|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ |
| УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ |
| «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» |
|  |
| Институт информационных технологий и управления в технических системах |
| (полное название института) |
|  |
| кафедра «Информационные системы» |
| (полное название кафедры) |

**ОТЧЁТ**

по лабораторной работе №4

на тему«Исследование способов интеграционного тестирования программного обеспечения»

по дисциплине **«**Тестирование программного обеспечения»

Вариант 22

Выполнил

студент группы ПИ/б-18-1-о

Маринин Игорь

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | | | | | | |
|  |  | (должность, учёная степень преподавателя) | | | | | | | | |
|  |  |  | | | | | | | | |
|  |  | (ФИО преподавателя) | | | | | | | | |
|  |  | « |  | » |  |  |  | 20 | 20 | г. |
|  |  |  | | | | | | | | |
|  |  | (оценка) | | | | | | | | |
|  |  |  | | | | | | | | |
|  |  |  | | | | | | | | |

Севастополь 2020

* 1. **Цель работы**

Исследовать основные принципы интеграционного тестирования программного обеспечения. Приобрести практические навыки организации интеграционных тестов для объектно-ориентированных программ.

* 1. **Постановка задачи**

Необходимо провести тестирование интеграционным способом. Для этого необходимо выбрать классы, между которыми существует какой-либо тип связи. В данном случае это класс Multiply и класс RectangleArea. В результате выполнения взаимодействия этих классов должна быть вычислена площадь прямоугольника.

* 1. **Спецификация тестового случая**
     1. Проверим работу программы на рабочем диапазоне данных.
     2. Проверим работу программы с нулевыми длинами границ.
     3. Проверим работу программы с длинами отрицательной величины.
  2. **Текст программы**

**main.py**

**class** **Multiply(object):**

'''

Этот класс организует возможность перемножения двух заданных значений типа int (x и y).

Их можно задать либо при создании объекта, либо обратившись к полям напрямую.

'''

x **=** **None**

y **=** **None**

**def** \_\_init\_\_**(**self**,** x**:int** **=** **None,** y**:int** **=** **None):**

self**.**x **=** x

self**.**y **=** y

**def** do**(**self**):**

'''

Этот метод умножает значение 'x' на значение 'y'

Результат: Результат перемножения переменых 'x' и 'y'

Возвращаемый тип: int

'''

# Проверка на существование значений

**if((**self**.**x **is** **None)** **or** **(**self**.**y **is** **None)):**

**raise** **ValueError(**"Пожалуйста, вначале укажите значения 'x' и 'y'"**)**

# Если одно из значений равно нулю, то и результат будет равен нулю

**if((**self**.**x **==** 0**)** **or** **(**self**.**y **==** 0**)):**

**return** 0

result **=** 0

# Запатентованная формула перемножения нескольких переменныйх типа int

**for** i **in** **range(**self**.**y**):**

result **+=** self**.**x

**return** result

**class** **RectangleArea(object):**

'''

Этот класс высчитывает площадь прямоугольника по заданным значениям (длины сторон) типа int (a и b).

Их можно задать либо при создании объекта, либо обратившись к полям напрямую

'''

a **=** **None**

b **=** **None**

**def** \_\_init\_\_**(**self**,** a**:int** **=** **None,** b**:int** **=** **None):**

self**.**a **=** a

self**.**b **=** b

**def** do**(**self**):**

'''

Этот метод высчитывает площадь прямоугольника по двум заданным значениям типа int (длинам сторон).

Результат: Площадь прямоугольника

Возвращаемый тип: int

'''

# Проверка на существование значений

**if((**self**.**a **is** **None)** **or** **(**self**.**b **is** **None)):**

**raise** **ValueError(**"Пожалуйста, вначале укажите длины сторон 'a' и 'b'"**)**

# Проверка на правильность ввода значений

**if(** **(**self**.**a **<** 0**)** **or** **(**self**.**b **<** 0**)** **):**

**raise** **ValueError(**"Введены неверные данные! Длины сторон не могут быть отрицательными"**)**

**elif** **(** **(**self**.**a **==** 0**)** **or** **(**self**.**b **==** 0**)** **):**

**raise** **ValueError(**"Введены неверные данные! Длины сторон не могут иметь нулевую длину"**)**

# Производим математическое вычисление площади прямоугольника по заданным значениям

patentedMultiply **=** Multiply**(**self**.**a**,** self**.**b**)**

**return** patentedMultiply**.**do**()**

**tests.py**

**import** os

**import** main

**import** unittest

**class** **testlab4\_IntegrateTests(**unittest**.**TestCase**):**

**def** testNormalTests**(**self**):**

inputVals **=** **[[**5**,** 3**],** **[**1**,** 1**],** **[**7**,** 1**],** **[**2**,** 2**],** **[**58**,** 17**]]**

expected **=** **[**15**,** 1**,** 7**,** 4**,** 986**]**

testingClass **=** main**.**RectangleArea**()**

actual **=** **[]**

**print()**

**for** testCase **in** inputVals**:**

testingClass**.**a **=** testCase**[**0**]**

testingClass**.**b **=** testCase**[**1**]**

result **=** testingClass**.**do**()**

**print(str(**testCase**[**0**])+**' \* '**+str(**testCase**[**1**])+**' = '**+str(**result**))**

actual**.**append**(**result**)**

self**.**assertEqual**(**expected**,** actual**,**

f'Expected: {expected}\nGot: {actual}'**)**

**def** testZeroLength**(**self**):**

testingClass **=** main**.**RectangleArea**()**

testingClass**.**a **=** 0

testingClass**.**b **=** 0

expected **=** **ValueError**

self**.**assertRaises**(**expected**,** testingClass**.**do**)**

**def** testNegativeValue**(**self**):**

testingClass **=** main**.**RectangleArea**()**

testingClass**.**a **=** **-**5

testingClass**.**b **=** **-**10

expected **=** **ValueError**

self**.**assertRaises**(**expected**,** testingClass**.**do**)**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

logFile **=** 'log\_file.txt'

**with** **open(**logFile**,** 'w'**)** **as** f**:**

runner **=** unittest**.**TextTestRunner**(**f**)**

unittest**.**main**(**testRunner **=** runner**)**

* 1. **Тестирование**
     1. Произведём запуск тестирования, обратив внимание на его результаты (рисунок 4.1).

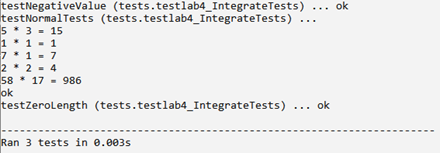


Рисунок 4.1 – Тестовый пример

**Вывод**

В ходе выполнения данной лабораторной работы были изучены основные принципы тестирования интеграционным методом. В основу этого метода положено тестирование (проверка) корректности реализации взаимодействия и связей между классами.

Таким образом были протестированы два класса (Multiply и RectangleArea). Для тестирования использовался метод do(), который производит вычисление. В результате проведённого тестирования можно сделать вывод о том, что программа составлена верно с точки зрения связей и зависимостей между классами.