

Міністерство освіти і науки України
Національний Технічний Університет України
«Київський Політехнічний Інститут»
Навчально-науковий комплекс
«Інститут прикладного системного аналізу»
Кафедра системного проектування

Лабораторна робота №5
з дисципліни
«Проектування інформаційних систем»
Модульне тестування (Unit-тести та рефакторинг)

Виконала:
студентка групи ДА-72
Потапова С. С.
Варіант 24

Київ – 2020

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

Задача:

1. Використовувати методологію Test Driven Development для створення класів архітектурної програмної моделі.
2. Скласти тестові сценарії, які продемонструють функціонування всіх методів проектованої моделі.
3. Виконати юніт-тестування складових частин (внутрішніх класів), що реалізують об'єкт моделювання.
4. Виконати "зовнішнє" юніт-тестування для API.
5. Провести рефакторинг коду програми, для поліпшення реалізації.

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

1 Скласти тестові сценарії, які продемонструють функціонування всіх методів проекрованої моделі

1	<ul style="list-style-type: none">- Авторизуватись- Переглянути задачі
2	<ul style="list-style-type: none">- Авторизуватись- Додати задачу- Переконатись у відображенні всіх пунктів доданої задачі- Переглянути задачу
3	<ul style="list-style-type: none">- Авторизуватись- Створити папку- Додати в папку задачі- Перевірити додання задач до папки- Переглянути вміст папки на наявність доданих задач
4	<ul style="list-style-type: none">- Відкрити додаток- Заповнити дані про реєстрацію- Отримати код підтвердження- Перевірити реєстрацію
5	<ul style="list-style-type: none">- Авторизуватись- Виставити налаштування на сповіщення- Перевірити вірне надходження сповіщень

2 Виконати юніт-тестування складових частин (внутрішніх класів), що реалізують об'єкт моделювання

```
py x tests.py x
import unittest
from main import Task

class TDD(unittest.TestCase):
    def test_create_task(self):
        tasks = Task()
        task = {"01":{"name":"to feed the cat", "description":"okay"}}
        task_end = tasks.create(task)
        self.assertDictEqual(task["01"], task_end)

    def test_get_task(self):
        tasks = Task()
        task = {"01":{"name":"to feed the cat", "description":"okay"}}
        task_end = task.get(task)
        self.assertEqual("name":"to feed the cat", "description":"okay", task_end)
```

```
Ran 1 test in 0.004s

OK

Process finished with exit code 0
```

```
Ran 1 test in 0.008s

OK
```

Застосування TDD передбачає виконання трьох наступних кроків:

- написання тесту, що дасть сбій для невеликого фрагменту функціоналу

Створимо тест, де будемо намагатись створити задачу, в тесті до якої будуть порівнюватись задачі, що не можуть бути рівними, причому друга порівнювана задача не має другого мінімально потрібного параметру для створення задачі.

```

class TDD(unittest.TestCase):
    def test_create_task(self):
        tasks = Task()
        task = {"01":{"name":"to feed the cat", "description":"okay"}}
        task2 = {"02":{"name":"to feed the cat"}}
        task_end = tasks.create(task)
        self.assertEqual(task["02"], task_end)

```

- реалізація функціоналу, який приведе до успішного проходження тесту

```

class TDD(unittest.TestCase):
    def test_create_task(self):
        tasks = Task()
        task = {"01":{"name":"to feed the cat", "description":"okay"}}
        task_end = tasks.create(task)
        self.assertEqual(task["01"], task_end)

```

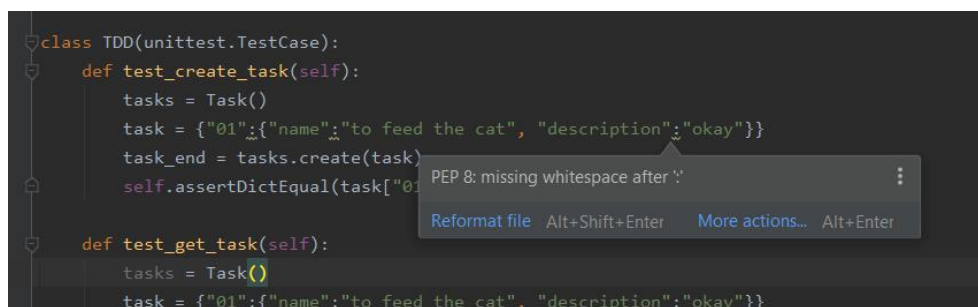
- рефакторинг старого і нового коду для того, щоб підтримувати його в добре структурованому читабельному стані

3 Провести рефакторинг коду програми, щоб задовольнити вимоги технічного завдання

Значна частина рефакторинга присвячується правильному складанню методів. У більшості випадків, коренем усіх зол є занадто довгі методи. Хитросплетіння коду всередині такого методу, ховають логіку виконання і роблять метод вкрай складним для розуміння, а значить і зміни. Та є більше прості задачі рефакторингу, які полягають у створенні простих для читання і розуміння значень змінних і так далі.

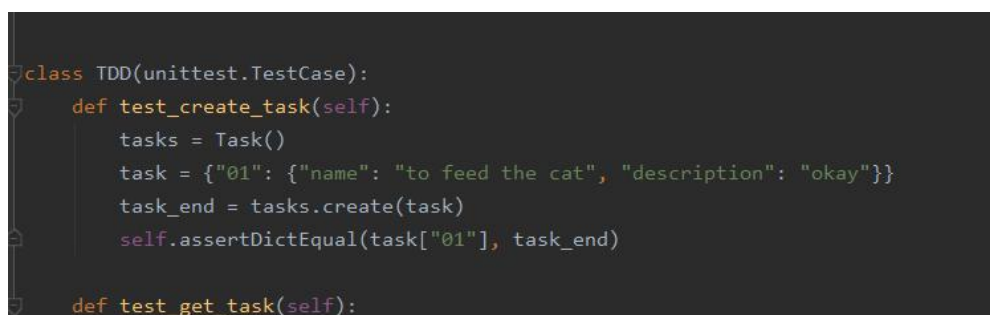
Також є доцільним сказати про PEP8. PEP8 можна визначити, як документ, що описує загальноприйнятий стиль написання коду на мові Python. Python Enhanced Proposal (PEP) - перекладається, як заявки щодо поліпшення мови Python.

Нижче можна побачити типічний приклад автоматичного використання PEP8.



```
class TDD(unittest.TestCase):
    def test_create_task(self):
        tasks = Task()
        task = {"01":{"name":"to feed the cat", "description":"okay"}}
        task_end = tasks.create(task)
        self.assertEqual(task["01"], task_end)

    def test_get_task(self):
        tasks = Task()
        task = {"01":{"name":"to feed the cat", "description":"okay"}}
```



```
class TDD(unittest.TestCase):
    def test_create_task(self):
        tasks = Task()
        task = {"01": {"name": "to feed the cat", "description": "okay"}}
        task_end = tasks.create(task)
        self.assertEqual(task["01"], task_end)

    def test_get_task(self):
```

Приклади рефакторингу

Метод рефакторингу “Витяг методу”

Використовується, коли є фрагмент коду, який можна згрупувати. Для вирішення цього, виділяється частина коду в новий метод чи функцію і викликається саме метод, замість старого коду.

```
def printOwing(self):  
    self.printBanner()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getOutstanding())
```

Рішення:

```
def printOwing(self):  
    self.printBanner()  
    self.printDetails(self.getOutstanding())  
  
def printDetails(self, outstanding):  
    print("name:", self.name)  
    print("amount:", outstanding)
```

Висновки

У ході виконання лабораторної роботи було проведено тестування, за допомогою TDD (Test Drive Development), що полегшує виявлення та усунення помилок у точках основного функціоналу додатку, так було проведено тестування функції отримання задачі згідно введеного ідентифікатора та підтвердження додавання задачі у базу даних порівнюючи результуючий і вхідний словники.

Також було проведено рефакторінг та приведено приклад і опись методу рефакторингу, що може бути застосований та описано тестові сценарії. Проблем з виконанням лабораторної роботи не було.