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

16.5.2021

Id: 102970

# Obsah

[Obsah 2](#_Toc72012983)

[1. Rámcové zadanie 3](#_Toc72012984)

[2. Štruktúra 4](#_Toc72012985)

[3. Kritéria 5](#_Toc72012986)

[3.1. Hlavné kritéria 5](#_Toc72012987)

[3.1.1 Náplň zadania 5](#_Toc72012988)

[3.1.2 Prvá hierarchia 5](#_Toc72012989)

[3.1.3 Druhá hierarchia 10](#_Toc72012990)

[3.1.4 Oddelenie aplikačnej logiky od používateľského rozhrania 14](#_Toc72012991)

[3.1.5 Organizácia do balíkov 14](#_Toc72012992)

[3.2. Ďalšie kritéria 15](#_Toc72012993)

[3.2.1 Návrhové vzory 15](#_Toc72012994)

[3.2.2 Výnimky 22](#_Toc72012995)

[3.2.3 GUI 26](#_Toc72012996)

[3.2.4 Multithreading 27](#_Toc72012997)

[3.2.5 RTTI 27](#_Toc72012998)

[3.2.6 Vhniezdené triedy 28](#_Toc72012999)

[3.2.7 Lambda výrazy 28](#_Toc72013000)

[3.2.8 Implicitná implementácia v rozhraní 29](#_Toc72013001)

[3.2.9 Použitie serializácie 30](#_Toc72013002)

[4. Verzie 31](#_Toc72013003)

# 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.

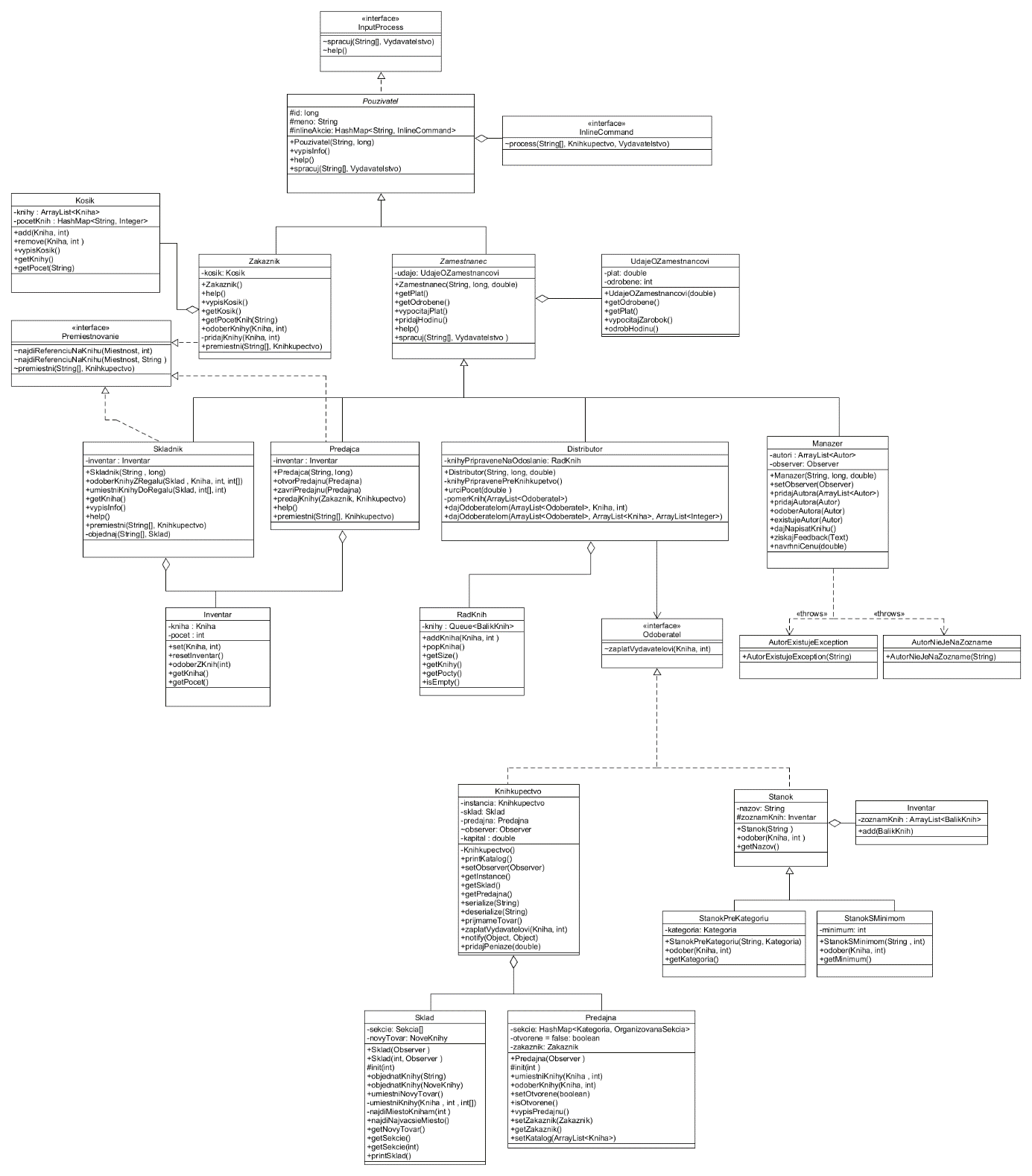
Autor musí najprv knihu napísať – dostane nápad, napíše ju, vymyslí názov. Potom dá svoju knihu vydavateľstvu.

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, zistí aký dopyt je po knihe a nakoniec distribútor povie koľko kníh sa má vytlačiť. V tomto procese sa knihy dajú na tlač, kde sa text pripevní k väzbe. Potom distribútor rozdelí knihy a pošle ich 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 dať knihy do košíka, ktoré mu následne predajca predá.

# Štruktúra

Obsahuje dve hlavné hierarchie dedenia Používateľov a odoberateľov.



# Kritéria

## Hlavné kritéria

### Náplň zadania

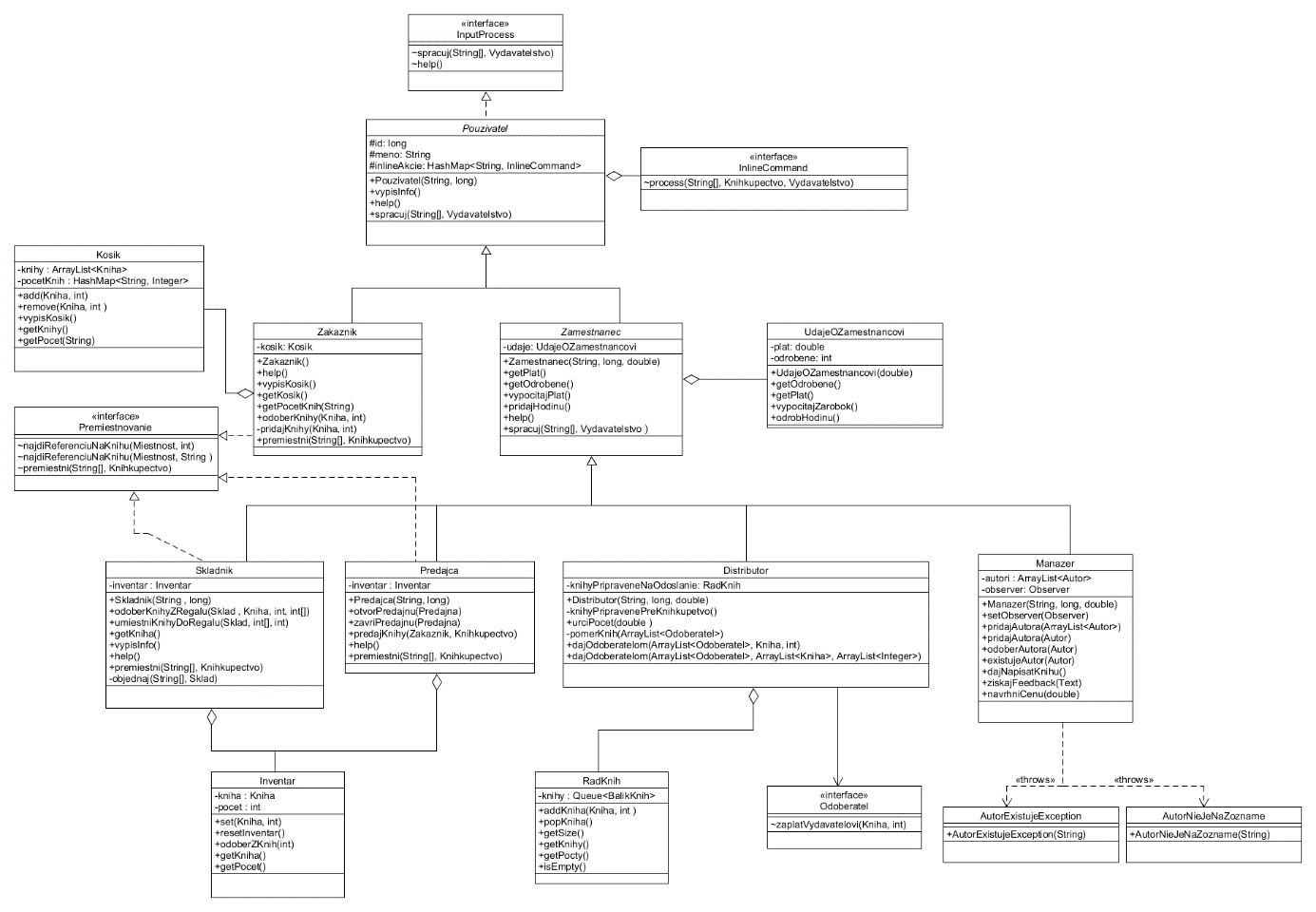
Podľa nášho názoru sa nám podarilo splniť ciel zadania, čo bolo vytvorenie procesu vydávania kníh ako aj plánovania ich distribúcie ceny a počtu vytlačených kníh. Knihy prijíma Kníhkupectvo iba ak sú splnené podmienky, že sa pošlú 4 rôzne knihy v hocijakom počte kusov a kníhkupectvo je schopné prijať nový tovar do svojho skladu. Ale napríklad na stánky sa nedávajú takéto limity, tie sú schopné prijať hocikedy, ale sú aj stánky, ktoré neprijmú knihy, ktoré nespĺňajú ich požiadavky, napr. prijímajú iba určitú kategóriu.

Proces vytvorenia knihy začína u manažéra, ktorý pošle požiadavku autorom, aby napísali knihu. Tí o svojom postupe informujú vydavateľstvo a následne pošlú texty vydavateľstvu, kde sa pridajú do radu čakania na vydanie. Manažér si môže vybrať akým spôsobom bude vydávať knihy, napr. vydanie iba jednej knihy alebo vydanie všetkých kníh na zozname. Keď sa vyberie kniha, tak najprv dizajnér navrhne obálku, následne si korektor prečíta text a opraví v ňom chyby a skráti text. Následne manažér, zistí aký je po knihe dopyt, podľa čoho sa určí počet výtlačkov a cena. Ešte pred samotným tlačením knihy, tlačiareň vytvorí ISBN identifikátor pre text a nakoniec tlačiareň spojí obálku a text do jednej knihy. Potom sa buď najprv kniha vloží do zoznamu kníh pripravených na distribúciu, s ktorým potom distribútor pracuje (ak vydávame všetky texty) alebo sa kniha pošle hneď distribútorovi na odoslanie odoberateľom. Distribútor určí pomer v akom sa budú knihy posielať odoberateľom a následne ich im pošle. Nepredané knihy sa vyhadzujú.

### Prvá hierarchia

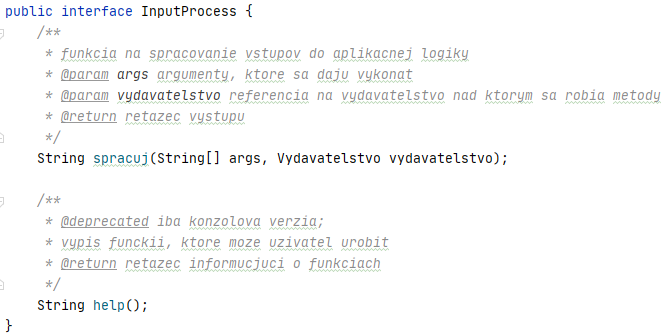
Prvá hierarchia dedenia je hierarchia zamestnancov/používateľov.

Na vrchu je rozhranie InputProcess, ktorý je implementovaný abstraktnou triedou Používateľ, následne táto trieda je dedená ďalšou abstraktnou triedou Zamestnanec a triedou zákazníka. Zamestnanca následne dedia Skladník, Predajca, Distribútor a Manažér, (aj vnorené triedy Korektor a Dizajnér).



**Rozhranie**

Používatelia implementujú rozhranie inputProcess, kde je zadefinovaná funkcia *String spracuj(String[] args, Vydavatelstvo vydavatelstvo)* na spracovanie príkazov.



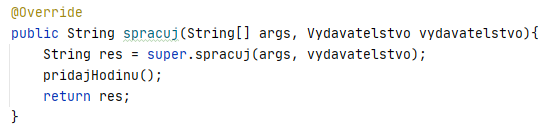


**Polymorfizmus**

Polymorfizmus sa v tejto triede vyskytuje tak, že Používateľ implementuje spôsob akým sa spúšťajú funkcie pomocou funkcie používateľa cez funkciu spracuj():



Následne táto metóda sa prekonáva v triede Zamestnanec, kde spracováva funkcie rovnako ako používateľ, ale pri každom zavolaní zvýši zamestnancovi počet odrobených hodín:

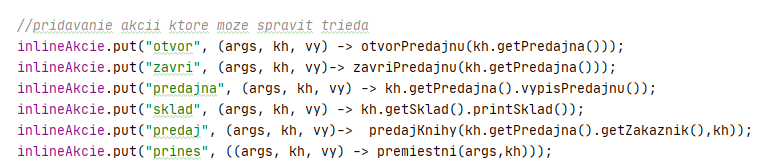


Funkcia je vyvolávaná z triedy model.java, kde sa pri zmenách používateľa nastaví do premennej typu Pouzivatel práve prihlásený používateľ. Z tejto premennej sa volá funkcia spracuj:



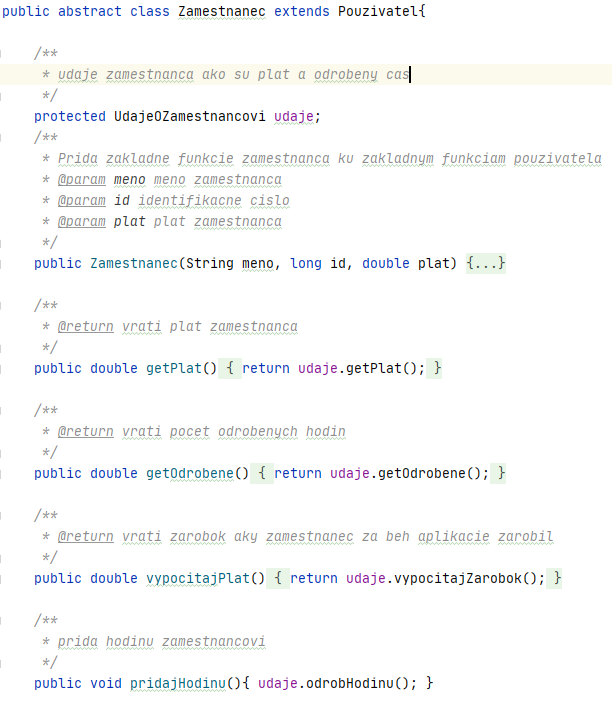


Funkcie sú zadávane do hašovacej tabuľky, kde kľuč tvoria názvy funkcií a ukladajú sa lambda výrazy, ktoré spúšťajú funkcie. Napr. funkcie predajcu:

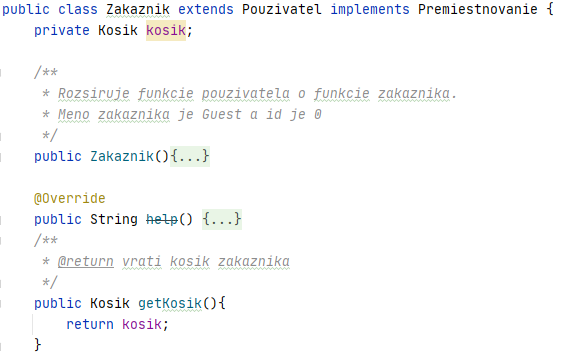


**Zapuzdrenie**

Každý atribút v hierarchii je privátny, prípadne pri niektorých sme použili protected. Ku mnohým atribútom ani nie je možne pristúpiť cez gettery a settery, keďže zostávajú iba vo funkciách. Prípadne, ak sa v atribútoch menia hodnoty, tak sa volajú funkcie, ktoré ich prepíšu:



Príklad getteru v hierarchii:



Setter v hierarchii:

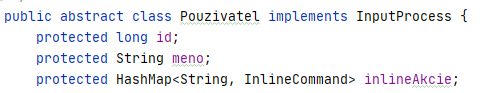






**Agregácia**

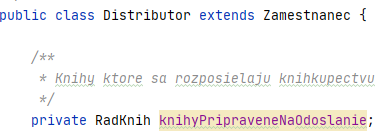
Agregácia bola použitá napr. v abstraktnej triede Pouzivatel, kde sa agregujú funkcie používateľov. Ďalej v Zamestnancovi sa agreguje trieda UdajeOZamestnancovi; v Zákaznikovy je trieda Kosik; v Predajcovi a Skladníkovi sa agreguje inventár, v Distribútorovi sa agreguje trieda RadKnih a v Manažérovi sa agregujú autori, ktorých upozorňuje.

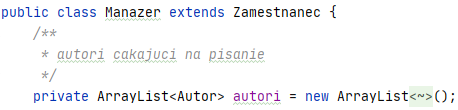






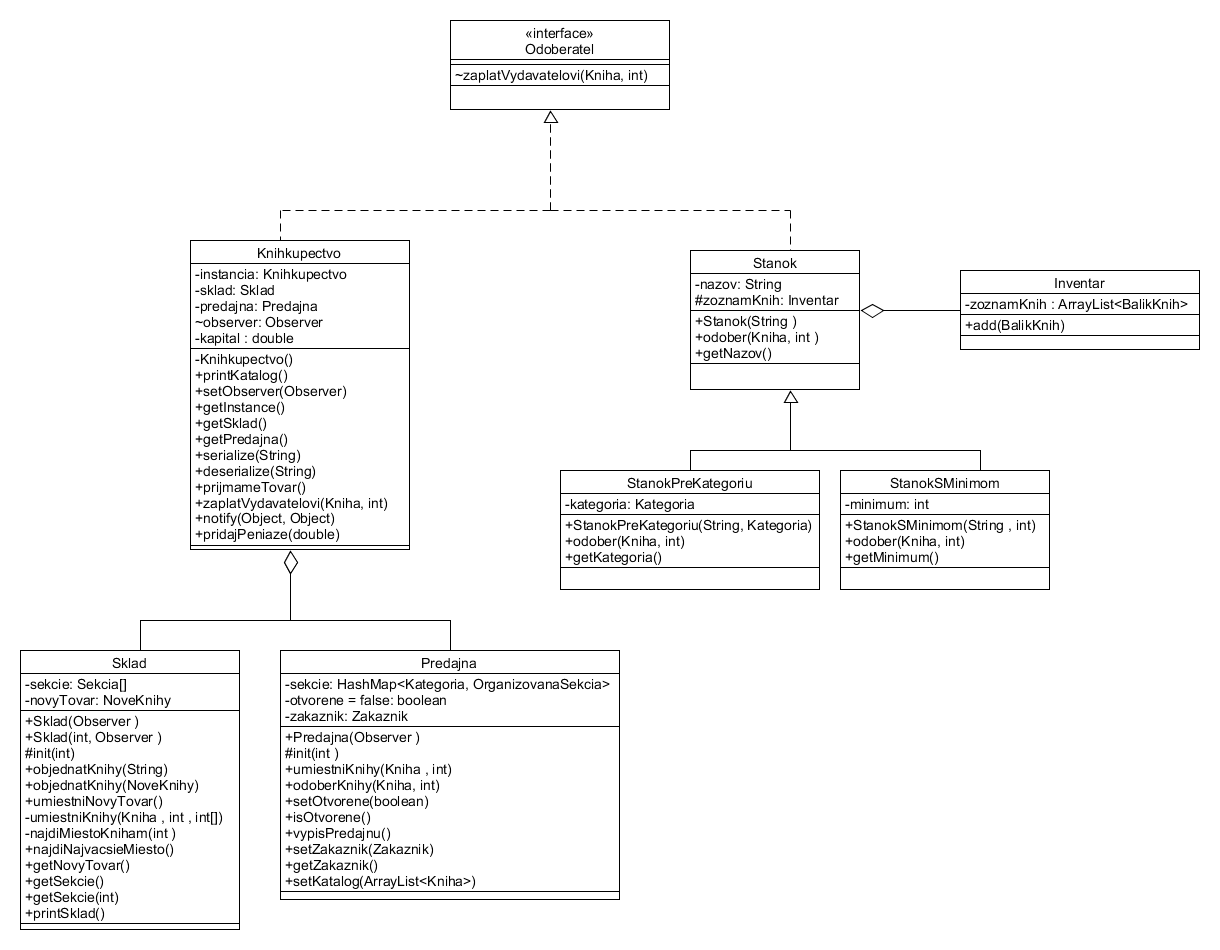






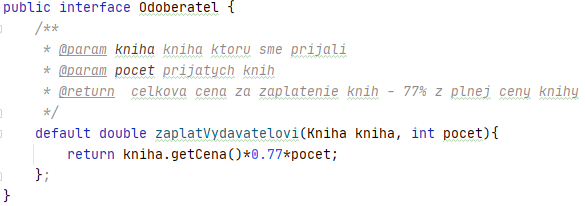
### Druhá hierarchia

Ako druhú hierarchiu máme odoberateľov, kde máme zadefinované rozhranie Odoberatel, ktoré implementujú triedy Knihkupectvo a Stanok. Ďalej je stánok dedený triedami StanokSMinimom a StanokPreKategoriu



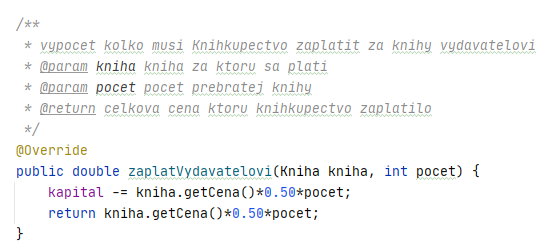
**Rozhranie**

Rozhranie v tejto triede definuje implicitne funkciu na zaplatenie vydavateľovi, ktorá vracia 77% percent ceny knihy.



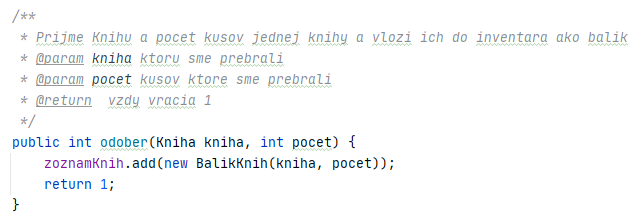
**Polymorfizmus**

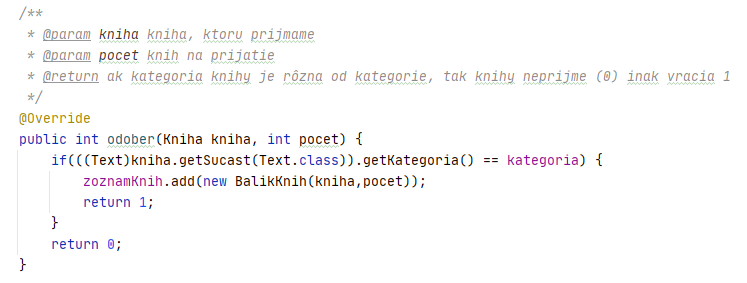
Metódu definovanú v rozhraní prekonáva kníhkupectvo, kde si platí iba 50% knihy a odpočítava si zo svojho kapitálu.



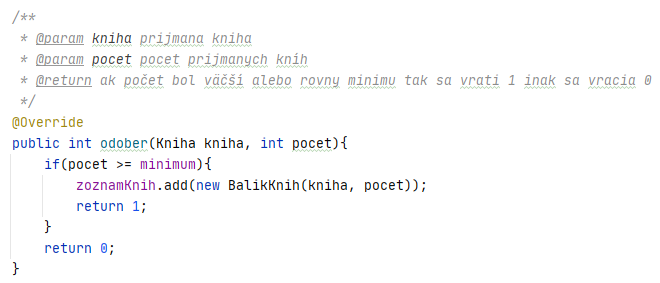
Ďalej v triede Stanok je metóda odober(...), ktorá pridá knihy na zoznam kníh bezpodmienečne, ale deriváty stánku napr. StanokSMinimom príma knihy len v prípade, že počet posielaných kníh presiahol minimum.

Stanok.java:



StanokPreKategoriu.java:

StanokSMinimom.java:



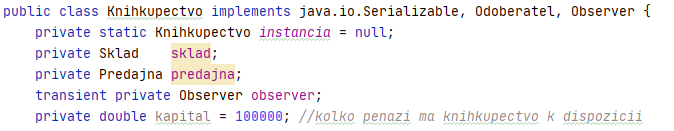
Využité v Distributor.java vo funkcií dajOdoberatelom(...):

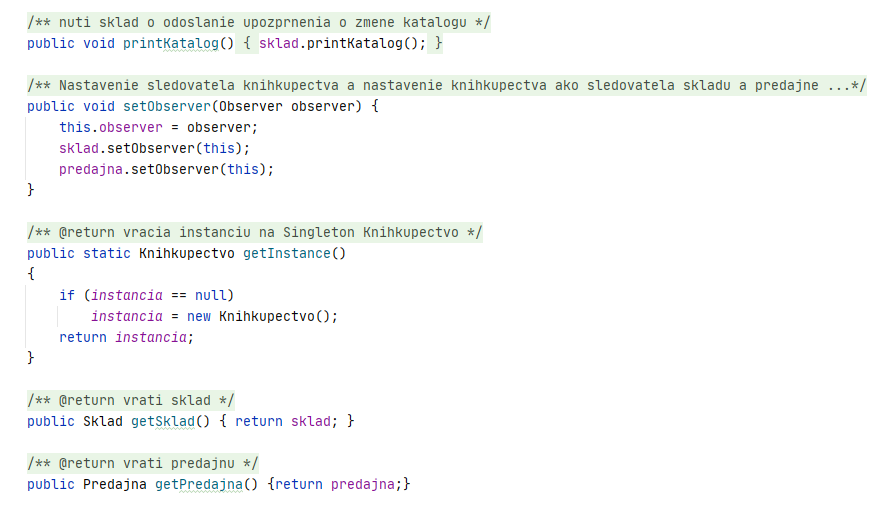


**Zapuzdrenie**

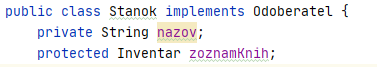
Každý atribút v kníhkupectve je private a v Stanku je názov private a inventar je protected, pretože, každý ďalší stánok potrebuje pracovať s inventárom, ale už si nebudú meniť názov.

Príklad z Knihkupectvo.java:



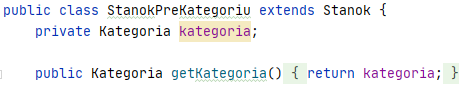


Stanok.java:

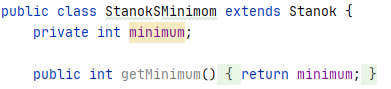




StanokPreKategoriu.java:



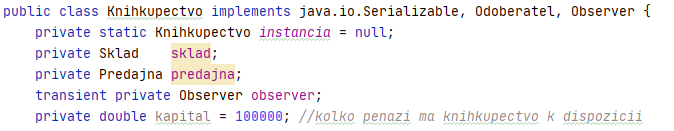
StanokSMinimom.java:



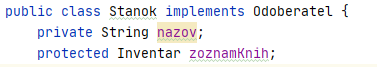
**Agregácia**

V kníhkupectve sa agreguje sklad, predajňa a objekty implementujúce observer. Ďalej v Stánku sa agreguje trieda inventár.

Knihkupectvo.java

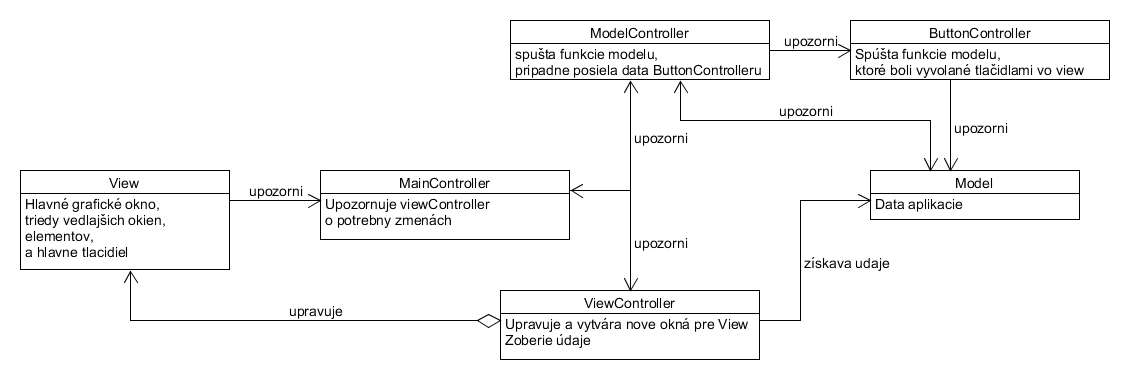


Stanok.java



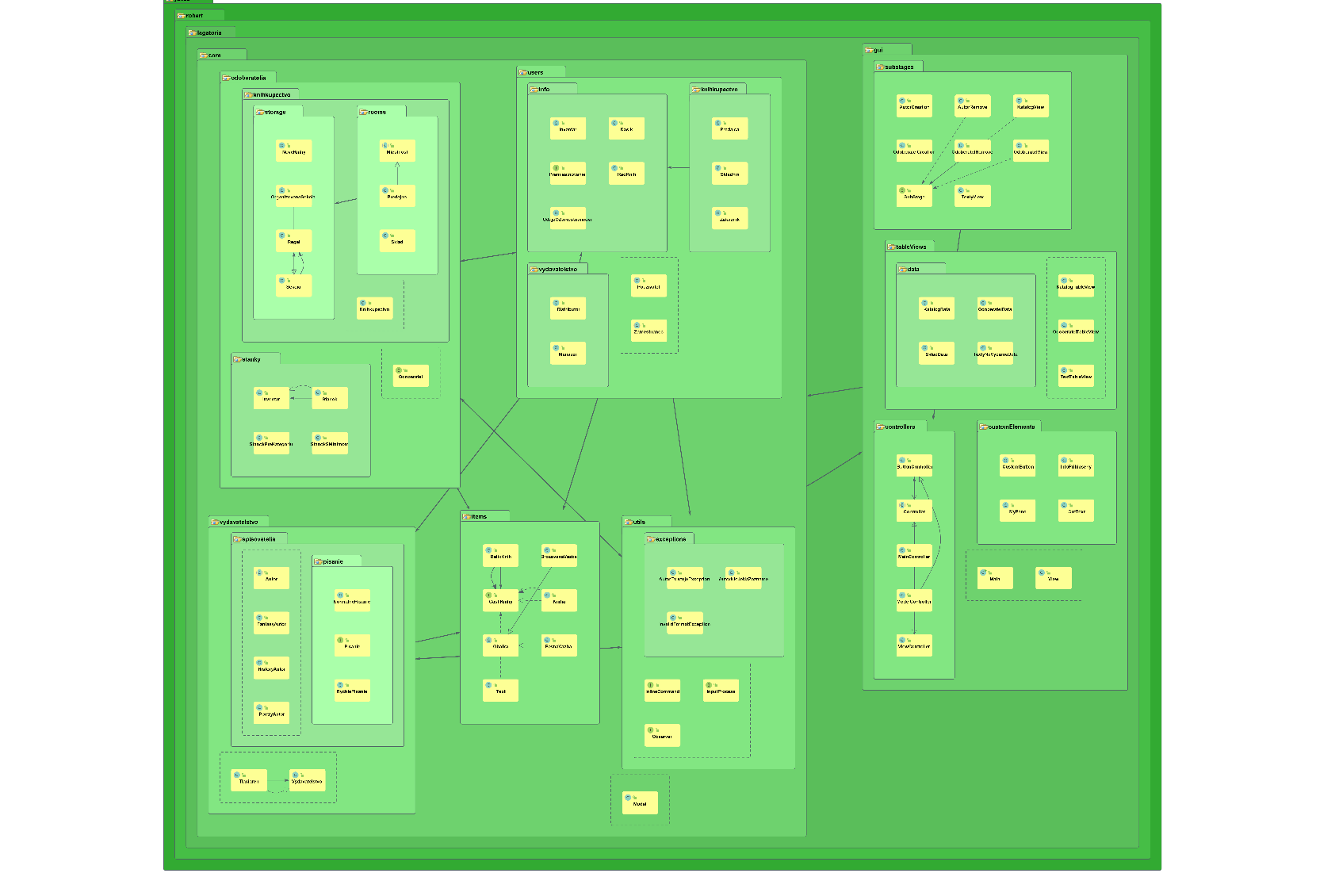
### Oddelenie aplikačnej logiky od používateľského rozhrania

Oddelenie aplikačnej časti od rozhrania sme spravili pomocou MVC modelu, kde aplikačná logika a rozhranie je rozdelené tak, že model nijako nepracuje s rozhraním a ani naopak. Namiesto toho komunikujú cez kontroller pomocou modelu observer.



### Organizácia do balíkov

* Junas.robert.lagatoria
  + core – triedy aplikačnej logiky.
    - Items – vytvárané produkty v našom prípade knihy.
    - Odoberatelia – triedy odoberateľov.
      * Knihkupectvo – triedy potrebne na správne fungovanie knihkupectva.
        + Rooms – miestnosti kníhkupectva.
        + Storage – ukladacie priestory, napr. regal.
      * Stanky – triedy stánkov a inventár stánku.
    - Users – triedy používateľov a triedy pre ich správne fungovanie
      * Info – triedy obsahujúce triedy ako inventár, UdajeOZamestnancovi atď.
      * Kníhkupectvo – používatelia kníhkupectva.
      * Vydavateľstvo – používatelia vydavatelstva.
    - Utils – pomocne triedy.
      * Enums – pomocne enumerácie.
      * Exceptions – triedy našich výnimiek.
    - Vydavateľstvo – obsahuje triedy na správne fungovanie vydavateľstva.
      * Spisovatelia – triedy autorov, ktorý môžu písať
        + písanie – obsahuje visitor triedy spôsobu ako autori píšu.
  + gui – triedy pre správne fungovanie grafického rozhrania.
    - controllers – ovládače na prepojenie modelu so zobrazením.
    - customElements – triedy vychádzajúce z javaFx elementov, napr. tlačidla, text, atď.
    - substages – sekundárne okná napr. tabuľkové zobrazenie, pridávanie autorov atď.
    - tableViews – definovanie tabuliek aké vieme zobrazovať
      * data – triedy na parsovanie dát z modelu do tabuliek.

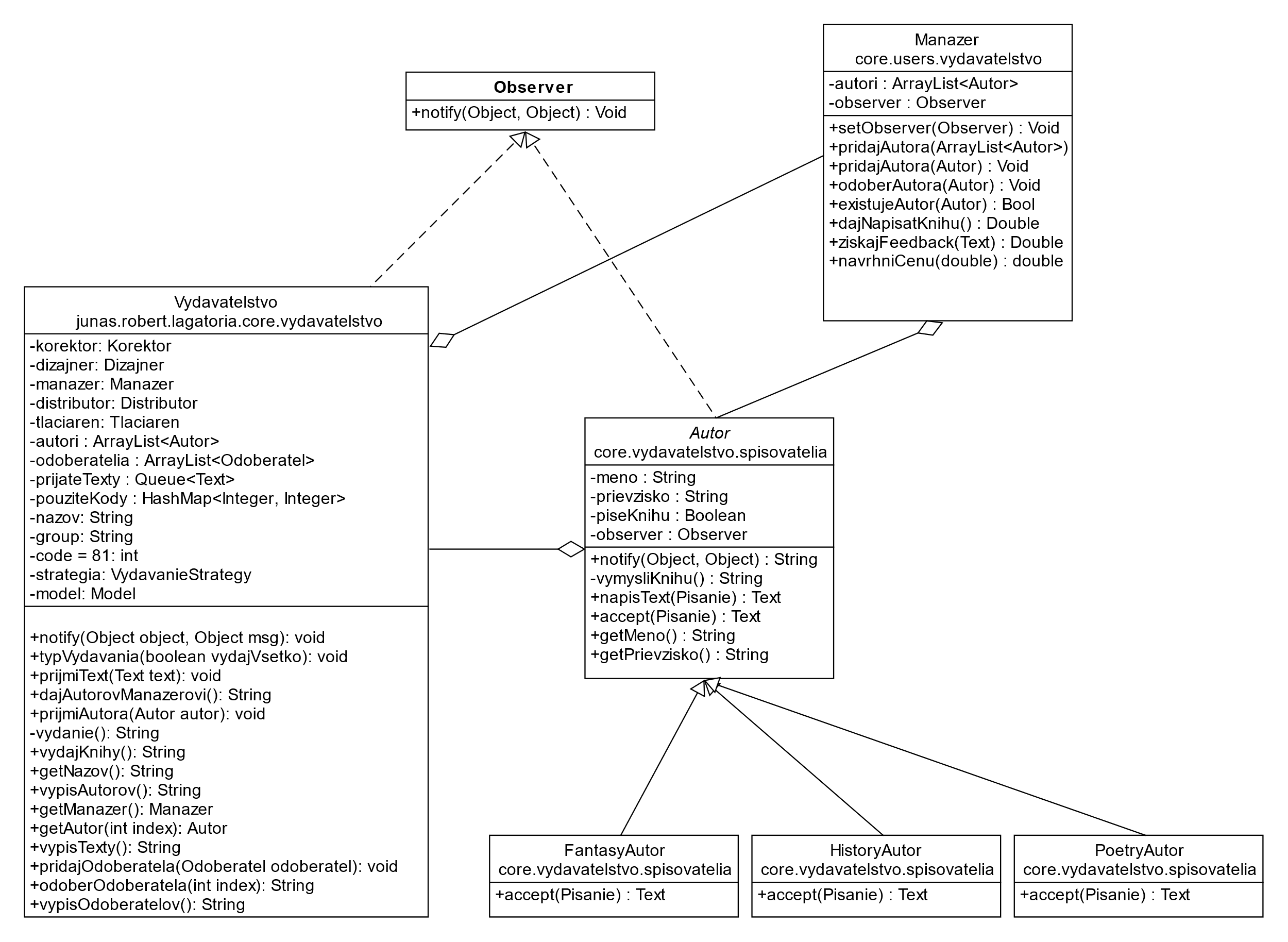


## Ďalšie kritéria

### Návrhové vzory

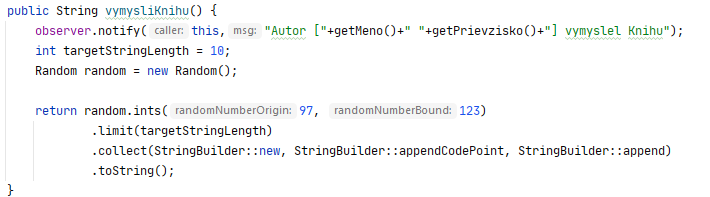
**Observer**

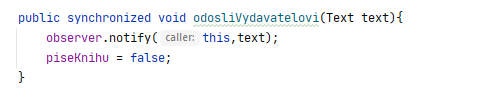
Model Observer sme využili naprieč celým projektom, či už na komunikáciu ovládačov s View alebo Modelom, ale aj medzi triedami Manažér a Autor, kde Manazer upovedomí svojich autorov, že chce aby napísali knihu.



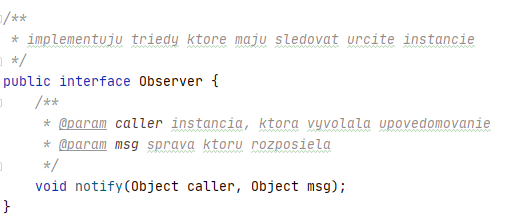


Observer bol využitý medzi autormi a vydavateľstvom, kde autor upovedomuje vydavateľstvo o svojom postupe v písaní textu.



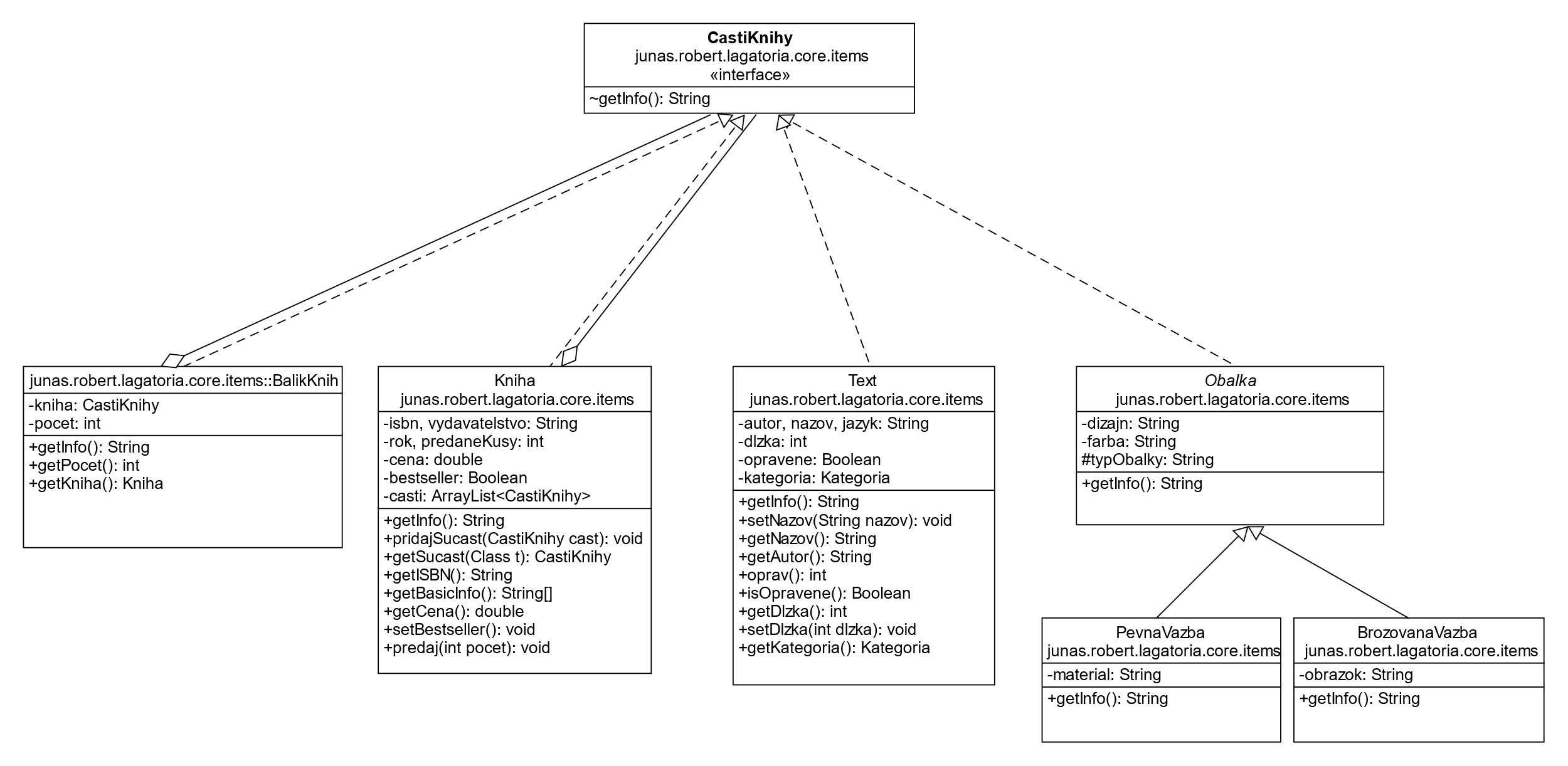


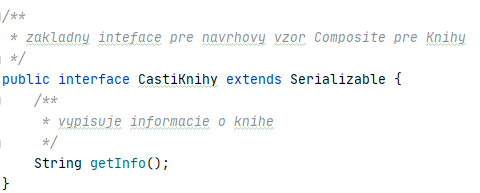
Máme definované aj rozhranie Observer, kde je definovaný prototyp funkcie notify(Object, Object).



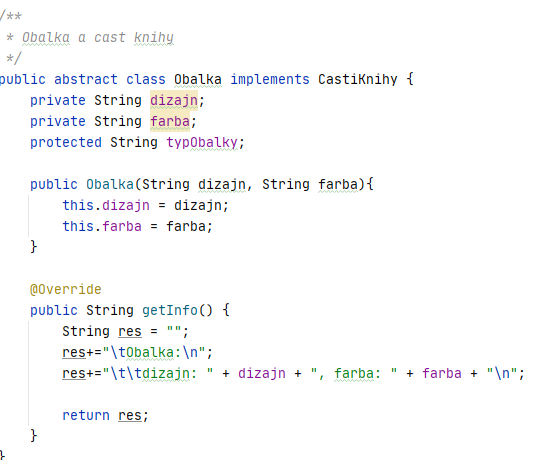
**Composite**

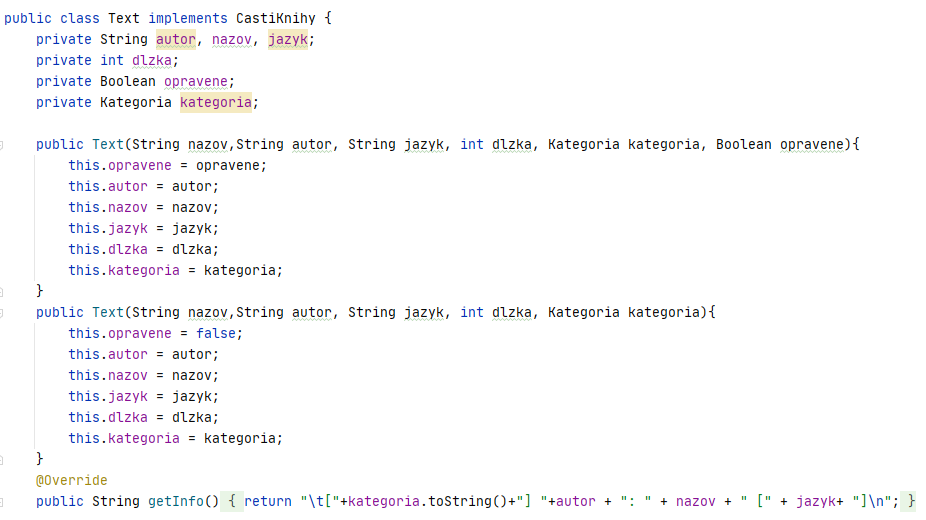
Využívaný v spojení s knihami, kde kniha sa skladá z textov a obálky a nakoniec aj knihy sa balia do balíkov, kde je ich väčší počet.





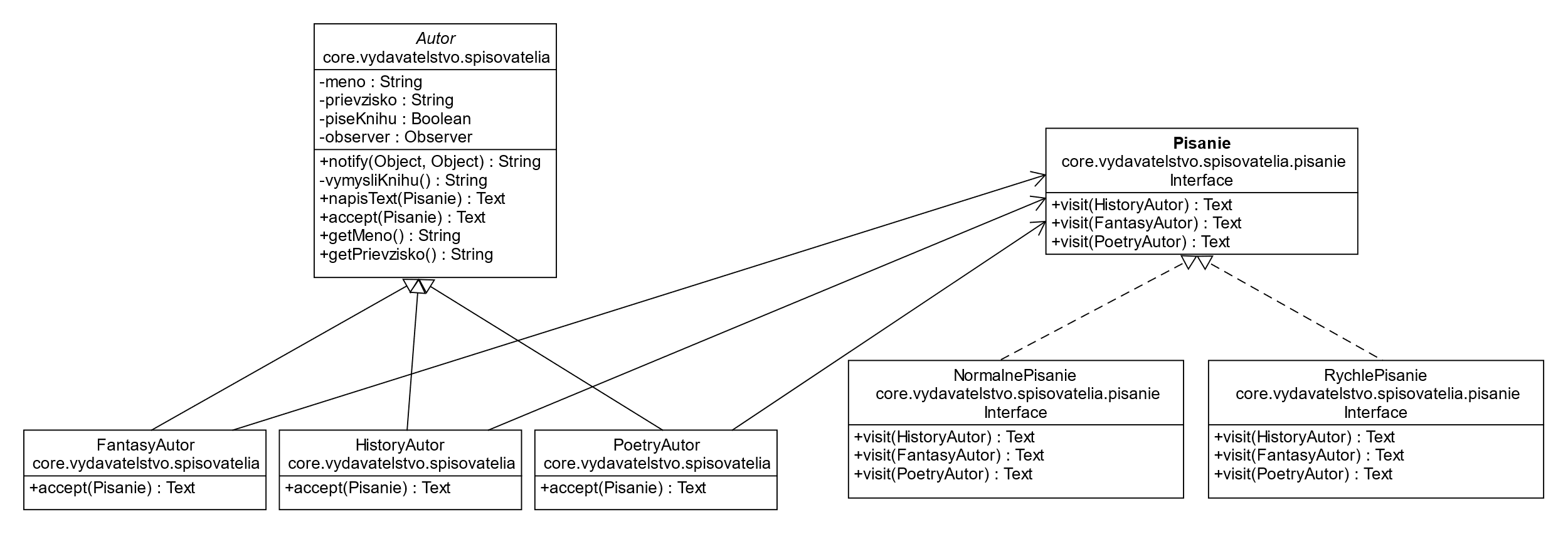


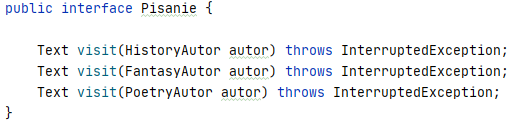




**Visitor**

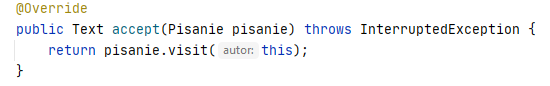
V projekte sa vyskytuje aj model Visitor a to tak, že každý druh autora píše iným spôsobom, teda trvá mu to kratšie alebo dlhšie, vzhľadom koľko práce musí dať do samotného písania. Máme definované dva spôsoby písania: Normálne a rýchle písanie. Počet strán sa odráža od rýchlosti akou autor písal.



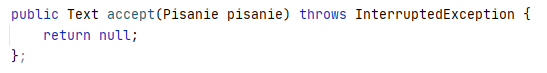








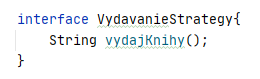
Použitie v triede Autor.java



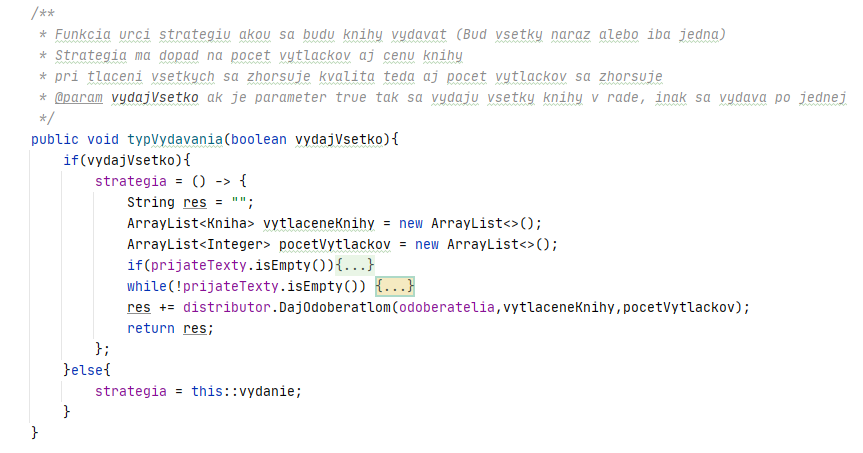


**Strategy**

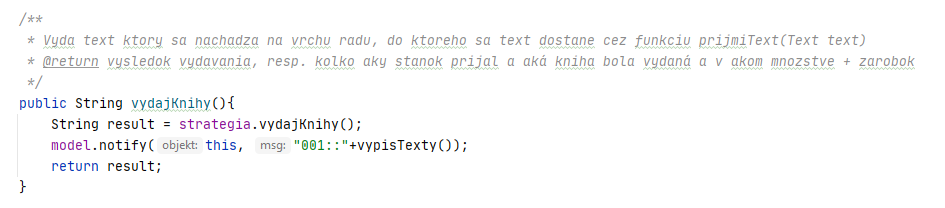
V projekte je využitý návrhový vzor Strategy, ktorý určuje stratégiu akou sa budú vydávať knihy. Je definovaný ako vnorené rozhranie triedy Vydavatelstvo.java, kde sa určuje akou formou sa bude vydávať.



Zmena stratégie vydávania, buď vydanie všetkých kníh alebo na vydanie jednej knihy, sa mení pomocou referencie na funkciu a lambda výrazov.



Zavolanie funkcie uloženej v premennej typu VydavanieStrategy vo Vydavatelstvo.java:

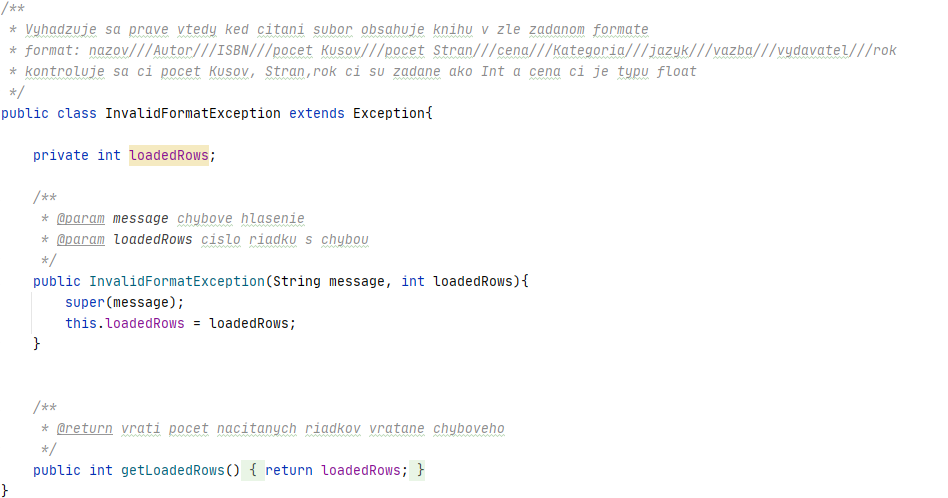


### Výnimky

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

InvalidFormatException sa vyhadzuje práve vtedy, keď v súbore z ktorého načítavame knihy, nie sú knihy zadané v správnom formáte. Správny formát je:

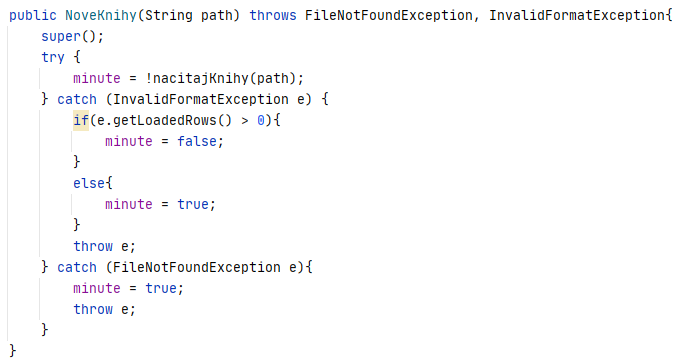
Nazov///Meno Prievzisko///ISBN///Kusy///Strany///Cena///Kategoria///Jazyk  
Nazov///Meno Prievzisko///ISBN///Kusy///Strany///Cena///Kategoria///Jazyk

InvalidFormatException:

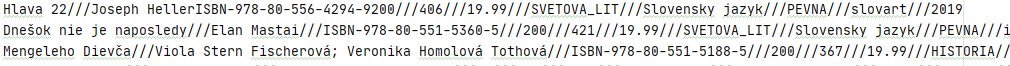
Funkcia vyhadzujúca výnimku:



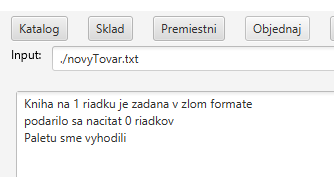
**Konštruktor chytá výnimku:**



Preukázanie vyhadzovania:

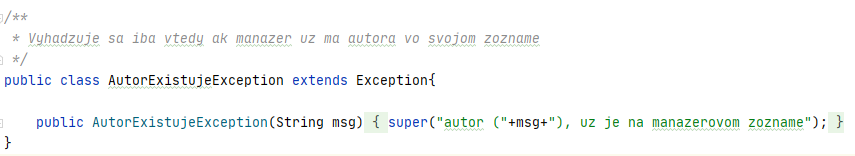


prvý riadok je v zlom formáte

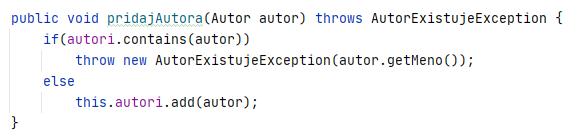


Výnimka **AutorExistujeException**  sa vyhadzuje práve vtedy, keď dáme pridanie autorov na manažérov zoznam autorov čakajúcich na písanie.

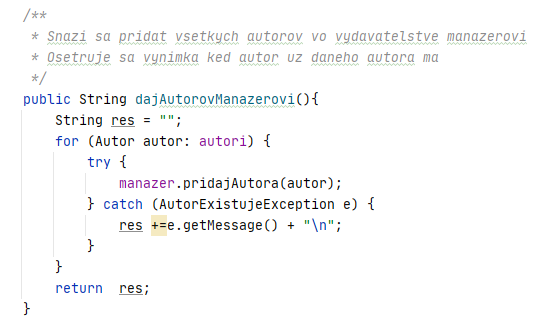
**AutorExistujeException:**



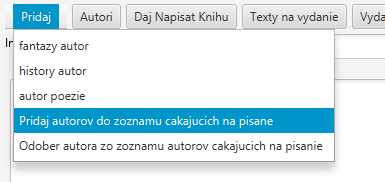
Funkcia vyhadzujúca výnimku:

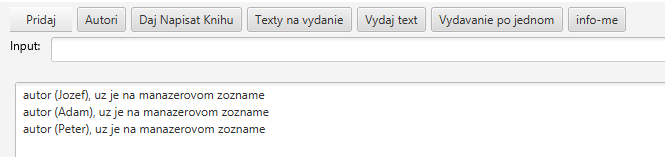


Metóda chytajúca výnimku:



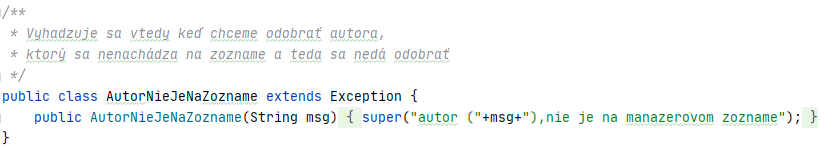
Preukázanie vyhadzovania:



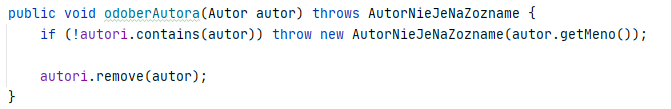


Výnimka **AutorNieJeNaZozname**  sa vyhadzuje práve vtedy, keď sa snažíme odobrať autora zo zoznamu čakajúcich na písanie.

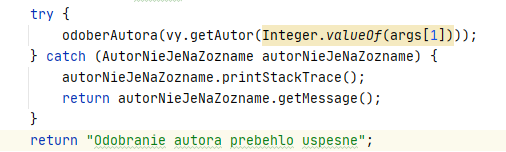
**AutorNieJeNaZozname:**

****

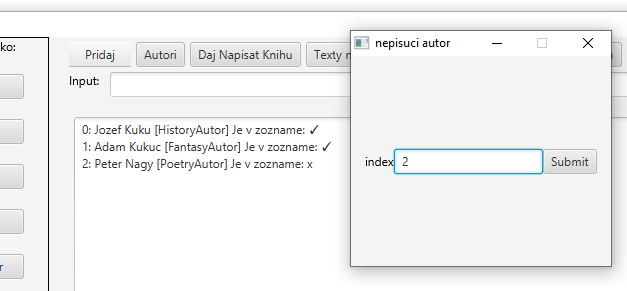
Funkcia vyhadzujúca výnimku:

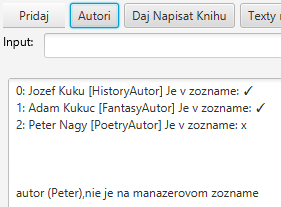


Metóda chytajúca výnimku:

****

Preukázanie vyhadzovania:





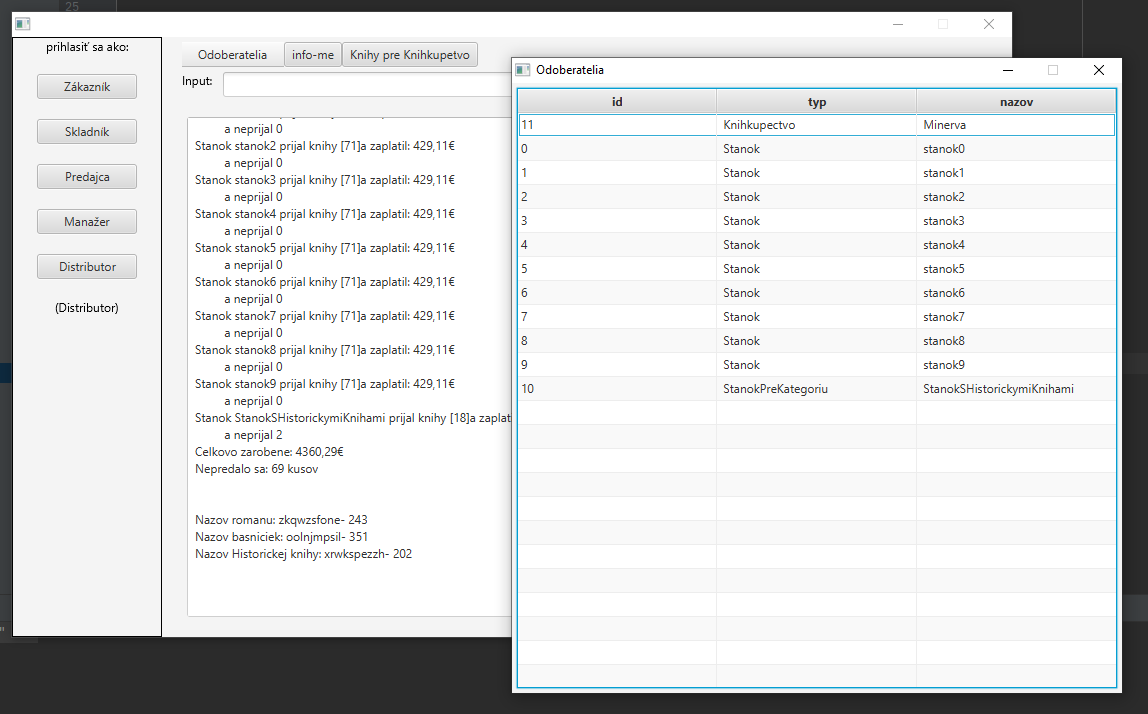
### GUI

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.

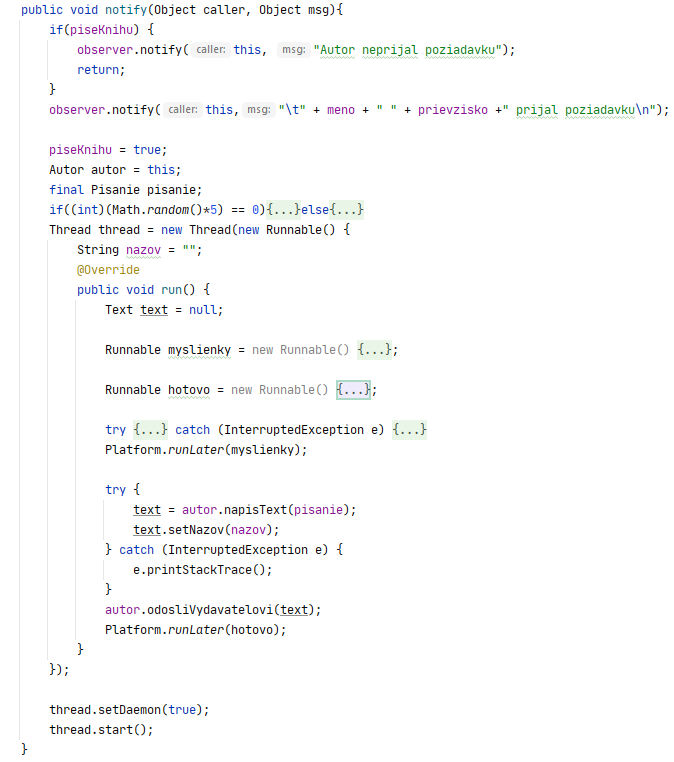
Model samotný sa kumuluje v Model.java, kde sa vykonáva celý model a sám o sebe nijako nezasahuje do View, prípadne upovedomí Controller o zmene, ktorú treba vykonať.

Celé grafické rozhranie bolo vytvorené ručne, bez žiadnych nástrojov na vytváranie prostredí.



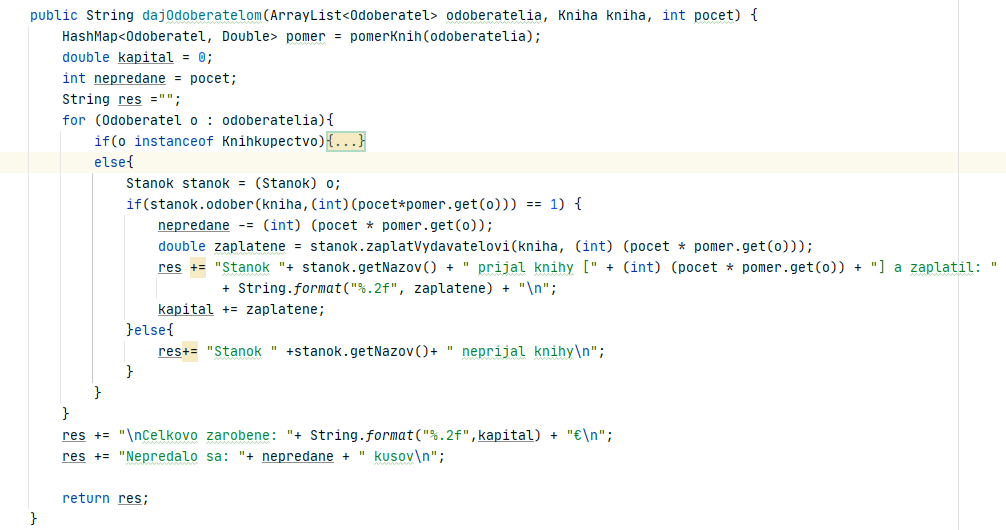
### Multithreading

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



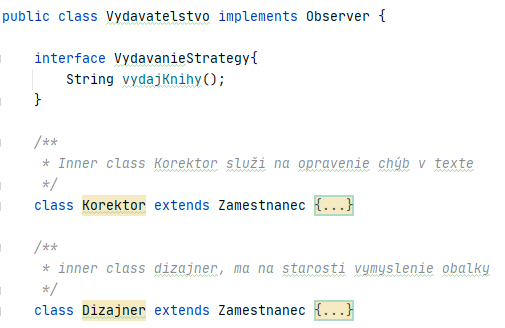
### RTTI

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



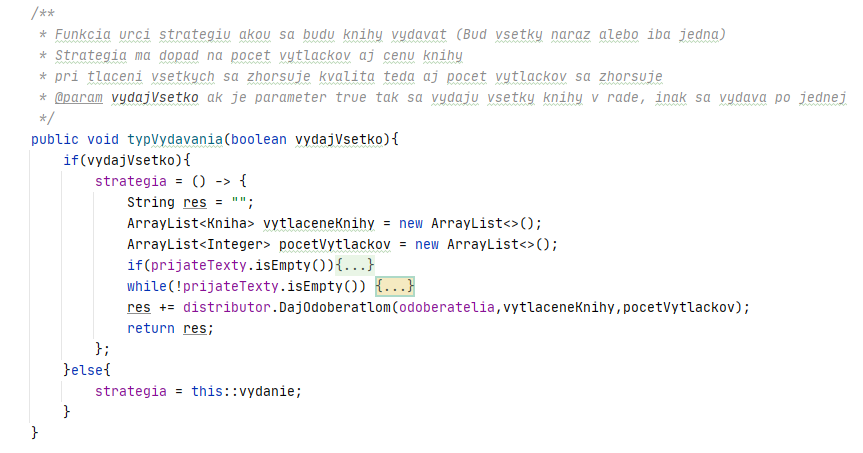
### Vhniezdené triedy

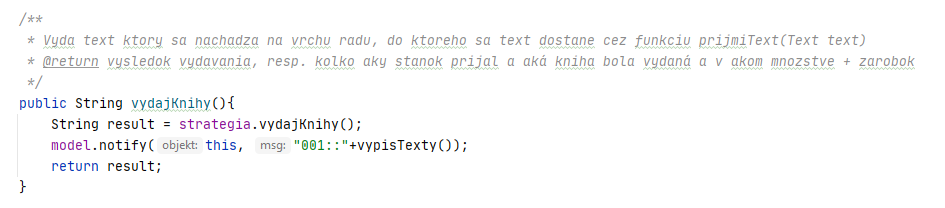
Vhniezdené triedy a rozhrania sa nachádzajú v triede Vydavatelstvo.java. Implementujeme tu stratégiu vydávania ako aj triedy Korektor a Dizajnér, ktoré by mali byť vždy iba súčasťou Vydavateľstva.



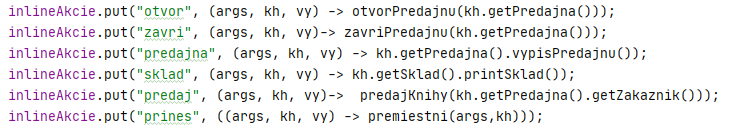
### Lambda výrazy

Jedno využitie lambda výrazov/referencií 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.



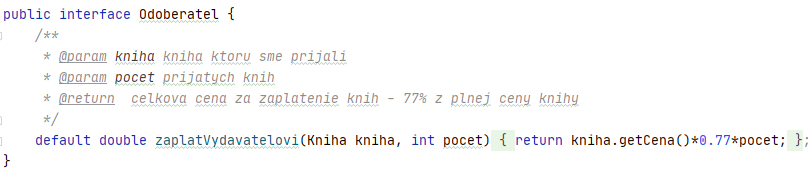


Ďalej sme lambda výrazy použili pri volaní funkcií používateľov

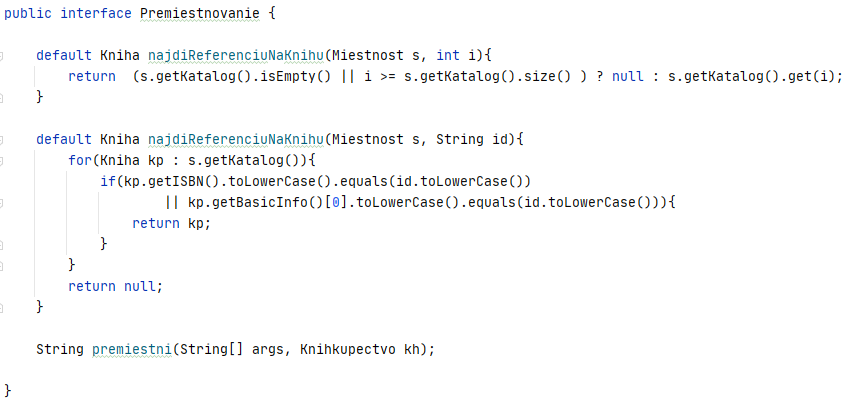


### Implicitná implementácia v rozhraní

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



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. Kníhkupectvo sa deserializuje pri spustení programu a serializuje sa, keď sa stlačí tlačidlo na vypnutie okna/programu. Výsledok serializácie sa ukladá do súboru knihkupectvo\_oop.ser v priečinku /res/. Funkcie na serializáciu sú definované v Knihkupectvo.java:



Serializujú sa triedy:







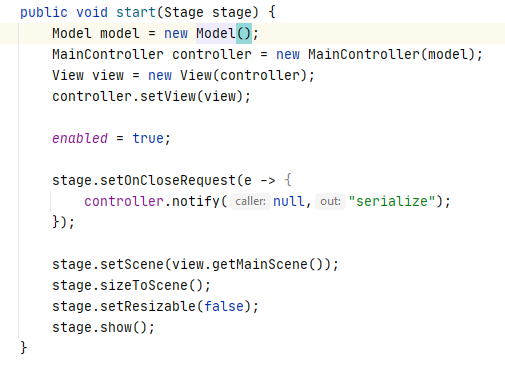




Volanie funkcii na de/serializáciu:

Keď sa inicializuje objekt Model tak sa deserializuje kníhkupectvo zo súboru. Serializuje sa vtedy, keď hlavné okno dostane požiadavku na vypnutie okna.





# 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, ktoré nahrádzajú 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 kníhkupectva
* 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