Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Algoritmy
a údajové štruktúry
2

1.Semestrálna práca

Zimný
semester

Jozef Staško

Št. skupina
5ZIB11

Obsah

1. Operácie a ich výpočtová zložitosť 3
1.1 Operácia vyhľadania parcely/ nehnuteľnosti podľa zadanej GPS 3
1.2 Operácia vyhľadanie parcely/ nehnuteľnosti podľa zadaného intervalu GPS 3
1.2.1 Popis implementácie viacrozmerného intervalového vyhľadávania 3
1.3 Operácia pridávania parcely/ nehnuteľnosti podľa zadaných dát a GPS súradnice 5
1.4 Operácia vymazania parcely/ nehnuteľnosti podľa zadaných dát a GPS súradnice 5
1.5 Operácia editácie parcely/ nehnuteľnosti 6
2. Použité štruktúry v systéme 7
3. UML Diagram tried 8
3.1 Presenter/Kontroler 8
3.2 Service 8
3.3 Model 9
3.4 View 9
3.5 Celý systém MVP 10
4. Záver 11

1. Operácie a ich výpočtová zložitosť

1.1 Operácia vyhľadania parcely/ nehnuteľnosti podľa zadanej GPS

Nech premenná N je počet nehnuteľností v aplikácii a nech premenná M je počet parciel v danej aplikácií.

Obidva tieto prvky sú uložené v štruktúre K-D strome a teda amortizovaná zložitosť hľadania v tejto štruktúre ma zložitosť log N pre nehnuteľnosti a log M pre parcely.

Ak by sa stalo najhoršie a pri vkladaní nám niektorá zo štruktúr degenerovala na lineárny zoznam, potom by zložitosť pre vyhľadanie nehnuteľnosti v najhoršom možnom prípade bola N a pre parcely M.

1.2 Operácia vyhľadanie parcely/ nehnuteľnosti podľa zadaného intervalu GPS

Nech premenná N je počet nehnuteľností v aplikácii a nech premenná M je počet parciel v danej aplikácií.

Obidva tieto prvky sú uložené v štruktúre K-D strome. Amortizovaná zložitosť operácie viacrozmerné intervalové vyhľadávanie na tomto strome má zložitosť S + log N pre nehnuteľnosti respektíve pre parcely S + log M. Kde S je počet vrcholov nájdených algoritmom inOrder.

1.2.1 Popis implementácie viacrozmerného intervalového vyhľadávania

Implementovanie viacrozmerného vyhľadávania začína v servise ktorá si vypýta od kontroléra dva intervaly. Týchto intervalov je dva, pretože v tejto aplikácii máme K-D strom. Tento strom má kľúč GPS. GPS zas využíva dve súradnice na popis nehnuteľnosti alebo parcely. To znamená, že v našom strome sú vrcholy, ktoré majú dvojrozmerné kľúče a preto teda potrebujeme na intervalové vyhľadávanie práve dva intervaly.

Následne sa tieto dva intervaly uložia do listu ktorý sa pošle do funkcie stromu find. V tejto funkcii si z počiatku overím, či počet intervalov sa rovná rozmeru K-D stromu. Následne skontrolujem či náhodou strom má koreň.

Po tomto overovaní pristúpim na hľadanie prvkov v tomto intervale a to upraveným algoritmom inOrder. Do funkcie inOrder vstupujú parametre:

Vrchol V v ktorom hľadanie začínam

List možných nájdených vrcholov ktoré môžu patriť do obidvoch intervalov

List intervalov, kde i-ty interval je pre i-ty kľúč

Funkciu začíname vytvorením Stack-u z knižnice java.util a ktorý bude slúžiť pri prehliadke inOrder. Zásobník je typ LIFO a teda má veľmi praktické využitie pri tejto prehliadke, keďže najprv potrebujeme prejsť všetkých možných ľavých synov a potom následne daný vrchol spracovať a potom prejsť pravého syna. Algoritmus sa skončí dvoma podmienkami. Buď zásobník je prázdny alebo vrchol V ukazuje na null.

V cykle teda prechádzame stromom pod podmienkou, ak nie je v danom vrchole ukazovateľ na null tak vložím následne vrchol do zásobníka, vypočítame index podľa ktorého sa budeme rozhodovať v danom vrchole na danej úrovni. Urobíme to podľa vzorca. Nech h je úroveň daného vrcholu a teda index vrcholu vypočítame ako h modul k. Kde k je počet sekundárnych kľúčov jedného vrcholu.

Ak máme tento index vieme potom určiť či sa daný kľúč nachádza v danom intervale alebo či je kľúč väčší ako daný interval. Pristúpime naň v liste intervalov s rovnakým indexom. Ak podmienka je splnená tak priradíme ľavého syna daného vrcholu V vrcholu V a teda V = V.lavySyn. Ak podmienka nevyšla znamená to že vľavo už ísť nemôžem, keďže viem, že nemám šancu nájsť vľavo žiadny vrchol, ktorý by mohol byť v danom intervale. Pretože môj vrchol už je menši ako aktuálny interval a teda priradím vrcholu V pravého syna daného vrcholu a teda V = V.pravySyn. Keďže mám šancu nájsť prvky vpravo, pretože sú tam väčšie prvky.

Ak však vo vrchole V máme ukazovateľ null. Postupujeme inak. Vyberieme zo zásobníka a pridáme do listu možných kandidátov, ktorý môžu patriť do obidvoch intervalov .Následne si zistíme jeho index , a ak je v intervale a zároveň nie je kľúč hornou hranicou daného intervalu priradíme do vrcholu V pravého syna daného vrcholu V a teda V = V.pravySyn. Ak sa podmienka nesplnila priradíme tak null. Robíme to takto preto, pretože vieme, že

predtým sme priradili do vrcholu V ľavého syna, ak bol v intervale, a teda ak teraz bude v intervale tak viem že mám šancu na zisk ďalšieho vrcholu ktorý potencionálne bude patriť do výsledku, keďže môžem nájsť väčší vrchol ale stále viem, že je šanca, že bude v intervale, pretože sme si to ohraničili možnosťou, že nemôže byť vrchol toho intervalu.

Po takto upravenej prehliadke inOrder mam v liste vrcholy, ktoré potencionálne patria do obidvoch intervalov. Následne takýto list, najlepšie bude ak to bude zreťazený zoznam, prechádzame a kontrolujeme, či sa kľúče vybraných prvkov vyskytujú v intervale. Ak áno tak si ich pridám najlepšie do ďalšieho listu, ktorý po ukončení prehliadky vrátim ako návratovú hodnotu funkcie a teda mám všetky nehnuteľnosti alebo parcely ktoré patria do daných GPS intervalov.

1.3 Operácia pridávania parcely/ nehnuteľnosti podľa zadaných dát a GPS súradnice

Nech premenná N je počet nehnuteľností v aplikácii a nech premenná M je počet parciel v danej aplikácií.

Pre zjednodušenie budem uvádzať len prípad pre pridanie nehnuteľnosti ale prípad pre pridanie parcely je analogicky.

Do funkcie v servise nám prídu teda dáta a GPS súradnica, na ktorej sa nachádza daný objekt. Ako prvé si musím vyhľadať s amortizovanou zložitosťou log M všetky parcely, s takými istými GPS súradnicami, ich počet označíme S . Následne sa pridá nehnuteľnosť do stromu nehnuteľností s amortizovanou zložitosťou log N. Nakoniec sa prejde zoznam parciel a každej parcele sa pridá do zoznamu nehnuteľností novo pridaná nehnuteľnosť so zložitosťou S.

Celková zložitosť tejto operácie je teda log M + log N + S kde M je počet parciel, N je počet nehnuteľností a S je počet nájdených parciel s rovnakou súradnicou.

1.4 Operácia vymazania parcely/ nehnuteľnosti podľa zadaných dát a GPS súradnice

Nech premenná N je počet nehnuteľností v aplikácii a nech premenná M je počet parciel v danej aplikácií.

Pre zjednodušenie budem uvádzať len prípad pre vymazanie nehnuteľnosti ale prípad pre pridanie parcely je analogicky.

Najprv si musím vyhľadať danú nehnuteľnosť sa amortizovaniu zložitosťou log N. Následne si musím vyhľadať všetky parcely na tej súradnici s amortizovanou zložitosťou log M, počet nájdených parciel si označíme písmenom S. Potom prejsť zoznam nájdených parciel a vymazať im zo zoznamu nehnuteľností práve túto nehnuteľnosť so zložitosťou S. Po tomto kroku už len nasleduje krok vymazania z konkrétneho stromu a to urobíme s nasledovnou zložitosťou. Nech L je počet vrcholov nájdených pod súradnicou nehnuteľnosti. Následne pod zložitosťou L nájdeme konkrétny vrchol podľa nejakého jedinečného kľúča. Nech K je počet rovnakých miním v pravom pod strome potom operácia mazania je (K+1)*logN . Následne budeme musieť pridať týchto K vrcholov naspäť do stromu so zložitosťou log N.

Výsledná zložitosť algoritmu teda bude log N + log M + S + (K + 1)log N + log N kde po zjednodušení dostaneme log N + log M + S, kde N je počet nehnuteľností , M je počet parciel, S je počet nájdených parciel podľa GPS nehnuteľnosti a K je počet opätovne vložených vrcholov rovnakých miním v pravom pod strome.

1.5 Operácia editácie parcely/ nehnuteľnosti

Nech premenná N je počet nehnuteľností v aplikácii a nech premenná M je počet parciel v danej aplikácií.

Pre zjednodušenie budem uvádzať len prípad pre editáciu nehnuteľnosti ale prípad pre pridanie parcely je analogicky.

Najprv si danú nehnuteľnosť nájdeme so zložitosťou log N . Následne máme dva konce . Ak editujeme danému objektu aj GPS súradnicu tak následne musíme tento objekt vymazať so zložitosťou log M + S + log N, kde M je počet parciel S je počet nájdených parciel a následne vložiť tento záznam s novým kľúčom so zložitosťou log N. Takže celková zložitosť bude log N + Log M + S + log N + log N kde po úprave dostaneme log N + log M + S kde N je počet nehnuteľností, M je počet parciel, S je počet nájdených parciel.

2. Použité štruktúry v systéme

V tomto systéme sa použilo tieto štruktúry :

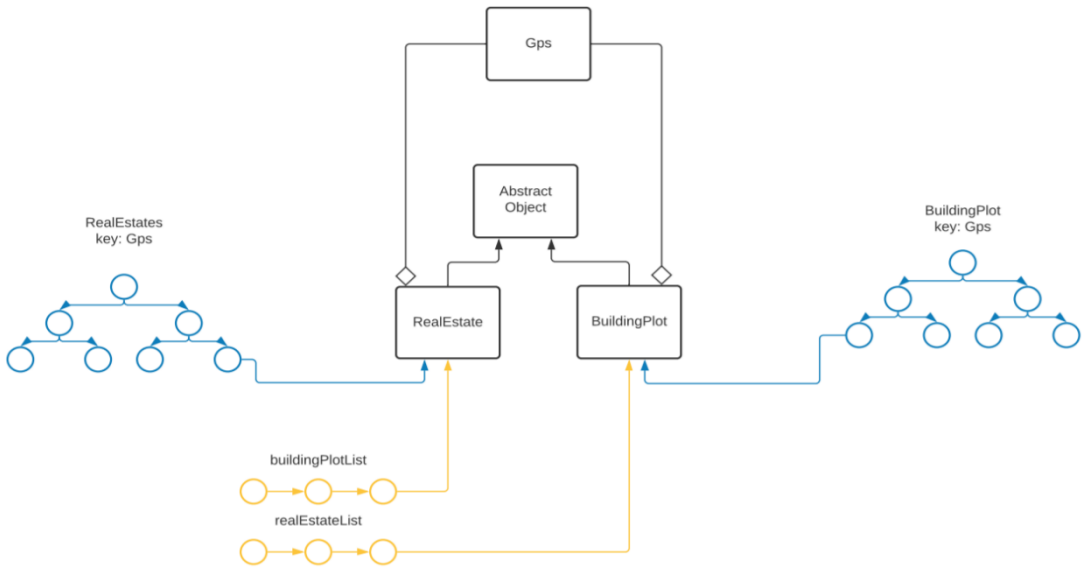
ArrayList - tento druh listu som použil preto lebo potreboval som dynamické pole, kde som potreboval pristupovať na daný index prvku

LinkedList – použil som ho preto lebo potreboval som ukladať informácie rovnakého druhu a vedieť ich všetky prehliadnuť a teda preto som si vybral zreťazený zoznam, a aj preto lebo pri týchto dátach nebola potrebná dynamická práca s poľom.

K-D Strom – túto štruktúru som si vybral na ukladanie informácií o nehnuteľnostiach a o parcelách, pretože obidva objekty majú v sebe takú istú dôležitú informáciu a to GPS. GPS sa skladá z viac sekundárnych kľúčov a teda chcel som aplikovať aj teda viacrozmerné intervalové vyhľadávanie. K-D Strom teda ponúka takéto vyhľadávanie spolu s tým že vo vrchole okrem K sekundárnych kľúčov (GPS má práve 2) máme uložene aj dáta teda naše parcely a nehnuteľnosti.

Ako doplnok treba zahrnúť použitie pomocných štruktúr ako napríklad zásobníka LIFO ako aj Front – FIFO. Tieto štruktúry sa používali pri prehliadkach K-D stromu a to konkrétne inOrder a levelOrde prehliadky.

Na obrázku vidíme grafické znázornenie použitých údajových štruktúr v systéme.

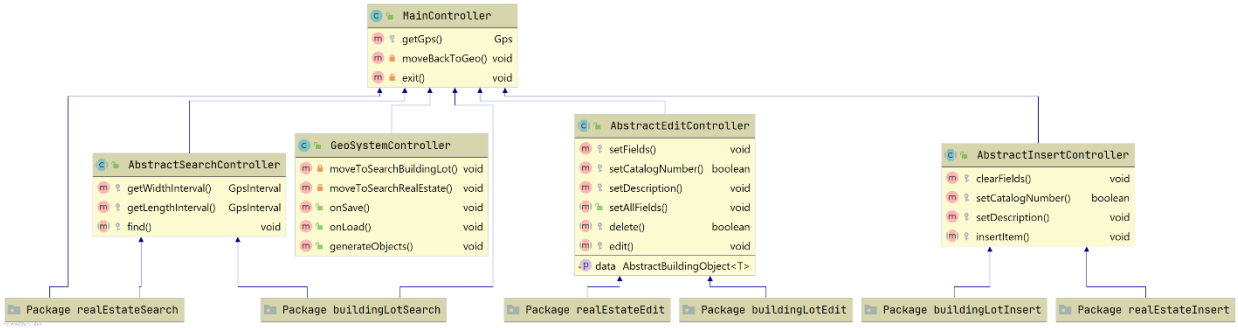


3. UML Diagram tried

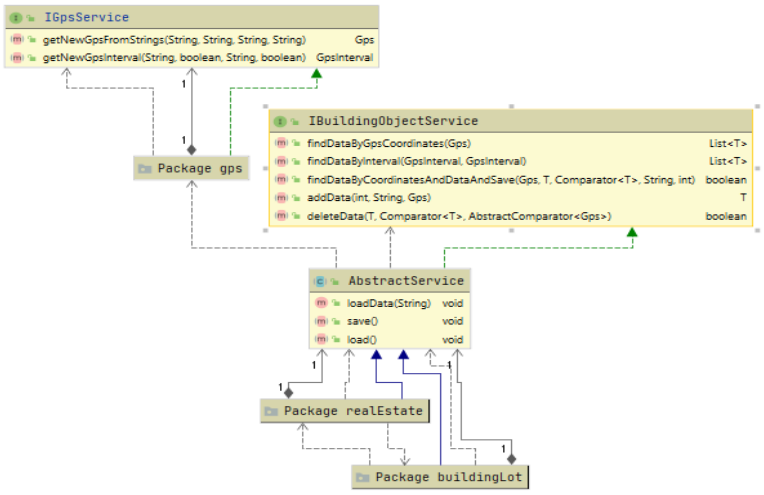
V tomto programe som sa rozhodol používať návrhový vzor MVP, teda Model-View-Presenter. Do môjho návrhu som však pridal jednu vrstvu navyše, ktorú som pomenoval Servis. Táto vrstva je vrstvou medzi presentérom/kontrolérom a modelom. Rozhodol som sa preto z troch dôvodov:

- Odčlenenie biznis logiky do samostatnej vrstvy
- Väčšia obrana vrstvy model – a teda z kontroléra do modelu sa vieme dostať len cez operácie vo vrstve servis a teda nieje tam priamy prístup
- Princíp „Loose coupling“ – a teda kontrolér vie len to čo on potrebuje vedieť na svoje fungovanie a to isté aj model

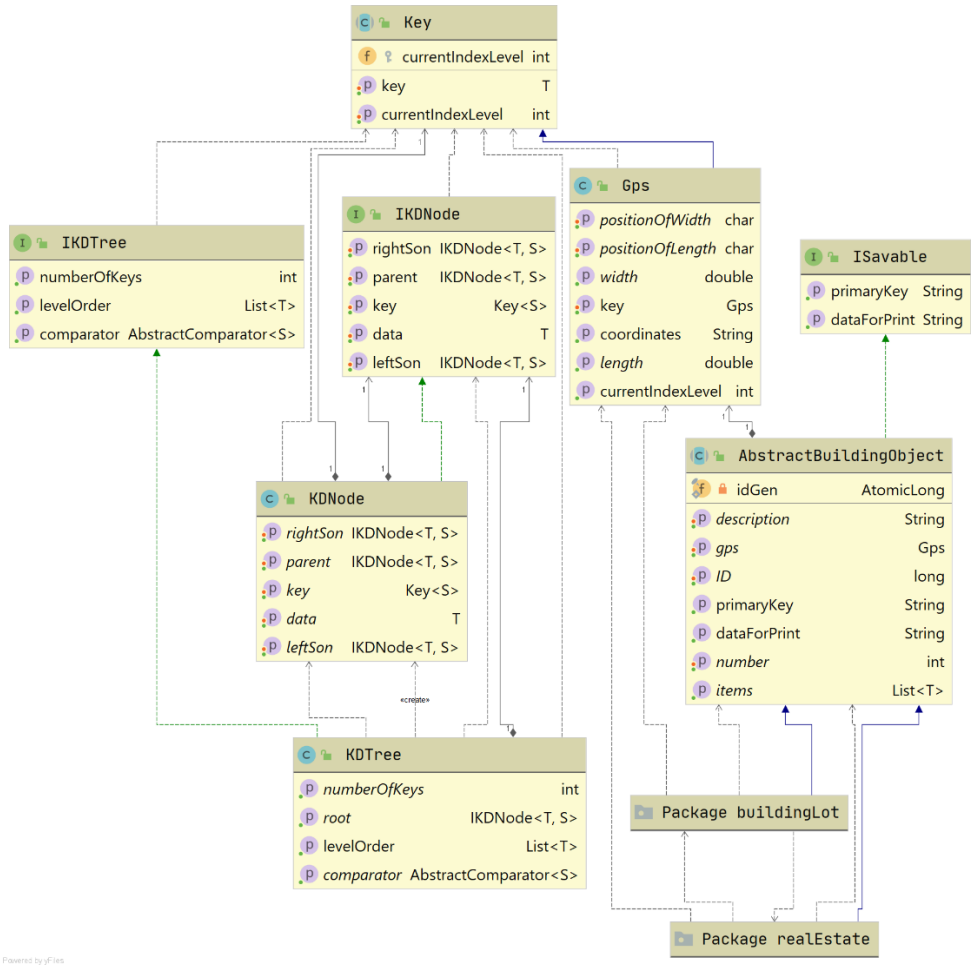
3.1 Presenter/Kontroler



3.2 Service



3.3 Model

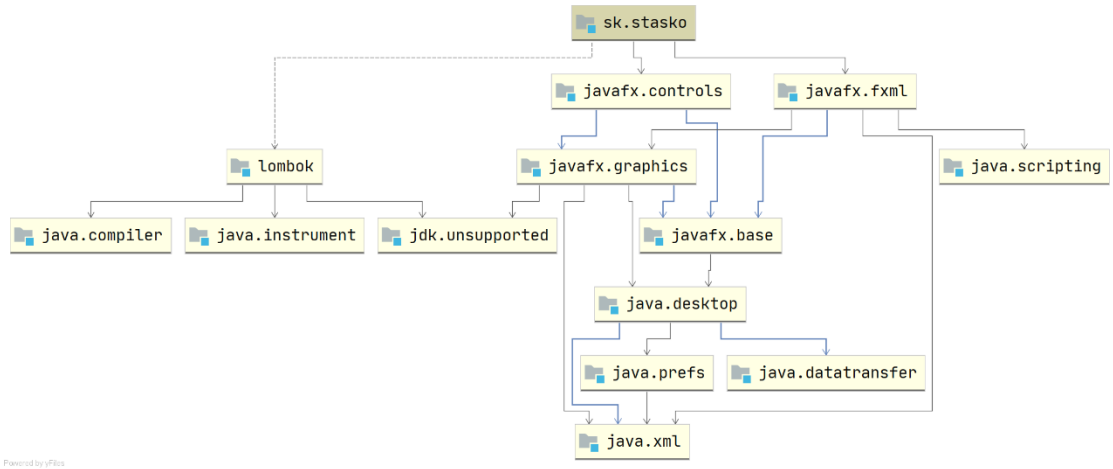


3.4 View

Táto časť je tvorená so súboromi s koncovkou .fxml . Je to jazyk, ktorý je založený na XML a poskytuje možnosť vybudovať používateľské rozhranie separátne od logiky aplikácie

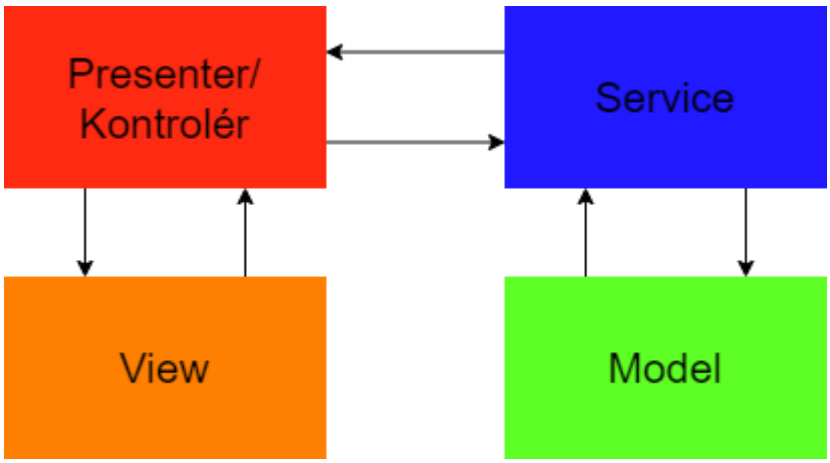
Na obrázku môžeme vidieť ako je aplikačná logika oddelená v package sk.stasko od javaFx.xml a aké závislosti a balíčky používa javaFX a jdkFX na to, aby vykreslila pohľady, ktoré boli navrhnuté v .fxml súboroch. Tieto súbory sú v hierarchii uložené v priečinku resources.

Na doplnenie môžeme vidieť, že projekt používa ešte jednu knižnicu a to lombok. Táto knižnica umožňuje písať kód v Jave viac priehľadný a s menej sa opakujúcimi komponentami pomocou jednoduchých anotácií.



3.5 Celý systém MVP

Na koniec si môžeme tieto komponenty poskladať do jedného systému a ukázať ako spolupracujú.



4. Záver

GeoSystém je aplikácia, ktorá používa namiesto databázy údajovú štruktúru KD strom , kde ukladá parcely a nehnuteľnosti a umožňuje v jednoduchom a intuitívnom rozhraní pre používateľa vykonávať jednoduché operácie pridaj, odober, vyhľadaj a uprav. Systém používa návrhový vzor MVP. Po štarte aplikácie užívateľ má na výber aj vygenerovanie si dát ale aj načítanie si starých dát zo súboru, ako aj ich uloženie. Následne si môže vybrať čo chce vykonávať na akých objektoch dané operácie.