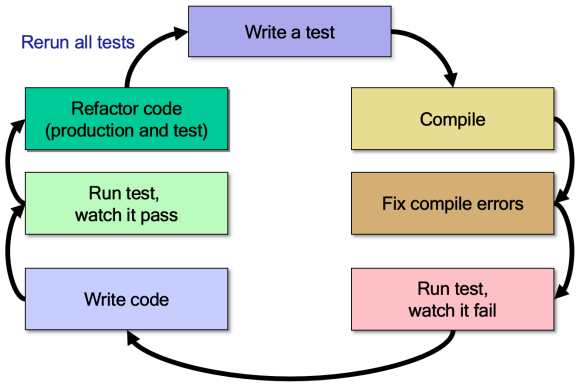
TDD Short Lab

The aim of this lab is to develop a very simple application based on the TDD development cycle.



You should therefore aim to write a test first and then write the code to handle this.

This lab is intended to be run on an individual basis (the next lab will be run in pairs or small groups).

The application will be to create a simple word generate.

FizzBuzz is one of the most famous coding exercises for beginners. It is a simple exercise but an excellent one to start learning the TDD flow with.
Requirements

1. Write a “fizzBuzz” function / method that accepts a number as input and returns a String.
   Notes:
   - start with the minimal failing solution
   - keep the three rules of TDD in mind and always write just sufficient / enough code
   - do not forget to refactor your code after each passing test
   - write your assertions / tests relating to the exact requirements
2. For Zero return “Error”
3. For a negative number return “must be a positive integer”
4. For non-negative numbers return the number as a string
5. For multiples of three return “Fizz”
6. For the multiples of five return “Buzz”
7. For numbers that are multiples of both three and five return “FizzBuzz” (extension point)

You can use whatever programming language you like, in whatever editor you want.

A sample solution is provided in JavaScript.

fizzbuzz.js

```
function fizzbuzz(number) {
    if (number == 0) {
        return "Error";
    } else if (number < 0) {
        return "must be a positive integer";
    } else if (number % 3 == 0 && number % 5 == 0) {
        return "FizzBuzz"
    } else if (number % 3 == 0) {
        // divisiable by 3
        return "Fizz";
    } else if (number % 5 == 0) {
        return "Buzz";
    }
    return number.toString();
}

module.exports = fizzbuzz;
```

fizzbuzz.spec.js

```
const fizzbuzz = require("../fizzbuzz.js");

describe("fizzbuzz function", () => {
  it("should return the string '1' for the integer 1", () => {
    result = fizzbuzz(1);
    expect(result).toEqual("1");
  });

  it("should return the string 'error' for the integer 0", () => {
    result = fizzbuzz(0);
    expect(result).toEqual("Error");
  });

  it("should return the string 'must be a positive integer' for
negative integers", () => {
    result = fizzbuzz(-1);
    expect(result).toEqual("must be a positive integer");
  });

  it("should return the string 'must be a positive integer' for
negative integers", () => {
    result = fizzbuzz(-10);
    expect(result).toEqual("must be a positive integer");
  });

  it("should return the string 'Fizz' for multiples of three - 3",
() => {
```

```
    result = fizzbuzz(3);
    expect(result).toEqual("Fizz");
  });

  it("should return the string 'Fizz' for multiples of three - 6", () => {
    result = fizzbuzz(6);
    expect(result).toEqual("Fizz");
  });

  it("should return the string 'Fizz' for multiples of three - 9", () => {
    result = fizzbuzz(9);
    expect(result).toEqual("Fizz");
  });

  it("should return the string 'Buzz' for multiples of five - 5", () => {
    result = fizzbuzz(5);
    expect(result).toEqual("Buzz");
  });

  it("should return the string 'Buzz' for multiples of five - 10", () => {
    result = fizzbuzz(10);
    expect(result).toEqual("Buzz");
  });

  it("should return the string 'FizzBuzz' for multiples of three and five - 15", ()
=> {
    result = fizzbuzz(15);
    expect(result).toEqual("FizzBuzz");
  });

});
```