Dokumentácia údajové štruktúry sem1.

# Návrh systému Obrázok, na ktorom je diagram, rad, vývoj, text Automaticky generovaný popis

Nehnuteľnosť obsahuje ArrayList Parciel kde ak leží daná nehnuteľnosť na danej parcele sa priradí a naopak Parcele sa priradí nehnuteľnosť že v nej leží.

# Popis štruktúry programu:

Môj program sa skladá z triedy „Main“ z ktorej sa aplikácia spúšťa, ďalej obsahuje triedy ktoré sa zaoberajú chodom programu, medzi tie patria „GPS, IData, LandShapes, Node, Property, QuadTree, Shape“. Na otestovanie som si vytvoril triedu Tester kde testujem ako funguje môj program. Poslednou triedou je GUI, ešte nie je poriadne dorobená momentálne tam je špageta, ktorú budem chcieť upraviť mám predpripravenú triedu „GuiMethods“ do ktorej dám tela buttonov ktoré sú použité a chcel by som ešte skúsiť dorobiť grafické rozhranie na výkres parciel a nehnuteľností na to mám vytvorenú triedu „Draw“ ale momentálne v nej nie je skoro nič.

GPS -> trieda má 4 atributy – x,y,horizontal,vertical.

IData -> je to interface ktorý využívam pri LandShapes a metóde compareTo.

LandShapes -> abstraktná trieda ktorá dedí a implementuje IDATA, využívam ju na parcely a nehnuteľnosti ako triedu z ktorej dedia compareTo aby som nemal duplicitný kód pre rovnaké triedy obsauje – GPS first,second , notes, identifier.

Property a Shape -> Snažil som sa všetko robiť po anglicky celý projekt nazval som Property ako nehnuteľnost a Shape ako parcelu. Obsahujú Property arraylist Shapov a Shapy arraylist Property.

Node-> je to list stromu obsahuje x1,y1,x2,y2 , depth , arraylist data a 4 svojich synov.

QuadTree -> trieda obsahuje funkcie ako Insert, Delete, Search, Health, ChangeDepth

## Optimalizácia:

Obrázok, na ktorom je rad, štvorec, rovnobežný, diagram

Automaticky generovaný popisMomentálne nemám spravenú nejako dobre optimalizáciu. Jediné čo je spravené je zmena výšky ktorá v určitých prípadoch zlepší zdravie stromu. Optimalizáciu by som chcel spraviť tak že by sa skontrolovalo kde prevládajú prvky. Potom by som celý strom nanovo spravil s tým že by sa nody nevytvárali len /2 ale ak by prevládali prvky vľavo dolu tak by som strom prvých 4 synov presunul na pomer 1:3 a potom delil 2. Nejako takto:

# Insert

Insert obsahuje parameter T data. Na začiatku si vytvorím Node<T> do ktorého vložím root. Vytvorím si pomocné zásobníky nazval som jeden queue a druhy newData. Metóda ide kým je nejaký prvok v zásobniku queue. Vytvorím si ešte premennú T dataQueue čo bude naznačovať prvok ktorý aktuálne riešim. Na začiatku ak je dataQueue null popnem prvok zo zásobníka queue. Pomocou compareTo rozhodnem kam má ísť. Postupne idem hlbšie do stromu ak sa prvok ešte nepriradil a v prípade že ich je viac ktoré treba priradiť dávam ich do queue. Ak náhodou nie sú synovia vytvorený a môžu sa vytvoriť, tak ich vytvorím. Druhý zásobní využívam v prípade keď daný node má viac dát tak skontrolujem či môžu ísť dam ich do queue a nemôžu dám ich do newData potom clearnem arraylist a nahádžem tie data z newData nanovo.

# Delete

Delete obsahuje parameter T data.Na začiatku si vytvorím zasa Node<T> nodekde dám root. Využívam Stack s názvom queue. Na začiatku si musím nájsť prvok ktorý chcem zmazať, node v ktorom sa nachádza uložím do node. Ak ho nájdem postupne pokračujem. Hneď ho zmažem a skontrolujem či daný node treba upravovať. Potom skontrolujem či má synov alebo nemá a pomocou cyklu kontrolujem smerom do hora stromu či sa môže mazať. Pokračujem kým sa nemôže zmazať a potom skončím.

# TreeHealth

Metóda nemá parameter. Postupne prechádzam celý strom a zapisujem si koľko má synov. Ak má synov a sú v nich dáta zistím koľko ich tam je a do zdravia si pripočítam 1/početPrvkovVNode. Nakoniec vydelím zdravie/synmi.

# Search

Metóda obsahuje 4 parametre: x1,y1,x2,y2. Je typu ArrayList<T>. Postupne prechádzam pomocou comparatora prvky a idem hlbšie do stromu. Prvky si ukladám do zásobníka a po každom cykle prebehnem všetky prvky a pomocou compareTo to porovnám či pretínajú danú oblasť ak áno priradím ich do arraylist. Cyklus ide pokým je v zásobniku viac ako 1 prvok.

# ChangeDepthTree

Metóda má jeden parameter a to newDepth. Na začiatku si priradím Node a postupne si ukladám prvky do zásobniku a idem hlbšie do stromu. Ak hĺbka sa má zväčšiť potom nájdem všetky prvky ktoré sa nachádzajú na starej hĺbke a nanovo ich insertujem a zmažem. Ak sa má hlbka zmenšiť postupne prechádzam od najnižšie syna až po novú výšku prvky a dám ich do otca. Problem bol insert ale vyriešil som ho tak že zmením root na daný node. Starý root si uložím a nakoniec root zasa zmením na starý root. Nakonci tiež zmením výšku.