Oblig5 INF1000-SIKT

Ruteplanlegging
Finn raske reiser med Trikk og T-bane i Oslo.

Innleveringsfrist: 15 november kl. 23.49 i Devilry

Formål

Det er 6 T-banelinjer (1,2,3,4,5 og 6) og 6 trikkelinjer (11,12,13,17,18 og 19) i Oslo. På mange av stoppestedene/stasjonene, men ikke på alle, er det mulig å gå over til en stasjon med samme navn på en annen linje. Eksempel på slike stasjoner med mange overganger er Forskningsparken, Majorstuen, Storo og Jernbanetorget. Alle opplysningene vi skal bruke om linjer og stasjoner ligger på en fil. Vitsen med denne planleggeren er ikke å finne eksakt reisetid for en bestemt avgang, men å finne den jevnt over beste reisemåten mellom to stasjoner ved å bruke maksimalt én overgang.

I denne obligen skal du derfor først lese inn filen: «TrikkOgTbane.txt» med alle t-bane-linjer og trikkelinjer i Oslo, og så kunne regne ut alternative måter å reise fra en gitt stasjon/holdeplass på en trikk- eller t-bane til en annen gitt stasjon ved å gjøre ***maksimalt én omstigning*** til en annen t-bane eller trikke-linje. Hvordan vi gjør det er nærmere beskrevet nedenfor. Vi skal også kunne regne ut en sannsynlig reisetid for de ulike reiserutene, og som en siste oppgave kunne skrive ut en forventet raskeste reise med bare én eller ingen omstigning. Vi kan tenke oss at det du lager er en første skisse til en ruteplanlegger for Ruter, som jo driver disse linjene. Vi skal se bort fra busser i denne oppgaven, men det vil relativt lett kunne legges inn, når du først har fått t-baner og trikker til å virke.

Oblig 5 skal leveres individuelt med egen kode (men dere kan selvsagt gi hverandre tips om måter å løse de ulike delproblemene) - men altså ingen kopi av andres kode.

Nærmere spesifikasjon

Det brukeren av dette systemet skal gjøre er å taste inn to stasjonsnavn – ett hun ønsker å reise fra (anta stasjon xxx) og ett hun ønsker å reise til (anta stasjon yyy). Systemet skal først sjekke at dette er navn som finnes på stasjoner i systemet, og be om om-tasting av hvert av navnene som eventuelt ikke finnes i systemet.

Brukeren skal så få tilbakemelding om hvilken linje hun skal reise med, og i hvilken retning, eventuell overgang, samt estimert reisetid (se under). Dersom det finnes en direkte-rute fra xxx til yyy, skal denne skrives ut. For eksempel, hvis hun ønsker å reise fra Majorstuen til Slemdal:

```
Ta t-bane linje 1 fra Majorstuen til Slemdal i retning
Frognerseteren. Estimert reisetid: 5.0 min.
```

Dersom det er nødvendig med en overgang, skal det skrives ut **alle** mulige reiseveier med en overgang. Utskrift av en av disse kan for eksempel se slik ut (for reise fra Blindern til Disen):

```
Ta t-bane linje 4 fra Blindern til Jernbanetorget i retning
Bergkrystallen og deretter trikk linje 12 retning Kjelsås til
Disen. Estimert reisetid: 21.0 min.
```

Vi antar følgende forenklinger i denne oppgaven:

- En direkte reise er alltid raskere enn en reise med omstigning.
- Vi regner med at alle trikker og t-baner går hvert 10. minutt, slik at vi estimerer at det ved overgang tar 5 minutter å vente på neste trikk/t-bane. Eventuell gangtid mellom holdeplassene regnes ikke med.
- Både trikker og t-baner bruker i snitt 1 minutt mellom to etterfølgende stasjoner.
- Vi regner ikke med noe ventetid på fra-stasjonen (xxx), bare tiden det tar fra man går på den første trikken/t-banen og til man er fremme på til-stasjonen (yyy).
- Ingen linje har mer enn 32 stasjoner og det er ikke høyere linjenummer enn 19.
- Vi regner med at vi alltid kan reise i begge reise-retningene på en linje fra alle stasjonene på linjen.
- T-bane 6 og 4 heter av og til Ringen, og det har vi ordnet i datafilen ved å legge inn en fiktiv siste stasjon som heter «Ringen» på linje 4 og 6. Vi tester da særskilt og aksepterer ikke «Ringen» som verken fra- eller til-stasjon.

Om selve løsningen

Vi anbefaler en løsning som bruker fem klasser: `Oblig5`, `Planlegger`, `Linje`, `Stasjon` og `Overgang`. `Oblig5` inneholder `main`. Den oppretter et objekt av klassen `Planlegger` og kaller en metode i denne for å lese inn filen mens den oppretter objektene av klassene `Linje` og `Stasjon`, samtidig som alle stasjonene legges inn i en HashMap med stasjonsnavnet som søkenøkkel.

Planlegger inneholder også en metode (`lesFraTil`) som leser inn fra brukeren mellom hvilke to stasjoner den skal finne reiseruter. `Planlegger` inneholder også en metode

```
void beregnRuter(Stasjon fraSt, Stasjon tilSt)
```

som finner disse reiseveiene. `Linje` og `Stasjon` er de klassene som inneholder 'alle' opplysningene om hvilke linjer og stasjoner vi har og hvor det er mulig med overgang. Det er viktig at `Stasjon` inneholder opplysninger om alle linjene som går gjennom stasjonen med det navnet og for hver slik linje hvilket nummer denne stasjonen har på den linjen.

Det er viktig at hvis navnet på en stasjon er samme på en linje som på en annen linje, skal vi ***ikke*** lage et nytt objekt av klassen `Stasjon`, men bruke det samme `Stasjons`-objektet vi allerede har laget. Vi ser da at objekter av klassen `Stasjon` må ha en liste eller tabell (array) over hvilke linjer den er med i. Tilsvarende bør objekter av klassen `Linje` ha en array over hvilke stasjoner den inneholder. Du bør også nummerere stasjonene langs en linje slik at `Stasjons`-objektene også vet hvilke nummer de har på hver av de linjene de er en stasjon på (det er jo på slike stasjoner som er med på to eller flere linjer at det er mulig med overgang til en annen linje). Dette kan lagres i en `int`-array for hvert `Stasjons`-objekt.

Vi kan se på starten av Linje 1 – t-banen fra Frognerseteren til Ellingsrudåsen.

```
*Linje* 1
Frognerseteren
Lillevann Skogen
Voksenlia
Holmenkollen
Besserud Midtstuen
Skådalen Vettakollen
Gulleråsen
Gråkammen Slemdal
```

Ris Gaustad Vinderen
Steinerud Majorstuen
Nationaltheatret
……

Det er totalt 32 Stasjoner på linje 1 som da slutter med Ellingsrudåsen. Det kan da være lurt å gi Frognerseteren nummer 1, Lillevann nummer 2, osv, og Ellingsrudåsen nummer 32 (vi trenger disse numrene senere for lettere å regne ut reisetider og retninger på reisen).

Framgangsmåten (algoritmen) for å søke etter en reise fra stasjon xxx til stasjon yyy kan se slik ut:

1. Finn først fra-stasjonen xxx. Stasjonsobjektet med det navnet inneholder pekere til alle linjer L1 som du kan reise med fra xxx.
2. Finn så til-stasjonen yyy. Stasjonsobjektet med det navnet inneholder pekere til alle linjer L2 som du kan reise med til yyy.
3. Hvis en av linjene i L1 også er i L2 har du funnet en god reisevei fra xxx til yyy, og du er nå ferdig med søket . Finn da ut om xxx har mindre eller større nummer på denne linja enn yyy, da det bestemmer reiseretningen (navnet til den endestasjonen som står på trikken eller t-banen) brukeren skal reise for å komme til yyy. Skriv så ut reisen som vist over.
4. Hvis de to stasjonene ikke ligger på samme linje, må vi finne en overgang som virker. Det er alltid mulig i Oslo. Vi går fram som følger:
    a. For alle linjene i L1 (en etter en) søker vi gjennom alle stasjonene, en etter en om noen av dem har en linje som inneholder til-stasjonen.
    b. Hvis ja, har vi da funnet en stasjon hvor vi kan foreta overgang, og vi gir en utskrift som vist over.
    c. Vi forsetter søket til vi har utskrift av alle mulige reiseveier med én overgang.
    d. (Kan droppes:) Til slutt skal du skrive ut den av reiseveiene som har kortest forventet reisetid (eller en av dem, hvis det er flere reiseveier med like kort tid). Hittil korteste reisevei bør da lagres fortløpende for enkelt å kunne svare på dette spørsmålet. Sannsynligvis er det best å bruke en egen klasse `Overgang` til dette.

Leveranse

I tillegg til Java-programmet som beskrevet over, skal du lage (og levere) et UML klassediagram med forholdet mellom klassene (Oblig5, Planlegger, Overgang, Linje og Stasjon). Husk å få med navn på forbindelsene og antall på hver side. Det er ikke nødvendig å angi variable eller metoder i klassene.

Tips

I tillegg til metodene nevnt over, kan det være nyttig å implementere følgende hjelpemetoder:

I klassen Stasjon:

```
void registrerNyLinje(Linje nyLinje,int stasjonsNummer)
// registrer at den angitte linja går gjennom denne Stasjonen og
// at denne stasjonen på denne linja har det oppgitte nummeret
```

I klassen Linje:

```
boolean inneholder(Stasjon st)
// Returnerer sann hvis st befinner seg på denne linjen

void nyStasjon(Stasjon s)
// Legger inn den angitte stasjonen på denne linjen

String type()
// returnerer 't-bane' eller 'trikk' for denne linjen

String retning(Stasjon fra, Stasjon til)
// returnerer navnet på endestasjonen på denne linjen i retning
// fra:'fra' til:'til' på denne linjen


double tid(Stasjon fra, Stasjon til)
// returnerer tiden det tar å kjøre mellom de to stasjonene
```

Lykke til !