# 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
Абуховский Иван
Александрович

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

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
```

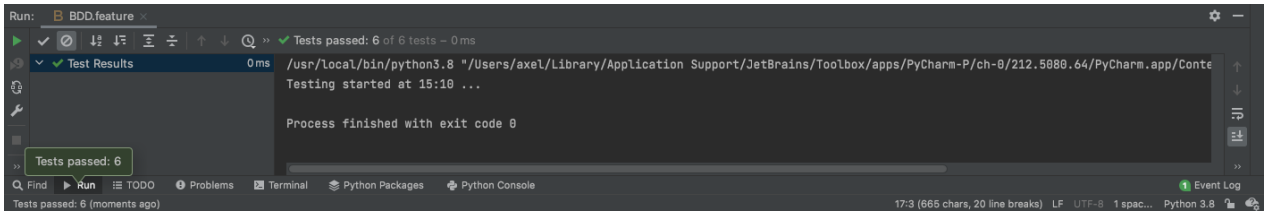
## Файл steps.py

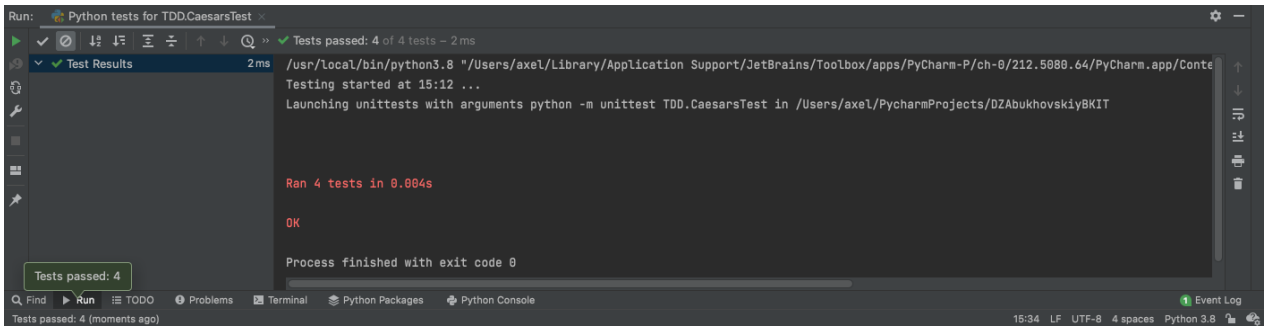
```
# -*- coding: utf-8 -*-
from behave import given, when, then
from caesars import encode, decode

@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: