|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №2**

по дисциплине

«**Тестирование и верификация программного обеспечения**»

**Тема: «Модульное тестирование»»**

|  |  |
| --- | --- |
| **Выполнили:**  Студенты группыИКБО-33-22 | Шило Юрий Сергеевич |
| **Проверил:** | ассистент Петрова А.А. |

**Этап №1. Разработка документации модуля программы.**

Программный продукт состоит из двух основных классов: Ui и Calculator. Каждый класс реализует отдельные функциональные блоки.

1. **Класс Ui** реализует графический интерфейс программы. В нем так же описана простая логика приложения, не требующая математических расчетов или выделения памяти для сохранения каких-либо значений;
2. **Класс Calculator** описывает вычисления внутри приложения, так же занимается сохранением и изменением значений вычислений и памяти для них.

**Алгоритм работы**:

1. Ui получает от пользователя число или вычисление и передает в класс Calculator для выполнения действия, запрошенного пользователем;
2. Calculator выполняет вычисления и возвращает результат классу Ui;
3. Ui отображает результат в графическом интерфейсе.

**Ui**

Реализует графический интерфейс программы Цезаря. Класс содержит в себе следующие методы:

* **Конструктор** — конструктор класса, описывающий инициализацию графического интерфейса с помощью PyQt5. Так же в конструктор прикрепляет к кнопкам соответствующие им символы;
* **calc** — функция, передающая значения в объект класса Calculator для проведения вычислений;
* **plusMinus –** меняет значение текущего числа с минуса на плюс;
* **clearLine** – очищает текущий результат или вычисление в строке;
* **showLastResult –** возвращает последний результат, посчитанный классом Calculator;
* **addNumber –** метод, проверяющий корректность ввода вычисления.

**Calculator**

Описывает вычисления внутри приложения, так же занимается сохранением и изменением значений вычислений и памяти для них.

В классе описаны следующие поля:

* **\_\_lastResult –** поле содержащие последний результат вычислений;
* **\_\_currentResult –** поле содержащие текущий результат вычислений.

В классе описаны следующие методы:

* **Конструктор -** конструктор класса, где задаются необходимые параметры: текущий результат и прошлый результат;
* **setLastResult –** метод необходимый для соблюдения инкапсуляции. Представляет собой сеттер для поля \_\_lastResult.
* **getLastResult –** метод необходимый для соблюдения инкапсуляции. Представляет собой геттер для поля \_\_lastResult.
* **calculate -** метод для нужный для проведения расчетов по полученной строке с помощью встроенной функции eval().
* **getCurrentResult -** метод необходимый для соблюдения инкапсуляции. Представляет собой геттер для поля \_\_currentResult

**Этап №2. Тестирование ПО.**

Для выполнения данного этапа, мной были написаны Unit-тесты. При выполнении которых один из тестов не был пройден. Прохождение тестов показана на рисунке 1.

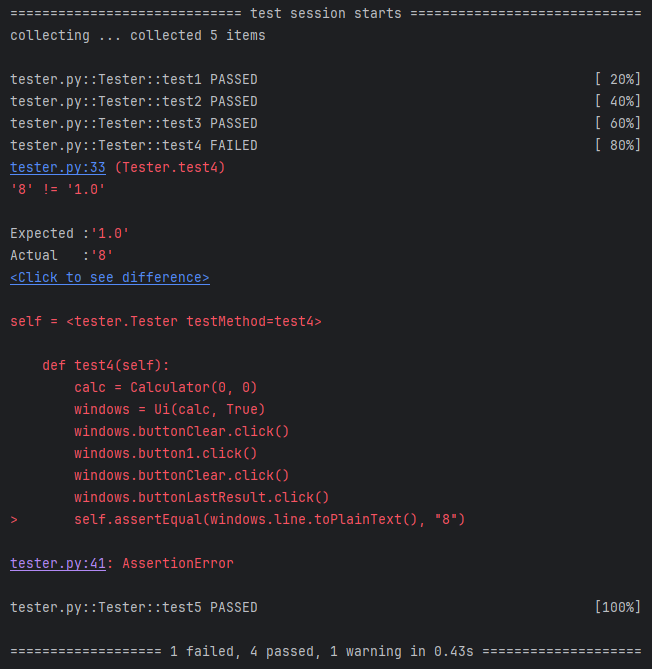


Рисунок 1 - Прохождение тестов с одной ошибкой

**Этап №3. Исправление ошибки.**

**Краткое описание ошибки**: «Некорректный показ последнего результата».

**Статус ошибки**: открыта («Open»).

**Категория ошибки**: серьезная («Major»).

**Тестовый случай**: «Проверка алгоритма функционирования

программы».

**Описание ошибки**:

1. Загрузить программу.
2. В поле ввода ввести 47
3. Нажать кнопку очистки
4. Нажать кнопку последнего результата
5. Полученный результат: 47.

**Ожидаемый результат**: 8.

**Этап №5. Итоговое тестирование.**

После отправки тикеты с описанием ошибки. Код программы был исправлен. После исправления все Unit-тесты были успешно пройдены. Прохождение тестов показана на рисунке 2.

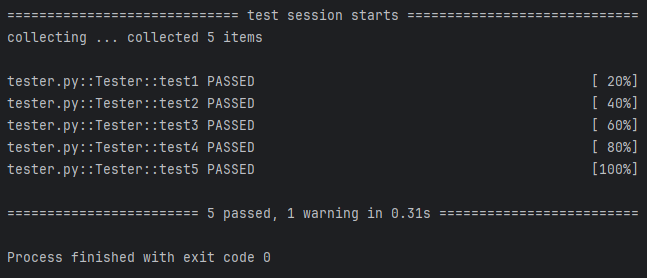
****

Рисунок 2 – Успешное прохождение всех тестов

**Вывод**

В результате выполнения данной практической работы нами были получены навыки работы и написания Unit-тестов.

**Приложение**

Приложение 1 - Unit-тесты

import colorama

import unittest

from imports import \*

app = QtWidgets.QApplication(sys.argv)

class Tester(unittest.TestCase):

def test1(self):

calc = Calculator(0,0)

windows = Ui(calc,True)

windows.button1.click()

windows.button2.click()

windows.button3.click()

windows.button4.click()

windows.button5.click()

windows.button6.click()

windows.button7.click()

windows.button8.click()

windows.button9.click()

self.assertEqual(windows.line.toPlainText(), "123456789")

def test2(self):

calc = Calculator(0, 0)

windows = Ui(calc, True)

windows.line.setText("------89+81")

windows.buttonResult.click()

self.assertEqual(windows.line.toPlainText(), "170")

def test3(self):

calc = Calculator(0, 0)

windows = Ui(calc, True)

windows.line.setText("(2+2)\*2")

windows.buttonResult.click()

self.assertEqual(windows.line.toPlainText(), "8")

def test4(self):

calc = Calculator(0, 0)

windows = Ui(calc, True)

windows.buttonClear.click()

windows.button1.click()

windows.buttonClear.click()

windows.buttonLastResult.click()

self.assertEqual(windows.line.toPlainText(), "0")

def test5(self):

calc = Calculator(0, 0)

windows = Ui(calc, True)

windows.buttonClear.click()

for i in range(0,100):

windows.buttonMinus.click()

windows.button1.click()

windows.buttonPlusMinus.click()

self.assertEqual(windows.line.toPlainText(), "1")

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()