

**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 расписания дисциплин. Она позволяет:

- Загружать данные о расписании дисциплин из файла, выбранного пользователем.
- Просматривать список расписания 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 трёх классов:

1. Discipline: Представляет собой объект расписания 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 создаются 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 расписании 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 расписания 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 расписании дисциплин из файла с названием `filename`.
  - Обработывает 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())`.
- Обработывает 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())`.
- Обработывает возможные ошибки, например, если данные не были получены.

- Метод `load_data_visual(self)`:

- Загружает данные расписания дисциплин в таблицу.

- Метод `load_data(self)`:

- Открывает диалоговое окно для выбора пользователем файла с данными о расписании 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 расписании 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 расписании дисциплин.
- Добавить возможность фильтрации списка расписания дисциплин.
- Добавить возможность вывода результатов сегментации в файл.
- Добавить возможность использования других типов диаграмм.
- Реализовать подключение к базе данных для хранения информации о расписании дисциплин.

- Добавьте возможность поиска по дисциплинам.
- Добавить возможность сортировки данных о расписании 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.