Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ГУИМЦ»
КАФЕДРА ИУ5 «Системы обработки информации и управления»

Дисциплина «Базовые компоненты интернет-технологий»
ОТЧЕТ

Домашнее задание

Студент: Соловьева А.М., группа ИУ5Ц-53Б
Преподаватель: Гапанюк Ю.Е.

2021 г.

СОДЕРЖАНИЕ

Цель лабораторной работы: изучение возможностей создания ботов в Telegram и их тестирования.

1. Описание задания:

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).

2. Листинг программы:

main.py

```
import telebot
import config
import db
import math
import numpy

bot = telebot.TeleBot(config.token)


@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, 'Решение биквадратного уравнения')
    db.set(db.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_FIRST_NUM.value)
    bot.send_message(message.chat.id, 'Введите первое число')



# Обработка первого числа
@bot.message_handler(func=lambda message: db.get(
    db.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_FIRST_NUM.value)
def first_num(message):
    text = message.text
    try:
        float(text)
        if a_equal_zero(text) == '0':
            bot.send_message(message.chat.id, 'Первое число не может быть ноль!')
            bot.send_message(message.chat.id, 'Введите первое число')
            return
        else:
            # Состояние не изменяется, выводится сообщение об ошибке
            # Меняем текущее состояние
            db.set(db.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_SECOND_NUM.value)
            # Сохраняем первое число
            db.set(db.make_key(message.chat.id,
config.States.STATE_FIRST_NUM.value), text)
            bot.send_message(message.chat.id, 'Введите второе число')
    except ValueError:
        # Состояние не изменяется, выводится сообщение об ошибке
        bot.send_message(message.chat.id, 'Пожалуйста, введите число!')
        return
```

```
# Обработка второго числа
@bot.message_handler(func=lambda message: db.get(
    db.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_SECOND_NUM.value)
def second_num(message):
    text = message.text
    try:
        float(text)
        # Меняем текущее состояние
        db.set(db.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_THIRD_NUM.value)
        # Сохраняем второе число
        db.set(db.make_key(message.chat.id, config.States.STATE_SECOND_NUM.value),
text)
        bot.send_message(message.chat.id, 'Введите третье число')

    except ValueError:
        bot.send_message(message.chat.id, 'Пожалуйста, введите число!')
        return


# Обработка третьего числа
@bot.message_handler(func=lambda message: db.get(
    db.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_THIRD_NUM.value)
def third_num(message):
    text = message.text
    try:
        float(text)
        # Сохраняем третье число
        db.set(db.make_key(message.chat.id, config.States.STATE_THIRD_NUM.value),
text)
        # Нахождение корней
        v1 = db.get(db.make_key(message.chat.id,
config.States.STATE_FIRST_NUM.value))
        v2 = db.get(db.make_key(message.chat.id,
config.States.STATE_SECOND_NUM.value))
        v3 = db.get(db.make_key(message.chat.id,
config.States.STATE_THIRD_NUM.value))
        a = float(v1)
        b = float(v2)
        c = float(v3)
        result = []
        D = b * b - 4 * a * c
        if D == 0.0:
            root = -b / (2.0 * a)
            result.append(root)
        elif D > 0.0:
            sqD = math.sqrt(D)
            root1 = (-b + sqD) / (2.0 * a)
            root2 = (-b - sqD) / (2.0 * a)
            result.append(root1)
            result.append(root2)
        resultfinal = []
        for x in result:
            if x > 0:
                resultfinal.append(numpy.sqrt(x))
                resultfinal.append(-numpy.sqrt(x))
            elif x == 0:
                resultfinal.append(0)
        bot.send_message(message.chat.id, "Имеем корни:{}".format(resultfinal))
        # Меняем текущее состояние
        db.set(db.make_key(message.chat.id, config.CURRENT_STATE),
```

```
config.States.STATE_FIRST_NUM.value)
        # Выводим сообщение
        bot.send_message(message.chat.id, 'Введите следующее первое число')
    except ValueError:
        bot.send_message(message.chat.id, 'Введите число!')
        return

def a_equal_zero(text):
    if text == "0":
        return 1
    else:
        return 0

def bikvadrat(a, b, c):
    result = []
    D = b * b - 4 * a * c
    if D == 0.0:
        root = -b / (2.0 * a)
        result.append(root)
    elif D > 0.0:
        sqD = math.sqrt(D)
        root1 = (-b + sqD) / (2.0 * a)
        root2 = (-b - sqD) / (2.0 * a)
        result.append(root1)
        result.append(root2)
    resultfinal = []
    for x in result:
        if x > 0:
            resultfinal.append(numpy.sqrt(x))
            resultfinal.append(-numpy.sqrt(x))
        elif x == 0:
            resultfinal.append(0)
    return resultfinal

if __name__ == '__main__':  # делает нам бесконечный цикл получения данных
    bot.infinity_polling()
```

test.py

```
import unittest
import sys, os

sys.path.append(os.getcwd())
from main import *


class Test_bikvadrat(unittest.TestCase):
    def test_a_equal_zero(self):
        self.assertEqual(a_equal_zero("0"), 1)
        self.assertEqual(a_equal_zero("3242432"), 0)

    def test_result_bikvadrat(self):
        self.assertEqual(bikvadrat(1, -2, 1), [1, -1])
        self.assertEqual(bikvadrat(1, 0, -4), [1.4142135623730951, -
1.4142135623730951])
```

myfeaturesteps.py

```
from behave import *

from tests.test import *


@given('Bot')
def first_step(context):
    context.a = Test_bikvadrat()


@when('test_a_equal_zero return OK')
def check_sneakers(context):
    context.a.test_a_equal_zero()


@when('test_result_bikvadrat return OK')
def check_slates(context):
    context.a.test_result_bikvadrat()


@then('good job')
def last_step(context):
    pass
```

myfeature.feature

```
# -- FILE: features/example.feature
Feature: Showing off behave

  Scenario: Function return message about creation
    Given Bot
    When test_a_equal_zero return OK
    And test_result_bikvadrat return OK
```

config.py

```
from enum import Enum

# Токент бота
token = '5043420848:AAFSQjtCTJCbUFlXrh_NVcSvRKtFCG0Qu0Q'

# Файл базы данных Vedis
db_file = "db.vdb"
```

```
# Ключ записи в БД для текущего состояния
CURRENT_STATE = "CURRENT_STATE"


# Состояния автомата
class States(Enum):
    STATE_START = "STATE_START"  # Начало нового диалога
    STATE_FIRST_NUM = "STATE_FIRST_NUM"
    STATE_SECOND_NUM = "STATE_SECOND_NUM"
    STATE_THIRD_NUM = "STATE_THIRD_NUM"
    STATE_OPERATION = "STATE_OPERATION"
```

db.py

```
from vedis import Vedis
import config


# Чтение значения
def get(key):
    with Vedis(config.db_file) as db:
        try:
            return db[key].decode()
        except KeyError:
            # в случае ошибки значение по умолчанию - начало диалога
            return config.States.S_START.value
```

3. Результат выполнения программы:

```
Ran 2 tests in 0.002s

OK

Process finished with exit code 0
```