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»
Отчёт по домашнему заданию

Выполнил:
студент группы ИУ5-31Б
Смыслов Дмитрий Олегович

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись:_____

Дата:_____

Москва, 2021 г.

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).

Текст программы

Бот реализует возможность шифрования текста с помощью шифра Виженера.

Бот реализует конечный автомат из четырёх состояний:

- Начало диалога и выбор действия (зашифровать или расшифровать)
- Выбор алфавита (русский или английский), символы которого будут шифроваться (шифруются только символы выбранного алфавита, остальные не изменяются, регистр сохраняется)
- Ввод текста (должен содержать хотя бы один символ выбранного алфавита)
- Ввод ключа (должен содержать хотя бы один символ выбранного алфавита) и вывод результата

Файл vigenere.py

```
eng_alphabet = "abcdefghijklmnopqrstuvwxyz"
rus_alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"


# Проверка текста и ключа на пригодность
def is_good(text, lng):
    if lng == "Английский":
        for s in text:
            if s.lower() in eng_alphabet:
                return True
    else:
        for s in text:
            if s.lower() in rus_alphabet:
                return True
    return False


# 'Очистка' ключа от лишних символов
def clear_key(key, lng):
    res = []
    if lng == "Английский":
        for s in key.lower():
            if s in eng_alphabet:
                res.append(s)
    else:
        for s in key.lower():
            if s in rus_alphabet:
```

```
                res.append(s)
    return ''.join(res)


# Зашифровка
def encode(text, key, lng):
    res = []
    ck = clear_key(key, lng)
    i = 0
    if lng == "Английский":
        for s in text:
            if s.lower() in eng_alphabet:
                ns = eng_alphabet[(eng_alphabet.find(s.lower()) +
eng_alphabet.find(ck[i])) % 26]
                i = (i+1) % len(ck)
                if s.islower():
                    res.append(ns)
                else:
                    res.append(ns.upper())
            else:
                res.append(s)
    else:
        for s in text:
            if s.lower() in rus_alphabet:
                ns = rus_alphabet[(rus_alphabet.find(s.lower()) +
rus_alphabet.find(ck[i])) % 33]
                i = (i+1) % len(ck)
                if s.islower():
                    res.append(ns)
                else:
                    res.append(ns.upper())
            else:
                res.append(s)
    return ''.join(res)


# Расшифровка
def decode(text, key, lng):
    res = []
    ck = clear_key(key, lng)
    i = 0
    if lng == "Английский":
        for s in text:
            if s.lower() in eng_alphabet:
                ns = eng_alphabet[(eng_alphabet.find(s.lower()) -
eng_alphabet.find(ck[i]) + 26) % 26]
                i = (i + 1) % len(ck)
                if s.islower():
                    res.append(ns)
                else:
                    res.append(ns.upper())
            else:
                res.append(s)
    else:
        for s in text:
            if s.lower() in rus_alphabet:
                ns = rus_alphabet[(rus_alphabet.find(s.lower()) -
rus_alphabet.find(ck[i]) + 33) % 33]
                i = (i + 1) % len(ck)
                if s.islower():
                    res.append(ns)
                else:
                    res.append(ns.upper())
            else:
```

```
                res.append(s)
    return ''.join(res)
```

Файл config.py

```
from enum import Enum

# Токен бота
TOKEN = "***"

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

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

# Ключ записи в БД для выполняемого действия
SELECTED_ACTION = "SELECTED_ACTION"

# Ключ записи в БД для выбранного алфавита
SELECTED_ALPHABET = "SELECTED_ALPHABET"


# Состояния автомата
class States(Enum):
    STATE_ACTION_SELECT = "STATE_ACTION_SELECT"  # Начало диалога и выбор
действия
    STATE_ALPHABET_SELECT = "STATE_ALPHABET_SELECT"
    STATE_TEXT = "STATE_TEXT"
    STATE_KEY = "STATE_KEY"
```

Файл dbworker.py

```
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.STATE_ACTION_SELECT.value


# Запись значения
def set(key, value):
    with Vedis(config.db_file) as db:
        try:
            db[key] = value
            return True
        except:
            return False


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

Файл bot.py

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

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

# Сообщения
mes_encode = "Зашифровать"
mes_decode = "Расшифровать"
mes_eng_alphabet = "Английский"
mes_rus_alphabet = "Русский"


# Начало диалога
@bot.message_handler(commands=['start'])
def cmd_start(message):
    bot.send_message(message.chat.id, 'Я умею шифровать и дешифровать
предложения шифром Виженера!')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_ACTION_SELECT.value)
    # Выводим кнопки выбора
    markup = types.ReplyKeyboardMarkup(row_width=1)
    itembtn1 = types.KeyboardButton(mes_encode)
    itembtn2 = types.KeyboardButton(mes_decode)
    markup.add(itembtn1, itembtn2)
    bot.send_message(message.chat.id, 'Выберите действие',
reply_markup=markup)


# 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, 'Сбрасываем результаты предыдущего
ввода.')
    dbworker.set(dbworker.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_ACTION_SELECT.value)
    # Выводим кнопки выбора
    markup = types.ReplyKeyboardMarkup(row_width=1)
    itembtn1 = types.KeyboardButton(mes_encode)
    itembtn2 = types.KeyboardButton(mes_decode)
    markup.add(itembtn1, itembtn2)
    bot.send_message(message.chat.id, 'Выберите 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_ACTION_SELECT.value)
def action_select(message):
    text = message.text
    if text != mes_encode and text != mes_decode:
        bot.send_message(message.chat.id, 'Выберите действие')
    else:
        # Меняем текущее состояние
        dbworker.set(dbworker.make_key(message.chat.id,
config.CURRENT_STATE), config.States.STATE_ALPHABET_SELECT.value)
        # Сохраняем выбранное действие
        dbworker.set(dbworker.make_key(message.chat.id,
```

```
config.SELECTED_ACTION), text)
        # Выводим кнопки выбора
        markup = types.ReplyKeyboardMarkup(row_width=1,
one_time_keyboard=True)
        itembtn1 = types.KeyboardButton(mes_rus_alphabet)
        itembtn2 = types.KeyboardButton(mes_eng_alphabet)
        markup.add(itembtn1, itembtn2)
        bot.send_message(message.chat.id, 'Выберите алфавит, символы которого
будут шифроваться', reply_markup=markup)


# Выбор алфавита
@bot.message_handler(func=lambda message:
dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_ALPHABET_SELECT.value)
def alphabet_select(message):
    text = message.text
    if text != mes_rus_alphabet and text != mes_eng_alphabet:
        bot.send_message(message.chat.id, 'Выберите алфавит, символы которого
будут шифроваться')
    else:
        # Меняем текущее состояние
        dbworker.set(dbworker.make_key(message.chat.id,
config.CURRENT_STATE), config.States.STATE_TEXT.value)
        # Сохраняем выбранное действие
        dbworker.set(dbworker.make_key(message.chat.id,
config.SELECTED_ALPHABET), text)
        # Запрашиваем ввод исходного текста
        bot.send_message(message.chat.id, 'Введите исходный текст')


# Ввод текста
@bot.message_handler(func=lambda message:
dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_TEXT.value)
def text_input(message):
    text = message.text
    if not vigenere.is_good(text,
dbworker.get(dbworker.make_key(message.chat.id, config.SELECTED_ALPHABET))):
        # В исходном тексте нет подходящих символов для шифровки
        bot.send_message(message.chat.id, "В введённом тексте нет подходящих
символов для шифровки\n"
                                          "Введите другой текст или с помощью
/reset вернитесь в начало")
    else:
        # Меняем текущее состояние
        dbworker.set(dbworker.make_key(message.chat.id,
config.CURRENT_STATE), config.States.STATE_KEY.value)
        # Сохраняем выбранное действие
        dbworker.set(dbworker.make_key(message.chat.id,
config.States.STATE_TEXT.value), text)
        bot.send_message(message.chat.id, 'Введите ключ')


# Ввод ключа
@bot.message_handler(func=lambda message:
dbworker.get(dbworker.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_KEY.value)
def key_input(message):
    key = message.text
    if not vigenere.is_good(key,
dbworker.get(dbworker.make_key(message.chat.id, config.SELECTED_ALPHABET))):
        # В исходном ключе нет подходящих символов для шифровки
        bot.send_message(message.chat.id, "В введённом ключе нет подходящих
```

```
символов для шифровки\n"
                                        "Введите другой ключ или с помощью
/reset вернитесь в начало")
    else:
        # Запоминаем ключ
        dbworker.set(dbworker.make_key(message.chat.id,
config.States.STATE_KEY.value), key)
        # Получаем введенные ранее данные
        text = dbworker.get(dbworker.make_key(message.chat.id,
config.States.STATE_TEXT.value))
        lng = dbworker.get(dbworker.make_key(message.chat.id,
config.SELECTED_ALPHABET))
        action = dbworker.get(dbworker.make_key(message.chat.id,
config.SELECTED_ACTION))
        # Находим результат
        if action == "Зашифровать":
            res = vigenere.encode(text, key, lng)
        else:
            res = vigenere.decode(text, key, lng)
        # Выводим результат
        bot.send_message(message.chat.id, f'Вы ввели:\n'
                                          f'Действие: {action}\n'
                                          f'Алфавит: {lng}\n'
                                          f'Исходный текст: {text}\n'
                                          f'Ключ: {key}\n'
                                          f'Результат: {res}')
        # Меняем текущее состояние
        dbworker.set(dbworker.make_key(message.chat.id,
config.CURRENT_STATE), config.States.STATE_ACTION_SELECT.value)
        # Выводим кнопки выбора
        markup = types.ReplyKeyboardMarkup(row_width=1)
        itembtn1 = types.KeyboardButton(mes_encode)
        itembtn2 = types.KeyboardButton(mes_decode)
        markup.add(itembtn1, itembtn2)
        bot.send_message(message.chat.id, 'Выберите действие',
reply_markup=markup)


if __name__ == '__main__':
    bot.infinity_polling()
```

В качестве TDD фреймворка использовался unittest
Файл TDD_tests.py

```
import unittest
from vigenere import encode, decode


class VigenereTest(unittest.TestCase):

    def test_ru_encode(self):
        res = encode("АбвГДеёж", "аБбГ", "Русский")
        self.assertEqual(res, "АвгЁДёжй")

    def test_eng_decode(self):
        res = decode("Tgoggtdo", "acdc", "Английский")
        self.assertEqual(res, "Telegram")


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

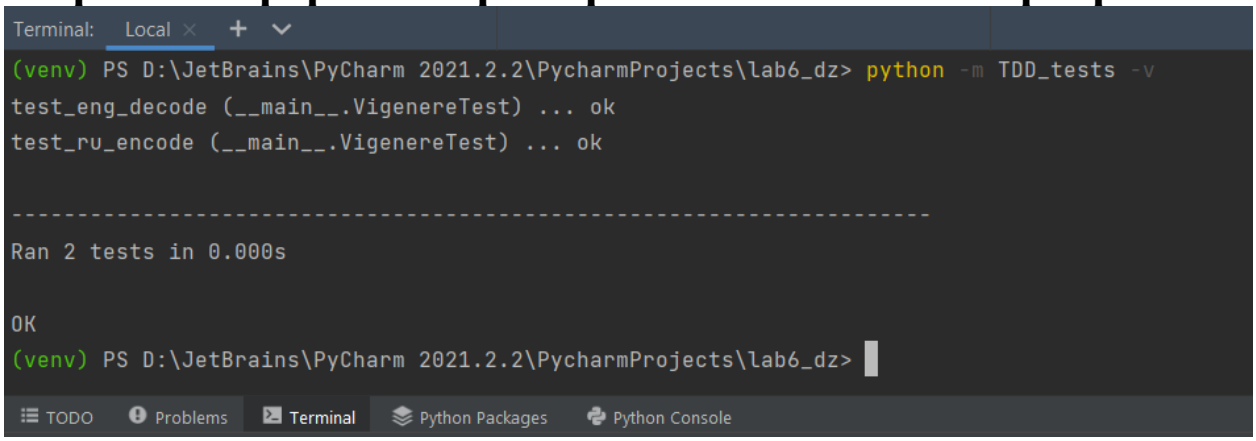
В качестве BDD фреймворка использовался behave
Файл с тестами: vigenere_tests.feature

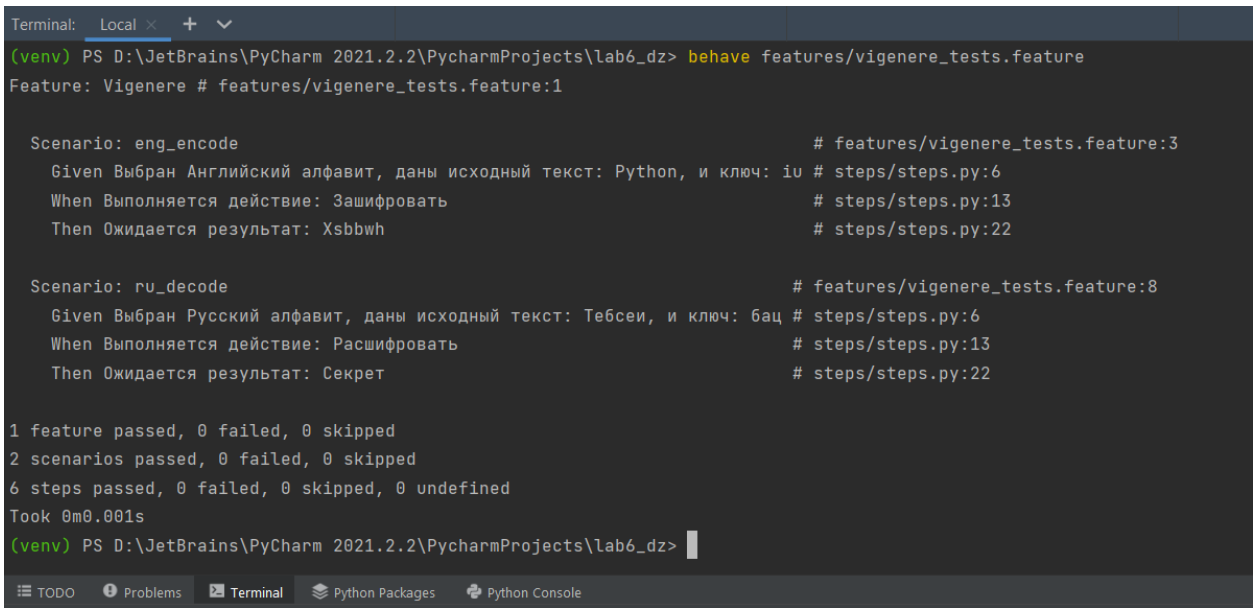
```
Feature: Vigenere

  Scenario: eng_encode
    Given Выбран Английский алфавит, даны исходный текст: Python, и ключ: iu
    When  Выполняется действие: Зашифровать
    Then  Ожидается результат: Xsbbwh

  Scenario: ru_decode
    Given Выбран Русский алфавит, даны исходный текст: Тебсеи, и ключ: бац
    When  Выполняется действие: Расшифровать
    Then  Ожидается результат: Секрет
```

Файл с реализацией шагов теста: steps.py

```
# -*- coding: utf-8 -*-
from behave import given, when, then
from vigenere import encode, decode


@given("Выбран {lng} алфавит, даны исходный текст: {txt}, и ключ: {key}")
def step_impl(context, lng: str, txt: str, key: str):
    context.lng = str(lng)
    context.txt = str(txt)
    context.key = str(key)


@when("Выполняется действие: {act}")
def step_impl(context, act: str):
    context.action = str(act)
    if context.action == "Зашифровать":
        context.res = encode(context.txt, context.key, context.lng)
    elif context.action == "Расшифровать":
        context.res = decode(context.txt, context.key, context.lng)


@then("Ожидается результат: {res}")
def step_impl(context, res: str):
    assert context.res == res
```

Экранные формы с примерами выполнения программы




Terminal: Local
(venv) PS D:\JetBrains\PyCharm 2021.2.2\PycharmProjects\lab6_dz> behave features/vigenere_tests.feature
Feature: Vigenere # features/vigenere_tests.feature:1
Scenario: eng_encode # features/vigenere_tests.feature:3
Given Выбран Английский алфавит, даны исходный текст: Python, и ключ: iu # steps/steps.py:6
When Выполняется действие: Зашифровать # steps/steps.py:13
Then Ожидается результат: Xsbbwh # steps/steps.py:22
Scenario: ru_decode # features/vigenere_tests.feature:8
Given Выбран Русский алфавит, даны исходный текст: Тебсеи, и ключ: бац # steps/steps.py:6
When Выполняется действие: Расшифровать # steps/steps.py:13
Then Ожидается результат: Секрет # steps/steps.py:22
1 feature passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
6 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.001s
(venv) PS D:\JetBrains\PyCharm 2021.2.2\PycharmProjects\lab6_dz>
TODO
Problems
Terminal
Python Packages
Python Console