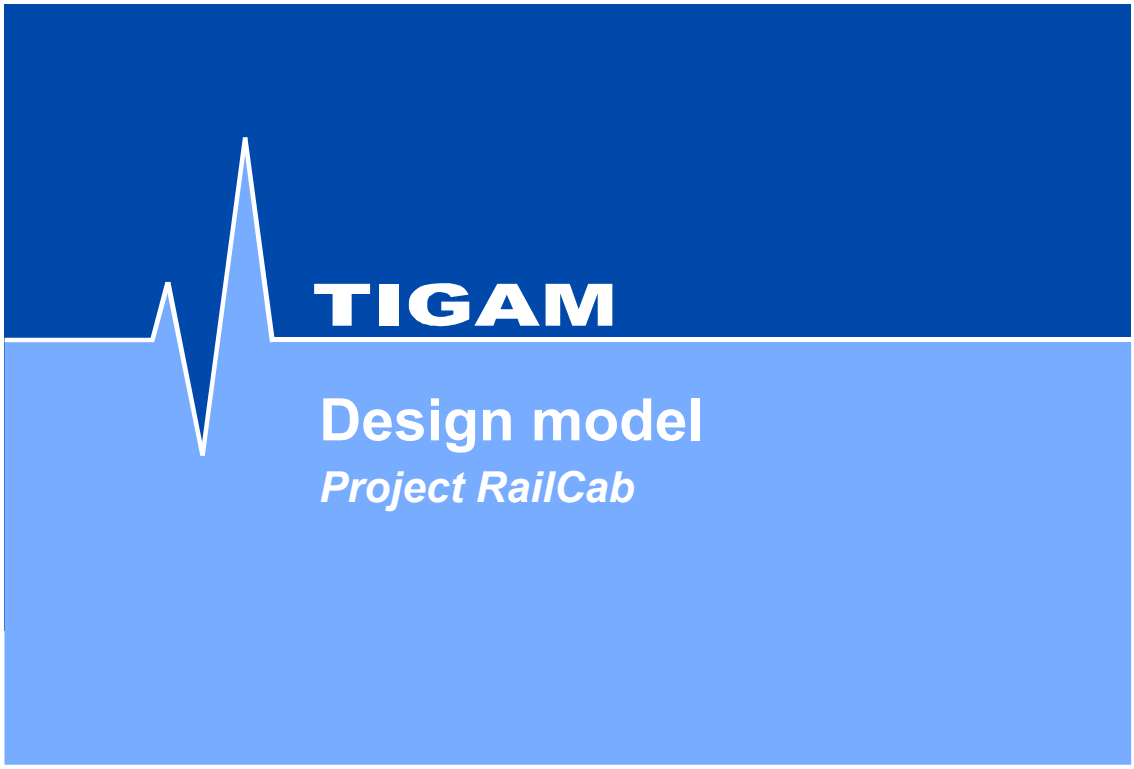


Design model

Project RailCab

Auteur(s): Team 7
Raymond van der Rots
Fabian van den IJssel
Maarten Bonsema
Martijn de Jongh
Erik Hoogland
Yorick Groot

Datum: 21 november 2007

TIGAM is onderdeel van de
Hogeschool van Amsterdam.

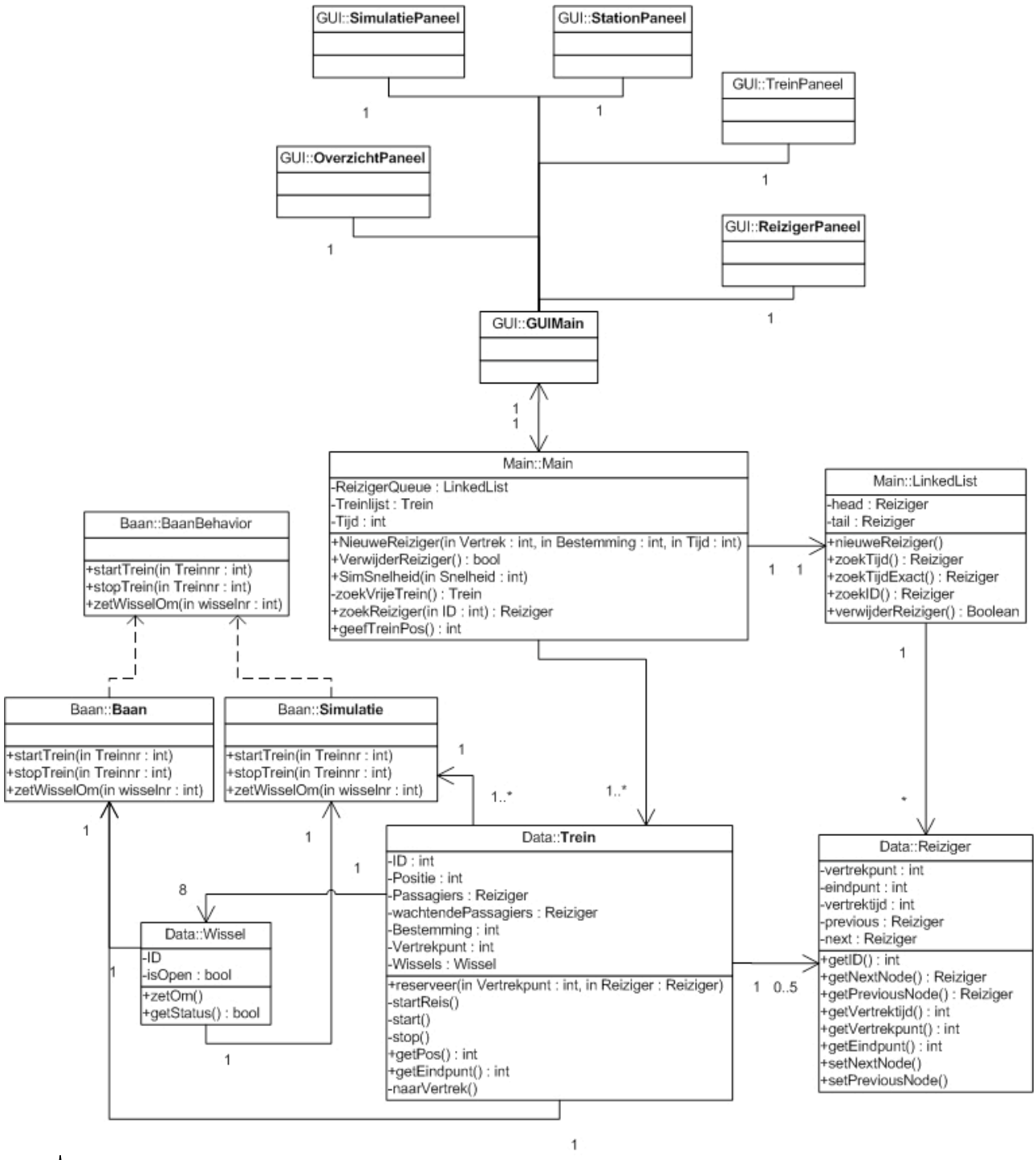
Inhoudsopgave

1. Systeem design

In dit eerste hoofdstuk wordt het systeem nader toegelicht aan de hand van het klassendiagram en een sequence diagram.

1.1 Overzicht

Hieronder wordt het klassendiagram van het systeem weergegeven.



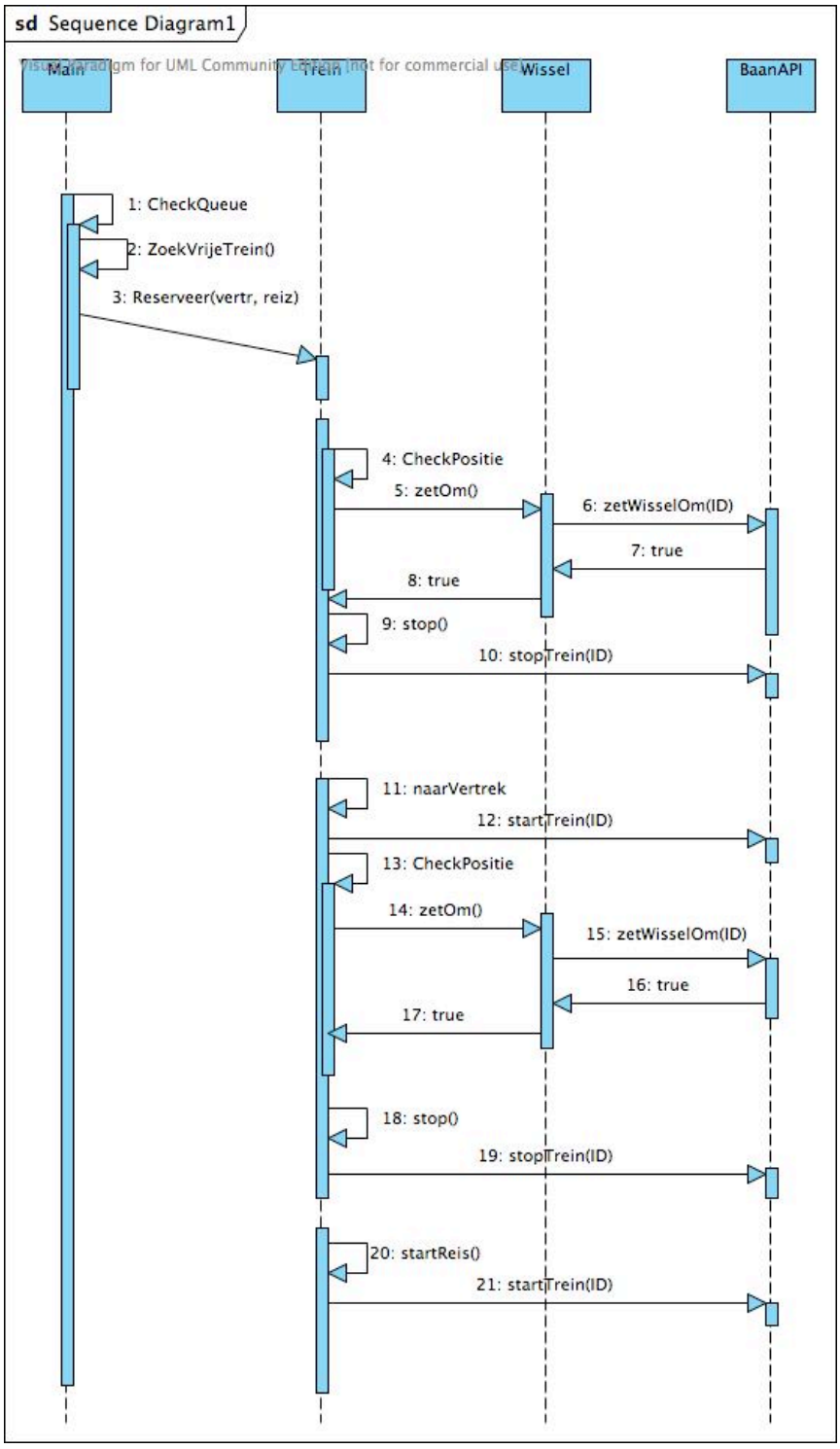
In het gui package wordt de grafische interface bevat. Elke klasse stelt een verschillend paneel voor die allemaal in hetzelfde venster weergegeven worden. Om de statistieken efficiënt en snel bij te kunnen houden hebben wij besloten om enkele relevante data in de guipackage te plaatsen. Telkens als er een passagier wordt toegevoegd zal zijn ID en vertrektijd opgeslagen worden in een hashmap. Als de reiziger in een trein stapt wordt zijn key verwijderd.
De weergave van de treinen op het simulatiepaneel zal geschieden door het pollen van de main methode. Deze geeft dan de positie van zijn treinen door.
Tenslotte wordt er een logboek bijgehouden (als bestand) waarin alle reizigers worden opgeslagen die in het systeem hebben gezeten. Zo kan de data naderhand nog bekeken worden en weergegeven worden.

In de main package zitten twee klassen. De linkedlist klasse dient ter hulpmiddel van de main klasse. De main klasse zorgt voor de allocatie van treinen en het beheren van de data in het systeem. Alle reizigers worden in een LinkedList opgeslagen en gesorteerd op vertrektijd. Dit geschiedt bij het toevoegen van een reiziger. De treinlijst is een array die alle treinen in het systeem bevat. Om de simulatiesnelheid aan te kunnen passen is er gekozen voor een timer die de integer tijd verhoogt. De periode van de timer kan aangepast worden waardoor het systeem dan vervolgens sneller zal uitvoeren.

De datapackage bevat drie klassen. De wissel klasse heeft slechts tot doel een wissel te kunnen omzetten en zijn status bij te houden. Hij heeft een directe interface tot de baan (of de baan simulaite; Simulatie). Elk object van de klasse trein stelt een trein in het systeem voor. De treinen rijden zelfstandig naar hun eindbestemming en worden daarin niet aangestuurd.
Tenslotte is er de klasse reiziger. Elk object hiervan stelt een reiziger voor in het systeem met een uniek ID.

1.2 Sequence diagram

Om het systeem verder toe te lichten wordt hieronder de normale werking van het systeem geïllustreerd met behulp van een sequence diagram. Deze acties vinden plaats als er reizigers in de queue zitten en als er verder geen invloed wordt uitgeoefend op het systeem.

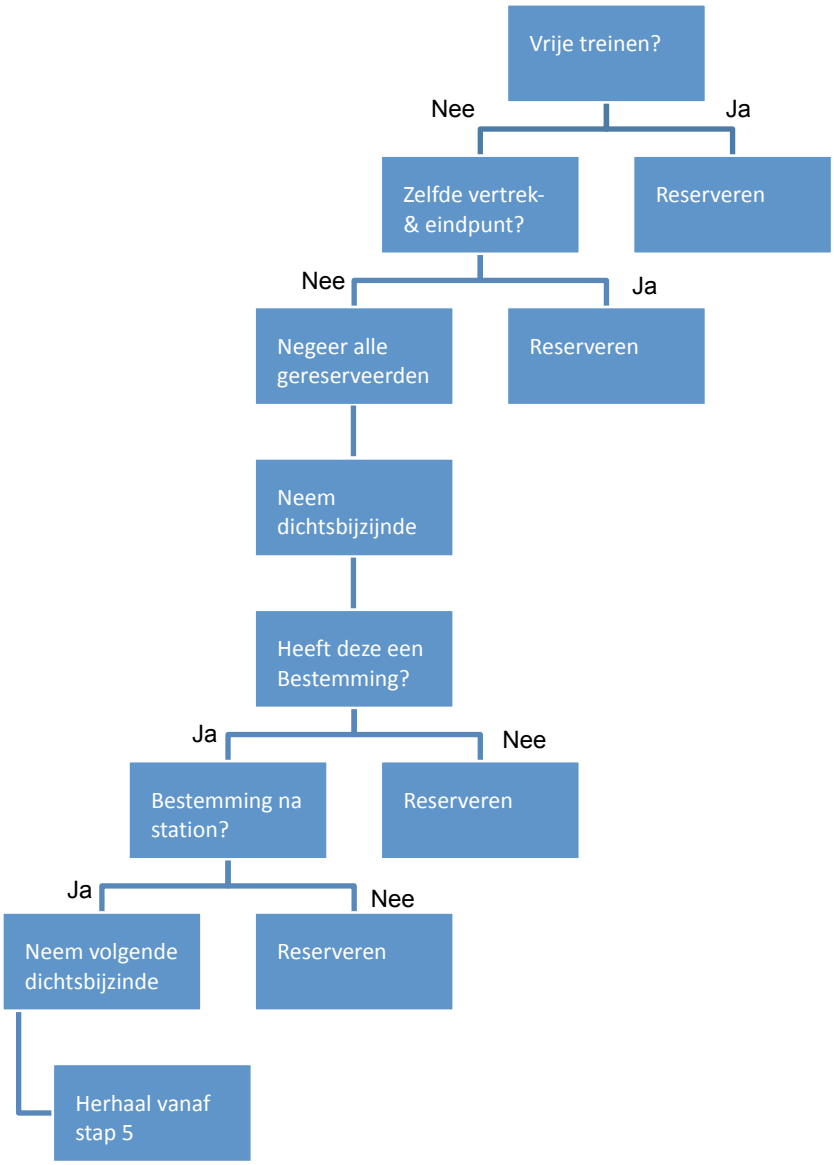


Main controleert het bovenste element van de reizigersqueue op zijn vertrektijd (Stap 1). Als deze nadert dan zal hij een trein moeten reserveren om de passagier weg te brengen.
Main moet daartoe eerst een beschikbare trein vinden (stap 2) die bovendien zo dicht mogelijk bij het station is. Alle treinen staan op volgorde van toevoegen in de treinlijst.

1. Eerst zal er gezocht worden naar vrije treinen (treinen zonder bestemming) zodat de passagier zo veel mogelijk comfort heeft.
2. Als er geen vrije treinen gevonden zijn zal er gezocht worden naar treinen die gereserveerd zijn voor passagiers met dezelfde begin- en eindbestemming.
3. Als deze ook niet gevonden zijn dan zullen alle gereserveerde treinen uitgezonderd worden.
4. Van de overige treinen zal hij de posities controleren om te kijken welke trein zich zo dicht mogelijk bij zijn station bevindt. Als deze gevonden is zal er gekeken worden of deze al een bestemming heeft. Als er een bestemming is zal deze ook gecontroleerd worden om te kijken of de trein niet voorbij het station rijdt. Als dit het geval is dan zal de tweede treinkeuze gecontroleerd worden. Mochten alle treintjes een bestemming hebben die na het vertrekstation liggen dan zal de eerste treinkeuze gereserveerd worden.

Uiteraard wordt er voor het reserveren bij elke stap gecontroleerd of de trein niet vol is.

In schema:



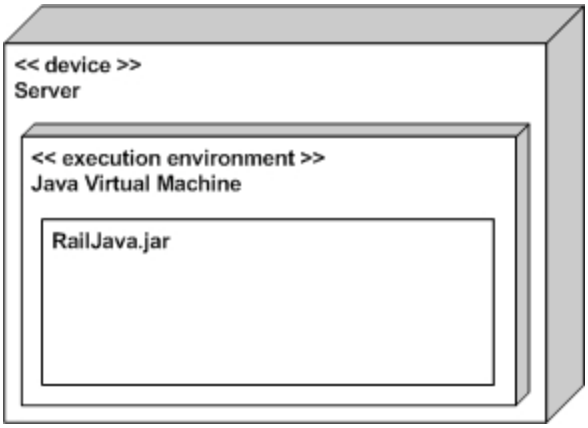
Bij het reserveren (stap 3) zal de methode reserveer(vertrekpunt, reizigers) worden aangeroepen. Hierbij wordt de passagier als wachtendePassagiers meegegeven en zijn vertrekpunt zal opgeslagen worden in het vertrekpunt van de trein. De passagier wordt verwijderd uit de queue. In het geval van het sequence diagram had de desbetreffende trein nog een reis die hij eerst moet afmaken. Bij het reizen controleert hij bij elke wijziging van zijn positie of hij vlak voor zijn doelstation is. Als hij vlak voor het station is zal hij zo nodig de betreffende wissel omzetten om het station binnen te rijden. Bij het omzetten van de wissel wordt teruggeven of het omzetten succesvol is geweest. Als de trein op het station is aangekomen dan zal hij stoppen en zijn passagiers verwijderen uit het systeem (deze worden wel in het logboek opgeslagen).

Als de trein zijn huidige reis heeft voltooid of als de trein geen huidige reis had zal hij zijn eigen methode naarVertrek() aanroepen. Hierbij zal hij naar zijn vertrekpunt rijden om zijn passagiers op te pikken. Zijn bestemming wordt gelijk gesteld aan zijn vertrekpunt. Als hij bij zijn vertrekpunt is aangekomen roept hij zijn eigen methode

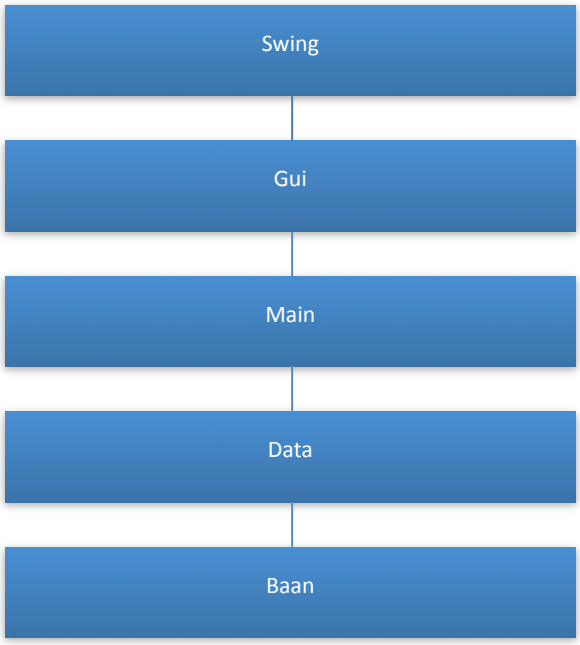
startReis() aan. Hierbij wordt zijn vertrekpunt gewist en worden de wachtendePassagiers in Passagiers gezet. De trein kan nu dus weer gereserveerd worden.

2. Deployment

Het systeem is ontworpen om op een enkele computer uitgevoerd te worden omwille de gebruiksvriendelijkheid en gemak van installatie. Het is dus een single-tier systeem. Het bestaat uit een uitvoerbaar JAR bestand en enkele logboeken om data in bij te houden en terug te lezen.



Het programma bestaat uit enkele lagen omwille van de modulariteit. Elke laag kan intern aangepast zonder dat dit effect heeft de lagen erboven of eronder.



Bovenaan bevind zich de ingebouwde Swing laag van Java om de interface te tekenen. De GUI voert daar zijn acties naartoe en slaat enkele data op. De main laag bevat het algoritme om de treinen te alloceren. Deze treinen zitten samen met de reizigers en de wissels in de data laag. Tenslotte is er de baan laag die de interface vormt naar de uiteindelijke hardwarematige baan.