

Proiect la Sisteme de Control Distribuit

Faur Ioan-Aurel, ISA, an 4, gr. 30143

Abstract

În cadrul orei de Sisteme de Control Distribuit sa pus problema implementării unui simulator de rețea ferată. Acesta, pornind de la o hartă feroviară și un tabel cu id-urile trenurilor care vor circula pe această hartă, va trebui să construiască dinamic orarul trenurile, specificând în prealabil ruta trenurilor și timpul minim și maxim în care trenul va parcurge o anumită locație din ruta specificată. Sistemul de simulare va trebui sa permită introducerea unor întârzieri pe o anumită rută și vizualizarea lor. Deasemenea utilizatorul ar trebui să fie atenționat cu privire la aceste modificări.

Pentru această problemă am propus un sistem software cu o arhitectură complet distribuită care va raspunde cerințelor de mai sus.

Introducere

Sistemul software propus să rezolve problema de mai sus se numește *Railsimulator* (Fig 1.) și prin abordarea pe care o are asupra cerinței, poate să îndeplinească toate obiectivele de mai sus.

Metoda propusă este de a împărți sistemul în mai multe aplicații mai mici, fiecare cu un scop bine definit și care să permită o arhitectură distribuită. Am ajuns astfel la un sistem compus din una sau mai multe *interfete grafice* care nu fac altceva decât să deseneze o hartă feroviară, sau o parte din ea, și să interpreteze comenzile transmise de o altă componentă numită *simulator*. Acesta este o mică aplicație care primește un orar al trenurilor de la un *controler* și la o anumită unitate de timp bine definită construiește o comandă de simulare pe care o trimite interfeței grafice la care este conectat.

Unitatea de timp este dată de un *server de sincronizare* care la un moment bine definit trimite o simulare a unui semnal de ceas simulatoarelor conectate la el. Se consideră ca trimiterea și primirea acestui semnal are loc aproape instantaneu. În realitate lucrurile stau cu totul altfel, astfel că sunt folosite metode mult mai avansate de sincronizare dar care sunt mai greu de implementat sau chiar mai

scumpe.

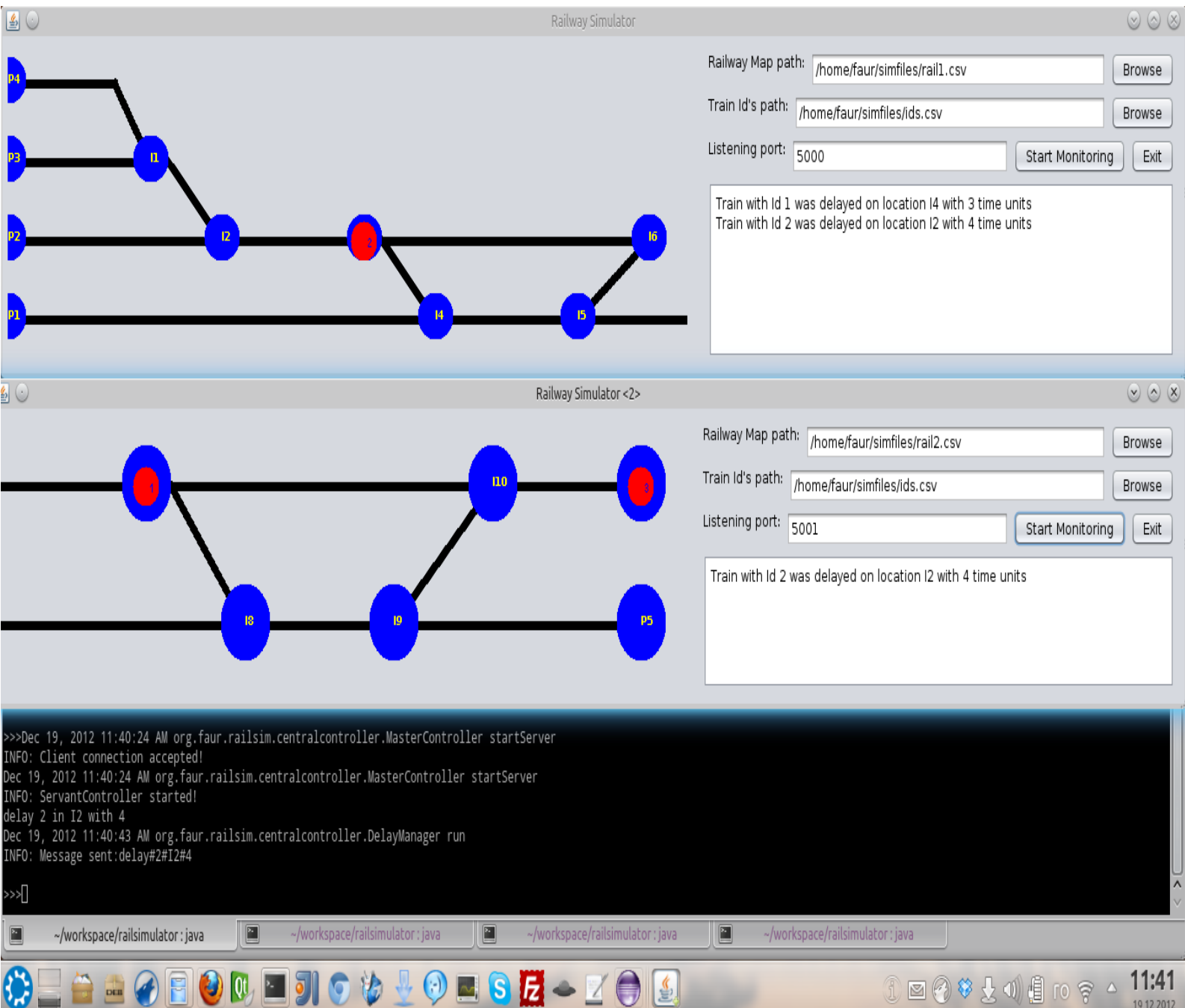


Fig. 1: Railsimulator

Componentele aplicației *Railsimulator* după cum se arată și în Fig. 2 sunt

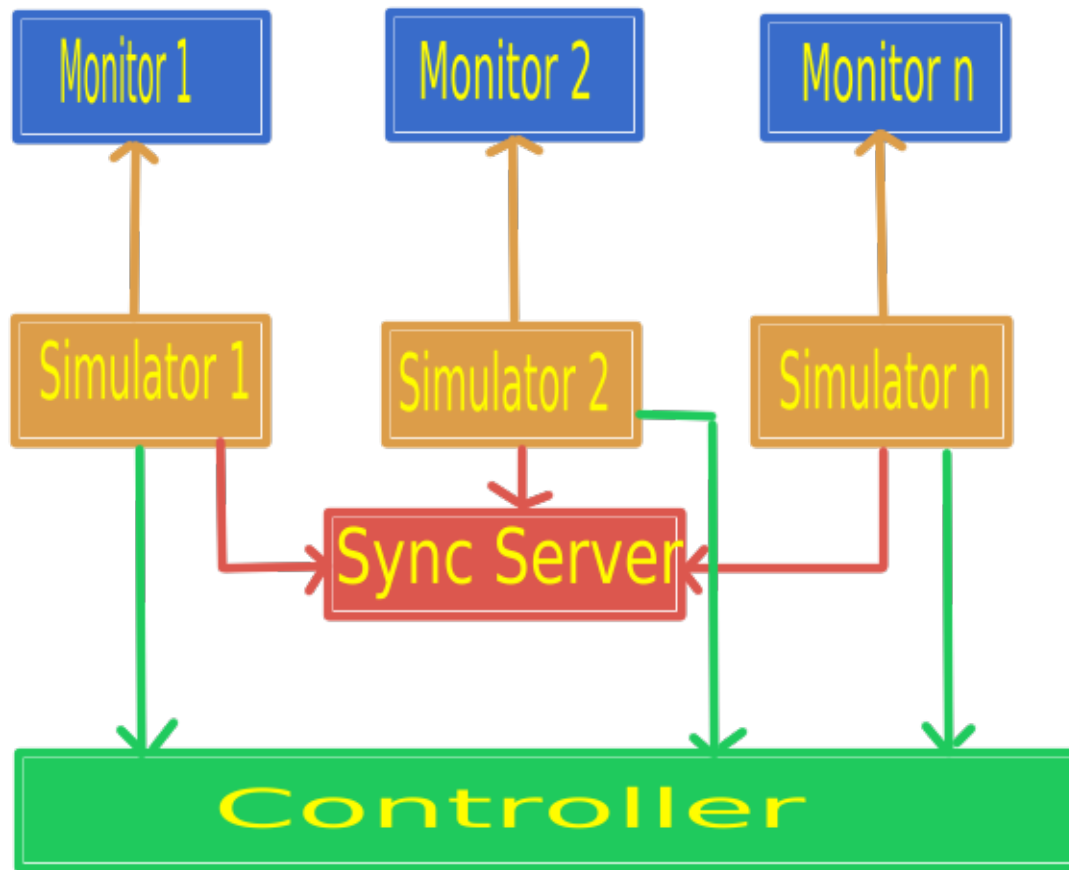


Fig. 2: Arhitectura aplicației Railsimulator

interfețele grafice, denumite **Monitor**, **Simulatoarele**, **Serverul de Sincronizare**, și **Controlerul**.

Fiecare din aceste componente este o aplicație de sine stătătoare iar numărul instanțelor pentru fiecare aplicație poate să fie oricât de mare. Ele pot fi rulate în locații diferite, comunicația între ele având loc prin intermediul protocolului TCP/IP.

După cum se poate observa în Fig. 2, există o legătură univocă între instanțele aplicației *Monitor* și instanțele aplicației *Simulator*. Astfel un singur *Simulator* este conectat doar la o singură interfață grafică. Toate instanțele simulatoarelor de rețea ferată sunt conectate prin TCP/IP la un *Server de Sincronizare* (Sync Server din Fig. 2). Acesta din urmă va asigura execuția simultană de către simulatoare a orarului specificat de către controler.

Tot din Fig. 1, observăm faptul că fiecare instanță de simulator este conectată la aceeași instanță de controler. Deși nu este necesar acest lucru, legarea tuturor simulatoarelor la un singur controler ne asigură că fiecare dintre aceste instanțe va prelucra același orar al trenurilor.

Arhitectura

În cazul în care proiectul *Railsimulator* nu a fost atașat acestei documentații, vă rog consultați pagina de internet <https://github.com/fioan89/railsimulator>

Tot aici se va putea găsi versiunea la zi a acestui sistem software.

Monitor

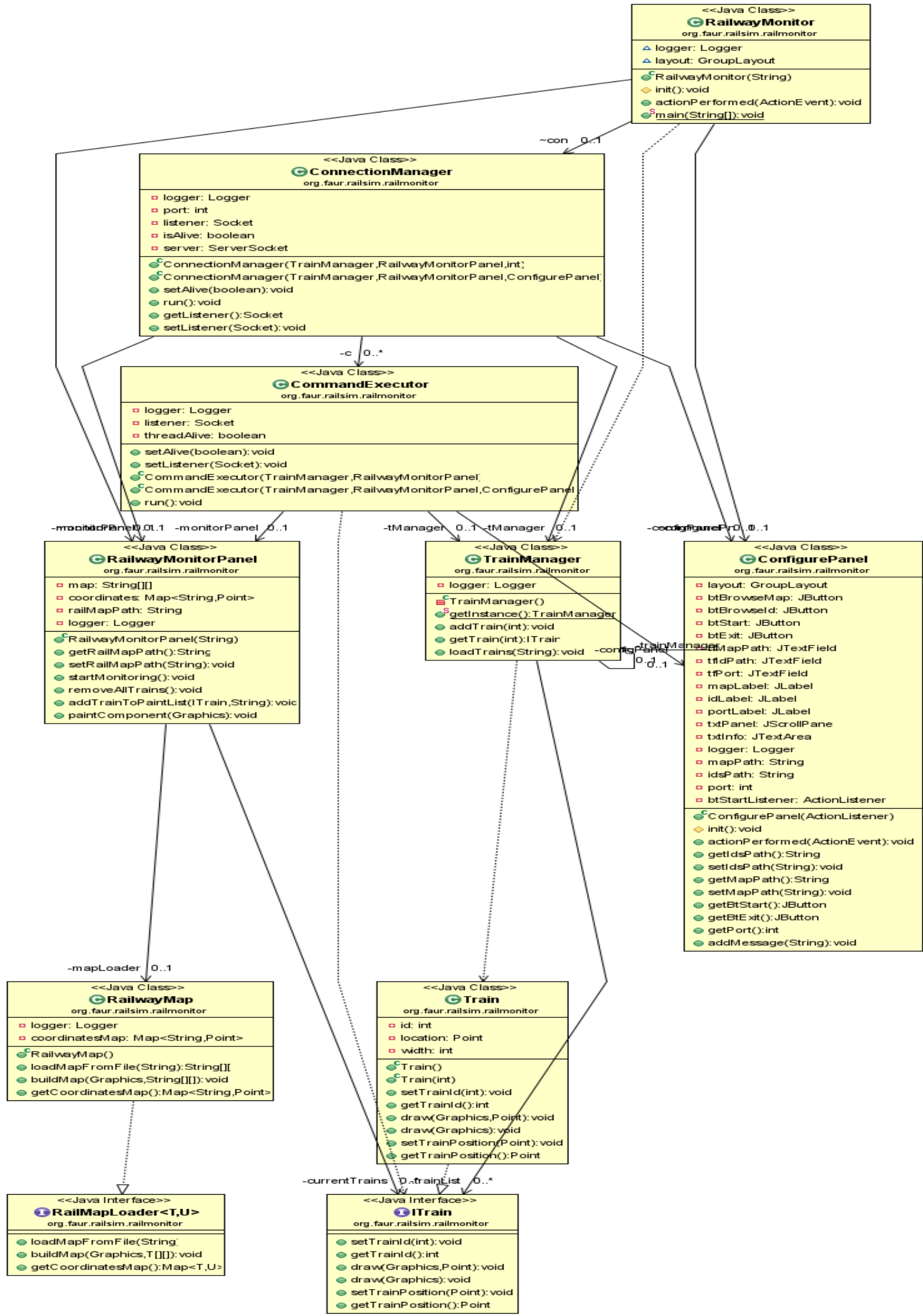
Aplicația Monitor este definită în pachetul *org.faur.railsim.railmonitor* din proiectul atașat acestei documentații. Rolul principal al acestui pachet software este de a simula grafic rețeaua feroviară. Această simulare cuprinde desenearea sistemului feroviar, primirea comenzilor de la simulator, interpretarea și randarea lor pe interfața grafică. Deasemenea *Monitor*-ul implementează un mecanism simplu prin care anunță utilizatorul de întârzieri-ile apărute în rețeaua feroviară.

Interfața grafică este foarte flexibilă dând-ui posibilitatea utilizatorului să încarce o hartă feroviară customizată dar și o listă cu trenurile care au voie să circule pe această hartă. Aceste opțiuni trebuie să fie date din fișiere în format *.csv*. Pentru mai multe detalii consultați fișierele *route1.csv* și *ids.csv* din directorul *simfiles* aflat în directorul cu documentația curentă. Atenție, pentru a vizualiza corect fisierul *.csv* trebuie ca editorul care deschide fișierul să folosească encoding-ul *UTF-8* iar ca și line spacing caracterul spațiu.

Comanda prin care se pornește o instanță a monitorului este:

```
java -cp ../target/classes:/target/classes/commons-cli-1.2.jar org.faur.railsim.railsimulator.RailwayMonitor
```

Important de notificat este faptul că interfața grafică trebuie să fie instanțiată înaintea simulatorului.



Simulator

Aplicația se găsește în pachetul *org.faur.railsim.railsimulator* și este responsabilă cu interpretarea tabelului primit de la controler. Acest tabel are o formă destul de simplă, pe prima linie fiind reprezentate toate locațiile prezente pe harta feroviară generală. Pe următoarele linii sunt introduse id-urile trenurilor care se află la un anumit moment pe locația respectivă.

Un astfel de tabel are forma:

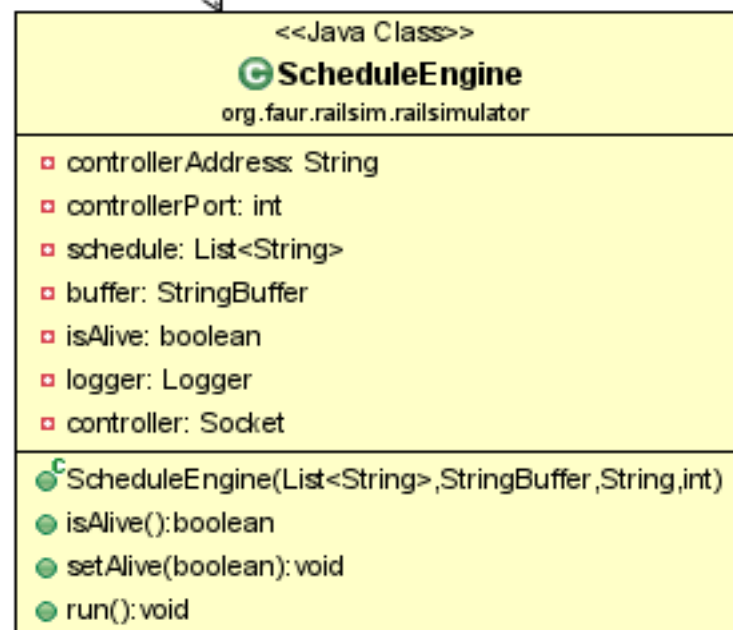
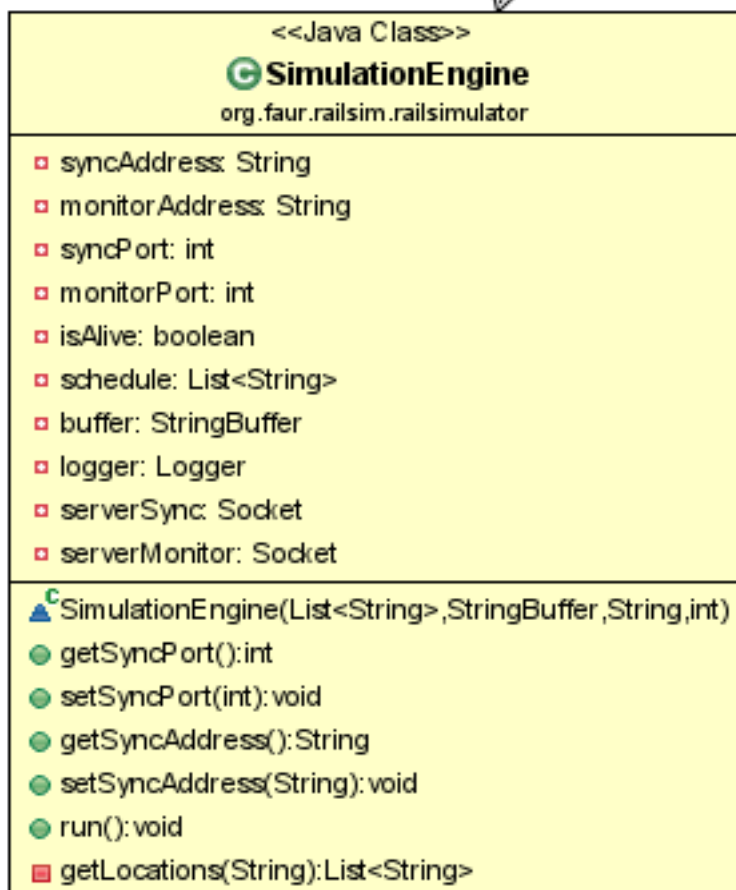
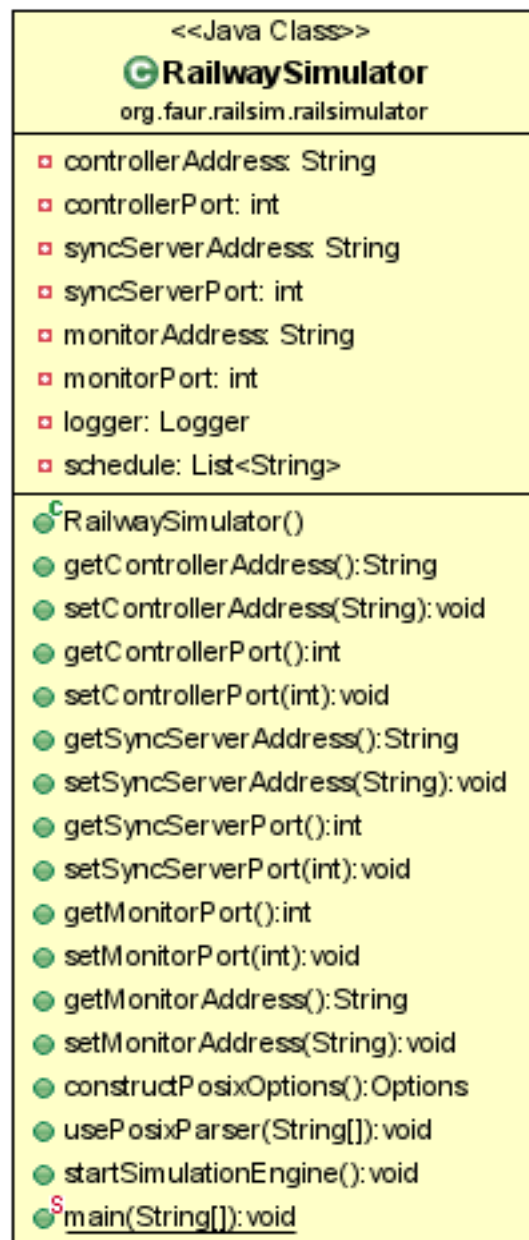
```
P1 I4 I5 I6 I7 I8 I9 P5 P4 I1 I2 I3 P6 I1P3
1 # # # # # # # # # # # # # # #
1 # # # # # # # 2 # # # # # # #
1 # # # # # # # 2 # # # # # # #
1 # # # # # # # 2 # # # # # # #
1 # # # # # # # 2 # # # # # # #
# 1 # # # # # # 2 # # # # # # #
```

După primirea tabelului simulatorul se conectează la interfața grafică și va începe să transmită comenzi. Aceste comenzi sunt deduse din tabel, spre exemplu pe linia 2, simulatorul va instrui interfața grafică să deseneze trenul cu id-ul 1 și trenul cu id-ul 2 în locațiile P1 respectiv P4, timp de o unitate de timp. Aceste comenzi nu sunt transmise la un moment de timp aleator, ci doar atunci când serverul de sincronizare la care simulatorul este conectat trimite semnalul de tact. Acest semnal de tact este trimis la o anumită unitate de timp specificate de user atunci când pornește serverul de sincronizare.

Comanda prin care se pornește o instanță a simulatorului este:

```
java -cp ../target/classes:../target/classes/commons-cli-1.2.jar org.faur.railsim.railsimulator.RailwaySimulator
-controllerPort 16000 -controllerAddress localhost -syncServerPort 10000 -syncServerAddress localhost
-monitorPort 5000 -monitorAddress localhost
```

Pentru a vedea lista de parametrii vă rugăm consultați documentația codului sursă. Important de notificat este că simulatorul este pornit doar după ce interfața grafică, serverul de sincronizare și controlerul au fost instanțiați.



Controler

Controler-ul se găsește în pachetul *org.faur.railsim.centralcontroller* și este cel mai complicat sistem software din întregul sistem *Railsimulator*. Asta se datorează algoritmilor necesari pentru crearea tabelului doar dintr-o lista de rute, modificarea lui atunci când utilizatorul o cere și trimiterea către simulator a orar-ului de fiecare dată când a fost modificat.

Primele operații pe care le face controler-ul sunt citirea rutelor stocat într-un fișier *.csv* (vezi *routes.csv*) în următoarea formă:

```
idTren
P1  I4  I5  I6  I7  I8  I9  P5
[5,7] [3,5] [2,4] [1,4] [2,3] [1,4] [1,2] [4,5]
idTren
P4  I1  I2  I3  I6  I7  I8  I9  P5
[5,8] [3,5] [2,4] [2,3] [1,5] [1,4] [1,3] [3,4] [4,6]
```

După citirea tabelului are loc algoritmul de rutare care stabilește fiecărui tren în parte ora la care pleacă din gară în funcție de ruta stabilită și timpii necesari pentru parcurgerea fiecărei locații. La sfârșitul operației de rutare se formează un tabel (vezi Simulator) care este trimis fiecărui simulator/client în parte.

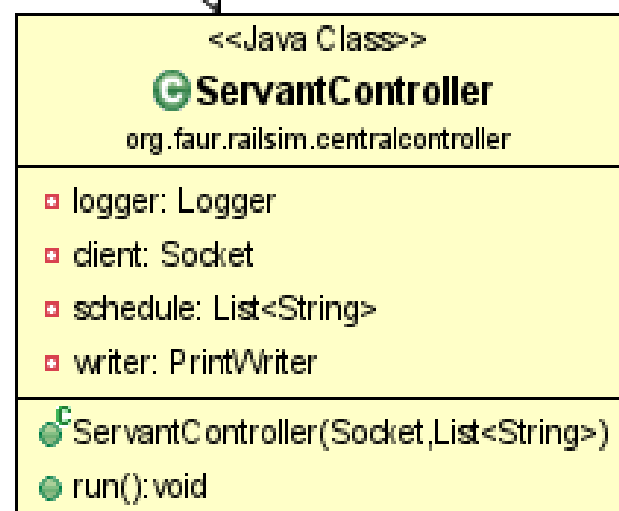
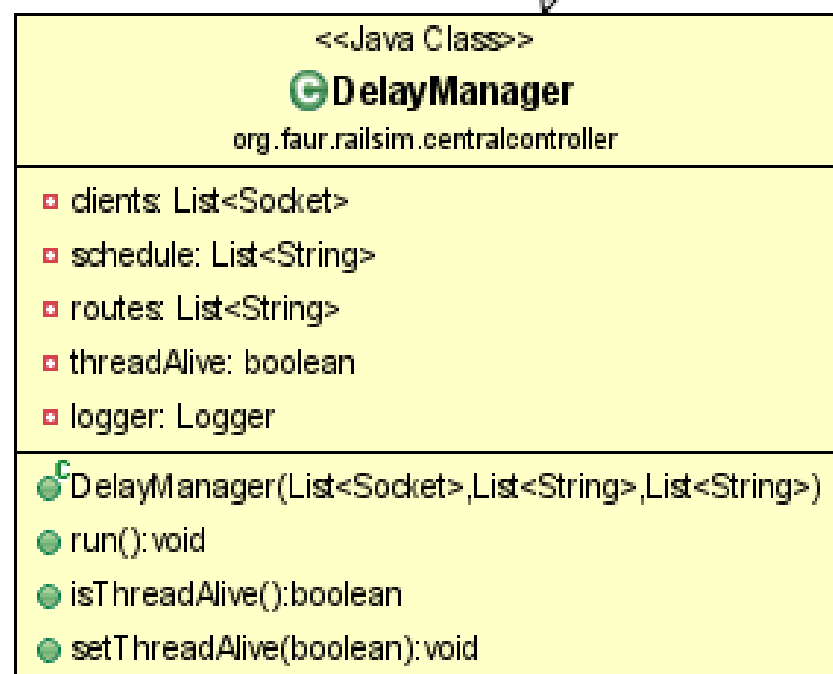
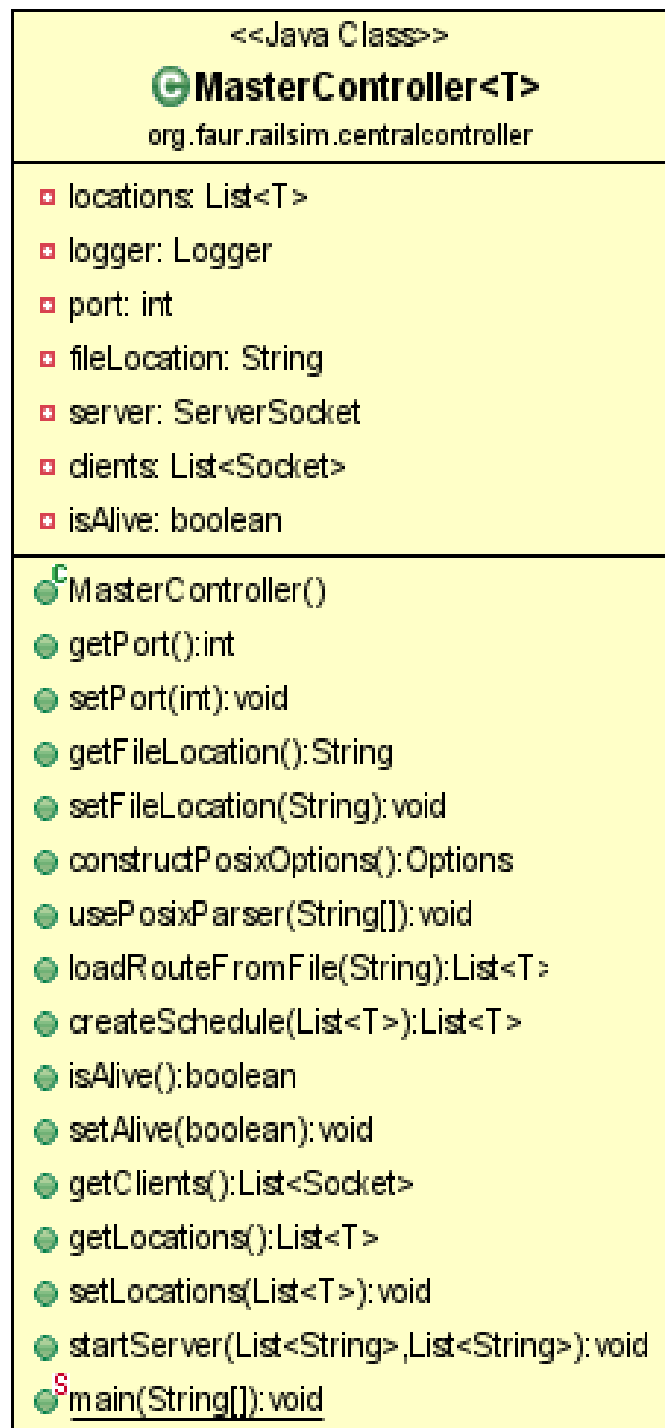
În urmatorul pas pe care controlerul îl execută, se verifică dacă trenurile au întârziat într-o anumită locație. Dacă se întâmplă acest lucru tabelul este refăcut și retransmis clienților. Întârzierea este introdusă de la tastatură printr-o comandă care are forma: *delay <trainId> in <location> with <x>*

Comanda prin care se pornește o instanță a controlerului este:

```
java -cp ../target/classes:../target/classes/commons-cli-1.2.jar
```

```
org.faur.railsim.centralcontroller.MasterController -port 16000 -file /patjh/to/routes.csv
```

Pentru a vedea lista de parametrii vă rugăm consultați documentația codului sursă. Important de reținut este că, controlerul, este pornit înaintea simulatorului.



Server sincronizare

Această aplicație se găsește în pachetul *org.faur.railsim.serversynchronizer* iar scopul ei este de a sincroniza clienții conectați la acest server. Fără acest server care să coordoneze operațiile, clienții (vezi Simulator) ar lucra haotic, iar noțiunea de simulator nu s-ar mai potrivi întregului sistem.

În general pașii pe care-i face server-ul de sincronizare sunt destul de simpli:

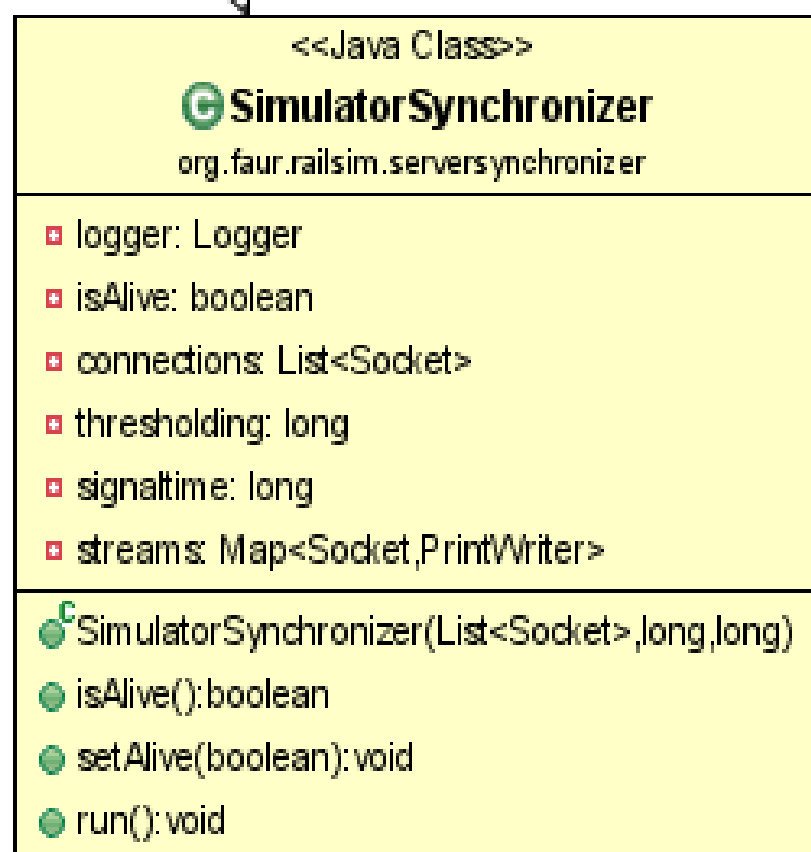
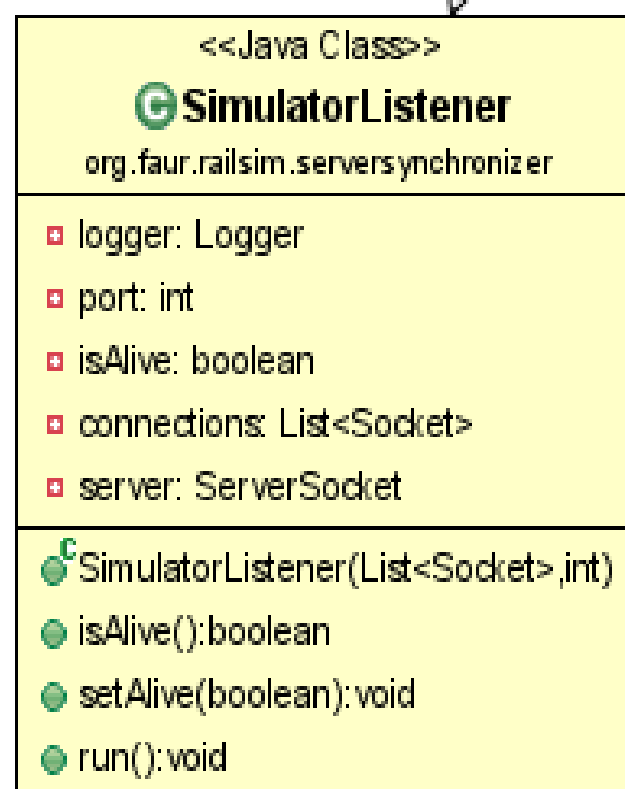
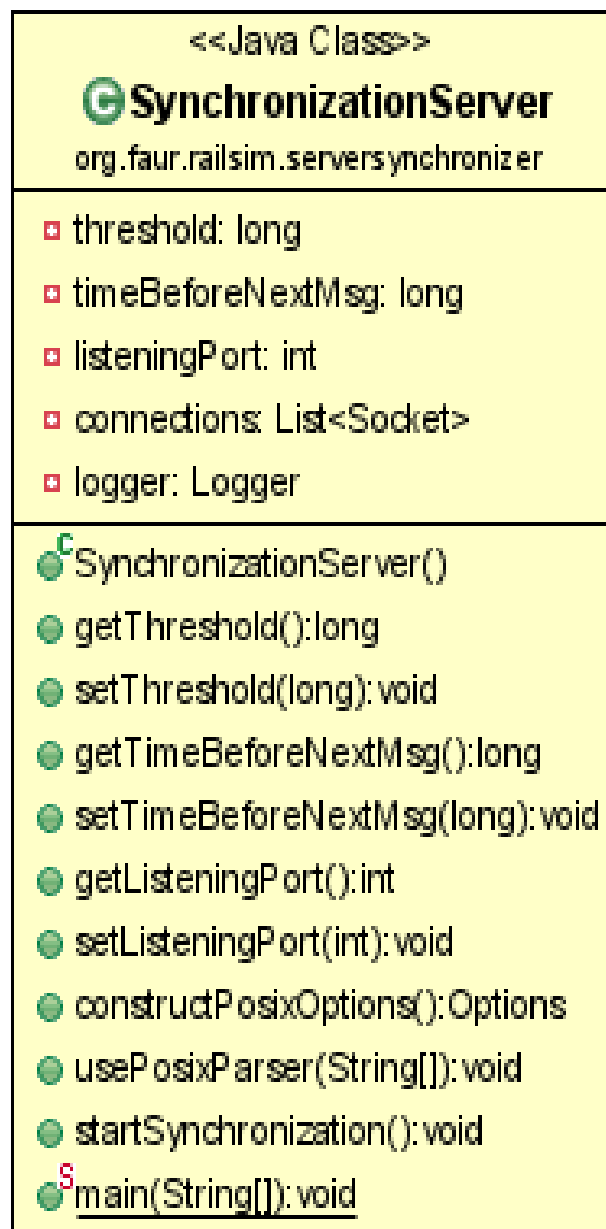
- 1. pornește cu un contor de la valoarea 1. Acest contor reprezintă o unitate de timp.*
- 2. trimite clienților valoare contorului.*
- 3. oprește firul un anumit număr de milisecunde specificat de către utilizator la pornirea serverului.*
- 4. incrementează contorul.*
- 5. dacă contorul a trecut peste o anumită valoare prag specificată de către utilizator la pornirea serverului, atunci resetează contorul la valoarea 1.*
- 6. revino la pasul 2.*

Comanda prin care se pornește o instanță a controlerului este:

```
java -cp ../target/classes:./target/classes/commons-cli-1.2.jar
```

```
org.faur.railsim.serversynchronizer.SynchronizationServer -port 10000 -threshold 10000 -signaltime 1500
```

Pentru a vedea lista de parametrii vă rugăm consultați documentația codului sursă. Important de reținut este că, serverul de sincronizare este pornit înaintea simulatorului.



Concluzii

Railsimulator prin arhitectura sa distribuită reușește să îndeplinească destul de bine cerințele enumerate la începutul acestui document, în secțiunea *Abstract* și *Introducere*. Ba mai mult, prin sistemul modular folosit, Railsimulator poate fi foarte ușor modificat și adaptat, unor noi cerințe sau de ce nu, înlocuirea anumitor componente cu altele care fac același lucru doar într-un mod mai eficient. Bineînțeles respectând protocolul impus în acest sistem.

Pentru o mai bună înțelegere a întregului sistem și a algoritmilor folosiți vă rugăm consultați codul sursă, împreună cu documentația acestuia.

TO DO

- În momentul când controlerul identifică o întârziere, un nou tabel este creat și trimis simulatorului. După primirea lui, simulator-ul se ocupă imediat de execuția lui din momentul specificat de serverul de sincronizare. Ar trebui ca simulator-ul să-și termine execuția vechiului tabel și doar după aceea să pună în aplicare noul tabel.
- Când monitorul primește o înștiințare prin care este anunțat că un tren este întârziat , acesta afișează un mesaj pe interfața grafică. Ar fi frumos ca trenul respectiv care este întârziat să fie desenat cu o alta culoare.
- Controlerul ar trebui să citească întârziier-ile de la un senzor. Momentan se face direct de la tastatură. Acest senzor ar trebui să fie un server care să comunice prin TCP/IP cu controler-ul.

