Regarding TDD KATA

* As a newbie to test-driven course assessment, I was unsure where to begin, so I reviewed the links provided by the company and read the article on unit testing for beginner.
* It introduced me to the red - green refactor principle, in which testers make their code fail on purpose (i.e., red) and then write the minimum amount of code to make that code pass (i.e., green) before improving it/ making it productive (i.e., refractor).
* And this will help in creating a high-quality software that will be easy to maintain.
* I will be using **Java** Language and **NetBeans** **IDE** as an editor for writing testing codes and used testing library.

The steps are:

1)Thinking about which test will move our code towards completion.

[When I want to return 0 for an empty string i will create a method

Calculator calculator=new Calculator();

public void emptyStringReturnsZero()

{

asserts.Equals(calculator.calculator(“ ”),0);

}]

2)**Red:**Then, intentionally writing code that fails(i.e.,

Ex:

public int calculator(String string)

{

return -1;

}

And now check it, observe the test fails.

3)**Green:** Now when the failed:1 is completed, write the code to correct the test case:

public int calculator(String string)

{

return 0;

}

Which is called the green phase. And observe the test passes.Green bar comes.

4)Refractor: And in the refractor phase we write the code in the most productive way by removing the duplicate.

[private Calculator calculator;

public void emptyStringReturnsZero() {

assertEquals(calculator.calculate(""),0);

}]

And in the same way we have to keep finding the test cases.

Some of these I've implemented, but not all. This experience allowed me to gain new knowledge and understanding.