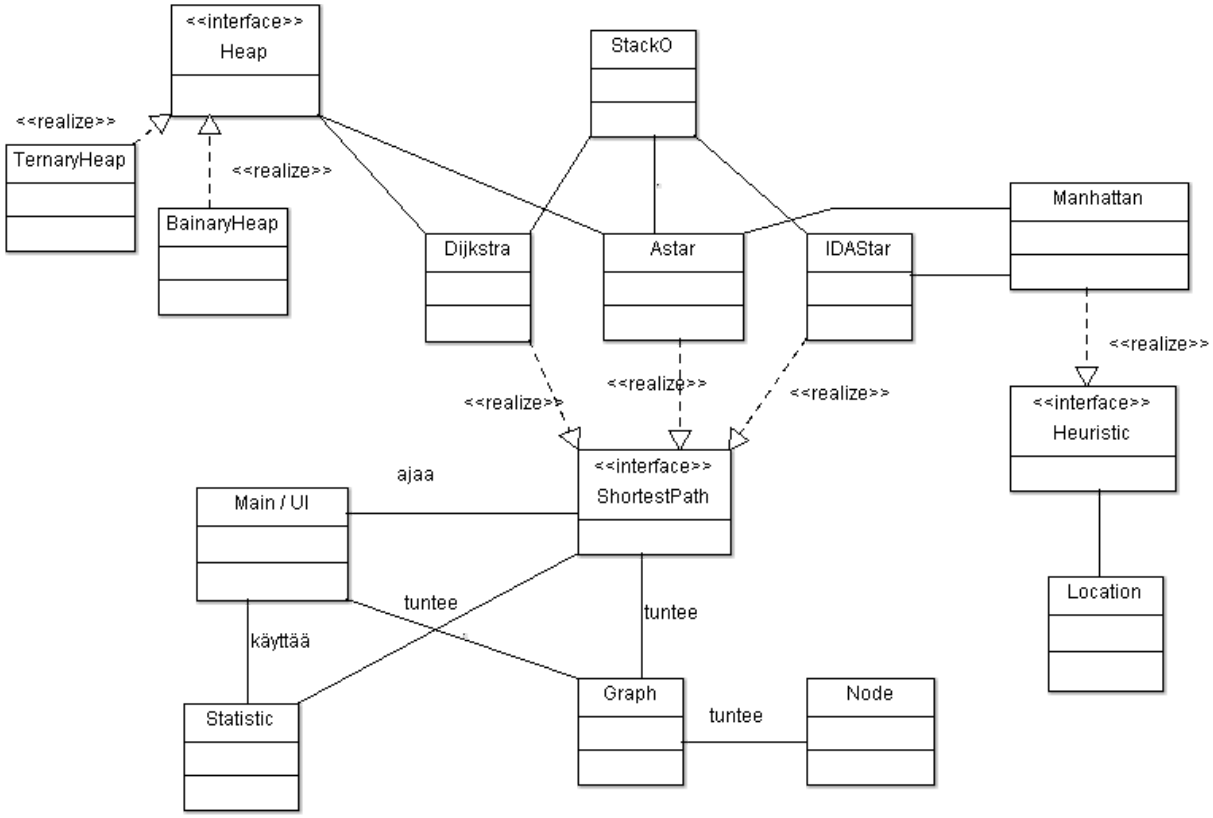
Ohjelman yleisrakenne

Alla ylätason luokkakaavio, joka kuvaa ohjelman rakennetta:



Saavutetut aika- ja tilavaativuudet

Dijkstra

- Alustus vie aikaa O(|V|), kun kaikille koordinaatiston pisteille eli verkon solmuille asetetaan etäisyys (initialize-metodi)
- Algoritmi käyttää minimikekoa. Keko voi olla 2-keko (vanhemmalla max 2 lasta) tai 3-keko (vanhemmalla max 3 lasta). Algoritmi käyttää heap-insert (add-metodi) ja heap-del-min operaatioita (poll-metodi). Keko-operaatiot on nimetty Javan PriorityQueue:n mukaan, jotta tietorakennetta voidaan helposti vaihtaa
    - 2-keko
        - keko-operaatioiden aikavaativuus O(log n), kun keossa n alkiota
        - Rivillä 52 tehdään |V| kappaletta heap-del-min eli aikaa kuluu O(|V| log |V|)
        - Rivillä 74 tehdään |E| kappaletta heap-insert operaatioita
        - Kokonaisuudessa aikaa kuluu (while luupissa riveillä 50-82) O(|E|+|V| log |V|)

    - 3-keko

        Heap-del-min: O(*d* log *n* / log *d*), missä d on lapsien määrä
        Heap-insert: O(log *n* / log *d*), missä d on lapsien määrä
        Kokonaisuudessa aikaa kuluu (while luupissa riveillä 50-82) ...
- Tilavaativuus on O(|V|), koska kaikille taulukoille ja keolle varataan tilaa solmujen lukumäärän verran.

A*

- A* algoritmin “ohjelmarunko” on sama kuin Dijkstralla. Ainoa ero on, että A* käyttää heuristiikkaa ja kutsuu sen getToEnd-metodia etäisyysarvion saamiseksi ja se voidaan päätellä vakioajassa, O(1).
- Myös tilavaativuus on sama kuin Dijkstralla, eli O(|V|), koska kaikille taulukoille ja keolle varataan tilaa solmujen lukumäärän verran.

IDA*

-

Suorituskyky- ja O-analyysivertailu (mikäli työ vertailupainotteinen)

Työn mahdolliset puutteet ja parannusehdotukset

Lähteet

- https://www.cs.helsinki.fi/u/jkivinen/opetus/tira/k16/luennot.pdf (Dijkstra, A*)
- https://en.wikipedia.org/wiki/D-ary_heap (3-keko)