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

Домашнее задание

по дисциплине «Базовые компоненты интернет-технологий»

«Изучение возможностей создания ботов в Telegram и их тестирования.

»

Выполнил:

Студент группы ИУ5-34Б

Хатин М.С.

2021 г.

Постановка задачи

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

Файл main.py

```
def operation(fv1, fv2, op):  # рефакторинг функции operation для проверки результата, который отправляет бот
    # Выполняем действие
    res = 0
    if op == 'Перевести':
        res = fv1*fv2
    elif op == 'Стоимость электрогитары Gibson SG в рублях':
        res = 500.0/fv2
    return res


def num_input(text):  # рефакторинг функции ввода числа first_num для проверки правильного изменения состояния
    cond = False
    try:
        a = float(text)
    except ValueError:
        cond = True
    if not cond and a <= 0:
        cond = True

    if cond:
        return 'CURRENT_STATE'  # если ошибка ввода, то не меняем состояние
    else:
        return 'STATE_COURSE'  # если всё хорошо, то переходим к вводу курса
```

Файл test.py

```
import unittest
from main import operation, num_input


true_operation = [3.5, 20_000]  # 140 рублей в долларах при курсе 0.025 и операции 'Перевести'
                                # стоимость электрогитары при курсе 0.025

true_input = ['CURRENT_STATE', 'CURRENT_STATE', 'STATE_COURSE']


class MyTestCase(unittest.TestCase):

    def test_operation(self):
        self.assertEqual(true_operation,\
                         [operation(140, 0.025, 'Перевести'),\
                          operation(20, 0.025, 'Стоимость электрогитары Gibson SG в рублях')])

    def test_input(self):
        self.assertEqual(true_input, [num_input('-10'), num_input('abc23'), num_input('140.2')])


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

Файл dbworker.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


# Запись значения
def set(key, value):
    with Vedis(config.db_file) as db:
        try:
            db[key] = value
            return True
        except:
            # тут желательно как-то обработать ситуацию
            return False


# Создание ключа для записи и чтения
def make_key(chatid, keyid):
    res = str(chatid) + '__' + str(keyid)
    return res
```

Файл config.py

```
from enum import Enum

# Токент бота
TOKEN = "5078527545:AAHq88BwbHGOo2MGbysfrRbZWz5PfMsm47U"

# 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_RUBLE = "STATE_RUBLE"
    STATE_COURSE = "STATE_COURSE"
    STATE_CONV = "STATE_CONV"
```

Файл bot.py

```
import telebot
from telebot import types
import config
import dbworker

# Создание бота
bot = telebot.TeleBot(config.TOKEN)


# Начало диалога
@bot.message_handler(commands=['start'])
def cmd_start(message):
    bot.send_message(message.chat.id, 'Тут мы будем осуществлять перевод рублей в доллары')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_RUBLE.value)
    bot.send_message(message.chat.id, 'Введите рубли:')


# По команде /reset будем сбрасывать состояния, возвращаясь к началу диалога
@bot.message_handler(commands=['reset'])
def cmd_reset(message):
    bot.send_message(message.chat.id, 'Сброс результатов предыдущего ввода.')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_RUBLE.value)
    bot.send_message(message.chat.id, 'Введите рубли:')


# Обработка первого числа
@bot.message_handler(func=lambda message: dbworker.get(
    dbworker.make_key(message.chat.id, config.CURRENT_STATE)) == config.States.STATE_RUBLE.value)
def first_num(message):
    text = message.text

    cond = False
    try:
        a = float(text)
    except ValueError:
        cond = True
    if not cond and a <= 0:
        cond = True

    if cond:
        # Состояние не изменяется, выводится сообщение об ошибке
        bot.send_message(message.chat.id, 'Проверка на дурака :) Введите положительное число:')
    else:
        bot.send_message(message.chat.id, f'Введено количество рублей - {text}')
        # Меняем текущее состояние
        dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_COURSE.value)
        # Сохраняем первое число
        dbworker.set(dbworker.make_key(message.chat.id, config.States.STATE_RUBLE.value), text)
        bot.send_message(message.chat.id, 'Теперь нужен курс доллара к рублю')


# Обработка второго числа
@bot.message_handler(func=lambda message: dbworker.get(
    dbworker.make_key(message.chat.id, config.CURRENT_STATE)) == config.States.STATE_COURSE.value)
def second_num(message):
    text = message.text

    cond = False
    try:
```

```
        a = float(text)
    except ValueError:
        cond = True
    if not cond and a <= 0:
        cond = True

    if cond:
        # Состояние не изменяется, выводится сообщение об ошибке
        bot.send_message(message.chat.id, 'Проверка на дурака :) Введите положительное число:')
    else:
        bot.send_message(message.chat.id, f'Вы ввели курс ({text})')
        # Меняем текущее состояние
        dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_CONV.value)
        # Сохраняем первое число
        dbworker.set(dbworker.make_key(message.chat.id, config.States.STATE_COURSE.value), text)
        markup = types.ReplyKeyboardMarkup(row_width=2)
        itembtn1 = types.KeyboardButton('Перевести')
        itembtn2 = types.KeyboardButton('Стоимость электрогитары Gibson SG в рублях')
        markup.add(itembtn1, itembtn2)
        bot.send_message(message.chat.id, 'Выберите, пожалуйста, действие', reply_markup=markup)

# Выбор действия
@bot.message_handler(func=lambda message: dbworker.get(
    dbworker.make_key(message.chat.id, config.CURRENT_STATE)) == config.States.STATE_CONV.value)
def operation(message):
    # Текущее действие
    op = message.text
    # Читаем операнды из базы данных
    val1 = dbworker.get(dbworker.make_key(message.chat.id, config.States.STATE_RUBLE.value))
    val2 = dbworker.get(dbworker.make_key(message.chat.id, config.States.STATE_COURSE.value))
    # Выполняем действие
    fv1 = float(val1)
    fv2 = float(val2)
    res = 0
    if op == 'Перевести':
        res = fv1*fv2
    elif op == 'Стоимость электрогитары Gibson SG в рублях':
        res = 500.0/fv2
    # Выводим результат
    markup = types.ReplyKeyboardRemove(selective=False)
    if op == 'Перевести':
        bot.send_message(message.chat.id, f'{val1} р. = {res}$', reply_markup=markup)
    else:
        bot.send_message(message.chat.id, f'Гитара стоит примерно {int(res)} рублей', reply_markup=markup)
    # Меняем текущее состояние
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE), config.States.STATE_RUBLE.value)
    # Выводим сообщение
    bot.send_message(message.chat.id, 'Введите рубли')


bot.polling()
```

Результат выполнения программы