Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»
Отчет по домашнему заданию

Выполнил:
студент группы ИУ5-31Б
Прошкин Георгий Павлович

Проверил:
преподаватель каф.ИУ5
Гапанюк Юрий Евгеньевич

Москва, 2021 г.

Задание

1. Модифицируйте код лабораторной работы №6 таким образом, чтобы он был пригоден для модульного тестирования.
2. Используя материалы лабораторной работы №4 создайте модульные тесты с применением TDD – фреймворка (2 теста) и BDD – фреймворка (2 теста).

Текст программы

main.py

```
import telebot
from telebot import types

TOKEN = '5051889518:AAHKAC-qk9yhVOMRQOqw-0UX0T7arhhDJiY'
bot = telebot.TeleBot(TOKEN)
global config
config = ['Первое число', 'Второе число', 'Третье число', 'Операции',
'Посчитать выражение']
global cases
cases = ['first', 'second', 'third', 'action', 'result']
global call
call = ''
global actions
actions = ['plus', 'multiplication']


def plus_out_put(config):
    config[4] = float(config[0]) + float(config[1]) + float(config[2])
    return f'{float(config[0])} + {float(config[1])} + {float(config[2])} =
{config[4]}'


def multiplication_out_put(config):
    config[4] = float(config[0]) * float(config[1]) * float(config[2])
    return f'{float(config[0])} * {float(config[1])} * {float(config[2])} =
{config[4]}'


@bot.message_handler(commands='start')
def start(message):
    msg = 'Добро пожаловать'
    markup = types.InlineKeyboardMarkup()
    btn = types.InlineKeyboardButton('Начать работу с калькулятором',
callback_data='work')
    markup.add(btn)
    bot.send_message(message.chat.id, msg, reply_markup=markup)


@bot.callback_query_handler(lambda message: message.data == 'work')
def work(message):
    msg = 'Введите данные'
    markup = types.InlineKeyboardMarkup(row_width=1)
    for i in range(5):
        btn = types.InlineKeyboardButton(f'{config[i]}',
callback_data=f'{cases[i]}')
        markup.add(btn)
```

```
    btn = types.InlineKeyboardButton('Сбросить данные',
callback_data='reset')
    markup.add(btn)
    bot.send_message(message.message.chat.id, msg, reply_markup=markup)


@bot.callback_query_handler(lambda message: message.data == cases[0] or
message.data == cases[1] or message.data == cases[2])
def nums(message):
    global call
    call = cases[cases.index(message.data)]
    msg = 'Введите число'
    bot.send_message(message.message.chat.id, msg)


@bot.callback_query_handler(lambda message: message.data == cases[3])
def action(message):
    msg = 'Выберите операцию'
    markup = types.InlineKeyboardMarkup(row_width=2)
    btn = types.InlineKeyboardButton('+', callback_data='plus')
    btn1 = types.InlineKeyboardButton('*', callback_data='multiplication')
    markup.add(btn, btn1)
    bot.send_message(message.message.chat.id, msg, reply_markup=markup)


@bot.callback_query_handler(lambda message: message.data in actions)
def act(message):
    if message.data == actions[0]:
        config[3] = actions[0]
    elif message.data == actions[1]:
        config[3] = actions[1]
    elif message.data == actions[2]:
        config[3] = actions[2]
    else:
        config[3] = actions[3]
    markup = types.InlineKeyboardMarkup(row_width=1)
    msg = 'Введите данные'
    for i in range(5):
        if not config[i].isdigit() and not config[i] in actions:
                btn = types.InlineKeyboardButton(f'{config[i]}',
callback_data=f'{cases[i]}')
                markup.add(btn)
    btn = types.InlineKeyboardButton('Сбросить данные',
callback_data='reset')
    markup.add(btn)
    bot.send_message(message.message.chat.id, msg, reply_markup=markup)


@bot.callback_query_handler(lambda message: message.data == cases[4])
def result(message):
    if config[0].isdigit() and config[1].isdigit() and config[2].isdigit()
and (config[3] in actions):
        if config[3] == 'plus':
            msg = plus_out_put(config)
        else:
            msg = multiplication_out_put(config)
        markup = types.InlineKeyboardMarkup()
        btn = types.InlineKeyboardButton('Сбросить данные',
callback_data='reset')
        markup.add(btn)
        bot.send_message(message.message.chat.id, msg, reply_markup=markup)
    else:
        msg = 'Недостаточно данных'
        markup = types.InlineKeyboardMarkup()
```

```
        for i in range(5):
            if not config[i].isdigit() and not config[i] in actions:
                btn = types.InlineKeyboardButton(f'{config[i]}',
callback_data=f'{cases[i]}')
                markup.add(btn)
        btn = types.InlineKeyboardButton('Сбросить данные',
callback_data='reset')
        markup.add(btn)
        bot.send_message(message.message.chat.id, msg, reply_markup=markup)


@bot.callback_query_handler(lambda message: message.data == 'reset')
def reset(message):
    msg = 'Данные сброшены'
    global config
    config = ['Первое число', 'Второе число', 'Третье число', 'Операции',
'Посчитать выражение']
    markup = types.InlineKeyboardMarkup()
    btn = types.InlineKeyboardButton('Продолжить', callback_data='work')
    markup.add(btn)
    bot.send_message(message.message.chat.id, msg, reply_markup=markup)


@bot.message_handler(content_types='text')
def text(message):
    if (call == cases[0] or call == cases[1] or call == cases[2]) and
message.text.isdigit():
        if call == cases[0]:
            config[0] = message.text
            bot.send_message(message.chat.id, f'Вы ввели первое число
{float(message.text)}')
        elif call == cases[1]:
            config[1] = message.text
            bot.send_message(message.chat.id, f'Вы ввели второе число
{float(message.text)}')
        elif call == cases[2]:
            config[2] = message.text
            bot.send_message(message.chat.id, f'Вы ввели третье число
{float(message.text)}')
        markup = types.InlineKeyboardMarkup(row_width=1)
        msg = 'Введите данные'
        for i in range(5):
            if not config[i].isdigit() and not config[i] in actions:
                btn = types.InlineKeyboardButton(f'{config[i]}',
callback_data=f'{cases[i]}')
                markup.add(btn)
        btn = types.InlineKeyboardButton('Сбросить данные',
callback_data='reset')
        markup.add(btn)
        bot.send_message(message.chat.id, msg, reply_markup=markup)
    if (call == cases[0] or call == cases[1] or call == cases[2]) and not
message.text.isdigit():
        msg = 'Ошибка, попробуйте еще раз'
        bot.send_message(message.chat.id, msg)

# bot.polling(none_stop=True)
```

tests.py

```
from main import plus_out_put, multiplication_out_put
import unittest
```

```
class Tests(unittest.TestCase):
    def test_plus(self):
        msg = plus_out_put(['1.0', '2.0', '3.0', 'plus', 'Посчитать
выражение'])
        self.assertEqual('1.0 + 2.0 + 3.0 = 6.0', msg)

    def test_multiplication(self):
        msg = multiplication_out_put(['3.0', '4.0', '5.0', 'multiplication',
'Посчитать выражение'])
        self.assertEqual('3.0 * 4.0 * 5.0 = 60.0', msg)
```

stepsPlus.py

```
# -*- coding: utf-8 -*-
import string
from main import *
from behave import given, when, then


@given(u'I have context for plusing: [\'{first}\', \'{second}\', \'{third}\',
\'{action}\', \'{result}\']')
def step_plus(context, first: string, second: string, third: string, action:
string, result: string):
    context.first = first
    context.second = second
    context.third = third
    context.action = action
    context.result = result


@when(u'I call plus_out_put')
def step_plus(context):
    context.msg = plus_out_put([context.first, context.second, context.third,
context.action, context.result])


@then(u'I expect to get message with plusing result: \'{msg}\'')
def step_plus(context, msg: string):
    assert context.msg == msg
```

stepsMultiplication.py

```
# -*- coding: utf-8 -*-
import string
from main import *
from behave import given, when, then


@given(u'I have context for multiplicationing: [\'{first}\', \'{second}\',
\'{third}\', \'{action}\', \'{result}\']')
def step_multiplication(context, first: string, second: string, third:
string, action: string, result: string):
    context.first = first
    context.second = second
    context.third = third
    context.action = action
    context.result = result


@when(u'I call multiplication_out_put')
def step_multiplication(context):
```

```
    context.msg = multiplication_out_put([context.first, context.second,
context.third, context.action, context.result])


@then(u'I expect to get message with multiplicationing result: \'{msg}\'')
def step_multiplication(context, msg: string):
    assert context.msg == msg
```

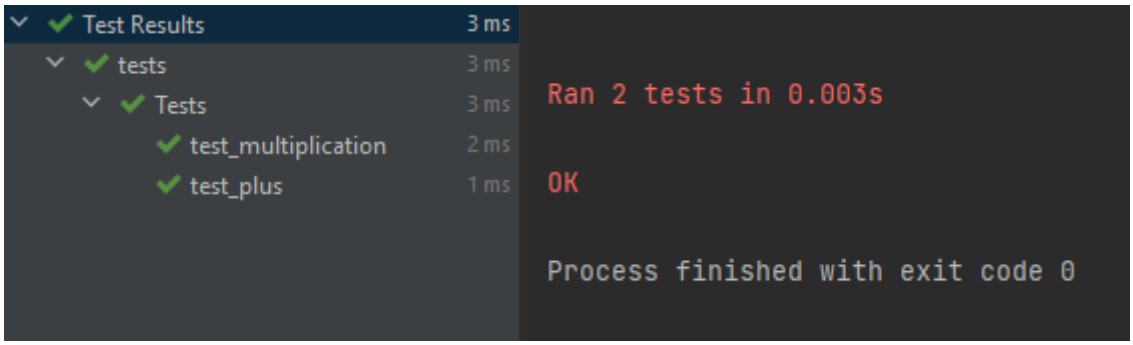
bddPlus.feature

```
Feature: plus
  Scenario: plus 5 and 11 and 15
    Given I have context for plusing: ['5.0', '11.0', '15.0', 'plus',
'Посчитать выражение']
    When I call plus_out_put
    Then I expect to get message with plusing result: '5.0 + 11.0 + 15.0 =
31.0'
```

bddMultiplication.feature

```
Feature: multiplication
  Scenario: multiplication 5 and 11 and 15
    Given I have context for multiplicationing: ['5.0', '11.0', '15.0',
'multiplication', 'Посчитать выражение']
    When I call multiplication_out_put
    Then I expect to get message with multiplicationing result: '5.0 * 11.0 *
15.0 = 825.0'
```

Примеры выполнения программы




```
Feature: multiplication # Features/bddMultiplication.feature:1

  Scenario: multiplication 5 and 11 and 15                                                                  # Features/bddMultiplication.feature:2
    Given I have context for multiplicationing: ['5.0', '11.0', '15.0', 'multiplication', 'Посчитать выражение'] # steps/stepsMultiplication.py:6
    When I call multiplication_out_put                                                                      # steps/stepsMultiplication.py:14
    Then I expect to get message with multiplicationing result: '5.0 * 11.0 * 15.0 = 825.0'                 # steps/stepsMultiplication.py:18

Feature: plus # Features/bddPlus.feature:1

  Scenario: plus 5 and 11 and 15                                                                # Features/bddPlus.feature:2
    Given I have context for plusing: ['5.0', '11.0', '15.0', 'plus', 'Посчитать выражение'] # steps/stepsPlus.py:6
    When I call plus_out_put                                                                  # steps/stepsPlus.py:14
    Then I expect to get message with plusing result: '5.0 + 11.0 + 15.0 = 31.0'              # steps/stepsPlus.py:18

2 features passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
6 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.003s
```