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»
Отчет по домашнему заданию

Выполнил:
студент группы ИУ5-31Б
Коваленко
Алексей Викторович

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись:_____

Дата:_____

Москва, 2021 г.

Домашнее задание

Описание задания

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

Файл DZbot.py

```
import asyncio

import discord
import sqlite3
from discord.ext import commands
from setting import settings

intents = discord.Intents(messages=True, guilds=True, members = True)
intents.reactions = True

TOKEN = settings['token']
client = commands.Bot(command_prefix='&', intents=intents)
client.remove_command('help')

connection = sqlite3.connect('oracle.db')
cursor = connection.cursor()



@client.event
async def on_ready():

    cursor.execute("""CREATE TABLE IF NOT EXISTS socrating (
        id INT,
        rating INT
    )""")
    connection.commit()

    cursor.execute("""CREATE TABLE IF NOT EXISTS warnings (
        user_id INT,
        admin_id INT,
        descr TEXT,
        amount INT,
        warntype TEXT,
        id INT
    )""")

    cursor.execute("""CREATE TABLE IF NOT EXISTS modcase (
        case_id INT,
        user_id INT,
        admin_id INT,
        descr TEXT
    )""")

    connection.commit()

    for guild in client.guilds:
        for member in guild.members:
            if cursor.execute(f"SELECT id FROM socrating WHERE id =
{member.id}").fetchone() is None:
                cursor.execute(f"INSERT INTO socrating
VALUES({member.id},30)")
            else:
                pass
    connection.commit()
    print('Join done')
```

```
    await weekly()


@client.event
async def on_memeber_join(member):
    if cursor.execute(f"SELECT id FROM socrating WHERE id =
{member.id}").fetchone() is None:
        cursor.execute(f"INSERT INTO socrating
VALUES({member.id},30,'{member}')")
        connection.commit()
    else:
        pass


@client.command(aliases = ['rating'])
async def __rating(ctx, member: discord.Member=None):
    if member==None:
        await ctx.send(embed=discord.Embed(
                description = ratingt(ctx.author,ctx.author.id)
            ))
    else:
        await ctx.send(embed=discord.Embed(
            description = ratingt(member,member.id)
        ))

def ratingt(member,id):
    return f"""Social rating of **{member}** is **{cursor.execute(f"SELECT
rating FROM socrating WHERE id = {id}").fetchone()[0]}** points"""


@client.command(aliases = ['addrating','add'])
@commands.has_any_role(*settings['admins'])
async def __addrating(ctx,member: discord.Member = None, amount : int=None):
    print('add triggered')
    if member==None:
        await ctx.send(embed=discord.Embed(
            description = "Please enter user, for who you want to manage
rating"
        ))
    else:
        if amount==None:
            await ctx.send(embed=discord.Embed(
                description = "Please enter amount of rating, which you want
to add"
            ))
        else:
            add(member.id,amount)

            await ctx.message.add_reaction('✅')
            await notification(member,client)
            await cap_check(member)

def add(id,amount):
    cursor.execute("UPDATE socrating SET rating = rating +{} WHERE id =
{}".format(amount,id))
    connection.commit()


@client.command(aliases = ['decreaserating','decrease','minus'])
@commands.has_any_role(*settings['admins'])
async def __decrating(ctx,member: discord.Member = None, amount : int=None):
    if member==None:
        await ctx.send(embed=discord.Embed(
            description = "Please enter user, for who you want to manage
```

```
rating"
        ))
    else:
        if amount==None:
            await ctx.send(embed=discord.Embed(
                description = "Please enter amount of rating, which you want
to decrease"
            ))
        else:
            cursor.execute("UPDATE socrating SET rating = rating - {} WHERE
id = {}".format(amount,member.id))
            connection.commit()
            await ctx.message.add_reaction('✅')
            await notification(member,client)
            await cap_check(member)



@client.event
async def on_command_error(ctx, error):
    if isinstance(error, commands.MissingAnyRole):
        await ctx.send("*bonk* You have no permissions here")


async def weekly():
    print ('weekly started')
    #await asyncio.sleep(604800)
    await asyncio.sleep(1000)
    for guild in client.guilds:
        for member in guild.members:
            cursor.execute("UPDATE socrating SET rating = rating + 5 WHERE id
= {}".format(member.id))
            await cap_check(member)
            await notification(member,client)
    connection.commit()

    await weekly()

async def cap_check(member): #our current cap is 100
    amount= cursor.execute("SELECT rating FROM socrating WHERE id =
{}".format(member.id)).fetchone()[0]
    if amount>100:
        #print('deacrease done')
        cursor.execute("UPDATE socrating SET rating = 100 WHERE id =
{}".format(member.id))
        connection.commit()


async def notification(member,client):
    pass
    #amount= cursor.execute("SELECT rating FROM socrating WHERE id =
{}".format(member.id)).fetchone()[0]
    #if amount<25:
       # print('notif done')
       # logs = client.get_channel(settings['logs'])
      #  await logs.send(embed=discord.Embed(
       #     description = f"""**{member}** reached low level of reputation.
Current points is **{cursor.execute(f"SELECT rating FROM socrating WHERE id =
{member.id}").fetchone()[0]}**."""
        #))


@client.command(aliases = ['warn'])
@commands.has_any_role(*settings['admins'])
```

```
async def __warn(ctx,member: discord.Member = None, amount : int=None,
description: str =None):
    if member==None:
        await ctx.send(embed=discord.Embed(
            description = "Please enter user, for who you want to warn"
        ))
    else:
        if amount==None:
            await ctx.send(embed=discord.Embed(
                description = "Please enter amount of rating, which you want
to decrease"
            ))
        else:
            cursor.execute("UPDATE socrating SET rating = rating - {} WHERE
id = {}".format(amount,member.id))
            connection.commit()
            if amount <10:
                warntype = 'lightwarning'
            elif amount >= 10 and amount <25:
                warntype = 'warning'
            else:
                warntype = 'hardwarning'
            warn_id=cursor.execute(f"SELECT MAX(id) FROM
warnings").fetchone()[0]
            print(warn_id)
            if warn_id==None:
                warn_id=0
            cursor.execute(f"INSERT INTO warnings
VALUES({member.id},{ctx.author.id},'{description}',{amount},'{warntype}',{war
n_id+1})")
            connection.commit()
            await ctx.message.add_reaction('✅')
            await notification(member,client)
            await cap_check(member)


@client.command(aliases = ['unwarn'])
@commands.has_any_role(*settings['admins'])
async def __unwarn(ctx,case_id:int=None):
    if case_id==None:
        await ctx.send(embed=discord.Embed(
            description = "Please enter id of warning, which you want to
delete"
        ))
    else:
        amount = cursor.execute(f"SELECT amount FROM warnings WHERE id =
{case_id}").fetchone()[0]
        member = cursor.execute(f"SELECT user_id FROM warnings WHERE id =
{case_id}").fetchone()[0]
        cursor.execute("UPDATE socrating SET rating = rating + {} WHERE id =
{}".format(amount,member))
        cursor.execute(f"DELETE FROM warnings WHERE id = {case_id}")
        connection.commit()
        await ctx.message.add_reaction('✅')
        await notification(member,client)
        await cap_check(member)

@client.command(aliases = ['warnings'])
async def __warning(ctx,member: discord.Member = None):
    if member==None:
        await ctx.send(embed=discord.Embed(
            description = "Please enter user"
        ))
    else:
```

```
        amount = cursor.execute(f"SELECT COUNT(*) FROM warnings WHERE user_id
= {member.id}").fetchone()[0]
        res = cursor.execute(f"SELECT * FROM warnings WHERE user_id =
{member.id}").fetchmany(amount)
        desc = f"Warnings of {member} \n"
        for i in range (0,amount):
            desc += f"Case {res[i][5]} of {res[i][4]}: User -
{client.get_user(res[i][0])}, Admin -
{client.get_user(res[i][1])},\nDescription:{res[i][2]}, Deacreased for
{res[i][3]}\n\n"
        await ctx.send(
            embed=discord.Embed(
                description = desc
            ))


global state
state = 0
global joke_line
joke_line=[]

@client.event
async def on_message(message):
    global state
    if message.author != client.user:
        if state !=0:
            chan = client.get_channel(920072025303830599)
            if message.content == ' reset':
                await reset()
                await chan.send(embed=discord.Embed(
                    description = 'Module reseted'))
            elif message.content == 'exit':
                await exit()
                await chan.send(embed=discord.Embed(
                    description = 'Module off'))
            elif state == 1 :
                if message.content == 'back':
                    await reset()
                    await chan.send(embed=discord.Embed(
                        description = 'We are back to previous stage'))
                else:
                    joke_line.append(message.content)
                    state+=1
                    await chan.send(embed=discord.Embed(
                        description = 'Enter second word to joke'))
            elif state ==2:
                if message.content == 'back':
                    state-=1
                    joke_line.pop()
                    await chan.send(embed=discord.Embed(
                        description = 'We are back to previous stage'))
                else:
                    joke_line.append(message.content)
                    state+=1
                    await chan.send(embed=discord.Embed(
                        description = 'Enter third word to joke'))
            elif state ==3:
                if message.content == 'back':
                    state-=1
                    joke_line.pop()
                    await chan.send(embed=discord.Embed(
                        description = 'We are back to previous stage'))
                else:
                    joke_line.append(message.content)
```

```
                    state+=1
                    await chan.send(embed=discord.Embed(
                        description = 'Do you want to see result of your
work?(Yes/No)'))
            elif state ==4:
                if message.content.lower()=='yes':
                    await chan.send(embed=discord.Embed(
                        description = f'Идет {joke_line[0]} по
{joke_line[1]}, видит, {joke_line[2]} горит. Сел в нее и сгорел.'))
                    await reset()
                elif message.content.lower()=='no':
                    await chan.send(embed=discord.Embed(
                        description = 'Im very sad to her that'))
                    await reset()
                else:
                    await chan.send(embed=discord.Embed(
                        description = 'I SAID YES OR NO!! moron'))
    await client.process_commands(message)


@client.command(aliases = ['start6'])
async def __start6(ctx):
    global state
    state = 1
    await ctx.send(embed=discord.Embed(
        description = 'Stipid joke module activated. Enter first word, whcih
we will add to our joke'
    ))

async def reset():
    global state
    state = 1
    global joke_line
    joke_line = []

async def exit():
    global state
    state = 0
    global joke_line
    joke_line = []


if __name__ == '__main__':
    client.run(TOKEN)
```

Файл tests.py

```
from DZbot import ratingt, add
import discord
import sqlite3
from discord.ext import commands
from setting import settings
import asyncio
import pytest

print(1)
id = 354972489459433472
user = "Holy Rem Crusader#6288"
print(1)
connection = sqlite3.connect('oracle.db')
cursor = connection.cursor()


def test_one():
    result ='Social rating of **Holy Rem Crusader#6288** is **310** points'
```

```
    assert ratingt(user,id)==result

def test_two():
    result = 10
    add(id, 0)
    assert result == cursor.execute(f"SELECT rating FROM socrating WHERE id =
{id}").fetchone()[0]
```

Файл Features.feature

```
# Created by Zver at 22.12.2021
Feature: Homework tests

  Scenario: test one, checking
    Given I want to know rating of Holy Rem Crusader#6288 with id
354972489459433472
    When I use command &rating
    Then I see messeage : Social rating of **Holy Rem Crusader#6288** is **-
50** points

  Scenario: test two, adding
    Given I want to add rating to Holy Rem Crusader#6288 with id
354972489459433472 for 10 points
    When I use command &add
    Then I database should be amount equal to -40
```

Файл step.py

```
from DZbot import ratingt, add
import discord
import sqlite3
from discord.ext import commands
from setting import settings
import asyncio
from behave import given, when, then

connection = sqlite3.connect('oracle.db')
cursor = connection.cursor()

@given(u'I want to know rating of {username} with id {id}')
def step_impl(context, username:str, id:int):
    context.username=str(username)
    context.id=int(id)

@when(u'I use command &rating')
def step_impl(context):
    context.result = ratingt(context.username,context.id)

@then(u'I see messeage : {message}')
def step_impl(context, message:str):
    print (context.result)
    assert context.result == message

@given(u'I want to add rating to {username} with id {id} for {amount}
points')
def step_impl(context, id:int,amount:int, username:str):
    context.id=int(id)
    context.amount = int(amount)

@when(u'I use command &add')
def step_impl(context):
    add(context.id, context.amount)

@then(u'I database should be amount equal to {result}')
def step_impl(context, result:int):
    context.result=int(result)
```

```
    print(cursor.execute(f"SELECT rating FROM socrating WHERE id =
{context.id}").fetchone()[0])
    assert context.result == cursor.execute(f"SELECT rating FROM socrating
WHERE id = {context.id}").fetchone()[0]
```

Экранные формы с примерами выполнения программы