МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное

учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

Лабораторная работа № 9

По дисциплине «Алгоритмы и структуры данных»

Тема «Расчасовка дисциплин»

Руководство программиста

Р.02069337.№23/690-№20 РП-01

Листов 6

Исполнитель:

студент гр. ИСТбд-22

Минибаева Е.Р.

«»_____2024 г.

2024 г.

1. Обзор программы

Программа представляет собой графический интерфейс (GUI) для управления списком расчасовки дисциплин. Она позволяет:

- Загружать данные о расчасовке дисциплин из файла, выбранного пользователем.
- Просматривать список расчасовки дисциплин.
- Выполнять сегментацию полного списка дисциплин по преподавателям и по видам учебной нагрузки.
- Отображать результаты сегментации в форме круговой диаграммы.

2. Структура программы

Программа состоит из двух классов:

1. Discipline: Представляет собой объект расчасовки дисциплин.

Имеет следующие атрибуты:

- name: Название дисциплины.
- type_hour: Вид учебной нагрузки.
- teacher: Имя преподавателя дисциплины.

2. DisciplineManager: Управляет списком дисциплин.

Имеет следующие методы:

- init(): создает список дисциплин
- load_data(filename): Загружает данные о дисциплинах из файла.
- segment_by_type_hour(): Выполняет сегментацию по видам учебной нагрузки.
- segment_by_teacher(): Выполняет сегментацию по преподавателям.
- visualize_data(segmentation, title): Отображает данные в виде диаграммы.

3. DisciplineVisual: Представляет собой визуальную составляющую программы

Имеет следующие методы:

- init(root): создает GUI
- load_data_visual(): Переносит данные из класса менеджер в GUI
- load_data(): Запускает в классе менеджер загрузку данных
- visualize_by_teacher(): Запускает в классе менеджер визуализацию по преподавателям
- visualize_by_type_hour(): Запускает в классе менеджер визуализацию по видам учебной нагрузки

3. Алгоритм работы программы

1. Инициализация:

- При запуске программы создается экземпляры классов DisciplineManager и DisciplineVisual.
- DisciplineVisual создает GUI программы.

2. Взаимодействие пользователя:

- Пользователь может нажать на кнопки "Загрузить данные", "Визуализация по преподавателям", "Визуализация по видам учебной нагрузки".
- При нажатии на кнопку " Загрузить данные " открывается диалоговое окно для выбора текстового файла с данными о расчасовки дисциплин.
- При нажатии на кнопку "Визуализация по преподавателям" открывается окно с круговой диаграммой данных по сегментации по преподавателям, если до этого были загружены данные. Иначе всплывает окно «Предупреждение» с текстом «Нет данных для визуализации. Загрузите данные»
- При нажатии на кнопку "Визуализация по видам учебной нагрузки" открывается окно с круговой диаграммой данных по сегментации по видам учебной нагрузки, если до этого были загружены данные. Иначе всплывает окно «Предупреждение» с текстом «Нет данных для визуализации. Загрузите данные»

3. Обработка данных:

- При нажатии на кнопки " Визуализация по преподавателям " или " Визуализация по видам учебной нагрузки" вызывается соответствующий метод класса DisciplineVisual (visualize_by_teacher() или visualize_by_type_hour()).
- Методы visualize_by_teacher() и visualize_by_type_hour() вызывают методы класса DisciplineManager(segment_by_teacher() и segment_by_type_hour())
- Методы класса DisciplineManager(segment_by_teacher() и segment_by_type_hour()) возвращают словарь с количеством дисциплин для каждого преподавателя или вида учебной нагрузки
- Результаты сегментации используются для создания диаграммы.

4. Отображение результатов:

- Результаты сегментации отображаются в виде диаграммы.
- Полный список расчасовки дисциплин отображается в таблице tree в GUI.

4. Описание кода

discipline.py:

```
from tkinter import *
from tkinter import ttk
from tkinter import filedialog, messagebox
import matplotlib.pyplot as plt
from collections import defaultdict
class Discipline:
    def __init__(self, name, teacher, type_hour):
        self.name = name
        self.teacher = teacher
        self.type_hour = type_hour
# ... (код класса DisciplineManager)
# ... (код класса DisciplineVisual )
 if __name__ == "__main__":
    root = Tk()
    app = DisciplineVisual(root)
    root.mainloop()
```

1. Класс Discipline:
   - Метод init(self, name, teacher, type_hour):
     - Инициализирует объект Discipline с заданными параметрами.
2. Класс DisciplineManager:
   - Метод __init__(self):
     - Инициализирует список disciplines.
   - Метод load_data(self, filename):
     - Загружает данные о расчасовке дисциплин из файла с названием filename .
     - Обрабатывает возможные ошибки, например, если файл имеет неправильный формат данных.
   - Метод segment_by_teacher(self):
     - Выполняет сегментацию по преподавателям.
     - Возвращает словарь с количеством дисциплин для каждого преподавателя.
   - Метод segment_by_type_hour(self):
     - Выполняет сегментацию по видам учебной нагрузки.

- Возвращает словарь с количеством дисциплин для каждого вида учебной нагрузки.

- Метод visualize_data(self, segmentation, title):
  - Отображает результаты сегментации segmentation в виде круговой диаграммы с заголовком title.

3. Класс DisciplineVisual:

- Метод __init__(self, root):
  - Создает главное окно root.
  - Устанавливает заголовок окна.
  - Определяет размеры окна.
  - Создает кнопки "Загрузить данные", " Визуализировать по преподавателям", " Визуализировать по видам учебной нагрузки".
  - Создает надпись "Полный список дисциплин".
  - Создает таблицу с заголовками столбцов "Дисциплина", "Преподаватель", "Вид учебной нагрузки".

- Метод visualize_by_teacher(self):
  - Вызывает метод класса DisciplineManager(segment_by_teacher()).
  - По полученным из метода segment_by_teacher() данным вызывает метод класса DisciplineManager(visualize_data()).
  - Обрабатывает возможные ошибки, например, если данные не были получены.

- Метод visualize_by_type_hour(self):
  - Вызывает метод класса DisciplineManager(segment_by_type_hour()),.
  - По полученным из метода segment_by_type_hour() данным вызывает метод класса DisciplineManager(visualize_data()).
  - Обрабатывает возможные ошибки, например, если данные не были получены.

- Метод load_data_visual (self):
  - Загружает данные расчасовки дисциплин в таблицу.

- Метод load_data(self):
  - Открывает диалоговое окно для выбора пользователем файла с данными о расчасовке дисциплин.
  - Запускает метод класса DisciplineManager(load_data()).
  - Запускает метод класса DisciplineVisual(load_data_visual()).

5. Рекомендации по использованию

- Программа использует текстовый файл, выбранный пользователем для загрузки данных. Файл должен быть создан вручную и содержать информацию о расчасовки дисциплин в формате:

(Название дисциплины, Имя преподавателя, Вид учебной нагрузки дисциплины)

- Для визуализации по преподавателям или по видам учебной нагрузки достаточно нажать соответствующую кнопку.

6. Дополнительные возможности

- Добавить возможность редактирования данных о расчасовке дисциплин.
- Добавить возможность фильтрации списка расчасовки дисциплин.
- Добавить возможность вывода результатов сегментации в файл.
- Добавить возможность использования других типов диаграмм.
- Реализовать подключение к базе данных для хранения информации о расчасовке дисциплин.
- Добавьте возможность поиска по дисциплинам.
- Добавить возможность сортировки данных о расчасовке дисциплин

7. Тестирование

- Тестирование программы должно включать:
  - Проверку загрузки данных из файла.
  - Проверку сегментации по преподавателям и видам учебной нагрузки.
  - Проверку корректности отображения результатов сегментации.

8. Документация

- Данное руководство программиста является основным документом по программе.
- Дополнительную документацию можно добавить в виде комментариев к коду.
- Важно использовать стандартные соглашения по наименованию переменных и функций.
- Документация должна быть четкой, понятной и доступной для пользователя.

9. Замечания

- Программа использует библиотеку tkinter для создания GUI.
- Программа использует библиотеку matplotlib для создания диаграмм.
- Программа использует библиотеку ttk - для стилизации элементов интерфейса;
- Программа использует библиотеку defaultdict – для сегментации данных.
- Программа использует библиотеку filedialog – для выбора пользователем файла с данными.
- Программа использует библиотеку messagebox – для необходимых сообщений пользователю.
- Программа использует кодировку utf-8 для чтения данных из файла.

10. Дополнительные замечания

- Программа может быть расширена и улучшена.
- Программа может быть использована как основа для создания более сложных приложений.
- Важно следовать принципам модульности, повторного использования кода и ясности написания кода.