

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра САПР**

**КУРСОВАЯ РАБОТА**  
**по дисциплине «Объектно-ориентированное программирование»**  
**Тема: Поиск дубликатов файлов**

Студент гр. 8309

\_\_\_\_\_

Ворончихин А.А.

Преподаватель

\_\_\_\_\_

Красильников А.В.

Санкт-Петербург

2021

## **ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студент Ворончихин А.А.

Группа 8309

Тема работы: Поиск дубликатов файлов

Исходные данные:

Требуется создать программу с графическим интерфейсом, производящую поиск одинаковых файлов и их удаление.

Содержание пояснительной записки:

Пояснительная записка содержит: введение, описание хода работы, демонстрацию работы, заключение и листинг с кодом на языке Python 3.9

Предполагаемый объем пояснительной записки:

Не менее 5 страниц.

Дата выдачи задания: 01.11.2020

Дата сдачи реферата: 06.03.2021

Дата защиты реферата: 06.03.2021

Студент

\_\_\_\_\_

Ворончихин А.А.

Преподаватель

\_\_\_\_\_

Красильников А.В.

## **АННОТАЦИЯ**

В курсовой работе содержится решение задачи и демонстрация работы. В ходе выполнения данной работы были получены важные практические навыки и закреплены теоретические знания материала. Результаты, которые были получены были проверены с помощью различных файлов, что подтверждает их работоспособность.

## **SUMMARY**

The course work contains a solution to the problem and a demonstration of the work. In the course of this work, important practical skills were obtained and theoretical knowledge of the material was consolidated. The results that were obtained were checked using various files, which confirms their operability.

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	5
1. ХОД РАБОТЫ .....	6
1.1. Применяемые языки, модули и библиотеки .....	6
1.2. Этапы проектирования курсовой работы.....	7
2. ЛИСТИНГ ПРОЕКТА .....	8
2.1. inter.py .....	8
2.2. main.py .....	9
2.3. workcode.py .....	10
ЗАКЛЮЧЕНИЕ.....	12

## **ВВЕДЕНИЕ**

Цель работы: написать приложение с графическим интерфейсом для поиска повторяющихся файлов в выбранной директории. Пользователь выбирает через диалог желаемую директорию. Программа осуществляет поиск всех возможных дубликатов файлов (по содержимому, а не по имени файла, конечно же) в этой директории и во всех вложенных и показывает пользователю результат. Должна быть предусмотрена возможность удаления дубликатов через интерфейс программы

Основная задача: написать программу на языке программирования Python 3.9 с применением графического интерфейса

# 1. ХОД РАБОТЫ

## 1.1. Применяемые языки, модули и библиотеки

В работе применялся язык программирования Python актуальной версии

3.9.2. При разработке применялись следующие модули/библиотеки:

Таблица 1 – Применяемые модули.

№	Модуль	Назначение
1	hashlib	Модуль предназначен для создания hash сравниваемых файлов. Hash разных файлов всегда отличается
2	os	Модуль предназначен для взаимодействия с операционной системой
3	sys	Модуль обеспечивает доступ к функциям интерпретатора, например, поиск путей объектов
4	PyQt5	Модуль предназначен для создания графического интерфейса. Вместе с ним применялся Qt Designer – конструктор интерфейсов
5	WPF*	Модуль предназначен для создания графического интерфейса
6	Windows Forms*	Модуль предназначен для создания графического интерфейса
7	__future__*	Модуль позволяет импортировать функционал более новых версий ЯП Python

Примечание: под символом «\*» отмечены модули, не попавшие в финальную версию программы. Отказ от WPF и Windows Forms связан с тем, что для их реализации используется версия ЯП IronPython, являющаяся модификацией синтаксиса Python версии 2.7. В связи с тем, что современная версия языка 3.9, код, который был написан к реализации интерфейса, не был совместим с этими модулями и с IronPython в целом. Модуль \_\_future\_\_ не смог повлиять на ситуацию, поэтому, в последний момент было принято решение перейти на модуль PyQt5, как на более современный и поддерживаемый.

Были применены IDE Microsoft Visual Studio с пакетами Python и IronPython, после PyCharm и Qt Designer

## 1.2. Этапы проектирования курсовой работы

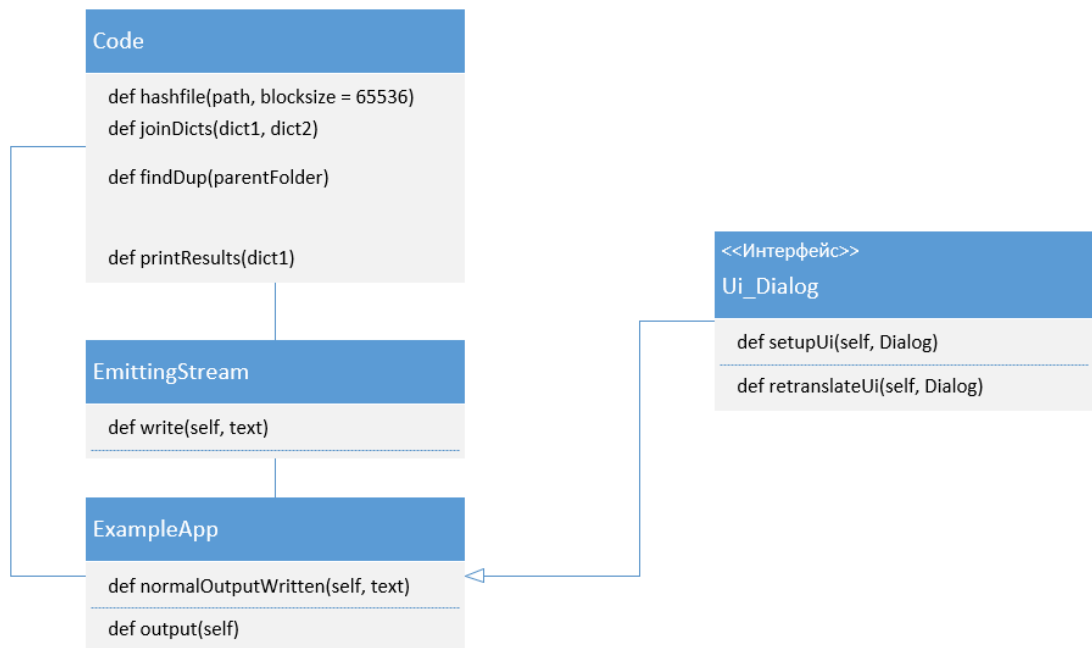


Рисунок 1 – Диаграмма классов UML

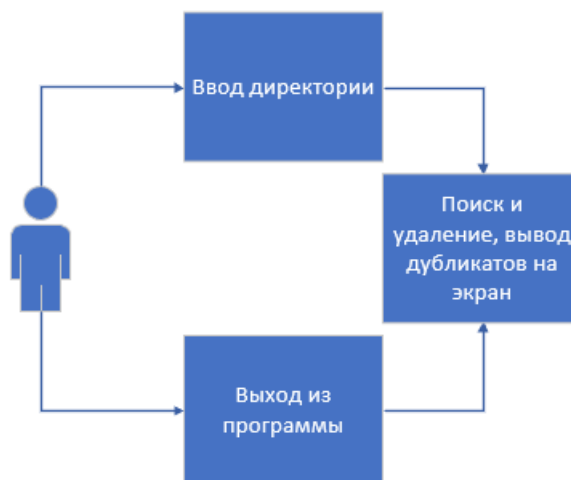


Рисунок 2 – Use – Case диаграмма

## 2. ЛИСТИНГ ПРОЕКТА

### 2.1. inter.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file
'D:\courseoop\inter.ui'
#
# Created by: PyQt5 UI code generator 5.15.3
#
# WARNING: Any manual changes made to this file will be lost when
pyuic5 is
# run again. Do not edit this file unless you know what you are doing.

from PyQt5 import QtCore, QtGui, QtWidgets
class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dublicate Search 1.0")
        Dialog.resize(599, 300)
        self.pushButton = QtWidgets.QPushButton(Dialog)
        self.pushButton.setGeometry(QtCore.QRect(480, 30, 101, 31))
        self.pushButton.setObjectName("pushButton")
        self.textEdit = QtWidgets.QTextEdit(Dialog)
        self.textEdit.setGeometry(QtCore.QRect(20, 30, 461, 31))
        self.textEdit.setObjectName("textEdit")
        self.textEdit_2 = QtWidgets.QTextEdit(Dialog)
        self.textEdit_2.setGeometry(QtCore.QRect(20, 60, 561, 221))
        self.textEdit_2.setObjectName("textEdit_2")
        self.label = QtWidgets.QLabel(Dialog)
        self.label.setGeometry(QtCore.QRect(20, 10, 441, 16))
        self.label.setObjectName("label")
        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)
    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dublicate Search 1.0", "Dublicate Search 1.0"))
        self.pushButton.setText(_translate("Dublicate Search 1.0", "Search"))
        self.label.setText(_translate("Dublicate Search 1.0", "Enter the path of directory:"))
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```



## 2.2. main.py

```
from PyQt5 import QtWidgets, QtCore, QtGui
import inter
import workcode
import sys

#F:\\test

class EmittingStream(QtCore.QObject):
    textWritten = QtCore.pyqtSignal(str)
    def write(self, text):
        self.textWritten.emit(str(text))

class ExampleApp(QtWidgets.QDialog, inter.Ui_Dialog):
    def __init__(self):
        super().__init__()
        self.setupUi(self) # Инициализация
        self.pushButton.clicked.connect(self.output)
        sys.stdout =
        EmittingStream(textWritten=self.normalOutputWritten)

    def __del__(self):
        # Restore sys.stdout
        sys.stdout = sys.__stdout__

    def normalOutputWritten(self, text):# Перехват командной строки
        cursor = self.textEdit_2.textCursor()
        cursor.movePosition(QtGui.QTextCursor.End)
        cursor.insertText(text)
        self.textEdit_2.setTextCursor(cursor)
        self.textEdit_2.ensureCursorVisible()

    def output(self):
        global data
        self.textEdit_2.setText('')# Обновление окна
        data = self.textEdit.toPlainText()
        self.textEdit_2.insertPlainText(data)
        self.textEdit_2.insertPlainText('\n')
        workcode.main1(data)

def main():
    app = QtWidgets.QApplication(sys.argv) # Новый экземпляр
    QApplication
    window = ExampleApp() # Создаём объект класса ExampleApp
    window.show() # Показываем окно
    app.exec_() # и запускаем приложение

if __name__ == '__main__':
    main()
```

## 2.3. workcode.py

```
import os
import hashlib
class code():
    def __init__(self):
        pass
    def hashfile(path, blocksize = 65536):
        afile = open(path, 'rb')
        hasher = hashlib.md5()
        buf = afile.read(blocksize)
        while len(buf) > 0:
            hasher.update(buf)
            buf = afile.read(blocksize)
        afile.close()
        return hasher.hexdigest()
    def findDup(parentFolder):
        dups = {}
        for dirName, subdirs, fileList in os.walk(parentFolder):
            for filename in fileList:
                path = os.path.join(dirName, filename)# Полный путь к
                file_hash = code.hashfile(path)# Вычисление хэша
                if file_hash in dups:
                    dups[file_hash].append(path)# добавление дубликата
                    os.remove(path)
                else:
                    dups[file_hash] = [path]# добавление нового типа
                    дубликата
        return dups
    def joinDicts(dict1, dict2):
        for key in dict2.keys():
            if key in dict1:
                dict1[key] = dict1[key] + dict2[key]
            else:
                dict1[key] = dict2[key]
    def printResults(dict1):
        results = list(filter(lambda x: len(x) > 1, dict1.values()))
        if len(results) > 0:
            print('Duplicates Found:')
            print('The following files are identical. The name could
            differ, but the content is identical')
            print('_____')
            for result in results:
                for subresult in result:
                    print(subresult)
            print('_____')
        else:
            print('No duplicate files found.')
```

```
def main1(data1):  
    dups = {}  
    code.joinDicts(dups, code.findDup(data1))  
    code.printResults(dups)
```

Для запуска программы требуется создать проект из этих трех файлов в одной директории, при этом необходимо, чтобы к IDE был подключен PyQt. Установщики PyQt можно легко найти в интернете, однако, установка иногда различается. Например, к PyCharm нельзя присоединить этот модуль через стандартные инструкции с применением командной строки. Дополнительно нужно в настройках PyCharm выбрать Tools> External Tools, там с помощью кнопки следует добавить пакеты PyQt.

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы мы улучшили наши навыки владения языком Python 3.9, а также среды создания графических приложений WPF на основе IronPython и PyQt на основе Python 3