UNIVERSITÉ Concordia UNIVERSITY

SOEN 6011-Software Engineering Processes

Project Report on **Function 8 : Standard Deviation (σ)**

Submitted to:
Dr. Pankaj Kamthan

Aniket Tailor (40195068)

August 05, 2022

1 Problem no. 1

Introduction

The Standard Deviation is a measure of how spread out numbers are around the mean. It is denoted by Greek letter sigma σ. A low standard deviation implies that the data are grouped around the mean, whereas a large standard deviation shows that the data are more dispersed. In contrast, a high or low standard deviation indicates that the data points are, respectively, above or below the mean. A standard deviation that is close to zero implies that the data points are close to the mean.

$$(StandardDeviation) \quad \sigma = \sqrt{\frac{\Sigma(x-\bar{x})^2}{n}}$$

Where:

σ = Standard Deviation
n = Size of the population
x = each value from population
$\bar{x}$ = Mean of the population

1.1 Domain and Co-Domain

Domain: Domain are the values which are given as input to the function. Hence in this case data values with natural and real numbers till infinity can be considered as domain.

Co-Domain: Co-Domain are the value which are given as output by the function. Therefore real numbers which are non negative and excluding imaginary numbers comes under Co-Domain.

1.2 Characteristics of Standard Deviation

- Only the spread or dispersion around a data set's mean is measured using the standard deviation.
- It can never be negative.
- The standard deviation is zero when all of the values in a data collection are the same because each value is equal to the mean.
- The higher the spread, the higher the standard deviation is for data with about the same mean.
- It is sensitive to outliers.

1.3 Context of Use Model

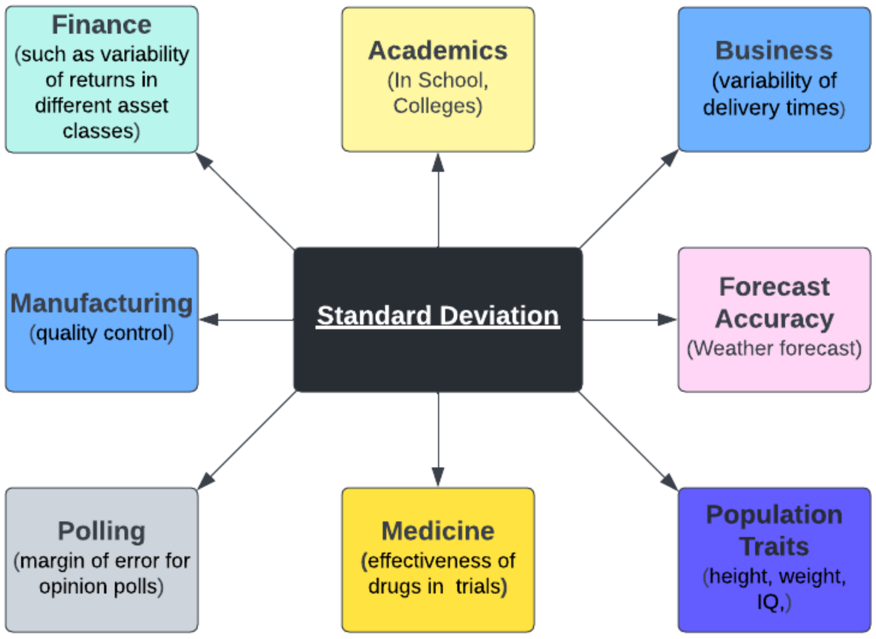


Figure 1.1: Context of use model

2 Problem no. 2

2.1 Assumptions

Any variable, as long as it can be sorted, can have its variance and standard deviation determined. However, the standard deviation is only a useful indicator of dispersion for a measurement variable when the data have a symmetrical distribution, which is frequently a normal distribution. If these presumptions are not true, using the standard deviation to show the variability of observations in range plots and box-and-whisker plots is misleading. This assumption is also a prerequisite for assumptions on the percentage of observations that fall within the range of agreement.

2.2 Requirements

1. **First Requirement**
    - **ID =** F1
    - **Type =** Functional Requirement
    - **Version =** 1.0
    - **Priority =** High
    - **Description =** Standard deviation only deals with numbers and not strings.
2. **Second Requirement**
    - **ID =** F2
    - **Type =** Functional Requirement
    - **Version =** 1.0
    - **Priority =** High
    - **Description =** Standard deviation is the square root of Variance, hence it's value should not be negative.
3. **Third Requirement**
    - **ID =** F3
    - **Type =** Functional Requirement
    - **Version =** 1.0
    - **Priority =** Moderate
    - **Description =** The code should return correct sum of the input data values in order to find Standard Deviation

4. **Fourth Requirement**
    - **ID =** F4
    - **Type =** Functional Requirement
    - **Version =** 1.0
    - **Priority =** High
    - **Description =** For calculating the mean, natural and real numbers should be considered.
5. **Fifth Requirement**
    - **ID =** F5
    - **Type =** Functional Requirement
    - **Version =** 1.0
    - **Priority =** High
    - **Description =** For calculating standard deviation, at-least two numbers should be given as input.
6. **Sixth Requirement**
    - **ID =** F6
    - **Type =** Functional Requirement
    - **Version =** 1.0
    - **Priority =** High
    - **Description =** If all the data values are same then the standard deviation should be 0.
7. **Seventh Requirement**
    - **ID =** NF7
    - **Type =** Non Functional Requirement
    - **Version =** 1.0
    - **Priority =** Low
    - **Description =** Code should be well indented and easy to understand with proper documentation.
8. **Eigth Requirement**
    - **ID =** NF8
    - **Type =** Non Functional Requirement
    - **Version =** 1.0
    - **Priority =** Moderate
    - **Description =** Code should be portable that is a program running on windows 8 should run on windows 10 as well despite of change in performance.

3 Problem no. 3

Algorithms for Standard Deviation

1. Pseudo code of Traditional (Iterative) approach

Algorithm 1 Traditional Algorithm

```
procedure STANDARDDEVIATION( )
    $Sum \leftarrow 0$
    $Mean \leftarrow 0$
    $Var \leftarrow 0$
    $SD \leftarrow 0$
    $Number \leftarrow 0$
    $Size \leftarrow Array.count()$
    $Sum \leftarrow Sum + Sum$
    for $i \leftarrow 0, size$ do
        $Sum+ = Array$
    end for
    $SD = squareRoot(SD/Size)$
    Return SD
end procedure
```

Which one to choose, Traditional or N-Pass?

- Standard deviation can be calculated more quickly and with considerably more efficiency using the n-pass technique, which can accept more data inputs in the form of files. Other than that, the problem is that because it employs recursion rather than iteration, the memory stack would fill up quickly.
- The standard deviation can be calculated using the traditional approach, but it can take some time if the input data is huge and the frequency of the numbers is high. In addition, it is more prone to errors. As a result, the Multi pass or n-pass algorithm is more faster and more effective than the Traditional approach for standard deviation.

Algorithms for Standard Deviation

2. Pseudo code of N-Pass approach

Algorithm 2 N-Pass Algorithm

```
1:  procedure STANDARDDEVIATION( )
2:      $Size \leftarrow Array.count()$
3:      for $i \leftarrow 0, size$ do
4:          $Sum += Array$
5:      end for
6:      $Mean \leftarrow Sum/Size$
7:      for $i \leftarrow 0, size$ do
8:          $SD = SD + power((Array[i] - Mean), 2)$
9:      end for
10:     $Variance = SD/Size$
11:     $SD = SquareRoot(Variance)$
12:     Return SD
13: end procedure

14: procedure POWER(x,y)
15:     for $i \leftarrow 1, y$ do
16:         $Result = Result * x$
17:     end for
18:     Return Result
19: end procedure

20: procedure SQUAREROOT(x)
21:     $temp$
22:     $sqrt \leftarrow \frac{x}{2}$
23:     $temp \leftarrow sqrt$
24:     $sqrt \leftarrow \frac{temp + \frac{x}{temp}}{2}$
25:     while $temp - sqrt \neq 0$ do
26:         $temp \leftarrow sqrt$
27:         $sqrt \leftarrow \frac{temp + \frac{x}{temp}}{2}$
28:     end while
29:     Return $temp$
30: end procedure
```

Mind Map

The below given mind map shows the flow of the pseudo code for finding Standard Deviation. The algorithm will start from Main() method by taking input of data values. It will then call the CalculateSum() procedure to calculate sum of the data elements. With mean and array size in hand, CalculateVariance() method will be called. Internally this method calls Power() function to compute power and atlast SquareRoot() is used to find Standard Deviation.

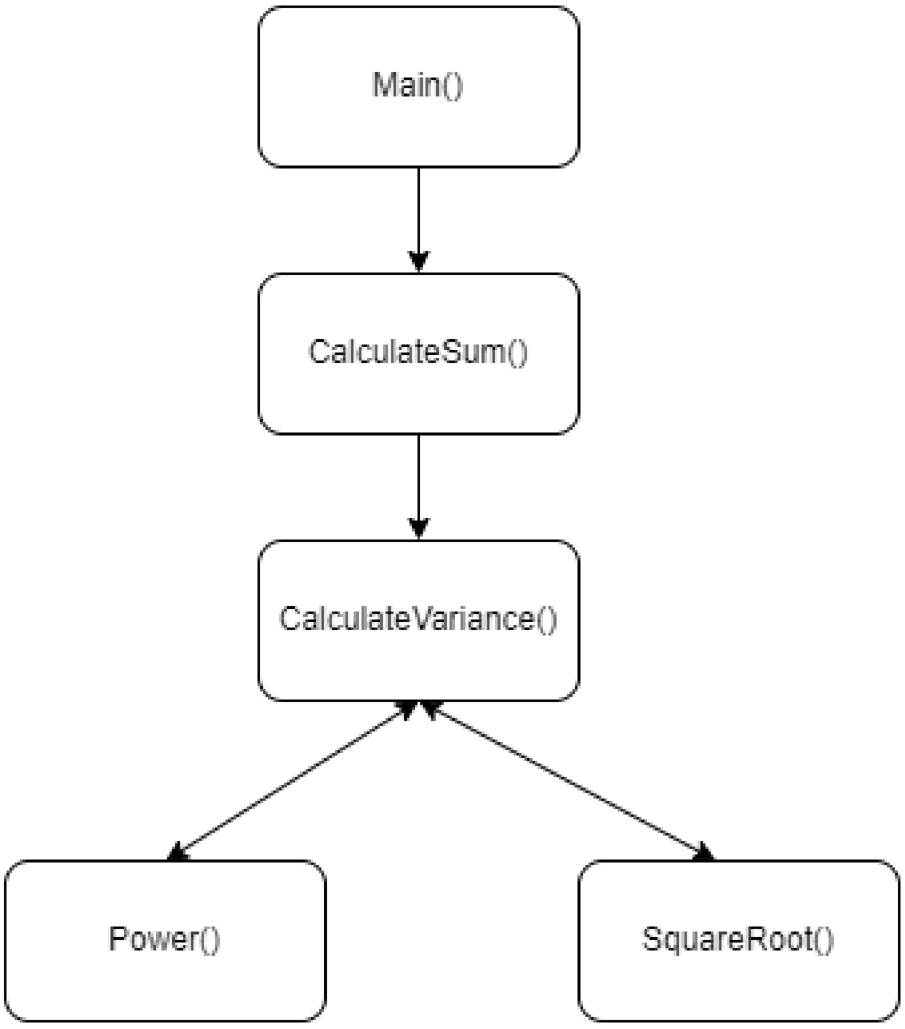


Figure 3.1: Mind Map

Pros of Traditional Algorithm

- This method does not uses the stack so it's faster.
- Traditional algorithm consumes less memory.
- The code is easy to understand and debug.

Cons of Traditional Algorithm

- The iterative alternative is to repeatedly dynamically allocate or resize memory blocks. On many platforms automatic allocation is much faster, to the point that its speed bonus outweighs the speed penalty and storage cost of recursive calls.
- It makes the code longer.
- Sometimes execution and calculation time becomes higher.

Pros of N-Pass Algorithm

- It is faster and much more efficient to use because it makes use of memory upto the mark.
- Can take large set of data to find standard deviation.
- It makes the code smaller.

Cons of N-Pass Algorithm

- The inbuilt garbage collector begins to operate when the memory stack becomes mostly full, deleting the unused memory that has filled the stack.

4 Problem no. 4

4.1 Error & Exception Handling

- Error management makes it possible to gracefully handle both hardware and software problems and enables interrupted execution to continue. Either the programmer creates the necessary codes to handle problems or uses software tools to handle errors when it comes to error handling in software. When mistakes cannot be categorised, they are typically handled by returning unique error codes. For some applications, there exist specialised programs called error handlers that can assist in handling errors. These programs can foresee errors, assisting in recovery without actually terminating the program.
- Responding to undesirable or unexpected events that occur while a computer program is running is known as exception handling. Without this process, exceptions would interfere with a program's regular functioning and cause it to crash. Exception handling deals with these events to prevent this from happening.
- In the given source code to find standard deviation, all the error and exception handling have been taken care of. To summarize a few exception handling errors,
  1. User cannot enter string as input.
  2. If while entering the numerical data values, user enters a string by error then he/she will be prompted to enter the number again at that index value.
  3. User will need to re-enter the size of the array if it is 0.
  4. User should enter at-least two numbers, in order to find Standard Deviation.
  5. If all the elements of the array are same then it should given output as 0.

4.2 Debugger

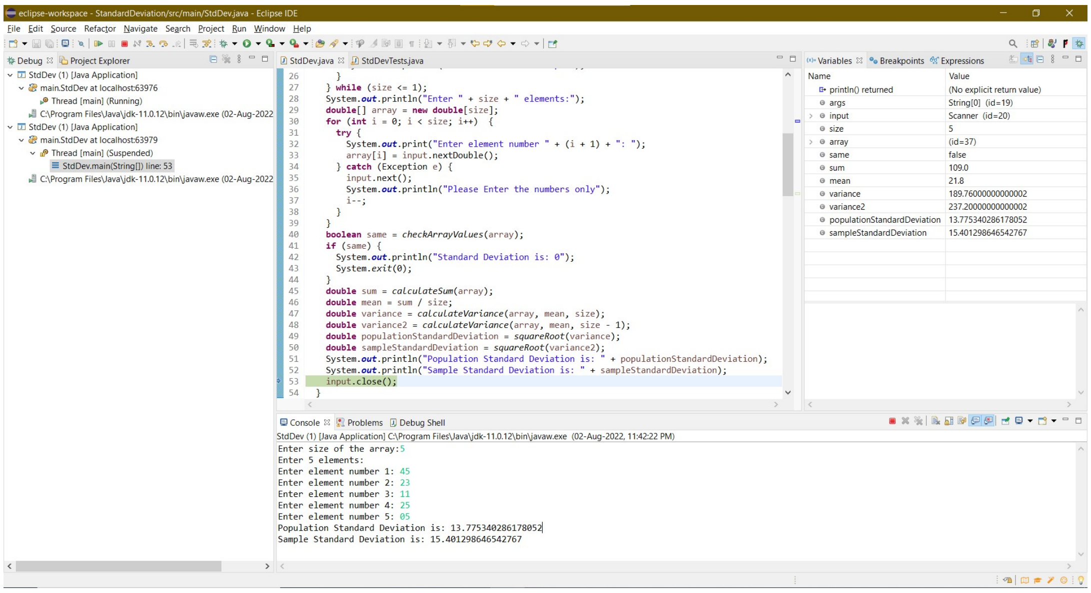
Debugging let you run a program interactively while keeping an eye on the variables and source code as they are being used. A break point in the source code indicates when the program's execution should halt while being fixed. Once the program has been paused, variables can be examined, their contents can be changed, etc. You can launch a Java program in Debug mode using Eclipse. Eclipse has a Debug viewpoint that offers you a ready-made selection of views. And through debug commands, Eclipse enables you to manage the execution flow.

4.2.1 Advantages of debugger

- Debugging helps in solving the unknown problems in the code For example, compile time and run time exceptions.
- We can step into and out of the Eclipse debugger and check the dependencies of other variables with regard to the currently skipped variable.
- Debugging is a very useful tools for inspecting the state of the objects and variables in your code at run time.

4.2.2 Disadvantages of debugger

- While in certain IDE's debuggers we can backtrack the debugger and examine its prior value, the Eclipse debugger does not allow us to do this.
- When multiple threads are active at once, debuggers frequently become stuck and refuse to advance.



eclipse-workspace - StandardDeviation/src/main/StdDev.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Project Explorer
StdDev.java
StdDevTests.java
Variables
Breakpoints
Expressions
Console
Problems
Debug Shell

```
StdDev (1) [Java Application]
  main.StdDev at localhost:63976
    Thread [main] (Running)
  C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (02-Aug-2022
StdDev (1) [Java Application]
  main.StdDev at localhost:63979
    Thread [main] (Suspended)
      StdDev.main(String[]) line: 53
  C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (02-Aug-2022
```

```java
    }
  } while (size <= 1);
  System.out.println("Enter " + size + " elements:");
  double[] array = new double[size];
  for (int i = 0; i < size; i++)  {
    try {
      System.out.print("Enter element number " + (i + 1) + ": ");
      array[i] = input.nextDouble();
    } catch (Exception e) {
      input.next();
      System.out.println("Please Enter the numbers only");
      i--;
    }
  }
  boolean same = checkArrayValues(array);
  if (same) {
    System.out.println("Standard Deviation is: 0");
    System.exit(0);
  }
  double sum = calculateSum(array);
  double mean = sum / size;
  double variance = calculateVariance(array, mean, size);
  double variance2 = calculateVariance(array, mean, size - 1);
  double populationStandardDeviation = squareRoot(variance);
  double sampleStandardDeviation = squareRoot(variance2);
  System.out.println("Population Standard Deviation is: " + populationStandardDeviation);
  System.out.println("Sample Standard Deviation is: " + sampleStandardDeviation);
  input.close();
}
```

Name	Value
println() returned	(No explicit return value)
args	String[0] (id=19)
input	Scanner (id=20)
size	5
array	(id=37)
same	false
sum	109.0
mean	21.8
variance	189.76000000000002
variance2	237.20000000000002
populationStandardDeviation	13.775340286178052
sampleStandardDeviation	15.401298646542767

```
StdDev (1) [Java Application] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe  (02-Aug-2022, 11:42:22 PM)
Enter size of the array:5
Enter 5 elements:
Enter element number 1: 45
Enter element number 2: 23
Enter element number 3: 11
Enter element number 4: 25
Enter element number 5: 05
Population Standard Deviation is: 13.775340286178052
Sample Standard Deviation is: 15.401298646542767
```

Figure 4.1: Debugger in use

4.3 Checkstyle

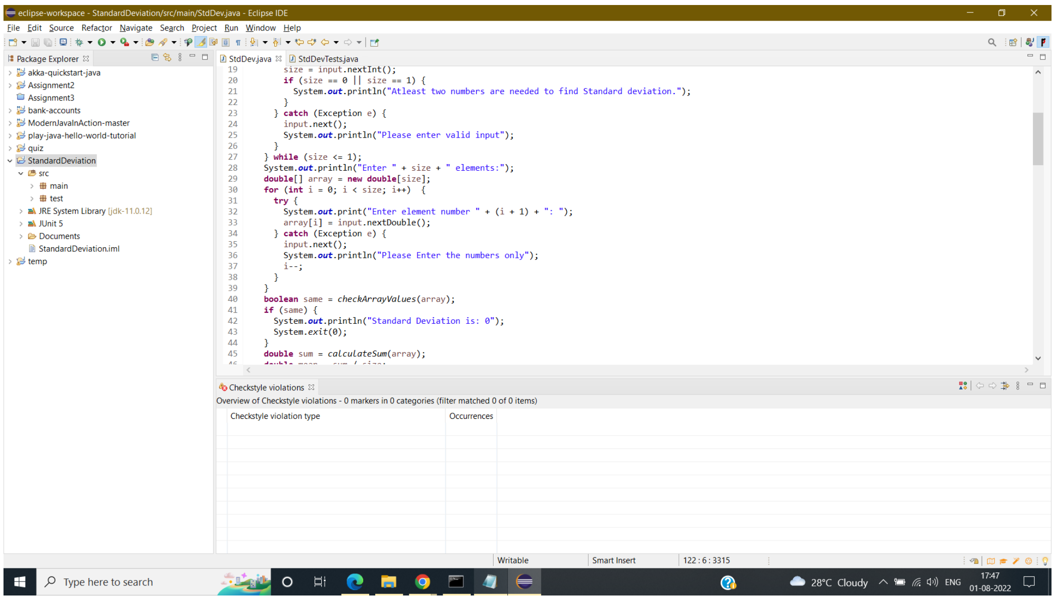
Programmers can use Checkstyle as a development tool to write Java code that follows a coding standard. To relieve humans of this tedious work, it automates the process of checking Java code. It is therefore perfect for initiatives that aim to impose a coding standard. It may examine a variety of elements in your source code. It can identify issues with class and method design. It can also check for formatting and code layout problems.

4.3.1 Advantages of Checkstyle

- Portable between IDEs [Eclipse And IntelliJ]
- Due to the fact that check style was truly intended to be an independent framework, integrating it with your external tools is considerably simpler.
- It lets you format the code with respect to global standards.
- Ability of creating your own rules. Eclipse defines a large set of styles, but checkstyle has more, and you can add your own custom rules.

4.3.2 Disadvantages of Checkstyle

- Removes the ability to do any 'special-case' formatting where an alternate format would make code more readable.
- The checks done by checkstyle do not confirm the correctness or completeness of the code.
- It ties you into using IDEs which support exactly the reformatting features you need.



```java
        size = input.nextInt();
        if (size == 0 || size == 1) {
          System.out.println("Atleast two numbers are needed to find Standard deviation.");
        }
      } catch (Exception e) {
        input.next();
        System.out.println("Please enter valid input");
      }
    } while (size <= 1);
    System.out.println("Enter " + size + " elements:");
    double[] array = new double[size];
    for (int i = 0; i < size; i++)  {
      try {
        System.out.print("Enter element number " + (i + 1) + ": ");
        array[i] = input.nextDouble();
      } catch (Exception e) {
        input.next();
        System.out.println("Please Enter the numbers only");
        i--;
      }
    }
    boolean same = checkArrayValues(array);
    if (same) {
      System.out.println("Standard Deviation is: 0");
      System.exit(0);
    }
    double sum = calculateSum(array);
```

Overview of Checkstyle violations - 0 markers in 0 categories (filter matched 0 of 0 items)

Checkstyle violation type	Occurrences

Figure 4.2: Checkstyle in use

5 Problem no. 5

5.1 Requirements Traceability

In this section, all the unit tests are traced with the requirements from Problem 2. Unit testing is done using the JUnit framework of Java.

Test Case number 1

- **Test Case ID:** UTC1
- **Corresponding Requirement ID:** F3, F4
- **Test Case Method:** testCalculateSum()
- **Description:** The testCalculateSum(), as the name suggests it verifies if the Calculate-Sum() method is calculating the total of all the array elements as expected.

Test Case number 2

- **Test Case ID:** UTC2
- **Corresponding Requirement ID:** F2
- **Test Case Method:** testPower()
- **Description:** The testPower() method, verifies if the Power() method is returning the correct power of a number.

Test Case number 3

- **Test Case ID:** UTC3
- **Corresponding Requirement ID:** F2, F4
- **Test Case Method:** testCalculateVariance()
- **Description:** The testCalculateVariance() method tests the calculateVariance() method in the source code. It makes sure if the responsible method is working fine and calculating variance as expected.

Test Case number 4

- **Test Case ID:** UTC4
- **Corresponding Requirement ID:** F2
- **Test Case Method:** testSquareRoot()
- **Description:** This method verifies the squareRoot() method in source code. As the name goes it checks the square root of a number.

Test Case number 5

- **Test Case ID:** UTC5
- **Corresponding Requirement ID:** F1, F5, NF7
- **Test Case Method:** testMain()
- **Description:** The testMain() method tests out driver function (Main). It tests the main() function with all different types of inputs such as String, Real numbers, Whole numbers. If any inappropriate input is given then proper error and exception handling is in place. It won't let the program terminate or crash.

Test Case number 6

- **Test Case ID:** UTC6
- **Corresponding Requirement ID:** F6
- **Test Case Method:** testMainSameValues()
- **Description:** When all the array elements are same, at that time the Standard Deviation turns out to be 0. Hence this unit test case checks the scenario where all the input values are same.

Bibliography

[1] https://en.wikipedia.org/wiki/Standard_deviation

[2] https://www.investopedia.com/terms/s/standarddeviation.asp

[3] https://www.nlm.nih.gov/nichsr/stats_tutorial/section2/mod8_sd.html

[4] https://benpfaff.org/writings/clc/recursion-vs-iteration.html

[5] https://www.tutorialspoint.com/what-are-the-differences-between-recursion-and-iteration-in-java

[6] https://influentialpoints.com/Training/variance_and_standard_deviation-principles-properties-assumptions

[7] https://www.techtarget.com/searchsoftwarequality/definition/error-handling

[8] https://www.techopedia.com/definition/16626/error-handling

[9] https://checkstyle.sourceforge.io/

[10] https://softwareengineering.stackexchange.com/questions/92256/advantages-and-disadvantages-of-forced-code-reformat