**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: Проверил:

студент группы ИУ5-31Б преподаватель каф. ИУ5  
Чичикин Тимофей Дмитриевич Гапанюк Юрий Евгеньевич

Подпись и дата: Подпись и 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**

Файл caesars.py

eng\_alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ"

rus\_alphabet = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

nums = "0123456789"

# Проверка текста на пригодность

def is\_good(text, lng):

if lng == "Английский":

for s in text:

if s in eng\_alphabet:

return True

else:

for s in text:

if s in rus\_alphabet:

return True

return False

# Очистка ключа от лишних символов

def clear\_key(key):

res = []

for s in key:

if s in nums:

res.append(s)

return ''.join(res)

# Шифровка

def encode(text, key, lng):

res = []

ck = clear\_key(key)

shift = int(ck)

if lng == "Английский":

for s in text:

place = eng\_alphabet.find(s)

new\_place = place + shift

if s in eng\_alphabet:

res.append(eng\_alphabet[new\_place])

else:

res.append(s)

else:

for s in text:

place = rus\_alphabet.find(s)

new\_place = place + shift

if s in rus\_alphabet:

res.append(rus\_alphabet[new\_place])

else:

res.append(s)

return ''.join(res)

# Расшифровка

def decode(text, key, lng):

res = []

ck = clear\_key(key)

shift = int(ck)

if lng == "Английский":

for s in text:

place = eng\_alphabet.find(s)

new\_place = place - shift

if s in eng\_alphabet:

res.append(eng\_alphabet[new\_place])

else:

res.append(s)

else:

for s in text:

place = rus\_alphabet.find(s)

new\_place = place - shift

if s in rus\_alphabet:

res.append(rus\_alphabet[new\_place])

else:

res.append(s)

return ''.join(res)

Файл config.py

from enum import Enum

# Токен бота

TOKEN = "5000448224:AAFJBCAT\_Nx7\_HCffybup3NPs7HE9mRd544"

# Файл базы данных Vedis

db\_file = "db.vdb"

# Ключ записи в базу данных для текущего состояния

CURRENT\_STATE = "CURRENT\_STATE"

# Ключ записи в базу данных для выполняемого действия

SELECTED\_ACTION = "SELECTED\_ACTION"

# Ключ записи в базу данных для выбранного алфавита

SELECTED\_ALPHABET = "SELECTED\_ALPHABET"

# Состояния конечного 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"

|  |
| --- |
| 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

from vedis import Vedis

# 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 caesars

# Создаём бота

bot = telebot.TeleBot(config.TOKEN)

# Прописываем типовые сообщения

mes\_encode = "Зашифровать"

mes\_decode = "Расшифровать"

mes\_eng\_alphabet = "Английский"

mes\_rus\_alphabet = "Русский"

# Приветствие и начало диалога

@bot.message\_handler(commands=['start'])

def cmd\_start(message):

bot.send\_message(message.chat.id, 'Приветствую! Этот бот может шифровать и дешифровать слова шифром Цезаря!')

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, 'Что мы хотим сделать?', reply\_markup=markup)

# /reset - сброс состояния, начало диалога

@bot.message\_handler(commands=['reset'])

def cmd\_reset(message):

bot.send\_message(message.chat.id, 'Сброс, откат!')

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, 'Что мы хотим сделать?', 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, 'Что мы хотим сделать?')

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, 'Какой алфавит у исходного сообщения?', 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, 'Какой алфавит у исходного сообщения?')

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 исходное слово...')

# Ввод сообщения пользователем

@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.upper()

if not caesars.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, "Ошибка! 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 (число)...')

# Ввод ключа шифровки/дешифровки

@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

dbworker.set(dbworker.make\_key(message.chat.id, config.States.STATE\_KEY.value), key)

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 = caesars.encode(text, key, lng)

else:

res = caesars.decode(text, key, lng)

bot.send\_message(message.chat.id, f'Получена задача: {action} сообщение. '

f'Был выбран {lng} алфавит, '

f'исходное слово - {text}, '

f'(ключ) сдвиг на {key} символа алфавита, '

f'получили - {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, 'Что мы хотим сделать?', reply\_markup=markup)

if \_\_name\_\_ == '\_\_main\_\_':

bot.infinity\_polling()

Файл TDD.py

import unittest

from caesars import encode, decode

class CaesarsTest(unittest.TestCase):

def test\_eng\_encode(self):

res = encode("HELLO", "15", "Английский")

self.assertEqual(res, "WTAAD")

def test\_eng\_decode(self):

res = decode("XIPIKVEQ", "4", "Английский")

self.assertEqual(res, "TELEGRAM")

def test\_ru\_decode(self):

res = decode("ЙШХЗЭ", "3", "Русский")

self.assertEqual(res, "МЫШКА")

def test\_ru\_encode(self):

res = encode("ИНТЕРНЕТ", "13", "Русский")

self.assertEqual(res, "ХЪЯСЭЪСЯ")

Файл BDD.feature

Feature: Caesars

Scenario: eng\_encode

Given **Английский** алфавит, исходный текст **HELLO**, ключ **15**

When Хочу **Зашифровать** сообщение

Then Должен увидеть **WTAAD**

Scenario: eng\_decode

Given **Английский** алфавит, исходный текст **XIPIKVEQ**, ключ **4**

When Хочу **Расшифровать** сообщение

Then Должен увидеть **TELEGRAM**

Файл steps.py

# -\*- coding: utf-8 -\*-

from behave import given, when, then

from caesars import encode, decode

@given("{lng} алфавит, исходный текст {txt}, 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("Хочу {act} сообщение")

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)

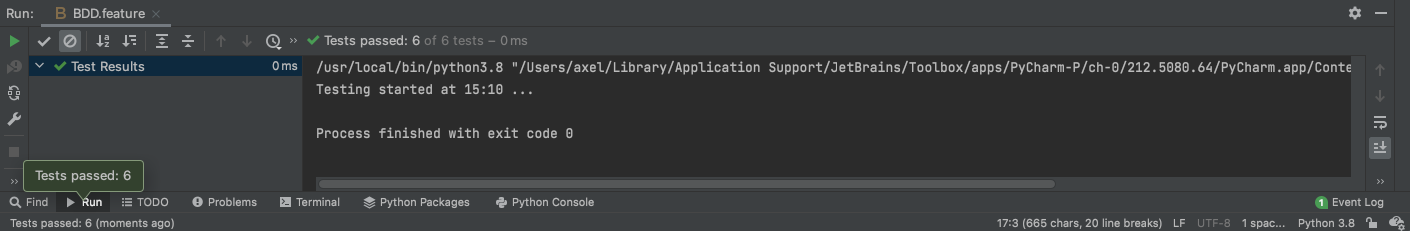
@then("Должен увидеть {res}")

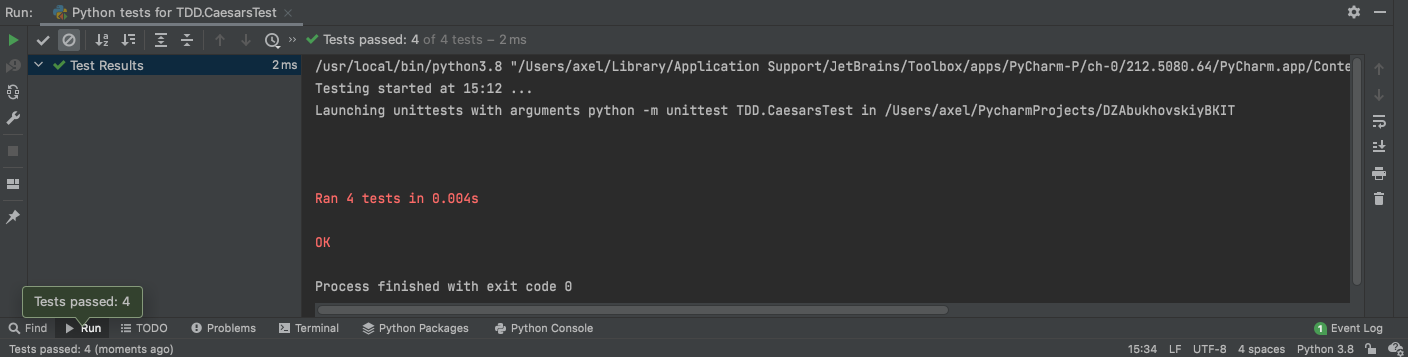
def step\_impl(context, res: str):

assert context.res == res

# Экранные формы с примерами выполнения программы

Тесты:





Бот:

