МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ІПСА

Кафедра Системного проектування

Лабораторна робота 5

з дисципліни: «ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ»

Модульне тестуваня (Unit-тести) та рефакторинг.

Виконав:

студент групи ДА-71

Михайловин Р. Г.

Київ – 2020

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

Код до рефакторингу

```
import json


class Pet:
    def   init  (self, name, w=None, y=None,p=None,f=None,c=None):
        self.name = name
        self.w = w if w else 0
        self.y = y if y else 0
        self.pfc = [p, f, c]
        self.history = []

    def analyze(self):
        pass

    def feed(self,p,f,c):
        self.history.append([p, f, c])

    def show(self):
        print('name: ', self.name, ' w: ', self.w, ' y: ', self.y, ' pfc: ',
self.pfc)

    def tojson(self):
        return json.dumps(self, default=lambda o: o.__dict__)


class Food:
    def __init__(self,name,p,f,c, w=None,left=None):
        self.name = name
        self.w = w if w else 0
        self.left = left if left else w
        self.pfc = [p, f, c]

    def show(self):
        print(self.name, ': w:', self.w, ' left: ', self.left, ' pfc: ',
self.pfc)

    def tojson(self):
        return json.dumps(self, default=lambda o: o.__dict__)


def main():
    with open('pets.json', 'r') as data_file:
        d = json.load(data_file)
    print(d)
    with open('food.json', 'r') as data_file:
        df = json.load(data file)
    print(df)
    pet = 0
    p_list = [Pet('cat'), Pet('dog')]
    meat = Food('meat', 50, 30, 20)
    f_list = [meat]
```

```
    print('0 - choose pet')
    print('1 - pet data')
    print('2 - pet history')
    print('3 - feed pet')
    print('4 - clear pet history')
    print('5 - food data')
    print('6 - add pet')
    print('7 - delete last pet')
    print('9 - exit')
    while 1:
        a = int(input())
        if a == 0:
            pet = int(input())
            print('pet: ', pet)
        elif a == 1:  # read
            p list[pet].show()
        elif a == 2:
            print(p_list[pet].history)
        elif a == 3:  # update
            p_list[pet].history.append(meat.pfc)
        elif a == 4:  # delete
            p_list[pet].history.clear()
        elif a == 5:  # read
            for food in f list:
                food.show()
        elif a == 6:  # create
            name = input('input pet name')
            p_list.append(Pet(name))
        elif a == 7:  # delete
            a = p_list.pop()
            if len(p list) == pet:
                pet -= 1
        elif a == 9:
            j list = [p.tojson() for p in p list]
            print(j_list[0])
            with open('pets.json', 'w') as p file:
                json.dump(j_list, p_file)
            jf_list = [f.tojson() for f in f_list]
            print(jf_list[0])
            with open('food.json', 'w') as f_file:
                json.dump(jf list, f file)

            exit()


if   name   == "  main  ":
    main()
```

Код після рефакторингу

```
import json


class Pet:
    def __init__(self, name, w=None, y=None, p=None, f=None, c=None,
```

```
history=None):
        self.name = name
        self.w = w if w else 0
        self.y = y if y else 0
        self.pfc = [p, f, c]
        self.history = history if history else []

    def analyze(self):
        pass

    def feed(self, pfc):
        self.history.append(pfc)

    def show(self):
        print('name: ', self.name, ' w: ', self.w, ' y: ', self.y, ' pfc: ',
self.pfc)

    def cl(self):
        self.history.clear()

    def tojson(self):
        return json.dumps(self, default=lambda o: o.__dict__)


class Food:
    def   init  (self, name, p, f, c, w=None, left=None):
        self.name = name
        self.w = w if w else 0
        self.left = left if left else self.w
        self.pfc = [p, f, c]

    def show(self):
        print(self.name, ': w:', self.w, ' left: ', self.left, ' pfc: ',
self.pfc)

    def tojson(self):
        return json.dumps(self, default=lambda o: o.__dict__)


def to_json(p_list, f_list):
    j list = [p.tojson() for p in p list]
    print(j_list[0])
    with open('pets.json', 'w') as p_file:
        json.dump(j_list, p_file)
    jf_list = [f.tojson() for f in f_list]
    print(jf list[0])
    with open('food.json', 'w') as f_file:
        json.dump(jf_list, f_file)


def main():
    p_list = []
    with open('pets.json', 'r') as data file:
        d = json.load(data_file)
    print(d)
    for i in d:
        s = json.loads(i)
        p list.append(Pet(s['name'], s['w'], s['y'], s['pfc'][0], s['pfc'][1],
s['pfc'][2], s['history']))
```

```
    with open('food.json', 'r') as data_file:
        df = json.load(data file)
    print(df)
    f_list = []
    for i in df:
        s = json.loads(i)
        f_list.append(Food(s['name'], s['pfc'][0], s['pfc'][1], s['pfc'][2],
s['w'], s['left']))
    pet = 0
    food = 0
    # p_list = [Pet('cat'), Pet('dog')]
    # meat = Food('meat', 50, 30, 20)
    # f_list = [meat]
    print('0 - choose pet')
    print('1 - pet data')
    print('2 - pet history')
    print('3 - feed pet')
    print('4 - clear pet history')
    print('5 - food data')
    print('6 - add pet')
    print('7 - delete last pet')
    print('9 - exit')
    while 1:
        a = int(input())
        if a == 0:
            pet = int(input())
            print('pet: ', pet)
        elif a == 1:  # read
            p list[pet].show()
        elif a == 2:
            print(p list[pet].history)
        elif a == 3:  # update
            p_list[pet].feed(f_list[food].pfc)
        elif a == 4:  # delete
            p_list[pet].cl()
        elif a == 5:  # read
            for food in f_list:
                food.show()
        elif a == 6:  # create
            name = input('input pet name')
            p list.append(Pet(name))
        elif a == 7:  # delete
            a = p_list.pop()
            if len(p_list) == pet:
                pet -= 1
        elif a == 9:
            to_json(p_list, f_list)
            exit()


if   name   == "  main  ":
    main()
```

Що змінилося?

1) Покращено читаємість
   Було:

```
def __init__(self, name,w=None,y=None,p=None,f=None,c=None):
```

Стало:

```
def __init__(self, name, w=None, y=None, p=None, f=None, c=None):
```

2) Збереження у json виведено в окрему функцію

Було:

```
j_list = [p.tojson() for p in p_list]
print(j_list[0])
with open('pets.json', 'w') as p_file:
  json.dump(j_list, p_file)
jf_list = [f.tojson() for f in f_list]
print(jf_list[0])
with open('food.json', 'w') as f_file:
  json.dump(jf_list, f_file)
```

Стало:

```
def to_json(p_list, f_list):
    j_list = [p.tojson() for p in p_list]
    print(j_list[0])
    with open('pets.json', 'w') as p_file:
        json.dump(j_list, p_file)
    jf_list = [f.tojson() for f in f_list]
    print(jf_list[0])
    with open('food.json', 'w') as f_file:
        json.dump(jf_list, f_file)
```

```
to_json(p_list, f_list)
```

Було:

```
        elif a == 3:  # update
            p_list[pet].history.append(meat.pfc)
        elif a == 4:  # delete
            p_list[pet].history.clear()
```

Стало:

```
        elif a == 3:  # update
            p_list[pet].feed(meat.pfc)
        elif a == 4:  # delete
            p_list[pet].cl()
```

Unit-Тести

```
import unittest
import lab4
```

```
class TestPet(unittest.TestCase):
    def setUp(self):
        self.cat = lab4.Pet('cat', 5, 2)
        self.dog = lab4.Pet('dog', 8, 3)
        self.cat.feed(10, 20, 30)

    def test init(self):
        self.assertEqual((self.cat.name, self.cat.w, self.cat.y), ('cat', 5, 2))
        self.assertEqual((self.dog.name, self.dog.w, self.dog.y), ('dog', 8, 3))

    def test_feed(self):
        self.assertEqual(self.cat.history, [[10, 20, 30]])


class TestFood(unittest.TestCase):
    def setUp(self):
        self.meat = lab4.Food('meat', 10, 20, 30)

    def test init(self):
        self.assertEqual((self.meat.name, self.meat.pfc), ('meat', [10, 20,
30]))


if   name   == '  main  ':
    unittest.main()
```

для тестів використовувалася бібліотека unittest

Unit-Тести результати

```
D:\Study\pais\lab4>python -m unittest -v test.py
test_init (test.TestFood) ... ok
test_feed (test.TestPet) ... ok
test_init (test.TestPet) ... ok

----------------------------------------------------------------
Ran 3 tests in 0.002s

OK
```

Висновки: У даній лабораторній роботі оволодів навичками створення програмного забезпечення за метолологією TDD та ознайомитися з процедурами рефакторинга. На мові програмування python ознайомився з бібліотекою unittest та отримав навички в тестуванні коду.