Mittuniversitetet
MID SWEDEN UNIVERSITY

Lab 6 – Trains

Objektorienterad programmering i C++

Syfte: Lösning av ett komplext problem.

Lab 6 – Trains

Bakgrund

Det utlandsbaserade järnvägsbolaget *Ironbend Train and Brain Railway Inc*. har lagt ut ett konsultuppdrag för att lösa ett internt informationsteknisk problem. Kontaktperson i Europa är undertecknad som åtagit sig att med hjälp av en uppsättning briljanta OOP-studenter försöka lösa problemet.
På sikt vill man konstruera ett system för att enkelt administrera sina lok och vagnar samt för att hålla ordning på alla tåg, vilket/vilka lok som drar ett tåg, vilka vagnar som ingår, i vilken ordning de kommer i tåget o.s.v. Från interna källor i företaget som helst vill vara anonyma har jag erfarit att det råder totalt kaos. Ingen vet var något fordon finns. Det lär vara ren tur om något tåg kan avgå mot rätt destination vid rätt tidpunkt, om det överhuvudtaget finns något tåg att tillgå. Ett akut konkurshot vilar likt ett Damokles-svärd över företaget.

Uppgift 1

Skapa en objektorienterad modell för tåg, lok och vagnar. Modellen ska baseras på ett antal klasser och hur dessa är relaterade till varandra. Utnyttja UMLs klassdiagram. Implementera därefter klasserna.

Utgå från följande fakta och beskrivningar:

- Dragande fordon: diesellok och el-lok
- Personvagnar: sittvagnar och sovvagnar
- Godsvagnar: täckta och öppna
- Varje tåg har ett (eller flera) lok samt ett antal vagnar. I samma tåg kan det finnas flera typer av både lok och vagnar.
- Alla fordon som rullar på rälsen har ett unikt id-nummer.
- Varje tåg har ett unikt tågnummer. Tågnummer är ett *logiskt* begrepp som avser en viss kommunikationsförbindelse vid en viss tidpunkt på dygnet. I begreppet ligger typ av lok, antal vagnar av olika typer, avgångsstation, destinationsstation, avgångstid och ankomsttid. T.ex. tåg nr 859:
  från South Bend Central kl 16.13, ankomst till Tahoma City 19.43 varje dag består alltid av ett el-lok och tre sittvagnar. Däremot säger inte tågnumret exakt vilka lok/vagnar som används, det kan variera från dag till dag.
- Varje tåg genomgår 6 tillstånd ('states') under sin levnad:
  - NOT ASSEMBLED : tåget existerar endast som logiskt begrepp, inga lok/vagnar har kopplats ihop
  - NOT READY : tåget är under uppbyggnad på aktuell station, vagnar kopplas ihop m.m.
  - READY : taget är klart för avgång från ursprungsstationen
  - RUNNING' : tåget är på väg från sin avgångsstation till sin destinationsstation

    - ARRIVED : tåget har kommit fram till slutstationen
    - FINISHED' : tåget har plockats isär och de ingående rälsfordonen finns parkerade på slutstationen och kan användas i andra tåg
- Ett tåg ska sättas ihop av tillgängliga lok och vagnar på avgångsstationen. För varje tåg ska användaren alltid kunna få information om bl.a.
    - tågnummer
    - ursprung och destination med tider
    - tillstånd
    - typ för de ingående rälsfordonen i rätt ordningsföljd
    - om tåget är hopkopplat: id-nummer för varje ingående rälsfordon i rätt ordningsföljd

Naturligtvis kan inte samma fordons-id förekomma två gånger, (detta lär ha förekommit!!) inte heller kan det vid samma tidpunkt existera två tåg med samma tågnummer.
Dessutom ska följande uppgifter om olika fordonstyper ingå i modellen:

- Personvagn - antalet sittplatser, Internet ja/nej
- Sovvagn - antalet bäddar
- Öppen godvagn - lastkapacitet i ton och lastyta i kvadratmeter
- Täckt godsvagn – lastvolym i kubikmeter.
- El-lok - max hastighet i km/h och effekt i kw.
- Diesel-lok - max hastighet i km/h och bränsleförbrukning i liter/timme

Unyttja virtuella funktioner för att kunna skriva ut information om en godtycklig samling lok och vagnar.

Uppgift 2
Du ska utveckla en prototyp till en applikation där verksamheten kan simuleras. Här ska du bl.a. använda klasserna från uppgift 1.

Prototypens funktionssätt kan beskrivas på följande sätt:
Körning av ett antal tåg ska simuleras från en viss tidpunkt och fram till en annan tidpunkt under samma dygn. Styrning av simuleringen ska göras genom att användaren själv stegar fram tiden i t.ex. 10-minuters intervaller. För varje sådant tidssteg ska nödvändiga förändringar i alla tågens aktuella tillstånd automatiskt utföras. Användaren ska när som helst kunna få information om tågen: tågnummer, state, avgångsstation, destination, ingående lok/vagnar o.s.v.

På stationerna finns en pool av olika rälsfordon som ska användas för att bygga upp tåg som ska avgå.

- Det aktuella tågsättet sätts ihop 30 minuter före angiven avgångstid och övergår då från tillståndet NOT ASSEMBLED till NOT READY.
- Tio minuter före avgångstid övergår tåget till tillståndet READY.

- Vid avgångstid inträder tillståndet RUNNING.
- När tåget kommit fram till slutstationen övergår tillståndet från RUNNING till ARRIVED.
- Efter 30 minuter i tillståndet ARRIVED återlämnas de ingående rälsfordonen till fordonspoolen vid slutstationen och tågets tillstånd blir FINISHED.

Information om fordonspoolernas aktuella innehåll för de olika stationerna ska också kunna fås i varje läge.

För varje tidssteg ska aktuell tid och alla tillståndsförändringar som skett under det senaste tidsintervallet skrivas ut på skärmen så att händelseförloppet kan övervakas kontinuerligt.

Alla stationer, fordon och deras attributvärden ska läsas in från textfil(er) vid simuleringens start. Där ska också framgå på vilken station varje fordon står vid simuleringens start. På samma sätt ska alla tåg och de uppgifter som hör samman med dem definieras i förväg genom att läsa in uppgifter från en textfil. Filerna ska arrangeras så att fordon/stationer/tåg kan läggas till/ändras inför en körning utan att programmet måste kompileras om. Det innebär att inga simuleringsdata får hårdkodas. Programmet ska bara känna till namn på högst tre filer som ska läsas in vid programstarten. Dessa filer ska innehålla alla nödvändiga data eller hänvisa till andra filer som ska läsas in.
Observera att startinformationen om ett tåg inte får innehålla exakta id-nummer för fordonen. Varje fordon i ett tåg ska bara anges med sin typ, t.ex. ”ELLOK” eller med någon annan kodning. Exakta fordons-id avgörs först när tåget sätts samman. Avgörande är då naturligvis vilka fordon som finns på stationen vid den aktuella tidpunkten.

Om tåget inte kan sättas samman för att lok eller vagnar av rätt typer saknas blir tåget försenat i väntan på fordon i ankommande tåg. Tåget ska då vara kvar i tillståndet NOT ASSEMBLED. Angångstiden och hela tidsschemat för tåget ska senareläggas 10 min och ett nytt försök ska göras då. Detta kan naturligtvis upprepas flera gånger.

När simuleringen är slutförd ska statistik presenteras över:
- lyckade tågkörningar
- försenade tåg , antalet minuter
- tåg som aldrig lämnade avgångsstationen

Krav på lösningen

- I klasshierarkin ska virtuella funktioner användas för att skriva ut information om fordon.

- Användaren ska alltid kunna få fullständig information om tåg, stationer och de fordon som finns där.
- Ett fordon ska kunna följas genom sitt id-nummer från en station via ett tåg till en annan station.
- Minneshanteringen ska vara korrekt, varje `new` ska matchas av ett `delete`.

Redovisning
Zippad fil med hela lösningen:

- Källkod med klassdefinitioner + implementationer + prototyp-applikationen + textfiler med simuleringsdata.
- Labbrapport som beskriver din lösning inkluderande UML-diagram.