Part 1: Test Cases:

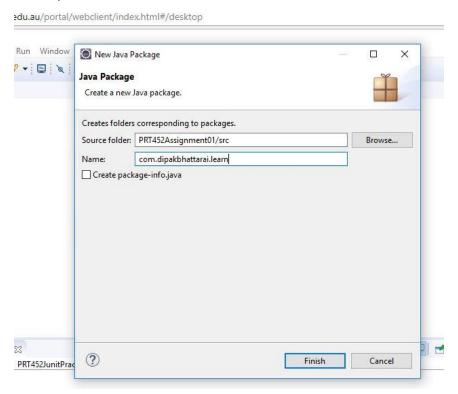
TDD Test Cases for gradient, distance and equation of line between two points:

S.N	Test Case Description	Input				Expected Output			Remarks
		X1	Y1	X2	Y2	Gradient (y2-y1)/ (x2- x1)	Distance $\sqrt{(x2-x1)^2}$ $-(y2-y1)^2$	Equation y-y1=m(x-x1) or y=mx +c	
1	User enters the coordinates A(0,0), B(0,0)	0	0	0	0	0	0	0	
2	User enters the coordinates A(1,1), B(1,1)	1	1	1	1	unidentified	0	Y=0	
3	User enters the coordinates A(-2,1), B(-1,1)	-2	1	-1	1	1	0	Y=1	
4	User enters the coordinates A(4,3), B(6,7)	4	3	6	7	2	√20	Y=2x - 5	
6	User enters the coordinates A(0,0), B(0,0)	2	4	4	6	1	√8	y = x + 2	

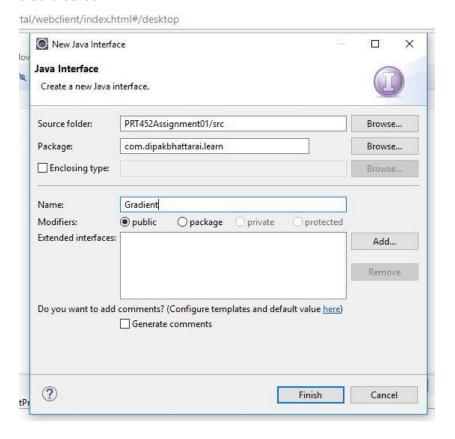
PRT452 - Software Engineering: Process and Tools

Part 2: Program developing screenshots:

Step 1: The project PRT452Assignment01 has been created and the package com.dipakbhattarai.learn is created.



Step 2: The Interface Gradient is created. Similarly, the interfaces Distance and Equation are also created.



PRT452 – Software Engineering: Process and Tools

Step 3: The interfaces are created with the methods with their respective parameters.

Interface1: Gradient and its method with parameters.

m/dipakbhattarai/learn/Gradient.java - Eclipse

```
Project Run Window Help

| Project Run Window He
```

Interface2: Distance and its method with parameters.

m/dipakbhattarai/learn/Distance.java - Eclipse

```
Project Run Window Help

Gradient.java

Distance.java

Distance.java

Distance.java

Distance.java

Mathematics.java

package com.dipakbhattarai.learn;

public interface Distance {

double distance (double x1, double y1, double x2, double y2);

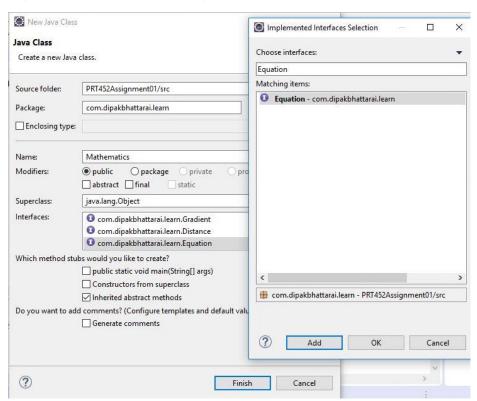
}
```

Interface3: Equation and its method with parameters.

uation.java - Eclipse

PRT452 – Software Engineering: Process and Tools

Steps 4: The java file "Mathematics.java" is created and the three interfaces are also implemented as shown in the picture:



Output: The class Mathematics.java is created with its methods implemented from the interfaces

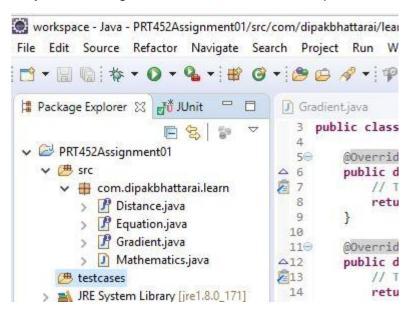
```
n/Mathematics.java - Eclipse
ndow Help

☑ Gradient.java

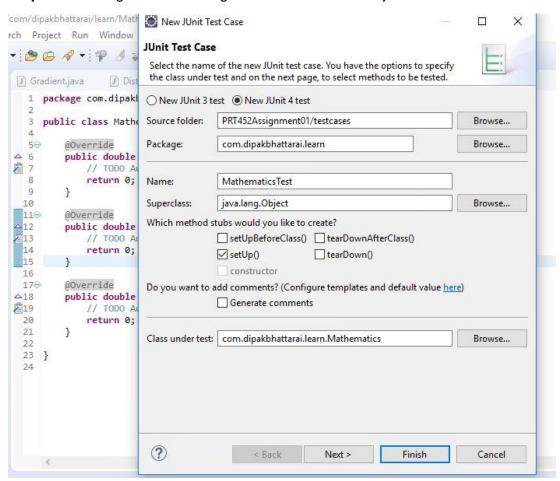
                              J Distance.java
                                             Equation.java
                package com.dipakbhattarai.learn;
                  public class Mathematics implements Gradient, Distance, Equation {
                3
                4
                50
                       @Override
                      public String equation(double x1, double y1, double gradient) {
                6
              1
                7
                          // TODO Auto-generated method stub
                8
                          return null;
                9
                      }
               10
               119
                      @Override
              △12
                      public double distance(double x1, double y1, double x2, double y2) {
              213
                          // TODO Auto-generated method stub
               14
                          return 0;
               15
                      }
               16
               17⊖
                       @Override
                       public double gradient(double x1, double y1, double x2, double y2) {
              418
               19
                          return 0;
               20
               21
               22
               23
```

PRT452 - Software Engineering: Process and Tools

Steps 5: Creating folder testcases in order to separate the test file.



Steps 6: Creating the JuntTesting of the file Mathematics.java



PRT452 - Software Engineering: Process and Tools

The follow picture shows the JunitTesting of the methods gradient, distance and equation:

```
arch Project Run Window Help
☑ Gradient.java
☑ Distance.java

☑ Equation.java ☑ Mathematics.java ☑ MathematicsTest.java ☒
   2 .
    3⊕ import static org.junit.Assert.*;[]
   8 public class MathematicsTest {
         @Before
  100
  11
         public void setUp() throws Exception {
  12
  13
  140
         @Test
   15
         public void test() {
  16
           fail("Not yet implemented");
  17
   18
  199
   20
         public void testGradient(){
   21
           fail("Not yet implemented");
   22
   23
  240
         public void testDistance(){
  25
           fail("Not yet implemented");
   26
   27
   28
  29⊝
   30
         public void testEquation(){
   31
           fail("Not yet implemented");
  32
   33
   34 }
```

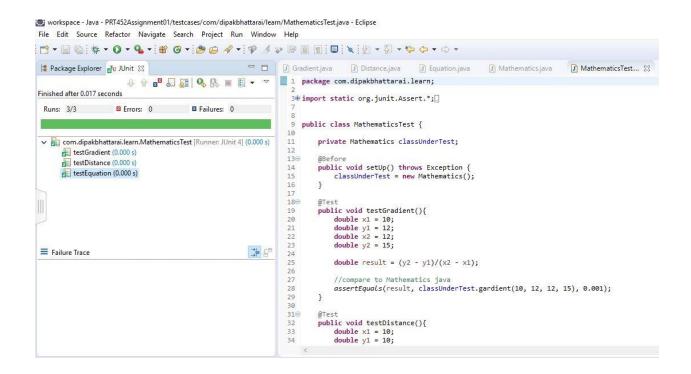
Steps 7: The testing fail showing the unmatched result.

```
🏽 workspace - Java - PRT452Assignment01/testcases/com/dipakbhattarai/learn/MathematicsTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
☐ Package Explorer ☑ JUnit 🎛
                                                - -
                                                         @Before
public void setUp() throws Exception {
Finished after 0.021 seconds
                                                                     classUnderTest = new Mathematics();
 Runs: 3/3 Errors: 0 Failures: 3

▼ iii com.dipakbhattarai.learn.MathematicsTest [Runner: JUnit 4] (0.000 s)

                                                                 public void testGradient(){
                                                                     double x1 = 10;
double y1 = 12;
double x2 = 12;
     testGradient (0.000 s)
      testDistance (0.000 s)
     testEquation (0.000 s)
                                                                     double y2 = 15;
                                                          25
26
27
                                                                     double result = (y2 - y1)/(x2 - x1);
                                                                     //compare to Mathematics java
                                                          28
29
30
31
                                                                     assertEquals(result, classUnderTest.gardient(10, 12, 12, 15), 0.001);
 Failure Trace
                                                                 public void testDistance(){
 Joint Java.lang.AssertionError: expected:<1.5> but was:<0.0>
                                                                    fail("Not yet implemented");
 at com.dipakbhattarai.learn.MathematicsTest.testGradient(MathematicsTe
                                                                 public void testEquation(){
                                                                    fail("Not yet implemented");
```

Steps 8: After implementing the codes the test is finally passed.



Steps 9: The final output of the program when user enters the two points.

```
System.out.println("Enter the x1 coordinate for point 1: ");

x1 = scan.nextInt():

Console 
Console 
Mathematics [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2Sep.,2018, 3:38:09 pm)

Enter the x1 coordinate for point 1:

Enter the y1 coordinate for point 1:

Enter the x2 coordinate for point 2:

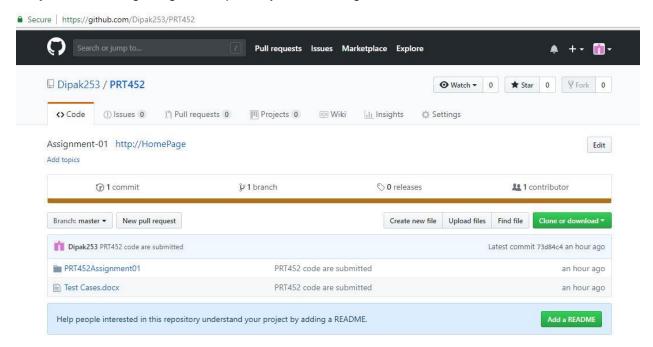
Enter the y2 coordinate for point 2:

| The distance between two points is: 5.656854249492381.

The gradient between two points is: 1.0.

The equation of line between two points is: y =1.0x +2.0.
```

Steps 10: Creating the github repository for the assignment.



Github link: https://github.com/Dipak253/PRT452/tree/master/PRT452Assignment01

Part 3: Code Smells

The five issues related to code smells are identified and the solution for them are discussed below as:

 Lazy Class: Creating a class costs money. If the class which is not doing enough should be eliminated.

Solution:

- Collapse Hierarchy: It will collapse all the subclasses that aren't doing enough.
- Inline Component: The useless components should be subjected using this method.
- 2) Temporary Field: An object needs its variables to be instantiated but sometimes some variables are only set for certain circumstances. Such code is difficult to understand.

Solution:

- **Extract Component**: It helps to collect all the concerns variables in the component.
- ➤ Introduce Null Object: It helps to eliminate conditional code and create an alternative component for invalid variables.

3) Middle Man: Sometimes the internal details of objects are encapsulated.

Solution:

- ➤ **Move Method** and **Move Field**: This methods are used to move features out the middle man into the other objects making the middle man empty.
- **4)** Large Class: In large class, there are number of objects and its variables but all these variables are not be used at all.

Solution:

- > Extract Component: It helps to bundle up the variables and
- > Extract subclass: If the subset of variables are constant then Extract subclass can be used to overcome this problem
- **5) Duplicated code:** It means repetitive use of the code in a program which is not a good practice.

Solution:

Extract Method: This method is used to separate the similar codes and invoke the code from other place.