Dôveryhodný systém - LibBook

Informačný systém pre knižnicu

Tento softvér vytvoril Matej Delinčák. Softvér simuluje informačný systém virtuálnej knižnice. Dôveryhodný softvér je skrytý u mňa v login systéme. Ak sa užívateľ chce zaregistrovať, musí splniť určité požiadavky. A zároveň tento nový učet musí pracovník potvrdiť v rámci aplikácie, inak je nefunkčný. Teda do systému sa nedá inak dostať, len cez interakciu z vnútra prostredia. Ak chcete program spustiť musia byť priložené súbory s dátami. Ak ich nemáte, v triede *LibraryEvidenceSystem* je zapoznamkovaný kód, ktorý vygeneruje vzorku.

Pre otestovanie programu mam štyroch uživateľov:

- Dieťa - zákaznik: login: xmatej heslo: Matej2000
- Dospelý - zákaznik: login: xpeter heslo: Peter2001
- Pracovník: login: xaja heslo: Andrea1999
- Knihovník: login: xroman heslo: Roman1885

Softvér dokáže: **Užívateľ (dva typy - Dieťa a Dospelý):**

- si môže rezervovať knihy,
- vypísať správy o tom ako dopadlo rezervovanie a následne,
- zobraziť knihy jemu dostupné,
- zobraziť knihy, ktoré vlastní
- zobraziť recenziu (ak existuje) knihy,
- vrátiť požičanú knihu

Pracovník:

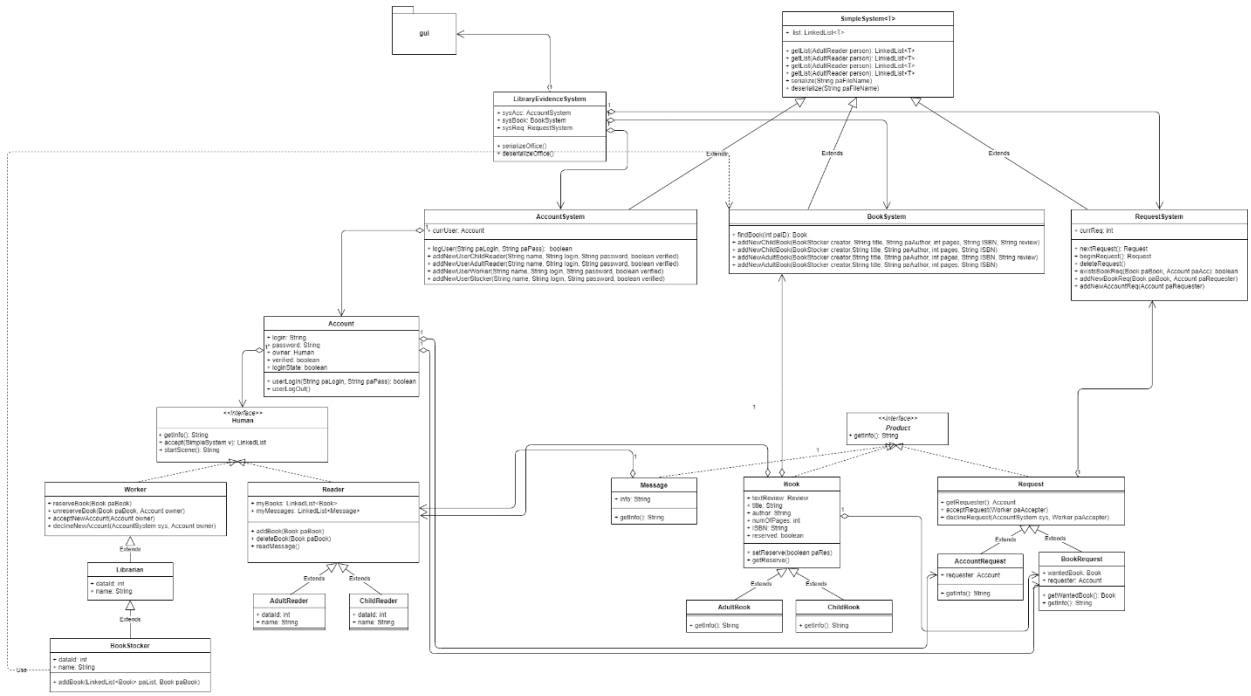
- zobraziť knihy,
- zobraziť všetky účty,
- potvrdiť rezervovanie knihy užívateľom (zobraziť účet žiadateľa),
- alebo požiadavku zamietnuť

Knihovník:

- dokáže všetko čo obyčajný pracovník
- vie vytvárať nové knihy

Do prostredia si môžete vytvoriť aj vlastný účet, ale s určitými podmienkami. Heslo musí mať aspoň 8 znakov, jedno číslo a jedno veľké písmeno.

Hlavný diagram:



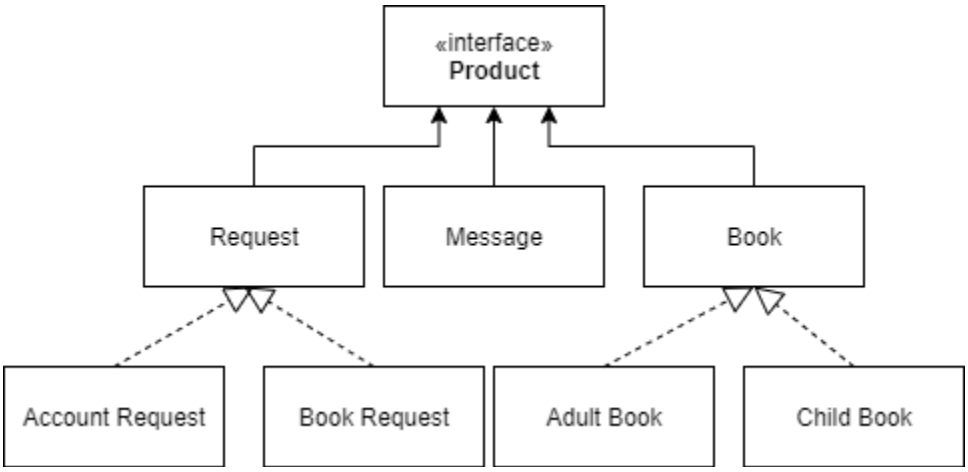
- tento sa da zobraziť aj v priečinku docs

V tejto aplikácií som využil:

Dve oddelené hierarchie tried

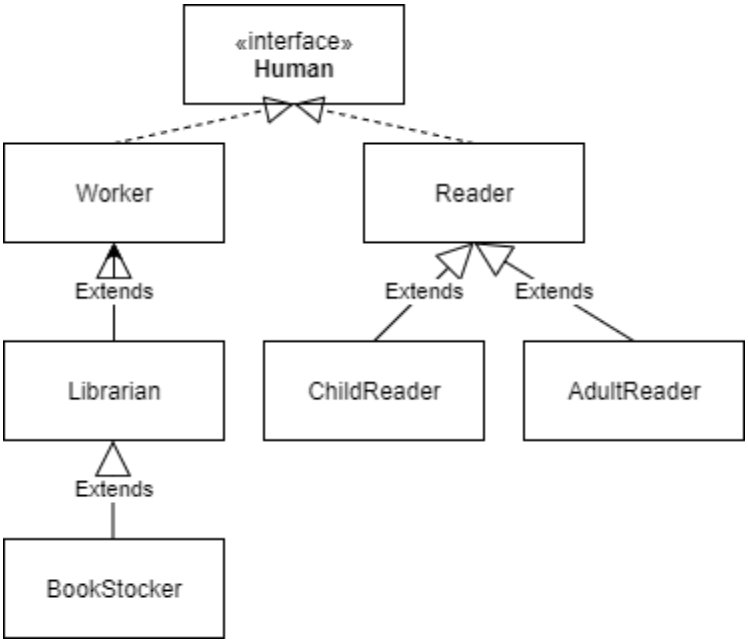
Hierarchia ľudí

V tejto hierachií je uplatnené dedenie, polymorfizmus, je použité rozhranie, agregácia ako aj korektné zapuzdrenie. Dedenie môžme vidieť medzi pracovníkom a knihovníkom. Knihovník dedí privilégia pracovníka.
Polymorfizmus môžme vidieť vo funkcií *startScene*, ktorá je prekrývaná jednotlivými triedami.
Rozhranie *Human* spája dve vetvy ľuďí.
Agregácia je schovaná v triede *Reader*, kde táto trieda vlastní knihy a správy.
Všetky atribúty su buď private alebo protected, zároveň su k ním vytvorené public gettre a settre.



Hierarchia produktov

V tejto hierachií je uplatnené dedenie, polymorfizmus, je použité rozhranie, agregácia ako aj korektné zapuzdrenie.
Dedenie môžme vidieť medzi Book a konkretnejšou triedou ChildBook.
Polymorfizmus môžme vidieť vo funkcií *getInfo*, ktorá je prekrývaná každou triedo v strome.
Rozhranie *Product* spája až tri vetvy tried.
Agregácia je schovaná v triede *AccountRequest*, kde táto trieda vlastní Account ako atribut.
Všetky atribúty su buď private alebo protected, zároveň su k ním vytvorené public gettre a settre.



Oddelenie aplikačnej logiky od GUI a využitie grafického rozhrania pre užívateľa

Samotné GUI som umiestnil do vlastného package-u pod menom *gui*. V tejto zložke sa nachádzajú Controllery pre rôzne scény aplikácie. Pri stlačení nejakého tlačidla, sa vykoná prislušná funkcia, ktorej logika je umiestnená len v triede LibraryEvidenceSystem. V objektoch Controller sa nachádzajú aj spracovatele udalostí.

Použitie návrhového vzoru Visitor

Využil som ho na upresnenie právomocí uživateľov. Napríklad čitateľ nemá prístup k databáze účtov, len ku knihám. A keď ide o dieťa, to má sprítupnené len určité knihy. Na obrázku je znázornený prístup pracovníka do databázy účtov.

```
public class AdultReader implements Reader, Serializable {
    //visitor accepter
    @Override
    public LinkedList accept(SimpleSystem v) { return v.getList( person: this); }
```

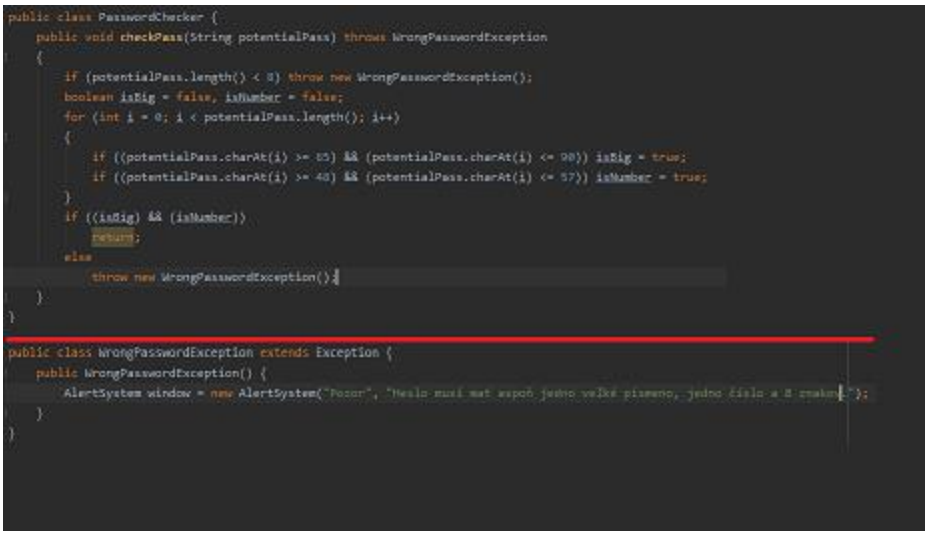
```
public class AccountSystem extends SimpleSystem implements Serializable {
    @Override
    public LinkedList getList(Librarian person) { return (LinkedList<Account>)list; }
```

Použitie návrhového vzoru Model-view-controller

Tento návrhový vzor som použil pri implementácií používateľského interface-u. Kde Model je moja aplikačná logika. View sú súbory zodpovedné za vizualizáciu (.fxml) a controller, sú všetky objekty s menom controller, ktoré updatujú aplikačnú logiku.

Vytvorenie vlastnej výnimky

Vytvoril som vlastnú výminku, ktorá vlastní text o nesprávnom vytvorení hesla (nie sú splenené určité požiadavky). Táto podmienka je aj vyhadzovaná ako aj ošetrovaná.



Použitie RTTI

RTTI je využité napríklad aj v tom, že sa pracuje s knihou, ktorú si až užívateľ vyberie. Alebo keď sa vytvára nový účet, tak si vyberá ,aký typ účtu to bude. Metóda v kinhe *showInfo* vráti info o knihe, až na základe jej atribútu reserved (ten sa mení pri rezervovaní knihy).

Použitie vhniezdenej triedy

Vhniezdenú triedu som využil v classe Book. Definoval som v nej triedu Review, s ktorou bude pracovat jedine kniha.

```java
public abstract class Book implements Serializable {
    //atributes
    protected Review textReview;
    protected final String title;
    protected final int numOfPages;
    protected final int dataId;
    protected final String ISBN;
    protected boolean reserved;

    //enclosed class
    protected class Review implements Serializable {
        private String text;

        public Review (String s) {this.text = s;}

        public String getText() { return text; }
    }
```

Serializáciu a deserializáciu

Serializujem tri systémy. A to systém účtov, kníh a žiadostí. Pri deserializácií tieto prepojenie naspäť vytvorím a program bude bežať ďalej ako serializáciou a vypnutím.

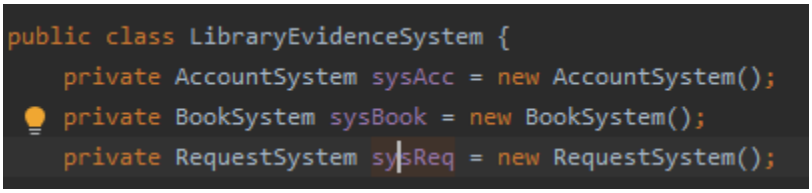
Použitie viacniťovosti - multithreading

Viacniťovosť som použil v triede LibraryEvidenceSystem vo funkcií serializeOffice. V tomto systéme mám tri rozlišné systémy pre účty, knihy a požiadavky. Tieto serializujem paralelne, pretože každý sa ukladá do vlastného súboru.

```java
//serializuje celu kniznicu
public void serializeOffice() throws InterruptedException {
    Thread t1 = new Thread (sysAcc,  name: "Account system");
    t1.start();
    Thread t2 = new Thread (sysBook,  name: "Account system");
    t2.start();
    Thread t3 = new Thread (sysReq,  name: "Account system");
    t3.start();
    t1.join();
    t2.join();
    t3.join();
}
```

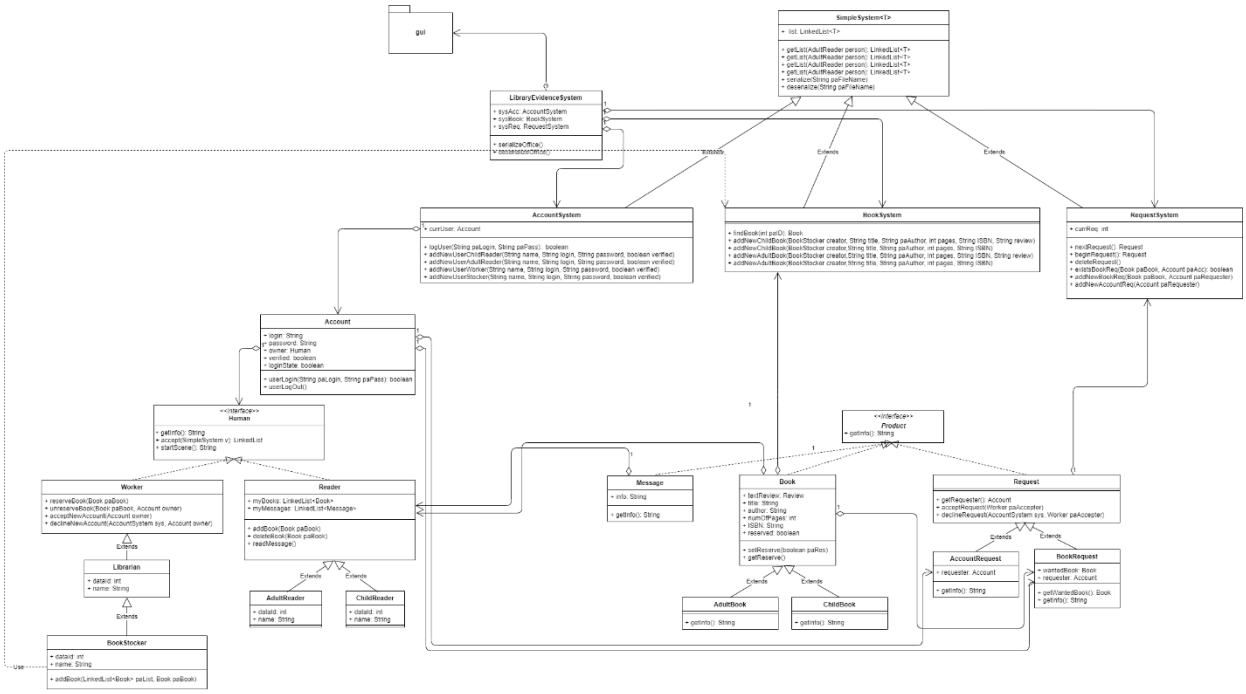
Využitie vlastnej generickej triedy

Generická trieda SimpleSystem je rodič troch systémov. Tento objekt pracuje so všeobecným objektom. Z neho su odvodené tri základné spracovské systémy mojej aplikácie.



Zoznam dôležitých comittov

Základné triedy:
https://github.com/OOP-FIIT/oop-2020-str-12-pu1-povazanova-mateju25/commit/f0860f37af10336dac4ae9e96b6c79b8416cb791
Viacero scén:
https://github.com/OOP-FIIT/oop-2020-str-12-pu1-povazanova-mateju25/commit/ecf1dba55dcaa7bc489667dabb5af3bc570aab99
Požiadavky:
https://github.com/OOP-FIIT/oop-2020-str-12-pu1-povazanova-mateju25/commit/67ef2d4e1431536d7a721ca6adb0eb27e61afa47
Login systém:
https://github.com/OOP-FIIT/oop-2020-str-12-pu1-povazanova-mateju25/commit/0822ce88ec4cddc302c09c7bbd2827eed79f53e9
Pridaný visitor:
https://github.com/OOP-FIIT/oop-2020-str-12-pu1-povazanova-mateju25/commit/3b83f26fb35daae0e55df710b9094812ef7786b5
Deserializácia:
https://github.com/OOP-FIIT/oop-2020-str-12-pu1-povazanova-mateju25/commit/0df0f30a3f535247615a34a008b21850a4ed7d52
Design pre GUI:
https://github.com/OOP-FIIT/oop-2020-str-12-pu1-povazanova-mateju25/commit/ab6bb12362aa7a1bc063e2601005724da32f6b46


gui
SimpleSystem<T>
+ list: LinkedList<T>
+ getList(AdultReader person): LinkedList<T>
+ getList(AdultReader person): LinkedList<T>
+ getList(AdultReader person): LinkedList<T>
+ getList(AdultReader person): LinkedList<T>
+ serialize(String pathName)
+ deserialize(String pathName)
LibraryEvidenceSystem
+ sysAcc: AccountSystem
+ sysBook: BookSystem
+ sysReq: RequestSystem
+ serializeOffice()
+ deserializeOffice()
Extends
Extends
Extends
AccountSystem
+ curUser: Account
+ logUser(String paLogin, String paPass): boolean
+ addNewUserChildReader(String name, String login, String password, boolean verified)
+ addNewUserAdultReader(String name, String login, String password, boolean verified)
+ addNewUserWorker(String name, String login, String password, boolean verified)
+ addNewUserStocker(String name, String login, String password, boolean verified)
BookSystem
+ findBook(int parID): Book
+ addNewChildBook(BookStocker creator, String title, String paAuthor, int pages, String ISBN, String review)
+ addNewChildBook(BookStocker creator,String title, String paAuthor, int pages, String ISBN)
+ addNewAdultBook(BookStocker creator, String title, String paAuthor, int pages, String ISBN, String review)
+ addNewAdultBook(BookStocker creator,String title, String paAuthor, int pages, String ISBN)
RequestSystem
+ curnReq: int
+ reqRequest(): Request
+ begirRequest(): Request
+ deleteRequest()
+ existsBookReq(Book paBook, Account paAcc): boolean
+ addNewBookReq(Book paBook, Account paRequester)
+ addNewAccountReq(Account paRequester)
Account
+ login: String
+ password: String
+ owner: Human
+ verified: boolean
+ loginState: boolean
+ userLogin(String paLogin, String paPass): boolean
+ userLogOut()
<<interface>>
Human
+ getInfo(): String
+ accept(IVisitSystem v): LinkedList
+ startScene(): String
<<Interface>>
Product
+ getInfo(): String
Worker
+ reserveBook(Book paBook)
+ unreserveBook(Book paBook, Account owner)
+ acceptNewAccount(Account owner)
+ declineNewAccount(AccountSystem sys, Account owner)
Reader
+ myBooks: LinkedList<Book>
+ myMessages: LinkedList<Message>
+ addBook(Book paBook)
+ deleteBook(Book paBook)
+ readMessage()
Message
+ info: String
+ getInfo(): String
Book
+ textReview: Review
+ title: String
+ author: String
+ numOfPages: int
+ ISBN: String
+ reserved: boolean
+ setReserve(boolean paRes)
+ getReserve()
Request
+ getRequester(): Account
+ acceptRequest(Worker paAcceptor)
+ declineRequest(AccountSystem sys, Worker paAcceptor)
Librarian
+ dataId: int
+ name: String
AdultReader
+ dataId: int
+ name: String
ChildReader
+ dataId: int
+ name: String
AdultBook
+ getInfo(): String
ChildBook
+ getInfo(): String
AccountRequest
+ requester: Account
+ getInfo(): String
BookRequest
+ wantedBook: Book
+ requester: Account
+ getWantedBook(): Book
+ getInfo(): String
BookStocker
+ dataId: int
+ name: String
+ addBook(LinkedList<Book> paList, Book paBook)
Extends
Extends
Extends
Extends
Extends
Extends
Extends
Extends
Extends
Extends
Use
1
1
1
1
1
1
1
1
1
1