Oblig5 - Ruteplanlegging, den siste obligen i INF1000h2013. 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 finne den jevnt over beste reisemåte mellom to stasjoner med maksimalt én overgang.

I denne obligen skal du derfor først lese inn fila: «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

Dette er et enkelt system som trenger fem klasser: Oblig5, Planlegger, Linje, Stasjon og Overgang. Oblig5 inneholder 'main'. Den oppretter et objekt av klassen Planlegger som så leser inn fila mens den oppretter objektene av klassene Linje og Stasjon. 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 (bare ett objekt for alle stasjoner som heter det samme) og for hver slik linje hvilket nummer denne stasjonen har på den linja.

Først leser du fila og oppretter objekter av klassene Linje og Stasjon. 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 over hvilke linjer den er med i. 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). 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).

Det brukeren av dette systemet skal gjøre er å taste inn to stasjonsnavn – ett hun ønsker å reise fra og ett hun ønsker å reise til. Si fra stasjon xxx til stasjon yyy. Systemet ditt skal først sjekke at dette er navn som finnes på stasjoner i systemet, og be om om-tasting av hvert av navnene du ikke finner det i systemet. Så starter søket. Framgangsmåten (algoritmen) for det du skal søke 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 (fordi oppgaven antar at en direkte reise alltid er raskere enn en reise med omstigning). Finn da ut om xxx har mindre eller større nummer på denne linja enn yyy. Det bestemmer retningen (navnet til den endestasjonen som står på trikken eller t-banen) du sier hun skal reise for å komme til yyy. Hvis hun f.eks ønsker å reise fra Majorstuen til Slemdal, skriver du ut:
   Ta t-bane linje 1 fra: Majorstuen til Slemdal i retning Frognerseteren. Estimert reisetid : 9.0 min.
   (Hvordan vi beregner reisetider, se punkt 5 nedenfor.)
4) Hvis de to stasjonene ikke lå 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, her vist med et eksempel: Utskrift av én mulig reise fra Blindern til Disen er:
      Ta t-bane linje 4 fra Blindern til Jernbanetorget retning Bergkrystallen
      og deretter trikk linje 12 retning Kjelsås til Disen. Estimert reisetid: 32.0 min.
      (det er mange, minst 15 mulige reiser mellom Blindern og Disen).
   c. Vi forsetter søket til vi har utskrift av alle mulige reiseveier med én overgang.

5) Beregningen av estimert tid skal vi gjøre litt forenklet. Vi regner bare med slik det er på dagtid, og da gjelder:
    a. Vi regner med at alle trikkene (de med linjenummer > 10) går hvert 10 minutt, slik at du kan regne med å vente på riktig trikk i 5 min. ved en overgang til en trikk.
    b. Vi regner med at alle t-banene (de med linjenummer <10) går hvert 15 min, slik at du kan regne med å vente i 7.5 min. ved en overgang til en t-bane.
    c. En trikk bruker i snitt 1,4 min mellom to etterfølgende stasjoner, mens t-banene bruker 1,8 min.
    d. Ved en overgang skal du (nesten) alltid over til et annet sted/plattform og du skal alltid beregne 3 min. på det.
    e. Vi regner ikke noe ventetid på den første stasjonen hvor du går på, bare selve reisetida.
    f. Fordi vi regner med kommatall (1,4 og 1,8) som ikke har en eksakt representasjon i maskinen, vil utskriften kunne bli stygg. Bruker du easyIO, vil `'String s = Format.center(d,6,1)'` omgjøre en doublevariabel d til en String s med ett siffer bak komma og lengde=6.

Leveranse

En fullstendig besvarelse av denne obligen er et UML-klassediagram og et .java-program som tilfredsstiller punktene a) – f) nedenfor.

a) Lag først et UML-diagram om forholdet mellom klassene : Oblig5, Planlegger, Overgang, Linje og Stasjon. Få med navn på forbindelsene og antall på hver side.
b) Skriv metoden 'lesFil' i Planlegger som leser 'TrikkOgTbane.txt' og oppretter de objektene og andre datastrukturer du trenger (Det kan her være fornuftig også å lage en HashMap med alle Stasjonene med navnene deres som søkenøkkel slik at du kan sjekke om det er kjente stasjonsnavn brukeren taster inn i neste punkt).
c) Skriv metoden: 'lesTilOgFra' som spør brukeren om fra-stasjonen og til-stasjonen og som ikke gir seg før brukeren har tastet to lovlige navn. I fila 'TrikkOgTbane.txt' er stasjonsnavn med to eller tre separate ord, som 'Lille Frogner allé', skrevet med bindestreker mellom ordene, slik: 'Lille-Frogner-allé'. Du får selv bestemme deg for om du vil be brukeren skrive disse bindestrekene, endre litt på 'TrikkOgTbane.txt, eller programmere slik at også blanke på disse plassene virker. Se også tips 5.
d) Skriv metoden '`boolean inneholder(Stasjon st)`' i klassen Linje, som returnerer sann hvis stasjonen `st` befinner seg på den linja som Linje-objektet representerer.
e) Skriv metoden '`void beregnRuter( )`' som ut fra L1 og L2 finner, og får skrevet ut enten én reisevei (punkt 3), eller en rekke reiseveier (punkt 4) fra xxx til yyy. Alle reiseveier som finnes skrives ut som vist i punkt 3 eller punkt 4.
f) Finn en av de reiseveiene du har funnet som er den med kortest forventete reisetid og skriv den ut til sist. Du ser at du i punkt e) bør lagre data om den hittil korteste reiseveien for enkelt å kunne svare på dette spørsmålet. Sannsynligvis trenger du en egen klasse Overgang til dette.

Tips, forenklinger og kommentarer

1. T-bane 6 og 4 heter av og til Ringen, og det har vi ordnet i datafila 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.
2. Ingen linje har mer enn 32 stasjoner og det er ikke høyere linjenummer enn 19.
3. På noen få stasjoner på trikkelinjene og én på t-banestasjon er det bare mulig å gå på i en retning, men det ser vi bort fra her. Vi regner med at vi alltid kan reise i begge reise-retningene på en linje fra alle stasjonene på linja.
4. Det er nok enklest i Stasjon-klassen å representere linjene som går gjennom en stasjon med en array med pekere til disse Linje-objektene; og tilsvarende, en array med pekere til de stasjonene som i er langs med en linje i Linje-klassen.
5. Hvis du tester programmet ditt på en Windows-maskin (helt greit) vil systemet ditt ikke virke når du taster inn stasjonsnavn du enten vil reise fra eller til som inneholder: æ, ø, å. (fordi Windows i kommandovinduet opererer med et annet tegnsett enn Java for: æ, ø, å). Det skal vi ikke ta hensyn til. Test da evt. programmet ditt bare med reiser til eller fra med navn som ikke har disse tre bokstavene i seg. På en linux-maskin har du ikke disse problemene.
6. Det er viktig for å få en enklere og oversiktlig løsning å lage en rekke hjelpemetoder – for eksempel 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
```

og i klassen Linje:

```
void nyStasjon(Stasjon s){ … }
// Legger inn den angitte stasjonen på denne linja

String type (){ … }
// returnerer 't-bane' eller 'trikk' for denne linja

String retning (Stasjon fra, Stasjon til) { … }
// returnerer navnet på endestasjonen på denne linje hvis vi kjører
// fra:'fra' til stasjonen 'til' på denne linja

double venteTid(){ … }
// returner 5 hvis dette er en trikkelinje, ellers 7,5

double tid(Stasjon fra, Stasjon til) { … }
// returnerer tida det tar å kjøre mellom de to oppgitte stasjonene
```

Lykke til !