

|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное бюджетное образовательное учреждение высшего образования  **«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Лабораторная работа № 2**

по дисциплине

«Программирование для ЭВМ»

на тему:

**«**Работа с файлами**»**

**Выполнил:**

Учебная группа: 1бПМ   
ФИО: Борисов И. М

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель лабораторной работы:**

Должность: старший преподаватель

Звание: б/з

ФИО: Кутейников И. А.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Целью:

Реализовать простейший файловый менеджер. (Для Java использовать пакеты

java.io|java.nio, класс File|Files). Программа должна обладать следующим

функционалом:

1 Выбор файла или каталога для работы;

2 Вывод абсолютного пути для текущего файла или каталога;

3 Вывод содержимого каталога;

4 Вывод всей возможной информации для заданного файла;

5 Изменение имени файла или каталога;

6 Создание нового файла или каталога по заданному пути;

7 Создание копии файла по заданному пути;

8 Вывод списка файлов текущего каталога, имеющих расширение,

задаваемое пользователем;

9 Удаление файла или каталога;

10.Поиск файла или каталога в выбранном каталоге;

Результаты лабораторной работы оформить в виде отчета с результатами

работы программы.

Код:

…

 def retranslateUi(self, Dialog):

        \_translate = QtCore.QCoreApplication.translate

        Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))

        self.AddFileButton.setText(\_translate("Dialog", "Add file"))

        self.FindFileButton.setText(\_translate("Dialog", "Find file"))

        self.DeleteFileButton.setText(\_translate("Dialog", "Delete file"))

        self.FindFolderButton.setText(\_translate("Dialog", "Find folder"))

        self.DeleteFolderButton.setText(\_translate("Dialog", "Delete folder"))

    def open\_dialog(self):

        # Открываем диалоговое окно для выбора файла

        file\_dialog = QFileDialog(self)

        file\_dialog.setNameFilter("Text files (\*.txt)")

        file\_dialog.setViewMode(QFileDialog.List)

        file\_dialog.setFileMode(QFileDialog.ExistingFiles)

        if file\_dialog.exec\_():

            selected\_files = file\_dialog.selectedFiles()

            file\_path = selected\_files[0] if selected\_files else ""

            file\_size = os.path.getsize(file\_path)

            with open(file\_path, "r", encoding='utf-8') as file:

                file\_text = file.read()

            file\_name = os.path.basename(file\_path)

            self.lineEdit.setText(f"{file\_path}")

            self.textEdit.setText(f"File: {file\_path}\nFile name: {file\_name}\nSize: {file\_size} bytes\nText:\n{file\_text}")

    def add\_file(self):

        self.create\_file = Create\_file\_UI.MyDialog()

        self.create\_file.show()

    def delete\_file(self):

        file\_path = self.lineEdit.text()

        if os.path.exists(file\_path):  # Проверяем существование файла

            os.remove(file\_path)  # Удаляем файл

            self.lineEdit.clear()  # Очищаем поле пути

            self.textEdit.clear()

            print("File deleted successfully.")

        else:

            self.open\_dialog()

    def select\_folder(self):

        folder\_path = QFileDialog.getExistingDirectory(self, "Select Folder", "")

        if folder\_path:

            self.lineEdit.setText(folder\_path)

    def delete\_folder(self):

        folder\_path = self.lineEdit.text()

        if not folder\_path:

            QMessageBox.warning(self, "Warning", "Please select a folder first.")

            return

        try:

            shutil.rmtree(folder\_path)

            QMessageBox.information(self, "Success", "Folder successfully deleted.")

            self.lineEdit.clear()

        except OSError as e:

            QMessageBox.critical(self, "Error", f"Failed to delete folder: {e.strerror}")

  def retranslateUi(self, Dialog):

        \_translate = QtCore.QCoreApplication.translate

        Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))

        self.SaveFileButton.setText(\_translate("Dialog", "Save"))

        self.BackButton.setText(\_translate("Dialog", "Back"))

    def save\_file(self):

        options = QFileDialog.Options()

        file\_name, \_ = QFileDialog.getSaveFileName(self, "Save File", "", "Text Files (\*.txt);;All Files (\*)", options=options)

        text = self.textEdit.toPlainText()

        if file\_name:

            with open(file\_name, "w") as file:

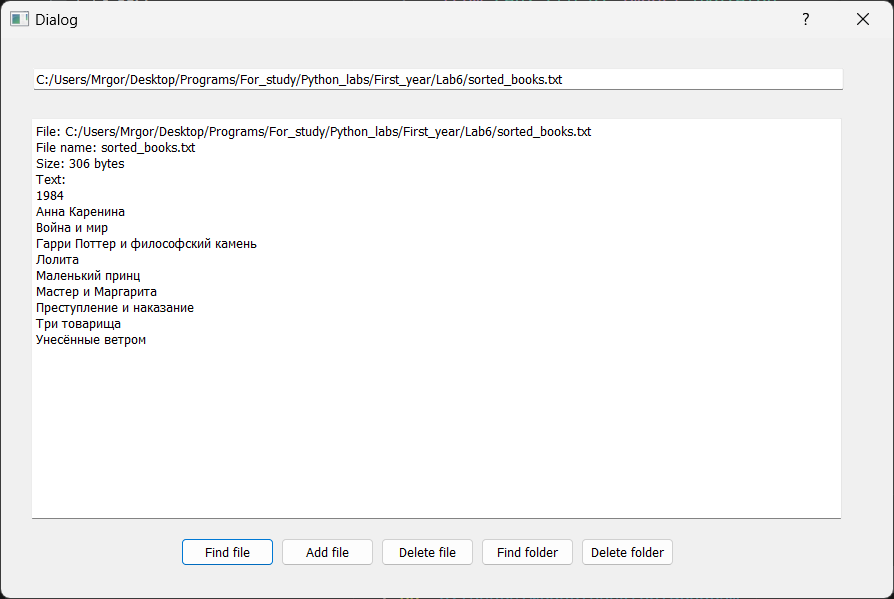
                file.write(text)

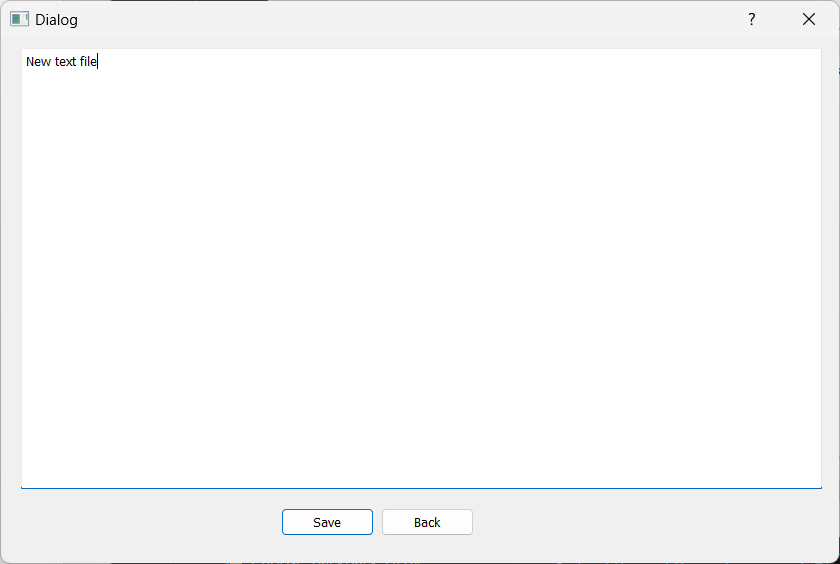
    def close\_window(self):

        self.close()

Результат:

Меню:





Вывод:

В ходе данной лабораторной работы был построен простейший файловый менеджер. Появилось лучшее понимание того, как работать с библиотекой io, а также произошло знакомство с библиотекой nio. Также появилось лучшее понимание того, как работать с файлами и синтаксисом.