



**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 ИУ5 «Системы обработки информации и управления»**

ДЗ

по дисциплине «Базовые компоненты интернет-технологий»

**Выполнил:
студент группы ИУ5-32Б
Долинский А.А.**

**Проверил:
Канев А.И.**

2021 г.

Задание:

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).

bot.py

```
import telebot
from DZ import dbworker, config
import random
from datetime import datetime

available_food_names = ["суши", "пицца", "паста", "бургер", "борщ",
"пельмени"]
available_colour_names = ["синий", "черный", "белый", "желтый", "красный",
"фиолетовый"]
bot = telebot.TeleBot(config.token)

@bot.message_handler(commands=['start'])
def keyboard(message):
    keyboard = telebot.types.ReplyKeyboardMarkup(row_width=3,
resize_keyboard=True)
    time = telebot.types.KeyboardButton(text="Показать текущее время")
    user = telebot.types.KeyboardButton(text="Добавить нового
пользователя")
    rand = telebot.types.KeyboardButton(text="Вывести случайное целое число
от 0 до 100")
    keyboard.add(time, user, rand)
    bot.send_message(message.chat.id, "Приветствую!",
reply_markup=keyboard)

# Начало диалога
@bot.message_handler(func=lambda message: message.text.lower() == 'добавить
нового пользователя')
def new_user(message):
    state = dbworker.get_current_state(message.chat.id)
    if state == config.States.S_ENTER_NAME.value:
        bot.send_message(message.chat.id,
"Кажется, кто-то обещал отправить своё имя, но так
и не сделал этого :( Жду...")
    elif state == config.States.S_ENTER_AGE.value:
        bot.send_message(message.chat.id,
"Кажется, кто-то обещал отправить свой возраст, но
так и не сделал этого :( Жду...")
    elif state == config.States.S_ENTER_FOOD.value:
        bot.send_message(message.chat.id,
"Кажется, кто-то обещал отправить своё любимое
блюдо, но так и не сделал этого")
    elif state == config.States.S_ENTER_COLOUR.value:
        bot.send_message(message.chat.id,
"Кажется, кто-то обещал отправить свой любимый
цвет, но так и не сделал этого")
    else: # Под "остальным" понимаем состояние "0" - начало диалога
        bot.send_message(message.chat.id, "Введите имя пользователя.")
        dbworker.set_state(message.chat.id,
config.States.S_ENTER_NAME.value)
```

```

# 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, "Что ж, начнём по-новой. Как вас
зовут?")
    dbworker.set_state(message.chat.id, config.States.S_ENTER_NAME.value)

@bot.message_handler(
    func=lambda message: dbworker.get_current_state(message.chat.id) ==
config.States.S_ENTER_NAME.value)
def user_entering_name(message):
    if not message.text.isalpha():
        bot.send_message(message.chat.id, "Имя должно состоять из букв,
попробуйте ещё раз!")
        return
    else:
        bot.send_message(message.chat.id, "Хорошо! А теперь укажите возраст
пользователя.")
        dbworker.set_state(message.chat.id,
config.States.S_ENTER_AGE.value)

# Обработка возраста пользователя
@bot.message_handler(
    func=lambda message: dbworker.get_current_state(message.chat.id) ==
config.States.S_ENTER_AGE.value)
def user_entering_age(message):
    # А вот тут сделаем проверку
    if not message.text.isdigit():
        # Состояние не меняем, поэтому только выводим сообщение об ошибке и
ждём дальше
        bot.send_message(message.chat.id, "Что-то не так, попробуйте ещё
раз!")
        return
    # На данном этапе мы уверены, что message.text можно преобразовать в
число, поэтому ничем не рискуем
    if int(message.text) < 4 or int(message.text) > 100:
        bot.send_message(message.chat.id, "Какой-то странный возраст.
Отвечайте честно, сколько Вам лет?")
        return
    else:
        # Возраст введён корректно, можно идти дальше
        bot.send_message(message.chat.id, 'Отлично! Укажите любимое блюдо
пользователя.')
        dbworker.set_state(message.chat.id,
config.States.S_ENTER_FOOD.value)

@bot.message_handler(
    func=lambda message: dbworker.get_current_state(message.chat.id) ==
config.States.S_ENTER_FOOD.value)
def user_entering_food(message):
    if food_check(message.text.lower()):
        bot.send_message(message.chat.id, "Пожалуйста, выберите другое Ваше
любимое блюдо.")
        return
    else:
        bot.send_message(message.chat.id,
'Прекрасно! А сейчас укажите Ваш любимый цвет.')
        dbworker.set_state(message.chat.id,
config.States.S_ENTER_COLOUR.value)

```

```

@bot.message_handler(
    func=lambda message: dbworker.get_current_state(message.chat.id) ==
    config.States.S_ENTER_COLOUR.value)
def user_entering_food(message):
    if colour_check(message.text.lower()):
        bot.send_message(message.chat.id, "Пожалуйста, выберите другой
любимый цвет.")
        return
    else:
        bot.send_message(message.chat.id,
            'Отлично! Больше ничего не требуется. Если
захотите добавить еще пользователя - '
            'нажмите на кнопку "Добавить нового
пользователя".')
        dbworker.set_state(message.chat.id, config.States.S_START.value)

@bot.message_handler(content_types="text")
def aaa(message):
    if message.text == 'Да':
        bot.send_message(message.chat.id, "Нет!")
    elif message.text == 'Показать текущее время':
        inlinekb = telebot.types.InlineKeyboardMarkup(row_width=2)
        item1 = telebot.types.InlineKeyboardButton("Спасибо",
callback_data='thanks')
        item2 = telebot.types.InlineKeyboardButton("Ого, кажется мне уже
пора", callback_data='bye')
        inlinekb.add(item1, item2)
        current_datetime = datetime.now().strftime("%d-%m-%Y %H:%M:%S")
        bot.send_message(message.chat.id, current_datetime,
reply_markup=inlinekb)
    elif message.text == 'Вывести случайное целое число от 0 до 100':
        bot.send_message(message.chat.id, str(random.randint(0, 100)))
    elif message.text != 'Добавить нового пользователя':
        bot.send_message(message.chat.id, message.text)

def food_check(text):
    if text not in available_food_names:
        return 1
    else:
        return 0

def colour_check(text):
    if text not in available_colour_names:
        return 1
    else:
        return 0

@bot.callback_query_handler(func=lambda call: True)
def callback_inline(call):
    try:
        if call.message:
            if call.data == 'thanks':
                bot.send_message(call.message.chat.id, 'Всегда пожалуйста')
            elif call.data == 'bye':
                bot.send_message(call.message.chat.id, 'Пока-пока')
    except Exception as e:
        print(repr(e))

```

```
bot.infinity_polling()
```

config.py

```
from enum import Enum

token = '5048250215:AAEPzaDNfSvmz_zFLBTkoXbC8HvCjcm9sfc'
db_file = "database.vdb"

class States(Enum):
    """
    Мы используем БД Vedis, в которой хранимые значения всегда строки,
    поэтому и тут будем использовать тоже строки (str)
    """
    S_START = "0" # Начало нового диалога
    S_ENTER_NAME = "1"
    S_ENTER_AGE = "2"
    S_ENTER_FOOD = "3"
    S_ENTER_COLOUR = "4"
```

dbworker.py

```
from vedis import Vedis
from DZ import config

# Пытаемся узнать из базы «состояние» пользователя
def get_current_state(user_id):
    with Vedis(config.db_file) as db:
        try:
            return db[user_id].decode()
        except KeyError: # Если такого ключа почему-то не оказалось
            return config.States.S_START.value # значение по умолчанию -
начало диалога

# Сохраняем текущее «состояние» пользователя в нашу базу
def set_state(user_id, value):
    with Vedis(config.db_file) as db:
        try:
            db[user_id] = value
            return True
        except:
            return False
```

TDD_Test.py

```
import unittest
import sys, os
from DZ.bot import *
sys.path.append(os.getcwd())

class TestBot(unittest.TestCase):
    def test_food_check(self):
        self.assertEqual(food_check("суши"), 0)
        self.assertEqual(food_check("щи"), 1)
        pass

    def test_colour_check(self):
        self.assertEqual(colour_check("синий"), 0)
        self.assertEqual(colour_check("бирюзовый"), 1)
        pass

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

BDD_Test.py

```
from behave import *
from tests.TDD_Test import *

@given("Bot")
def Bot(context):
    context.a = TestBot()

@when("Test food return OK")
def food_check(context):
    context.a.test_food_check()

@when("Test colour return OK")
def colour_check(context):
    context.a.test_colour_check()

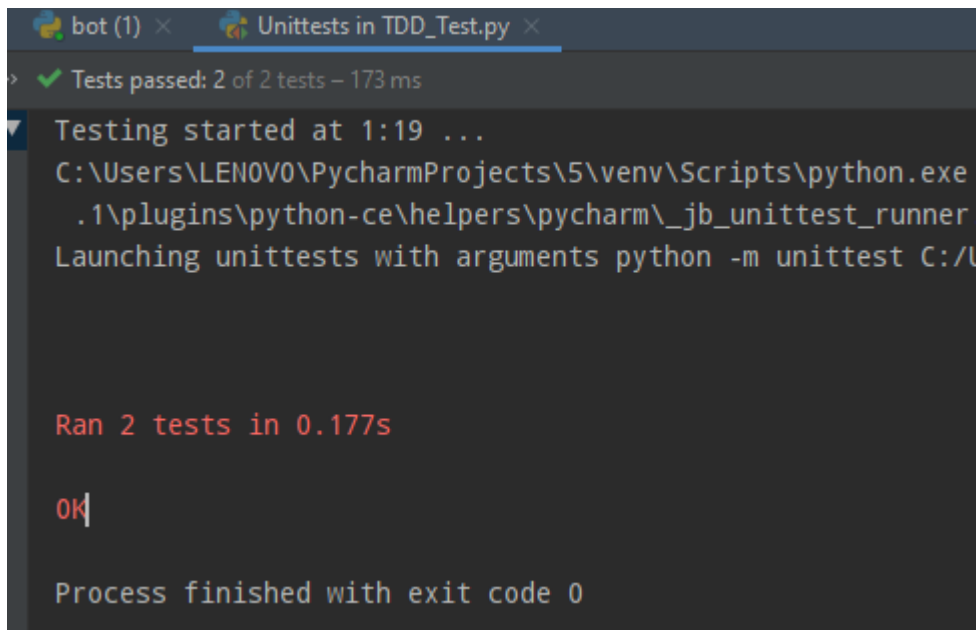
@then("Great Great!!!!")
def check_result(context):
    pass
```

BDD_Test.feature

```
Feature: Test
  Scenario: Test bot
    Given Bot
    When Test food return OK
    When Test colour return OK
    Then Great Great!!!!
```

Экранные формы

TDD_Test



The screenshot shows a PyCharm terminal window with two tabs: 'bot (1)' and 'Unittests in TDD_Test.py'. The terminal output indicates that 2 out of 2 tests passed in 173 ms. It also shows the command used to launch unittests: 'python -m unittest C:/Users/LEN0V0/PycharmProjects/5/venv/Scripts/python.exe .1\plugins\python-ce\helpers\pycharm\jb_unittest_runner'. The output concludes with 'Ran 2 tests in 0.177s' and 'OK', followed by 'Process finished with exit code 0'.

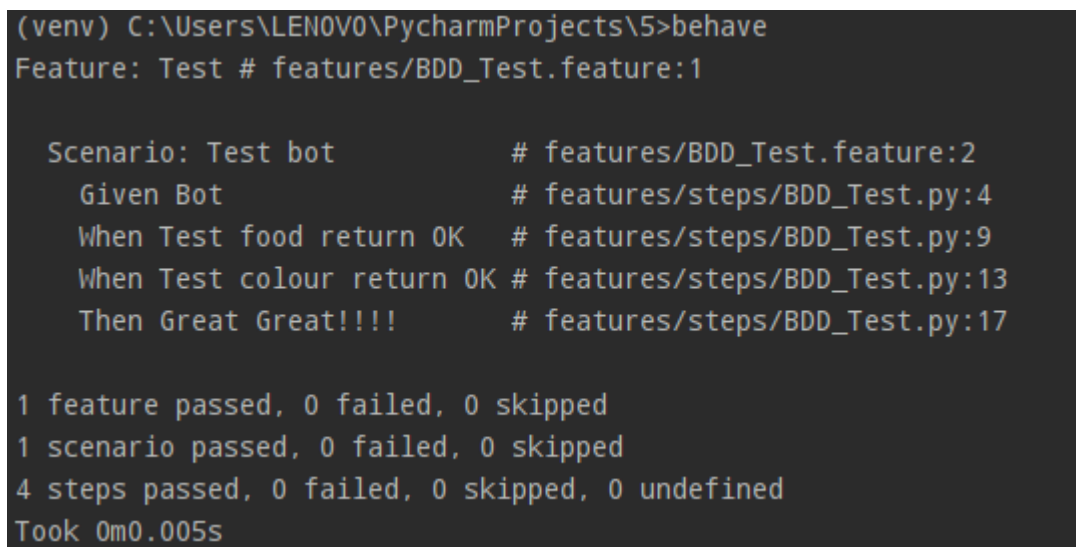
```
bot (1) × Unittests in TDD_Test.py ×
> ✓ Tests passed: 2 of 2 tests – 173 ms
Testing started at 1:19 ...
C:\Users\LEN0V0\PycharmProjects\5\venv\Scripts\python.exe
.1\plugins\python-ce\helpers\pycharm\jb_unittest_runner
Launching unittests with arguments python -m unittest C:/l

Ran 2 tests in 0.177s

OK

Process finished with exit code 0
```

BDD_Test



The screenshot shows a terminal window with the command '(venv) C:\Users\LEN0V0\PycharmProjects\5>behave'. The output lists a feature 'Test # features/BDD_Test.feature:1' and a scenario 'Test bot # features/BDD_Test.feature:2'. The scenario steps are: 'Given Bot # features/steps/BDD_Test.py:4', 'When Test food return OK # features/steps/BDD_Test.py:9', 'When Test colour return OK # features/steps/BDD_Test.py:13', and 'Then Great Great!!!! # features/steps/BDD_Test.py:17'. The final summary shows '1 feature passed, 0 failed, 0 skipped', '1 scenario passed, 0 failed, 0 skipped', '4 steps passed, 0 failed, 0 skipped, 0 undefined', and 'Took 0m0.005s'.

```
(venv) C:\Users\LEN0V0\PycharmProjects\5>behave
Feature: Test # features/BDD_Test.feature:1

  Scenario: Test bot # features/BDD_Test.feature:2
    Given Bot # features/steps/BDD_Test.py:4
    When Test food return OK # features/steps/BDD_Test.py:9
    When Test colour return OK # features/steps/BDD_Test.py:13
    Then Great Great!!!! # features/steps/BDD_Test.py:17

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
4 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.005s
```