**Руководство программиста.**

**Руководство программиста на разработку программы «Медицинские договоры».**

**1. Обзор программы**

Программа представляет собой графический интерфейс (GUI) для управления списком договоров на медицинское обслуживание. Она позволяет:

* Загружать данные о договорах из текстового файла.
* Просматривать список загруженных договоров в табличном виде.
* Выполнять сегментацию списка договоров по видам услуг.
* Визуализировать результаты сегментации по видам услуг в виде круговой диаграммы.
* Выполнять сегментацию списка договоров по врачам.
* Визуализировать результаты сегментации по врачам в виде круговой диаграммы.
* Контролировать корректность формата вводимых данных

**2. Структура программы**

Программа состоит из одного класса:

a) **Service:** Представляет собой объект договора на медицинское обслуживание. Имеет следующие атрибуты:

* manager: Имя врача, предоставляющего услугу.
* service: Название услуги.
* price: Стоимость услуги.

Также в программе определены следующие функции:

* entery(): Отвечает за загрузку данных о договорах из текстового файла, а так же проверяет корректность вводимых данных.
* pieService(): Выполняет сегментацию списка договоров по видам услуг и отображает результаты в виде круговой диаграммы.
* pieManager(): Выполняет сегментацию списка договоров по врачам и отображает результаты в виде круговой диаграммы.

**3. Алгоритм работы программы**

1. **Инициализация:**
   * При запуске программы создается главное окно (root) с элементами управления (кнопки, таблица).
   * При первом запуске программы firstRun устанавливается в True.
2. **Загрузка данных:**
   * Пользователь нажимает кнопку “Загрузить файл”.
   * Открывается диалоговое окно выбора файла формата \*.txt
   * Программа считывает данные из выбранного файла построчно, разделяя каждую строку на имя врача, название услуги и цену.
   * Создается объект класса Service для каждой корректно считанной строки.
   * Данные из файла отображаются в таблице.
3. **Сегментация данных:**
   * Пользователь нажимает кнопку “Сегментация по видам услуг” или “Сегментация по врачам”.
   * Вызывается соответствующая функция pieService() или pieManager().
   * Функции выполняют сегментацию данных, подсчитывая суммарную стоимость услуг по видам или врачам
   * Данные используются для построения круговой диаграммы.
4. **Отображение результатов:**
   * Круговая диаграмма, отображающая результаты сегментации, выводится на экран.
   * В основном окне программы отображается таблица со всеми загруженными данными (Имя врача, услуга, цена).
   * Если при чтении файла возникли ошибки неверного формата выводится сообщение.
   * При повторной загрузке файла выводится запрос на удаление старых данных

**4. Описание кода**

Основная логика программы содержится в одном файле, который включает в себя функции и класс Service, а также GUI, построенный с использованием tkinter.

**1. Класс Service:** \* \_\_init\_\_(self, manager, service, price): Конструктор класса, инициализирует объект Service с заданными параметрами: manager (имя врача), service (название услуги) и price (цена услуги). Также вызывает метод update\_counts для обновления статистики. \* update\_counts(self, manager, service): Метод класса, обновляет словари manager\_counts и service\_counts, добавляя цену услуги к суммарной стоимости по врачу и по услуге. \* count\_manager\_instances(cls, manager): Метод класса, возвращает суммарную стоимость оказанных услуг по конкретному врачу. \* count\_service\_instances(cls, service): Метод класса, возвращает суммарную стоимость оказанных услуг по конкретной услуге. \* unique\_managers(cls): Метод класса, возвращает список уникальных имен врачей. \* unique\_services(cls): Метод класса, возвращает список уникальных названий услуг.

**2. Функции:**

* entery():
  + Открывает диалоговое окно для выбора текстового файла с данными о договорах.
  + Читает данные из файла построчно, разделяя каждую строку на имя врача, название услуги и цену.
  + Создает объект класса Service для каждой корректно считанной строки.
  + Добавляет данные в таблицу.
  + Проверяет, что цена это именно число с плавающей точкой.
  + Выводит сообщение об ошибке, если формат данных не верен.
* Выводит сообщение если файл пустой. \* Предлагает стереть старые данные, если файл открывается повторно.
* pieService():
  + Выполняет сегментацию списка договоров по видам услуг.
  + Подсчитывает суммарную стоимость услуг по каждому виду.
  + Отображает результаты сегментации в виде круговой диаграммы с подписями, отображающими суммарные стоимости.
* pieManager():
  + Выполняет сегментацию списка договоров по врачам.
  + Подсчитывает суммарную стоимость услуг каждого врача.
  + Отображает результаты сегментации в виде круговой диаграммы с подписями, отображающими суммарные стоимости.

**3. Графический интерфейс (GUI):**

* Создается главное окно root с заданными размерами и заголовком.
* Кнопки “Загрузить файл”, “Сегментация по видам услуг” и “Сегментация по врачам” для взаимодействия с пользователем.
* Таблица для отображения загруженных данных.
* В обработчиках нажатий на кнопки вызываются соответствующие функции, описанные выше.

**5. Рекомендации по использованию**

* Программа ожидает данные в текстовом файле, выбранном пользователем при нажатии на кнопку “Загрузить файл”. Файл должен содержать информацию о договорах на медицинское обслуживание в формате: "<Имя врача>, <Название услуги>, <Цена>". Каждая запись должна находиться на отдельной строке.
* При повторном открытии файла старые данные стираются, но при этом программа выведет запрос на подтверждение.
* Для загрузки данных о договорах, нажмите кнопку “Загрузить файл” и выберите нужный текстовый файл.
* После загрузки данных, они отобразятся в таблице.
* Для просмотра сегментации по видам услуг, нажмите кнопку “Сегментация по видам услуг”. Отобразится круговая диаграмма, показывающая долю стоимости каждого вида услуги от общего оборота.
* Для просмотра сегментации по врачам, нажмите кнопку “Сегментация по врачам”. Отобразится круговая диаграмма, показывающая долю стоимости услуг каждого врача от общего оборота.
* Программа автоматически обрабатывает ошибки формата данных в файле, выводя предупреждение в случае некорректных данных.
* Программа выведет ошибку, если файл пустой.

**6. Дополнительные возможности**

* Добавить возможность сохранения загруженных данных в файл.
* Добавить возможность редактирования данных о договоре непосредственно в таблице.
* Добавить возможность сортировки данных в таблице по разным столбцам.
* Реализовать возможность сохранения диаграмм в виде изображений.
* Добавить возможность выбора типа диаграмм для визуализации данных (например, гистограмма).
* Реализовать поиск по таблице.
* Добавить возможность вывода более подробной информации о каждой услуге и враче.

**7. Тестирование**

Тестирование программы должно включать:

* **Проверку загрузки данных из файла:**
  + Убедиться, что программа корректно загружает данные из текстового файла, выбранного пользователем.
  + Проверить, что программа обрабатывает пустой файл, а также сообщает об ошибке в случае неверного формата данных в файле.
  + Проверить, что при повторном открытии файла, программа запрашивает подтверждение на удаление старых данных.
* **Проверку корректности отображения данных в таблице:**
  + Убедиться, что данные отображаются в таблице корректно.
* **Проверку сегментации по видам услуг и по врачам:**
  + Убедиться, что сегментация по видам услуг и по врачам работает корректно.
  + Проверить, что на круговой диаграмме отображаются правильные данные о доле стоимости услуг.
  + Убедиться, что подписи к секторам диаграммы содержат верные названия и процентные соотношения.
* **Проверку обработки ошибок:**
* Проверить что программа корректно обрабатывает все возможные ошибки и выводит сообщение пользователю.

**8. Документация**

* Данное руководство программиста является основным документом по программе.
* Дополнительные пояснения можно найти в комментариях к коду.
* В коде используются стандартные соглашения по именованию переменных и функций, чтобы сделать его понятным и читаемым.
* Документация стремится быть четкой, понятной и доступной для пользователей с разным уровнем технической подготовки.
* При внесении изменений в код, рекомендуется обновлять и дополнять документацию для поддержания ее актуальности.
* Если вы столкнулись с неясностями или возникли вопросы по работе программы, обратитесь к комментариям в коде или обратитесь за помощью к разработчику.

Отлично, вот разделы “Замечания” и “Дополнительные замечания” для твоего кода:

**9. Замечания**

* Программа использует библиотеку tkinter для создания графического интерфейса (GUI).
* Программа использует библиотеку matplotlib для визуализации данных в виде круговых диаграмм.
* Программа обрабатывает ошибки при чтении данных из файла, включая неверный формат и пустой файл.
* Данные о договорах хранятся только в оперативной памяти и не сохраняются в файл.
* Программа не использует базы данных, работа происходит только с данными в оперативной памяти.

**10. Дополнительные замечания**

* Программа может быть расширена для добавления новых функций, таких как сохранение данных в файл, возможность редактирования данных и другие типы визуализации данных.
* Программа может быть улучшена за счет добавления более информативного графического интерфейса.
* Код программы написан таким образом, чтобы его было легко читать и модифицировать.
* Важно помнить о принципах модульности, повторного использования кода и ясности при дальнейшей разработке и расширении функционала программы.
* Рекомендуется использовать PEP8 при написании кода