Отчет

Лабораторная работа 4-5

ПКИЯП

Вариант 23

Студент: Цугель А.С.

Группа: ИБМ3-34Б

Преподаватель: Гапанюк Ю.

Отчёт по лабораторной работе

Тема: Модификация кода и разработка модульных тестов (TDD, BDD, Mock)

Задание:

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. В качестве справочника шаблонов можно использовать следующий каталог. Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
2. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк.
    - BDD - фреймворк.
    - Создание Mock-объектов.

1. Выбор фрагмента кода

В качестве основы выбран код из лабораторной работы №1 — программа решения биквадратного уравнения $A \cdot x^4 + B \cdot x^2 + C = 0$.

2. Модификация кода

Код был переработан для возможности модульного тестирования. Логика вычислений вынесена в отдельную чистую функцию, а ввод и вывод параметризованы.

3. TDD-тесты

Для тестирования использован фреймворк unittest. Реализованы тесты для случаев четырёх корней, отсутствия корней и ошибочного ввода.

4. Использование Mock-объектов

Mock-объекты применены для подмены функции вывода print при тестировании main().

5. BDD-тесты

Для BDD-тестирования использован фреймворк behave. Реализованы сценарии в формате Given-When-Then.

6. Вывод

Код успешно модифицирован и протестирован с использованием TDD, BDD и Mock-объектов.

```
import sys, math, subprocess

from pathlib import Path
```

```
bi = """import math,sys
def solve(a,b,c):
   if a==0: raise ValueError
   D=b*b-4*a*c
   y=[]; x=[]
   if D>=0:
       s=math.sqrt(D)
       y1=(-b+s)/(2*a); y2=(-b-s)/(2*a)
       y=[y1,y2]
       for t in (y1,y2):
           if t>0:
               r=math.sqrt(t); x+= [r,-r]
           elif t==0:
               x.append(0.0)
       x=sorted(set(x))
   return {"D":D,"y":y,"x":x}


def main(argv=None, out=print):
   if argv is None: argv=sys.argv[1:]
   a=float(argv[0]); b=float(argv[1]); c=float(argv[2])
   r=solve(a,b,c)
   out("D="+str(r["D"]))
   if not r["y"]: out("no"); return
   out("y1="+str(r["y"][0])); out("y2="+str(r["y"][1]))
   for v in r["x"]: out("x="+str(v))
"""
```

```
tdd = """import unittest
from unittest.mock import Mock
import biquadratic as m


class T(unittest.TestCase):
   def test1(self):

self.assertEqual(m.solve(1,-5,4)["x"],[-2.0,-1.0,1.0,2.0])
   def test2(self):
       self.assertEqual(m.solve(1,0,1)["x"],[])
   def test3(self):
       with self.assertRaises(ValueError):
           m.solve(0,1,1)


class TM(unittest.TestCase):
   def test4(self):
       p=Mock()
       m.main(["1","-5","4"],out=p)
       s="\\n".join(a[0][0] for a in p.call_args_list)
       self.assertIn("D=9.0",s)
       self.assertIn("x=-2.0",s)
       self.assertIn("x=2.0",s)


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

```
feat = """Feature: biquad
 Scenario: four roots
   Given A=1 B=-5 C=4
   When solve
   Then count 4
   And has -2.0,-1.0,1.0,2.0

 Scenario: no roots
   Given A=1 B=0 C=1
   When solve
   Then count 0

 Scenario: two roots
   Given A=1 B=-2 C=1
   When solve
   Then count 2
   And has -1.0,1.0
"""

steps = """from behave import given,when,then
from biquadratic import solve

@given('A={a} B={b} C={c}')
def g(cn,a,b,c):
   cn.a=float(a); cn.b=float(b); cn.c=float(c)

@when('solve')
```

```
def w(cn):
   cn.r=solve(cn.a,cn.b,cn.c)


@then('count {n:d}')
def t(cn,n):
   assert len(cn.r["x"])==n


@then('has {vals}')
def th(cn,vals):
   exp=[float(x.strip()) for x in vals.split(",")]
   for v in exp:
       assert v in cn.r["x"]
"""


def w(p,s):
   p.parent.mkdir(parents=True,exist_ok=True)
   p.write_text(s,encoding="utf-8")


root = Path.cwd()/"lab_onefile"
root.mkdir(exist_ok=True)
w(root/"biquadratic.py",bi)
w(root/"tests"/"test_unit.py",tdd)
w(root/"features"/"biquad.feature",feat)
w(root/"features"/"steps"/"steps.py",steps)


print("unittest:")
```

```
subprocess.run([sys.executable,"-m","unittest","discover","-s",
"tests","-p","test_*.py"],cwd=root)


try:
    import behave
except Exception:

subprocess.run([sys.executable,"-m","pip","install","behave"])


print("\nbehave:")
subprocess.run([sys.executable,"-m","behave"],cwd=root)
```

Примеры выполнения кода:

```
C:\Users\artsu\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\artsu\PycharmProjects\PythonProject\4lab.py
unittest:
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK

behave:
USING RUNNER: behave.runner:Runner
Feature: biquad # features/biquad.feature:1

  Scenario: four roots        # features/biquad.feature:2
    Given A=1 B=-5 C=4        # features/steps/steps.py:4
    When solve                # features/steps/steps.py:8
    Then count 4              # features/steps/steps.py:12
    And has -2.0,-1.0,1.0,2.0 # features/steps/steps.py:16

  Scenario: no roots  # features/biquad.feature:8
    Given A=1 B=0 C=1 # features/steps/steps.py:4
    When solve        # features/steps/steps.py:8
    Then count 0      # features/steps/steps.py:12

  Scenario: two roots  # features/biquad.feature:13
    Given A=1 B=-2 C=1 # features/steps/steps.py:4
    When solve         # features/steps/steps.py:8
    Then count 2       # features/steps/steps.py:12
    And has -1.0,1.0   # features/steps/steps.py:16
```

```
1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
11 steps passed, 0 failed, 0 skipped
Took 0min 0.006s

Process finished with exit code 0
```