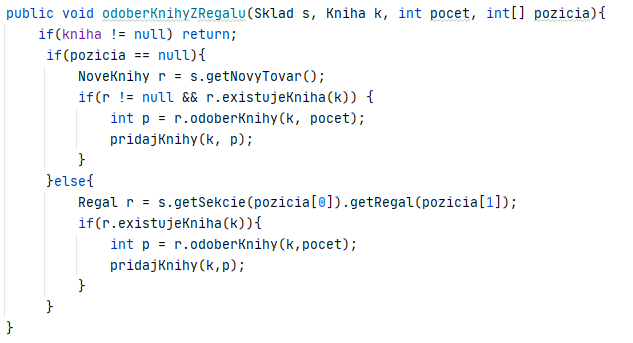
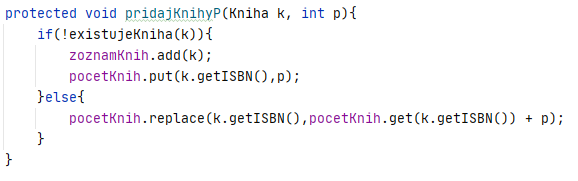
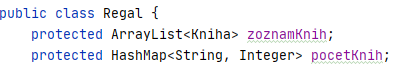
# Princípy OOP

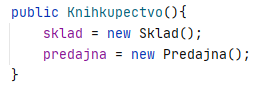
**Asociácia –** používam hlavne v triedach používateľov, menovite v **skladnik.java**, kde veľa funkcií vyžaduje objekt **sklad** a **kniha** na správne fungovanie. Príklad funkcie z triedy **skladnik.java**



**Agregácia –** tento princíp sme použili pri spôsobe ukladania kníh v **regáli** (**Regal.java**)a **palete (NoveKnihy.java). Knihy** nie sú súčasťou jedného regálu ale môžu sa premiestňovať po všetkých regáloch a navyše **sklad.java** pracuje s katalógom **kníh**, kde sú uložene referencie na knihy.



**Kompozícia –** je súčasťou kníhkupectva ako celku, pretože **kníhkupectvo (knihkupectvo.java)** by nemalo existovať bez toho aby v sebe nemalo **sklad (sklad.java)** a **predajňu (predajna.java)**. A zase **sklad** ani **predajna** by nemala existovať bez knihkupectva. Tento vztah je aj medzi **skladom** a **sekciami**; a ešte medzi **sekciami** a **regalmi.**



**Dedenie –** umožňuje, aby funkcionalita z jednej triedy bola súčasťou druhej. Tento vzťah sme uplatnili medzi **Miestnost.java** a**Sklad.java/Predajna.java**. A medzi **Regal.java** a **NoveKnihy.java,** ktoré obidve slúžia na ukladanie kníh určitou formou. **Viacnásobne dedenie** sme využili pri triedach používateľov, kde napr. **skladník** dedí atribúty a funkcie **zamestnanca** a ten zase dedí **používateľa**, ktorý je ešte dedení zákazníkom.





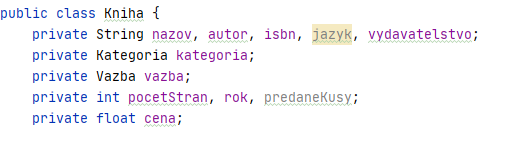
Pretože niektoré triedy nepotrebujú aby sa dali vytvoriť ich inštancie, napr. **Pouzivatel.java, Zamestnanec.java** a **Miestnost.java**, tak na tieto triedy sme použili **abstrakciu**. Z toho som využil **Polymorfizmus** typu **Upcasting,** kde mám premennú typu **Pouzivatel** používam v **Knihkupectvo.java** vo funkcií **main.** Táto premenná ukazuje na prihláseného používateľa a spúšťa funkciu na **spracovanie vstupu**.

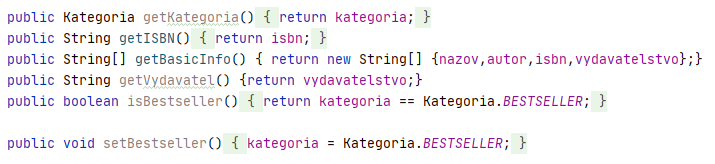
...



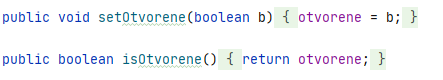


**Enkapsulácia –** všetky nestatické atribúty sú **private**, prípadne ak sa v triede používa atribút svojej rodičovskej triedy, tak v rodičovskej triede je tento atribút **protected.** Skoro všetky tieto atribúty majú nastavení **getter** ale iba niektoré majú nastavene **setter**.



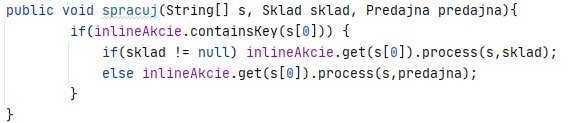


Trieda i Kniha.java

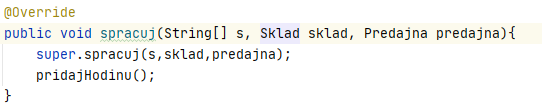


Trieda ii Predajna.java

**Overriding –** tento princíp nám dovoľuje z dedených tried zobrať funkcie a upraviť ich na nami žiadanú funkcionalitu. Využil som ho hlavne v dedeniach triedy **používateľ**, kde sa nachádza funkcia **spracuj**, ktorá spracováva vstupné príkazy. Ďalej som ho použil v triede **NoveKnihy.java**, ktorá dedí triedu **Regal.java** , tak že funkcia **odoberKnihy** namiesto uvoľňovania miesta kontroluje či ešte zostali knihy v **NoveKnihy.**

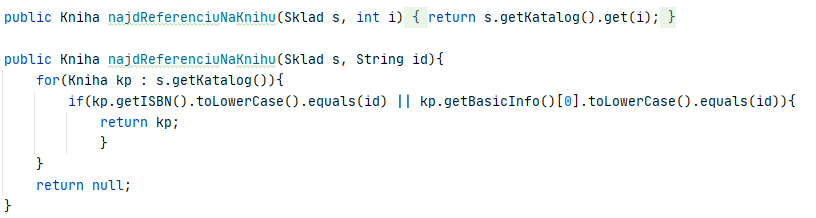


Trieda iii Pouzivatel.java



Trieda iv Zamestnanec

**Overloading –** príklad overloading-u mám v triede **skladnik.java**, kde skladník môže nájsť referencie na **knihu** pomocou reťazca znakov alebo podľa katalógového čísla. Využil som aj preťažovanie konštruktorov; hlavne v **Sekcia.java**





**Interface –**

**Lambda?**