Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „ФИТА“

**Документация на проект**

по дисциплината „**Обектно ориентирано програмиране - част 1**”

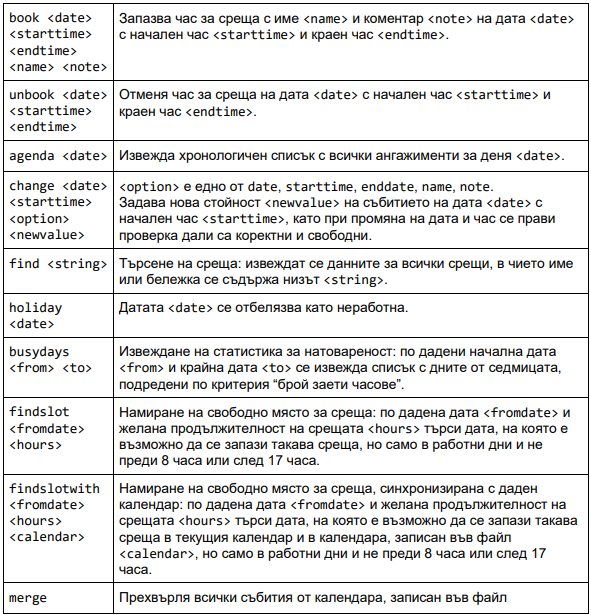
на тема: „**Личен календар**”

|  |  |
| --- | --- |
| Изготвил: Николай Веселинов Георгиев |  |
| Специалност: СИТ |  |
| Група: 2 б) |  |
| Факултетен номер: 23621689 |  |

2025

**Задание на проекта**

Да се напише програма, реализираща информационна система, която поддържа личен календар, като го записва във файл. След като приложението отвори даден файл, то трябва да може да извършва посочените по-долу операции, в допълнение на общите операции (open, close, save, save as, help и exit):



**Глава 1: Увод**

* 1. **Описание и идея на проекта**

Проектът „Личен календар“ представлява Java конзолно приложение за управление на събития, празници и различни календари. Чрез команден ред (CLI) потребителят може да си записва и да търси срещи, събития и празници, да определя кога има свободни часове, както и справки за ден или период от време.

Основната идея на проекта е да се демонстрира използването на обектно-ориентирано програмиране (ООП) чрез реализация на реална система с ясна структура, модулност и лесно разширение.

* 1. **Цел и задачи на разработката**

Целта на проекта е изграждане на цялостна, разширяема и функционална система за работа със събития и текстови файлове, реализирана чрез ООП на Java. Основните задачи включват:

* Реализация на обекти, отразяващи реални концепции: събития, празници, разпределение на времето през деня
* Създаване на потребителски интерфейс, чрез команден ред
* Поддръжка на действия като добавяне и премахване на срещи, легенда за деня или даден период
* Изграждане на командна система чрез интерфейси и отделни класове за всяка операция
* Запис и зареждане на информация от и към текстови файл
* Улеснена поддръжка и бъдещо разширяване на проекта чрез използване на шаблони за проектиране
  1. **Структура на документацията**

**Документацията е структурирана в следните глави:**

* **Глава 1: Увод**

Представя основната идея на проекта, цели и задачи на разработката

* **Глава 2: Преглед на предметната област**

Описва концепциите, моделите и използваните шаблони

* **Глава 3: Проектиране**

Анализиране на структурата

* **Глава 4: Реализация на класове и команди**

Подробно описание на всички класове и тяхната функционалност

* **Глава 5: Тестване**

Преглед на реални входни данни и валидиране на функционалностите

* **Глава 6: Заключение**

Оценка на резултатите, придобити знания и възможности за развитие

* **Използвана литература**

Източници, използвани при реализацията и анализа

**Глава 2: Преглед на предметната област**

* 1. **Основни концепции и обекти**

Проектът „Личен календар“ реализира система за управление на срещи, почивки и свободно време чрез команден ред. Основните концепции са:

**Събития като текстови записи** – Всяко събитие се представя като текстов ред, съдържащ дата, начален и краен час, заглавие и описание. Те се съхраняват във файлове, групирани по дати чрез структура Map<String, List<String>>.

**Управление на файлове** – Класът FileHandler реализира четене, запис и изтриване на събития от файл. Поддържа зареждане на календар при open, запис при save, saveas и обединяване при merge.

**Командна система** – Всяка операция е имплементирана като клас, реализиращ интерфейса Command. Командите обработват въведени от потребителя аргументи и изпълняват действия като:

* + BookCommand – добавя ново събитие
  + UnbookCommand – премахва събитие
  + AgendaCommand – извежда дневен план
  + FindCommand, HolidayCommand, MergeCommand и др.

**Интерфейс за взаимодействие (CLI)** – Класът CLI приема текстови команди от потребителя, парсва ги и предава към подходящата инстанция на Command.

**Формат на данните** – Всеки ред във файла представлява едно събитие със следната структура: yyyy-MM-dd HH:mm HH:mm <заглавие> <описание>

**Работа с дати и часове** – Използват се стандартните Java класове LocalDate, LocalTime, DateTimeFormatter за валидиране на въведени дати и времена.

### **2.2 Използвани шаблони за проектиране (Design Patterns)**

**Command Pattern**  
Всеки клас, реализиращ интерфейса Command, представлява отделна команда с метод execute(String[] args). Това позволява лесно добавяне на нови функционалности без промяна в централната логика.

**Factory-like подход чрез CommandHandler / CLI**  
Макар да няма клас CommandFactory, логиката за избор на команда е реализирана чрез метода getCommand() в CommandHandler. Това изпълнява ролята на фабрика за команди.

**Интерфейси за абстракция**  
Интерфейсът FileOperations дефинира методите за файлово взаимодействие (load, save, deleteEvent). Това позволява различни реализации при нужда.

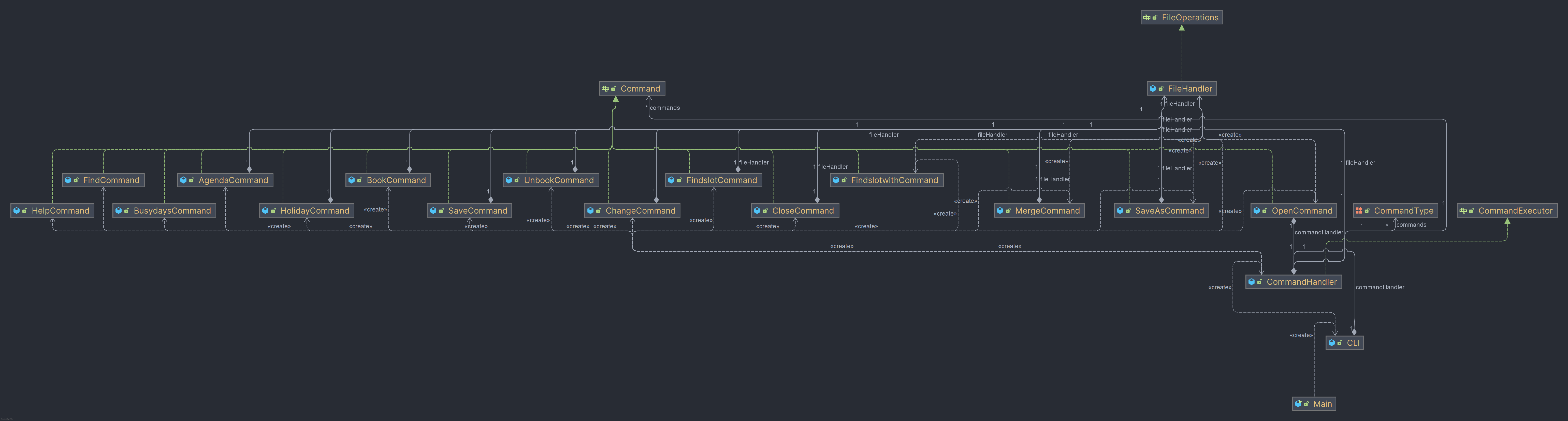
**Глава 3: Проектиране**

**3.1 Общ преглед на архитектурата**

Проектът следва модулен и обектно-ориентиран дизайн, при който основните отговорности са разпределени между няколко логически слоя:

* **Потребителски интерфейс (CLI слой)**  
  Класът CLI приема команди от потребителя, валидира входа и го препраща към CommandHandler.
* **Обработка на команди (Command слой)**  
  Пакетът commands съдържа всички операции като book, unbook, agenda, find, merge, реализирани като отделни класове. Всеки от тях имплементира интерфейса Command.
* **Файлов слой (filehandler)**  
  Класът FileHandler обработва четенето и писането към файл чрез интерфейса FileOperations. Позволява централизирано и разширяемо управление на данните.
* **Бизнес логика и структура на данните**  
  Въпреки че няма специални класове за „събитие“ или „календар“, проектът използва структура Map<String, List<String>> (дата → списък със събития за тази дата), което осигурява ефективен достъп до нужната информация.

**3.2 Класова диаграма (описание)**



* CLI → използва CommandHandler
* CommandHandler → връща обекти от тип Command
* Command → интерфейс, реализиран от:
  + BookCommand, UnbookCommand, AgendaCommand, HolidayCommand, MergeCommand, FindCommand, OpenCommand, SaveCommand, SaveAsCommand, ChangeCommand, FindslotCommand, FindslotwithCommand, CloseCommand, HelpCommand
* FileHandler → имплементира FileOperations

**3.3 Структура на файловете**

Проектът е организиран в следните основни пакети:

| **Пакет** | **Отговорност** |
| --- | --- |
| cli | Обработва потребителски вход |
| commands | Съдържа всички команди, базирани на Command |
| filehandler | Съдържа логиката за работа с файловата система |
| src/Main.java | Начална точка на изпълнение |
|  |  |

**Глава 4: Реализация на основни класове и команди**

**Main.java**

**Описание:**

Класът Main съдържа входната точка на програмата. Стартира приложението, като създава екземпляр на класа CLI.

**Методи:**

* - public static void main(String[] args) – стартира командния интерфейс на системата.

**CLI.java**

**Описание:**

Класът CLI реализира потребителския интерфейс чрез команден ред (CLI). Чете команди от потребителя, предава ги към CommandHandler и извиква изпълнението им.

**Полета:**

* - Scanner scanner – обект за четене от конзолата.
* - CommandHandler commandHandler – обработчик на командите.
* - Map<String, List<String>> events – основната структура с всички събития (по дати).

**Методи:**

* - start() – основен метод, който стартира непрекъснато четене и обработка на команди.

**FileHandler.java**

**Описание:**

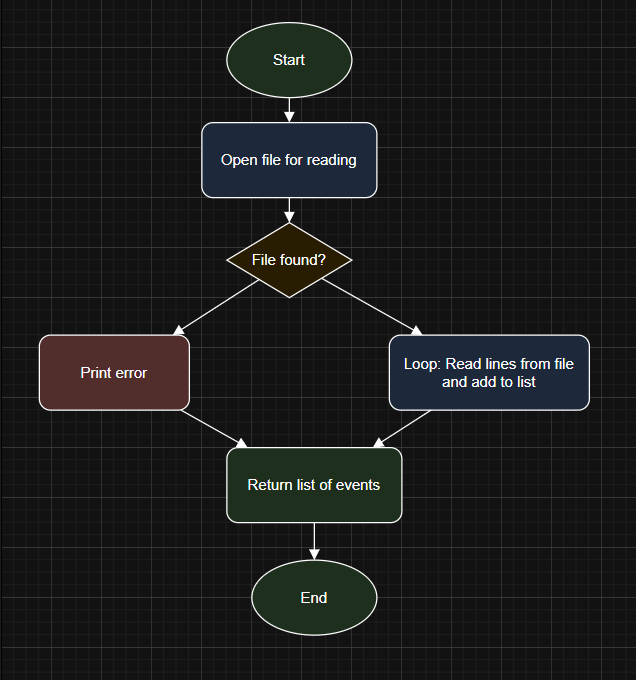
Класът FileHandler реализира четене и запис на събития от/към текстов файл.

**Полета:**

* - String filePath – път към файла с календар.

**Методи:**

* - load() – връща списък със събития от файла.
* - save(String event) – записва ново събитие.
* - save(List<String> events) – записва списък от събития.
* - deleteEvent(String event) – премахва конкретно събитие от файла.



**FileOperations.java**

**Описание:**

Интерфейс, дефиниращ действията, които трябва да поддържа всеки клас за файлов достъп.

**Методи:**

* - List<String> load()
* - void save(String event)
* - void save(List<String> events)
* - void deleteEvent(String event)

**Command.java**

**Описание:**

Интерфейс за всички команди, които могат да се подават през CLI.

**Методи:**

* - void execute(String[] args) – изпълнява конкретната команда с подадените аргументи.

**CommandHandler.java**

**Описание:**

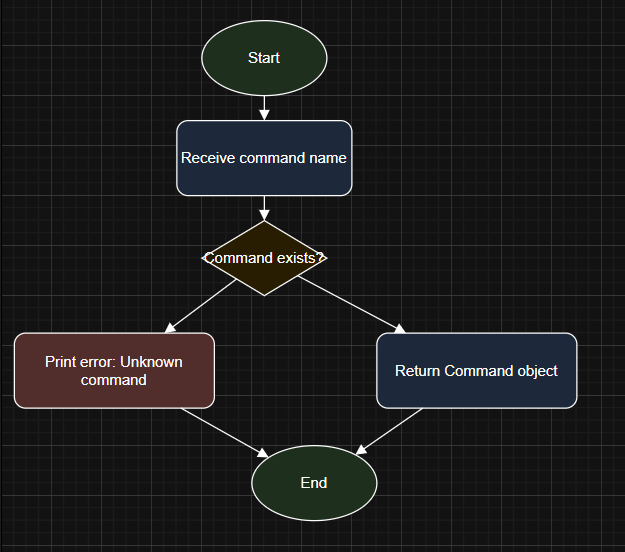
Класът CommandHandler избира подходящата команда според потребителския вход.

**Полета:**

* - Map<String, Command> commands – регистър с всички налични команди.
* - FileHandler fileHandler – за операции с файл.
* - Map<String, List<String>> events – текущите събития.

**Методи:**

* - getCommand(String commandName) – връща съответната команда.
* - setFileHandler(FileHandler handler) – сменя активния файлов мениджър.



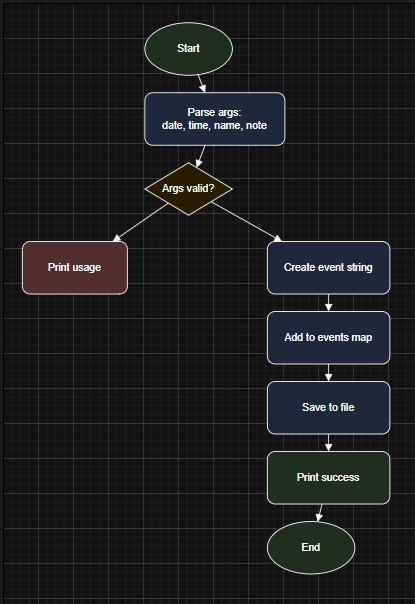
**BookCommand.java**

**Описание:**

Добавя ново събитие в календара с определена дата, време, име и бележка.

**Методи:**

* - execute(String[] args) – валидира данните, създава събитие и го записва.



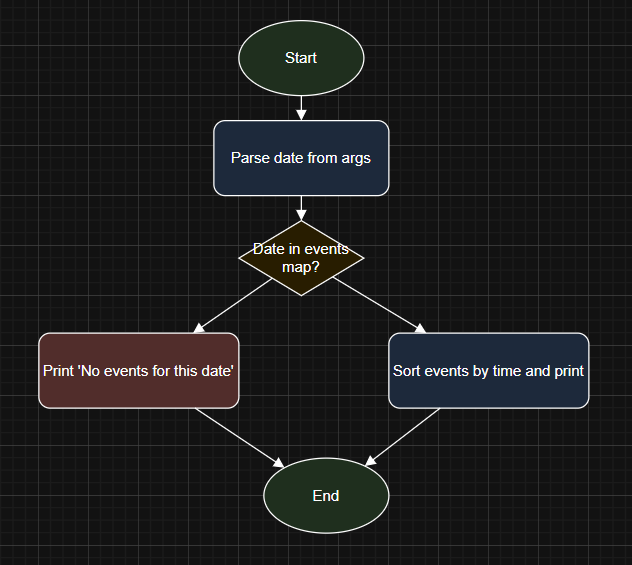
**AgendaCommand.java**

**Описание:**

Извежда всички събития за подадена дата в хронологичен ред.

**Методи:**

* - execute(String[] args) – намира и извежда събитията за конкретната дата.



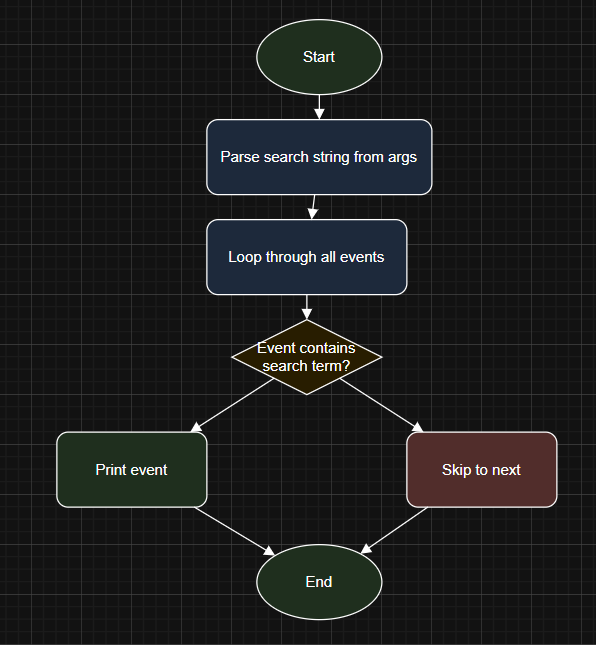
**FindCommand.java**

**Описание:**

Търси низ във всички събития и извежда резултатите, в които той се съдържа.

**Методи:**

* - execute(String[] args) – филтрира и извежда съвпадения.



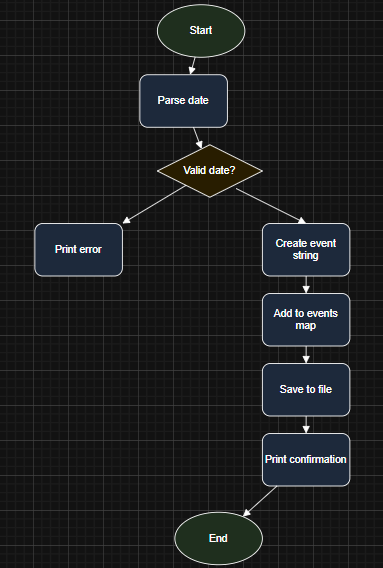
**HolidayCommand.java**

**Описание:**

Маркира дадена дата като празник, като създава събитие със специфично обозначение.

**Методи:**

* - execute(String[] args) – създава празнично събитие.



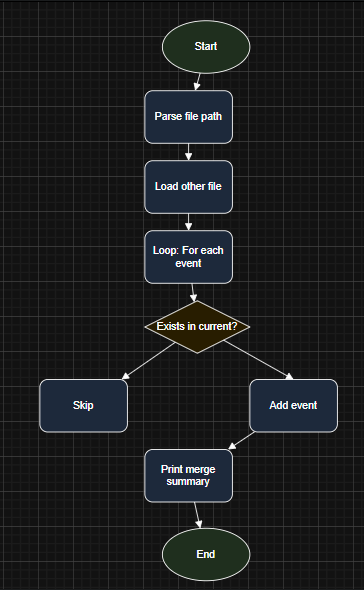
**MergeCommand.java**

**Описание:**

Обединява събития от друг календарен файл с текущия, като добавя несъществуващите.

**Методи:**

* - execute(String[] args) – зарежда външен файл и прехвърля събития.



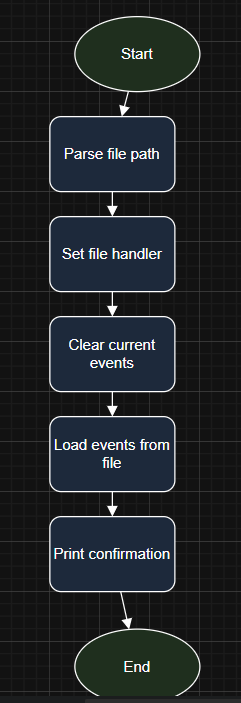
**OpenCommand.java**

**Описание:**

Зарежда календар от файл. Ако файлът не съществува, се създава нов.

**Методи:**

* - execute(String[] args) – задава нов файл и зарежда данните от него.



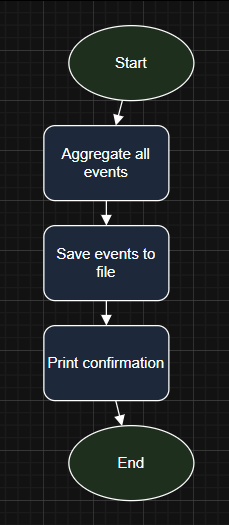
**SaveCommand.java**

**Описание:**

Записва текущите събития във файла, свързан с отворения календар.

**Методи:**

* - execute(String[] args) – обединява всички събития и ги записва.



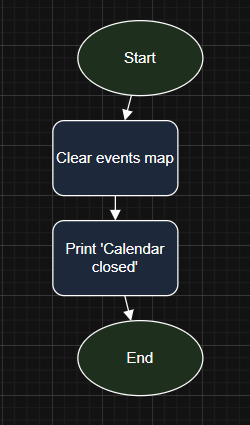
**CloseCommand.java**

**Описание:**

Изчиства всички текущи събития от паметта и „затваря“ календара.

**Методи:**

* - execute(String[] args) – изпразва структурата events.



**HelpCommand.java**

**Описание:**

Извежда списък с всички налични команди и техния синтаксис.

**Методи:**

* - execute(String[] args) – показва помощна информация.

**Глава 5: Тестване**

**5.1 Метод на тестване**

Тестването на приложението „Личен календар“ се извършва чрез ръчно въвеждане на команди в терминал (CLI). Потребителят взаимодейства с програмата чрез текстови инструкции, като всяка команда се валидира и изпълнява в реално време.

Стъпки за тестване:

1. Стартиране на приложението чрез изпълнение на класа Main.

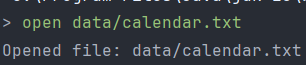
2. Въвеждане на команди в конзолата.

3. Наблюдение на резултатите и валидация чрез визуален изход.

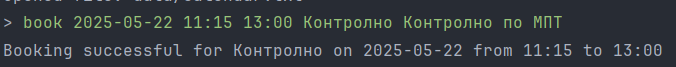
4. Тестване на гранични случаи – невалидни данни, липсващи файлове, дублиращи се събития и др.

**5.2 Тествани команди и резултати**

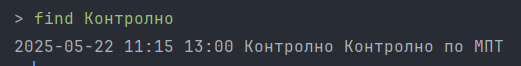
open



book



find



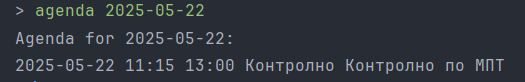
holiday



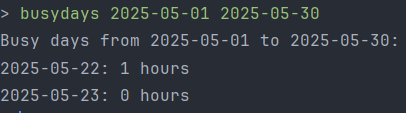
save



agenda



busydays



**Глава 6: Заключение**

**Оценка на резултатите**

Проектът „Личен календар“ изпълнява успешно заложените цели – създаване на гъвкаво и разширяемо Java приложение, което позволява на потребителя да управлява календарни събития през команден интерфейс. Всички основни команди, като запис, изтриване, търсене, обединяване и извеждане на информация, функционират коректно и демонстрират обектно-ориентирания подход чрез добре разделени отговорности и използване на интерфейси и шаблони.

Системата е устойчива на невалидни входове и дава ясни съобщения за грешки, което я прави подходяща за реална употреба. Също така, модулната ѝ структура улеснява бъдещо разширяване и поддръжка.

**Придобити знания**

По време на разработката на проекта бяха усвоени и затвърдени ключови принципи и практики в обектно-ориентираното програмиране:

* Работа с интерфейси и абстракции за осигуряване на гъвкавост
* Прилагане на шаблона Command за командна обработка
* Разделяне на логика в отделни модули (UI, файлов достъп, логика)
* Работа с колекции и стандартни Java класове като Map, List, LocalDate, LocalTime
* Обработка на изключения и гранични случаи

**Възможности за развитие**

Проектът може да бъде надграден по няколко начина:

* Добавяне на графичен потребителски интерфейс (GUI) за по-широка достъпност
* Използване на база данни вместо текстов файл за по-ефективно съхранение
* Създаване на мобилна или уеб версия чрез подходящи технологии
* Автоматични напомняния чрез имейл или нотификации
* Поддръжка на повтарящи се събития, категории и тагове

**Използвана литература**

-Университетски материали по ООП (лекции и упражнения) Конкретни концепции като обектно моделиране, йерархии от класове, и шаблони (patterns), които са използвани при създаването на архитектурата на приложението

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/temporal/ChronoUnit.html>

-StackOverflow