МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И. С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 6

на тему: «Сетевое программирование»

по дисциплине: «Программирование на языке Python»

Выполнил: Евдокимов Н.А.

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

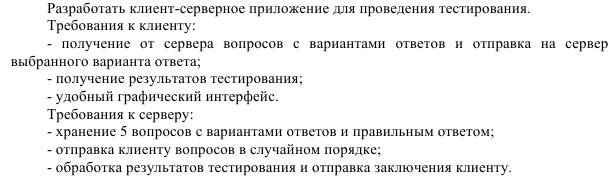
Проверила: Захарова О.В.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орёл, 2019

Задание



Код

Клиент

from PyQt5 import QtWidgets, uic

from PyQt5.QtWidgets import QAction

import sys

import socket

from lab6.server.question import Question

import json

class UI(QtWidgets.QMainWindow):

def \_\_init\_\_(self):

super(UI, self).\_\_init\_\_()

uic.loadUi('lab6.ui', self)

self.handler = Handler(self)

self.chosen\_answer = 0

self.quizLayout = self.findChild(QtWidgets.QVBoxLayout, 'quizLayout')

self.errorLabel = self.findChild(QtWidgets.QLabel, 'errorLabel')

self.scoreLabel = self.findChild(QtWidgets.QLabel, 'scoreLabel')

self.quizText = self.findChild(QtWidgets.QTextBrowser, 'quizText')

btnContainer = self.errorLabel.parent()

self.buttons = []

for i in range(3):

btn = btnContainer.findChildren(QtWidgets.QPushButton)[i]

self.buttons.append(btn)

send0 = lambda: self.handler.send\_answer(self.buttons[0].text())

send1 = lambda: self.handler.send\_answer(self.buttons[1].text())

send2 = lambda: self.handler.send\_answer(self.buttons[2].text())

self.buttons[0].clicked.connect(send0)

self.buttons[1].clicked.connect(send1)

self.buttons[2].clicked.connect(send2)

self.scoreLabel.setVisible(False)

self.errorLabel.setVisible(False)

self.handler.start()

self.show()

def show\_question(self, question\_text: str, answer\_list: list):

self.quizText.setText(question\_text)

self.buttons[0].setText(answer\_list[0])

self.buttons[1].setText(answer\_list[1])

self.buttons[2].setText(answer\_list[2])

def send\_answer(self, answer\_text: str):

self.handler.send\_answer(answer\_text)

def show\_result(self, num\_right: int):

self.scoreLabel.setText('Ваш результат: {}'.format(num\_right))

self.scoreLabel.setVisible(True)

self.quizText.hide()

for btn in self.buttons:

btn.hide()

class Handler:

HOST = 'localhost'

PORT = 1234

def \_\_init\_\_(self, ui: UI):

self.ui = ui

self.socket = None

def connect(self):

self.socket = socket.socket()

self.socket.connect((self.HOST, self.PORT))

print('Client connected')

def disconnect(self):

if self.socket is not None:

self.socket.close()

print('Closed client connection')

def send\_answer(self, answer\_text: str):

self.socket.send(answer\_text.encode('utf-8'))

self.request\_question()

def request\_question(self):

server\_msg = str(self.socket.recv(1024).decode())

is\_end = 'num\_right' in server\_msg

if is\_end:

result = json.loads(server\_msg)

self.ui.show\_result(result['num\_right'])

self.disconnect()

else:

question = Question.from\_json(server\_msg)

self.ui.show\_question(question.text, question.answers)

def start(self):

self.connect()

self.request\_question()

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

app = QtWidgets.QApplication(sys.argv)

window = UI()

app.exec()

Сервер

import socket

import sys

from lab6.server.handler import Handler

class Server:

HOST = ''

PORT = 1234

def \_\_init\_\_(self):

self.handler = Handler()

def start(self):

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

print('Created socket')

try:

s.bind((self.HOST, self.PORT))

except socket.error as msg:

print('Bind failed. Error Code : ' + str(msg.errno) + ' Message ' + msg.strerror)

sys.exit()

print('Socket bind complete')

s.listen(3)

print('Socket is listening...')

while True:

conn, addr = s.accept()

self.handler.handle(conn)

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

server = Server()

server.start()

Handler.py

import socket

import random

from lab6.server.question import Question

import json

class Handler:

def \_\_init\_\_(self):

self.\_questions = [

Question('Кто из президентов США написал свой собственный рассказ про Шерлока Холмса?',

['Джон Кеннеди', 'Франклин Рузвельт', 'Рональд Рейган'],

1),

Question('Какую пошлину ввели в XII веке в Англии для того чтобы заставить мужчин пойти на войну?',

['Налог на тунеядство', 'Налог на трусость', 'Налог на отсутствие сапог '],

1),

Question('Откуда пошло выражение «деньги не пахнут?',

['От подателей за провоз парфюмерии', 'От сборов за нестиранные носки', 'От налога на туалеты'],

2),

Question('Туристы, приезжающие на Майорку, обязаны заплатить налог…',

['На плавки', 'На пальмы', 'На солнце'],

2),

Question('Российский мультфильм, удостоенный «Оскара», — это…',

['Старик и море', 'Винни-Пух', 'Простоквашино'],

0),

]

def handle(self, conn: socket):

randomized\_questions = self.\_questions.copy()

random.shuffle(randomized\_questions)

try:

num\_right = 0

for question in randomized\_questions:

question\_json = question.to\_json()

conn.send(question\_json.encode('utf-8'))

answer = conn.recv(1024).decode()

if answer == question.answers[question.right\_answer]:

num\_right += 1

quiz\_results = {'num\_right': num\_right}

quiz\_results\_json = json.dumps(quiz\_results)

conn.send(quiz\_results\_json.encode('utf-8'))

conn.close()

except socket.error:

print('Socket error occurred!')

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

h = Handler()

h.handle(None)

Question.py

import json

class Question:

def \_\_init\_\_(self, text: str, answers: list, right\_answer: int):

self.right\_answer = right\_answer

self.answers = answers

self.text = text

def to\_json(self):

return json.dumps(self,

default=lambda o: o.\_\_dict\_\_,

sort\_keys=True,

indent=4)

def from\_json(json\_str: str):

loaded\_json = json.loads(json\_str)

text = loaded\_json['text']

answers = loaded\_json['answers']

right\_answer = loaded\_json['right\_answer']

question = Question(text, answers, right\_answer)

return question

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

json\_str = '{"text": "asd", "answers": [1, 2, 3], "right\_answer": 1}'

a = Question.from\_json(json\_str)