|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

Выполнил:

студент группы ИУ5-35Б Герасимов А.Д

г. 2021

# 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).

# Текст программы

Botmain.py

import telebot  
from telebot import types  
import config  
import dbworker  
import unittest  
import TDD  
  
from calcfunc import Calculaion  
bot = telebot.TeleBot(config.TOKEN)  
  
  
class BotMain():  
  
 @bot.message\_handler(commands=['start'])  
 def cmd\_start(message):  
 bot.send\_message(message.chat.id, 'Я умею вычислять пириметр и площадь фигуры!')  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE), config.States.STATE\_SELECT\_SHAPE.value)  
 bot.send\_message(message.chat.id, 'Выберете или введите название фигуры!')  
 itembtn1 = types.KeyboardButton('Прямоугольник')  
 itembtn2 = types.KeyboardButton('Круг')  
 itembtn3 = types.KeyboardButton('Квадрат')  
 markup = types.ReplyKeyboardMarkup(row\_width=3)  
 markup.add(itembtn1, itembtn2, itembtn3)  
 bot.send\_message(message.chat.id, 'Введите название фигуры!', 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\_SELECT\_SHAPE.value)  
 bot.send\_message(message.chat.id, 'Выберете или введите название фигуры!')  
 itembtn1 = types.KeyboardButton('Прямоугольник')  
 itembtn2 = types.KeyboardButton('Круг')  
 itembtn3 = types.KeyboardButton('Квадрат')  
 markup = types.ReplyKeyboardMarkup(row\_width=3)  
 markup.add(itembtn1, itembtn2, itembtn3)  
 bot.send\_message(message.chat.id, 'Введите название фигуры!', reply\_markup=markup)  
  
 # Обработка названия фигуры  
 @bot.message\_handler(func=lambda message: dbworker.get(  
 dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_SELECT\_SHAPE.value)  
 def select\_shape(message):  
 text = message.text  
 print(text)  
 if text == 'Круг' or text == 'круг' or text == 'circle':  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE),  
 config.States.STATE\_CIRCLE\_RADIUS.value)  
 bot.send\_message(message.chat.id, 'Введите радиус круга!')  
  
 elif text == 'Прямоугольник' or text == 'прямоугольник' or text == 'Rectangle':  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE),  
 config.States.STATE\_RECTANGLE\_HEIGHT.value)  
 bot.send\_message(message.chat.id, 'Введите высоту прямоугольника!')  
  
 elif text == 'Квадрат' or text == 'квадрат' or text == 'square':  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE),  
 config.States.STATE\_SQUARE\_SIDE.value)  
 bot.send\_message(message.chat.id, 'Введите размер стороны квадрата!')  
  
 @bot.message\_handler(func=lambda message: dbworker.get(  
 dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_SQUARE\_SIDE.value)  
 def square\_side(message):  
 text = message.text  
 if not text.isdigit():  
 # Состояние не изменяется, выводится сообщение об ошибке  
 bot.send\_message(message.chat.id, 'Пожалуйста введите размер стороны квадрата!')  
 return  
 else:  
 bot.send\_message(message.chat.id, f'Вы ввели размер стороны квадрата: {text} ')  
 # Меняем текущее состояние  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE),  
 config.States.STATE\_CALCULATION\_SQUARE.value)  
 # Сохраняем первое число  
 dbworker.set(dbworker.make\_key(message.chat.id, config.States.STATE\_SQUARE\_SIDE.value), text)  
 itembtn1 = types.KeyboardButton("Посчитать площадь.")  
 itembtn2 = types.KeyboardButton("Посчитать периметр.")  
 itembtn3 = types.KeyboardButton("Вычислить площадь и периметр.")  
 markup = types.ReplyKeyboardMarkup(row\_width=3)  
 markup.add(itembtn1, itembtn2, itembtn3)  
 bot.send\_message(message.chat.id, 'Выберете или введите операцию!', reply\_markup=markup)  
  
 @bot.message\_handler(func=lambda message: dbworker.get(  
 dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_CALCULATION\_SQUARE.value)  
 def square\_calculation(message):  
 operation = message.text  
 side = dbworker.get(dbworker.make\_key(message.chat.id, config.States.STATE\_SQUARE\_SIDE.value))  
 float\_side = float(side)  
 calculate\_area = 0  
 calculate\_perimeter = 0  
 multiplication = "\*"  
 markup = types.ReplyKeyboardRemove(selective=False)  
 if operation == "Посчитать площадь.":  
  
 bot.send\_message(message.chat.id,  
 f'Площадь квадрата равна: {float\_side}{multiplication}{float\_side}={str(Calculaion.calculation\_square\_area(float\_side))}',  
 reply\_markup=markup)  
 elif operation == "Посчитать периметр.":  
  
 bot.send\_message(message.chat.id,  
 f'Периметр квадрата равен: ({float\_side}+{float\_side})\*2={str(Calculaion.calculation\_square\_perimetr(float\_side))}',  
 reply\_markup=markup)  
 elif operation == "Вычислить площадь и периметр.":  
  
  
 bot.send\_message(message.chat.id,  
 f'Площадь квадрата равна: {float\_side}{multiplication}{float\_side}={str(Calculaion.calculation\_square\_area(float\_side))}',  
 reply\_markup=markup)  
 calculate\_perimeter = (float\_side + float\_side) \* 2  
 bot.send\_message(message.chat.id,  
 f'Периметр квадрата равен: ({float\_side}+{float\_side})\*2={str(Calculaion.calculation\_square\_perimetr(float\_side))}',  
 reply\_markup=markup)  
 bot.send\_message(message.chat.id, 'Введите /reset !', reply\_markup=markup)  
  
 @bot.message\_handler(func=lambda message: dbworker.get(  
 dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_RECTANGLE\_HEIGHT.value)  
 def rectangle\_heiht(message):  
 text = message.text  
 if not text.isdigit():  
 # Состояние не изменяется, выводится сообщение об ошибке  
 bot.send\_message(message.chat.id, 'Пожалуйста введите высоту прямоугольника!')  
 return  
 else:  
 bot.send\_message(message.chat.id, f'Вы ввели высоту прямоугольника: {text} ')  
 # Меняем текущее состояние  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE),  
 config.States.STATE\_RECTANGLE\_WIGHT.value)  
 # Сохраняем первое число  
 dbworker.set(dbworker.make\_key(message.chat.id, config.States.STATE\_RECTANGLE\_HEIGHT.value), text)  
 bot.send\_message(message.chat.id, 'Введите ширину прямоугольника!')  
  
 @bot.message\_handler(func=lambda message: dbworker.get(  
 dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_RECTANGLE\_WIGHT.value)  
 def rectangle\_wight(message):  
 text = message.text  
 if not text.isdigit():  
 # Состояние не изменяется, выводится сообщение об ошибке  
 bot.send\_message(message.chat.id, 'Пожалуйста введите ширину прямоугольника!')  
 return  
 else:  
 bot.send\_message(message.chat.id, f'Вы ввели ширину прямоугольника: {text} ')  
 # Меняем текущее состояние  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE),  
 config.States.STATE\_CALCULATING\_RECTANGLE.value)  
 # Сохраняем первое число  
 dbworker.set(dbworker.make\_key(message.chat.id, config.States.STATE\_RECTANGLE\_WIGHT.value), text)  
 itembtn1 = types.KeyboardButton("Посчитать площадь.")  
 itembtn2 = types.KeyboardButton("Посчитать периметр.")  
 itembtn3 = types.KeyboardButton("Вычислить площадь и периметр.")  
 markup = types.ReplyKeyboardMarkup(row\_width=3)  
 markup.add(itembtn1, itembtn2, itembtn3)  
 bot.send\_message(message.chat.id, 'Выберете или введите операцию!', reply\_markup=markup)  
  
 @bot.message\_handler(func=lambda message: dbworker.get(  
 dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_CALCULATING\_RECTANGLE.value)  
 def rectangle\_calculation(message):  
 operation = message.text  
 heiht = dbworker.get(dbworker.make\_key(message.chat.id, config.States.STATE\_RECTANGLE\_HEIGHT.value))  
 wight = dbworker.get(dbworker.make\_key(message.chat.id, config.States.STATE\_RECTANGLE\_WIGHT.value))  
 float\_heiht = float(heiht)  
 float\_wight = float(wight)  
 calculate\_area = 0  
 calculate\_perimeter = 0  
 multiplication = "\*"  
 markup = types.ReplyKeyboardRemove(selective=False)  
 if operation == "Посчитать площадь.":  
 bot.send\_message(message.chat.id,  
 f'Площадь прямогольника равна: {float\_heiht}{multiplication}{float\_wight}={str(Calculaion.calculate\_rectangle\_area(float\_heiht, float\_wight))}',  
 reply\_markup=markup)  
 elif operation == "Посчитать периметр.":  
 bot.send\_message(message.chat.id,  
 f'Периметр прямоугольника равен: ({float\_heiht}+{float\_wight})\*2={str(Calculaion.calculation\_rectangle\_perimetr(float\_heiht, float\_wight))}',  
 reply\_markup=markup)  
 elif operation == "Вычислить площадь и периметр.":  
 bot.send\_message(message.chat.id,  
 f'Площадь прямоугольника равна: {float\_heiht}{multiplication}{float\_wight}={str(Calculaion.calculate\_rectangle\_area(float\_heiht, float\_wight))}',  
 reply\_markup=markup)  
 bot.send\_message(message.chat.id,  
 f'Периметр прямоугольника равен: ({float\_heiht}+{float\_wight})\*2={str(Calculaion.calculation\_rectangle\_perimetr(float\_heiht, float\_wight))}',  
 reply\_markup=markup)  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE), config.States.STATE\_SELECT\_SHAPE.value)  
 # Выводим сообщение  
 bot.send\_message(message.chat.id, 'Введите /reset !')  
  
 @bot.message\_handler(func=lambda message: dbworker.get(  
 dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_CIRCLE\_RADIUS.value)  
 def circle\_radius(message):  
 text = message.text  
 if not text.isdigit():  
 # Состояние не изменяется, выводится сообщение об ошибке  
 bot.send\_message(message.chat.id, 'Пожалуйста введите радиус круга!')  
 return  
 else:  
 bot.send\_message(message.chat.id, f'Вы ввели радиус круга: {text} ')  
 # Меняем текущее состояние  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE),  
 config.States.STATE\_CALCULATING\_CIRCLE.value)  
 # Сохраняем первое число  
 dbworker.set(dbworker.make\_key(message.chat.id, config.States.STATE\_CIRCLE\_RADIUS.value), text)  
 itembtn1 = types.KeyboardButton("Посчитать площадь.")  
 itembtn2 = types.KeyboardButton("Посчитать периметр.")  
 itembtn3 = types.KeyboardButton("Вычислить площадь и периметр.")  
 markup = types.ReplyKeyboardMarkup(row\_width=3)  
 markup.add(itembtn1, itembtn2, itembtn3)  
 bot.send\_message(message.chat.id, 'Выберете или введите операцию!', reply\_markup=markup)  
  
 @bot.message\_handler(func=lambda message: dbworker.get(  
 dbworker.make\_key(message.chat.id, config.CURRENT\_STATE)) == config.States.STATE\_CALCULATING\_CIRCLE.value)  
 def calculating\_circle(message):  
 operation = message.text  
  
 radius = dbworker.get(dbworker.make\_key(message.chat.id, config.States.STATE\_CIRCLE\_RADIUS.value))  
 float\_radius = float(radius)  
 calculate\_area = 0  
 calculate\_perimeter = 0  
 multiplication = "\*"  
 markup = types.ReplyKeyboardRemove(selective=False)  
 if operation == "Посчитать площадь.":  
 bot.send\_message(message.chat.id,  
 f'Площадь круга равна: {3.14}{multiplication}{float\_radius}{multiplication}{float\_radius}={str(Calculaion.calculate\_circle\_area(float\_radius))}',  
 reply\_markup=markup)  
 elif operation == "Посчитать периметр.":  
 bot.send\_message(message.chat.id,  
 f'Периметр круга равен: {2}{multiplication}{3.14}{multiplication}{float\_radius}={str(Calculaion.calculate\_circle\_perimetr(float\_radius))}',  
 reply\_markup=markup)  
 elif operation == "Вычислить площадь и периметр.":  
 bot.send\_message(message.chat.id,  
 f'Площадь круга равна: {3.14}{multiplication}{float\_radius}{multiplication}{float\_radius}={str(Calculaion.calculate\_circle\_area(float\_radius))}',  
 reply\_markup=markup)  
 bot.send\_message(message.chat.id,  
 f'Периметр круга равен: {2}{multiplication}{3.14}{multiplication}{float\_radius}={str(Calculaion.calculate\_circle\_perimetr(float\_radius))}',  
 reply\_markup=markup)  
  
 dbworker.set(dbworker.make\_key(message.chat.id, config.CURRENT\_STATE), config.States.STATE\_SELECT\_SHAPE.value)  
 # Выводим сообщение  
 bot.send\_message(message.chat.id, 'Введите /reset !', reply\_markup=markup)  
  
  
  
# class TestShapeFunctions(unittest.TestCase):  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bot.infinity\_polling()

calcfunc.py

class Calculaion():  
 def calculate\_circle\_area(float\_radius):  
 calculate\_area = 0  
 calculate\_area = 3.14 \* float\_radius \* float\_radius  
 return calculate\_area  
  
 def calculate\_circle\_perimetr(float\_radius):  
 calculate\_perimeter = 0  
 calculate\_perimeter = 2 \* 3.14 \* float\_radius  
 return calculate\_perimeter  
  
 def calculate\_rectangle\_area(float\_heiht, float\_wight):  
 calculate\_area = 0  
 calculate\_area = float\_heiht \* float\_wight  
 return calculate\_area  
  
 def calculation\_rectangle\_perimetr(float\_heiht, float\_wight):  
 calculate\_perimeter = 0  
 calculate\_perimeter = (float\_heiht + float\_wight) \* 2  
 return calculate\_perimeter  
  
 def calculation\_square\_area(float\_side):  
 calculate\_area = 0  
 calculate\_area = float\_side \* float\_side  
 return calculate\_area  
  
 def calculation\_square\_perimetr(float\_side):  
 calculate\_perimeter = 0  
 calculate\_perimeter = (float\_side + float\_side) \* 2  
 return calculate\_perimeter

config.py

from enum import Enum  
  
# Токент бота  
TOKEN = "PrivateToken"  
  
# 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\_SELECT\_SHAPE = "STATE\_SELECT\_SHAPE"  
 STATE\_RECTANGLE\_HEIGHT = "STATE\_RECTANGLE\_HEIGHT"  
 STATE\_RECTANGLE\_WIGHT = "STATE\_RECTANGLE\_WIGHT"  
 STATE\_CIRCLE\_RADIUS = "STATE\_CIRCLE\_RADIUS"  
 STATE\_CALCULATING\_CIRCLE = "STATE\_CALCULATING\_CIRCLE"  
 STATE\_SQUARE\_SIDE = "STATE\_SQUARE\_SIDE"  
 STATE\_CALCULATING\_RECTANGLE = "STATE\_CALCULATING\_RECTANGLE"  
 STATE\_CALCULATION\_SQUARE = "STATE\_CALCULATION\_SQUARE"

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

TDD.py

from botmain import Calculaion  
import unittest  
classic = Calculaion  
class TestShapeFunctions(unittest.TestCase):  
 def test\_calculate\_area\_of\_rectangle(self):  
  
 self.assertEqual(classic.calculate\_rectangle\_area(2,2), 4, "Should be 4")  
  
 def test\_calculate\_perimeter\_of\_rectangle(self):  
  
 self.assertEqual(classic.calculation\_rectangle\_perimetr(2,2), 8, "Should be 8")  
  
 def test\_calculate\_area\_of\_Square(self):  
  
 self.assertEqual(classic.calculation\_square\_area(2), 4, "Should be 4")  
  
 def test\_calculate\_perimeter\_of\_Square(self):  
  
 self.assertEqual(classic.calculation\_square\_perimetr(2), 8, "Should be 8")  
  
 def test\_calculate\_area\_of\_circle(self):  
  
 self.assertEqual(classic.calculate\_circle\_area(2), 12.56, "Should be 12.56")  
  
 def test\_calculate\_perimeter\_of\_circle(self):  
  
 self.assertEqual(classic.calculate\_circle\_perimetr(2), 12.56, "Should be 12.56")  
  
  
suite = unittest.TestLoader().loadTestsFromTestCase(TestShapeFunctions)  
unittest.TextTestRunner(verbosity=2).run(suite)

steps.py

from behave import given, when, then, step  
import telebot  
from telebot import types  
from vedis import Vedis  
from calcfunc import Calculaion  
  
  
@given("Я хочу получить площадь прямоугольника")  
def initial\_impl(context):  
 pass  
  
  
@when("Стороны прямоугольника равны {float\_heiht:d} {float\_wight:d}")  
def area\_impl(context,float\_heiht, float\_wight):  
 context.result = Calculaion.calculate\_rectangle\_area(float\_heiht, float\_wight)  
 pass  
  
  
@then("Получаю площадь, равную {result:d}")  
def result\_impl(context, result):  
 assert context.result == result  
 pass  
  
  
# # Square  
  
@given("Я хочу получить площадь квадрата")  
def initial\_impl(context):  
 pass  
  
  
@when("Сторона квадрата равна {float\_side:d}")  
def area\_impl(context,float\_side):  
 context.result = Calculaion.calculation\_square\_area(float\_side)  
 pass  
  
  
@then("Получаю площадь, равную {result:d}")  
def result\_impl(context, result):  
 assert context.result == result  
 pass

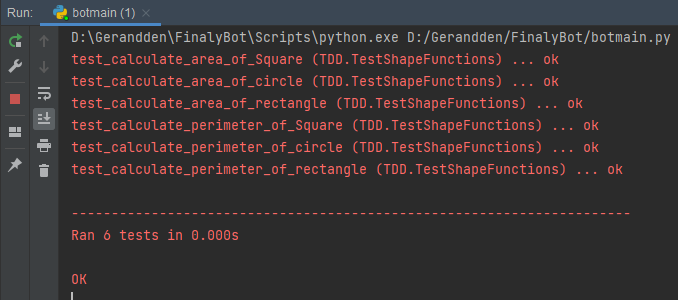
rectangle.feature

Feature: Rectangle  
 Scenario: Arguments for given, when, then  
 Given Я хочу получить площадь прямоугольника  
 When Стороны прямоугольника равны **2 3** Then Получаю площадь, равную **6**

square.feature

Feature: square  
 Scenario: Arguments for given, when, then  
 Given Я хочу получить площадь квадрата  
 When Сторона квадрата равна **2** Then Получаю площадь, равную **4**

Результаты тестирования:

TDD: 

BDD: 