Документация на Library Project

Светослав Цветков, 4MI0700207

1. Анализ на задачата и подход за решение

1.1. Задача

Система за библиотека, която поддържа - библиотечни единици: книги, периодични издания, поредици; потребители: читатели и администратори. Командите включват: добавяне, търсене, извеждане (със странициране), вземане/връщане на книги, управление на потребители.

1.2. Основни предизвикателства

1. **Голям обем данни**: Добавяне при нужда, За да не зареждат всички единици в паметта в самото начало.
2. **Хетерогенни обекти**: различни наследници на `LibraryUnit` и `User` в един контейнер.
3. **Интерактивен интерфейс**: писане на произволни команди и съответствено показване на резултати
4. **Надеждна устойчивост**: сериализация/десериализация със strong exception safety.

1.3. Архитектура

- **Domain Layer**: йерархия на `LibraryUnit` (Book, Periodical, Series) и `User` (Reader, Administrator).
- **Persistence Layer**: `Repository`–класове за I/O (LibraryUnitRepository, UserRepository) с бинарни файлове.
- **Command Pattern**: базов клас `Command` с наследници за всяка команда (`CommandAdd`, `CommandFind`, `CommandReturn` и др.).
- **Factory Pattern**: `CommandFactory` избира и създава обекта `Command` според токените.
- **Singleton**: `System` е глобален двигател, единствен екземпляр, съдържа `run()` метод и състояние.

2. Описание на създадените класове

2.1. Domain класове

2.1.1. LibraryUnit (абстрактен)

- **Член-данни**: `std::string title, publisher, genre, description; unsigned short releaseYear, rating; unsigned uniqueNumber; unsigned short copies, taken;`

- **Виртуални методи**:
  - `print()` – показва основна информация.
  - `writeToBinary(std::ostream&)`, `readFromBinary(std::istream&)` – сериализация.
  - `clone()` – клониране за хетерогенно копиране.
- **Наследници**:
  - `Book` (допълнителни: `author`, `Optional<std::string> ISBN`, `StringArray keyWords`)
  - `Periodical` (допълнителни: `unsigned short month`, `Optional<std::string> ISSN`, `std::vector<Article> articles`)
  - `Series` (множствено наследяване от `Book` и `Periodical`)

2.1.2. User (абстрактен)

- **Член-данни**: `std::string name, password; Date registerDate, lastLogin;`
- **Виртуални методи**: `print()`, I/O методи (binary/text), `clone()`, `getType()`, `getPrintLines()`.
- **Наследници**:
  - `Reader`: допълнително `std::vector<LibraryUnitTaken> takenUnits`.
    - `LibraryUnitTaken`: копие на зает `LibraryUnit*`, `Date borrowDate, returnDate`.
  - `Administrator`: допълнително `std::string email`.

2.2. Repositories

LibraryUnitRepository

- **Член-данни**: `std::fstream booksStream, periodicalsStream, seriesStream; std::vector<LibraryUnit*> units; std::string booksFile, periodicalsFile, seriesFile;`
- **Методи**:
  - `readAllBooks()`, `readAllPeriodicals()`, `readAllSeries()`, `readToEndAllFiles()`
  - `readUntilFindByUniqueNumber(unsigned)`
  - `writeAllToFiles()`, `openStream()`, `closeAllStreams()`

UserRepository

- **Член-данни**: `std::fstream readersStream, administratorsStream; std::vector<User*> users; size_t adminsCount; std::string readersFile, administratorsFile;`
- **Методи**:
  - `readAllReaders()`, `readToEndAllFiles()`, `readUntilFindUser(const std::string&)`
  - `writeAllToFiles()`, `openStream()`, `closeAllStreams()`

2.3. Command Framework

Command (абстрактен)

- **Метод**: `virtual void execute(System& sys, const std::vector<std::string>& tokens) const = 0;`

CommandFactory (Singleton)

- **Метод**: `Command* create(System&, const std::vector<std::string>& tokens) const;`
    - Разпознава и връща конкретен `Command*`, или `nullptr`.

Примери за Command класове

- `CommandLogin`, `CommandLogout`
- `CommandAll`, `CommandFind`, `CommandInfo`
- `CommandAdd`, `CommandChangeUnit`, `CommandRemoveUnit`
- `CommandTake`, `CommandReturn`
- `CommandAddUser`, `CommandRemoveUser`, `CommandChangeUser`

2.4. Помощни класове

Optional

- Собствена имплементация на std::optional, обект, който може да има/няма стойност. Специализация за `std::string` за binary I/O.
- Functions (предназначен за работа със стрингове)
- Методи от namespace FunctionsForBinary:
    - `writeString()`, `readString()`, `writeStringArray()`, `readStringArray()`
        - записват дължина + raw байтове, гарантират exception safety.
- Други методи:
    - fromDigitToChar(unsigned short), generateNString(), tokenizeString(std::string);

Date

- Опакова `unsigned date` (DDMMYYYY). Поддържа `+=/-=(months)`, проверка валидност, операции за сравнение, binary I/O.

3. Идеи за бъдещи подобрения

1. **По-добър метод за непълно четене на данни**
    - Индексиране в памет, зареждане на част от данните при нужда.
2. **Разширени функции за анализ**
    - Статистически анализи
3. **Добавяне на:**
    - Нови команди
    - Нови видове юзъри
    - Нови видове библиотечни единици

4. Заключение

Системата работи въз основа на изискванията: ефективно съхранение и достъп до различни типове библиотечни единици, защита на данните чрез валидация и обработка на изключения, както и разграничаване на правата на администратори и читатели. Тя коректно чете и записва от/във файлове и изпълнява коректно командите си. Работи на принцип strong exception safety.