**Dokumentácia k semestrálnej práci AUS**

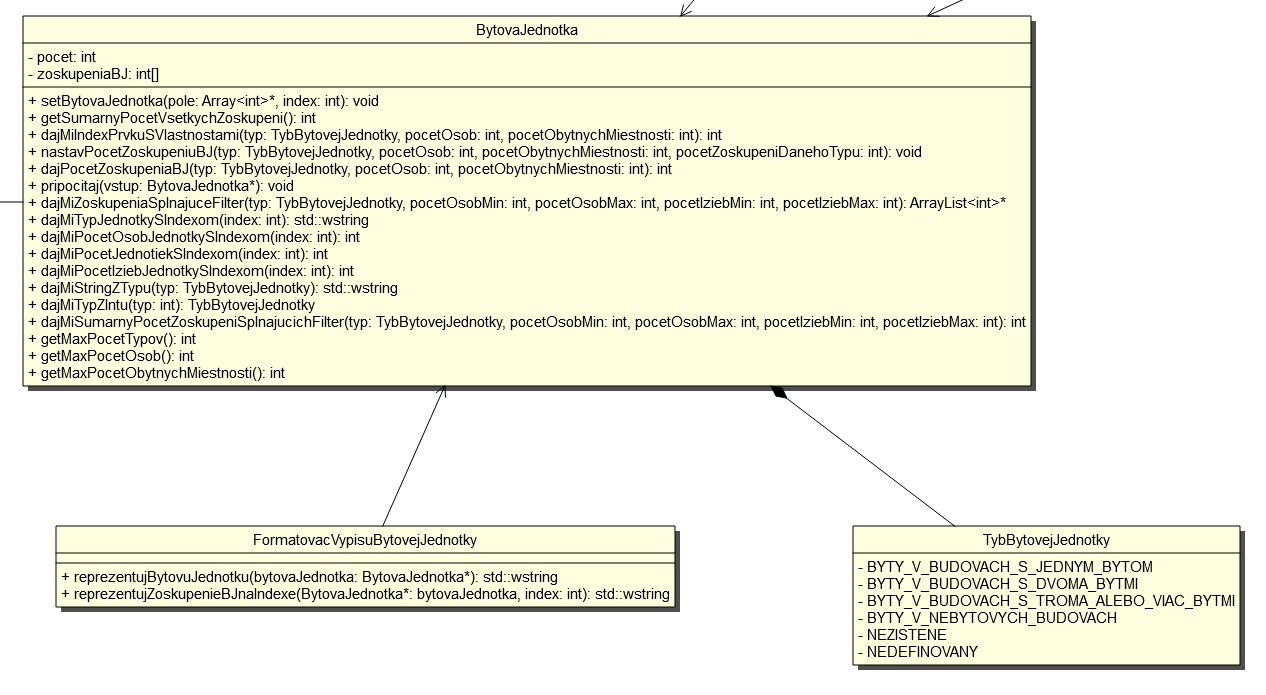
Vypracoval: Andrej Michalek

**UML diagram**

Uvedomujem si, že UML diagram je menej prehľadný, lebo je príliš veľký, a preto som pre lepšiu prehľadnosť priložil výseky z UML diagramu ku popisu, ktorý sa danej časti UML diagramu týka. Celý UML diagram v plnom rozlíšení sa zároveň nachádza ako obrázok v prílohe.



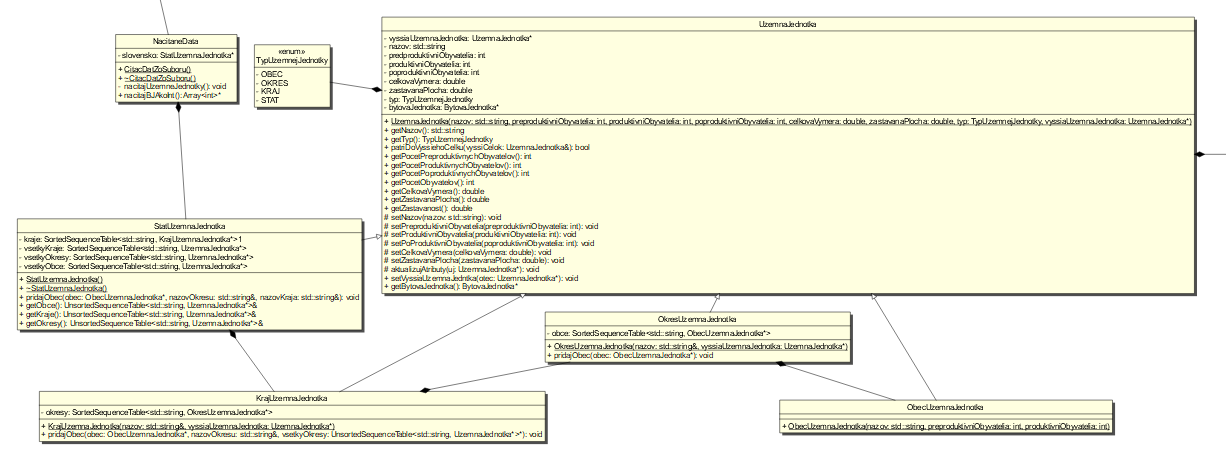
**Metóda načítania dát a použité údajové štruktúry**



Trieda bytová jednotka obsahuje všetky informácie o zoskupeniach bytových jednotiek v danej (jednej konkrétnej) územnej jednotke. Keďže zoskupenie bytovej jednotky je jednoznačne definované počtom osôb, počtom obytných miestností a typom bytovej jednotky, rozhodoval som sa medzi dvoma spôsobmi uloženia dát – medzi stromom, kde by jedna úroveň stromu predstavovala typ, druhá počet osôb a tretia počet obytných izieb a medzi trojrozmernou maticou. Pri strome by bola komplikovanejšia impelementácia, pamäťová náročnosť (réžijné smerníky) a vyhľadávanie by spomaľovali skoky medzi úrovňami (podľa zadania je potrebný extrémne rýchly prístup, keďže každá UJ obsahuje až 550 typov zoskupení bytových jednotiek). Trojrozmerná matica je pre tento konkrétny prípad použitia omnoho rýchlejšia, keďže vyššie úlohy vyžadujú filtrovanie územných jednotiek na základe intervalov v počte osôb a počte izieb. V trojrozmernej matici je trojica vlastností – typ bytovej jednotky, počet osôb a počet obytných izieb jednoznačne definovaný xovou, yovou a zkovou súradnicou a číselná hodnota na tomto indexe znamená, koľko obsahuje územná jednotka zoskupení bytových jednotiek daného typu. Filtrovanie podľa zadaných vlastností znamená jednoduché oseknutie intervalu x-ovej, y-ovej a/alebo z-ovej súradnice a je na tejto množine prakticky okamžité. Pri tvorbe druhej semestrálnej práce som si všimol, že skoky medzi objektami spomaľujú prístup, preto som sa rozhodol implementovať trojrozmernú maticu do jedného poľa s hashovacou funkciou a nie ako pole polí.

Bytovú jednotku je možné inicializovať viacerými spôsobmi. Ak ju nastavujem v obci, poznám konkrétne konečné hodnoty a na tento účel som si pripravil funkciu setBytovaJednotka, ktorá prevezme pole a index do neho a hodnoty počtov zoskupení bytových jednotiek nastaví ako 550 prvkov (POCET\_TYPOV \* POCET\_OSOB \* POCET\_OBYTNYCH\_MIESTNOSTI) od indexu, ktorý dostane v parametri. Skúšal som aj druhú alternatívu, a síce alokovanie nového Arrayu intov pre každú bytovú jednotku samostatne, ale alokovanie nových objektov zaberá čas a došlo k spomaleniu celkového načítania o jednu minútu (v Debug móde x86), preto som to nakoniec implementoval takýmto spôsobom.

Ak nastavujem počty bytových jednotiek kraju, okresu alebo Slovensku, potom celkový počet v dobe vytvárania objektu nepoznám, a preto som si pripravil aj metódu pripocitaj, ktorá prevezme smerník na druhú bytovú jednotku a prvky svojho poľa na jednotlivých indexoch pozvyšuje o hodnoty poľa na identických indexoch v druhej bytovej jednotke. Tým je možné docieliť, že vyššie územné jednotky majú počet zoskupenia bytových jednotiek daného typu rovný súčtu týchto zoskupení v nižších územných jednotkách.



Pre uchovanie dát o územných jednotkách som pripravil niekoľko tried, aby sa s dátami čo najlepšie pracovalo.

Trieda UzemnaJednotka je predkom pre triedy StatUzemnaJednotka, KrajUzemnaJednotka, OkresUzemnaJednotka a ObecUzemnaJednotka. Do tejto triedy som implementoval všetky atribúty a funkcionalitu, ktorú by mali mať tieto triedy spoločnú.

Atribúty preproduktivniObyvatelia, produktivniObyvatelia, poproduktivniObyvatelia, celkovaVymera a zastavanaPlocha obsahujú informácie o hodnotách, ktoré je pre jednotlivé obce možné načítať zo súboru „1 Obce SR.csv“ a pre vyššie územné jednotky – okresy, kraje a celé Slovensko, je ich možné jednoducho vypočítať súčtom hodnôt atribútov nižších územných jednotiek. Atribút TypUzemnejJednotky špecifikuje, či ide o kraj, obec, okres alebo štát a je typu enum TypUzemnejJednotky. Každá územná jednotka má zároveň smerník na svoju bytovú jednotku.

Atribút vyssiaUzemnaJednotka obsahuje smerník na vyššiu územnú jednotku v danej hierarchii, aby bolo možné splniť časť zadania, ktorá hovorí o tom, že sa vypíše K UJ Názov pre územnú jednotku a pre všetky vyššie územné jednotky, do ktorých územná jednotka patrí. V obci je teda tento smerník inicializovaný na okres, do ktorého patrí, v okrese na kraj, do ktorého daný okres patrí a v kraji je inicializovaný na objekt typu StatUzemnaJednotka predstavujúci celé Slovensko. Triedy KrajUzemnaJednotka, ObecUzemnaJednotka a StatUzemnaJednotka obsahujú tabuľky typu SortedSequenceTable (sekvenčné utriedené tabuľky) reprezentujúce množinu synov, pričom smerník vyssiaUzemnaJednotka je možné považovať za smerník na otca. Hoci tento vzťah nie je daný konkrétnou údajovou štruktúrou, dá sa povedať, že územné jednotky sú načítané v multi-cestnom strome (keďže každý okres má iný počet obcí, každý kraj iný počet okresov atď.).

Trieda obsahuje zároveň getery pre príslušné atribúty a pre vlastnosti, ktoré boli popísané v zadaní semestrálnej práce. Metódu patriDoVyssiehoCelku, ktorú následne využíva kritérium s obdobným názvom, som naprogramoval tak, že sa daná územná jednotka spýta, či sa rovná jej názov s názvom vyššieho celku, ktorý do nej prichádza ako parameter, ak sa rovnajú vráti true, ak nie, metóda skontroluje, či sa smerník na vyššiu územnú jednotku rovná nullptr, ak áno, vráti false (lebo to znamená, že vyššie v hierarchii už nie je ďalšia Územná jednotka), ak nie, tak zavolá rekurzívne tú istú metódu nad o stupeň vyšším objektom. Tým pádom dokáže táto metóda vrátiť výsledok v rýchlosti nanajvýš 4 (ak ide o územnú jednotku typu obec), pretože sa musí opýtať sama seba, potom svojho okresu, kraja a nakoniec Slovenska.

Dôležitou metódou v tejto triede je aj aktualizujAtribúty, ktorá preberá smerník na územnú jednotku. Pri pridávaní novej územnej jednotky do hierarchie je potrebné vo vyšších jednotkách pripočítať zastavanosť, celkovú výmeru a ďalšie hodnoty atribútov pridávanej jednotky tak, aby sedeli (napríklad celková výmera okresu je rovná súčtu výmer jeho obcí, pri pridávaní každej obce sa tak zvyšuje aktuálna hodnota atribútu celková výmera o hodnotu atribútu danej obce). V tejto metóde sa zároveň zavolá metóda pripočítaj nad bytovou jednotkou, aby sa aktualizovali aj počty bytových jednotiek podľa novo zaradenej územnej jednotky.

Trieda StatUzemnaJednotka je potomok triedy UzemnaJednotka. V atribútoch má okrem tabuľky sortedSequenceTable svojich krajov (kvôli stromovej štruktúre) aj tabuľky SortedSequenceTable kvôli rýchlemu bodovému vyhľadaniu pri filtrovaní podľa názvu, kde kľúč je wstring a dáta sú smerník na objekt typu UzemnaJednotka. Ide o atribúty vsetkyKraje, vsetkyObce a vsetkyOkresy.

Proces pridávania obce je sprostredkovaný metódou pridajObec, ktorá preberá v parametri smerník na vytvorenú obec typu ObecUzemnaJednotka a parametre názovOkresu a názovKraja ako wstring. Toto riešenie som zvolil preto, že objekt typu územná jednotka sa dá bez problémov vytvoriť z dát v súbore „1 Obce SR.csv“ a na tom istom riadku sa v súbore „2 Územné členenie SR.csv“ nachádza informácia o tom, do ktorého okresu a kraja daná obec patrí. Na načítanie akejkoľvek obce z vonku sa tak dá jednoducho zavolať táto metóda a Slovensko si pridá obec presne na to miesto, kam patrí.

V metóde sa najskôr pýtam, či utriedená tabuľka krajov obsahuje kraj a názov kraja z parametra dávam ako parameter. Vďaka voľbe utriedenej sekvenčnej tabuľky mi metóda tryFind využívajúca interpolačné vyhľadávanie vráti informáciu o tom, či je alebo nie je daný kraj už vytvorený a zaradený rýchlosťou (log N). Ak už daný kraj existuje, zavolá sa nad this (tj. objekt Slovenska) metóda aktualizujAtribúty (do parametra ide pridávaná obec), do utriedenej tabuľky všetky obce sa pridá obec a nad krajom vráteným metódou tryFind zavolám jeho metódu pridajObec. Tá preberá okrem iného aj referenciu na tabuľku všetky okresy, aby v nej zostávali aktuálne údaje v prípade, že bude vytvorený a pridaný nový okres.

V prípade, že daný kraj neexistuje, vytvorím ho a pridám ho do utriedenej tabuľky krajov reprezentujúcich synov v strome a aj do utriedenej tabuľky všetkých krajov. Následne zavolám opäť metódu pridajObec nad this s tými istými parametrami a keďže som príslušný kraj práve pridal, je isté, že po tomto volaní sa vykoná prvá vetva tejto metódy.

Trieda KrajUzemnaJednotka obsahuje utriedenú tabuľku svojich okresov. Metóda pridaj obec cez tryFind skontroluje, či sa v nej už nachádza okres s názvom, ktorý vstúpil ako parameter. Ak áno, volá sa metóda aktualizuj atribúty a nad krajom, do ktorého spadá pridávaná obec sa volá jeho metóda pridajObec. Ak daný okres neexistuje, vytvorí sa nový okres, smerník naň sa pridá do tabuľky všetkyOkresy v Slovensku aj do tabuľky okresov daného kraja a následne sa nad this zavolá táto istá metóda s rovnakými parametrami, ale tentokrát už sa vykoná v prvej vetve.

Trieda OkresUzemnaJednotka obsahuje sekvenčne utriedenú tabuľku obcí, ktoré do daného okresu patria. Metóda pridaj obec prevezme smerník na obec ako vstupný parameter, nad daným okresom sa zavolá metóda aktualizujAtributy, vyššia územná jednotka danej obce sa nastaví na this a do tabuľky obce sa pridá daná obec. Pri vytváraní nového okresu a kraja sa this posiela ako referencia na vyššiu územnú jednotku už priamo do konštruktora.

Tento spôsob ukladania údajov som zvolil preto, aby bolo nahrávanie čo najrýchlejšie a najefektívnejšie. Strom zabezpečí, že každá územná jednotka má informácie o svojej príslušnosti a utriedené sekvenčné tabuľky zaručia veľmi rýchle vyhľadanie podľa názvu a zaradzovanie územnej jednotky do svojich vyšších jednotiek.

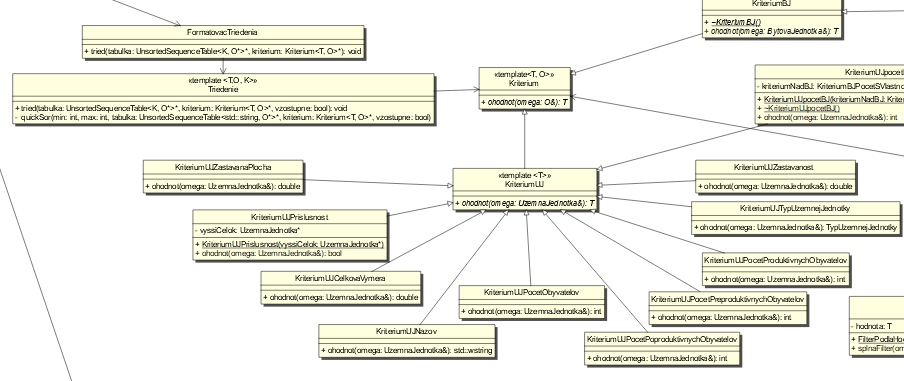
Proces načítania funguje tak, že sa najskôr do jedného poľa načítajú všetky informácie o bytových jednotkách zo súboru „4 Bytové jednotky SR.csv“ prostredníctvom metódy nacitajBJAkoInt v triede NacitaneData. Tento súbor som zoradil podľa názvu bytových jednotiek tak, aby poradie obcí v ňom zodpovedalo poradiu obcí v súboroch „1 Obce SR.csv“ a „2 Územné členenie SR.csv“. Toto extrémne zrýchli načítavanie, pretože nemusím nijako riešiť kategóriu daného zoskupenia bytových jednotiek. Poradie indexov zodpovedá poradiu indexov, s ktorými pracuje hashovacia funkcia triedy Bytová jednotka (na nultom indexe bude počet zoskupení bytových jednotiek typu Byty v budovách s jedným bytom, kde žije jedna osoba a majú jednu izbu atď.).

Po načítaní obce a vytvorení jej objektu sa do jej bytovej jednotky pošle smerník na pole všetkých BJ a index, od ktorého začínajú zoskupenia BJ danej obce. Index sa s každou obcou zvyšuje o POCET\_TYPOV \* POCET\_OSOB \* POCET\_OBYTNYCH\_MIESTNOSTI, vďaka čomu si každá bytová jednotka obce prečíta z poľa svoje údaje.

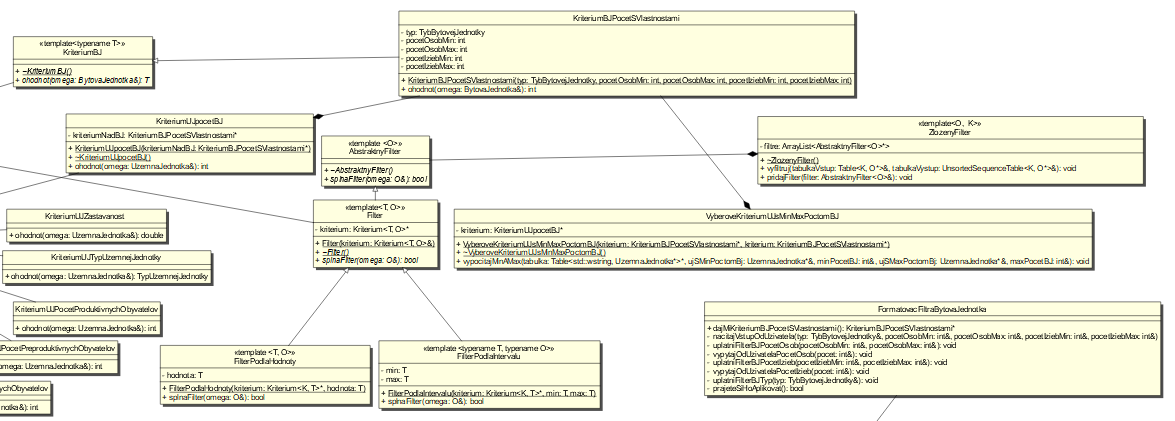
Celé načítavanie tak prebehne so zložitosťou pocetObci\*(log(pocetKrajovPriDanomVolani) + log(pocetOkresovVDanomKrajiPriDanomVolani) + log(pocetObciVDanomOkrese) + log(pocetVsetkychOkresovPriDanomVolani) + log(pocetVsetkychObciPriDanomVolani)). U niektorých obcí je ale potrebné vytvoriť nový kraj a okres a u nich sa táto zložitosť ešte zväčší o pridanie prvku do utriedenej tabuľky, čo je zložitosť log(pocetPrvkovVTabulke) – ak by sa pridával okres aj kraj, tak treba započítať túto zložitosť dvakrát. Načítanie bytových jednotiek má zložitosť načítania (POCET\_TYPOV \* POCET\_OSOB \* POCET\_OBYTNYCH\_MIESTNOST – pre zjednodušenie budem uvádzať ako 11\*10\*5 = 550)\*počet\_obcí. Pri každej obci sa ale kopíruje 550 číselných hodnôt do nej plus 550 sa pripočítava do kraja, do ktorého patrí, 550 do okresu a 550 do štátu.

Trieda NacitaneData má ako atribút objekt typu StatUzemnaJednotka reprezentujúci Slovensko. V konštruktore ho vytvorí a následne je zavolaná privátna metóda nacitajUzemneJednotky. Tá na začiatku vytvorí dva objekty typu fstream – citacObce1 (pre súbor "1 Obce SR.csv") a citacObce2 (pre súbor "2 Územné členenie SR.csv"), cez getline oba objekty prečítajú prvé riadky (aby sa pri načítaní odignorovali popisy stĺpcov v csv súboroch) a následne sa cez while cyklus volá getline. Riadky sú pomocou objektu stringstream rozkúskované v miestach, kde sú bodkočiarky, pretypované tak, aby splnili dátové typy, ktoré berie konštruktor obce. Následne sa vytvorí nový objekt typu obec, do jeho bytovej jednotky sa pošle pole s počtom zoskupení BJ spolu s príslušným indexom. Objekt obce sa spolu s názvom okresu a názvom kraja, ktorý sa získal obdobným spôsobom, len z druhého súboru posielajú do metódy pridajObec nad objektom Slovensko.

**Kritériá a filtre**



Aby bolo možné využívať univerzálny triediaci a filtrovací algoritmus, vytvoril som triedu kritérium ako všeobecného predka s metódou ohodnoť, následne jej potomkov KriteriumUJ a KriteriumBJ, ktorí za jeden z templatových parametrov dosádza územnú jednotku (resp. Bytovú Jednotku) a následne aj KriteriumUJNazov, KriteriumUJCelkovaVymera atď., ktoré dosádzajú za templatový parameter konkrétny typ, ktorý daná vlastnosť reprezentuje. Dané kritérium teda zavolá príslušený getter územnej jednotky a vráti jeho hodnotu.



Filter obsahuje atribút Kritérium a obsahuje virtuálnu metódu splnaFilter, ktorú prepisujú jeho potomkovia FilterPodlaHodnoty a FilterPodlaIntervalu. Filter podľa hodnoty obsahuje atribút hodnota, ktorého hodnota prechádza ako parameter konštruktora. Metóda spĺňa filter sa pozrie, či hodnota, ktorú vracia kritérium nad objektom O je rovnaká ako hodnota v atribúte, ak je rovnaká, vráti true, ak nie, vráti false. FilterPodľaIntervaluje funguje podobným spôsobom, ale namiesto jednej hodnoty berie interval (T min a T max) a zisťuje sa, či návratová hodnota vrátená daným kritériom leží medzi minimom a maximom.

Aby bolo možné pri prechádzaní jednej tabuľky efektívne použiť viacero filtrov, vytvoril som ešte jedného predka filtra – táto trieda sa volá Abstraktný Filter a má len jeden templatový parameter O a virtuálnu metódu spĺňa filter.

V triede ZloženýFilter sa vďaka tomuto riešeniu dá vytvoriť ArrayList filtrov, ktoré môžu obsahovať Filtre s rôznymi kritériami (kritériá môžu mať rôzne návratové hodnoty, je len dôležité, aby všetky filtre pracovali s rovnakými vstupnými objektmi, teda v tomto prípade s bytovými jednotkami). Metóda pridajFilter umožňuje pridať filter do ArrayListu. Metóda vyfiltruj preberá tabuľku tabulkaVstup a prázdnu neutriedenú sekvenčnú tabuľku tabulkaVystup. Prvky vstupnej tabuľky sú prechádzané vo foreach cykle a vo vnútornom cykle sú prechádzané cez for each aj filtre v ArrayListe. Vo vonkajšom cykle sa nastavuje hodnota booleanového atribútu splna na true, ak ktorýkoľvek z filtrov nad daným objektom vráti false, aj hodnota splna sa nastaví na false. Po vykonaní vnútorného cyklu sa cez if odkontroluje, či je splna == true a ak áno, prvok sa pridá do výstupnej tabuľky. Tým je možné dosiahnuť, že pri filtrovaní tabuľky je celková zložitosť početPrvkovTabuľky\*pocetFiltrov\*rychlostHodnoteniaFiltra, pričom rýchlosť hodnotenia daného filtra je možné považovať za konštantnú. Duplicita údajov je v tomto bode z logického hľadiska vylúčená, takže pridáva do neutriedenej tabuľky bez kontroly kľúča – jednotkovo.

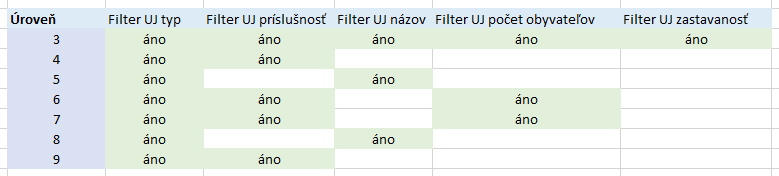
Trieda Triedenie preberá neutriedenú sekvenčnú tabuľku, kritérium a informáciu o tom, či má danú tabuľku utriediť vzostupne alebo zostupne (boolean). Na triedenie využívam algoritmus QuickSort, ktorý má teoretickú zložitosť až do O(n2), avšak v praxi sa ukazuje, že jeho rýchlosť je bližšie (n\*log n). Testoval som to na našich dátach – utriedenie tabuľky všetkých obcí Slovenska trvalo bubble sortu s n^2 zložitosťou niekoľko minút, zatiaľ čo QuickSort to zvládol takmer okamžite, preto si myslím, že je pre toto použitie vhodný.

**Užívateľské rozhranie**

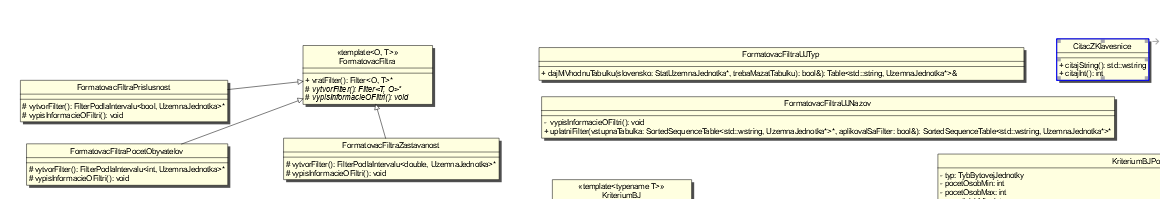
Pri návrhu konzolového užívateľského rozhrania som sa snažil držať oddelenú aplikačnú vrstvu od vrstvy samotného UI. Na to som použil niekoľko obaľovacích tried, ktoré spolupracujú s triedami popísanými vyššie, ale z konzoly pýtajú od užívateľa príslušné atribúty a následne ich plnia tak ako treba.

Trieda UzivatelskeProstredie v konštruktore inicializuje atribút dáta typu NacitaneData a ešte predtým užívateľa informuje o tom, že prebieha načítavanie dát zo súboru (samotné načítavanie aj s Bytovými jednotkami prebehne v Release móde do niekoľkých sekúnd, ale v debug móde x86 môže trvať až 2 minúty). Metóda menu sa volá priamo z main, obsahuje do while cyklus, kde si užívateľ môže vybrať jednu z možností, pričom každá sa snaží reflektovať jeden bod zo zadania. Pri výbere jednej z možností za zavolá metóda prislúchajúca k danej možnosti. Ak užívateľ vyberie 0, program sa ukončí.

Všimol som si, že každá úroveň (resp. bod) v semestrálnej práci pracuje s množinou územných jednotiek, ktoré sú vyfiltrované špecifickou kombináciou filtrov. Pre jednotlivé úrovne sú to nasledujúce filtre:



Aby som zjednodušil kód a minimalizoval duplicitu, rozhodol som sa vytvoriť jednu funkciu, ktorá bude spoločná pre každú úroveň a bude mať za úlohu poskytnúť vyfiltrovanú tabuľku územných jednotiek podľa filtrov, ktoré si zvolil užívateľ. Pri každom filtri sa môže rozhodnúť, či daný filter chce alebo nechce, preto by aj takáto implementácia mala spĺňať požiadavky semestrálnej práce (ak daný filter v úrovni nebol vyžadovaný, užívateľ si ho môže nevybrať).



Táto metóda v triede Uzivatelske prostredie sa volá filtrujUzemneJednotky. Na začiatku vytvorí objekt typu FormatovacFiltraUJTyp, vytvorí lokálnu premennú smerník na utriedenú tabuľku podľa názvu – tabulkaVstupna a do tejto premennej priradí to, čo vráti metóda dajMiVhodnuTabulku triedy FormatovacFiltraUJTyp. Metóda sa od užívateľa opýta, aký typ UJ potrebuje a podľa výberu zavolá nad Slovenskom getter na utriedenú tabuľku okresov, krajov atď. Ak užívateľ nechce tento filter aplikovať, vytvorí sa utriedená sekvenčná tabuľka, do ktorej sa pridajú všetky kraje, okresy, obce aj samotné Slovensko.

Následne sa vytvorí formátovač filtra UJ názov, ktorého metóda uplatniFilter preberá utriedenú tabuľku a vracia utriedenú tabuľku. Táto metóda sa užívateľa spýta, či chce aplikovať tento filter. Ak chce, tak si od neho vypýta názov územnej jednotky a vytvorí prázdnu utriedenú tabuľku. Z vstupnej utriedenej tabuľky sa prostredníctvom metódy tryFind pokúsi získať územnú jednotku so zvoleným názvom. Ak sa jej to nepodarí, vráti prázdnu tabuľku, ak sa jej to podarí, pridá do novovytvorenej tabuľky smerník na územnú jednotku spĺňajúci filter názov a vráti ju. Ak užívateľ nechce aplikovať filter názov, nevytvára novú tabuľku, ale vráti smerník na vstupnú tabuľku.

Tieto dva filtre som sa rozhodol implementovať inak ako ostatné (menej univerzálne) kvôli rýchlosti. Pri type nie je nutné prehľadávať všetky územné jednotky, jednoduchšie je vrátiť konkrétnu tabuľku a pri názve sa dá vďaka utriedeniu podľa názvu a interpolačnému vyhľadávaniu v utriedenej sekvenčnej tabuľke dosiahnuť zložitosť log(n) oproti zložitosti N, ak by sa prehľadávali všetky prvky.

Filter typ aj názov dávajú cez booleanovský parameter informáciu o tom, či vytvárali novú tabuľku a ak áno, táto tabuľka je na konci metódy odstránená.

Následne sa v metóda filtrujUzemneJednotky vytvorí objekt zloženého filtra. Formátovače filtra počet obyvateľov, zastavanosť a príslušnosť si od užívateľa vypýtajú informáciu o tom, či chce dané filtre aplikovať. Ak chce, vrátia vytvorený filter a ak nechce, vrátia null. Ak nevrátia null, filter sa pridá do zloženého filtra a nad ním sa na konci zavolá metóda vyfiltruj. Tá preberá tabuľku, ktorú vrátil filter názov a referenciu na prázdnu výstupnú tabuľku. Zložený filter sa každého prvku spýta, či spĺňa zadané kritériá a ak áno, uloží smerník naň do výstupnej tabuľky. Na konci metóda vráti výstupnú tabuľku.

Celková zložitosť filtrovania pre územné jednotky v celej tejto semestrálnej práci záleží od toho, akú kombináciu filtrov si užívateľ vybral a je nasledovná:

1) Ak si vybral typ, názov a 0 až všetky filtre v zloženom filtri

1 (získanie smerníka tabuľky zo Slovenska) + log(pocetUJvTabulke) + 1(filtrovanie)

2) Ak si vybral typ, nevybral si názov a vybral si 0 až všetky filtre v zloženom filtri

1 (získanie smerníka tabuľky zo Slovenska) + 1 (vrátenie tejto tabuľky filtrom názov) + pocetUJvTabulke (filtrovanie)

3)Ak si nevybral typ, vybral si názov a 0 až všetky filtre v zloženom filtri

pocetVsetkychUJ \* log(aktualnyPocetPrvkovVTabulke) + log(pocetVsetkychUJ) + 1

4) Ak si nevybral typ, nevybral si názov a 0 až všetky filtre v zloženom filtri

pocetVsetkychUJ \* log(aktualnyPocetPrvkovVTabulke) + 1 + pocetVsetkychUJ

Na tieto zložitosti budem ďalej v dokumentácií odkazovať ako na zložitosť metódy filtrujUzemneJednotky (táto zložitosť sa pripočíta ku zložitosti každej úrovne).

**3) Vypísanie informácií o územných jednotkách**

Metóda informacieOUzemnychJednotkach reflektuje požiadavky 3. úrovne semestrálnej práce. Od metódy filtrujUzemneJednotky získa neutriedenú sekvenčnú tabuľku územných jednotiek. Následne cez foreach cyklus prechádza túto tabuľku a volá metódu getVypisUzemnejJednotky, kde ako parameter posiela územnú jednotku.

Túto metódu som vytvoril, pretože vo viacerých úrovniach je potrebné vypisovať tie isté informácie o územných jednotkách. Metóda vytvorí reťazec obsahujúci a pre výstup správne naformátovaný – typ UJ, názov UJ, názov a typ vyšších jednotiek, kam daná územná jednotka patrí, počet obyvateľov, počet preproduktívnych obyvateľov, počet produktívnych obyvateľov, počet poproduktívnych obyvateľov, zastavanosť, celkovú výmeru a zastavanú plochu.

Zložitosť úrovne vypísanie informácií o územných jednotkách je teda zložitosť metódy filtrujUzemneJednotky + vratenyPocetUJ \* vypisanie N údajov o každej z nich.

**4) Zoradenie územných jednotiek spĺňajúcich daný filter**

Zoradenie územných jednotiek zabezpečuje metóda zoradenieUzemnychJednotiekSFiltrom. Tá volá metódu filtrujUzemneJednotky a následne sa opýta užívateľa, podľa čoho chce triediť. Podľa jeho voľby sa vytvorí príslušné kritérium a formátovač triedenia. Jeho metóda trieď si od užívateľa vypýta, či chce triediť vzostupne alebo zostupne, vytvorí objekt triedenia a pošle mu neutriedenú tabuľku a booleanovský atribút vzostupne = true/false podľa toho, čo zadal užívateľ. Triedenie quicksortom utriedi tabuľku a tá sa následne späť v užívateľskom rozhraní vypisuje cez foreach cyklus, pričom vždy sa vypíše názov územnej jednotky a hodnota kritéria, podľa ktorého sa triedilo.

Celková zložitosť tejto operácie je teda zložitosť metódy filtrujUzemneJednotky + zložitosť quick sortu (početUJvTabulke \* log(početUJvTabulke)) + početUJvTabulke \* vypísanie názvu a kritéria.

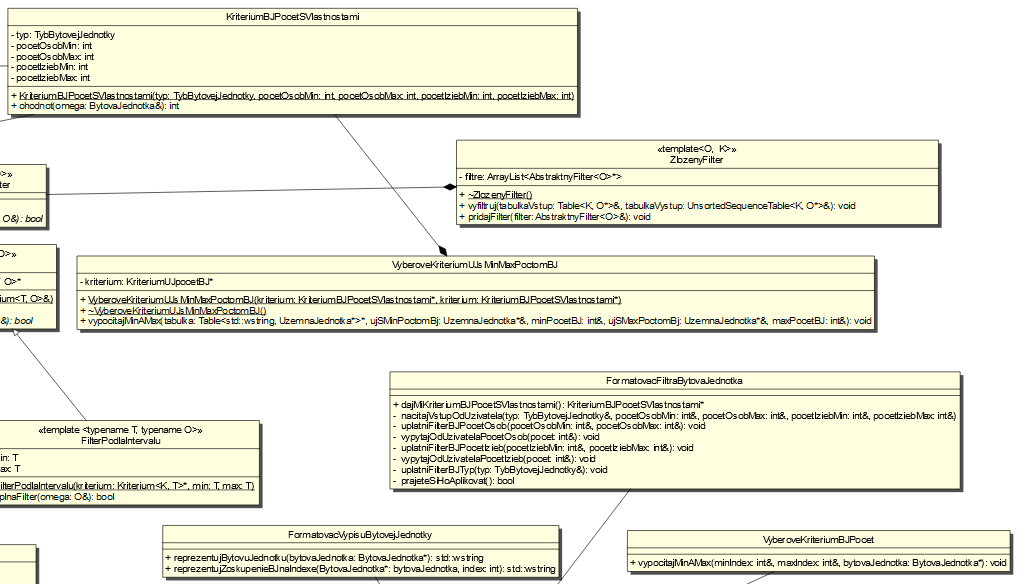
**5) Vypísanie informácií o zoskupeniach bytových jednotiek vo zvolenej územnej jednotke**

Túto funkcionalitu zabezpečuje metóda informacieOBytovychJednotkach. Na začiatku si vypýta tabuľku od metódy filtrujUzemneJednotky. Následne sa vytvorí objekt FormatovacVypisuBytovejJednotky. Výsledné územné jednotky sa prechádzajú cez foreach cyklus a nad každou je volaná metóda getVypisUzemnaJednotka. Nad jej bytovou jednotkou je volaná metóda reprezentujBytovúJednotku formátovača.

Tá preberá smerník na bytovú jednotku, v cykle cez všetky typy, všetky počty osôb a všetky počty obytných miestností prechádza všetkých 550 zoskupení bytových jednotiek a pre každé vypisuje kritériá TypBJ, počet osôb, počet izieb a BJpočet, pričom formátovanie výstupu je podobné ako v CSV súbore zoskupení BJ, aby bolo možné výstupy jednoducho odkontrolovať. Objekt bytová jednotka obsahuje getery, ktoré umožňujú tejto triede získať počet zoskupení daného typu v danej bytovej jednotke na základe typu, počtu osôb a počtu izieb. Geter využíva hashovaciu funkciu, takže má v podstate jednotkový prístup k údajom.

Celková zložitosť tejto úlohy je teda zložitosť metódy filtrujUzemneJednotky + početUJvTabulke \* (vypísanie údajov o UJ + vypísanie 550 údajov o zoskupení bytových jednotiek v danej UJ).

**6) Vypísanie sumárnych informácií o počtoch bytových jednotiek vo zvolených územných jednotkách**



Funkcionalitu 6. úrovne reflektuje metóda informacieSumarnychInformaciiOPoctochBJ. Na začiatku si vypýta tabuľku od metódy filtrujUzemneJednotky.

Potom vytvorí objekt typu FormatovacFiltraBytovaJednotka, ktorý má za úlohu vypýtať si od užívateľa dáta potrebné na vytvorenie kritéria KriteriumBJPocetSVlastnosťami. Užívateľ si môže vybrať, či chce Zoskupenia bytových jednotiek filtrovať podľa počtu osôb, počtu izieb a typu (môže si to vybrať pre každý filter samostatne). Ak chce, potom si od neho trieda vypýta príslušné údaje – typ alebo hranice intervalov a vytvorí KriteriumBJPocetSVlastnosťami.

Toto kritérium preberá bytovú jednotku a vracia počet zoskupení bytových jednotiek v nej, ktoré spĺňajú definované vlastnosti – typ, interval počtu osôb a interval počtu izieb, pričom každá z týchto troch vlastností môže a nemusí byť zadaná.

Uvedomujem si, že toto riešenie trpí na univerzálnosť, ale dosiahol som výrazné zvýšenie rýchlosti. Vďaka usporiadaniu zoskupení BJ v BytovejJednotke v trojrozmernej matici je možné priamo v triede bytová jednotka zostrojiť metódu, ktorá preberá príslušné vlastnosti a vracia počet zoskupení BJ, ktoré tieto vlastnosti spĺňajú.

Trieda obsahuje metódu dajMiSumarnyPocetZoskupeniSplnajucichFilter, ktorá preberá typ, minimálny počet osôb, maximálny počet osôb, min počet izieb a max počet izieb a na základe nich vie nastaviť interval prehľadávania matice na osi x, y a z. Ak nejaká hodnota nie je zadaná, intervaly sú nastavené na min a max možnú hranicu. Následne sa cez vnorené foreach cykly (jeden ide cez typy – ak bol zadaný, tak len cez 1, druhý ide cez obytné miestnosti a tretí cez izby) vypočíta celkový počet zoskupení BJ a vráti sa z metódy. KriteriumBJPocetSVlastnosťami volá v metóde ohodnoť túto metódu nad Bytovou jednotkou, ktorá vstupuje ako parameter.

V metóde užívateľského prostredia sa následne cez foreach cyklus prechádzajú územné jednotky, tie sa vypisujú a ku nim sa vypisuje počet zoskupení BJ spĺňajúcich filter, a to volaním metódy ohodnoť na vytvorenom kritériu – KriteriumBJPocetSVlastnosťami.

Celková zložitosť tejto úlohy je teda zložitosť metódy filtrujUzemneJednotky + početUJvTabulke \* (vypísanie údajov o UJ + zrátanie počtu zoskupení BJ spĺňajúcich filter (tj. rozsah intervalu počet izieb\* rozsah intervalu typ \* rozsah intervalu počet osôb) + vypísanie počtu ).

**7. Zoradenie územných jednotiek podľa sumárneho počtu zvolených bytových jednotiek**

Funkcionalitu 7. úrovne reflektuje metóda zoradenieUJPodlaSumarnehoPoctuZvolenychBJ. Na začiatku si vypýta tabuľku od metódy filtrujUzemneJednotky.

Proces funguje veľmi podobne ako v 6. úrovni. Vytvorí formátovač filtra bytová jednotka a ten vráti KriteriumBJPocetSVlastnostami. Následne sa vytvorí formátovač triedenia a KriteriumUJPocetBJ, ktoré spolupracuje s kritériom KriteriumBJPocetSVlastnostami, ktoré preberá ako parameter. Rozdiel medzi nimi je len v tom, že jedno pracuje s územnou jednotkou a druhé s bytovou jednotkou.

Nad triedením sa zavolá metóda trieď, ktorá pošle tabuľku vrátenú metódou filtrujUzemneJednotky a KriteriumUJPocetBJ. Prebehne quicksort, ktorý územné jednotky zoradí podľa sumárneho počtu zvolených zoskupení BJ spĺňajúcich dané filtre.

Následne sa cez foreach vypíše každá jednotka spolu s počtom vyhovujúcich BJ.

Celková zložitosť tejto úlohy je teda zložitosť metódy filtrujUzemneJednotky + triedenie(početUJvTabulke \* log(početUJvTabulke)) + početUJvTabulke \* (vypísanie údajov o UJ + zrátanie počtu zoskupení BJ spĺňajúcich filter (tj. rozsah intervalu počet izieb\* rozsah intervalu typ \* rozsah intervalu počet osôb) + vypísanie počtu ).

**8. Vypísanie výberových kritérií pre zoskupenia bytových jednotiek vo zvolených územných jednotkách**

Funkcionalitu 8. úrovne reflektuje metóda vyberoveKriteriaPreBJ. Na začiatku si vypýta tabuľku od metódy filtrujUzemneJednotky.

Na začiatku sa vytvorí objekt Výberové kritérium BJ počet. Ten obsahuje metódu vypocitajMinAMax, pričom vráti index typu zoskupenia BJ v bytovej jednotke vstupujúcej v parametri s najväčším a najmenším počtom. Funguje tak, že na začiatku inicializuje atribúty min = IN\_MAX a max = 0 (reprezentujú počet zoskupení BJ daného typu) a indexMinima/indexMaxima – index daného zoskupenia v poli bytovej jednotky a následne prechádzajúc cez for cyklus všetkých 550 zoskupení. Ak sa nájde zoskupenie, ktoré má menší počet ako minimum, prestavia sa premenné min a indexMinima, ak sa nájde zoskupenie, ktoré má väčší počet ako max, prestavia sa príslušné hodnoty. Takto dokáže pre každú Bytovú Jednotku nájsť zoskupenie bytovej jednotky v nej, ktorého je v nej najmenší alebo najväčší počet.

V užívateľskom prostredí sa následne všetky územné jednotky prechádzajú cez foreach cyklus a pre každú sa vypočíta zoskupenie BJ s najväčším a najmenším počtom. Tie sa vypisujú cez metódu reprezentujZoskupenieBJNaIndexe objektu typu FormatovacVypisuBytovejJednotky. Táto metóda preberá index zoskupenia BJ a vypisuje jeho vlastnosti – typ bytovej jednotky, počet osôb, počet izieb a počet. Okrem toho sa vypisuje aj názov územnej jednotky.

Celková zložitosť tejto úlohy je teda zložitosť metódy filtrujUzemneJednotky + početUJvTabulke \* (vypísanie názvu UJ + nájdenie min a max zoskupenia BJ (tj. 550) + vypísanie týchto dvoch zoskupení (metóda na výpis volá na získanie vlastností hashovaciu funkciu bytovej jednotky, ktorá sa dá označiť za jednotkovú)).

**9. Vypísanie výberových kritérií pre územné jednotky na základe zvolených zoskupení bytových jednotiek**

Funkcionalitu 9. úrovne reflektuje metóda vyberoveKriteriaPreUJ. Na začiatku si vypýta tabuľku od metódy filtrujUzemneJednotky.

Táto metóda spolupracuje s objektom triedy VyberoveKriteriumUJsMinMaxPoctomBJ, ktorá vyžaduje kritérium typu KriteriumBJPocetSVlastnostami. To si na začiatku vypýtam od príslušného formátovača. Metóda vypočítaj min a max objektu VyberoveKriteriumUJsMinMaxPoctomBJ preberá tabuľku územných jednotiek a cez referencie v parametroch vracia územnú jednotku s maximálnym počtom daných zoskupení bytových jednotiek a územnú jednotku s minimálnym počtom daných zoskupení bytových jednotiek.

Funguje na podobnom princípe ako predchádzajúce výberové kritérium, ale namiesto prechádzania bytových jednotiek prechádza územné jednotky a hľadá tú, ktorá má najväčší / najmenší počet zoskupení BJ spĺňajúcich filter typ, počet izieb a príslušnosť.

Ak metóda nevráti null (vrátila by, keby bola tabuľka UJ prázdna), tak sa vypíše názov územnej jednotky s najväčším sumárnym počtom BJ a počet BJ plus názov územnej jednotky s najmenším sumárnym počtom BJ a príslušný počet BJ.

Celková zložitosť tejto úlohy je zložitosť metódy filtrujUzemneJednotky + početUJvTabulke \* (zistenie počtu BJ spĺňajúcich filter (tj. interval typ \* interval počet osôb \* interval počet izieb) + vypísanie dvoch UJ spolu s počtom.