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

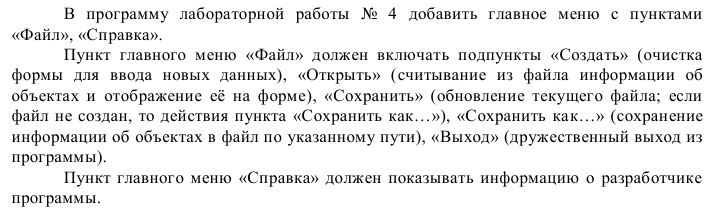
Проверила: Захарова О.В.

Отметка о зачете:

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

Орёл, 2019

Задание



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()