.
2. Saves memory.
3. Iterative approach can enhance time and space requirement.
4. Fast in execution.

Disadvantages:

1. The iterative repeatedly dynamically allocate or resize memory blocks.
2. Time consuming to Recursive approach.
3. Contains duplicate code.
4. Iteration makes the code longer.

Algorithm 2 calculateMean(array) And CalculateStandardDeviation (array[])

begin:

1. SET Counter=0
2. FOR Counter<length THEN
3. Add all values
4. INCREMENT Counter by 1
5. mean= Total/length
6. RETURN mean

end

begin:

1. COMPUTE Mean(array)
 2. SET Counter=0
 4. FOR Counter<length THEN
 5. Subtract each value from Mean
 6. Square the subtracted value and keep adding with previous squared values
 7. INCREMENT Counter by 1
 9. result= calculatedsum/length
 10. COMPUTE Squareroot(result)
 11. RETURN computed result
 12. REPEAT the algorithm for new value
- end
-

3.2 Using Recursive Algorithm

Advantages:

1. Allows to allocate additional automatic objects at each function call.
2. Faster compared to Iterative approach.
3. Makes the problem more elegant.
4. Reduces Time complexity of a program.

Disadvantages:

1. Makes the execution slower.
2. Takes up more of stack storage.
3. Difficult to understand and trace.

Algorithm 3 calcStd(List)

begin:

calcAvg(List)

1. COMPUTE calcSum(list,0)
2. Divide sum by size of list
3. RETURN result

end

begin:

calcSum(List, i)

1. IF $i < \text{size of list}$ THEN
2. Add each element(i) with its next element(calcSum(list, $i+1$))
3. RETURN result
4. ELSE
5. RETURN 0

end

begin:

calcpow(List,avg,i)

1. IF $i < \text{size}$ THEN
2. Subtract each value with Avg
3. Square the result
4. RETURN square
5. ELSE
6. RETURN 0

end

begin:

sumSquareDiffs(List,avg,i)

1. IF $i < \text{size}$ THEN
2. COMPUTE calcpow(List,avg,i)
3. keep adding result of each element
4. ELSE
5. RETURN 0

end

begin:

calcStd(List)

1. COMPUTE calcAvg(List)
 2. COMPUTE sumSquareDiffs(List,avg,i) taking avg value from step2
 3. COMPUTE Squareroot(sum) take sum value from step3 and Squareroot() from algorithm 1
 4. RETURN result
- end
-

4 Changes from D1 to D2

Added Extra Functional Requirements

2.1.8 Entering consecutive repetition of same number.

2.1.9 Showing difference between sample and population SD

5 Problem 4

5.1 Debuggers

Debugging is the common process of identifying and removing bugs, errors or abnormalities from programs. It is a required skill for all Java developers, because it helps to find subtle bug that are not visible during code reviews or that only happens when a specific condition occurs. The Eclipse Java IDE provides many debugging tools and views grouped in the Debug Perspective to help the you as a developer debug effectively and efficiently. There are many improvements in Eclipse Java Development Tools(JDT) included from Eclipse Oxygen on-wards.

The debuggers used for this project are:

1. Command Line Debuggers
2. IDE Debuggers

The Images for 1 and 2 types have been attached in *Appendix section* at the end.

Challenges faced: Also tried installing Eclipse-Mirur visual Debugger which gives one dimensional view of the debugging result but after installing it was not working.

1.Java debugger(JDB): is a tool used to debug java program in command Line. It implements the java platform debugger architecture. It helps in detecting and fixing bugs in program with help of java debug interface.

Advantages

1. Allows breakpoints, an explicit way of stop or pause application.
2. Enables stepping process, its a debugger feature that lets you to execute line by line.
3. It also throws uncaught exception with its cause.

Disadvantages

1. It is slow compared to IDE debuggers.
2. Complete visual flow of debugging process in not available.
3. Not user friendly, as compared to IDE.

2. IDE Debugger: These are built in debugging tool with eclipse for java. Eclipse allows to begin java program in debug mode. It has all features that consists in JDB and also allows extra features such as watchpoint-is a break-point set on field. Exception breakpoints, method breakpoints, step filter, hit count, remote debugging and drop to frame- allows to select level of frame.

Advantages

1. Debug perspective offers additional view.
2. Stepping commands(stop,start,resume) are easily accessible, as its built as buttons.
3. Enables creation of own debugger.
4. Consumes less time and effort for organizing resource, provides shortcuts and track mistakes.

Disadvantages

1. As high end desktop application, these cannot run on production machine.
2. Possibility of remote debugging will be ruled out for complex environments.
3. Learning curve- maximizing benefits will require lot of time and patience.
4. Will not fix bad code,coding standard or performance problems.

5.2 Quality Attributes

Program is set such that it satisfies quality as expected by user.

1. *Correctness*:

Is satisfied by verifying the results of standard deviation given from the program. This is achieved by comparing with manual calculation or calculating from online tool. Unit testing is also written such that it measures correctness.

2. *Efficiency*:

Measured by time taken to run the program and display the output. Here the program is utilizing minimum amount of resources as all functions are implemented and in same class memory required is also less. All these results in more efficiency.

3. *Maintainable*:

As the program is structured using coding standards it is easily maintainable and flexible to future changes. User can easily follow the program. GUI is also maintainable and extendable as its structured and simple to handle.

4. *Robust*:

safety measures are taken to cope up with the errors during execution by incorporating *try and catch block* with letting user know from proper message popup and also GUI buttons are made clear so its less prone to errors.

5. *Usable*: Program is divided into different function so that each function can be executed individually and can be reused in other programs easily. Usage is also not restricted to once.

***Challenges faced*:**

1. Tried installing Ecipse-Jdepend and Eclipse-JDeodorant for estimating quality attributes, but after installation had hard time finding them.
2. Lack of explanation in documentation and as they are huge required information is hard to find.

5.3 Quality Checker

Quality of the program or source code is verified by exposing code under checking tools. For this project there are two tools used: 1.Checkstyle and 2.PMD .

5.3.1 Checkstyle

The source code standard Checkstyle checks are applicable to java program and requires no external libraries. Checkstyle provides many checks such as naming, location of annotation, nested blocks, constant names, empty block, illegal token, type, javadoc, tags, method name, count and much more.

Advantages

1. Portability between IDE's - If we want to use visual studio or my team member is using other IDEs still it exhibits consistency.
2. Easy to use as its integrated as an external tool designed which can be hooked into the code easily. Whereas in Eclipse styler we need to locate the plugin to do same task.
3. Ability of creating own rules- checkstyle provide user to add his own custom rules.

Disadvantages

1. Errors are fixed manually, there is no automatic fix.
2. The symbol used to display errors are same as eclipse warnings which leads to confusion.
3. Few errors are not understandable.
4. It just mentions error does not say in which part user should keep doing trial and error to fix it.

5.3.2 Eclipse-PMD

PMD is a static source code analyzer which finds programming flaws such as unused variables, empty catch blocks, unnecessary object creation, and so forth. Once installed, every time while saving work it also checks for potential problems like possible bugs, duplicate, dead or complicated code. Eclipse PMD offers quick fixes that automatically fixes the problem which can be of single fix or all occurrences in entire code. It categorizes errors based on severity. This is bit advanced tool compared to checkstyle with having more features as its packed with 149 rules.

Advantages

1. Scalability is high as tool runs well for very large source code.
2. Comply with coding standards and delivery quality code.
3. Used by teams to change the nature of code reviews.
4. Reduces software maintenance cost. As its open source tool and time spent is less.
5. PMD can also be customized to meet organization coding standard and quality.

Disadvantages

1. Even this intend user to correct errors manually. Cannot find runtime issues
2. As PMD comes by rules there is certain time clash of rules when used with other tools.
3. The documentation guide is overwhelming to users, it does not give examples of output or how to best run PMD.
4. Creating new rule requires understanding of Abstract syntax tree(AST).

Images for both checkstyle and PMD are added in Appendix section.

6 Problem 6- Unit Testing

6.1 Traceability Record

The below table shows the mapping of test case with requirements.

Table 1: Table shows Traceability Record.

Test Case	Test Case Name	Related Requirement(s)
1	CalculateMeanTest	2.1.5
2	CalculateStdDevTest	2.1.6
3	CalculateStdDevTest	2.1.4
4	CalculateStdDevTest	2.1.3
5	Graphical user Interface	2.1.2
6	Graphical user Interface	2.1.1
7	Graphical user Interface	2.1.7
8	CalculateStdDevTest	2.1.8
9	CalculateStdDevTest	2.1.9

The diagram below shows the results of testing. For testing JUnit framework is used and test results have been tracked.

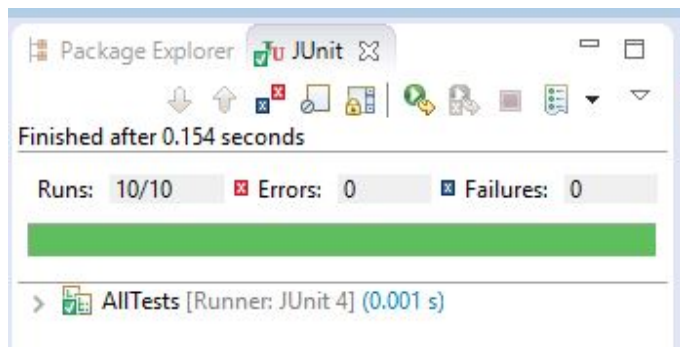


Figure1: Shows the results of JUnit

To check missing scenarios, test coverage was checked using Elc-Emma code coverage tool. Below diagram shows the coverage of testing. Test results for GUI are not written as all the error messages are visible in the interface.

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Test	44.4 %	450	563	1,013
src	44.4 %	450	563	1,013
(default package)	44.4 %	450	563	1,013
StdDevGui.java	0.0 %	0	559	559
AllTests.java	0.0 %	0	3	3
StandardDeviation.java	99.4 %	157	1	158
CalculateMeanTest.java	100.0 %	89	0	89
CalculateSqrtRootTest.java	100.0 %	84	0	84
CalculateStdDevTest.java	100.0 %	120	0	120

Figure2: Shows the test coverage for StandardDeviation.java file

This project have been implemented and interfaced using graphical(GUI) user interface. For GUI implementation Java swings have been used. Below diagram shows the snippet of Interface.

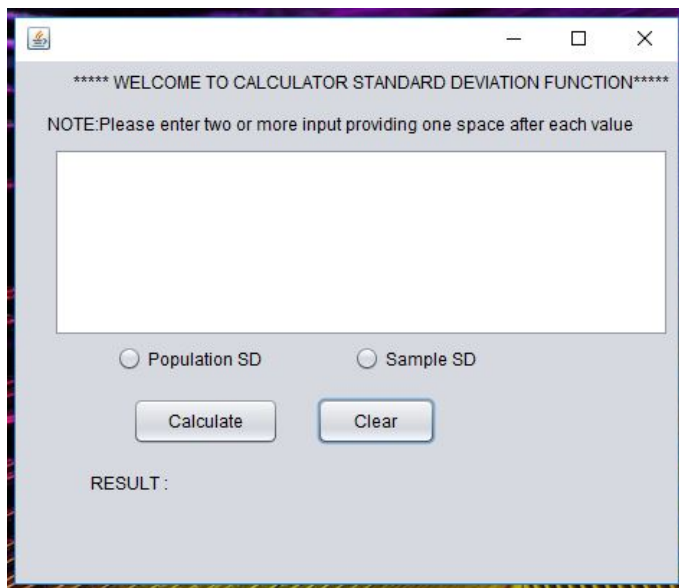


Figure3: Shows the Graphical User Interface(GUI)

- Deliverable 3 -

7 Problem 5 - Review of source code

Review of function F1: arccos(x)

The source code review for function F1 is carried out manually as well as automatically. The review for above function is being carried by multiple ways:

1. Manually
2. Tools
 - (a) Eclipse Find Bugs
 - (b) Eclipse – PMD
 - (c) Eclipse – checkstyle

For manual check "google java style standard guidelines" were used, as per group decision made with the team during problem 4. Along with manual, for individual review, filtered few tools in which some of feature of google java style is inbuilt. And they were PMD(Programming Mistake Detector) and Checkstyle.

7.1 Results observed through manual Review

1. Naming convention are not followed according to coding standard. Example:

(a) **Class name** :Acos

Name is very short and looking at name user will not recognize or predict function as there is no function called Acos. Instead it could be ArccosImp.

(b) **Method name** : sum

Short name and it does not say sum of which function.

2. Function name says sum(), along with calculation of sum there is also calculation of range inside function. So, I recommend changing function name.

3. When the wrong input is given, along with the error message result is also printed, which confuses the user. Example: when input is 2, output printed is 0.0000 radian which is wrong. Avoid printing results for wrong input.

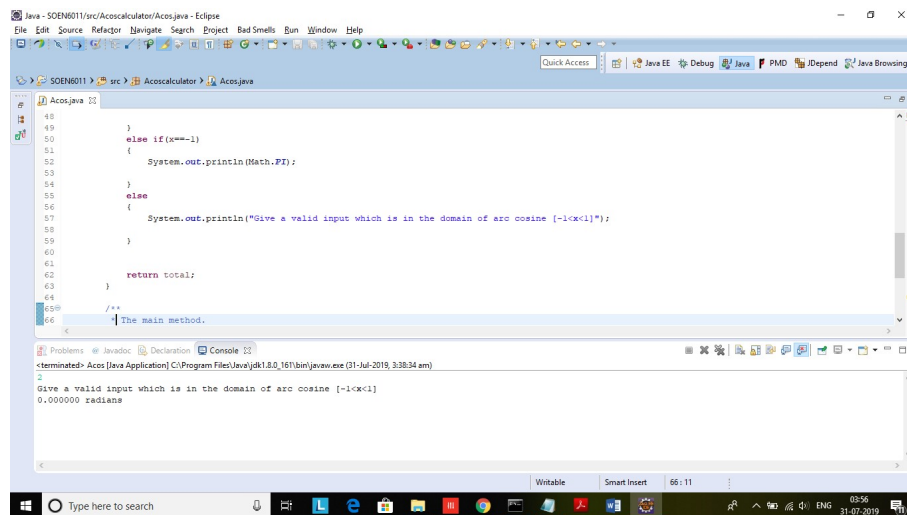


Figure4: Screenshot displaying result

4. Error message is not (above image) promising.

5. Problem statement does not say to restrict the display rounding to 5 decimal places. Instead it predicted to be calculator function, so it should display complete value.

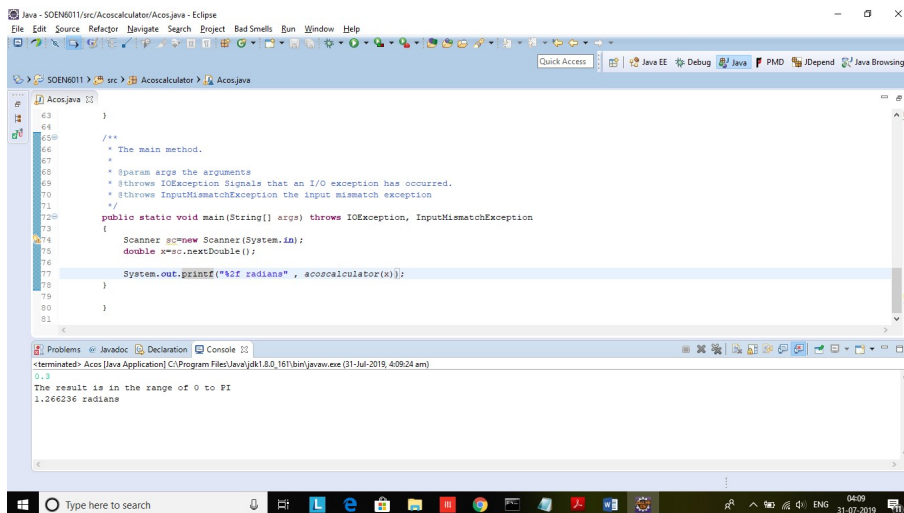


Figure5: Screenshot displaying result

7.2 Tools

7.2.1 Results are observed by running Find-bugs tool

FindBugs is an open source code analyser tool which detects possible bugs in java. Here errors are classified into four groups: (a)scariest, (b) scary, (c) troubling and (d) of concern.

Result:No bugs encountered. Screenshot below exhibit proof.

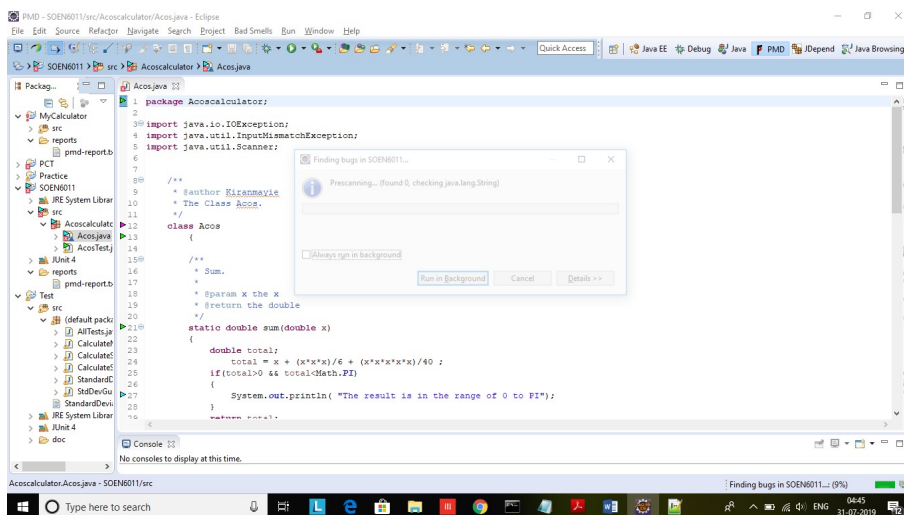


Figure6: Compilation of FindBug tool

7.2.2 Below are the results generated from PMD tool

PMD is eclipse plug-in to check quality of source code. Described more in problem 4. The below screenshot shows the result generated by PMD tool. Showed one example in

screenshot showing how to trace error. In PMD file each error is repeated twice, just ignore repetition. I feel most of the errors are really valid. Example: error on class name, error for using println in unnecessary places etc. Attached complete PMD bug report with deliverable.

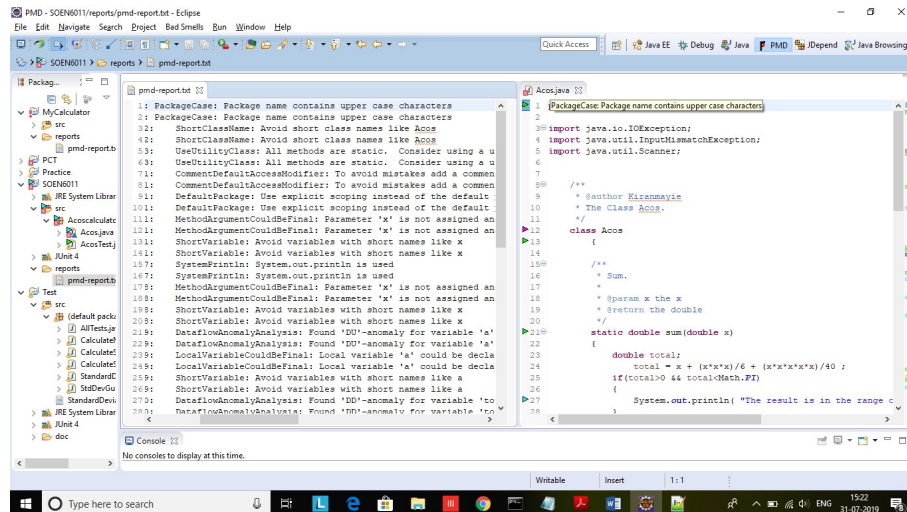


Figure7: Screenshot displaying result of PMD

7.2.3 Following are the results observed through Check style tool

1. Indentation: Observed in whole program.(Same principle as followed in google java style)
2. Whitespace expected around equals, curly brackets etc.

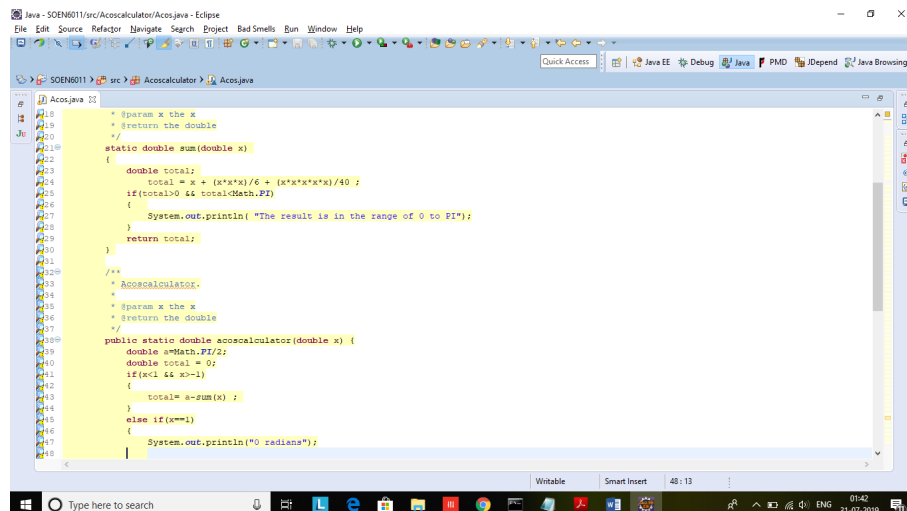


Figure8: Screenshot displaying result of checkstyle

8 Problem 7 - Review Test function

Review of function F2:tan(x)

Testing of the above assigned function is done manually in comparison with requirements mentioned in problem 2 and also reviewed the test cases written using Junit. All the observations and variation are noted and recorded.

8.1 The following are the Manual Testing result

Table 2 : Shows manual testing results

Requirement No:	Input given	Expected O/P	Actual O/P	TestResult	Comments
2.1.Hardware	Calculator.java	Must run on both Mac OS and Windows.	AsExpected	Pass	NA
2.2.Software	Calculator.java	The application shall use only standard Java functions.	AsExpected	Pass	NA
2.3.Operating System	Calculator.java	Must run on both Mac OS and Windows.	NA	NA	Repeated requirement(same as requirement 2.1)
2.4.Human Interfaces	Calculator.java	The application shall function in a similar way as a regular calculator.	AsExpected	Pass	NA
2.4.1.Input	Calculator.java	The application shall allow the user to enter the numbers manually through his/her keyboard.	AsExpected	Pass	NA
2.4.2.Output	Calculator.java	The application shall display the results on the appropriate output area.	AsExpected	Pass	NA

3.1.Input (major function require- ments)	Calculator.java	The input func- tion shall accept the input from the user via the terminal, vali- date it and send it to the appro- priate function to calculate the output.	AsExpected	Pass	NA
3.2.Calculate	Calculator.java	This function will accept the correct in- put from the input func- tion(through arguments)and will perform the operations requested by the user.	AsExpected	Pass	NA
3.3.Output	Calculator.java	Function will display the result, when the equals(=) sign button is pressed(GUI).	AsExpected	Pass	NA
4.1.Illegal Input Sequence	Calculator.java	Appropriate er- ror message shall be printed.	AsExpected	Pass	It is displayed in console, user has to clear message manually. If user forgets to clear input is take along with message (Figure 2)
4.2.Division By zero	Calculator.java	shall be detected by the program and an appro- priate error mes- sage is printed.	Not AsEx- pected	Fail	Message not clear enough. (Figure 3)
4.3.Overflow/ Underflow	Calculator.java	Overflow and underflow do not need to be detected.	AsExpected	Pass	NA

Screenshots for above Commented requirements

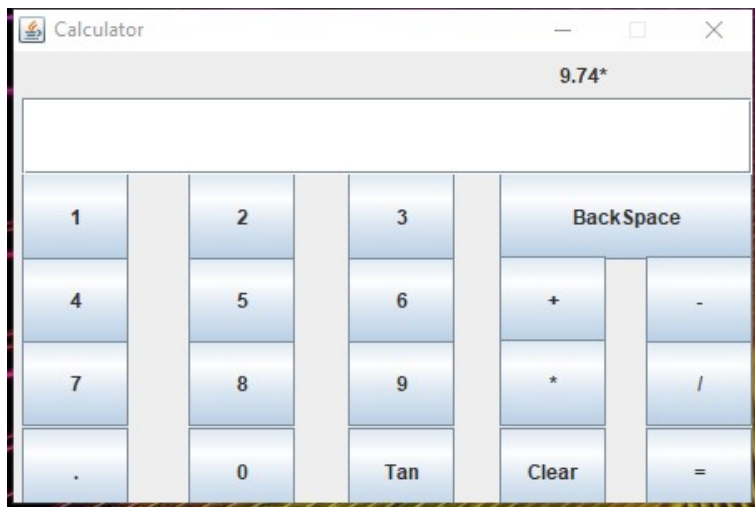


Figure9: Screenshot for Requirement 2.4.1(Human Interface)

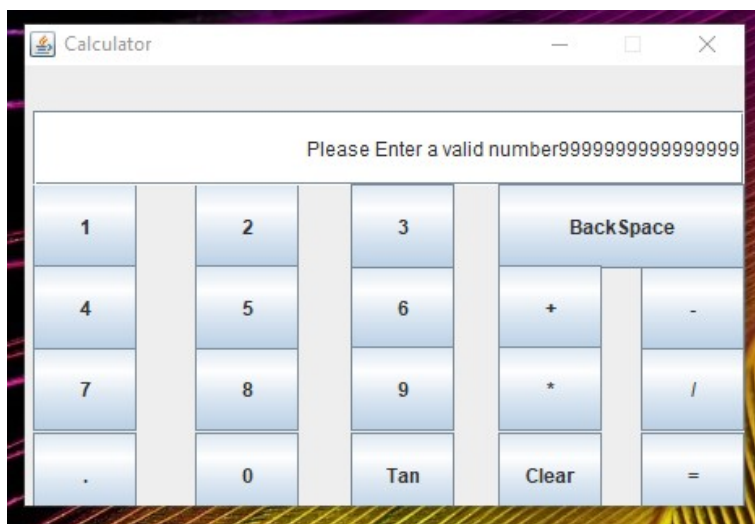


Figure10: Screenshot for 4.1. Illegal Input Sequence

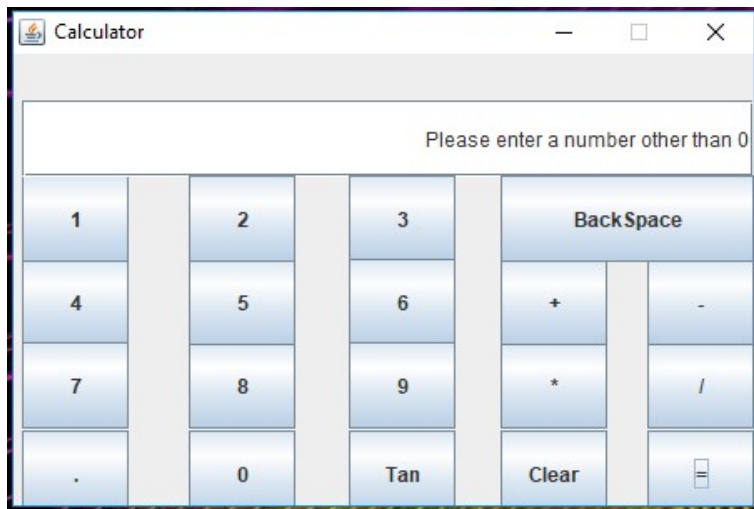


Figure11: Screenshot for requirement 4.2. (Division By zero)

8.2 Junit Review Results

Summary of work exist:

1. Test cases are noted for all functions and are running successfully.
2. Extra functions such as add, multiply, subtract and divide are implemented along with assigned `tan()` function. Same are tested.
3. There are ten testcase written for `tan()`, one each for add, subtract, multiply and two for division.

My observations:

1. For Calculating `tan()` function test cases are not written Negative values.
Action- verified for negative values and application is working as expected.
2. Not tested for null value and zero .
Action- Verified testing and is working as expected.
3. Not tested for rational and irrationals numbers.
Action- Verified testing and is working as expected.

9 Appendix

```

CA: Command Prompt - jdb StandardDeviation

Population standard deviation=0.5
Sample Standard deviation=0.7071067811865475
d:\SEP\Nayana\Project\Deliverable 2>jdb StandardDeviation
Initializing jdb ...
> run
run StandardDeviation
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
>
VM Started: ****WELCOME TO CALCULATOR F10 FUNCTION****
Lets begin.....
Press 0(Zero) to exit
Enter length of input :
> stop in StandardDeviation.calculatePopulationStdDev
Set breakpoint StandardDeviation.calculatePopulationStdDev
> run StandardDeviation
VM already running. Use 'cont' to continue after events.
> stop at StandardDeviation.squareRoot
Use 'stop at' to set a breakpoint at a line number
Usage: stop at <class>:<line_number> or
      stop in <class>.<method_name>[(argument_type,...)]
> stop in StandardDeviation.squareRoot
Set breakpoint StandardDeviation.squareRoot
> run StandardDeviation
VM already running. Use 'cont' to continue after events.
> cont
Nothing suspended.
> run StandardDeviation
VM already running. Use 'cont' to continue after events.
>

```

Figure1: Command-line Debugger used for F10 function (Problem 4)

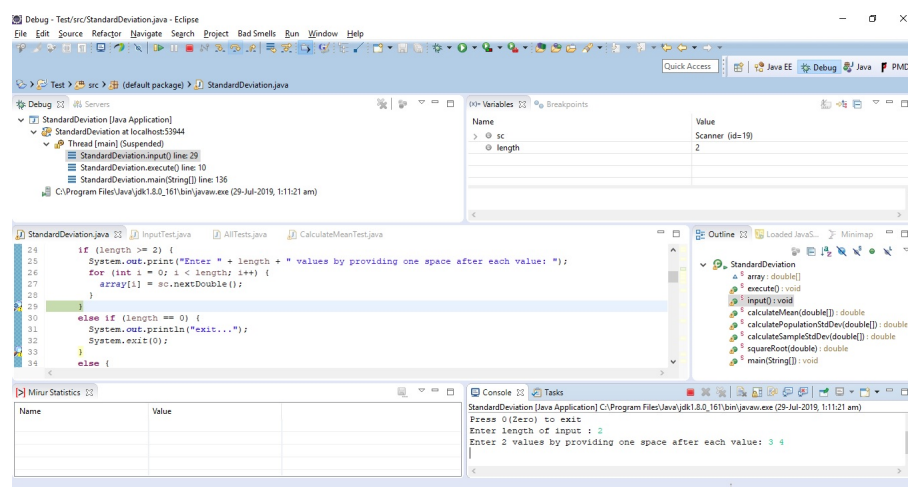


Figure2: Eclipse IDE-Debugger (Problem 4)

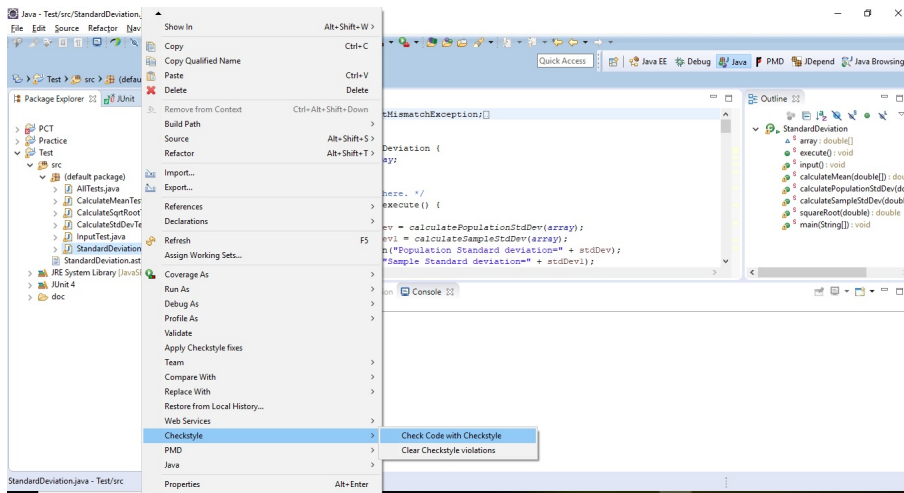


Figure3: Eclipse-CheckStyle (Problem 6)

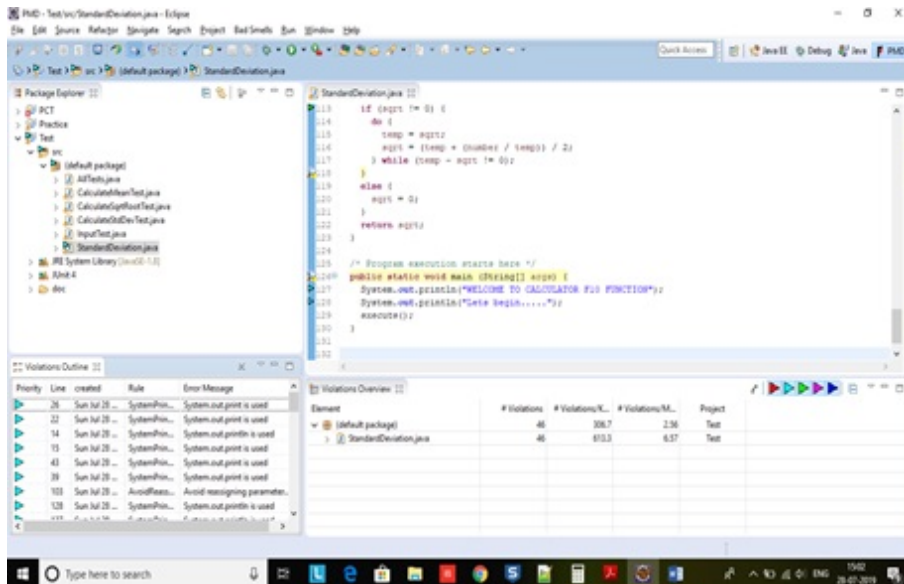


Figure4: Eclipse-PMD quality checker (Problem 6)

10 Reference

<https://ieeexplore.ieee.org/stamp/stamp.jsp>
<https://www.calculator.net/standard-deviation-calculator.html>
<https://www.mathsisfun.com/data/>
<https://tex.stackexchange.com/questions/88388/how-to-have-the-title-at-the-top-of-a-latex-document>
<https://tex.stackexchange.com/questions/456051/standard-deviation>
<https://www.reqview.com/doc/iso-iec-ieee-29148-srs-example.html>
<https://users.csc.calpoly.edu/~jdb/bey/SWE/pdl_std.html>
<https://benpfaff.org/writings/clc/recursion-vs-iteration.html>
<https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>
<https://stackoverflow.com/questions/29022672/calculating-standard-deviation-of-array-recursively>
<https://tutorialspoint.com/jdb/>
<https://docs.oracle.com/javase/7/docs/technotes/tools/windows/jdb.html>
<https://dzone.com/articles/definitive-list-7-javaIDE>
<https://stackoverflow.com/questions/1of-using-checkstyle-rather-than-using-eclipse-built-in-code-formatter>
<https://marketplace.eclipse.org/content/eclipse-pmd>
<https://www.eclipsezone.com/articles/pmd/>
<http://www.cs.cmu.edu/~aldrich/courses/654/tools/hsu-pmd-07.pdf>
<https://en.wikipedia.org/wiki/FindBugs>
<https://google.github.io/styleguide/javaguide.html#s4.2-block-indentation>
<https://www.mathsisfun.com/scientific-calculator.html>
<https://www.overleaf.com/>