МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
„ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ”

ЛАБОРАТОРНА РОБОТА №4

з курсу: *«Проектування інформаційних систем»*
на тему: „ **Модульне тестування (Unit-тести) та рефакторинг.** ”

виконав: студент IV курсу
групи ДА-71
Циолковський Р.Р.

КИЇВ
2020

Мета роботи: оволодіти навичками створення програмного забезпечення за метолологією TDD та ознайомитися з процедурами рефакторинга.

Завдання:

1. Розробити методику випробувань з використанням ISO/IEC/IEEE 29119.

2. Розробити код програми архітектурної моделі. Використовувати Test Driven Development.

3. Провести рефакторинг коду програми, щоб задовольнити вимоги технічного завдання.

Хід роботи:
Створимо тестовий приклад програми, який будемо тестувати – простий калькулятор мовою програмування Python.

```
class Calculator:
    def __init__(self):
        pass

    def add(self, x1, x2):
        return x1 + x2

    def multiply(self, x1, x2):
        return x1 * x2

    def subtract(self, x1, x2):
        return x1 - x2

    def divide(self, x1, x2):
        if x2 != 0:
            return x1 / x2
```

Тепер розробимо тести, що перевіряють функціональність програми

```
import unittest
class TestCalculator(unittest.TestCase):
    def setUp(self):
        self.calculator = Calculator()

    def test_add(self):
        res = self.calculator.add(17, 7)
        self.assertEqual(res, 24)

    def test_subtract(self):
        res = self.calculator.subtract(8, 12)
        self.assertEqual(res, -4)

    def test_multiply(self):
        res = self.calculator.multiply(8, 2)
        self.assertEqual(res, 16)

    def test_divide(self):
        res = self.calculator.divide(68, 4)
        self.assertEqual(res, 17)
```

```
if __name__ == "__main__":
    unittest.main()
```

Запустимо тестування із консолі:

```
C:\Users\HP-Omen 15\PycharmProjects\PizzaBot>python tests.py -v
test_add (__main__.TestCalculator) ... ok
test_divide (__main__.TestCalculator) ... ok
test_multiply (__main__.TestCalculator) ... ok
test_subtract (__main__.TestCalculator) ... ok

----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
```

Unit-тестування та Test Driven Development:

Test Driven Development – техніка розробки програмного забезпечення, яка базується на повторенні коротких циклів розробки:

1. Написання тесту до нової функціональності
2. Написання коду, що реалізує функціонал
3. Якщо код задовольняє результатам тесту, то проводиться рефакторинг – приведення коду до індустріальних стандартів без зміни функціоналу

TDD дуже сильно пов’язане з поняттям модульного тестування – таких тестів, що тестують окремий модуль коду або функціонал (принцип «тестування одного елемента за раз»). За умови якісного написання юніт-тестів TDD забезпечує неперервну роботу програмного продукту навіть за умови додавання нової функціональності: без проходження тестів код не буде використовуватися.