Programmeerproject 2: Specificatie: Fase 2

Maxim Brabants

0576581

Maxim.Lino.Brabants@vub.be

Academiejaar 2021-2022

Inhoud

1 Beschrijving componenten

1.1 Trein-ADT

Dit type maak ik aan zodat ik later in mijn programma op elk gewenst moment alle treinen die op het spoor aanwezig zijn ter beschikking heb in de vorm van verpakte objectjes waarbij elk objectje van dit type beschikt over zijn unieke identificatienummer, de huidige richting naar waar de trein rijdt, het segment waarin hij start en ten slotte de momentele snelheid die de trein aanneemt. Daarbij zijn er ook nog twee belangrijke operaties die nodig zijn.

Naam	Signatuur
maak-trein	(symbol symbol symbol → Trein)
huidige-snelheid	(Ø → number)
verander-snelheid!	(number → Ø)

- Om een instantie te kunnen maken van dit type dient een uniek identificatienummer, de richting en het startsegment meegegeven te worden.
- Om de huidige snelheid van de trein weer te geven maak ik een operatie die de procedure van interface.rkt oproept om de snelheid van de locomotief op te vragen
- Analoog aan verander-snelheid! roepen we de operatie op de interface.rkt om de snelheid van de huidige trein te veranderen naar een nieuwe snelheid.

1.2 Wissel-ADT

Als we gaan kijken naar de wissels die zich op verschillende plekken op het spoor kunnen bevinden, dan moet het mogelijk kunnen zijn om elk van deze wissels individueel te verzetten en de huidige stand op te vragen. Hiervoor hebben we ook twee operaties voorzien die respectievelijk dit doen.

Naam	Signatuur
maak-wissel	(symbol symbol → Wissel)
huidige-stand	(Ø → number)
verander-stand!	(number → Ø)

1.3 Treinreeks-ADT

Om alle instanties van het 'Trein'-type bij te kunnen houden doorheen ons programma, maken we een extra type aan die een oneindige reeks van treintypen kan bijhouden. Op die manier kunnen we ongeacht op welk moment een gewenste operatie uitvoeren op een bepaalde trein.

Naam	Signatuur
maak-treinreeks	(Ø → Treinreeks)
voeg-trein-toe!	(Trein → Ø)
verwijder-trein!	(Trein → Ø)
snelheid-trein	(symbol → number)
wijzig-snelheid-trein!	(symbol number → Ø)

- Om zo een treinreeks aan te maken roepen we **maak-treinreeks** op en verkrijgen we daaruit een soort van lijstabstractie waaraan we later instanties van het type 'Trein' kunnen toevoegen.

- **voeg-trein-toe!** const dan zo'n type 'Trein' aan de reeds bestaande treinreeks. Je geeft een instantie van het type 'Trein' mee en die wordt vervolgens toegevoegd. Daarbij wordt de trein ook onmiddellijk op het spoor gezet. (add-loco)

- **verwijder-trein!** doet het omgekeerde en haalt de meegegeven trein uit de reeks alsook van het spoor. (remove-loco)

- Om de snelheid van een bepaalde locomotief in de reeks te verkrijgen gebruik je de operatie '**snelheid-trein'** die de snelheid van een aanwezige locomotief teruggeeft. (get-loco-speed)

- Je geeft het identificatienummer van een bepaalde trein en zijn nieuwe snelheid mee aan **wijzig-snelheid-trein!** om de huidige snelheid van die bepaalde trein te veranderen naar de nieuwe meegegeven snelheid. (set-loco-speed!)

- **detectieblok-trein** geeft voor een gegeven trein terug of hij zich al dan niet in een detectieblok bevindt. Zo ja, dan wordt het identificatienummer van dat detectieblok teruggegeven.

1.4 Spoornetwerk-ADT

Van dit type maken we slechts één instantie aan, aangezien we ons focussen op één spooropstelling tegelijk. Deze houdt alle informatie bij die eigen is aan een spooropstelling zoals alle wissels en detectieblokken die behoren tot een bepaalde spooropstelling alsook de treinen die erop rijden, maar initieel bij de aanmaak van een spoornetwerk rijden er logischerwijs nog geen treinen op. Deze kunnen later natuurlijk nog wel toegevoegd worden aan het spoor. We houden in dit type alle wissel –en detectiebloknummers bij (get-switch-ids en get-detection-block-ids)

Naam	Signatuur
maak-spoornetwerk	(Ø → Spoornetwerk)
voeg-wissel-toe!	(Wissel→ Ø)
voeg-nieuwe-trein-toe!	(Trein → Ø)
wijzig-stand-switch!	(symbol number → Ø)

- Per keer dat we een spooropstelling bekijken, wordt **maak-spoornetwerk** aangeroepen.
- We houden ook nog een lijst bij met effectieve wissels die aangemaakt werden door 'maak-wissel'. Bij oproep van **voeg-wissel-toe!** wordt een nieuwe wissel hieraan toegevoegd.
- Zoals eerder werd vermeld houden we ook alle treinen bij die op het spoor rijden. Natuurlijk kunnen er in het begin nog geen treinen aanwezig zijn op het spoor. We houden hiervoor een instantie bij van 'maak-treinreeks' waarbij de lijst met treinen initieel leeg is.
- **wijzig-stand-switch!** verandert de stand van een switch met meegegeven id naar de meegegeven stand.

1.5 INFRABEL

Dit type/ADT is feitelijk een beetje een extra abstractie bovenop het spoornetwerk. Het wordt gebruikt als tussencomponent om onderdelen van het spoornetwerk te manipuleren en/of op te vragen.

Naam	Signatuur
maak-programma	(Ø → Infrabel)
start-programma	(lambda() (void) → Ø)
zet-trein-op-spoor	(string string string → Ø)
verhoog-snelheid-trein!	(symbol → Ø)
verlaag-snelheid-trein!	(symbol → Ø)
geef-snelheid-trein	(symbol → number)
geef-wissel-ids	(Ø → List <symbol>)
geef-detectieblok-ids	(Ø → List <symbol>)
verander-wisselstand!	symbol number → Ø)

- **maak-programma** heeft een lokale staat dat het spoor voorstelt. Die wordt geinitialiseerd door 'start-programma'.

- **start-programma!** verwacht één van de setup functies die inbegrepen zijn bij de simulator en stelt deze vervolgens intern in. Ook instantiëren we de spoorvariabele met een aanroep van 'maak-spoornetwerk'.

- **zet-trein-op-spoor!** delegeert door naar de 'voeg-nieuwe-trein-toe!' van het Spoornetwerk-ADT.

- **verhoog-snelheid-trein!** delegeert door naar de 'wijzig-snelheid-trein!' van het Spoornetwerk-ADT.

- **verlaag-snelheid-trein!** delegeert door naar de 'wijzig-snelheid-trein!' van het Spoornetwerk-ADT.

- **geef-snelheid-trein** delegeert door naar de 'snelheid-trein' van het Spoornetwerk-ADT.

- **geef-wissel-ids** vraagt de wissel-ids aan het Spoornetwerk-ADT.

- **geef-detectieblok-ids** vraagt de detectieblok-ids aan het Spoornetwerk-ADT.

- **verander-wisselstand!** vraagt aan spoornetwerk om stand van wissel te veranderen.

1.6 NMBS

Deze component draait op onze computer en vanuit onze GUI kunnen we deze gaan aanspreken. Het bevat alle operaties die betrekking hebben tot het beheren van treinen, trajectberekening, etc. Andere operaties die betrekking hebben op een verandering van de spoorsituatie of dergelijke ,delegeren we door naar INFRABEL.

Naam	Signatuur
maak-programma	(Ø → Infrabel)
start-programma	(lambda() (void) → Ø)
zet-trein-op-spoor	(string string string → Ø)
verhoog-snelheid-trein!	(symbol → Ø)
bereken-traject	(symbol symbol → Ø)

bereken-traject moet eerst een pad berekenen waarna deze het verkregen pad doorstuurt naar INFRABEL die het pad zal laten uitvoeren.

1.7 Graphical User Interface

Tot slot hebben we een user interface gemaakt die ons toelaat onze functionaliteit te testen op enkele opstellingen in de simulator. Hier krijgen we de simulator met de gekozen spooropstelling te zien en verschijnt ons GUI-venster waarmee we operaties kunnen gaan uitvoeren met betrekking tot die opstelling. We kunnen ook nieuwe treinen toevoegen aan de gekozen opstelling en deze via de GUI manipuleren.

(meer hierover in de handleiding)

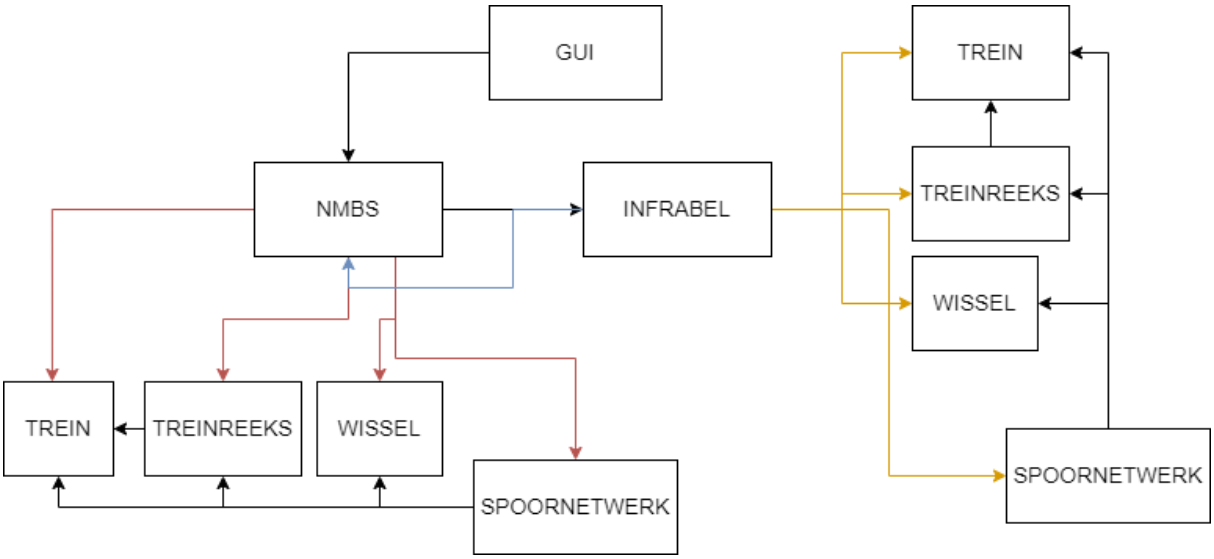
1.8 Opstelling parser

In de projectmap (op hetzelfde niveau als het GUI-bestand) is er ook nog een bestand 'opstelling-parser' te vinden. Om de overzichtelijkheid en duidelijkheid een beetje te bewaren, is dit bestand aangemaakt om alles te bundelen dat te maken heeft met het inlezen van een zelf gedefinieerde opstelling. Van zodra we van onze GUI de input voor de naam van het tekstbestand hebben opgevangen waarin de opstelling gedefinieerd staat, dan delegeren we direct door naar deze parser die het verdere werk zal afhandelen.

Hier staat namelijk een functie in die een filestream opent met het meegegeven bestand dat overeenkomt met de opgegeven bestandsnaam. Daarna volgt de procedure om effectief het bestand regel per regel uit te lezen. Dit gebeurt volgens een aantal voorwaarden die beschreven staan in de handleiding. De componenten en verbindingen tussen deze worden uiteindelijk meegegeven aan een functie in de simulator die het finale werk zal afronden om effectief de opstelling grafisch weer te geven.

2 Architectuur

Er zijn bepaalde types die niet correct gaan kunnen werken zonder het bestaan van andere. Er is dus een bepaalde afhankelijkheid die dient gerespecteerd te worden om een goede werking van het gehele programma te kunnen garanderen.



Op het eerste niveau hebben we het Trein-ADT dat één enkele trein voorstelt op de sporen. Dit is de zuiverste vorm van abstractie die mogelijk is om een trein als type voor te stellen. Daarboven creëren we een verzameling van Treintypes. We hebben op dit niveau een pijl getrokken naar onder met de simpele reden dat een treinreeks in het begin nog leeg is en we dus hier min of meer kunnen zeggen dat een treinreeks niet echt kan bestaan zonder dat er enige treinen bestaan. Ondanks het feit dat we toch 'maak-treinreeks' kunnen aanroepen zonder dat er eerst een trein bestaat, stelt een lege treinreeks niet echt veel voor in ons programma…

Het idee dat een spoornetwerk kan bestaan zonder treinen lijkt me nog wel logisch, omdat je sporen hebt, maar er niet per se treinen over hoeven te rijden. De pijl verklaart de omvattingsrelatie, het feit dat je een spoor kunt hebben dat treinen omvat maar niet per se noodzakelijk. Hetzelfde geldt hier voor een wissel. Een bepaalde sporenopstelling kan wissels omvatten/bevatten maar niet noodzakelijk, aangezien er ook opstellingen 'bestaan' die geen wissels hebben. (simulator)

NMBS is de component die het dichtst ligt bij de GUI en alle berichten of communicatie van de GUI zal ontvangen en in het geval nodig actie zal ondernemen (spoorsituatie veranderen). Deze staat in verbinding met INFRABEL.
Infrabel is in feite de component die de operaties doordelegeert naar de Command & Control van het spoornetwerk. De NMBS component communiceert met Infrabel om zo op een juiste manier de treinen en het spoor aan te sturen. De bedoeling is om een tussencomponent tussen NMBS en INFRABEL te voorzien op zo'n manier dat elk bericht van NMBS naar INFRABEL en omgekeerd kan verlopen via die tussencomponent. Later kan die tussencomponent vervangen worden door de netwerkcommunicatie. Deze tussencomponent is nog niet voorzien.