Документация на проект **БИБЛИОТЕКА**

Автор: Александрина Бончева, специалност Информатика, факултетен номер 45344

Структура на документацията:

*1. Увод.*

*2. Списък на имплементираните класове в решението,подреден по ниво на абстракция.*

*3. Архитектура. (UML диаграма)*

*4. Подробно описание на използваните класове – идея, член-методи, член-данни, връзки между тях.*

*5. Допълнителна информация за ходa на изпълнение на програмата.*

*6. Идеи за оптимизация и подобрения.*

1. *Увод*

Представеното решение реализира информационна система,която се използва при поддръжката на библиотека.Съхраняват се и се управляват,както наличните книги,така и наличните потребители.

Целта на разработката е да предостави оптимално решение за бързо,лесно и разнообразно управление на данните.

1. *Списък на имплементираните класове в решението,подредени по ниво на абстракция.*

1. Book

2. User

3. BooksManager

4. UsersManager

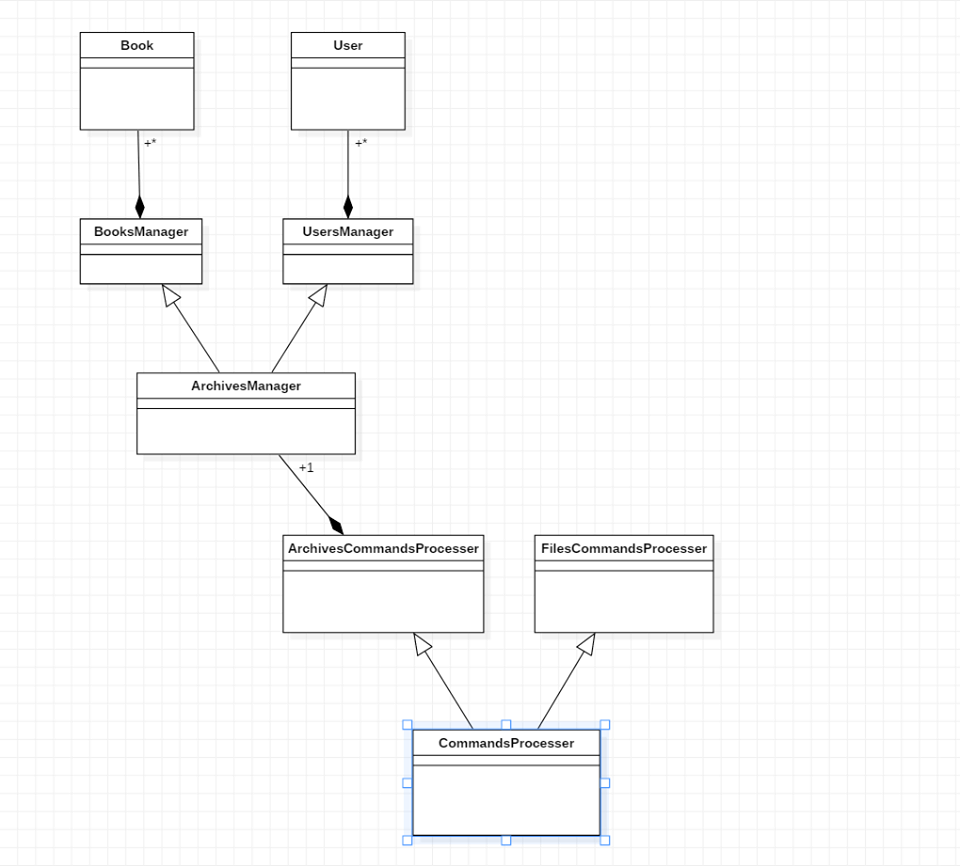
5. ArchivesManager

6. ArchivesCommandsProcesser

7. FilesCommandsProcesser

8. CommandsProcesser

1. *Архитектура (UML диаграма)*



1. *Подробно описание на използваните класове – идея,член-методи,член-данни,връзки межу тях.*

-Book – класът отговаря за представянето на представянето на книга в паметта.

*Публични член-методи:*

Book() – конструктор

Book& makeDummyBookFromConsole() – създава книга,като инициализира само заглавието и автора,прочетени от стандартния вход.

Book& makeDummy(const BookSearchCriteria&, const std::string&) – създава книга по зададен критерий,като инициализира само тази член-данна,която отговаря на критерия. Критериите са: заглавие,автор,таг.

Използва се при търсене на книга в архива.

Book& makeDummy(const BookSearchCriteria&, const unsigned& num); - създава книга по зададен критерий,като инициализира само тази член-данна,която отговаря на критерия. Критериите са: идентификационен номер и година на издаване.

void showMainInfo() const – извежда на стандартния изход основната информация за книгата.

void showAllInfo() const – извежда подробна информация за книгата.

int compare(const BookSearchCriteria&, const Book&) const – сравнява две книги по даде критерий.

bool operator==(const Book&) const – сравнява две книги по автор и заглавие.

bool empty() const – проверява дали даден обект е инициализиран.

friend std::ostream& operator<<(std::ostream&, const Book&) – предефиниран operator<<.

friend std::istream& operator>>(std::istream&, Book&) – предефиниран

operator>>.

*Частни член-методи:*

void setId() – инициализира всяка книга с уникален идентификационен номер.

void showKeyWords() const – извежда на стандартния изход ключовите думи за дадена книга.

bool findTag(const std::vector<std::string>&) const – проверява дали дадена книга съдържа някой от таговете,подадени на метода като аргумент.

int compareIntegers(unsigned, unsigned) const – сравнява две целочислени числа.

int compareRate(double) const – сравнява две дробни числа.

std::string toLower(const std::string&) const – трансформира всички главни букви на даден стринг в малки.

int compareTitle(const Book&) const – сравнява две книги по заглавия.

int compareAuthor(const Book&) const – сравнява две книги по автор.

*Частни член-данни:*

std::string author – съхранява автор.

std::string title – съхранява заглавие.

std::string genre – съхранява жанр;

std::string description – съхранява описание.

unsigned publishedIn – съхранява година на издаване.

std::vector<std::string> keyWords – съхранява ключови думи.

double rate – съхранява рейтинг.

unsigned id – съхранява уникален идентификационен номер.

-User – класът отговаря за представянето на потребител в паметта*.*

*Публични член-методи:*

bool empty() const - проверява дали обектът е инициализиран.

bool isActive() const – проверява дали потребителят има започната сесия.

bool isAdministrator() const – проверява дали потребителят има администраторски права.

bool isDefaultAdmin() const – проверява дали потребителят е администраторът по подразбиране.

void activateSession() – активира сесията на потребителя.

void endSession() – прекратява сесията на потребителя.

bool operator==(const User&) const – сравнява двама потребители по име и парола.

User getDefaultAdmin() – създава потребител с име и парола по подразбиране.

const std::string& getUsername() const – връща името на потребителя.

friend std::ostream& operator<<(std::ostream&, const User&) – предефиниран operator<<.

friend std::istream& operator>>(std::istream&, User&) – предефиниран оператор>>.

*Частни член-методи:*

void setDefaultUsername() – инициализира името на потребител с име по подразбиране.

void setDefaultPassword() – инициализира паролата на потребител с парола по подразбиране.

*Частни член-данни:*

bool loggedIn – съхранява дали потребител е в активна сесия.

bool isAdmin – съхранява дали потребител е с администраторски права.

std::string username – съхранява потребителско име.

std::string password – съхранява парола.

-BooksManager – класът отговаря за съхранение на архив с книги и позволва извършването на операции върху него.

*Публични член-методи:*

void showAllBooks(const int desc = 0) const – извежда информация на стандартния изход за всички книги в архива във възходящ или низходящ ред според зададения аргумент.

template<typename T>

void findBookBy(const BookSearchCriteria&, const T&) const – търси книга по зададени критерий и стойност на търсената член-данна.

Търсенето не е чувствително към главни и малки букви.

void addBook(const Book&) – добавя книга към архива.

void removeBook(const Book&) – премахва книга от архива.

void setBooks(const std::vector<Book>&) – инициализира архива с книги със стойнстта, подадена като аргумент.

void freeBooks() – изтрива всички книги от архива.

void ascendingSortBooks(const BookSearchCriteria&) – сортира архива от книги във възходящ ред според зададен критерий.

const std::vector<Book>& getBooks() const – връща колекция от всички книги в архива.

*Частни член-методи:*

void printFromBeg() const – извежда на стандарния изход всички книги от архива,обхождайки го от началото към края.

void printFromEnd() const – извежда на стандартния изход всички книги от архива,обхождайки го от края към началото.

void mergeArrays(int, int, int, const BookSearchCriteria&) – помощна функция на void mergeSort(int, int, const BookSearchCriteria&).

void mergeSort(int, int, const BookSearchCriteria&) – сортира архива от книги по метода „Сортиране чрез сливане“. Получава начален индекс,краен индекс и критерий на сортиране.

*Частни член-данни:*

std::vector<Book> allBooks – колекция,в която се съхраняват всички книги в системата.

-UsersManager – класът отговаря за съхранение на архив с потребители и позволва извършването на операции върху него.

*Публични член-методи:*

UsersManager() – конструктор.

void logIn(const User&) – активира сесията на даден потребител.

void logOut() – прекратява сесията на даден потребител.

void addUser(const User&) – регистрира нов потребител в архива с потребители.

void removeUser(const User&) – премахва потребител от архива с потребители.

void setUsers(const std::vector<User>&) – инициализира колекцията от потребители със стойността,подадена като аргумент.

const std::vector<User>& getUsers() const – връща колекция от всички потребители в архива.

*Защитени член-данни:*

bool activeUser – съхранява дали в системата има активен потребител.

bool activeAdmin – съхранява дали в системата има активен администратор.

*Частни член-методи:*

User& findActiveUser() – намира активния потребител в колекцията.

void removeUserHelper(const User&) – помощна функция на void removeUser(const User&).

*Частни член-данни:*

std::vector<User> allUsers – колекция,в която се съхраняват всички потребители в системата.

-ArchivesManager - наследява BooksManager и UsersManager. Отговаря за извършването на действия с архивите от книги и потребители. Позволява външният свят да не знае кой архив и по какъв начин се манипулира.

*Публични член-методи:*

*Собствените методи на класа предефинират някои от наследените от BooksManager. В тях предоставената функционалност се допълва,като се взима предвид наличието на активен потребител или администратор.*

void showAllBooks(const int desc = 0) const;

template<typename T>

void findBook(const BookSearchCriteria&, const T&) const;

void addBook(const Book&);

void removeBook(const Book&);

void ascendingSortBooks(const BookSearchCriteria&);

-ArchivesCommandsProcesser – класът отговаря за изпълнението на команди,получени от стандартния вход,свързани с манипулация на архивите.

*Публични член-методи:*

ArchivesCommandsProcesser(ArchivesManager& archives) : archivesInSystem(&archives) {} – конструктор.

*Защитени член-методи:*

void processLogin() – изпълнява команда ‘login’.

void processLogout() const – изпълнява команда ‘logout’.

void processBooksAll() – изпълнява команда ‘books all’.

void processBooksFind(); - изпълнява команда ‘books find’.

void processBooksSort() – изпълнява команда ‘books sort’.

void processBooksInfo() const – изпълнява команда ‘books info’.

void processBooksAdd() – изпълнява команда ‘books add’.

void processBooksRemove() – изпълнява команда ‘books remove’.

void processUsersAdd() – изпълнява команда ‘users add’.

void processUsersRemove() – изпълнява команда ‘users remove’.

void setBooks(const std::vector<Book>&) – инициализира колекцията от книги.

void setUsers(const std::vector<User>&) – инициализира колекцията от потребители.

void deleteBooksFromMemory() – изпълнява команда ‘close’

const std::vector<Book>& getBooks() const – връща стойността на колекцията от книги.

const std::vector<User>& getUsers() const – връща стойността на колекцията от потребители.

*Частни член-данни:*

ArchivesManager\* archivesInSystem;

-FilesCommandsProcesser – класът отговаря за изпълнението на команди,получени от стандартния вход,свързани с манипулация на файлове.

*Защитени член-методи:*

void openBooksFile() – отваря файла,съхраняващ книгите.

void openUsersFile() – отваря файла,съхраняващ потребителите.

std::vector<Book> deserializeBooksFileAndClose() – десериализира файла,съхраняващ книгите,и го затваря.

std::vector<User> deserializeUsersFileAndClose() – десериализира файла,съхраняващ потребителите,и го затваря.

void serializeBooks(const std::vector<Book>&, const std::string& fileName = "") – сериализира архива от книги.

void serializeUsers(const std::vector<User>&) – сериализира архива от потребители.

*Защитени член-данни:*

std::string path – съхранява път към даден файл.

*Частни член-данни:*

const std::string usersFile = "Users.txt" – файлът с потребители (един за цялата система).

std::ifstream inputBooks – стрийм за четене на книги от файл.

std::ofstream outputBooks – стрийм за писане на книги във файл.

std::ifstream inputUsers – стрийм за четене на потребители от файл.

std::ofstream outputUsers – стрийм за писане на потребители във файл.

-CommandsProcesser – наследява FilesCommandsProcesser и ArchivesCommandsProcesser. Отговаря за прочитането на команди от стандартния вход и изпълнението им.

*Публични член-методи:*

CommandsProcesser(ArchivesManager& archives) – конструктор.

~CommandsProcesser() – деструктор.

void proccessCommand() – прочита команда от стандартния вход и я изпълнява.

*Частни член-методи:*

void expectCommand() – прочита команда от стандартния вход.

void extractBooksFromFile() – прочита файла с книги и инициализира колекцията.

void exctractUsersFromFile() – прочита файла с потребители и инициализира колекцията.

void deleteBooksFromMemory() – предефинирана функция,наследена от ArchivesCommandsProcesser.

void saveBooksToFile() – записва книгите от колекцията във файла,от когото са били заредени.

void saveBooksToNewFile() – записва книгите от колекцията в нов файл.

void saveUsersToFile() – записва колекцията от потребители във файла с потребители.

*Частни член-данни:*

std::string command – съхранява команда.

std::string option – съхранява опция към дадена команда.

1. *Допълнителна информация за ходa на изпълнение на програмата.*

При първоначално стартиране на програмата автоматично се създава файл ‘Users.txt’,ако той не съществува,в който ще се съхраняват всички потребители.При всяко следващо изпълнение този файл се зарежда в паметта и колекцията от потребители се инициализаира със съответното съдържание.При прекратяване на изпълнението архивът с потребители се записва във файла.След първото изпълнение във файла ще бъде записан само потребителят по подразбиране,ако не бъдат добавени други.

При ‘login’ се въвеждат последователно име,парола и ниво на достъп на съответния потребител ( 1 – администратор, 0 – обикновен потребител). По същия начин потребителите са записани във файла с потребители. \*Потребителят по подразбиране не може да бъде изтрит.

Командите ‘open’ и ‘close’ отговарят само за зареждането на данни за книги в паметта. Книга може да бъде въведена или от файл,или от стандартния вход. Въвеждането от стандартния вход трябва да стане по същия начин,както е форматирано съдържанието на файла (Всеки атрибут на нов ред с изключение на ключовите думи. Те са изброени една след друга на един ред,като в началото на реда е указан техният брой.)

След въвеждане на команда ‘books remove’ се натиска “Enter”.

Чак тогава ще бъдат изведени съобщения за въвеждане на заглавие и автор на книгата,която предстои да бъде премахната.

В архива,който съхранява кода на програмата и тази документация,е приложен и примерен файл с информация за книги.

1. *Идеи за оптмизация и подобрения.*

* Класът ArchivesCommandsProcesser може да бъде разделен на два други – такъв,отговарящ за изпълнението на команди,свързани с потребители,и друг,отговарящ за изпълнението на команди,свързани с книги.

- Проверка дали архивът от потребители е променян.Ако не е,записването във файла при изход от програмата е излишно.

- Пренареждане на член-данните на класовете с цел по-добро подравняване в паметта.

- Обработване на грешки.

- Добавяне на нови функционалности.

Пример: смяна на парола (за потребител), добавяне на ревю за дадена книга от съответен потребител, добавяне на нов архив от автори.