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 № 7

на тему: «Разработка приложения, взаимодействующего с базой данных»

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

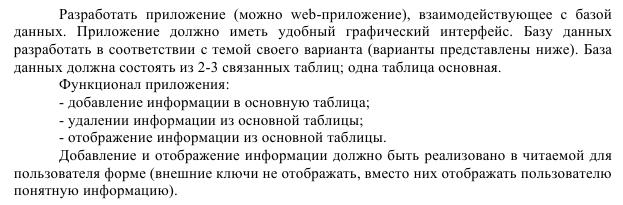
Проверила: Захарова О.В.

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

import mysql.connector

class UI(QtWidgets.QMainWindow):

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

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

uic.loadUi('lab7.ui', self)

self.handler = Handler(self)

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.setWindowTitle('Lab7')

self.handler.read\_books\_file()

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 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()

isbn = self.leIsbn.text()

authors = self.leAuthors.text().replace(' ', '').upper().split(',')

publisher = self.lePublisher.text()

price = self.sbPrice.value()

if len(name) <= 0 or n\_pages <= 0 or len(isbn) <= 0 or len(authors) <= 0 or len(publisher) <= 0 or price <= 0:

message = QtWidgets.QMessageBox()

message.setText('Недостаточно данных о книге')

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

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 = []

self.db = mysql.connector.connect(

host='localhost',

user='root',

passwd='password',

database='python\_lab\_7'

)

def read\_books\_file(self):

self.book\_list = []

cursor = self.db.cursor()

sql\_statement = 'SELECT books.id, books.num\_pages, books.pub\_year, books.price, books.name, ' \

'books.isbn\_id, books.publisher\_id FROM books'

cursor.execute(sql\_statement)

selected\_books = cursor.fetchall()

for selected\_book in selected\_books:

# get isbn

sql\_statement = 'SELECT isbn.name FROM isbn WHERE isbn.id = %s'

cursor.execute(sql\_statement, (selected\_book[5], ))

isbn = cursor.fetchall()[0][0]

# get publisher

sql\_statement = 'SELECT publisher.name FROM publisher WHERE publisher.id = %s'

cursor.execute(sql\_statement, (selected\_book[6], ))

publisher = cursor.fetchall()[0][0]

# get\_authors

sql\_statement = 'SELECT author.name FROM books ' \

'INNER JOIN author\_has\_books ON author\_has\_books.books\_id = books.id ' \

'INNER JOIN author ON author\_has\_books.author\_id = author.id ' \

'WHERE books.id = %s'

cursor.execute(sql\_statement, (selected\_book[0], ))

authors = [author\_name[0] for author\_name in cursor.fetchall()]

book = Book(selected\_book[4], selected\_book[2], selected\_book[1],

isbn, authors, publisher, selected\_book[3])

self.book\_list.append(book)

cursor.close()

self.window.repaint\_table()

def add\_book(self, name, publication\_year, n\_pages,

isbn, authors, publisher, price):

book = Book(name, publication\_year, n\_pages, isbn, authors, publisher, price)

self.book\_list.append(book)

# Check for ISBN

cursor = self.db.cursor()

sql\_statement = 'SELECT isbn.id, isbn.name FROM isbn WHERE isbn.name = %s'

cursor.execute(sql\_statement, (isbn, ))

select\_result = cursor.fetchall()

is\_isbn\_stored = len(select\_result) > 0

if not is\_isbn\_stored:

sql\_statement = 'INSERT INTO isbn (`name`) VALUES (%s)'

cursor.execute(sql\_statement, (isbn, ))

self.db.commit()

isbn\_id = cursor.getlastrowid()

else:

isbn\_id = select\_result[0][0]

# Check for publisher

sql\_statement = 'SELECT publisher.id, publisher.name FROM publisher WHERE publisher.name = %s'

cursor.execute(sql\_statement, (publisher, ))

select\_result = cursor.fetchall()

is\_publisher\_stored = len(select\_result) > 0

if not is\_publisher\_stored:

sql\_statement = 'INSERT INTO publisher (`name`) VALUES (%s)'

cursor.execute(sql\_statement, (publisher,))

self.db.commit()

publisher\_id = cursor.getlastrowid()

else:

publisher\_id = select\_result[0][0]

# check for authors

author\_ids = []

sql\_statement = 'SELECT author.id from author WHERE author.name = %s'

for author in authors:

cursor.execute(sql\_statement, (author, ))

select\_result = cursor.fetchall()

if len(select\_result) <= 0:

sql = 'INSERT INTO author (`name`) VALUES (%s)'

cursor.execute(sql, (author, ))

self.db.commit()

author\_ids.append(cursor.getlastrowid())

else:

author\_ids.append(select\_result[0][0])

sql\_statement = 'INSERT INTO books (num\_pages, pub\_year, price, `name`, isbn\_id, publisher\_id) ' \

'VALUES (%s, %s, %s, %s, %s, %s)'

cursor.execute(sql\_statement,

(book.n\_pages, book.publication\_year, book.price, book.name, isbn\_id, publisher\_id))

self.db.commit()

book\_id = cursor.getlastrowid()

for author\_id in author\_ids:

sql\_statement = 'INSERT INTO author\_has\_books (author\_id, books\_id) VALUES (%s, %s)'

cursor.execute(sql\_statement, (author\_id, book\_id))

self.db.commit()

cursor.close()

def remove\_book(self, book\_name):

cursor = self.db.cursor()

# get book id

sql\_statement = 'SELECT books.id FROM books WHERE books.name = %s'

cursor.execute(sql\_statement, (book\_name, ))

book\_id = cursor.fetchall()[0][0]

# delete from author\_has\_books

sql\_statement = 'DELETE FROM author\_has\_books WHERE author\_has\_books.books\_id = %s'

cursor.execute(sql\_statement, (book\_id, ))

self.db.commit()

sql\_statement = 'DELETE FROM books WHERE books.name = %s'

cursor.execute(sql\_statement, (book\_name, ))

cursor.close()

self.db.commit()

cursor.close()

self.read\_books\_file()

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

app = QtWidgets.QApplication(sys.argv)

window = UI()

app.exec()

# read\_books\_file()

# repaint\_table()