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

Разработать клиент-серверное приложение для проведения тестирования.
Требования к клиенту:
- получение от сервера вопросов с вариантами ответов и отправка на сервер выбранного варианта ответа;
- получение результатов тестирования;
- удобный графический интерфейс.
Требования к серверу:
- хранение 5 вопросов с вариантами ответов и правильным ответом;
- отправка клиенту вопросов в случайном порядке;
- обработка результатов тестирования и отправка заключения клиенту.

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)
```