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 № 5
на тему: «Работа с файлами»
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

В программу лабораторной работы № 4 добавить главное меню с пунктами «Файл», «Справка».

Пункт главного меню «Файл» должен включать подпункты «Создать» (очистка формы для ввода новых данных), «Открыть» (считывание из файла информации об объектах и отображение её на форме), «Сохранить» (обновление текущего файла; если файл не создан, то действия пункта «Сохранить как…»), «Сохранить как…» (сохранение информации об объектах в файл по указанному пути), «Выход» (дружественный выход из программы).

Пункт главного меню «Справка» должен показывать информацию о разработчике программы.

Код

```
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import QAction
from lab3.Book import Book
import sys
import pickle as p


class UI(QtWidgets.QMainWindow):
    def __init__(self):
        super(UI, self).__init__()
        uic.loadUi('lab5_m.ui', self)

        self.handler = Handler(self)
        self.init_tool_bar()

        self.btnClear = self.findChild(QtWidgets.QPushButton, 'btnClear')
        self.btnAdd = self.findChild(QtWidgets.QPushButton, 'btnAdd')
        self.btnDelete = self.findChild(QtWidgets.QPushButton, 'btnDelete')
        self.btnMessage = self.findChild(QtWidgets.QPushButton, 'btnMessage')
        self.btnSearch = self.findChild(QtWidgets.QPushButton, 'btnSearch')

        self.leName = self.findChild(QtWidgets.QLineEdit, 'leName')
        self.leIsbn = self.findChild(QtWidgets.QLineEdit, 'leIsbn')
        self.leAuthors = self.findChild(QtWidgets.QLineEdit, 'leAuthors')
        self.lePublisher = self.findChild(QtWidgets.QLineEdit, 'lePublisher')
        self.leSearch = self.findChild(QtWidgets.QLineEdit, 'leSearch')

        self.sbNPages = self.findChild(QtWidgets.QSpinBox, 'sbNPages')
        self.sbPrice = self.findChild(QtWidgets.QDoubleSpinBox, 'dsbPrice')
        self.sbPublicationYear = self.findChild(QtWidgets.QSpinBox, 'sbPublicationYear')

        self.cbMessageType = self.findChild(QtWidgets.QComboBox, 'cbMessageType')

        self.cbWithIcon = self.findChild(QtWidgets.QCheckBox, 'cbWithIcon')

        self.twBooks = self.findChild(QtWidgets.QTableView, 'twBooks')
        self.twBooks.setSelectionBehavior(QtWidgets.QAbstractItemView.SelectRows)

        self.btnAdd.clicked.connect(self.btn_add_clicked)
        self.btnClear.clicked.connect(self.clear_input_fields)
        self.btnDelete.clicked.connect(self.btn_delete_clicked)
        self.btnMessage.clicked.connect(self.btn_message_clicked)

        self.show()

    def closeEvent(self, event) -> None:
```

```
        answer = QtWidgets.QMessageBox.question(self, 'Выход', 'Вы точно хотите выйти?')
        if answer == QtWidgets.QMessageBox.No:
            event.ignore()
        else:
            super(QtWidgets.QMainWindow, self).closeEvent(event)

    def init_tool_bar(self):
        open_act = QAction('Открыть', self)
        open_act.triggered.connect(self.handler.read_books_file)

        exit_act = QAction('Выйти', self)
        exit_act.triggered.connect(self.close)

        save_act = QAction('Сохранить', self)
        save_act.triggered.connect(self.handler.save_books_file)

        save_as_act = QAction('Сохранить как...', self)
        save_as_act.triggered.connect(self.handler.save_book_file_as)

        self.statusBar()

        menu_bar = self.menuBar()
        file_menu = menu_bar.addMenu('Файл')
        about_act = QAction('Справка', self)
        about_act.triggered.connect(self.show_about_msg)
        menu_bar.addAction(about_act)

        file_menu.addAction(open_act)
        file_menu.addAction(save_act)
        file_menu.addAction(save_as_act)
        file_menu.addAction(exit_act)

        self.setWindowTitle('Lab5')

    def show_about_msg(self):
        message = QtWidgets.QMessageBox(self)
        message.setText('Евдокимов Н.А. 71-ПГ')
        message.setWindowTitle('Сообщение')
        message.setIcon(QtWidgets.QMessageBox.Information)

        message.exec()

    def read_books_file(self):
        self.handler.read_books_file()
        self.repaint_table()

    def btn_delete_clicked(self, event):
        if len(self.twBooks.selectedIndexes()) <= 0:
            return

        index = [idx.row() for idx in self.twBooks.selectionModel().selectedRows()][0]
        book_name = self.handler.book_list[index].name
        self.handler.remove_book(book_name)
        self.repaint_table()

    def repaint_table(self):
        self.twBooks.setRowCount(0)
        self.twBooks.setColumnCount(7)

        for book in self.handler.book_list:
            row_pos = self.twBooks.rowCount()
            self.twBooks.insertRow(row_pos)
            self.twBooks.setItem(row_pos, 0, QtWidgets.QTableWidgetItem(book.name))
```

```
            self.twBooks.setItem(row_pos, 1, QtWidgets.QTableWidgetItem(str(book.publication_year)))
            self.twBooks.setItem(row_pos, 2, QtWidgets.QTableWidgetItem(str(book.n_pages)))
            self.twBooks.setItem(row_pos, 3, QtWidgets.QTableWidgetItem(book.isbn))
            self.twBooks.setItem(row_pos, 4, QtWidgets.QTableWidgetItem(','.join(book.authors)))
            self.twBooks.setItem(row_pos, 5, QtWidgets.QTableWidgetItem(book.publisher))
            self.twBooks.setItem(row_pos, 6, QtWidgets.QTableWidgetItem(str(book.price)))

    def btn_message_clicked(self, event):
        message_type = str(self.cbMessageType.currentText())

        is_sad = message_type == 'Грустное сообщение'
        if is_sad:
            message_str = ':('
        else:
            message_str = ':)'

        message = QtWidgets.QMessageBox()
        message.setText(message_str)
        message.setWindowTitle('Сообщение')

        if self.cbWithIcon.isChecked() and is_sad:
            message.setIcon(QtWidgets.QMessageBox.Critical)
        elif self.cbWithIcon.isChecked() and not is_sad:
            message.setIcon(QtWidgets.QMessageBox.Information)

        message.exec()

    def btn_add_clicked(self, event):
        name = self.leName.text()
        pub_year = self.sbPublicationYear.value()
        n_pages = self.sbNPages.value()

        if len(name) <= 0 or n_pages <= 0:
            message = QtWidgets.QMessageBox()
            message.setText('Недостаточно данных о книге. Нужны: название, год издания, кол-во страниц')
            message.setIcon(QtWidgets.QMessageBox.Critical)
            message.setWindowTitle('Книга не добавлена')
            message.exec()

            return

        for book in self.handler.book_list:
            if book.name == name:
                message = QtWidgets.QMessageBox()
                message.setText('Такая книга уже есть')
                message.setIcon(QtWidgets.QMessageBox.Critical)
                message.setWindowTitle('Книга не добавлена')
                message.exec()

                return

        isbn = self.leIsbn.text()
        authors = self.leAuthors.text().replace(' ', '').split(',')
        publisher = self.lePublisher.text()
        price = self.sbPrice.value()
        self.handler.add_book(name, pub_year, n_pages, isbn, authors, publisher, price)
        self.repaint_table()

    def clear_input_fields(self):
        self.leName.setText('')
        self.lePublisher.setText('')
        self.leAuthors.setText('')
        self.leIsbn.setText('')
```

```
        self.sbPublicationYear.setValue(0)
        self.sbPrice.setValue(0)
        self.sbNPages.setValue(0)


class Handler:
    def __init__(self, ui):
        self.window = ui
        self.file_path = None
        self.book_list = []

    def read_books_file(self):
        path = QtWidgets.QFileDialog.getOpenFileName(self.window, 'Выберите файл')[0]
        if len(path) == 0:
            return

        self.file_path = path
        with open(path, 'rb') as f:
            try:
                self.book_list = p.load(f)
            except EOFError:
                self.book_list = []
        self.window.repaint_table()

    def save_books_file(self):
        if self.file_path is None:
            self.save_book_file_as()
            return

        path = self.file_path
        with open(path, 'wb') as f:
            p.dump(self.book_list, f)

    def save_book_file_as(self):
        path = QtWidgets.QFileDialog.getSaveFileName(self.window, 'Выберите файл')[0]

        if len(path) == 0:
            return

        self.file_path = path
        with open(path, 'wb') as f:
            p.dump(self.book_list, f)

    def add_book(self,
                 name,
                 publication_year,
                 n_pages,
                 isbn=None, authors=None, publisher=None, price=None):

        book = Book(name, publication_year, n_pages, isbn, authors, publisher, price)
        self.book_list.append(book)
        # self.write_books_file()

    def remove_book(self, book_name):
        index = -1
        for i, book in enumerate(self.book_list):
            if book.name == book_name:
                index = i
                break

        if index < 0:
            return
```

```
        self.book_list.pop(index)


if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    window = UI()
    app.exec()
```