SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií  
Ilkovičova 2, 842 16 Bratislava 4

Lagatoria

Objektovo-orientované programovanie

Róbert Junas

FIIT STU

Cvičenie: streda 14:00

Cvičiaci: Ing. Anna Považanová

24.4.2021

Id: 102970

# Obsah

[Obsah 2](#_Toc70194649)

[1. Rámcové zadanie 3](#_Toc70194650)

[2. Štruktúra 4](#_Toc70194651)

[3. Kritéria 5](#_Toc70194652)

[3.1. Hlavné kritéria 5](#_Toc70194653)

[3.1.1 Dedenie 5](#_Toc70194654)

[3.1.2 Polymorfizmus 5](#_Toc70194655)

[3.1.3 Rozhrania 5](#_Toc70194656)

[3.1.4 Zapuzdrenie 5](#_Toc70194657)

[3.1.5 Agregácia 5](#_Toc70194658)

[3.1.6 Oddelenie aplikačnej logiky od používateľského rozhrania 5](#_Toc70194659)

[3.1.7 Organizácia do balíkov 5](#_Toc70194660)

[3.2. Ďalšie kritéria 5](#_Toc70194661)

[3.2.1 Návrhové vzory 5](#_Toc70194662)

[3.2.2 Výnimky 10](#_Toc70194663)

[3.2.3 GUI 12](#_Toc70194664)

[3.2.4 Multithreading 13](#_Toc70194665)

[3.2.5 RTTI 13](#_Toc70194666)

[3.2.6 Vhniezdené triedy 14](#_Toc70194667)

[3.2.7 Lambda výrazy 14](#_Toc70194668)

[3.2.8 Implicitná implementácia v rozhraní 15](#_Toc70194669)

[3.2.9 Použitie serializácie 16](#_Toc70194670)

[4. Verzie 17](#_Toc70194671)

# Rámcové zadanie

Knihy sú dôležitou súčasťou každého z nás. Preto ich plánovanie a vydávanie je veľmi dôležité. Môj projekt sa zaoberá vydávaním kníh do stánkov a vydavateľovej predajne. Teda výsledkom plánovania bude kniha predávajúca sa v kníhkupectve.

Autor musí najprv knihu napísať – dostane nápad, napíše ju, vymyslí názov a určí žáner knihy. Potom dá svoju knihu vydavateľstvu.

Vydavateľstvo sa skladá z ľudí – manažéra, ktorý je v kontakte s autorom, dizajnér, ktorý navrhne obálku a vyberie väzbu knihy; a korektor opravujúci chyby v knihe. Manažér navrhne cenu knihy a nakoniec distribútor povie koľko kníh sa má vytlačiť a rozdelí medzi odoberateľov. Po tomto procese sa knihy dajú na tlač, kde sa vytlačia strany, pripevnia sa k väzbe a pošlú sa odoberateľom.

Kníhkupectvo sa skladá z predajne a skladu. Predajňa môže byť organizovaná do viacerých kategórií, ako je žáner/druh. V sklade sa ale ukladajú knihy podľa toho, na ktoré miesto, prípadne viac priľahlých miest, sa zmestia. V kníhkupectve pracuje predajca a skladník. Predajca obsluhuje zákazníka a dopĺňa knihy do predajne. Skladník pridáva knihy do skladu a premiestňuje ich do iných sektorov skladu, aby mohol urobiť miesto pre nové knihy.

Zákazník môže nájsť knihu v predajni, vidieť informácie o knihe.

# Štruktúra



# Kritéria

## Hlavné kritéria

### Dedenie

### Polymorfizmus

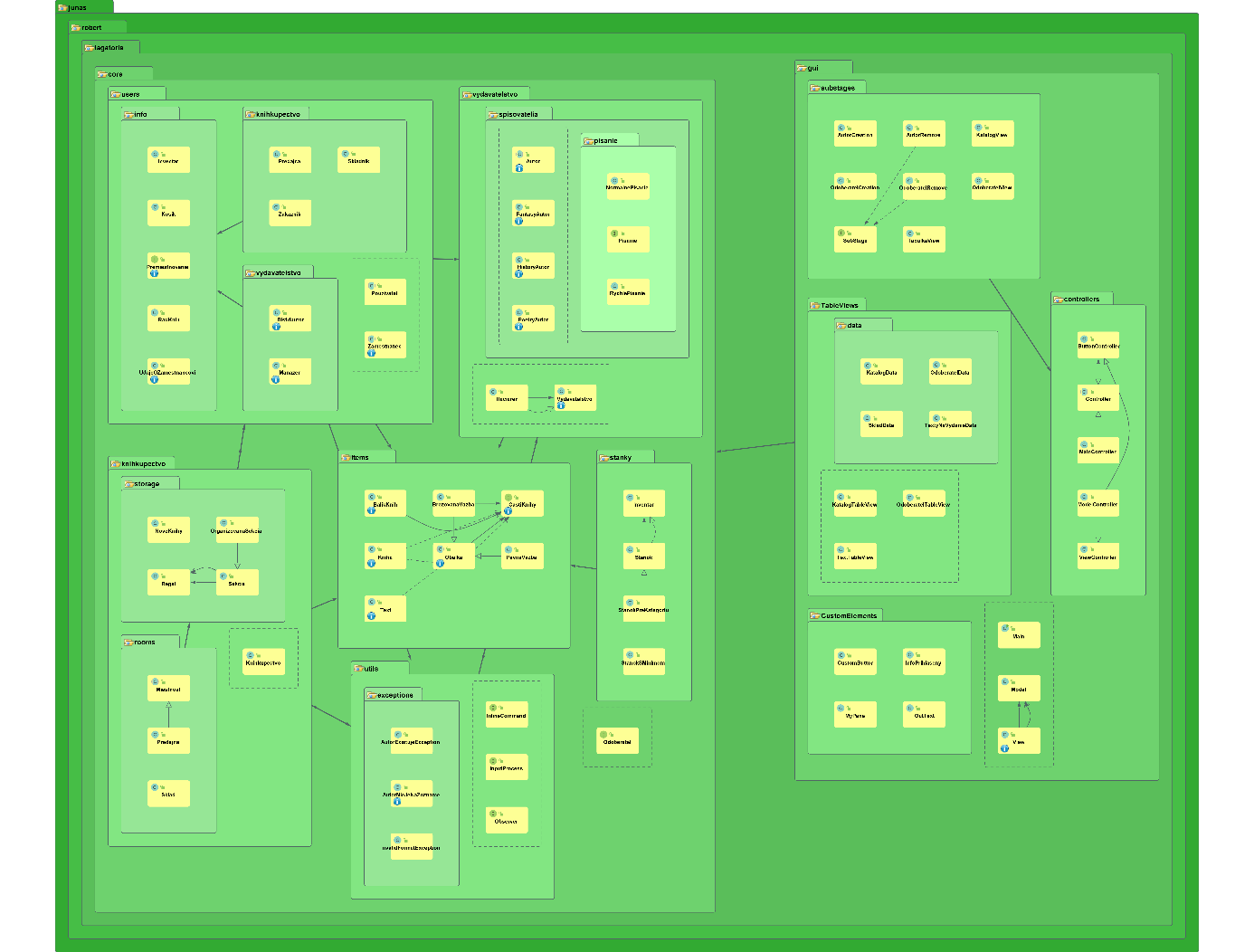
### Rozhrania

### Zapuzdrenie

### Agregácia

### Oddelenie aplikačnej logiky od používateľského rozhrania

### Organizácia do balíkov

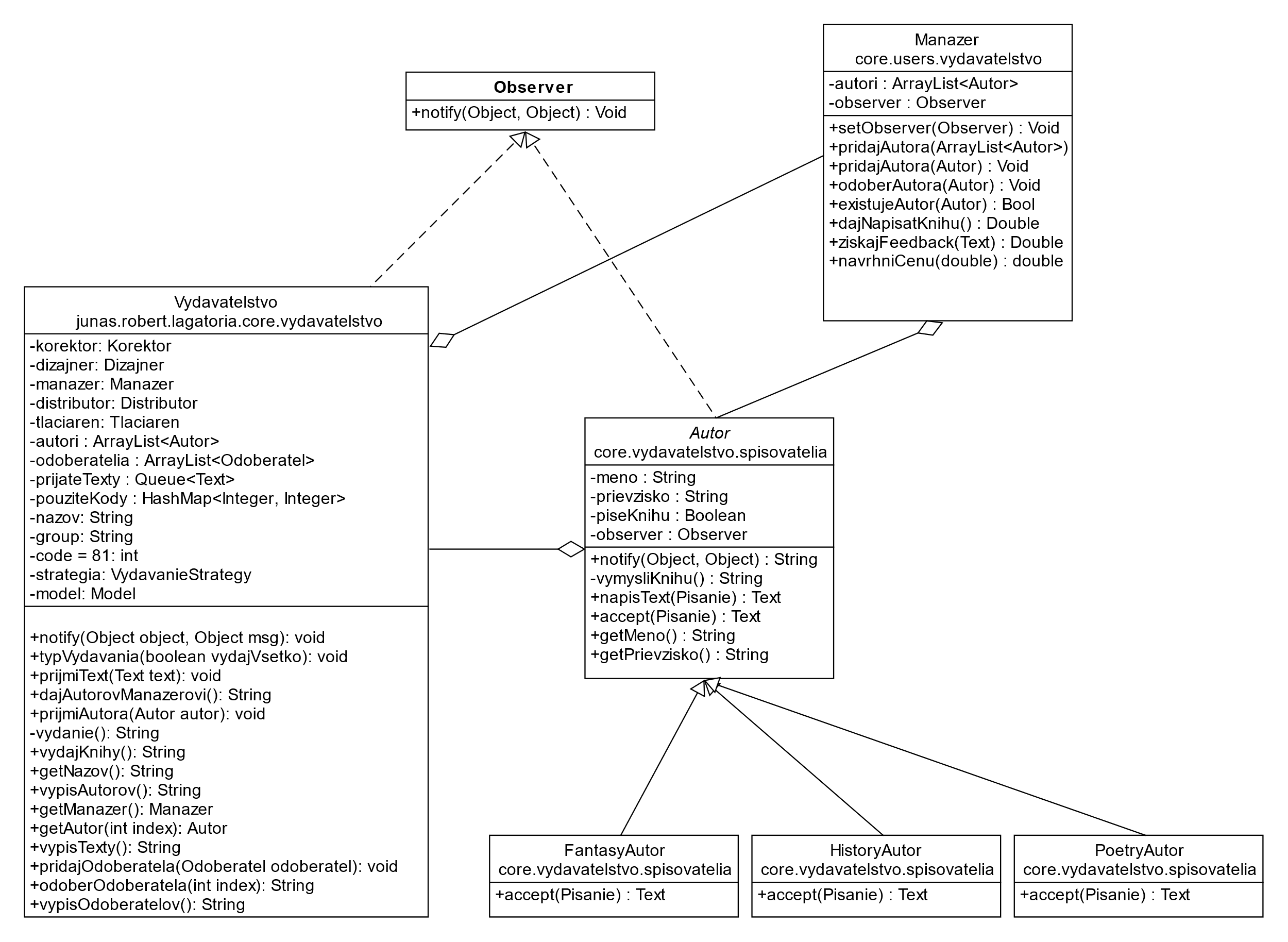


## Ďalšie kritéria

### Návrhové vzory

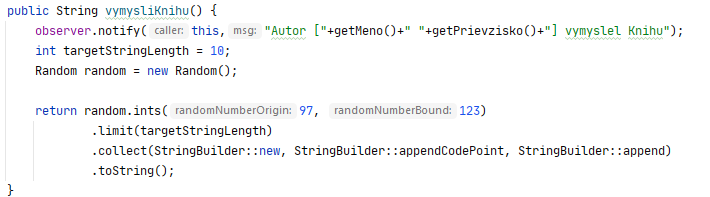
**Observer**

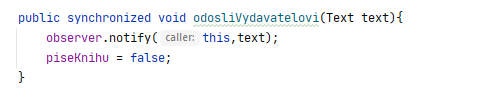
Model Observer sme využili naprieč celým projektom, či už na komunikáciu ovládačov s View alebo Modelom, ale aj medzi triedami Manažér a Autor, kde Manazer upovedomí svojich autorov, že chce aby napísali knihu.





Observer bol využitý medzi autormi a vydavateľstvom, kde autor upovedomuje vydavateľstvo o svojom postupe v písaní textu.

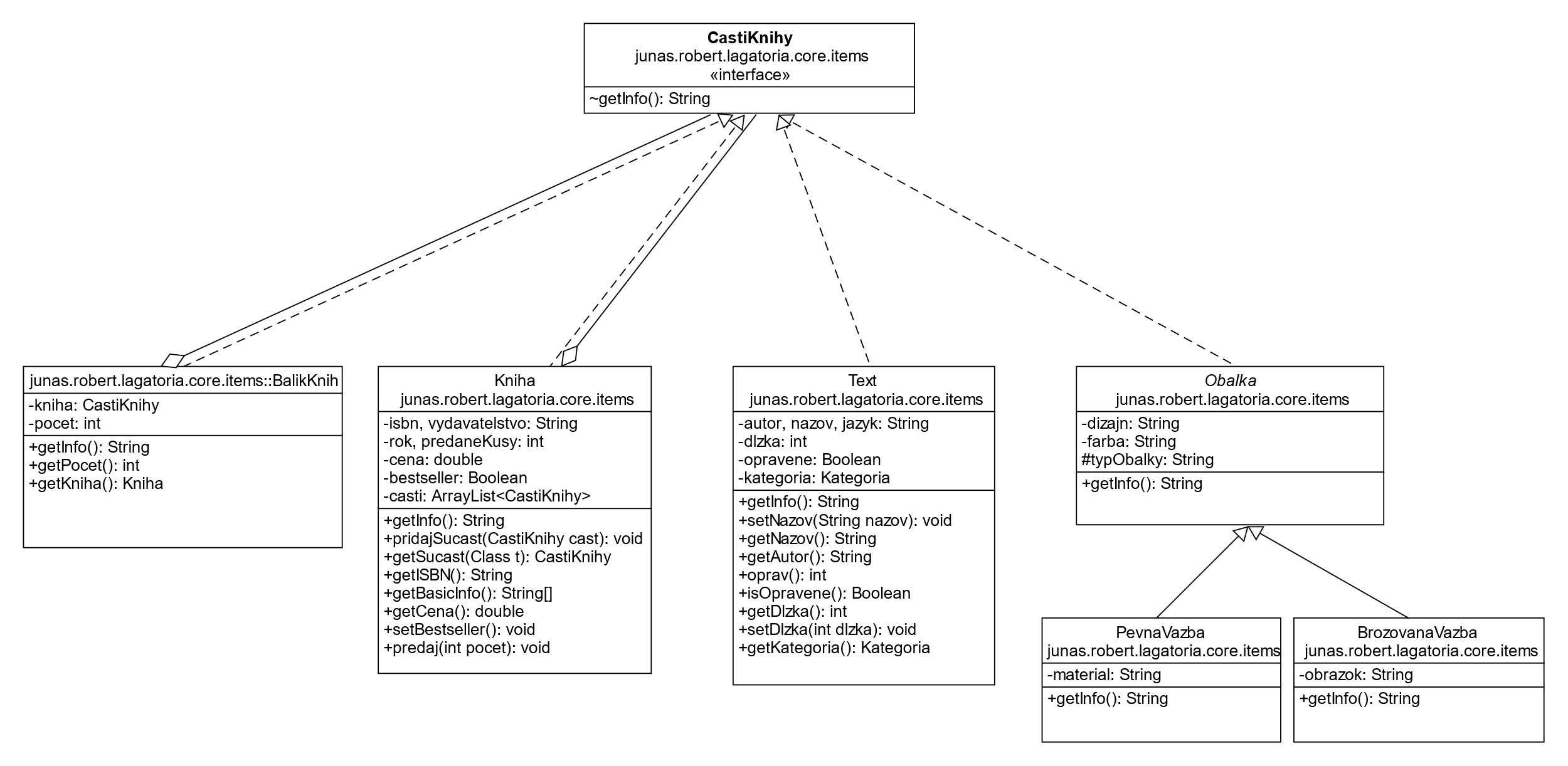


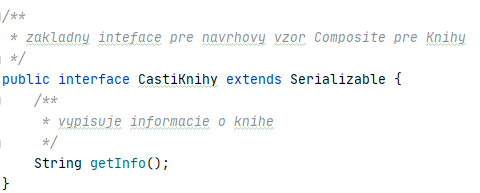


Máme definované aj rozhranie Observer, kde je definovaný protyp funkcie notify(Object, Object)

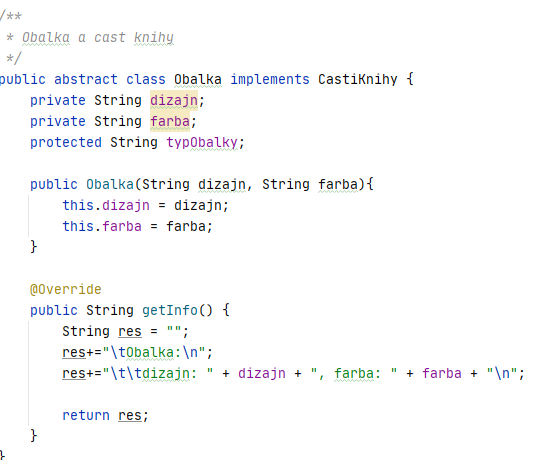
**Composite**

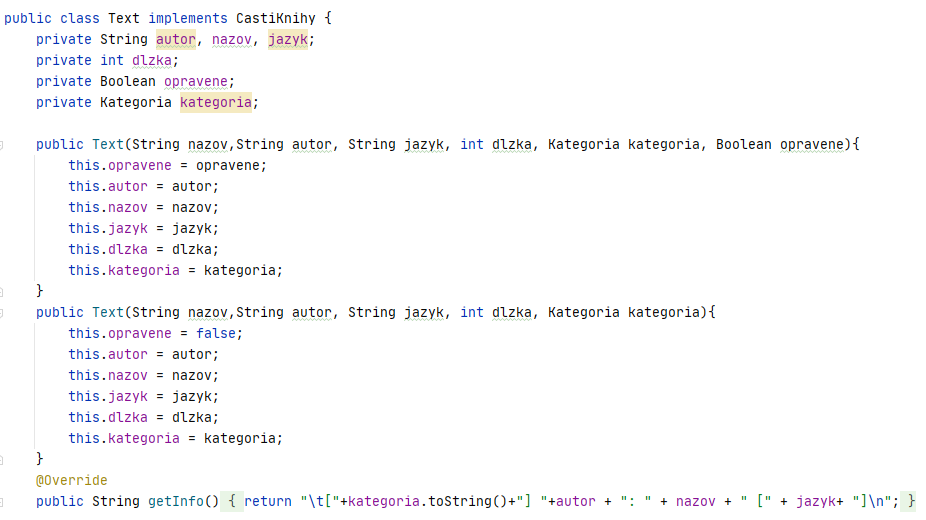
Využívaný spojení s knihami, kde kniha sa skladá z textov a obálky a nakoniec aj knihy sa balia do balíkov kde je ich väčší počet.





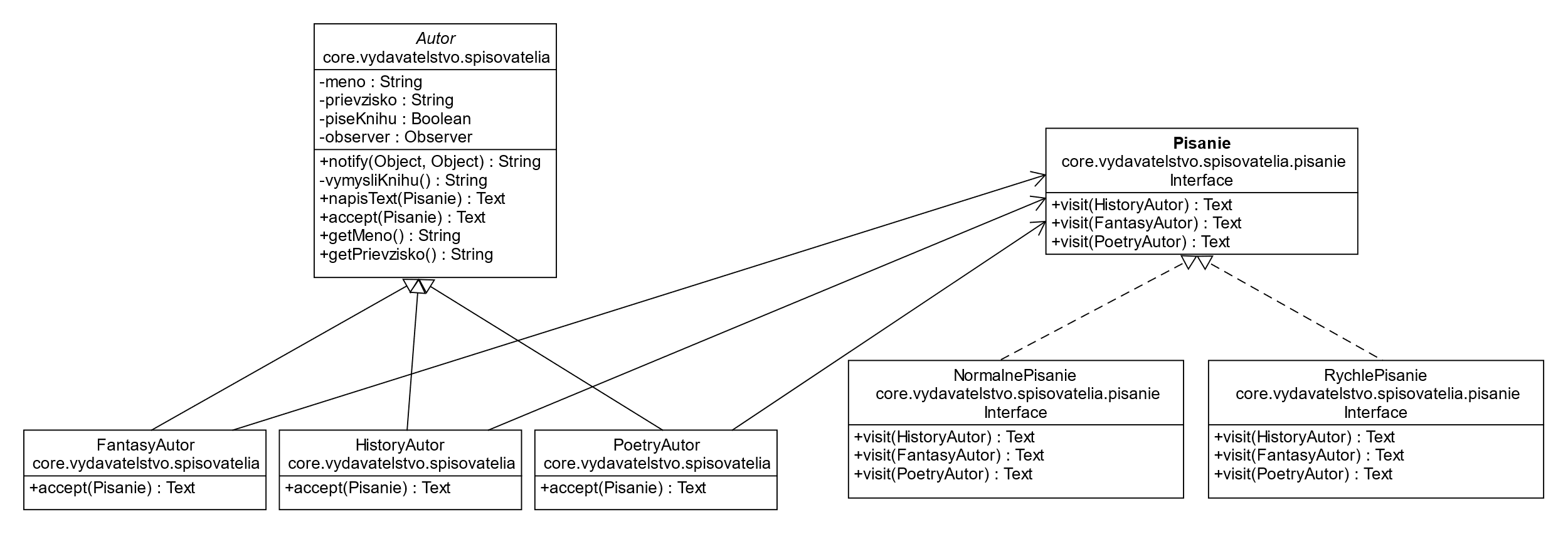


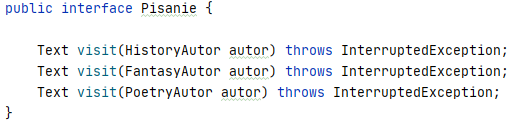




**Visitor**

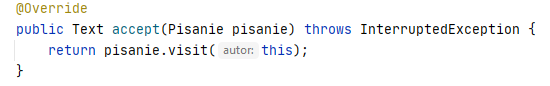
V projekte sa vyskytuje aj model Visitor a to spôsobom, že každý druh autora píše iným spôsobom, teda trvá mu to kratšie alebo dlhšie, vzhľadom koľko práce musí dať do samotného písania. Máme definované dva spôsoby písania: Normálne a rýchle písanie. Kde sa odráža počet strán od rýchlosti akou autor písal.



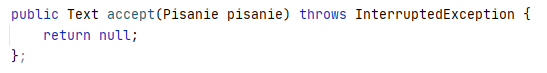








Použitie v triede Autor.java



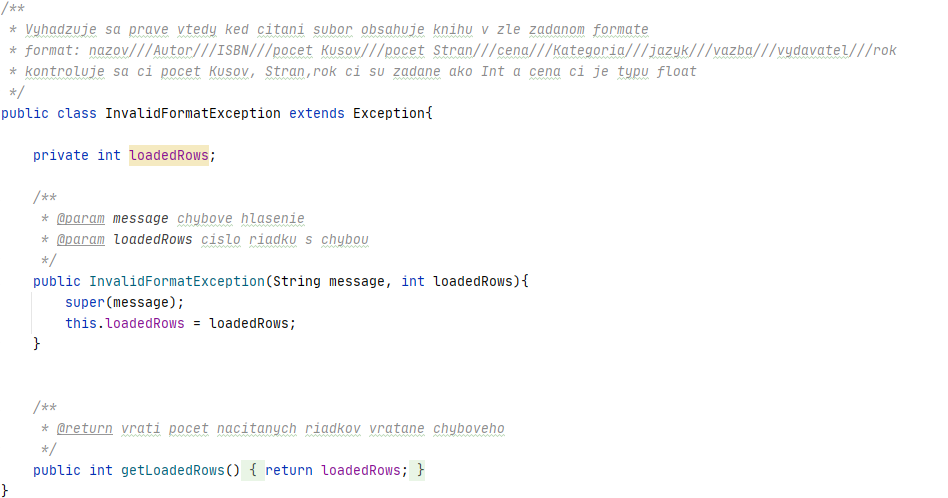


**Strategy**

??????

### Výnimky

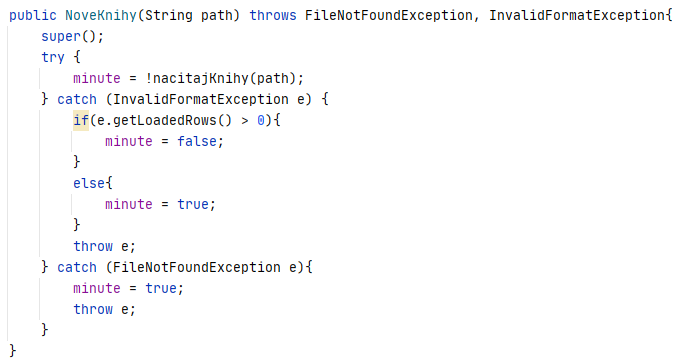
V aplikácií máme implementovaných niekoľko výnimiek: AutorExistujeException, AutorNieJeNaZozname, InvalidFormatException.

InvalidFormatException:

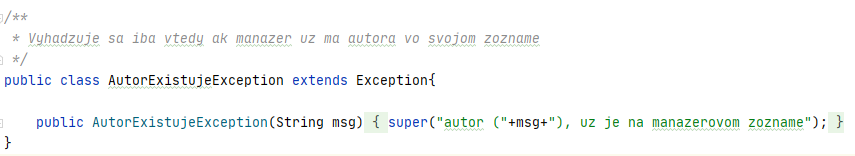
Funkcia vyhadzujúca výnimku:



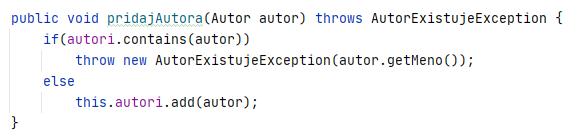
**Konštruktor chytá výnimku:**



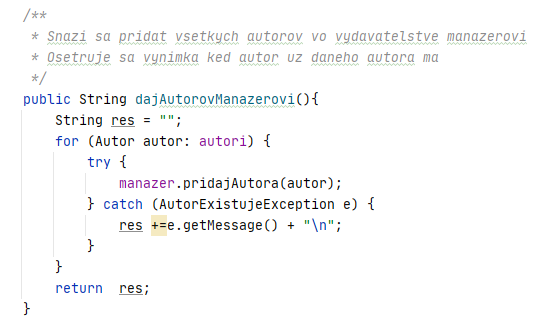
**AutorExistujeException:**



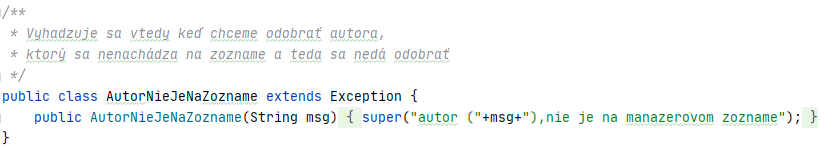
Funkcia vyhadzujúca výnimku:



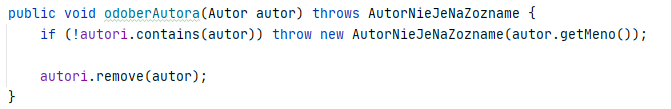
Metóda chytajúca výnimku:



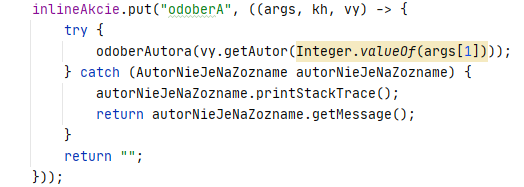
**AutorNieJeNaZozname:**

****

Funkcia vyhadzujúca výnimku:



Metóda chytajúca výnimku:

****

### GUI

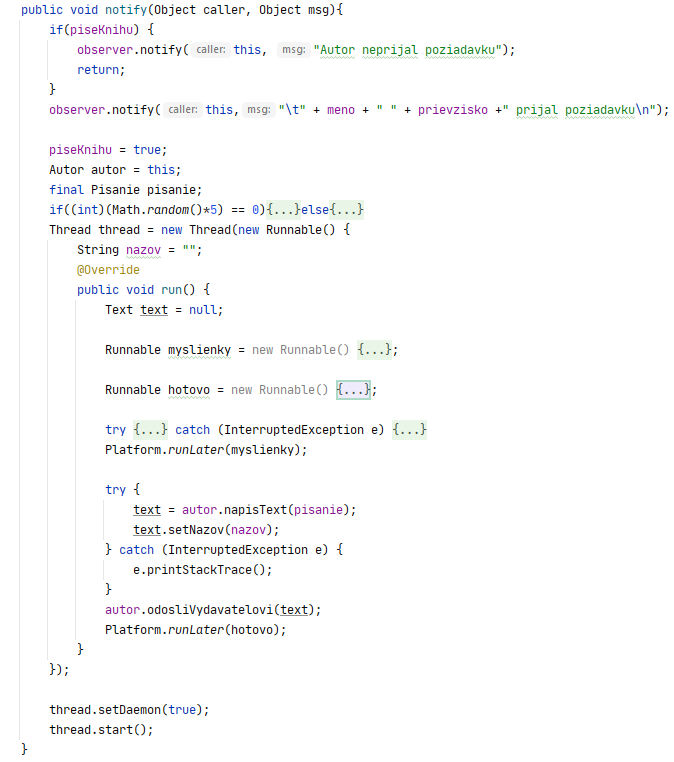
Aplikovali sme model MVC, kde hlavná scéna je vytvorená vo View.java a máme definované aj ďalšie okná ako sú okna na pridanie odobranie autorov/odoberateľov a okná s tabuľkovým zobrazením katalógu kníh, odoberateľov a textov pripravených na vydanie.

Tieto zobrazenia spracováva ViewController, ktorý je upovedomovaný MainController-om, ktorý spracováva vstupy z hlavného okna. Z vedľajších to robí samotný ViewController, ktorý upovedomí Controller, ktorý je nad ním. Prácu s modelom spúšťa ModelController s podkontrolerom ButtonController (spracováva vstupy vyvolané tlačidlami). Zavolajú funkcie v modeli a ak je potrebné tak oznámia ViewControlleru, že treba upraviť View.

Model samotný sa kumuluje v Model.java, kde sa vykonáva celý model a sám o sebe nijako nezasahuje do View, prípadne upovedomí Controller o zmene, ktorú treba vykonať.

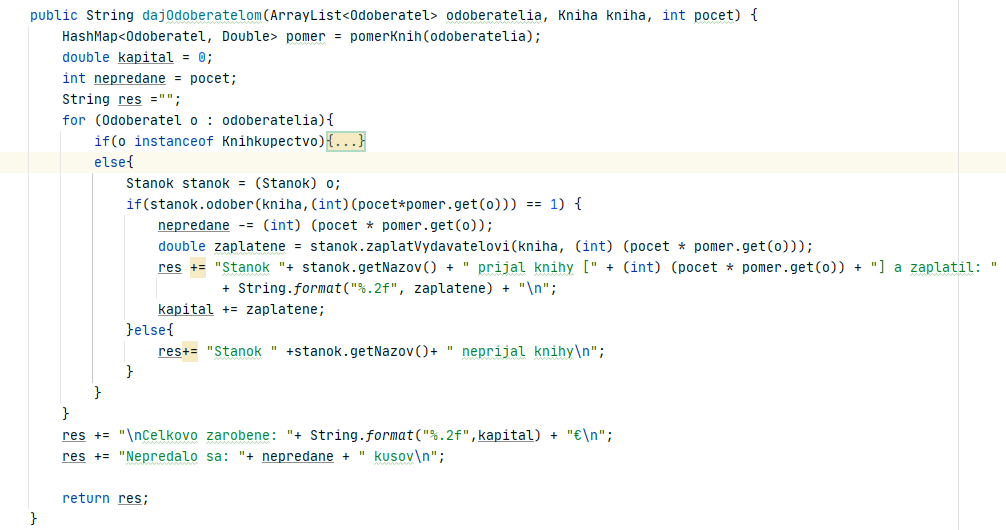
### Multithreading

Viacnitovosť bola využitá v Autoroch, ktorý sú schopný naraz písať texty, ktoré následne pošlú vydavateľovi.



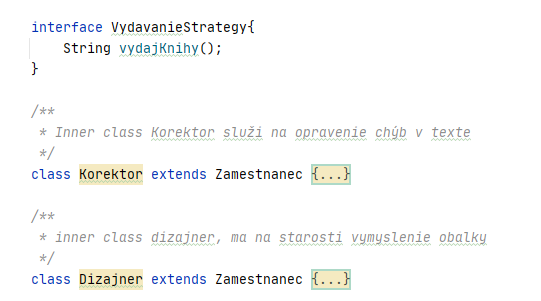
### RTTI

Použitie RTTI je v triede Distributor, vo funkciach *dajOdobertalom* a*pomerKnih*, kde sa výrazne líši správanie Stánkov a Kníhkupectva. Kníhkupectvo nie vždy musí hneď prijať knihy od Vydavatelstva a neprijaté knihy sa pridávajú na list kníh čakajúcich na prijatie. Na druhú stranu Stanky, prijímajú knihy neustále, bez zbytočného vyčkávania.



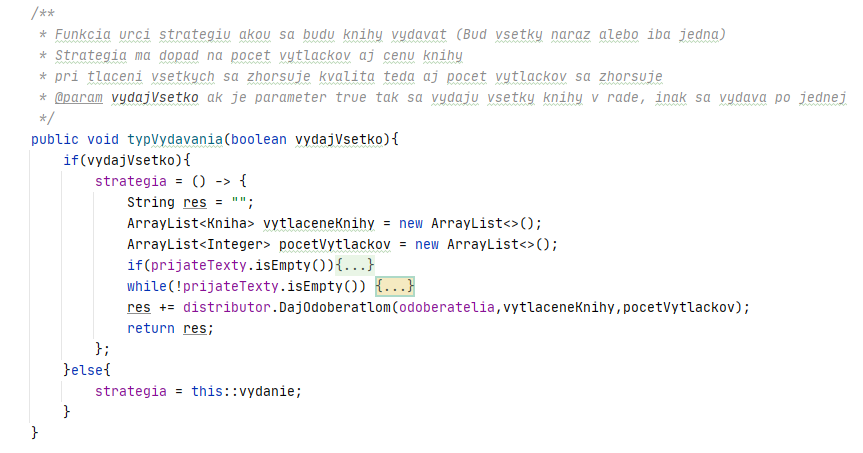
### Vhniezdené triedy

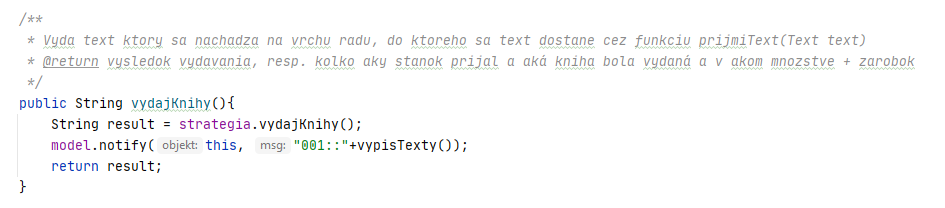
Vhniezdené triedy a rozhrania sa nachádzajú v triede Vydavatelstvo.java. Implementujeme tu stratégiu vydávania ako aj triedy Korektor a Dizajnér, ktoré by mali byť vždy iba súčasťou Vydavateľstva.



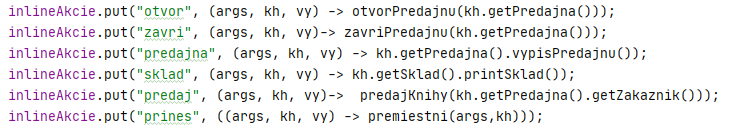
### Lambda výrazy

Jedno využitie lambda výrazov/refrencií na metódy sme implementovali v triede Vydavatelstvo.java, kde si ukladáme spôsob vydávania, teda vydá sa práve jedna kniha alebo všetky knihy čakajúce na zozname.



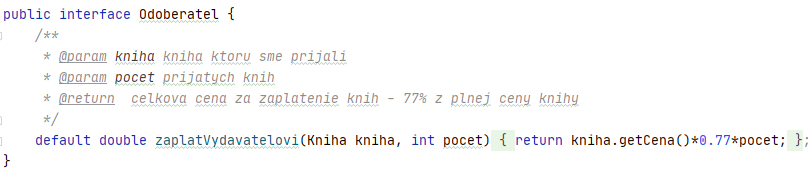


Ďalej sme lambda výrazy použili pri volaní funkcií používateľov

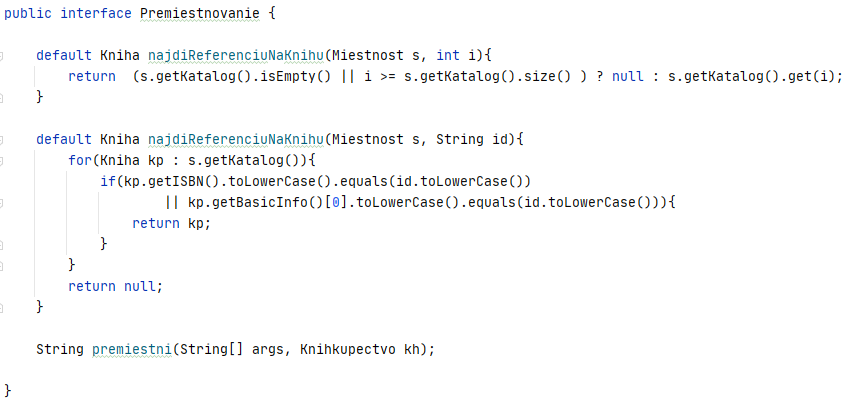


### Implicitná implementácia v rozhraní

Implicitné implementácie metód v rozhraniach sme využili napríklad v rozhraní Odoberatel:



Ale aj v rozhraní Premiestňovanie, ktoré implementujú triedy užívateľov kníhkupectva:



### Použitie serializácie

Serializáciu používame na uloženie stavu kníhkupectva, teda na uloženie stavu predajne, skladu, sekcií, regálov, poličiek, ale aj kníh v uložených v regáloch. Výsledok serializácie sa ukladá do súboru knihkupectvo\_oop.ser v priečinku /res/. Funkcie na serializáciu sú definované v Knihkupectvo.java:



Obrázok 1 Funkcie na serializáciu deserializaciu

Serializujú sa triedy:











# Verzie

**Commit - 6a16c955b1a998761afc408d17bf35fd213d173b** – *„rozdelenie controllera do viacerych casti“*

* Rozdelenie controllerov do viacerých tried
  + MainController – spracovanie vstupu od používateľa.
  + ViewController – spracovanie požiadaviek na úpravu view (výpis/otvorenie ďalšieho okna...)
  + ModelController – posielanie požiadaviek modelu
  + ButtonController – súčasť ModelControllera – spracovanie požiadaviek z tlačidiel.
* Prepojenie controllerov pomocou modelu observer.

**Commit - 8272aa2ea6d19d04c5caf858b657488b79711e8c –** *„Vytvorenie tabuliek v GUI, prepojenie niektorych tried pomocou observer“*

* Vytvorenie GUI tabuliek pre katalóg kníh v kníhkupectve, pre odoberateľov, pre texty prijaté na vydanie
* Vytvorenie tried na premieňanie String outputov na dáta vyložiteľné do tabuľky
* Vytvorenie interface Observer (implementujú Autor, Vydavatelstvo, Knihkupectvo a triedy dedené rozhraním Miestnosť)
* Prerobenie ako Controller posiela dáta View (pomocou Observer)
* Vytvorenie vlastných grafických elementov implementujúcich Observer.

**Commit - ab54cc26c8f8a19a047d17de55b63fc65e33b975 –** *“podokna presunuté z View do vlastných tried; vytvorenie tried BalikKnih, tried v users.info a vytvorenie viacerých stánkov”*.

* V tomto commit-e sme presunuli podokná z View do vlastných tried a ich vytváranie spracováva Controller.
* Ďalej sme pridali triedu BalikKnih a RadKnih, ktore nahrádzaju potrebu mať v triede, kde sa kumulujú knihy mať 2 polia na uloženie Knihy a jej počtu (v Distributor.java).
* Ďalej sme vytvorili viaceré druhy stánkov a aj možnosť ich vytvárať pomocou GUI. S novými stánkami upravené funkcie DajOdoberatelom().
* Vytvorenie triedy UdajeOZamestnancovi pre uchovanie údajov o zamestnancoch (plat a odrobený čas)
* Vytvorenie triedy users.info.Inventar, ktorá slúži na uchovanie knihy u zamestnanca knihkupectva
* Vytvorenie triedy stanky.Inventar na uloženie všetkých kníh, ktoré má stánok.

**Commit - 30d0cce22bc7d08e58d3fbb43f7fe17381f749d3 –** *„knihkupectvo gui - v1.0“*

* Dokončenie prvej verzie GUI – podporované iba kníhkupectvo

**Commit - 2376a809d83267c261df1c929bf364c0e49f2687** – *„vytvorenie vydávania“*

* Nastavenie getterov a setterov v triede Text.java
* Úprava Kníhkupectva aby dokázal prijať knihy od Vydavateľstva
* Implementované metódy zamestnancov Vydavateľstva a implementácia Vydavateľstva a Tlačiarne
* Použitie návrhového vzoru visitor pre Autor.
* Vytvorenie nite pre autorovo písanie v Autor.java

**Commit - 92d987fc3f86b5866c0ee725318bf8edebf522a7 –** „*gui try”*

* Úprava triedy Kniha.java na návrhový model Composite. Rozdelenie do tried Kniha, Text a Obálka.
* Úprava funkcií na načítanie kníh zo súboru, aby používali novú úpravu.
* Pridanie triedy Organizovaná sekcia (použitá v predajni)
* Vytvorenie súborov tried pre vydavateľstvo
* Pridanie atribútov triede kniha

**Commit - 1cfd4a4f8210c2c375dfc52e72b6c91117de898b –** „*predajňa dokončená”*

* Dokončenie funkcií predajcu a zákazníka
* Vstup zákazníka do predajne
* Funkcie na nájdenie referencie na knihy v kníhkupectve.

**Commit - b9d82d1537f8754e32b7346ffd769d743c3e5***b22 – „ui overhaul”*

* Spustenie funkcií používateľov pomocou lambda funkcií.
* Implementácia niektorých funkcií predajcu

**Commit - b9d82d1537f8754e32b7346ffd769d743c3e5b22 –** „*serialization”*

* Implementácia serializácie, ukladanie dát kníhkupectva.

**Commit - 28362d3be8e9a6641e4257419f574e2214ef5277 –** „*UI for skladnik”*

* Vytvorenie druhej verzie prijímania vstupu od používateľa
* Vstup mohli byť funkcie skladníka, zamestnanca alebo používateľa

**Commit - 8f93454d7817df98b00747359f5421e019e9f382 –** „*viacnásobne dedenie”*

* Vytvorenie abstraktnej triedy Zamestnanec pre lepšie rozoznanie Zamestnancov od Zákazníkov
* Umiestňovanie kníh do regálov v sklade.

**Commit - 917c70f42e6126a012a89fd2a8040503cdbbd6d8** *–„ ja nechápem“*

* Prvá verzia prijímania vstupu z konzole
* Implementácia niektorých funkcií skladnika
* Jednoduché implementácie funkcií Predajcu a abstraktnej triedy Pouzivatel
* Implementácia ďalších funkcií skladu.

**Commit - d44447e21aa4736dcf34876c5cb5dc85ec50ebec –** „*zoop projekt”*

* Prvá verzia aplikácie
* Vytvorenie kníhkupectva a užívateľov aplikácie: (Skladník, Predajca, Zákazník)
* Vytvorenie regálov, sekcií a skladu
* Predajňa nie ešte dokončená
* Objednávanie tovaru iba zo súboru