A black and white logo

AI-generated content may be incorrect.

**Библитека**

**Антонио Бояджиев, 8MI0400192**

1. ***Уводна част – идея на проекта***

**Целта е проектът да реализира библиотечна система, която работи с различни единици като – книги, периодични издания, серии.**

**Системата поддържа два типа потребители – читатели и админи, като двата вида потребители се различават по своите права в системата и методите, до които имат достъп**

**Текущата реализация поддържа следните команди:**

**Създаване на потребител, създаване на администратор, влизане в системата, излизане от системата, създаване на единица – книга, периодично издание или серия, премахване на потребител, премахване на единица от системата, търсене по ключова дума, принтиране на наличните единици в момента, детайлно принтиране на информация за единица, промяна съдържанието на единица или потребител**

**Текущата реализация НЕ поддържа фукнционалностите за заемане и връщане на книга**

1. ***Допълнителни фукнционалности***

**Текущата реализация използва наготово фукнционалностите от стандартната библиотека, както следва:**

**Std::vector**

**Std::string**

**Std::exception**

**Std::except**

**Std::sstream**

**В допълнение – имаме класове Date, Article,IDGenerator – тези класове се грижат за правилното създаване на дати в системата, след като работим с тях, реализацията за класа Дата е леко модифицирана версия от една задача, която беше възложена през семестъра.**

**Kласът Article – в реда на реализация се установи, че периодичните издания съдържат списък от такива статии, затова тази структура беше създадена, за да улесни по-нататъшната работа с тях, като капсулира логиката, свързана с една статия в целия клас.**

**IDGenerator – представлява клас, който се грижи за задавенато на уникални идентификационни номера, той го подсигурява чрез писане и четене на последно извечен идентификатор в своя собствена база данни – двоичен файл ‘’last\_id.dat’’ . Вътре има само статични методи, тъй като не се нуждаем от инстанция, той просто отделя логиката за задаването от основните класове, но е важен.**

1. ***Основни класове и допълнителна информация***

**Реализацията се случва с два базови абстракти класa**

**– libraryUnit и libraryPerson, те задават поведенческото поведение за единиците и потребителите. Те изнасят общата информация за всички единици и потребители в себе си, като след това тяхната логика бива преизползвана от наследниците на тези класове.**

**Преки наследници на тези класове:**

* 1. ***Book***

**Класът книга – се използва за представяне на книга в нашата система, тя съдържа нужната информация от базовата част – единица, както и собствени уникални данни за една книга**

* 1. ***Periodicals***

**Периодичните издания са реализирани чрез този клас, той отново наследява единицата, като отново има уникални собствени данни**

* 1. ***Series***

**Пряк наследник на книгата и периодичното издание, в този момент от реализацията серията НЕ притежава собствени данни, различни от наследените от родителите си – книга и периодично издание.**

**Въпреки това чрез virtual public – наследяване от книгата и периодичното издание решаваме диамантения проблем в системата, като нямаме ambigious – error, както сме говорили серията получава само едно копие на базовата част, а не две, както може би се очаква.**

* 1. ***Reader***

**Пряк наследник на потребител, това е клас, който спомага за представянето на читателите в системата, читателите са специален вид потребители, които имат различни права, както и ограниченя – например, те ( в текущата реализация) единтсвени могат да заемат книги, както и да връщат такива, за жалост както беше споменато по-горе тази фукнционалност липсва в нашето представяне.**

* 1. ***Administrator***

**Администраторът е потребител в системата, който обаче НЕ може да заема книги, както и да връща такива, той има служебен имейл адрес, като допълнение на базовата си част от потребителя. Важно е да се спомене, че адиминистраторите могат единствени да създават нови потребители, единствени могат да създават нови издания – както и да махат създадени потребители и издания от системата, единствено те могат и да променят вече налични издания, както и да търсят други потребители от системата.**

* 1. ***Factory***

**Фабриката в системата е реализирана чрез класа LibraryFactory, тя притежава четири статични метода за създаване на потребители и единици, две за файлове, и две интерактивно, тя капсулира логиката по създаването или зарерждането на потребителите и единиците, като вътрешно в себе си използва специфични констуктури, до които само тя има достъп от класовете, които споменахме отгоре.**

* 1. ***System***

**Това е основната част, където всичко “оживява”, системата притежава четири вектора, както и имената на два файла – имената на файловете представляват именно имената на файловете, от които системата се зарежда, както и пише в тях нанесените промени. Важно ограничение на задачата е, че се изисква стрикно спазване и избягване на зареждането на цялата информация от файла директно в паметта на системата, това се случва единствено, когато е необходимо. Текущата реализация се справя с този проблем ефикасно, като изцяло работи синхорнно с файловете, тук се прави trade-off между памет и време, макар операциите да не са толкова тежки откъм памет, те са бавни и тромави, тъй като претърсват двата файла, с които системата разполага, след което зарежда търсеният потребител или единица.**

1. ***File organization***

**Струва ми се много полезно да кажем експлицитно каква е структурата на файловете, с които нашата система работи**

***Units.dat***

**Представлява двоичен файл, в който записваме единиците, както и след това четем от него същите тези единици. Този файл е структуриран, както следва:**

**В началото му винаги е написано общият брой на всички единици, които можем да видим вътре, следва състояние на текущата единица, това състояние се променя – или валидно или невалидно, веднъж невалидно – то остава невалидно, но все пак за да избегнем по-сложни операции с файлвоете – оставаме данните за него въте и следим това състояние, в бъдеще може да се помисли този проблем да бъде отстранен, след състоиянието се записва и типът на дадената единица – той е или книга, или периодично издание или серия, следват данните специфични за всяка от единиците в зависимот от това какъв е обекта, който се записва или съответно чете, съответно писането и четеното се извършват с виртуални фукнции, които вътре в себе си използват невиртуални фукнции от самия клас и неговият родител- serialize -> serializeBase, serializeBookPart и etc.**

**Избран е този подход, тъй като за мен е най – чист и прост.**

***Users.dat***

**Е отново двоичен файл, съдържащ в себе си информацията за потребителите , като отново първо се пише размерът на всички потребители, следва състояние на потребител, както и вече данните на потребителите, важно уточнине за потрбителите – състоянието при тях отново е валидно или невалидно, като невалидно състояние може да се достигни, когато се извършва промяна – тогава се прави чисто нов запис, като старият се маркира като невалиден, отново същото ограничение както при единиците се наблюдава и тук, като това отново може да е бъдещо подобрение спрямо текущата реализация на системата.**

1. **Tестове**

**Програмата е заредена и пусната с готов тест -> на място може да се разиграе отново, но тестът е направен както следва:**

**Създаване на единици – книга, периодично издание, серия, изход и записване на данните, създаване на потребители – антонио и нов админ – newAdminUser, следва изход и запазване на данните, следва тест на методите за извеждане на информация – books all, series all, periodicals all, all, info. Промяна на периодичното издание - > new\_x\_spider и останалите му данни, remove на книгата Grozno Patence, remove на newAdminUser, change на user antonio – нова парола се слага.**

**Накрая find – unit**

**Find – user**

**Няма return, take – не е довършено**