

**Assignment 1**

**PRT 452 SOFTWARE ENGINEERING: PROCESS AND TOOLS**

**SHRIJANA BHANDARI**

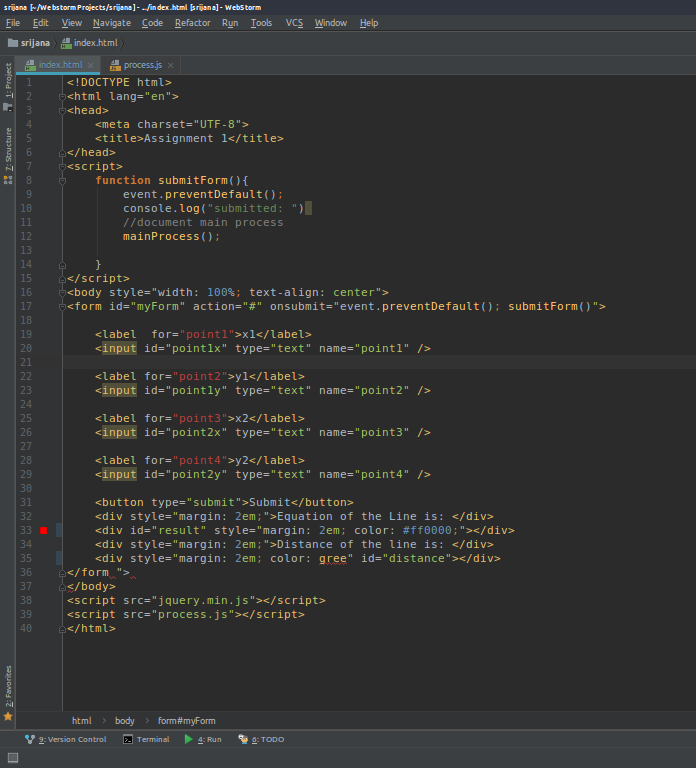
**(s300775)**

**Question 1:**

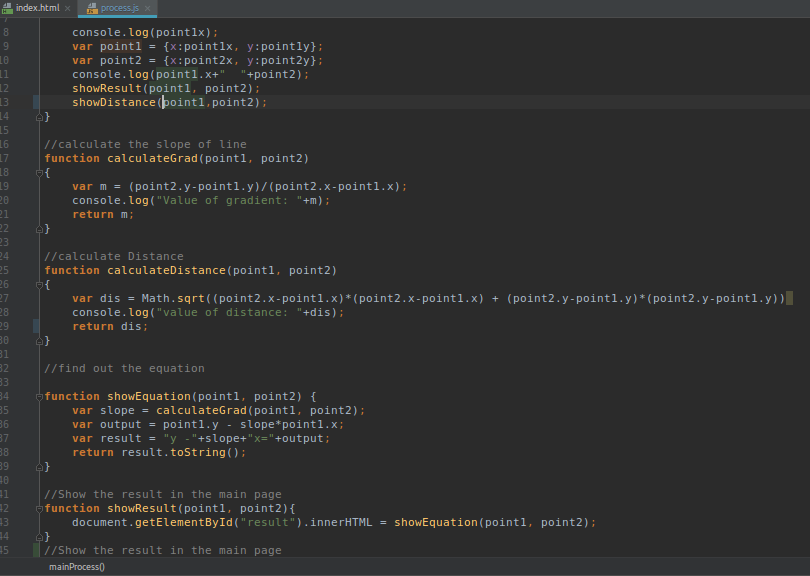
**Screenshots and the steps I followed to create the program**

**Before refactoring.**

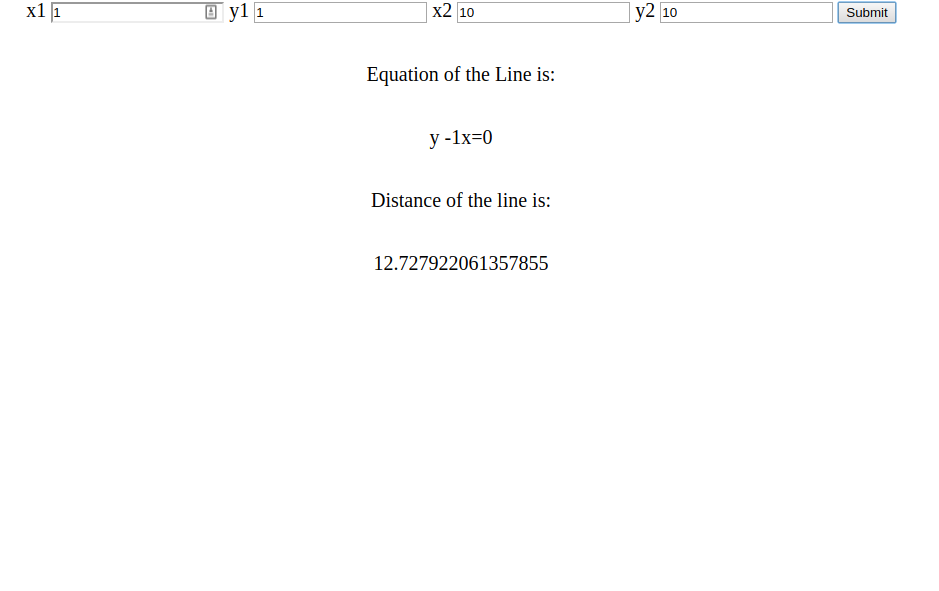
1: This is a Html filewhere the users can put inputs and creates a submit file.

****

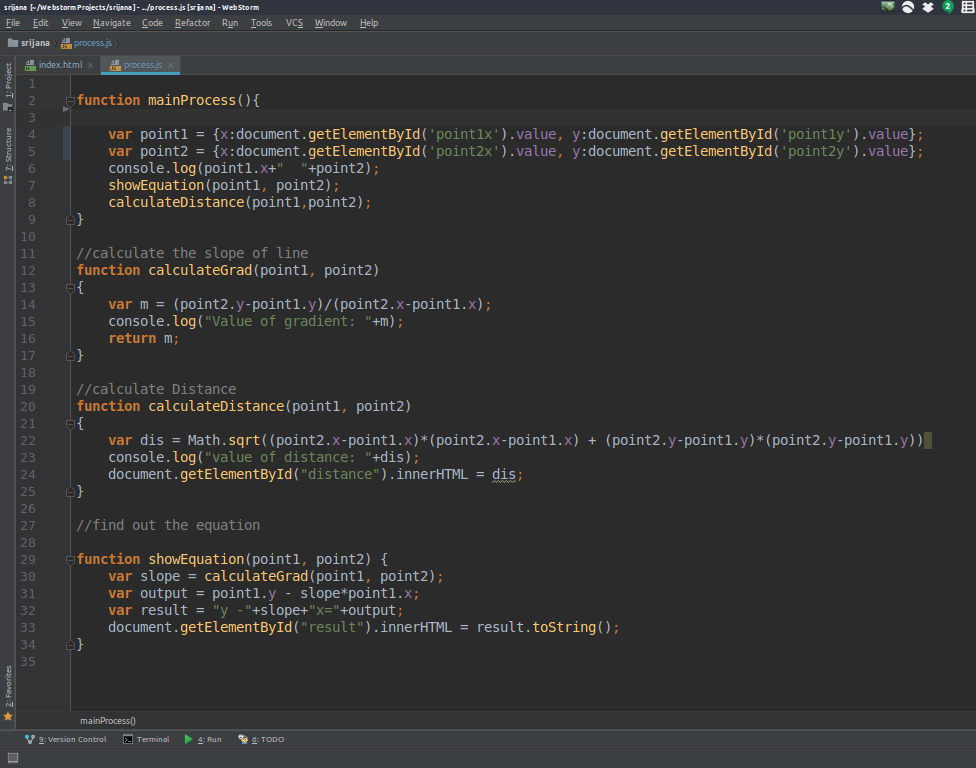
2. In this step logic has been defined to calculate the gradient distance and equation of line.



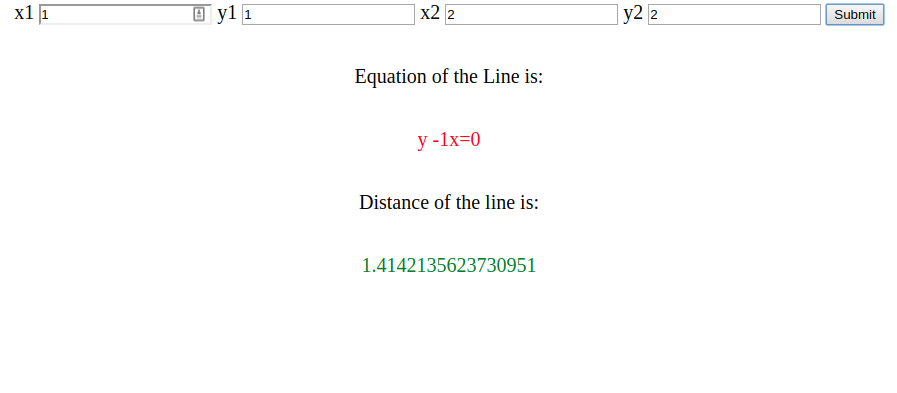
3. Output page



Code after refactoring:

2. Process file 

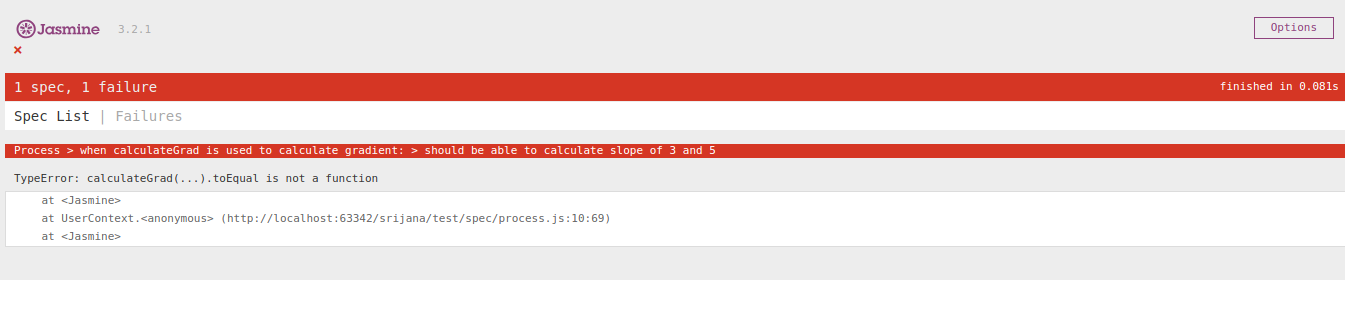
3. Output page

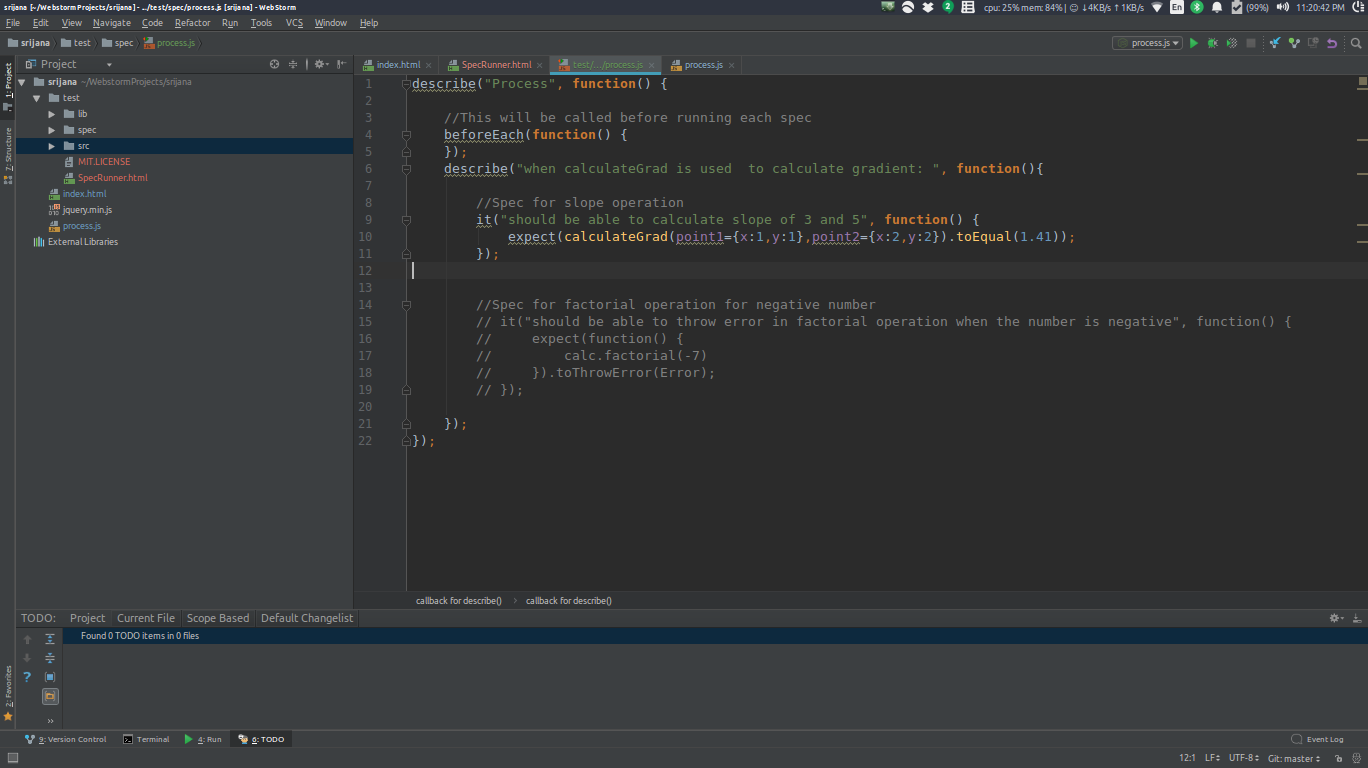


For test driven approach I have used jasmine testing library for javascript.

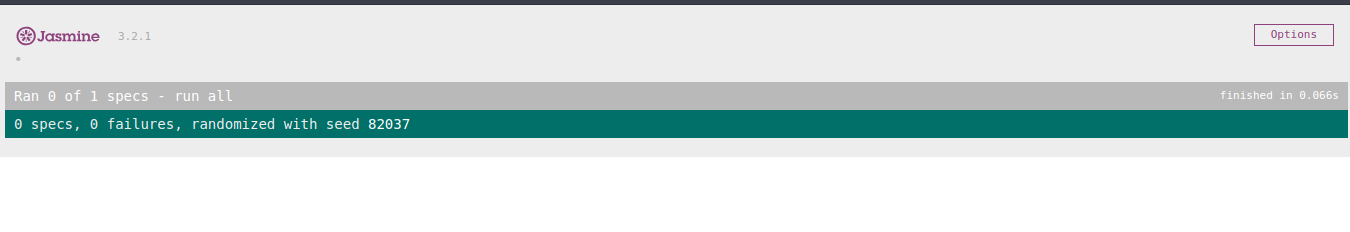
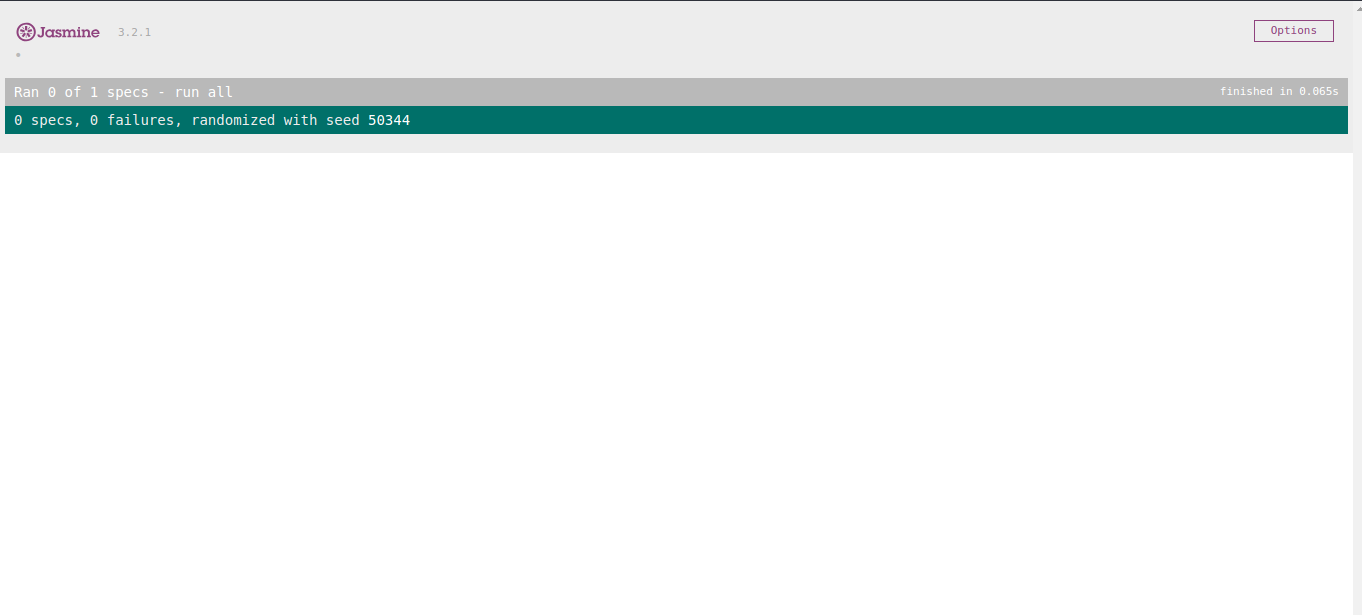
1. Failed case

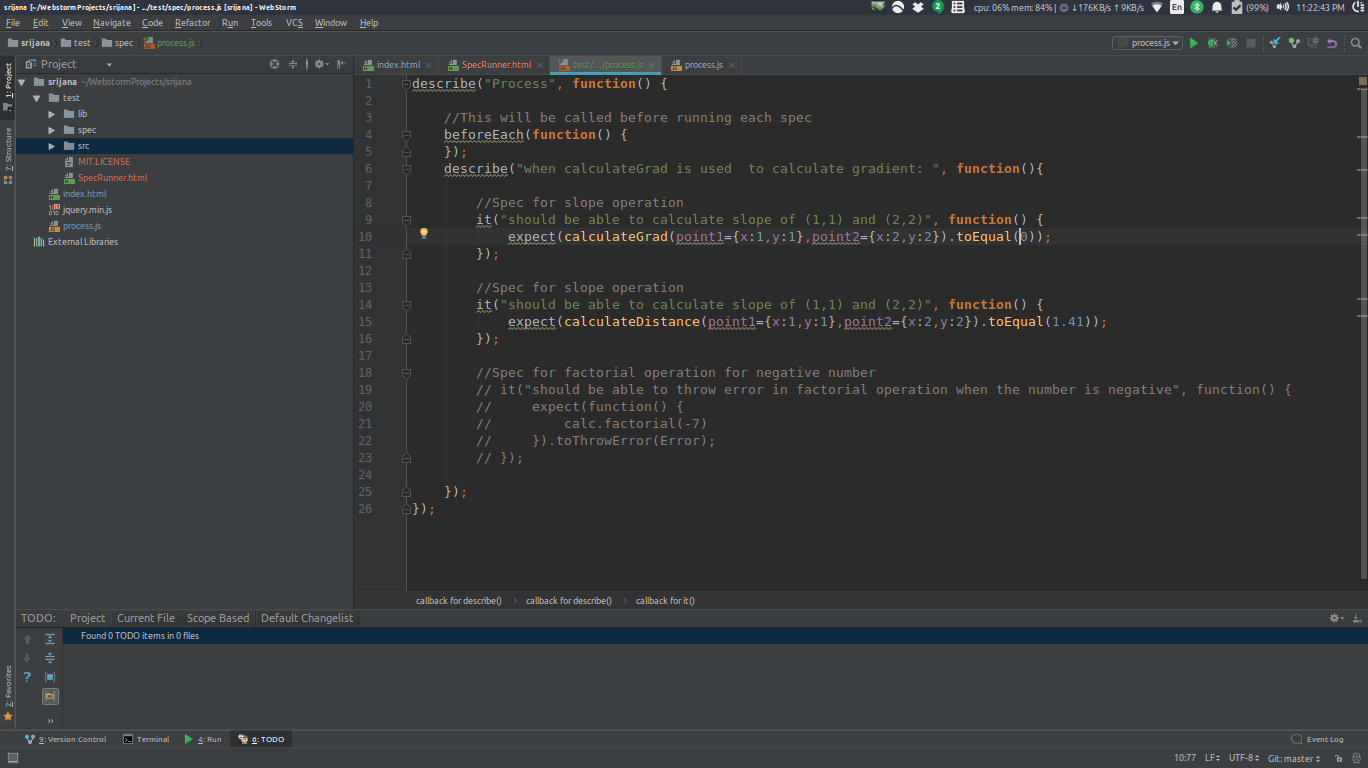
The test was failed when wrong input point was given to the function.





After Refactoring the code, the function passed all the individual unit tests as shown in the figure below:



****

**Question 2:**

**Below I have listed 5 issues and their related code smells with the solutions to fix them.**

# **Error 1:**

# From common IO: 2.5 ExceptionInInitializerError

# Status: Resolved

# Priority: Major

# Created: 09/May/2017

# Updated: 17/May/2017

# Resolved: 17/May/2017

Description:

In its static block, org.apache.commons.io.Java7Support executes:

ClassLoader cl = Thread.currentThread().getContextClassLoader();

This can be null.

In that case, I believe the class should fall back to using:

Java7Support.class.getClassLoader();

But someone with a better understanding of the security implications should weigh in on that change.

Thus, here the Message box code smell arise. So, the company is resolved this problem.

**Error 2:**

Form common logging: **Wrong sample code in org/apache/commons/logging/package.html (using static and this.class)**

Status: closed

Priority: Minor

Created: 10/April/2003

Updated: 29/December/2009

Resolverd: 29 December/2009

Description:

In the org/apache/commons/logging/package.html file, there is sample code:

public class Foo

{ static Log log = LogFactory.getLog(this.class); ... }

This code cannot be compiled because of using "static" and "this.class" in the   
same statement. Should be change to Foo.class.

Thus, here data clumps code smell arises.

**Error 3:**

From Common Net: SubnetUtils.SubnetInfo.isInRange("0.0.0.0") returns true for CIDR/31, 32

Status: Resolved

Priority: Minor

Created: 09/July/2017

Updated: 04/August/2017

Resolved: 04/August/2017

Description:

Code:  
import org.apache.commons.net.util.SubnetUtils;

public class A {  
public static void main(String[] args)

{

System.out.println(new SubnetUtils("192.168.1.0/30").getInfo().isInRange("0.0.0.0")); System.out.println(new SubnetUtils("192.168.1.0/31").getInfo().isInRange("0.0.0.0")); System.out.println(new SubnetUtils("192.168.1.0/32").getInfo().isInRange("0.0.0.0")); }

}

Result:  
false  
true  
true

Expected:  
false  
false  
false

Thus, the Duplicate code smell arises here which can be resolved.

**Error 4:**

From common logging 147: SimpleLog.log - unsafe update of shortLogName

Status: Closed

Priority: Major

Created: 18/july/2012

Updated: 20/March/2012

Resolved: 18/July/2012

Problem:

switch(type)

{

case SimpleLog.LOG\_LEVEL\_TRACE: buf.append("[TRACE] ");

break;

case SimpleLog.LOG\_LEVEL\_DEError: buf.append("[DEError] ");

break;

case SimpleLog.LOG\_LEVEL\_INFO: buf.append("[INFO] ");

break;

case SimpleLog.LOG\_LEVEL\_WARN: buf.append("[WARN] ");

break

case SimpleLog.LOG\_LEVEL\_ERROR: buf.append("[ERROR] ");

break;

case SimpleLog.LOG\_LEVEL\_FATAL: buf.append("[FATAL] ");

break;

}

Here, the data clumps code smell arises.

**Error 5:**From Common Logging: RandomStringUtils' random method infinite loop 1 of 162

Status: Closed

Resolution: Invalid

Created: 11/Nov/2016

Updated: 11/Nov/2016

Resolved: 11/Nov/2016

Code error:

RandomStringUtils.random(1, 0, 0, false, true, new char[]

{ 'a' }

)

The above code has infinite loops and lots of parameters. So, this code is considered as Message Chain smell.

References

1. FOWLER, M., BECK, K., BRANT, J., OPDYKE, W. & ROBERTS, D. (1999) *Refactoring: Improving the Design of Existing Code*, Addison Wesley.
2. Fowler, M. (2006). *bliki: CodeSmell*. [online] martinfowler.com. Available at: <https://martinfowler.com/bliki/CodeSmell.html>.
3. Refactoring.guru. (2014). *Data Clumps*. [online] Available at: <https://refactoring.guru/smells/data-clumps>.
4. Refactoring.guru. (2014). *Message Chains*. [online] Available at: <https://refactoring.guru/smells/message-chains>.
5. Refactoring.guru. (2014). *Remove Middle Man*. [online] Available at: <https://refactoring.guru/remove-middle-man>.
6. Different kinds of code smells and how to refactor them,
7. <http://www.testingeducation.org/pt/Refactoring-smells.pdf>
8. How to approach code smells, journal of software: evolution and process published by john wiley & sons Ltd
9. <https://onlinelibrary.wiley.com/doi/full/10.1002/smr.1886>
10. What are the code smells and different what are the different approaches to clean that code smell
11. <https://github.com/lee-dohm/code-smells>
12. Refactoring to patterns by Joshua kerievsky, Chapter 4 code smells
13. <https://www.safaribooksonline.com/library/view/refactoring-to-patterns/0321213351/ch04.html>
14. Gradient, distance formulas for idea.

<https://www.mathsaccelerator.com/algebra/gradient>