



**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 ИУ5-35Б

Титов Е.А.

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):
    # Выполняем действие
    res = 0
    if op == 'Перевести':
        res = fv1*fv2
    elif op == 'Стоимость комбика gibson в евро':
        res = int(500.0/fv2)
    return res

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

    if cond:
        return 'CURRENT_STATE'
    else:
        return 'STATE_COURSE'
```

### Файл test.py

```
import unittest
from main import operation, num_input

true_operation = [600, 416] # 500 долларов в Евро и стоимость комбика в евро при
                           курсе 1.2

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

class MyTestCase(unittest.TestCase):

    def test_operation(self):
        self.assertEqual(true_operation, \
                           [operation(500, 1.2, 'Перевести'), \
                            operation(500, 1.2, 'Стоимость комбика gibson в евро')])

    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 = "5009819170:AAHKKrGMlaGEiGuwCxH03Y_1G6qa8ZGuvwQ"

# 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_EUR = "STATE_EUR"
```

```
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 евро в доллары')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_EUR.value)
    bot.send_message(message.chat.id, 'Введите евро:')

# 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_EUR.value)
    bot.send_message(message.chat.id, 'Введите евро:')

# Обработка первого числа
@bot.message_handler(func=lambda message: dbworker.get(
    dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_EUR.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, ' Введите положительное число:')
    else:
        bot.send_message(message.chat.id, f'Введено количество евро - {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_EUR.value), text)
        bot.send_message(message.chat.id, 'Теперь нужен курс доллара к евро')

# 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, '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 комбика gibson в евро')
        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_EUR.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 комбика gibson в евро':
        res = 500.0/fv2
    # Выводим результат
    markup = types.ReplyKeyboardRemove(selective=False)
    if op == 'Перевести':
        bot.send_message(message.chat.id, f'{val1} евро = {res}$',
reply_markup=markup)
    else:
        bot.send_message(message.chat.id, f'Комбик стоит примерно {int(res)} евро',
reply_markup=markup)
    # Меняем текущее состояние
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_EUR.value)
    # Выводим сообщение
    bot.send_message(message.chat.id, 'Введите евро')

bot.polling()

```

## Результат выполнения программы



После ввода кол-ва евро и курса бот предлагает действие на выбор:

