radiuscollege

“UML fundamentals”

Serviceprogramma

Periode 5

2e leerjaar AO en MED

Schooljaar 2015-2016

Versie 2.0

Inhoudsopgave

1 Inleiding

Herken je jezelf hierin:
“Ik ben een visueel ingesteld persoon. Grote lappen tekst doen het niet voor mij, die lees ik maar half. Maar plaatjes, en vooral architectuurplaatjes, die bestudeer ik.”
Dan wordt het voor jou tijd om de Unified Modeling Language (UML) te leren.
UML is een verzameling van methoden en technieken om je applicatie op een eenduidige manier te ontwerpen. UML is erop gericht je een beter beeld te geven van de functionaliteit van een te bouwen object georiënteerde applicatie. UML bestaat voornamelijk uit een aantal schematechnieken.

1.1 Wat kun je met UML?

Met UML maak je een blauwdruk van je applicatie, zodat precies bekend is wat de verschillende onderdelen moeten doen en hoe de relaties tussen deze onderdelen liggen. De techniek leent zich dan ook zeer goed voor een objectgeoriënteerde aanpak. Wanneer je in teamverband aan een project werkt, kun je met UML schema's ook goed het werk verdelen. UML is een modelleertaal die we kunnen gebruiken om situaties uit te schrijven of te tekenen alvorens we beginnen te programmeren. UML zorgt er voor dat er eerst over het probleem moet worden nagedacht voordat we beginnen met het schrijven van de code. UML zorgt er ook voor dat anderen een bepaald proces kunnen verstaan zonder dat kennis van enige programmeertaal nodig is.

Met UML kunnen niet alleen beschrijvingen worden gemaakt van statische verschijnselen, maar ook van dynamische processen. Het dient vooral als een veelzijdig te gebruiken instrument dat in verschillende fasen van de systeembouw kan worden toegepast. Een van de krachtige aspecten van UML is dat er op relatief eenvoudige wijze meta-beschrijvingen kunnen worden gemaakt.
In tegenstelling tot wat vaak wordt gedacht, is UML zelf geen methode, maar een notatiewijze die bij verschillende methodes kan worden gebruikt.

UML dient ook voor het documenteren van het programma. Nieuwe programmeurs moeten zich op die manier niet eerst door alle code worstelen om het hele programma te begrijpen. Ze kunnen de UML documenten doornemen om zo te begrijpen hoe de applicatie is opgebouwd.

1.2 Wat gaan we doen tijdens dit serviceprogramma?

Tijdens dit serviceprogramma UML richt je je op de basisconcepten van UML. In het serviceprogramma komen de onderdelen van UML aan bod die nodig zijn om de structuur en het gedrag van een systeem te beschrijven. Welke uitgangspunten komen aan bod bij deze vorm van ontwikkeling? Hoe pas je deze eigenlijk toe? Binnen het serviceprogramma gebruiken we verschillende UML schematechnieken om een algemene functionele beschrijving en een gedetailleerd model van je applicatie te maken. Je leert use-case diagrammen, use-case templates, activiteitendiagrammen, klassediagrammen en sequentiediagrammen kennen en toepassen.

1.3 De diagrammen en hun samenhang

UML biedt een aantal diagrammen die gezamenlijk de specificatie van het softwaresysteem vormen:

- Het **use-case-diagram** (ook wel requirementsdiagram genoemd) toont hoe het systeem gebruikt kan worden door externe entiteiten zoals menselijke gebruikers.
- Het **activiteitendiagram** toont de activiteiten die door een deel van het systeem worden uitgevoerd, inclusief eventueel parallellisme. Activiteitendiagrammen kunnen ook gebruikt worden voor het beschrijven van bedrijfsprocessen en de workflow.
- Het **klassediagram** (is een statisch diagram) toont de statische structuur van het softwaresysteem weergegeven als klassen en hun relaties.
- Het **sequentiediagram** (is een dynamisch diagram) toont de volgorde in tijd van de boodschappen die in het systeem verstuurd en ontvangen worden.

Opmerking:
Naast de diagrammen biedt de UML-standaard een taal waarin beperkingen, condities en regels die gelden in de diagrammen weergegeven kunnen worden. Deze taal heet de Object Constraint Language (OCL). In dit serviceprogramma gaan we hier niet op in.

Alle diagrammen kunnen op elk moment tijdens het systeemontwikkelingstraject gemaakt worden. In de praktijk zullen veel van de diagrammen ‘meegroeien’ tijdens de ontwikkeling. Soms kan het nodig zijn om een onderscheid te maken tussen verschillende stadia van ontwikkeling waarin de diagrammen zich kunnen bevinden.

Doordat UML een standaard is die voorschrijft hoe begrippen en diagrammen moeten worden weergegeven, ontstaat er voor alle partijen (eindgebruikers, ontwikkelaars, programmeurs en projectmanagers) duidelijkheid over wat er bedoeld wordt.

2 Use-case diagram

2.1 Inleiding

Use-cases binnen UML zijn een middel om de functionele eisen (zie par. 2.2) die gesteld worden aan het softwaresysteem weer te geven. Bij het maken van use-cases gaat men uit van de gebruiker van het systeem. Use-cases beschrijven hoe de gebruiker met het systeem om wil gaan.
Het maken van use-cases is een tamelijk informele techniek. Het gaat erom een goed beeld te vormen van de situatie en wensen van de gebruiker en de mogelijke manieren om hieraan te voldoen.
Iedere analist dient use-cases te kunnen opstellen, terwijl iedere ontwerper en programmeur in staat moet zijn om use-cases te begrijpen en op basis hiervan te beginnen met het ontwerpen en bouwen van een applicatie.
Use-case diagrammen worden voornamelijk in de analysefase gebruikt.

2.2 Functionele requirements in use-cases

De systeemeisen (Eng. requirements) vormen het uitgangspunt voor het maken van een systeem. De systeemeisen zijn de voorwaarden waaraan het (toekomstig) systeem moet voldoen. De systeemeisen worden in drie soorten verdeeld:

- functionele
- niet-functionele en
- pseudo.

In de functionele systeemeisen staat de functionaliteit van het systeem beschreven. Dit gaat over de functies die de toekomstige gebruiker met het systeem kan uitvoeren.

De niet-functionele systeemeisen stellen voorwaarden aan de manier waarop het systeem de betreffende functionaliteit dient te leveren. Hierbij valt te denken aan responsetijden, snelheid, geheugengebruik, de hard- en software-omgeving waarin het moet werken, dat een order pas kan worden toegevoegd als hij bij een klant hoort, enzovoort.

Deze requirements zeggen dus niets over wat het systeem moet kunnen, maar meer welke beperkingen er zijn (**business constraints**).

De pseudo-requirements hebben zelfs niets te maken met het systeem, maar meer met de organisatie of omgeving waarin het systeem wordt gemaakt. Zo is het niet gebruikelijk om voor elke applicatie een nieuw computersysteem aan te schaffen. Een eis van de organisatie kan bijvoorbeeld zijn dat de applicatie moet kunnen draaien op het door de organisatie gebruikte computersysteem. Ze worden daarom pseudo-requirements genoemd. Een andere pseudo-requirement is bijvoorbeeld de eis van de organisatie dat de broncode gerealiseerd moet worden in Java, omdat deze programmeertaal bij de organisatie bekend is.

Tijdens de interviews zullen veel van deze requirements worden genoemd. Het is heel belangrijk om deze goed te noteren, omdat ze eisen aangeven die de gebruikers stellen aan het te realiseren systeem. Hierbij betekent *goed* noteren vooral *meetbaar* noteren. Zo zullen veel gebruikers het hebben over gebruikersvriendelijk en gemakkelijk in het gebruik. Het is de taak van de interviewer om door te vragen naar wat de gebruiker dan verstaat onder gebruikersvriendelijk of gemakkelijk.

Een systeem wordt zelden gemaakt met een volkomen blanke lei als startpunt. Bijna altijd is er een bestaande situatie waarbij het systeem dient aan te sluiten. Voorbeelden hiervan zijn: bestaande hardware en netwerken, bestaande databases waaruit gegevens gebruikt moeten worden, noodzakelijk hergebruik van bestaande systeemcomponenten, aansluiting bij standaard gebruikersinterface richtlijnen, enzovoorts. Al deze niet-functionele systeemeisen die in het eerste stadium van een project bekend zijn, dienen te worden gedocumenteerd en beschouwd als randvoorwaarden waarbinnen de ontwikkeling plaats gaat hebben.

De niet-functionele systeemeisen worden gebruikt in latere stadia van de systeemontwikkeling.

Opdracht 2.2.1

Geef het type aan van onderstaande requirements:

a. Wijzigen van klantgegevens
b. Het systeem moet met behulp van Java gebouwd worden.
c. Inloggen
d. Toevoegen van product
e. Van een klant moet altijd de achternaam bekend zijn
f. Het afdrukken van een klantrapport mag niet langer dan 3 minuten duren.

2.3 Use-cases en actoren

In het use-case diagram leggen we alleen de functionele requirements vast. In het diagram worden deze zichtbaar in use-cases, weergegeven door een ovaal met daarin een werkwoord en meestal een zelfstandig naamwoord.

Elke gebruiker die betrokken is bij een use-case wordt weergegeven d.m.v. een poppetje. De meeste gebruikers zullen het systeem benaderen vanuit hun functie. Deze functie wordt onder het poppetje geplaatst, als naam. Het poppetje zelf wordt 'actor' genoemd. Het is een entiteit die buiten het systeem staat en die direct communiceert met het systeem. Een actor kan zowel een mens, als een ander systeem, als een database zijn. Eén mens kan ook meerdere actoren representeren. Vanuit het systeem gezien zijn de actoren de representatie van de complete buitenwereld. Het systeem communiceert alleen met actoren.

Actieve actoren (de actoren die het systeem in werking stellen) worden links geplaatst, passieve actoren (ondersteunende actoren) rechts.

Een use-case is een beschrijving van een reeks van interacties tussen één of meer actoren en het systeem. Het gezichtspunt hierbij is het externe gedrag van het systeem, vanuit het gezichtspunt van de gebruiker. Een use-case beschrijft één manier waarop het systeem gebruikt kan worden. De beschrijving kan iteraties, keuzen en parameters bevatten.

Soms geven we de functionaliteiten per actor aan in aparte use-case diagrammen. Dit om het overzicht te kunnen behouden. Functionele eisen leiden we af uit de informatie die de actoren verstrekken. Dat geldt bij al bestaande systemen, waarbij er wel op moet worden gelet dat een onderscheid gemaakt wordt tussen *wat* een systeem doet en *hoe* het dit doet. Het laatste moet immers door het nieuwe systeem worden verbeterd.

2.4 De scope

In het use-case diagram wordt het systeem weergegeven door een rechthoek waarbinnen de use-cases getekend worden. We noemen dit de *scope* (=de grenzen aan het te ontwikkelen systeem).

De actoren staan altijd buiten het systeem. Als een actor deelneemt aan een use-case wordt dat genoteerd door een lijn (dus geen pijl) tussen de actor en de use-case.

Bovenin wordt de naam van het use-casediagram gezet.

Hieronder vind je een voorbeeld van een use-case diagram.

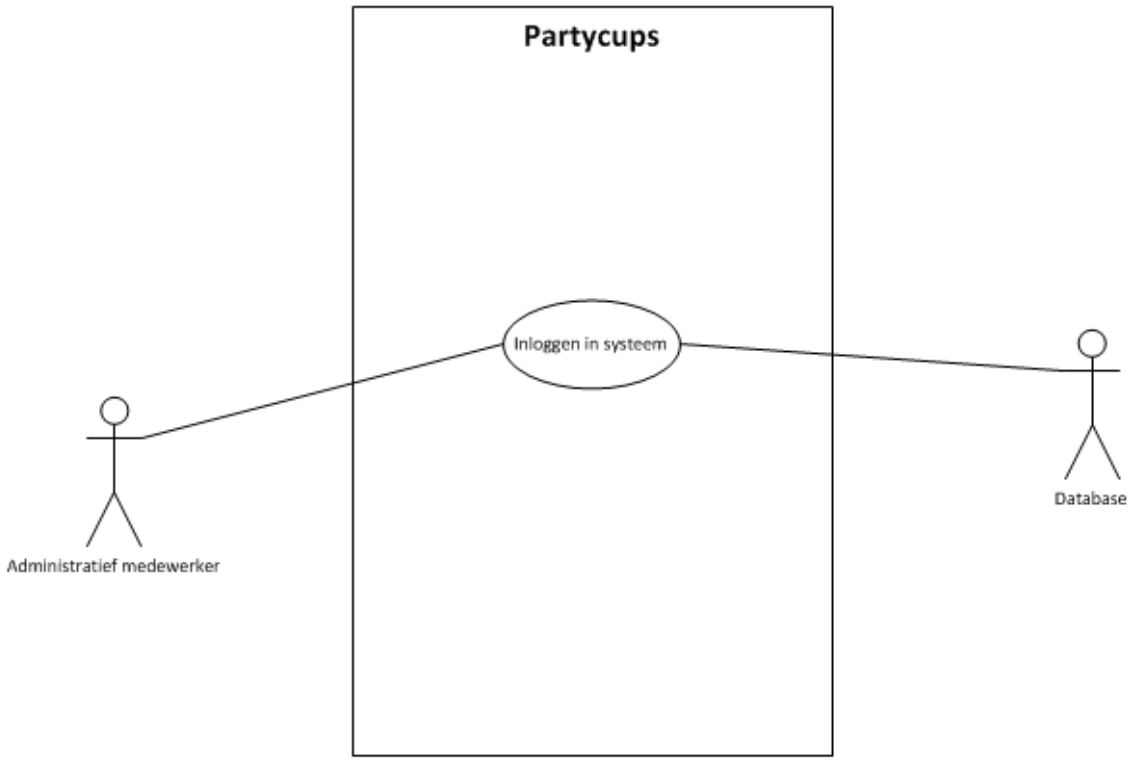
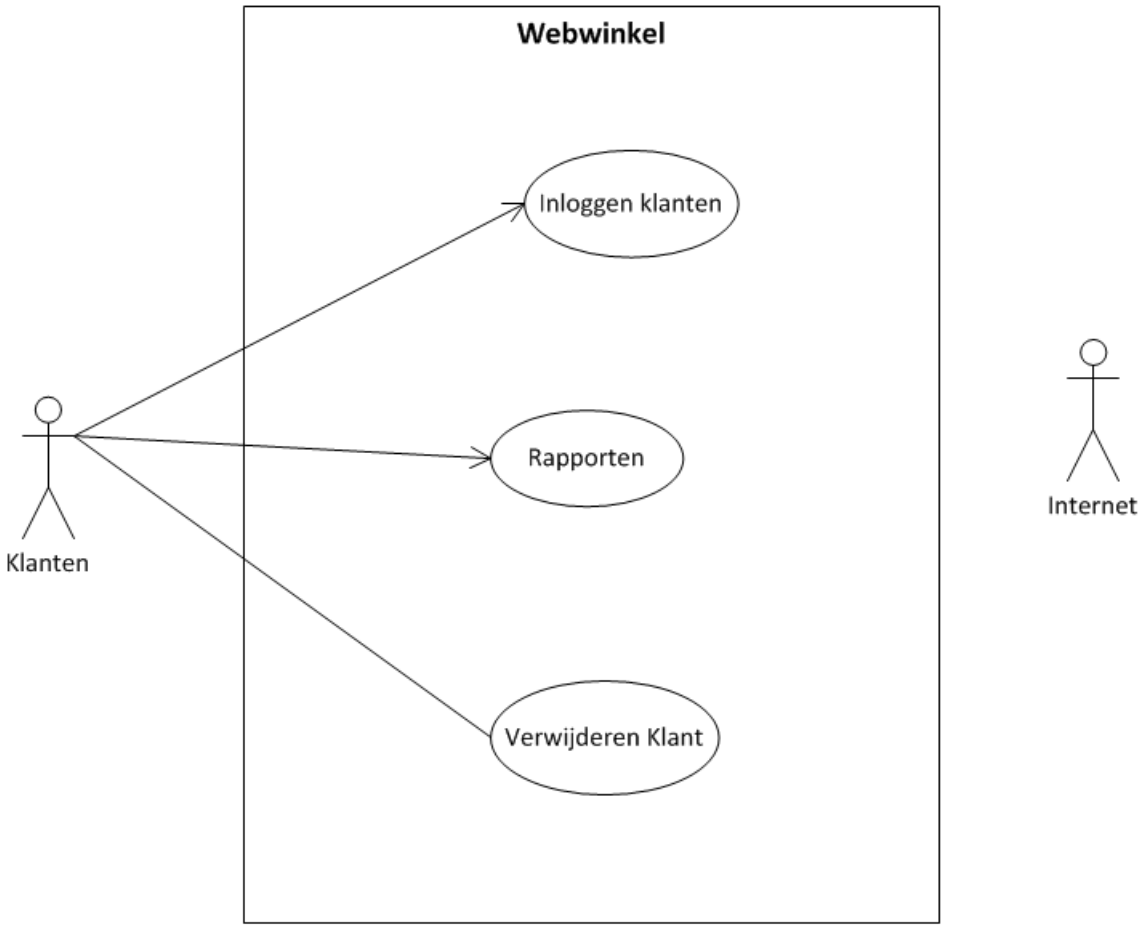


Fig. 2-1Simpel use case diagram

De lijnen tussen actoren en use-cases geven aan welke actoren bij welke use-cases betrokken zijn.

Opdracht 2.4.1

Welke fouten zie je in de volgende tekening? (Het betreft een niet-webbased systeem.)



Houd de use-case diagrammen overzichtelijk. Dus niet zoiets:

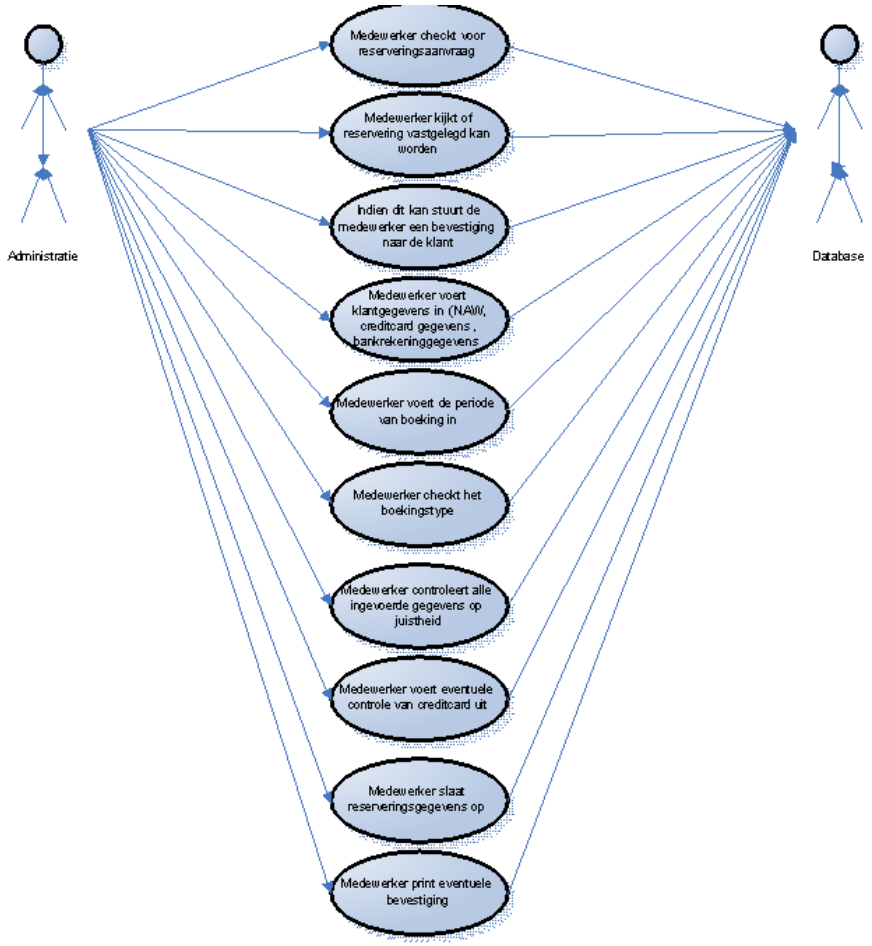


Fig. 2-2 Onoverzichtelijk use-case diagram

Maar liever iets als:

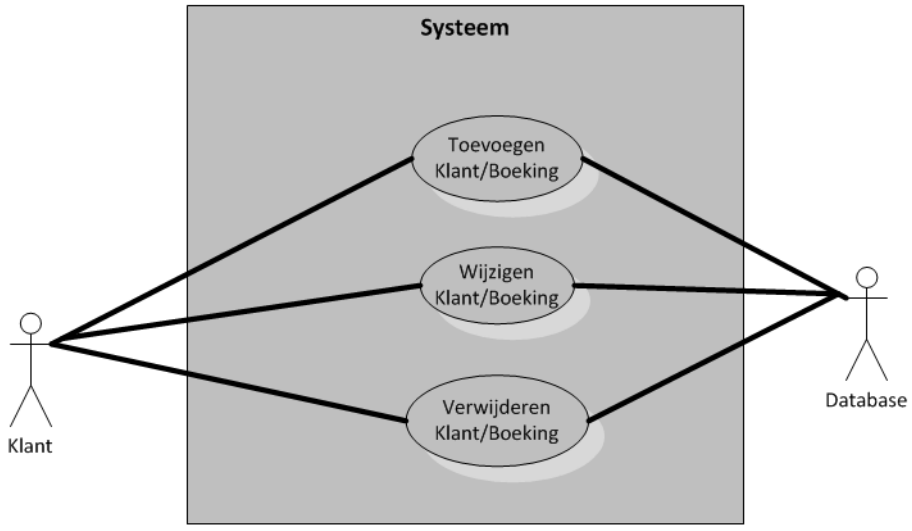
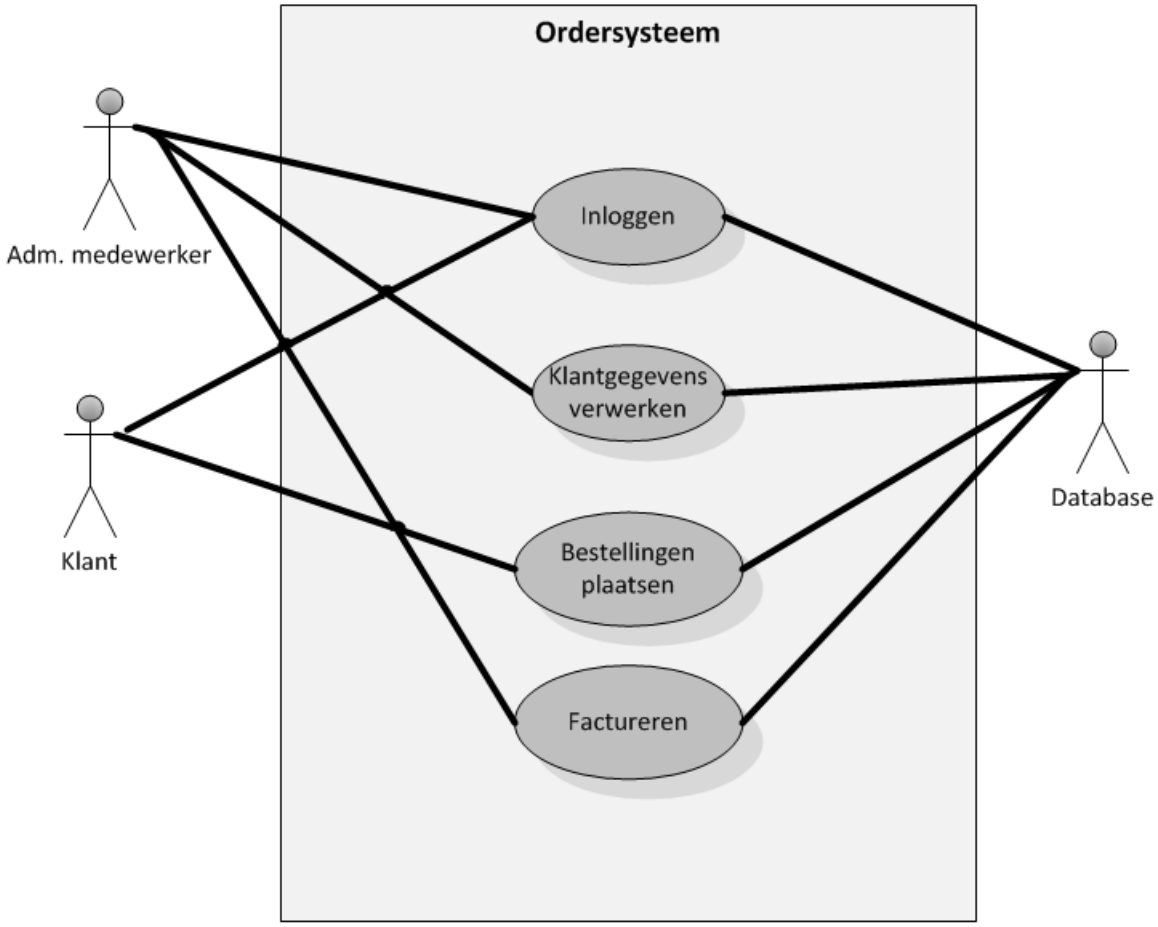


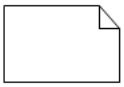
Fig. 2-3 Duidelijk use-case diagram

Opdracht 2.4.2

Welke fout zie je in de onderstaande tekening?



Wil je extra informatie kwijt die niet in het use-case symbool past, dan kun je het opmerking-symbool gebruiken wat je aan de use-case koppelt.



Werkwijze

De use-cases worden opgesteld volgens dit stappenplan:
1. Identificeer de grens van het systeem en vind de actoren.
2. Vind use-cases voor iedere actor.
3. Bepaal (per use-case) de preconditie.
4. Bepaal (per use-case) de interactie.
5. Bekijk (per use-case) mogelijke uitzonderingen.
6. Maak het use-case diagram.

2.5 Use-case template

Voor het beschrijven van een use-case wordt een template gebruikt. Deze vind je hieronder. (Dus voor elke use-case wordt een use-case template gemaakt.)
We houden daarbij de volgende indeling aan: Erachter staat een omschrijving gegeven.

Naam	*De naam die gebruikt wordt om aan de use-case te refereren*
Versie	*Nummer van de revisie*
Actor	*Alle actoren die in de use-case een rol spelen*
Preconditie	*De toestand van het systeem bij de start van de use-case*
Beschrijving	*De uitgebreide, complete beschrijving van de interactie tussen systeem en actor(en)*
Uitzonderingen	*Speciale, te voorziene uitzonderingsgevallen die optreden tijdens de use-case.*
Niet-functionele eisen	*Beperkingen en overige eisen aan de use-case*
Postconditie	*De toestand van het systeem na afloop van de use-case*

Verdere aanvulling:

- Bij **Preconditie** wordt beschreven aan welke voorwaarden voldaan dient te zijn om de use-case te mogen starten. Hierin wordt de toestand beschreven waarin het systeem zich moet bevinden voorafgaand aan toepassing van deze use-case.
- De **Beschrijving** geeft de hoofdlijn van een use-case aan.
- Onder **Uitzonderingen** staan speciale gevallen beschreven, die niet in de hoofdlijn staan. Hierdoor blijft de normale gang van zaken zoals die in de beschrijving gegeven wordt, eenvoudiger en duidelijker. Het is een goede gewoonte om iedere uitzondering een naam te geven.
- **Postconditie** beschrijft aan welke voorwaarden na afloop van de use-case voldaan wordt.

Hieronder volgt een voorbeeld:

Naam	Rekening toevoegen
Versie	1.0
Actor	Baliemedewerker, database
Preconditie	Baliemedewerker heeft beschikking over NAW-gegevens van de klant
Beschrijving	De baliemedewerker maakt aan het systeem bekend dat een nieuwe rekening aangemaakt moet worden en geeft de NAW-gegevens van de klant. Als de klant een bedrijf is, dan wordt ook het Kamer van Koophandelnummer ingevuld. Het systeem checkt of de klant al bekend is. Is dit het geval dan worden de klantrekeningen gecontroleerd op rood staan. Als de klant rood staat dan treedt een uitzondering op. Het systeem maakt het nieuwe rekeningnummer aan de baliemedewerker bekend.
Uitzonderingen	[Rood staan] Als één rekening van de klant rood staat dan wordt hiervan door het systeem melding gegeven. De baliemedewerker kan dan naar de use-case Storten overgaan om de klant de

	gelegenheid te geven het saldo aan te vullen. Als het saldo is aangevuld, dan wordt de rest van deze use-case uitgevoerd.
Niet-functionele eisen	Moet binnen 4 minuten in te voeren zijn.
Postconditie	De klant heeft minstens één rekening.

Opdracht 2.5.1

Tijdens een interview wordt het volgende gezegd:

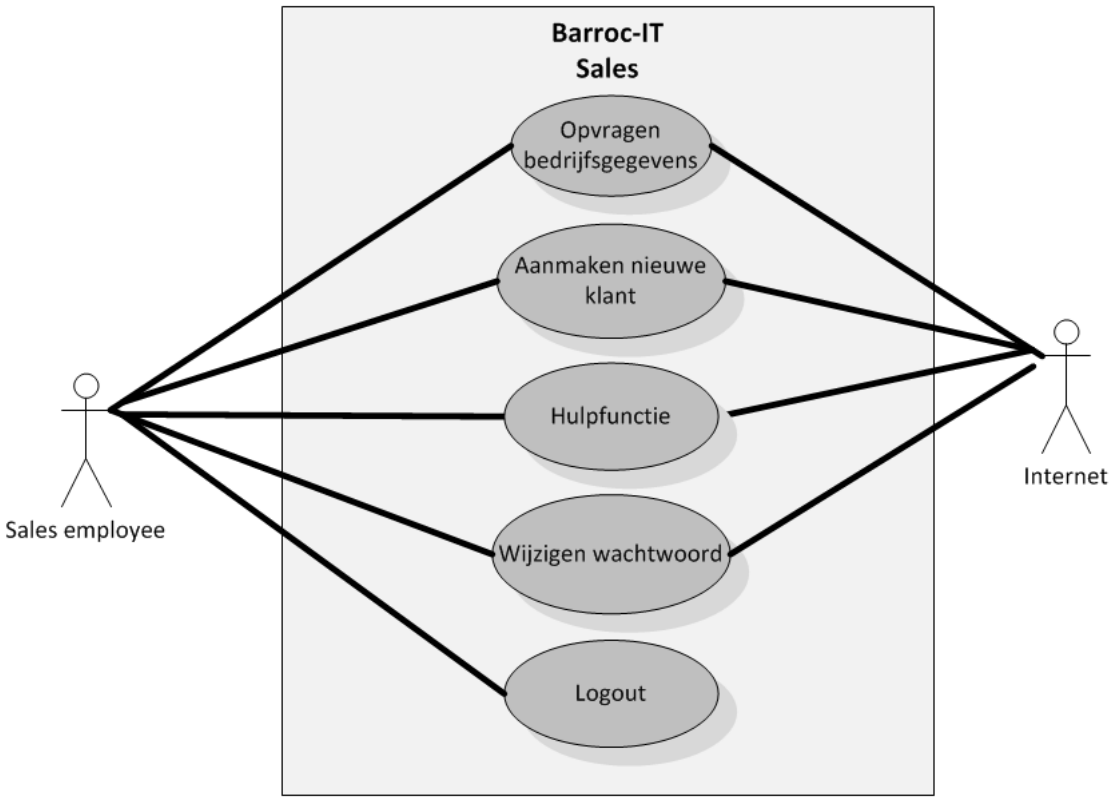
"*Bij het toevoegen van een klant voer ik de naam in en de adresgegevens, bovendien noteer ik de contactmogelijkheden, telefoon, mobiele telefoon, fax en emailgegevens. En in het notitieveld de ordergegevens, die voeg ik vooraan in. De datum waarop ik de invoer doe, zet ik voor de ordergegevens, zodat in het notitieveld steeds de laatste gegevens voorop staan, voorzien van datum kenmerk. Als de gegevens niet leesbaar zijn, voer ik de gegevens die ik heb van de klant wel in, maar dan leg ik het formulier apart om te controleren bij de vertegenwoordiger. Oh ja, de vertegenwoordiger leg ik ook vast bij de klant, elke klant heeft een vaste vertegenwoordiger. Dit leg ik vast door een regiocode toe te kennen aan de klant."*

a. Maak een use-case template van het toevoegen van een klant.
b. Maak een use-case template voor het verwijderen van klantgegevens.

2.6 Opdrachten

Opdracht 2.6.1

Wat is er niet goed aan het volgende diagram?



Opdracht 2.6.2

Bij een schoolbibliotheek worden boeken aan leerlingen en docenten uitgeleend. De uitleentermijn voor leerlingen is een maand, voor docenten is deze een jaar. Aan het eind van de uitleentermijn ontvangt de lener een herinnering. Elke lener krijgt een pas, zodra deze bij de school is toegelaten of in dienst getreden.

a. Geef aan welke actoren er in de tekst van deze opgave staan.
b. Welke actor/actoren ontbreken in de tekst van deze opgave?
c. Geef aan welke use-cases in de tekst van deze opgave aanwezig zijn.

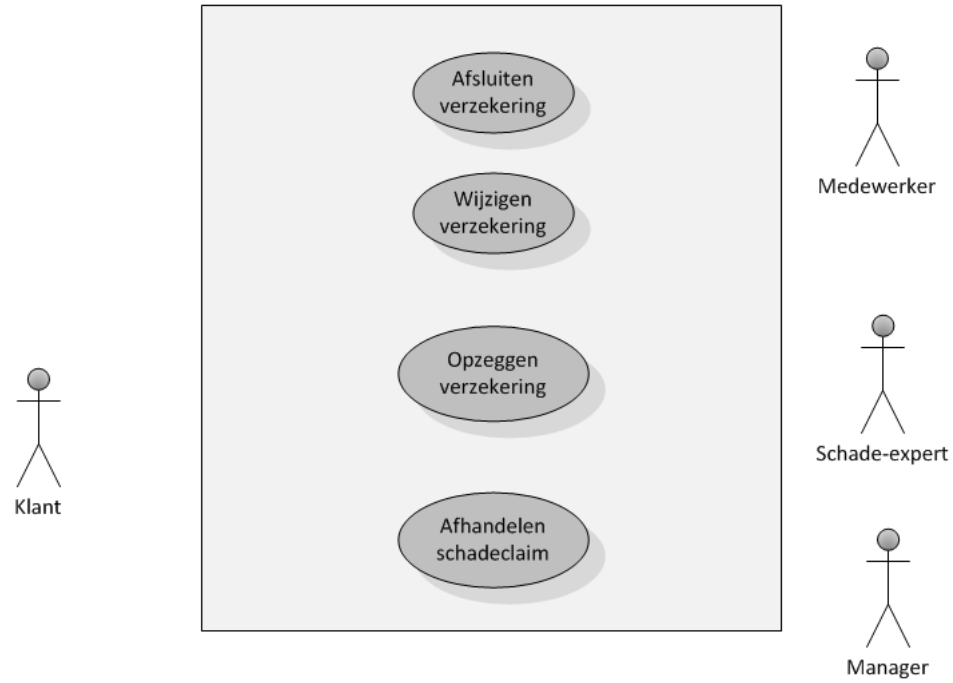
Opdracht 2.6.3

Een diëtiste heeft een programma dat een lijst met producten toont en, na selectie van een product, een aantal gegevens over het product (zoals aanwezige mineralen en vitaminen, de hoeveelheid calorieën, de hoeveelheid koolhydraten en de hoeveelheid vet). De lijst met producten kan door de diëtiste zelf worden uitgebreid met nieuwe, met de daarbij behorende gegevens. De diëtiste kan ook gegevens bij een product wijzigen of een product verwijderen.

a. Maak het use-case diagram met daarin de verschillende use-cases.
b. Werk de use-case "Toevoegen product' uit in een use-case template.

Opdracht 2.6.4

Maak, zonder dat je verdere gegevens hebt, het use-case diagram van de onderstaande figuur af. Verbeter het indien nodig.



Opdracht 2.6.5

Hotel Amsterdam stelt een aantal kamers beschikbaar aan reisorganisatie Vermeer om verder door te verhuren, uiteraard tegen een vaste jaarlijkse vergoeding. Het hotel heeft daarnaast een aantal kamers waarvoor het zelf rechtstreeks de bezetting regelt. Met Vermeer is afgesproken dat, mochten hotelkamers die Vermeer in de verhuur heeft niet bezet worden, het hotel deze kan gebruiken als de eigen kamers volgeboekt zijn. Deze kamers komen pas een week voor de actuele datum vrij voor eventuele boeking.
Het reserveren van hotelkamers kan door Vermeer gebeuren of bij het hotel zelf. De reserveringen die Vermeer afsluit, worden direct doorgegeven aan het hotel in verband met de bezetting en maaltijden.
De reserveringen kunnen binnenkomen per brief, telefoon of e-mail. De volgende gegevens worden daabij steeds vastgelegd:

- Reserveringsnummer,
- NAW-gegevens (Naam, Adres, postcode en Woonplaats) van de klant,
- Klantcode,
- Datum van reservering,
- Volgnummer (indien meerdere kamers voor deze klant worden gereserveerd anders gevuld met 0),
- Datum aankomst,
- Datum vertrek,
- Kamertype (een-, twee- of vierpersoons met bad/douche),
- Boekingstype (met ontbijt, met lunch, met diner of combinaties).

Per kamer kunnen afhankelijk van het type kamer de gastnamen worden ingevuld. Deze namen hoeven niet overeen te komen met de klantnaam. De klanten waarmee het hotel werkt, zijn immers vaak bedrijven of reisorganisaties waaronder reisbureaus of organisaties zoals Vermeer. Indien het om een particuliere klant gaat die nog niet eerder zaken heeft gedaan met het hotel, worden ook creditcardgegevens van de klant vastgelegd.
Alle reserveringen worden in volgorde van binnenkomst door de receptie van het hotel afgehandeld. Overdag wordt elk uur de e-mail geraadpleegd of er reserveringen zijn binnengekomen. De reserveringen worden vastgelegd met de reserveringsadministratie. Het reserveringsoverzicht kan daarbij steeds worden geraadpleegd. Dit overzicht geeft aan welke kamers vrij zijn in een bepaalde periode van een bepaald kamertype.
Als de reservering akkoord is, wordt een reserveringsbevestiging naar de klant gestuurd. Als er geen kamers van het gewenste type in een periode beschikbaar zijn, wordt dit aan de klant meegedeeld (dit kan zowel telefonisch, fax of per e-mail).
Op de dag van aankomst meldt de gast zich bij de receptie van het hotel om in te boeken. Op dat moment wordt aan de gast een kamernummer toegewezen, tenzij dit al bij de reservering heeft plaats gevonden. De inboeking wordt vastgelegd in de reserveringsadministratie.
Door het inboeken wordt er voor de gast ook een rekening geopend waarmee hij of zij in het hotel allerlei voorzieningen kan gebruiken (telefoon, restaurant, minibar, consumpties in de bars van het hotel). Het bijwerken van de gastenrekening gebeurt aan het eind van elke dag, op basis van de diverse uitgeschreven nota's per kamernummer die ondertekend dienen te zijn door de klant (uitgezonderd telefoon en minibar).
Komen gasten niet op de afgesproken aankomstdag aan, dan vervalt de reservering, tenzij op de dag van aankomst bericht van vertraging is ontvangen. Als er geen bericht is binnengekomen en de reservering is opgeheven, wordt de kamer opnieuw in de verhuur opgenomen. De klant die de reservering heeft afgesloten krijgt een boetedrag berekend dat

afhankelijk is van de reserveringslengte (in dagen) en de mogelijke verhuur van de kamer(s) aan anderen.
Op de dag van vertrek meldt de gast zich weer bij de receptie om uit te boeken. Na betaling van de factuur (contant of per creditcard), wordt de gastenrekening afgesloten en overgebracht naar de historische gegevens. Aan het einde van elke maand worden rapportages afgedrukt op basis van deze historische gegevens.

De volgende activiteiten binnen het hotel zijn te onderscheiden:

- Afhandelen reservering. Het ontvangen van de reserveringsaanvraag. Het nagaan of de reservering kan worden vastgelegd. Indien dit kan, dan een bevestiging sturen, anders een mededeling doen. Verder dienen de klantgegevens verwerkt te worden en de creditcard gecontroleerd.
- Wijzigen van reserveringen. Het ontvangen van een reserveringswijziginig. Het nagaan of de reservering bestaat en of de wijziging kan worden doorgevoerd. Als dit kan, wordt een herziene reserveringsbevestiging opgestuurd. Als het niet kan, wordt dit aan de klant meegedeeld.
- Inboeken gasten. Bijwerken van de gegevens, controle of gastgegevens juist zijn vastgelegd, visuele controle creditcard, toewijzing van kamers en overhandiging van sleutel(s).
- Bijwerken gastenrekening. Bijboeken van de gastenrekeningen met behulp van de overzichten van de telefoon- en minibarlijsten en de ondertekende consumptienota’s van het restaurant en de bars.
- Uitboeken van de gast en financiële afhandeling. Een vervallen reservering moet worden omgezet in een boetefactuur voor de klant die de reservering heeft geregeld. De gastenrekening die wordt afgesloten moet worden overgeheveld naar het historische bestand.
- Overzichten opstellen. Aan het einde van iedere maand worden rapporten uitgedraaid met daarin overzichten van de omzetten van de kamerverhuur, het restaurant, de minibars, de twee bars in het hotel en het telefoongebruik.

Werk het use-case diagram uit van het reserveringssysteem voor Hotel Amsterdam.

Oefening 1 „De Fietsfabriek”

Gegeven is de volgende context: (zie hieronder)

a. Maak een Use-Case Diagram in Visio waarin je de hoofdfuncties verwerkt.
b. Maak van elke hoofdfunctie ook een Use-Case Template.

De Fietsfabriek

Een voormalige fietsenmaker is een fietsenfabriek “De Fietsfabriek” begonnen. De fabriek koopt de onderdelen voor de fietsen in en assembleert zelf de fietsen. De ondernemer vraagt om de automatisering van de nieuwe organisatie op te zetten: Hij wil een applicatie om de inkoop, voorraad en verkoop bij te houden. Ook moeten klanten via Internet producten (fietsen en accessoires) kunnen bestellen en betalen. Klanten kunnen zowel privépersonen zijn, als fietswinkels.

Om kort te gaan wil “De Fietsfabriek” de volgende functies:

- *Inkopen van onderdelen bij leveranciers*: De Fietsfabriek heeft verschillende leveranciers waarbij ze producten (fietsonderdelen) inkoopt. Hierbij gaat het om nieuwe voorraad, maar ook voor het aanvullen van bestaande voorraad. Als er besloten wordt om een nieuw product aan te schaffen door de afdeling Marketing, wordt er contact gelegd met de betreffende leverancier en een bestelling gedaan. De leverancier stuurt een bevestiging van de bestelling en de bestelling wordt verzonden samen met de factuur. Zodra de bestelling is aangekomen, wordt deze door de afdeling Voorraadbeheer opgeborgen en de gegevens van de bestelling worden geregistreerd in het systeem. De factuur gaat naar de afdeling Financiën, die deze factuur betaalt. Op vrijdag draait het systeem een lijst uit van alle producten waarvan het aantal onder de vastgestelde limiet komt. Deze worden dan bijbesteld en verwerkt op dezelfde wijze als nieuwe producten.
- *Aanmaken klantaccounts*: Een klant kan via internet zijn eigen account aanmaken. Deze heeft hij nodig om fietsen te kunnen kopen. Hierbij worden NAW- en rekeninggegevens geregistreerd. Elke klant heeft een unieke combinatie van inlognaam en wachtwoord.
- *Verkopen van producten aan klanten*: Een klant kan via internet producten (fietsen en accessoires) kopen. Nadat de klant heeft ingelogd, kan hij producten selecteren om in zijn winkelwagentje te doen. Zodra hij klaar is met selecteren, kan hij de bestelling afronden. Hierbij kan hij eventueel het afleveradres veranderen. Betalingen kunnen gedaan worden via creditcard, paypal of ideal. Zodra de betaling binnen is, geeft de afdeling Financiën een bericht door aan de afdeling Voorraadbeheer. Deze verzamelt de bestelde producten, verpakt ze samen met een verpakkingsbon en laat ze ophalen door het vervoersbedrijf. De voorraden worden direct bijgewerkt in het systeem.
- *Marketing*: Deze afdeling houdt zich bezig met het in en uit de handel nemen van producten. Als het om een nieuw product gaat, registreert deze afdeling de basisgegevens (naam, kleur, maat, minimumvoorraad, et cetera) in het systeem. Wekelijks wordt er een verkooplijst uitgedraaid. Aan de hand van deze lijst, beslist de afdeling Marketing of een product behouden of afgestoten wordt. Dit is afhankelijk van de verhouding verkoopvoorraad en de snelheid waarmee de voorraad verkocht wordt. Wordt een product uit de handel genomen, dan wordt de afdeling Voorraadbeheer hiervan op de hoogte gesteld en wordt het product van de website verwijderd.

3 Klassediagram

3.1 Inleiding

In dit hoofdstuk ligt de focus op de statische structuur van het softwaresysteem.

Alle andere diagrammen maken gebruik van de concepten die we hier modelleren. Daarmee staat het klassediagram zo centraal dat iedereen in staat moet zijn om zo'n diagram te begrijpen. De werkwijze richt zich voornamelijk op het leren maken van een klassediagram in het analysestadium.

3.2 Basisconcepten

In dit deel van het hoofdstuk worden de basisconcepten uit het klassediagram en de bij-behorende UML-notatie stuk-voor-stuk beschreven.

3.2.1 Object of instantie

Een **object** (Eng. object) is iets dat een zelfstandig bestaan leidt, in de werkelijkheid of in de geest. Belangrijk is dat het gezien kan worden als een zelfstandig iets. Over het algemeen is een object een afspiegeling van ofwel een fysiek ding uit de werkelijkheid, zoals een stoel of een auto, ofwel een concept uit die werkelijkheid, zoals een vakantie of een bankrekening. Meer operationeel gezien wordt een object gekenmerkt door zowel zijn toestand (data) als, zijn gedrag (operaties).

Een **instantie** (Eng. Instance) is een andere term die voor object gebruikt wordt. Deze term wordt vooral gebruikt om aan te geven in welke klasse (zie volgende paragraaf) een object thuishoort: het object ‘auto met nummerbord XJ-50-VW' is een instantie van de klasse Automobiel'. Een object of instantie kan in UML op verschillende manieren worden getekend:

- als een rechthoek met daarin onderstreept[1] de symbolische naam van het object, gevolgd door een dubbele punt en de naam van de klasse; (Volgens de gangbare conventies in objecttechnologie worden klassennamen met een hoofdletter en namen van instanties met kleine letter geschreven.)
- als een rechthoek met daarin onderstreept[2] een dubbele punt en de naam van de klasse.

[1] In tegenstelling tot bij sequentiediagrammen (zie verderop) waar het niet onderstreept wordt.
[2] In tegenstelling tot bij sequentiediagrammen (zie verderop) waar het niet onderstreept wordt.

Onder de naam mogen eventueel de namen van de attributen (zie par. 3.2.3) en hun waarden geschreven worden. Voorbeelden van de notatie staan in figuur 3-1.

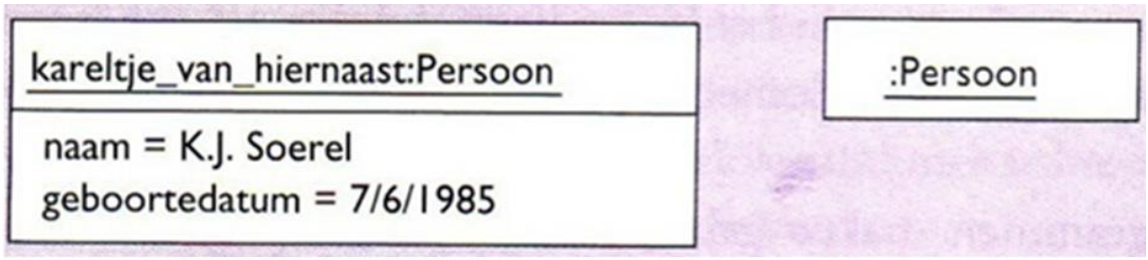


Fig. 3-1 Een instantie

3.2.2 Klasse

Een klasse (Eng. class) is een verzameling van objecten met overeenkomstige eigenschappen. De **klassebeschrijving** geeft de naam van de klasse en de beschrijving van de eigenschappen van de instanties. Deze eigenschappen worden verdeeld in attributen en operaties.

We hebben geen juiste klassen om van uit te gaan, maar moeten geschikte klassen zelf bedenken. Een classificatie is geschikt wanneer alle voor ons interessante objecten uit het domein (de werkelijkheid waarin het systeem een rol speelt of gaat spelen) in te delen zijn in een van de gedefinieerde klassen.

De UML-notatie voor een klasse is een rechthoek met drie compartimenten. Bovenin de rechthoek staat de naam van de klasse. Daaronder de attributen en vervolgens de operaties. Al naargelang het gewenste detailniveau kunnen de operaties en/of attributen weggelaten worden. (Zie figuur 3-2)

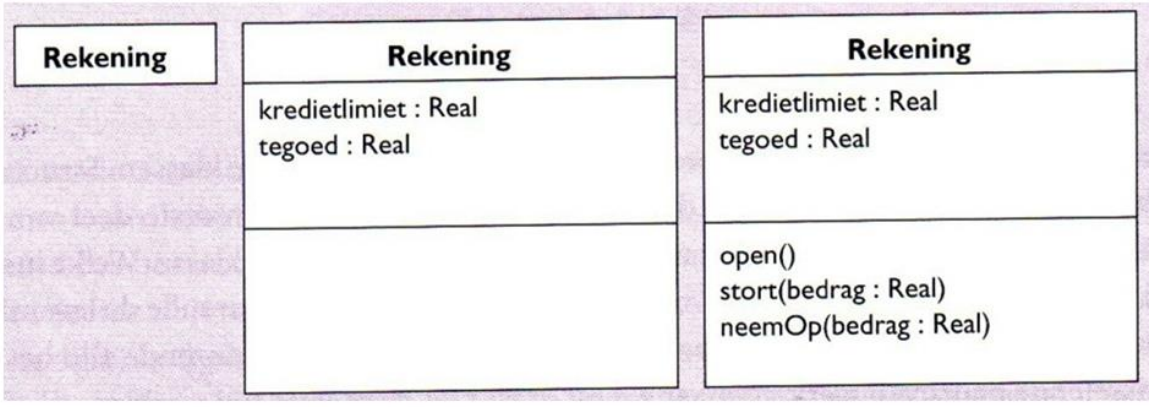


Fig. 3-2 Een klasse met attributen en operaties

3.2.3 Attribuut

Een **attribuut** (Eng. attribute) is informatie die door een object beheerd wordt. Bijvoorbeeld naam, geboortedatum, leeftijd en gewicht zijn attributen van objecten van de klasse Persoon. leder attribuut heeft precies één waarde voor iedere instantie van de klasse. De waarden van al zijn attributen representeren de toestand van een object. Twee of meer objecten uit dezelfde klasse waarvan op een gegeven moment de waarden van al hun attributen dezelfde is, zijn in dezelfde toestand maar blijven altijd verschillende objecten. Ieder object heeft altijd zijn eigen unieke identiteit: de **objectidentiteit**.

Het type van een attribuut wordt weergegeven achter een dubbele punt achter de naam van het attribuut. Enkele voorbeelden: gewicht: Real en naam: String. (Zie figuur 3-2.)

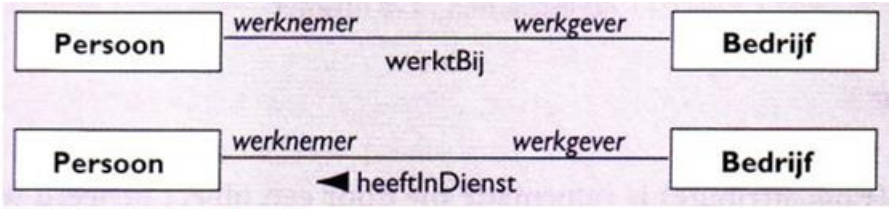
3.2.4 Operatie

Een **operatie** (Eng. operation) beschrijft een service die een object levert. De verzameling operaties bij een object representeert het gedrag van het object. Omdat alle instanties van een klasse dezelfde operaties hebben, worden de operaties beschreven bij de klasse. Een operatie kan argumenten hebben en een resultaat (returnwaarde) opleveren. Objecten communiceren met elkaar door middel van het sturen van boodschappen. Een boodschap (Eng. message) is een verzoek van een object (of een actor), de zender, aan een ander object, de ontvanger, om een bepaalde service te leveren. De ontvanger die de boodschap krijgt voert als gevolg hiervan een operatie uit. Operaties worden in het UML-klassediagram weergegeven in het onderste deel van de klasse. Parameters worden genoteerd tussen ronde haken na de operatienaam. Het type van een eventuele returnwaarde wordt weergegeven achter een dubbele punt na de parameters.

Een voorbeeld: stort (bedrag: Real). (Zie figuur 3-2.)

3.2.5 Associatie

Een **associatie** (Eng. association) is een structurele relatie tussen twee klassen. Structureel betekent hier dat een instantie uit de ene klasse gedurende het grootste deel van zijn bestaan in het systeem verbonden is met een instantie uit de tweede klasse. Welke instantie uit de tweede klasse dit is, is hiervoor niet relevant en dit kan gedurende de levensloop dan ook wijzigen. Zo kan in het voorbeeld in figuur 3-3 Persoon gedurende zijn bestaan bij wisselende bedrijven werken, maar zal hij of zij niet de meeste tijd werkloos zijn.



Figuur 3-3 Een associatie

Tussen twee klassen kunnen ook meerdere associaties liggen. Zo is de associatie *isKlantVan* tussen Persoon en Bedrijf een totaal andere associatie dan *werktBij*.

In UML wordt een associatie aangegeven door een doorgetrokken lijn tussen de twee klassen. De naam van de associatie wordt naast de lijn geschreven. De conventie is dat namen van links naar rechts of van boven naar beneden gelezen worden. In figuur 3-3 geldt dus: *Persoon werktBij Bedrijf* en niet *Bedrijf werktBij Persoon*. Wanneer de naam van rechts naar links, of van onder naar boven gelezen dient te worden, wordt dat aangegeven door bij de naam van de associatie een pijl in de leesrichting te zetten. Het voorbeeld in figuur 3-3 kan dus op twee manieren worden genoteerd.

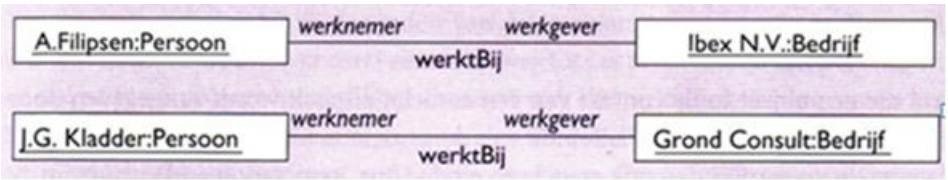


Fig 3-4 Een link

3.2.6 Multipliciteit

Bij een associatie-einde of attribuut kan de multipliciteit aangegeven worden. De **multi-pliciteit** (Eng. multiplicity) geeft het aantal instanties dat het attribuut of het associatie-einde mag bevatten.

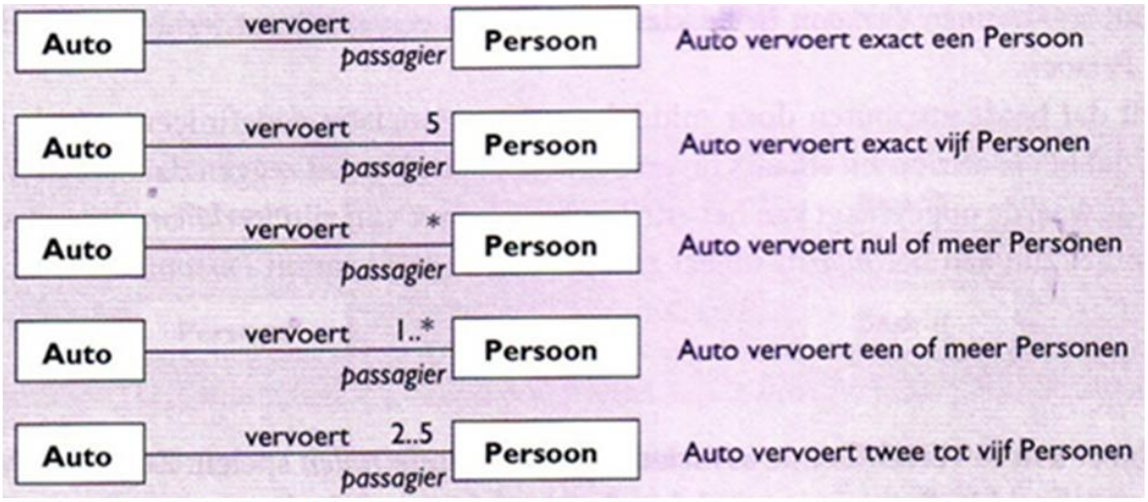
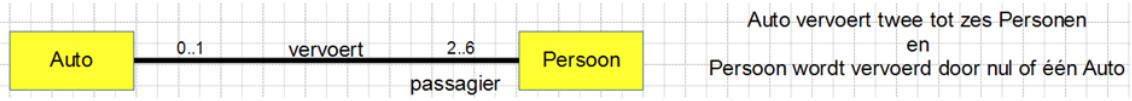


Fig. 3-5 Notatie van multipliciteit bij een associatie

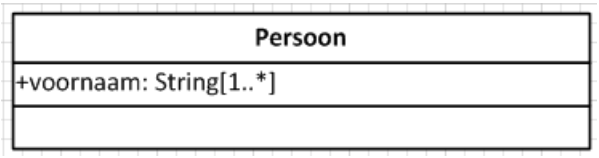
In UML wordt de multipliciteit aangegeven door middel van een sterretje, of met behulp van expliciete cijfers. Wanneer er geen teken staat betekent het dat de multipliciteit één is. Een overzicht van de mogelijke aanduidingen van multipliciteit staat in figuur 3-5. De aanduiding 0..* heeft dezelfde betekenis als een enkel sterretje.



Figuur 3-6 Associatie met multipliciteiten aan beide einden

Multipliciteiten kunnen bij beide einden van de associatie aangegeven worden, zoals in figuur 3-6. In dit voorbeeld vervoert een auto twee tot zes personen en een persoon wordt door nul of één auto vervoerd.

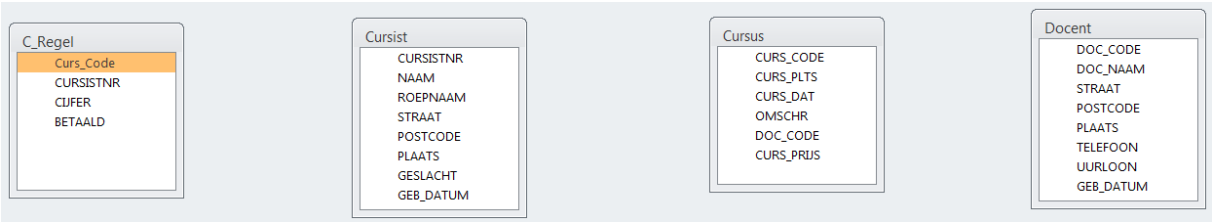
Bij attributen wordt de multipliciteit aangegeven tussen twee rechte haken direct na het type van het attribuut. In het onderstaande voorbeeld heeft de klasse Persoon bijvoorbeeld een attribuut genaamd Voornaam. Omdat veel personen meerdere voornamen hebben is het zinvol dit attribuut de multipliciteit 1..* te geven.



Let hierbij op naamgevingsconventies. Zo'n conventie zou kunnen zijn dat alle attributen en associatie-einden die een meervoudige multipliciteit hebben ook een (rol)naam moeten hebben die deze meervoudigheid aanduidt. Het attribuut in dit voorbeeld zou dan Voornamen genoemd kunnen worden.

Opdracht 3.2.6.1

Geef in de afbeeldingen hieronder de multipliciteiten aan.



3.2.7 Generalisatie en overerving

Generalisatie (Eng. generalisation) is het classificeren van klassen. Net als bij objecten gaan we op zoek naar klassen die identieke kenmerken (operaties/gedrag of attributen/ toestand) hebben. Als we een aantal van dergelijke klassen gevonden hebben, kunnen we een nieuwe klasse specificeren die de gezamenlijke kenmerken bevat. Deze nieuwe klasse heet de **superklasse** (Eng. superclass) van de klassen. De klassen die dezelfde kenmerken hadden noemen we **subklassen** (Eng. subclass). Andere terminologie die vaak gebruikt wordt noemt de superklasse een **generalisatie** (Eng. generalisation), de subklasse heet dan een **specialisatie** (Eng. specialisation).

In UML wordt een supersubklasserelatie aangegeven door een pijl met een grote, gesloten pijlpunt. De pijl is gericht naar de superklasse. (Zie figuur 3-7.) In figuur 3-7 zijn SalarisRekening, SpaarRekening en DividendRekening subklassen van Rekening. Rekening heet hier de superklasse van de andere klassen.

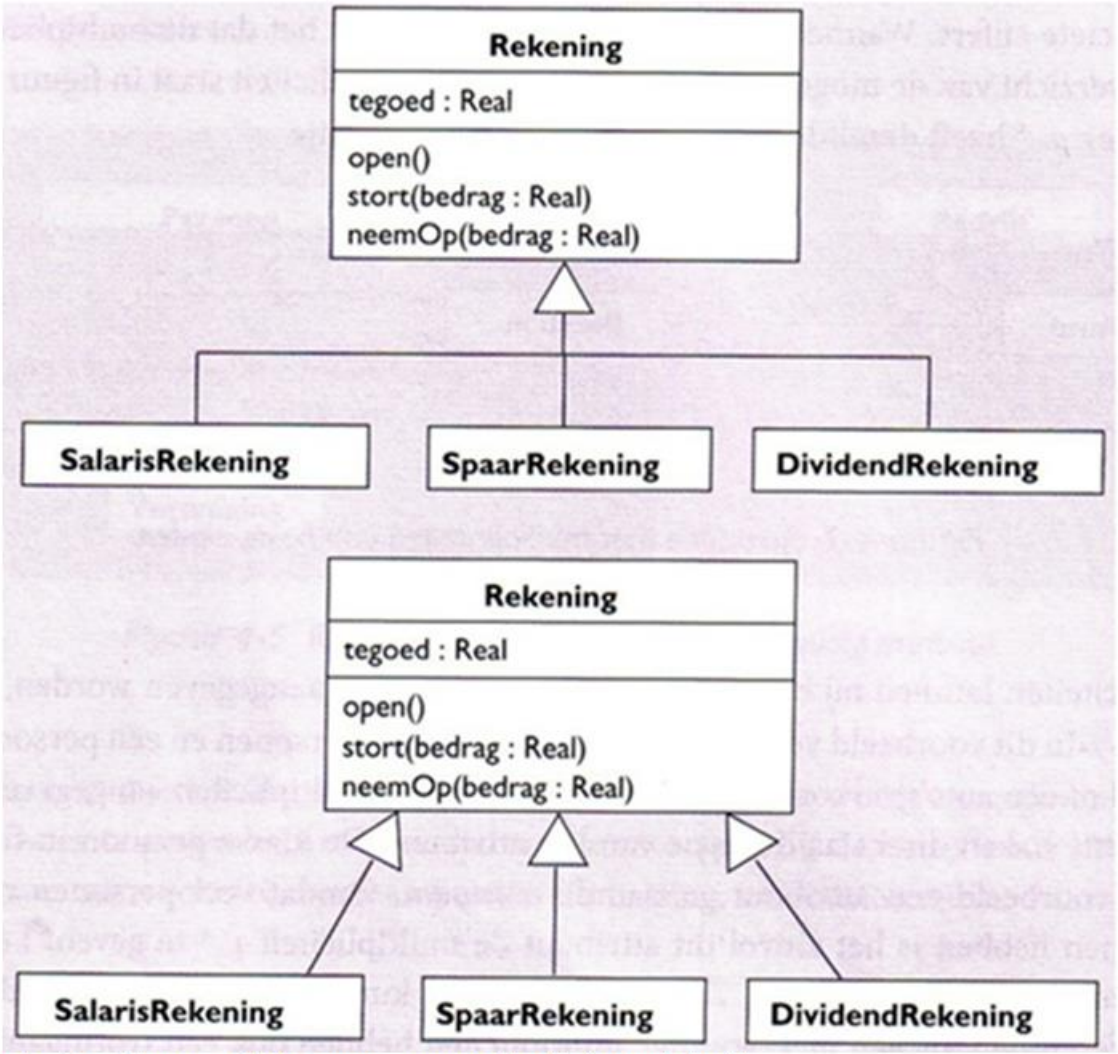


Fig. 3-7 Twee notaties voor generalisatie/specialisatie

Overerving (Eng. inheritance) betekent dat iedere subklasse alle eigenschappen van zijn superklasse erft. In figuur 3-7 hebben dus alle drie de subklassen het attribuut *tegoed* en de operaties *open*, *stort* en *neemOp*. Iedere toevoeging aan de superklasse Rekening wordt automatisch toegevoegd aan al zijn subklassen.

Dit betekent dat overerving altijd betekent: een instantie van een subklasse is-een instantie van zijn superklasse. In figuur 3-7 geldt dus: een Spaarrekening is-een Rekening.

3.2.8 Commentaar

Commentaar kan toegevoegd worden in een zogenaamde **note-box**. Een note-box kan met behulp van een onderbroken lijn gekoppeld worden aan een element in een diagram. In onderstaande figuur staat een voorbeeld van een note-box.

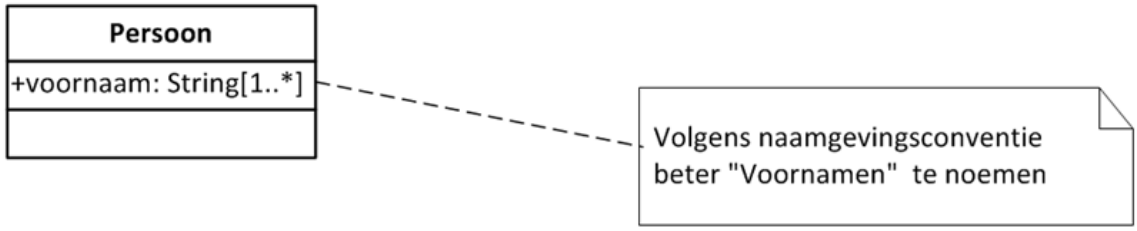


Fig. 3-8 Een note-box met commentaar

3.3 Een voorbeeld van een klassediagram

In de voorgaande paragrafen zijn de losse concepten en de bijbehorende notatie besproken, welke gebruikt worden bij het opstellen van een klassediagram. Ter illustratie staat in figuur 3-11 een voorbeeld van een compleet klassediagram. Het model beschrijft Vluchten, die uitgevoerd worden tussen Luchthavens. Iedere Vlucht wordt uitgevoerd door één Vliegtuig. Vliegtuig heeft twee specialisaties: Straalvliegtuig en PropellerVliegtuig. Ieder Vliegtuig is eigendom van één Luchtvaartmaatschappij. Zoals uit figuur 3-11 blijkt, kunnen details zoals argumenten van operaties en types van attributen weggelaten worden, afhankelijk van het doel van het diagram.

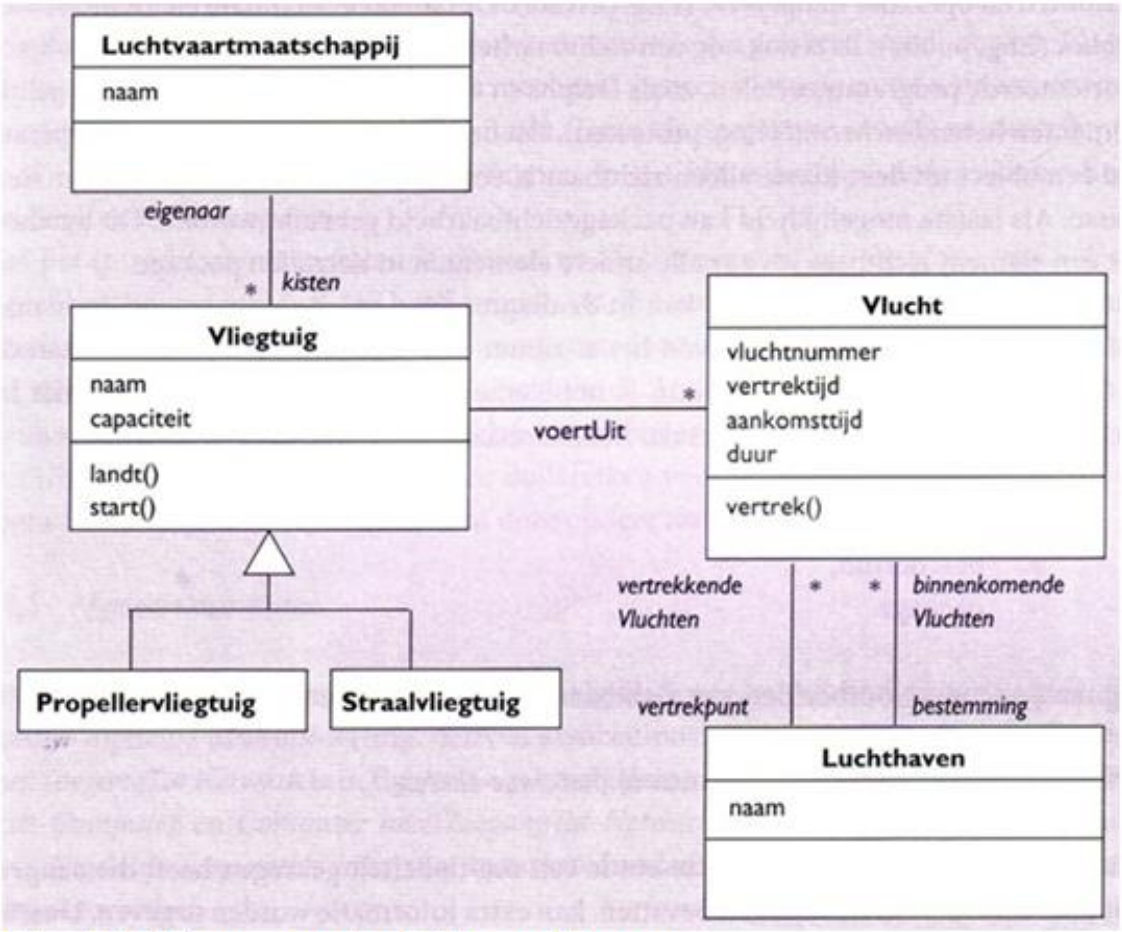


Fig. 3-11 Een compleet klassediagram

3.4 Geavanceerde concepten

3.4.1 Afgeleide attributen

Als de waarde van een attribuut afgeleid kan worden uit de waarden van één of meer andere attributen, associaties of objecten, dan heet dit attribuut een afgeleid attribuut (Eng. derived attribute). De waarde van een afgeleid attribuut hoeft niet per se opgeslagen te worden binnen het object. Het is belangrijk om te beseffen dat een attribuut dus niet iets is dat altijd in het object opgeslagen wordt. Het is informatie waar het object om gevraagd kan worden. Hoe het object aan die informatie komt is voor de klanten van het object van geen enkel belang. Een afgeleid attribuut wordt genoteerd door een schuine streep (/) voor de attribuutnaam te plaatsen. Zie ook figuur 3-12, waarin *nogOpTeNemen* een afgeleid attribuut is.

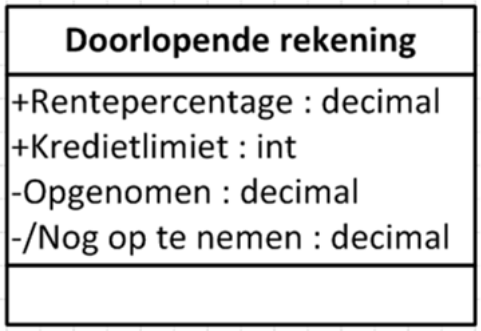


Fig. 3-12 Een klasse met zichtbaarheid van attributen

Vraag 3.4.1.1 (Deze vraag alleen als de SP's Normaliseren al zijn gegeven.)

a. Hoe wordt bij het normalisatieproces een afgeleid attribuut genoemd?
b. Hoe wordt bij het normalisatieproces een afgeleid attribuut (daar wordt dus een andere naam gehanteerd) aangegeven?

3.4.2 Zichtbaarheid van attributen en operaties

De zichtbaarheid van een attribuut of operatie geeft aan in hoeverre het attribuut of de operatie aan de omgeving van het object bekend is. In principe verbergen objecten zoveel mogelijk van hun attributen en operaties voor hun omgeving.
De compleet verborgen attributen en operaties heten **privé** (Eng. private).
De compleet zichtbare elementen heten **publiek** (Eng. public).
Dit onderscheid in zichtbaarheid kan in de diagrammen worden weergeven door middel van een teken voor de naam van het attribuut of de operatie:

+ publiek,
- privé.

Figuur 3-12 toont voorbeelden van zichtbaarheid van attributen.

3.5 Werkwijze

Dit deel beschrijft hoe we te werk gaan om tot een klassediagram te komen. Binnen UML is hiervoor geen eenduidige werkwijze vastgelegd. Deze paragraaf is gebaseerd op ervaring met diverse objectgeoriënteerde systeemontwikkelmethoden.
De werkwijze voor het maken van een klassediagram is het volgende 7-stappenplan:

1. Identificeer alle mogelijke kandidaatklassen
2. Selecteer de klassen uit de lijst van kandidaten.
3. Maak een modeldictionary
4. Identificeer associaties
5. Identificeer attributen
6. Identificeer operaties
7. Generaliseer met behulp van overerving

De volgende paragrafen geven een uitgebreide beschrijving van bovenstaande stappen.

3.5.1 Identificeer alle mogelijke kandidaatklassen

Het doel van deze stap is om zonder beperkingen zoveel mogelijk kandidaatklassen te vinden. In de volgende stap gaan we hier een selectie uit maken. Het belang van deze eerste stap is dat we ons blikveld zoveel mogelijk verruimen. We laten onszelf maar al te vaak van tevoren inperken, om allerlei redenen die niet expliciet gemaakt worden. Veel van deze beperkingen blijken achteraf vaak storend geweest te zijn. Er zijn twee manieren om mogelijke klassen te vinden.

De eerste manier is het onderstrepen van alle zelfstandige naamwoorden in de tekst van de probleembeschrijving. Ook de zelfstandige naamwoorden in eventueel eerder gemaakte use-cases kunnen worden meegenomen, want de use-cases zijn eigenlijk een gedetailleerdere beschrijving van de gebruikerswensen dan de probleembeschrijving. Hoewel we hierdoor zeker een aantal kandidaatklassen kunnen vinden, heeft deze techniek zijn beperkingen. De formulering van de probleembeschrijving speelt hierbij namelijk een veel te grote rol. Het is altijd mogelijk om deze te herschrijven met gebruik van andere woorden, zodat we andere kandidaatklassen vinden. De kans dat we goede kandidaten missen is dan ook levensgroot.

De tweede manier, de manier die wij aanraden voor jullie project Barroc-IT, is het organiseren van brainstormsessies. Tijdens dergelijke sessies worden de stappen 1 t/m 3 van het 7-stappenplan uitgevoerd. Het doel van stap 1 is om zoveel mogelijk potentiële objecten te benoemen. Daarbij is één regel van doorslaggevend belang:

Niemand mag commentaar geven op de kandidaatklassen die door andere deelnemers genoemd worden. Er mag wel om uitleg van de genoemde term gevraagd worden, maar men mag niet de mening ventileren of het wel of niet een goede klasse is.

Het doel van deze regel is om de deelnemers zo open mogelijk hun ideeën te laten uiten. Wanneer er direct commentaar komt, slaan mensen snel dicht en gaan veel ideeën verloren. Het resultaat van deze stap is een grote lijst van kandidaatklassen.

3.5.2 Selecteer de klassen uit de lijst van kandidaten

Uitgaande van de lijst met termen uit de voorgaande stap gaan we nu voor iedere term bepalen of het al dan niet als klasse in ons systeem opgenomen zal worden. We doen dit door termen die geen klasse zijn te verwijderen. De overgebleven termen vormen dan onze eerste verzameling klassen. Tijdens het bespreken van de termen maken we van iedere term een definitie (of beschrijving).
Stel je bij elke kandidaatklasse de volgende vraag:

Heeft de kandidaatklasse een duidelijke verantwoordelijkheid in het systeem?

Wanneer we deze vraag met `ja' beantwoorden, dan is de kandidaat een klasse. In een aantal gevallen is het beantwoorden van deze vraag lastig.
Behalve het positief beantwoorden van deze vraag kunnen we ook kijken waarom een kandidaat géén klasse zou zijn. Hiervoor kunnen we een aantal heuristieken gebruiken. Deze bestaan uit veelvoorkomende redenen waarom een term geen klasse zou zijn. De heuristieken zijn:

Redundante klassen

Een klasse is redundant wanneer hij exact dezelfde verantwoordelijkheden heeft en informatie bevat als een andere klasse. De term die de betekenis het beste weergeeft moet als klasse behouden blijven, de andere verdwijnt. De termen Klant en Afnemer betekenen in veel gevallen bijvoorbeeld hetzelfde.

Irrelevante klassen

Een klasse is irrelevant wanneer hij buiten het probleemdomein ligt. Harde regels zijn hier niet voor te geven. Feitelijk gaan we op dit moment weer de grenzen van het probleemdomein bepalen.
Niet alle objecten uit de werkelijkheid zijn interessant binnen het domein. Hoewel een bal een essentieel object uit de werkelijkheid van een voetbalwedstrijd is, is deze irrelevant wanneer we een planningssysteem voor de voetbalcompetitie modelleren.

Vage klassen

Een klasse moet een duidelijke betekenis hebben. Wanneer het moeilijk blijkt om een definitie van de klasse op papier te krijgen betekent dit meestal dat het om een vage klasse gaat. Zaken als tijd of managementinformatie zijn voorbeelden van vage klassen.

Niet kwantificeerbare zaken

Zaken als water of geld zijn minder vaag, maar kunnen niet gekwantificeerd worden. ‘Eén water' of `drie geld' zijn onzinnige uitspraken. In deze gevallen dienen meer specifiekere termen gekozen te worden, zoals een zee of een watergebied en drie euro's.

Attributen

Attributen zijn dingen waarvan alleen de waarde interessant is. Van het attribuut hoeven we verder niets te weten.

Een attribuut kan in de ene context een object zijn, maar in een andere context slechts een attribuut. De term ‘kleur’ zal bij de verkoopafdeling van een autodealer slechts een attribuut bij auto zijn. In een productiesysteem van een verffabriek zal kleur echter een belangrijk object zijn en waarschijnlijk zelf attributen hebben zoals dekking en mengverhoudingBasiskleuren.

Operaties

Operaties zijn zaken waarvan alleen (het resultaat van) het gedrag interessant is. Soms wordt een operatie als zelfstandig naamwoord geschreven. We kunnen bijvoorbeeld spreken over `het versturen van een poststuk'. Versturen is in deze zin wel een zelfstandig naamwoord en daarom kandidaatklasse, maar waarschijnlijk is deze in het klassediagram beter op zijn plaats als een operatie.

3.5.3 Maak een modeldictionary

Na het vinden van de klassen stellen we een modeldictionary op. Alle gevonden termen nemen we daarin op. De definitie of beschrijving van de klassen is grotendeels al gemaakt in de vorige stap.

3.5.4 Identificeer associaties

In deze stap zoek je naar paren van klassen die iets met elkaar te maken hebben.

3.5.5 Identificeer attributen

We gaan het klassediagram nu nader invullen. Bij alle klassen proberen we attributen te vinden. Een aantal attributen kunnen we uit de lijst van kandidaatklassen uit de eerste stap halen. Iedere kandidaat die we niet als object gekozen hebben omdat we vonden dat het een attribuut was, dienen we nu als attribuut aan een van de objecten toe te voegen. Ook kunnen we attributen vinden uit de beschrijving van de klassen in de modeldictionary.

3.5.6 Identificeer operaties

De volgende stap is op zoek gaan naar operaties bij de objecten. Een aantal operaties kunnen we uit de lijst van kandidaatklassen uit de eerste stap halen. Iedere kandidaat die daar als operatie aangemerkt staat, dienen we nu aan een bepaald object toe te voegen. De belangrijkste vraag die we bij ieder object stellen is:

Wat wil ik dat dit object voor mij doet? Wat is de verantwoordelijkheid van dit object en hoe kan het deze vervullen?

Veel operaties komen in een later stadium naar boven. Dit betekent dat het tijdens deze fase niet de bedoeling is te pogen volledig te zijn. Belangrijker is om voor iedere klasse een aantal typische operaties te vinden, die tezamen een goed inzicht geven in de verantwoordelijkheden van de klasse en haar rol in het geheel. Net als bij de attributen geldt hier ook ‘overdaad schaadt’. Te veel operaties per klasse maken het klassediagram onoverzichtelijk.

3.5.7 Generaliseer met behulp van overerving.

We hebben op dit moment een eerste versie van het klassemodel: klassen met attributen en operaties, en associaties tussen de klassen. We gaan dit model bestuderen om te zien of we generalisaties kunnen vinden van de bestaande klassen. Dit kan dus leiden tot nieuwe (super)klassen.
Generaliseren is het vinden van overeenkomsten in verschillende klassen. We kunnen dus kijken naar de operaties en/of attributen die we bij de klassen gedefinieerd hebben. Zien we meerdere klassen met een aantal identieke operaties en/of attributen, dan kunnen we voor deze klassen een superklasse definiëren. Alle identieke operaties en attributen halen we dan uit de oorspronkelijke klasse en verplaatsen we naar de nieuwe superklasse. Van iedere operatie en ieder attribuut houden we dus maar één definitie over (in de superklasse).
Bij het gebruik van overerving is een waarschuwing op zijn plaats: Subklassen maken een model specifieker en in een aantal gevallen dus minder flexibel.

In figuur 3-14 zijn plastic en glazen flessen als aparte subklassen van de klasse Fles gedefinieerd. Wanneer we nu een fles van een ander materiaal tegenkomen past die niet in het model en moet het model aangepast worden. Een andere mogelijkheid is om slechts één klasse Fles te modelleren, met een attribuut materiaal. Flessen van willekeurig welk materiaal kunnen zonder verandering in het model ondergebracht worden.

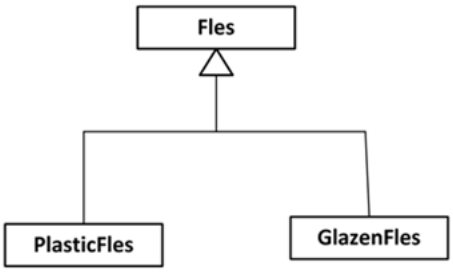


Fig. 3-15 Overerving versus attributen

3.6 Uitgewerkt voorbeeld

Men wil de PC's van alle PC-gebruikers in een database zetten. Lees nu de volgende tekst:

De keuze voor een computer hangt vooral af van het soort gebruik en de maximaal te betalen prijs. Een spelfanaat zal graag wat meer willen betalen voor een computer met uitgebreide multimediamogelijkheden terwijl een zakelijke computergebruiker vooral de stabiliteit van het systeem als belangrijkste aandachtspunt heeft. Toch kan men zich gemakkelijk vergissen. Tegenwoordig is een ingebouwde CD-brander zowel door de spelfanaat als de zakelijke gebruiker gewenst.

Als een zelfstandig naamwoord meer dan één keer voorkomt, onderstrepen we alleen de eerste keer dat het voorkomt. In een enkel geval nemen we ook het bijvoeglijk naamwoord mee als dit een verduidelijking van de kandidaat-klasse oplevert.
Als we gaan onderstrepen, levert dit de volgende zelfstandige naamwoorden op:

De keuze voor een computer hangt vooral af van het soort gebruik en de maximaal te betalen prijs. Een spelfanaat zal graag wat meer willen betalen voor een computer met uitgebreide multimediamogelijkheden terwijl een zakelijke computergebruiker vooral de stabiliteit van het systeem als belangrijkste aandachtspunt heeft. Toch kan men zich gemakkelijk vergissen. Tegenwoordig is een ingebouwde CD-brander zowel door de spelfanaat als de zakelijke gebruiker gewenst.

We kunnen deze zelfstandige naamwoorden in een tabel opnemen en daarnaast aangeven of we deze ook daadwerkelijk als klasse gaan opnemen of afkeuren met opgave van reden.

Kandidaatklasse	Beslissing
Keuze	KandidaatKlasse
Computer	KandidaatKlasse
Gebruik	Attribuut van Gebruiker
Prijs	Attribuut van Computer
Spelfanaat	Wordt kandidaatklasse Gebruiker
Multimediamogelijkheden	Attribuut van Computer
Zakelijke computergebruiker	Wordt kandidaatklasse Gebruiker
Stabiliteit	Attribuut van Computer
Systeem	Synoniem met Computer
Aandachtspunt	Te vaag
CD-brander	Attribuut van Computer

Er worden twee soorten gebruikers vermeld (Spelfanaat en Zakelijke computergebruiker).
Deze vervangen we door de algemenere klasse Gebruiker.
Zo weten we het aantal klassen te beperken tot drie:
Keuze
Computer
Gebruiker

Als we nu verder kijken besluiten we om Keuze als operatie op te nemen in de klasse Gebruiker.
We houden zo dus twee klassen over; Computer en Gebruiker.

Als we er van uitgaan, dat we nu het optimale resultaat bereikt hebben, is de volgende stap om te bepalen welke klassen relaties met elkaar hebben. De gebruiker maakt zijn keuze en bepaalt hoe hij zijn computer gaat gebruiken. Een klassediagram kan er dan zo uitzien:

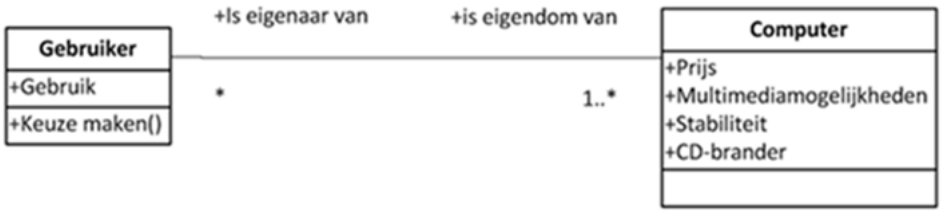


Fig. 3-16 Een klassediagram voor een computer

3.7 Opdrachten

Opdracht 3.7.1

a. Stel uitgaande van de volgende context een klassediagram op. Neem voorlopig alleen maar attributen op die in de context voorkomen.

Op het Radiuscollege wordt een “primitieve” manier van klachtregistratie voor computerproblemen gehanteerd. Aan jou wordt gevraagd om dit te automatiseren. Hiervoor dient een dynamische website opgezet te worden waar iemand zijn probleem kan melden. Dit probleem moet dan zo snel en efficiënt mogelijk opgelost worden.
De volgende eisen worden er aan gesteld:

- Het dient een dynamische website te worden die ontwikkeld is in de programmeertaal PHP.
- De website zal worden ontwikkeld voor FireFox 3.0
- De website moet het mogelijk maken tickets op te slaan, terug te vinden, te modereren, afwerken en archiveren.
- Bij het melden van een klacht worden de volgende zaken genoteerd: naam, achternaam, klas, email adres, computernummer en het lokaal.
- Er moet een inlogsysteem zijn voor een helpdeskmedewerker en een Administrator. Deze hebben ieder hun specifieke rechten.
- Over de lay-out zal van te voren gecommuniceerd worden met de opdrachtgever.
- In het systeem moet opgenomen kunnen worden voor welk lokaal de melding geldt.
- Omdat elk schooljaar het aantal beschikbare lokalen wordt bepaald moet dit makkelijk aangepast kunnen worden.

b. Plaats andere attributen en operaties in de klasse(n) zodat het een reëel klassediagram wordt.

Opdracht 3.7.2

Lees de volgende case door en bepaal de tabellen.

De meeste klantcontacten lopen via de vertegenwoordigers. Ook zijn er klanten die telefonisch of schriftelijk via het kantoor in contact komen met PartyCups. Zodra een klant is opgenomen in het klantenbestand wordt hij toegewezen aan een van de vertegenwoordigers. Bij bestellingen die spontaan bij PartyCups binnenkomen, wordt geen provisie berekend voor de vertegenwoordiger, in alle andere gevallen krijgt de vertegenwoordiger voor zelf aangeleverde orders 2% provisie en over de andere orders vanuit het klantenbestand van de vertegenwoordiger 1%.

De werkweek van een vertegenwoordiger bestaat grotendeels uit het bezoeken van klanten. Het streven is om bestaande klanten ten minste tweemaal per jaar te bezoeken. Dit wordt op dit moment niet altijd gehaald, omdat het soms onduidelijk is wanneer het laatste contact is geweest. Of omdat het inplannen van het bezoek vergeten wordt. Daarnaast moet de vertegenwoordiger natuurlijk ook nieuwe contacten bezoeken. Die nieuwe contacten kunnen door de vertegenwoordiger zelf worden benaderd of zij zijn op eigen inititaief naar PartyCups gekomen. Ook worden potentiële klanten (leads) via het kantoor doorgegeven naar vertegenwoordigers.

Opdracht 3.7.3

Hieronder vind je het klassediagram voor de orderdatabase van PartyCups.

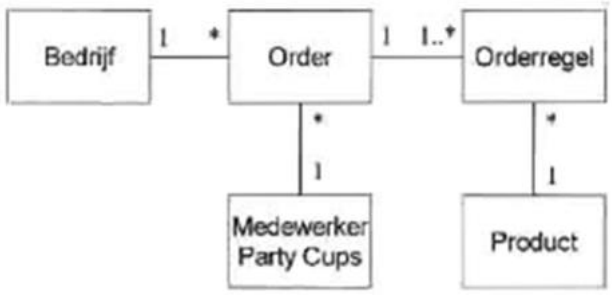


Fig. 3-17 Klassediagram PartyCups

Vaak zal PartyCups de producten uit de bestellingen van zijn klanten moeten doorplaatsen bij zijn leveranciers.

Maak een klassediagram voor de bestellingen bij leveranciers.

Opdracht 3.7.4

Gegeven is de volgende context: (zie De Fietsfabriek blz. 18 en 19.)

a. Maak een tabel van kandidaatsklassen en de beslissing die je daarover neemt.
b. Maak van je uiteindelijke klassen een klassediagram met attributen en operaties.
c. Geef de relaties en multipliciteiten aan tussen de klassen.

3.8 Samenvatting

Een klasse wordt aangegeven door een rechthoek. Deze rechthoek verdelen we in drie vakken. In het bovenste vak zetten we de naam van de klasse, in het vak daaronder de attributen en in het onderste vak de operaties.
De verbindingslijn tussen twee klassen geeft de relatie aan. Klassen kunnen attributen en operaties erven van hogere klassen (superklassen). We geven dit aan met een pijl met een driehoekige, niet-ingekleurde pijlpunt. Deze overerving noemt men in het Engels inheritance.
Voor het maken van een klassediagram bestaat een eenvoudig trucje: In de beschrijving onderstrepen we alle zelfstandige naamwoorden; deze vormen de kandidaatklassen. We bepalen welke aanduidingen vaag zijn, welke aanduidingen terugverwijzen naar andere zelfstandige naamwoorden, enzovoort. Van de overgebleven zelfstandige naamwoorden bepalen we de binding met de andere zelfstandige naamwoorden. Vervolgens tekenen we het klassediagram. Een goed klassediagram heeft zo min mogelijk bindingen. Achter elk attribuut en elke operatie kunnen we het datatype aangeven. De mogelijke datatypes zijn deels afhankelijk van de programmeertaal die gebruikt gaat worden.

Een klassediagram is het kerndiagram van UML. De werkwijze voor het opstellen van een klassediagram vereist veel communicatie en is per definitie sterk iteratief. Ga er vanuit dat er in de loop van het systeemontwikkelingstraject een aantal keer een nieuwe versie van dit diagram nodig is. Deze wijzigingen komen mede voort uit het voortschrijdend inzicht dat we krijgen door het opstellen van de diagrammen.

4 Activiteitendiagrammen

Een activiteitendiagram is een procesgerichte beschrijving van een deel van een systeem. Het wordt onder meer gebruikt om een werkstroom (Eng: workflow) te beschrijven, maar kan ook gebruikt worden om een operatie te beschrijven. Een activiteitendiagram kan activiteiten van meer dan één object beschrijven.

4.1 Leerdoelen

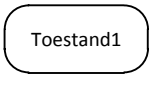
Je leert hoe activiteitendiagrammen gebruikt kunnen worden om use-cases preciezer te modelleren. Door dit te doen maken we de stap van de specificatie van een use-case naar de bouw van een use-case kleiner en dus eenvoudiger. Dit gebruik van activiteitendiagrammen is van belang voor mensen die de analyse uitvoeren.

4.2 De samenstelling van een activiteitendiagram

4.2.1 Activiteit

Een activiteit is een proces, dat eventueel samengesteld kan zijn uit andere activiteiten. Een activiteit die niet opgedeeld wordt heet een atomaire activiteit. Hiervoor wordt soms ook de term actie gebruikt. Als een activiteit zelf weer opgedeeld wordt in subactiviteiten, wordt dit ook wel een samengestelde activiteit genoemd. Deze opdeling wordt dan verder beschreven in een apart activiteitendiagram. Hiermee kunnen hiërarchische activiteitendiagrammen gemaakt worden. Atomaire en samengestelde activiteiten kunnen in een activiteitendiagram naast elkaar voorkomen.

Een activiteit wordt in een activiteitendiagram aangegeven door middel van een symbool met rechte boven- en onderkant en ronde zijkanten. In de rechthoek wordt de zogenaamde activiteitexpressie geschreven. Niet te verwarren met het toestandsymbool () welke rechte zijkanten heeft en afgeronde hoeken.)



Wanneer een activiteit samengesteld is, dan wordt rechtsonder in het activiteitssymbool een icoontje van een mini-activiteitendiagram getekend, dat aangeeft dat het om een geneste structuur gaat. (Visio ondersteunt dit niet. Edraw wel.)

Fig. 4.1 Drie activiteiten en control flow met start en eind

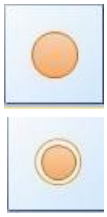
In de figuur hierboven staan drie activiteiten, waarvan maakLevering een samengestelde activiteit is. Hiervoor zal dus een apart activiteitendiagram gemaakt moeten te worden.

4.2.2 Externe activiteit

Een externe activiteit is een activiteit die niet uitgevoerd wordt door het (software) systeem dat beschreven wordt, maar door een entiteit daarbuiten. Net als actoren in use-cases kunnen deze entiteiten mensen zijn, maar ook externe (software) systemen. Wanneer het diagram een softwaresysteem beschrijft, dan zullen de entiteiten vaak overeenkomen met de actoren. Het kan echter ook voorkomen dat een entiteit wel een externe activiteit uitvoert, maar geen directe interactie heeft met het systeem. De entiteit is dan geen actor. Een externe activiteit wordt weergegeven als een activiteit met het stereotype «external». Het voorbeeld in figuur 4.1 geeft aan dat de activiteit leverAf door een entiteit buiten het beschreven systeem, bijvoorbeeld een pakketbezorgdienst, uitgevoerd wordt.

4.2.3 Flow of control

De control flow in een activiteitendiagram geeft de volgorde van de activiteiten aan. Zodra een activiteit beëindigd is, zal de volgende activiteit starten.
De flow of control wordt in een activiteitendiagram weergegeven door een pijl van een activiteit naar de volgende activiteit. In figuur 4.1wordt dus eerst de activiteit ontvangBestelling uitgevoerd, vervolgens de activiteit maakLevering en daarna de activiteit leverAf.
In een activiteitendiagram wordt het begin van het diagram aangegeven met een gesloten cirkel en minimaal één keer een gesloten cirkel met een open cirkel er omheen als eindsymbool.
Je ziet ze ook in figuur 4.1.



4.2.4 Activiteitsparameter

Een activiteitsparameter geeft een input en/of output voor de betreffende activiteit aan. Een activiteitsparameter heeft altijd een naam. Een activiteitsparameter wordt genoteerd door middel van een rechthoek op de grens van het activiteitsymbool met daarin de naam van de parameter. In figuur 4.2 heeft de activiteit vraagAutorisatie het object client_info als inputparameter. De richting van de pijl die bij de parameter staat bepaalt of het een input- of outputparameter betreft.

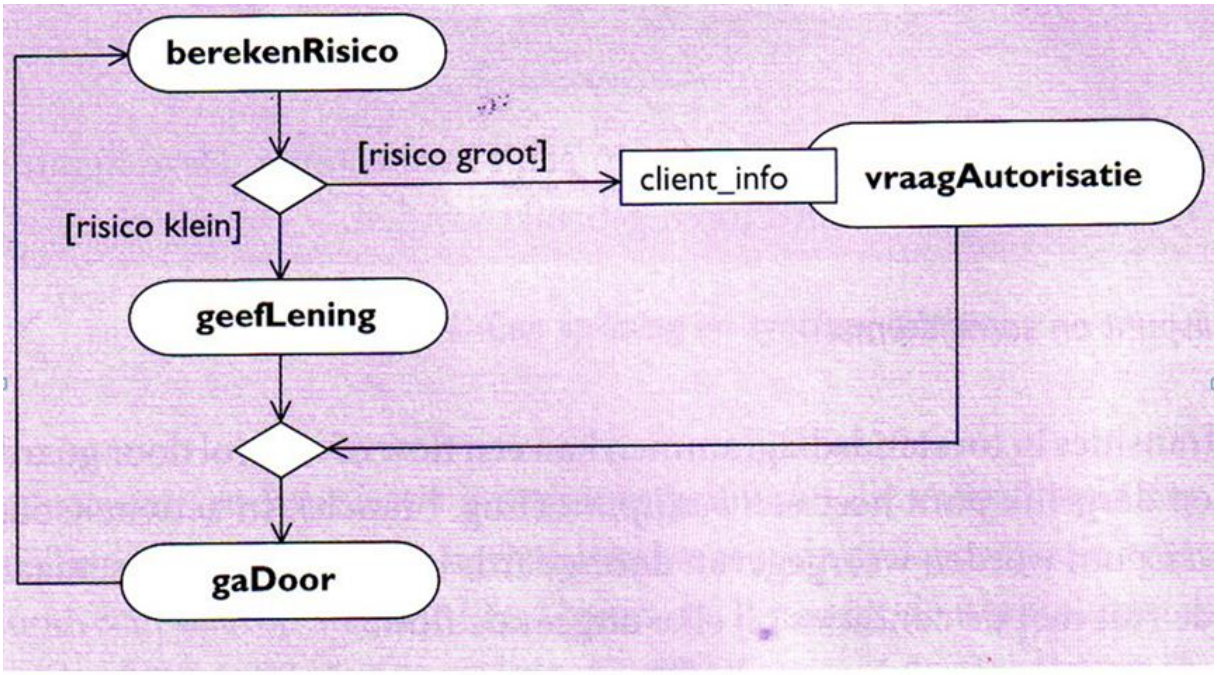


Fig. 4.2 Beslispunt, samenkomst, parameter

4.2.5 Connector

Een connector is een indicatie dat de flow of control-pijl waaraan de connector is vastgemaakt onderbroken is. Een dergelijke onderbreking heeft geen enkele betekenis voor het beschreven systeem, het is alleen voor de tekening van belang. Een connector wordt getekend als een kleine ronde cirkel met daarin het connectorlabel. De flow of control pijl gaat verder bij de connector met dezelfde letter als de letter in de connector van de onderbroken flow of control pijl.

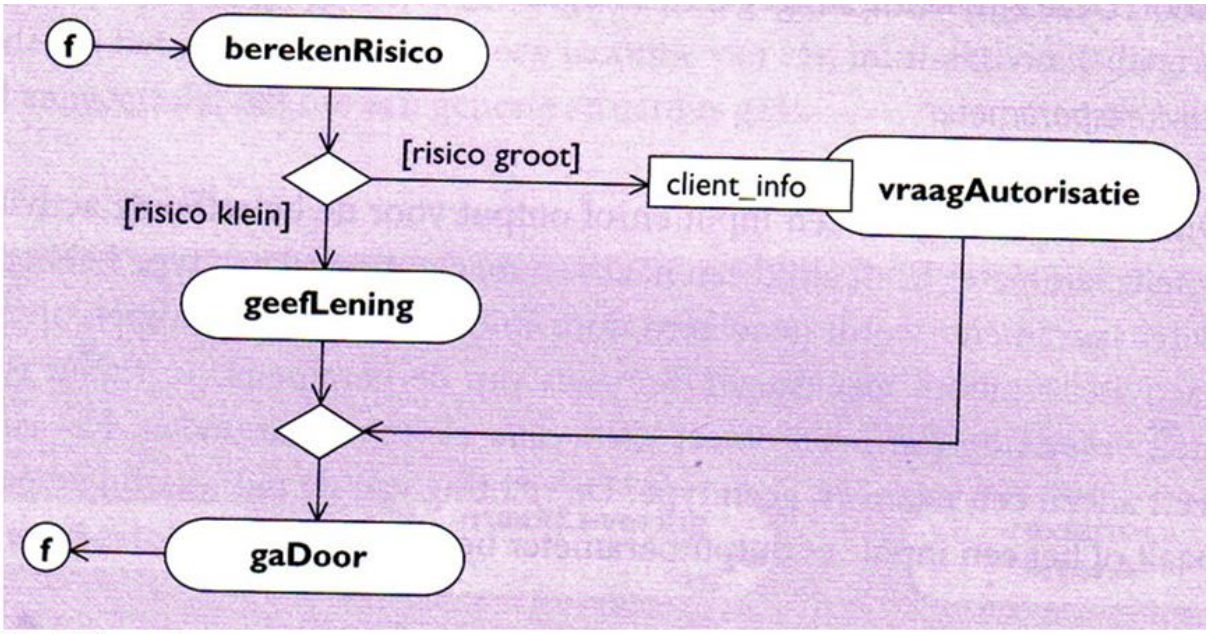


Fig. 4.3 Connectoren

In de bovenstaande figuur zijn de twee cirkels met daarin de letter ‘f’, connectoren. Deze figuur is identiek aan figuur 4.2 waarin geen connectoren gebruikt worden.

4.2.6 Beslispunt en samenkomst

In activiteitendiagrammen wordt een **beslispunt** weergegeven door een liggende ruit met de condities bij elke uitgaande flow.
Figuur 4.2 laat een beslispunt zien, dat begint na de activiteit berekenRisico en kan resulteren in de activiteit geefLening of vraagAutorisatie.
De plaats waar een keuze weer bij elkaar komt heet een **samenkomstpunt** (Eng. merge). Dit wordt ook aangegeven met een liggende ruit, waar twee pijlen binnenkomen en één pijl vertrekt naar de vervolgactiviteit.

4.2.7 Splitsing en synchronisatie

Een **splitsing** (Eng. fork) is een control flow van waaruit twee of meer activiteiten plaatsvinden. In tegenstelling tot een beslispunt worden bij een splitsing alle volgende activiteiten altijd uitgevoerd. Een splitsing geeft aan dat de activiteiten niet van elkaar afhankelijk zijn en dat de volgorde niet van belang is. Ze kunnen sequentieel achter elkaar uitgevoerd worden, maar ze kunnen eventueel ook parallel plaatsvinden.
Een **synchronisatie** (Eng. join) is een control flow vanuit twee of meer activiteiten die eindigt in één activiteit. Verschillende activiteiten, geïntroduceerd door een splitsing kunnen zo weer bij elkaar komen.
De notatie voor een splitsing is een lijn, gevolgd door een loodrecht daarop staande dikke lijn. Vanuit die laatste lijn komen de normale pijlen naar de vervolgactiviteiten. In figuur 4.4 staat een voorbeeld van een splitsing die komt vanuit activiteit geefLening en eindigt in de activiteiten reserveerGeld en maakContract.

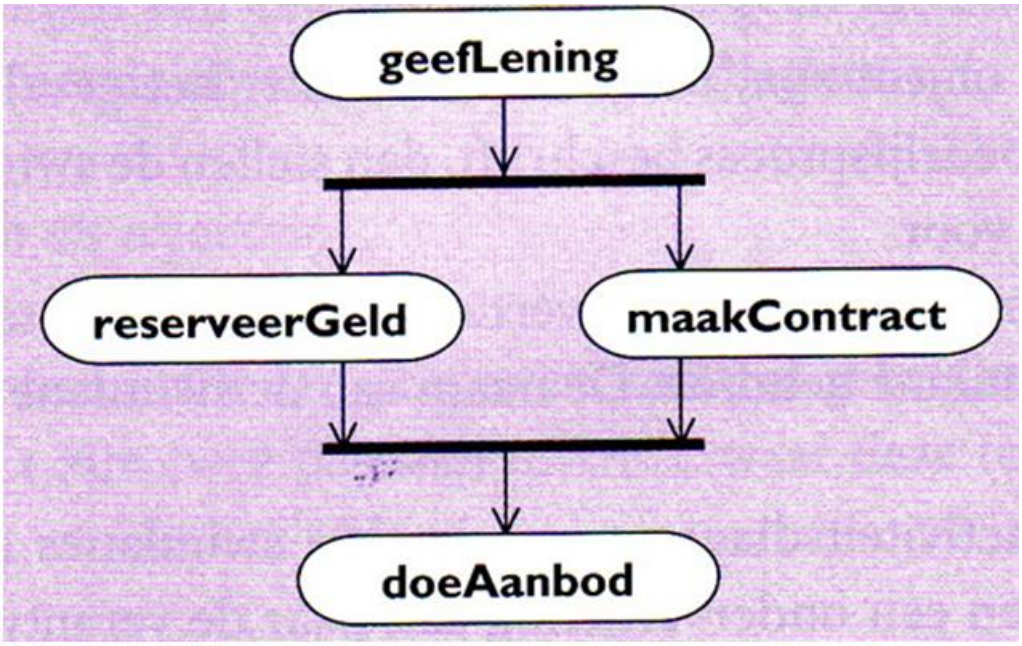


Fig. 4.4 Een splitsing en synchronisatie

De notatie voor een synchronisatie is ongeveer gelijk aan die van een splitsing, maar dan omgekeerd. Twee lijnen vanuit de activiteiten waarin de synchronisatie begint, gevolgd door een loodrecht daarop staande lijn. Vanuit die lijn volgt de normale pijl naar de vervolgactiviteiten. Figuur 4.4 toont een synchronisatie vanuit de activiteiten reserveerGeld en maakContract naar de activiteit doeAanbod.
We tekenen de synchronisatie in het geval twee of meer activiteiten naast elkaar kunnen plaatsvinden, dus niet volgtijdelijk.

4.2.8 Swimlane

In een activiteitendiagram wordt vaak aangegeven wie er verantwoordelijk is voor het uitvoeren van een activiteit. Hiertoe wordt een activiteitendiagram opgedeeld in swimlanes. Een **swimlane** is een verticale baan. Alle activiteiten binnen een swimlane behoren tot één verantwoordelijke eenheid.

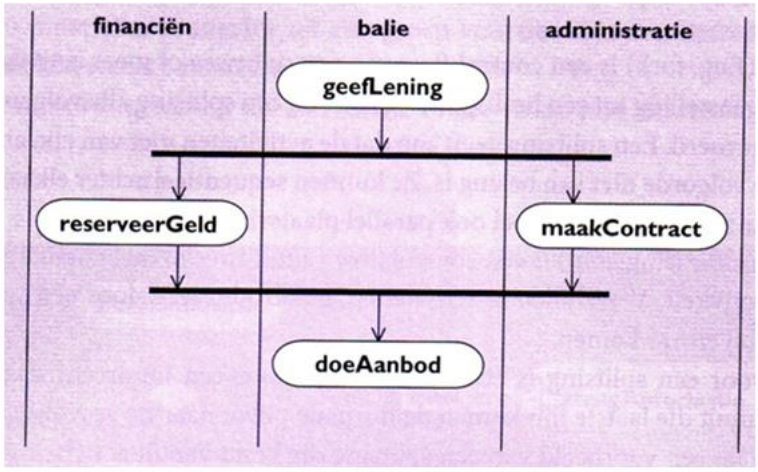


Fig. 4.5 Een activiteitendiagram met swimlanes

Wanneer een activiteitendiagram een bedrijfsproces beschrijft, dan stellen de swimlanes vaak de verantwoordelijke afdelingen voor.
Een swimlane wordt omgeven door twee verticale lijnen, waartussen zich de activiteiten bevinden die tot de swimlane behoren. De naam van de swimlane staat bovenaan tussen de lijnen.
Fig. 4.5 toont een activiteitendiagram waarin drie swimlanes te onderscheiden zijn.
De swimlanes hier geven een onderverdeling aan naar de verantwoordelijke afdelingen binnen een bedrijf.

Een alternatief voor het gebruik van swimlanes is het markeren van elke activiteit met de naam van de uitvoerende eenheid. Deze naam moet tussen haakjes staan boven de naam van de activiteit maar onder de eventuele «external» stereotype. Figuur 4.6 toont een voorbeeld.

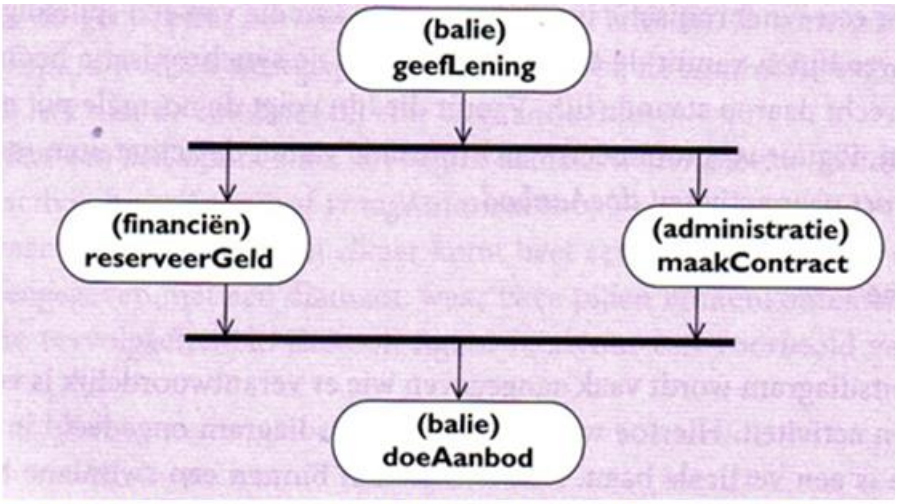


Fig. 4.6 Activiteitendiagram met alternatieve notatie voor swimlanes

4.3 Toepassen van activiteitendiagrammen

Activiteitendiagrammen kunnen binnen UML op verschillende plaatsen gebruikt worden. We beschrijven hier de verschillende plaatsen waar ze gebruikt kunnen worden en hoe ze in deze gevallen samenhangen met de andere UML-diagrammen.

4.3.1 Activiteitendiagram voor workflow

Op hoog niveau kunnen activiteitendiagrammen gebruikt worden voor het beschrijven van bedrijfsprocessen. In dit geval zijn de verantwoordelijke eenheden in de swimlanes de verschillende afdelingen van een bedrijf en is er geen directe relatie met de andere UML-diagrammen.

4.3.2 Activiteitendiagram als algoritme

Een activiteitendiagram beschrijft een proces en kan dus gebruikt worden om een algoritme te beschrijven. Er zijn twee plaatsen in UML waar deze toepassing veel gebruikt wordt.

Operaties
Een activiteitendiagram wordt vaak gebruikt om een operatie van een klasse te definiëren. Pas hierbij op dat het niveau van het diagram hoger blijft dan de programmacode, anders kan dit leiden tot visueel programmeren. Te veel lage details in een activiteitendiagram kunnen leiden tot onleesbare modellen. Over het algemeen wordt een activiteitendiagram gemaakt voor slechts een klein aantal operaties uit een systeem.

Use-cases
Soms wordt een activiteitendiagram ook gebruikt voor het in detail beschrijven van een use-case.
Dit is de toepassing die wij het meest zullen gaan gebruiken.

4.4 Werkwijze

Voor het maken van activiteitendiagrammen is de werkwijze relatief eenvoudig. Het volgende stappenplan wordt gebruikt:

1 Selecteer operaties en/of use-cases en plaats een titel boven het te maken diagram.
2 Vind de activiteiten en de bijbehorende flow of control.
3 Bepaal per activiteit welk object verantwoordelijk is.
4 Splits activiteiten indien nodig.

We gaan de stappen hieronder toelichten.

4.4.1 Selecteer operaties en/of use-cases

De operaties van de klassen uit het klassediagram worden doorlopen om te bezien welke hiervan een activiteitendiagram nodig hebben. Kies hier alleen operaties die werkelijk complex zijn. Meestal is dit een relatief klein aantal. Stap twee en verder worden vervolgens per operatie uitgevoerd.

4.4.2 Vind de activiteiten en de bijbehorende flow

Schrijf de activiteiten op in de volgorde waarin ze uitgevoerd dienen te worden. Wanneer de volgorde niet van belang is, gebruik dan een splitsing waarbij de verschillende activiteiten parallel gedaan worden. Probeer zoveel mogelijk om niet noodzakelijke volgorde te vermijden. Dat levert in de latere implementatie meer vrijheid op. In deze stap kijken we nog niet naar de objecten die verantwoordelijk zijn voor de verschillende activiteiten, we bepalen alleen de activiteiten en de flow. Als het diagram erdoor verduidelijkt wordt, voeg dan connectoren toe.

4.4.3 Bepaal per activiteit welk object verantwoordelijk is

Bepaal voor iedere activiteit door welk object die uitgevoerd wordt. Het object dat de operatie uitvoert kan zelf een activiteit uitvoeren, maar een activiteit kan ook gedelegeerd worden naar een ander object. Een activiteit die door een ander object wordt uitgevoerd, bestaat alleen uit het versturen van een boodschap naar het betreffende object. Bepaal tevens of het uitvoerende object intern is in het systeem of extern.

4.4.4 Splits activiteiten indien nodig

Wanneer in stap 3 voor een activiteit geen enkel verantwoordelijk object gevonden kan worden, dan dient die activiteit opgesplitst te worden in deelactiviteiten waarvoor wel verantwoordelijke objecten te vinden zijn. Deze opdeling kan binnen hetzelfde activiteitendiagram plaatsvinden, maar kan eventueel ook leiden tot een subactiviteitendiagram waarin de samengestelde activiteit beschreven wordt.
Wanneer activiteiten gesplitst zijn, wordt de werkwijze iteratief totdat alle verantwoordelijke objecten gevonden zijn.

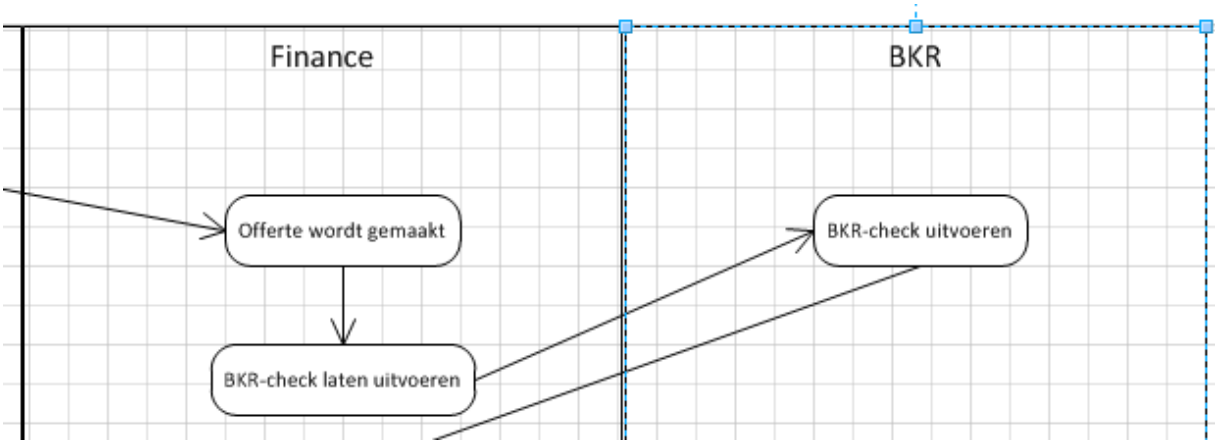
4.5 Samenvatting

Met activiteitendiagrammen worden in UML processen gemodelleerd. Door aan iedere activiteit een verantwoordelijk object te koppelen, leggen we de relatie naar het klassediagram.

4.6 Opdrachten

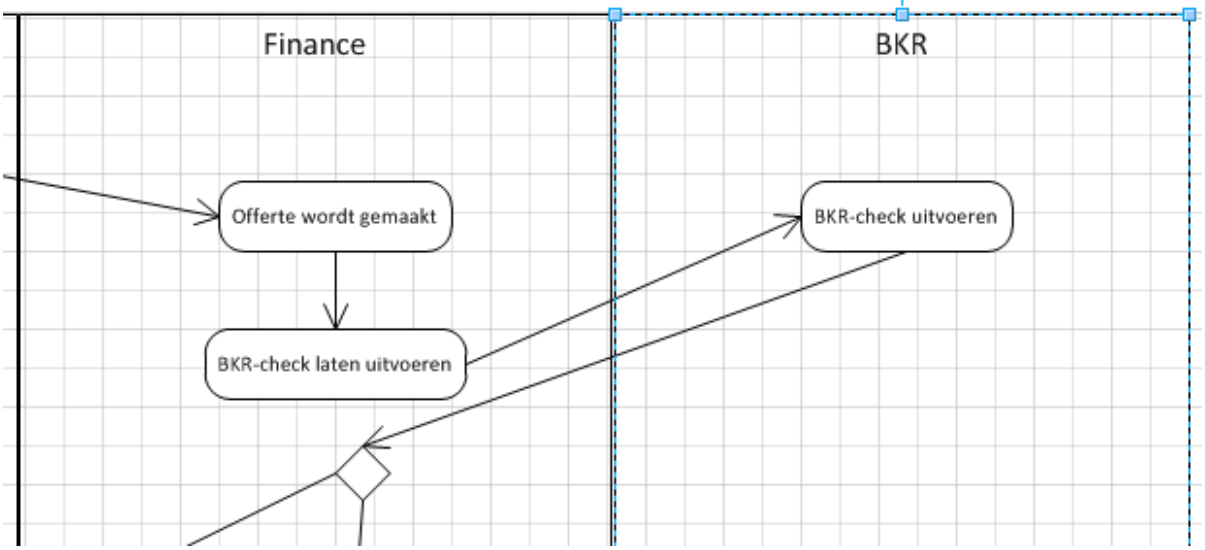
Opdracht 4.6.1

Wat vind je van het volgende (gedeelte van een) schema?



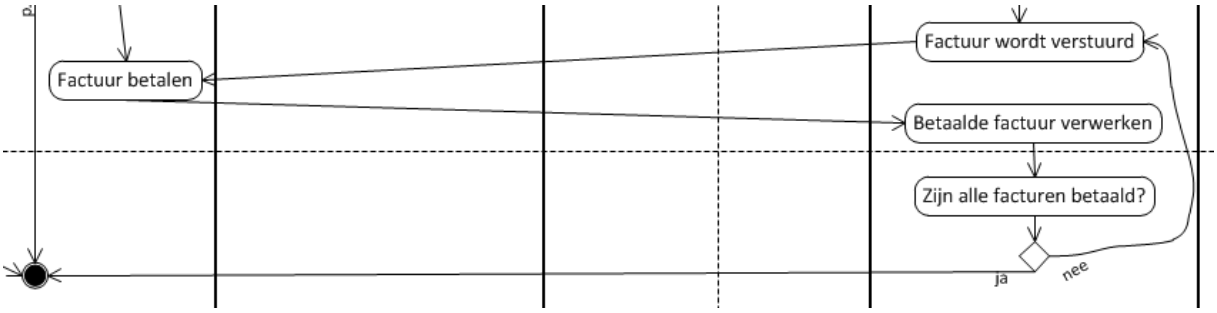
Opdracht 4.6.2

Wat vind je van het volgende (gedeelte van een) schema v.w.b. het beslispunt?



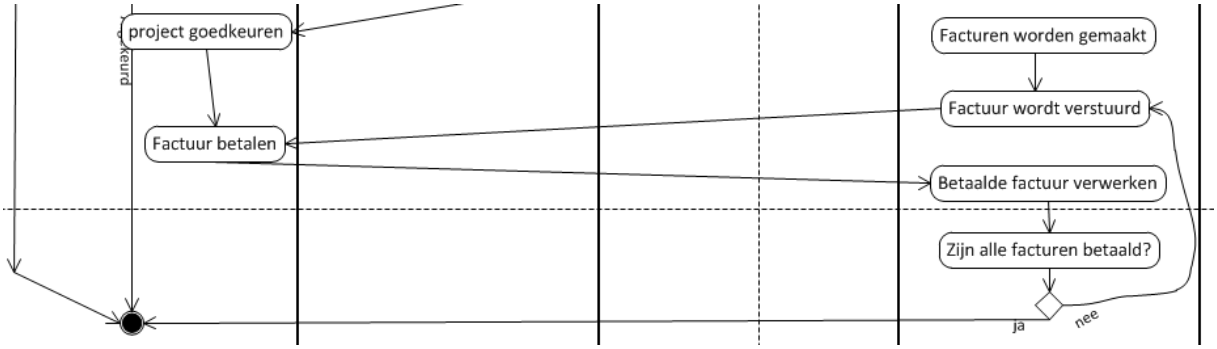
Opdracht 4.6.3

a. Wat vind je van het volgende (gedeelte van een) schema v.w.b. de herhalingslus?
b. Hoe zou je dit kunnen verbeteren?



Opdracht 4.6.4

a. Wat vind je van het volgende (gedeelte van een) schema v.w.b. de activiteit ‘facturen worden gemaakt?



b. Hoe zou je dit kunnen verbeteren?

Opdracht 4.6.5

a. Maak van je lopende project een activiteitendiagram voor het invoeren van een project
b. Maak van je lopende project een activiteitendiagram voor het opstellen van facturen.

Opdracht 4.6.6

Een zwemvereniging organiseert regelmatig Techniekwedstrijden voor synchroonzwemmen. Hierbij komen meisjes bij elkaar om vier figuren te zwemmen. De figuren, die door de meisjes individueel worden uitgevoerd, zijn bijzondere waterballetbewegingen, zoals het op de rug zwemmen met één been recht omhoog gestoken (=balletbeen). Voor de figuren worden door juryleden punten gegeven. Meisjes moeten voor een wedstrijddag hun naam, KNZB-startnummer en verenigingsnaam opgeven. Om het geven van punten te vergemakkelijken krijgt elke deelneemster een startnummer.

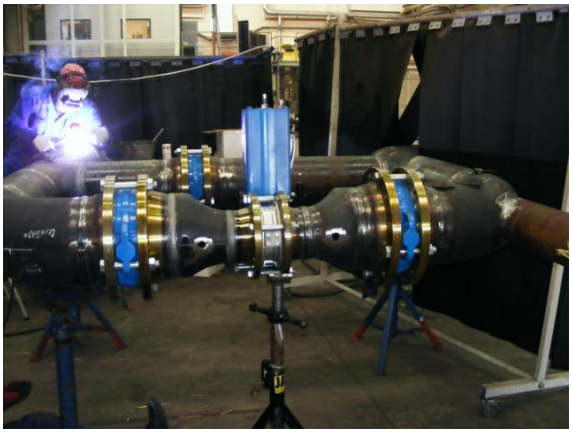
Op de wedstrijddag worden er tegelijkertijd vier verschillende figuren bij verschillende (wedstrijd)tafels gezwommen die rond het zwembad zijn opgesteld.

De deelneemsters worden opgedeeld in vier groepen en elke groep begint met een ander figuur. Als een meisje met een bepaald figuur klaar is, gaat ze naar een volgende tafel voor een andere figuur.

Elk jurylid geeft een cijfer door een genummerde kaart (00 t/m 99) omhoog te steken. De cijfers worden door de puntentellers geregistreerd en omgerekend tot een score. Hierbij wordt het hoogste en het laagste cijfer buiten beschouwing gelaten en het gemiddelde van de overige cijfers wordt vermenigvuldigd met de moeilijkheidsfactor van de figuur. Aan het eind van de wedstrijddag worden de prijzen uitgereikt aan de drie meisjes met de hoogste scores. Er zijn vier verschillende leeftijdscategorieën, met drie prijzen voor elke categorie.

Het te bouwen systeem moet alle informatie kunnen leveren en vastleggen die nodig is voor het plannen, registreren en scoren. Leg het registratiesysteem zwemwedstrijden vast in een activiteitendiagram.

Opdracht 4.6.7



Wanneer bij Dursun Industrie Services BV een lasproject klaar is dient het werk beoordeeld te worden. De afdeling Bewaking projecten schakelt hiervoor een certificeringsbureau in. Dit bureau neemt twee testen. Wanneer deze negatief zijn (dus er is goed werk geleverd) wordt het project goedgekeurd. Als een van de twee testen positief uitvalt (er is dus een fout geconstateerd) wordt alles gecontroleerd. Na afloop hiervan wordt het certificaat of een rapport opgestuurd.

Maak hiervan een activiteitenschema.

Opdracht 4.6.8

a. Maak van de volgende context een activiteitendiagram.

Bij het organiseren van een feest is het gebruikelijk om eerst een budget vast te stellen. Daarnaast is het bepalen van wie wordt uitgenodigd een activiteit. Ook het vaststellen van de catering, het soort vermaak en de locatie horen bij het organiseren van een feest. Soms wordt ook een thema aan een feest gegeven. De locatie is vaak weer afhankelijk van het aantal deelnemers enzovoort.

Als we zelf een informeel feest organiseren worden de activiteiten meestal verdeeld over de organisatoren. Bij meer formele feesten zijn verschillende activiteiten uitbesteed aan functionarissen of organisaties. In alle gevallen is het belangrijk om een goede afstemming te krijgen tussen de betrokkenen.

Stel dat een klant een groot feest wil organiseren in verband met een bedrijfslustrum. Zij huurt een activiteitenbureau in om het feest te realiseren. Het activiteitenbureau zal geregeld overleg hebben met de klant, maar ook met allerlei leveranciers. Enkele daarvan zijn een cateraar (hapjes en drankjes), een leverancier van feestbenodigdheden zoals serviesgoed (we noemen deze Partycups), een entertainmentbedrijf en/of muziekgroep. Al deze actoren voeren een aantal activiteiten uit in relatie tot het feest.

b. Het valt natuurlijk op dat een activiteitendiagram snel groter wordt zodra er meerdere actoren bij een proces betrokken zijn. Daarom wordt een activiteitendiagram vaak gesplitst.
   Vereenvoudig het diagram wat je bij a. hebt gemaakt door alleen de actoren Klant en Activiteitenbureau op te nemen.

Opdracht 4.6.9

Lees de volgende case door:

Bij de start van PartyCups werd het werk door drie mensen gedaan. Ook toen was er echter al een taakverdeling. Zo werden de grotere klanten met vaste jaarcontracten door Miep Gazouw geholpen, terwijl de kleinere accounts door Marloes Grevel werden verzorgd. Jan Jerkovatz deed in die tijd de administratie. In die tijd ontstond ook het idee van het halfjaarlijkse PartyCupfestijn. In het voor- en najaar wordt intussen volgens traditie door PartyCups zelf een feest georganiseerd voor haar medewerkers en vaste klanten. Tijdens dit feest worden de meeste productintroducties door leveranciers van PartyCups aan de klanten gepresenteerd. Oorspronkelijk leverde PartyCups alleen de plastic wegwerpbekers, bestek en borden. Intussen is het leveringspakket enorm uitgebreid. PartyCups kan alle 'hardware'-benodigdheden voor een feest leveren.

Het productassortiment bestaat tegenwoordig uit wegwerpartikelen (bekers, bestek, borden, servetten) al of niet met opdruk, verhuur van aardewerkservies, glaswerk, bestek, tafels, stoelen, afdekkleden, opdienschalen, barbecuematerialen, grote barbecues, partytenten en feestartikelen, waaronder slingers, ballonnen, confetti, maskers. Alle producten kunnen het heel jaar door worden besteld, hoewel een aantal producten duidelijk seizoensgebonden is.

In het begin bestond de klantenkring uitsluitend uit grotere evenementorganisaties die voornamelijk wegwerpbekers bestelden. In die tijd waren die nog niet voorzien van een opdruk. PartyCups kreeg deze bekers geleverd door het productiebedrijf PlasticStans. In overleg met enkele klanten is het idee ontstaan om ook ander wegwerpmateriaal te laten maken en leveren. Zo kunnen feestgangers niet alleen van drank, maar ook van (warme) hapjes worden voorzien. In het assortiment zijn inmiddels naast de bekers ook borden en bestek opgenomen. Nog altijd is PlasticStans de huisleverancier voor het plastic wegwerpmateriaal van PartyCups. Naast het plastic wegwerpmateriaal is ook kartonnen materiaal in het assortiment opgenomen.

De klantenkring van PartyCups is intussen heel divers. Nog steeds zijn de grotere evenementorganisaties klant bij PartyCups. Daarnaast zijn de cateraars een belangrijke doelgroep geworden, terwijl ook veel wordt geleverd aan winkels voor feestartikelen. De laatste tijd komen ook veel bars en discotheken spontaan informeren naar de herbruikbare plastic bekers. Het land is inmiddels verdeeld in een zestal regio's. Elke regio heeft een eigen vertegenwoordiger met als taak de bestaande klantenkring te onderhouden en nieuwe klanten te werven.

Bij een klantbezoek wordt de catalogus met de producten van PartyCups doorgenomen. De catalogus is ingedeeld in productgroepen: wegwerpartikelen, herbruikbaar plastic, serviesverhuur, meubilair (hieronder vallen ook de tenten en barbecues) en feestartikelen. Zoals al eerder aangegeven, worden de bestellingen genoteerd op een orderformulier. Elke week worden deze orderformulieren afgeleverd op het hoofdkantoor, waarna ze worden verwerkt. Orderformulieren die door de administratie niet verwerkt kunnen worden omdat ze onvolledig of onleesbaar zijn worden door de administratie terzijde gelegd. Pas na

telefonisch contact met de vertegenwoordiger worden deze of aangevuld of verbeterd en alsnog verwerkt.

Na verwerking krijgt de klant een orderbevestiging die ondertekend teruggefaxt moet worden, daarna wordt de order uitgevoerd. Eventuele productiebestellingen worden geplaatst bij de desbetreffende leveranciers. De vertegenwoordiger krijgt elke maand een overzicht van de behaalde provisie en gerealiseerde orders (orders ondertekend door de klant). De looptijd van een order beslaat al gauw een paar weken. Als een klant niet ondertekent, is het de bedoelding dat de vertegenwoordiger de klant telefonisch benadert. Dit wordt soms ook vergeten waardoor een aantal orders niet wordt uitgevoerd.

Ten slotte behoort de vertegenwoordiger elk half jaar een overzicht te krijgen van de klanten en hun geplaatste orders. Daarnaast wordt een overzicht gemaakt van klanten die het laatste halfjaar niets hebben besteld. Met behulp van deze lijsten stelt de vertegenwoordiger weer een bezoeklijst op voor het komende half jaar. Bovendien geeft de vertegenwoordiger aan, welke klanten uitgenodigd moeten worden voor het halfjaarlijkse PartyCupfestijn.

Maak het activiteitendiagram van het opstellen van de bezoeklijst. Ga er vanuit dat het initiatief bij de administratie ligt. Er zijn twee actoren bij betrokken: de administratief medewerker en de vertegenwoordiger.

Opdracht 4.6.10

Lees de volgende case door.

De inkoop van bestaande producten uit het assortiment wordt door Jan Jerkovatz verzorgd. Nieuwe producten die in het assortiment kunnen worden opgenomen worden gezamenlijk met Miep Gazouw en Marloes Grevel bekeken. De uiteindelijke beslissing voor assortimentsuitbreiding ligt bij Miep Gazouw. Bij de nieuwe productintroducties op het halfjaarlijkse PartyCupfestijn worden ook flyers gemaakt met informatie. Daarna nemen de vertegenwoordigers deze flyers mee naar hun klanten, samen met de nieuwe catalogi.

Het samenstellen van de halfjaarlijkse productcatalogus van het bedrijf is de verantwoordelijkheid van Marloes Grevel. Elk halfjaar begint met het aanschrijven door het hoofd Verkoop van alle leveranciers om nieuwe informatie te verkrijgen. Prijzen worden op aparte inlegvellen vastgelegd zodat deze maandelijks kunnen worden bijgesteld. Naast nieuwe informatie wordt ook het oude materiaal herzien en eventueel vervangen. Bijvoorbeeld nieuw beeldmateriaal van bestaande producten. De catalogus wordt naar grotere klanten toegestuurd, de andere worden verspreid op het PartyCupsfestijn of via de vertegenwoordiger bij de klant afgeleverd.

Maak het activiteitendiagram van het samenstellen van de productcatalogus. (Zie laatste alinea.)

Opdracht 4.6.11

Lees de volgende case door.

Bij klachten over de orders wordt de vertegenwoordiger altijd betrokken bij de oplossing. Soms klopt er iets niet met aantallen, andere keren is vergeten een korting door te berekenen. Een heel enkele keer wordt geklaagd over het materiaal zelf. Dan wordt ook de leverancier op de hoogte gebracht van de klacht. Alleen de klant kan de klacht opheffen. In een enkel geval kan het betekenen dat de afdeling Administratie de hele order crediteert. Als dat gebeurt wordt de klacht ook als afgehandeld gezien. Uiteraard moet dan ook eventuele provisie van de vertegenwoordiger worden teruggedraaid.

Maak het activiteitendiagram van de klachtafhandeling. Ga daarbij uit van de volgende actoren: Klant, Administratief medewerker, Vertegenwoordiger en Leverancier.

5 Sequentiediagrammen

In de UML-modellen gaat het steeds over gedrag. Hiervòòr hebben we steeds bepaald wèlke objecten met elkaar communiceren, in dit hoofdstuk geven we aan hóe ze dat doen. We geven de volgorde van de berichten tussen klassen en objecten aan en ook of er antwoord verwacht wordt. Het hulpmiddel dat UML hiervoor aanbiedt is het *sequentiediagram*. Sequentiediagrammen behoren tot de *interactiediagrammen* van UML.

Sequentiediagrammen worden voornamelijk in de ontwerpfase van een project gebruikt.

5.1 Het doel van een sequentiediagram

In sequentiediagrammen geven we aan *hoe een proces verloopt*. Elke handeling wordt ingetekend; elke keer dat een object aangesproken wordt, nemen we dat op in het sequentiediagram. Daarmee lijkt het veel op het activiteitendiagram. Het verschil is dat we ons bij een activiteitendiagram meer richten op het menselijk handelen, terwijl we nu over **programmaonderdelen** praten. We zijn nu dus meer met het programmaontwerp bezig. We hebben al klassen gevormd, we weten op welke wijze de klassen contact met elkaar hebben. Nu wordt het tijd om vast te leggen hoe de berichtenstroom tussen de klassen verloopt. Dat lijkt heel simpel, maar we moeten ook rekening houden met alle mogelijke uitzonderingssituaties.

Een **interactie** is een reeks van gebeurtenissen die plaatsvindt tijdens één specifieke (deel)sessie met het systeem. Uitgaande van de interactie tussen de actor en het systeem wordt nu ook de interactie met behulp van boodschappen tussen objecten in het systeem beschreven.

De instanties in een sequentiediagram hebben dezelfde notatie als in het klassediagram, exclusief de onderstreping. (Sommige methoden kiezen wel voor onderstreping. Wij doen dat hier niet.) Aan iedere rechthoek is tevens een zogenaamde levenslijn gekoppeld. Deze levenslijn symboliseert dezelfde instantie als de rechthoek. De figuur hieronder toont een sequentiediagram.

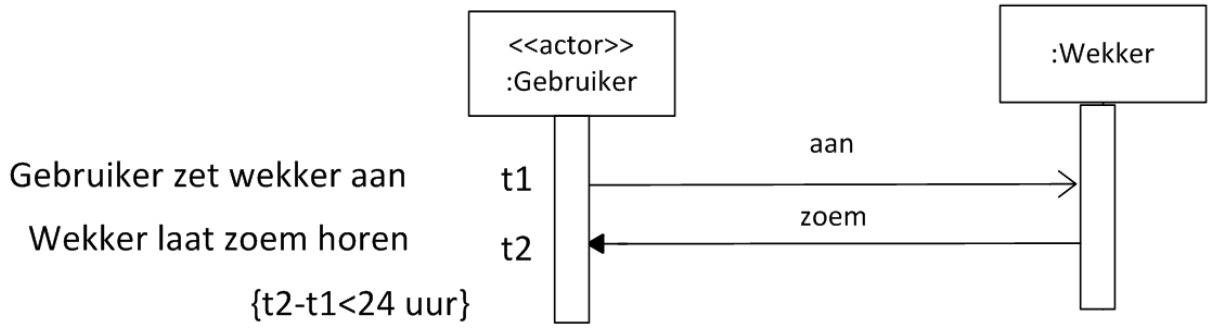


Fig. 5-1 Eenvoudig sequentiediagram

Hoewel een actor zich buiten het systeem bevindt wordt hij in de interactiediagrammen gemodelleerd als een object. In de notatie wordt binnen het actorobject het stereotype <<actor>> geplaatst. Een actor kan ook worden aangegeven door het actorsymbool.

Onderstaande tekening is dus precies hetzelfde als de bovenstaande.

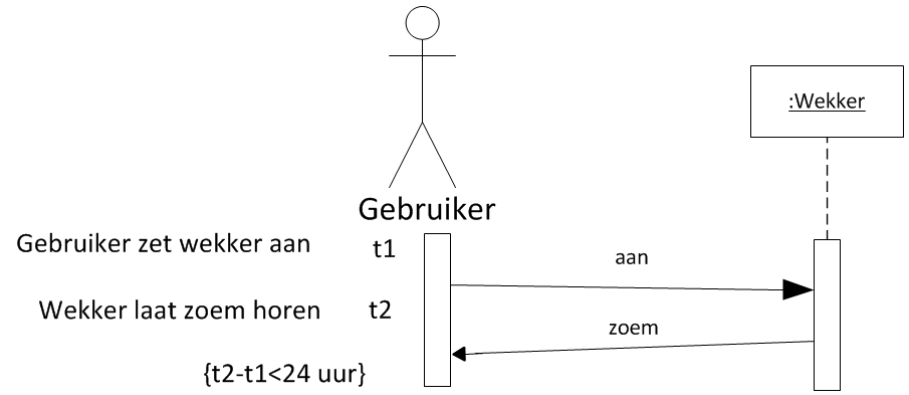


Fig. 5-2 Eenvoudig sequentiediagram

Een sequentiediagram kan als geheel in een frame geplaatst worden. Linksboven in het frame staat dan het keyword *sd*, gevolgd door de naam van het sequentiediagram.

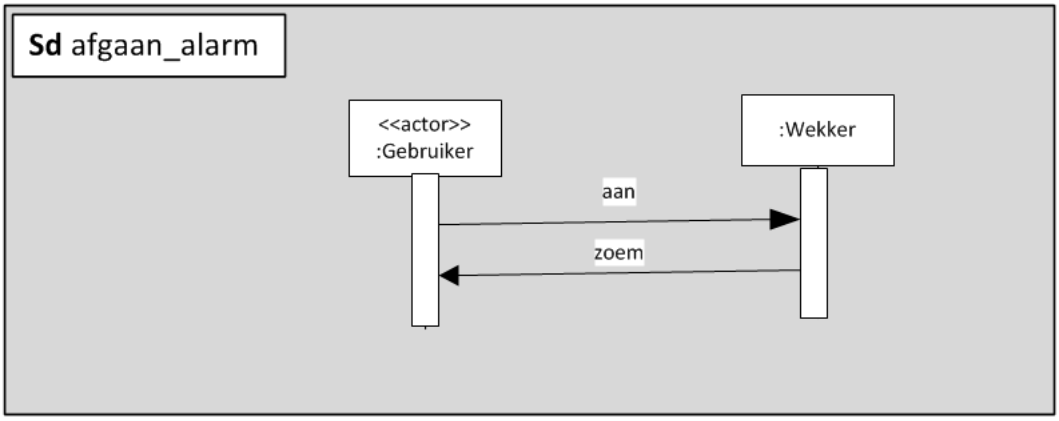


Fig. 5-3 Eenvoudig sequentiediagram in een frame

Een **boodschap** is een stimulus die van een object naar een ander object gestuurd wordt. Voorbeelden van boodschappen zijn operatieaanroepen en het sturen van signalen. Een **interactie** is een serie boodschappen.

Een boodschap wordt in een sequentiediagram weergegeven als een pijl tussen twee levenslijnen met de naam van de boodschap erbij.
In een sequentiediagram is de tijdsvolgorde van de boodschappen van boven naar beneden.

In een sequentiediagram kunnen beperkingen aangegeven worden op het tijdsverloop tussen twee boodschappen. Dit noemen we een **tijdconstraint** (Eng. Timing constraint). Een

beperking op het tijdsverloop kan bijvoorbeeld zijn: maximaal 6 milliseconden tussen twee boodschappen.

Een tijdconstraint wordt in de kantlijn van het sequentiediagram genoteerd. De boodschappen krijgen een merkteken waarnaar in de constraint verwezen kan worden. Deze constraint wordt tussen accoladen geplaatst. In de figuur hierboven staat een simpel voorbeeld. De wekker is hier ten behoeve van het voorbeeld nog als één object beschouwd.

In een sequentiediagram kan aangegeven worden dat een object gedurende bepaalde tijd actief is. De tijdsperiode dat een object een actie uitvoert wordt aangegeven door een levenslijn te laten zien als een witte balk. De tijdperiode dat een object geen actie uitvoert wordt aangegeven door de levenslijn te stippelen. Zolang de actor of het object beschikbaar is, loopt de stippellijn door. In de figuur hiernaast zie je hiervan een simpel voorbeeld.
Het rechterobject ken je als object MijnComputer van de klasse Computer. De pijl vertelt wat er gedaan wordt. In de figuur zet de actor Gebruiker dus MijnComputer aan.

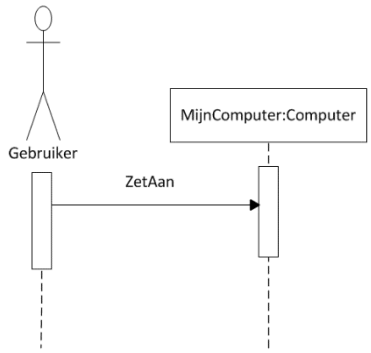


Fig. 5-4 Sequentiediagram met duidelijke levenslijn

5.2 Synchrone en asynchrone boodschap

Boodschappen die een operatie aanroepen, zijn meestal **synchroon**. Dat wil zeggen dat de zender wacht totdat de ontvanger klaar is. Er volgt een terugmelding dat het resultaat bereikt is.
Boodschappen hoeven echter niet altijd op het moment van verzending verwerkt te worden door het ontvangende object. Dergelijke boodschappen noemen we **asynchroon**. Hierbij zendt dus het actieve object een bericht, maar wacht niet op een reactie. Er kan direct een volgend bericht gestuurd worden.

Het verschil tussen synchrone en asynchrone boodschappen wordt aangegeven door middel van de pijlpunt. Een gesloten zwarte pijlpunt (→) geeft een synchrone boodschap aan, een open pijlpunt (→) geeft een asynchrone boodschap aan.

In de onderstaande figuur is ‘geefData’ een synchrone en ‘stuurMail’ een asynchrone boodschap. (Let op, in UML1.3 en eerder werden asynchrone boodschappen door middel van een halve pijlpunt aangegeven. De open pijlpunt was gelijk aan een synchrone aanroep. Deze pijlen zijn nog steeds opgenomen in UML-Tools of bijv. Visio. Maak dus een bewuste keuze. Hiernaast zie je de keuze die Visio biedt.)

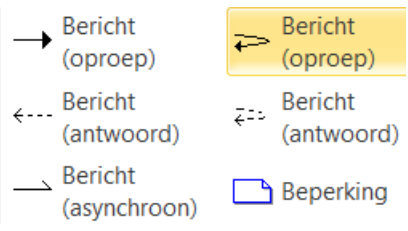


Fig. 5-5 Pijlkeuze in Visio

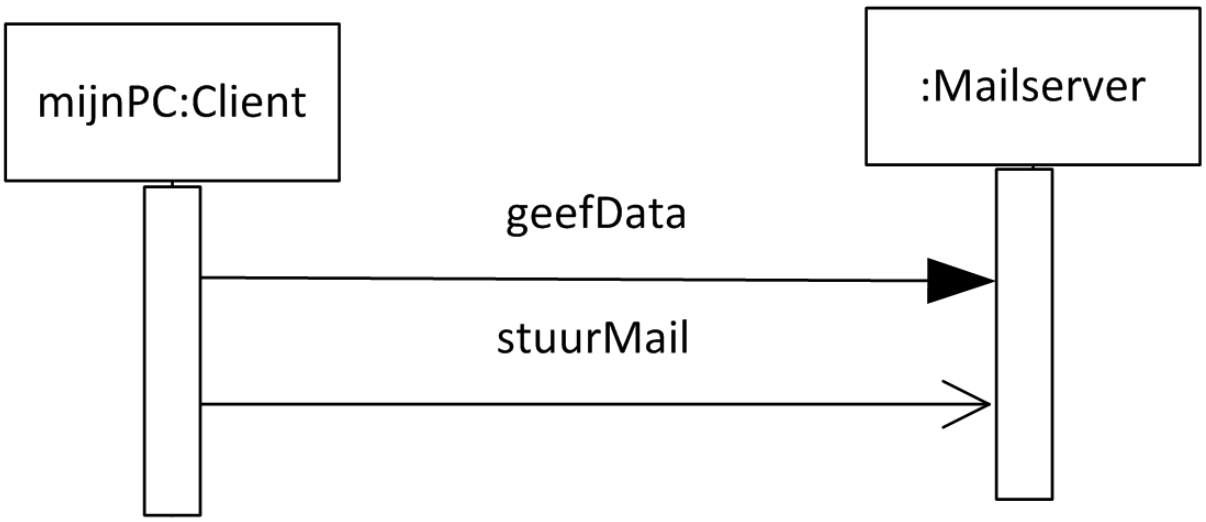


Fig. 5-6 Aanduiding synchroon en asynchroon bericht

Soms is het zinnig om aan te geven dat de besturing vrijgegeven wordt. Meestal is dat vanzelfsprekend. Als we de reactie willen laten zien, kunnen we gebruikmaken van de **Return-pijl**. Dit is een onderbroken lijn met een normale pijlpunt die terugverwijst naar het object waar het bericht vandaan kwam (⇠).
(Zie hieronder bij Iteraties voor een simpel voorbeeld.)

5.3 Tijdgebonden overgangen

Normaal gesproken worden alle berichten als horizontale lijnen aangegeven. Eigenlijk is dit vreemd als we bedenken dat we tijd als verticale as in het activiteitendiagram aangeven. We tekenen dus dat de activering door een actor of object tijd kost, maar het sturen van berichten niet. Vaak is het ook zo dat de bewerking door een object veel meer tijd kost dan het zenden of ontvangen van een bericht. Wat we tekenen is dus niet helemaal juist, maar de afwijking van de werkelijkheid is slechts gering. Als we willen aangeven dat het zenden van een bericht wel tijd kost, dan kunnen we dit doen door de pijl niet horizontaal, maar schuin naar beneden gericht te tekenen. Het is nu duidelijk dat dit bericht echt tijd kost. We doen dit vooral in applicaties die echt tijd kritisch zijn. We geven dan niet alleen aan dat het zenden en ontvangen van een bericht tijd kost, maar ook hoeveel tijd het kost.

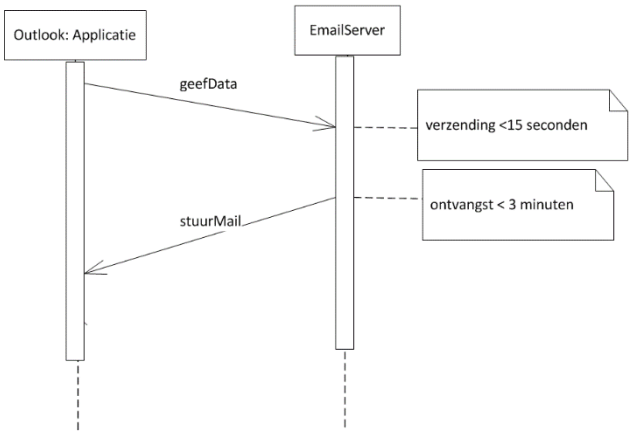


Fig. 5-7 Tijdgebonden acties

Als we kunnen benoemen hoe lang de toegestane tijd is, kunnen we dit toevoegen aan ons sequentiediagram. We hebben dit hierboven al gezien (constraint). Zie hierboven weer voor een voorbeeld.

5.4 Iteraties

Iteraties zijn herhalingen: we doen iets vaker dan één keer. Soms staat van tevoren vast hoe vaak we iets doen, soms gaan we door met herhalen tot een bepaalde toestand bereikt is.

Als we herhaling in een sequentiediagram willen aangeven:

- tekenen we een rechthoek om de te herhalen routine.
- geven we linksboven in de hoek van deze rechthoek het keyword *loop*;
- we plaatsen de voorwaarde (de guard-conditie). Deze begint met een sterretje (*) en vervolgens komt de voorwaarde tussen rechte haken ([en]).

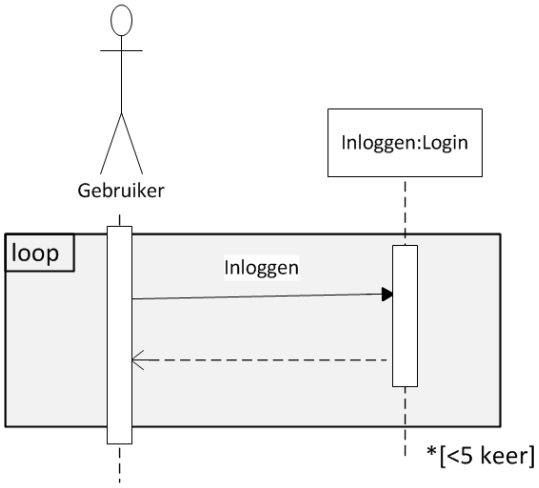


Fig. 5-8 Iteratie

Vraag 5.4.1

Zie bovenstaande figuur: Hoe vaak kan er een inlogpoging gedaan worden?

In een wat uitgebreider sequentiediagram zien we van links naar rechts altijd dezelfde volgorde:

1. De actor
2. De Graphical User Interface (GUI)
3. De klassen die gebruikt worden in dit proces.

Bij het opzetten van een sequentiediagram maken we gebruik van de use-case beschrijving of van enkele door de klant aangegeven scenario's. We willen immers de communicatie met het te ontwerpen systeem eenduidig vastleggen. Het kan gebeuren dat we tijdens het opstellen van een sequentiediagram ontdekken dat de klant onvolledige of soms zelfs onjuiste, tegenstrijdige informatie heeft verstrekt. Dit heeft niets te maken met onwil van de klant, maar vooral met de moeite om precies te formuleren wat er werkelijk gebeurt. Een beschrijving die voor mensen hanteerbaar is, bijvoorbeeld een procedurebeschrijving, zit vaak nog vol met hiaten voor een te ontwerpen systeem.

Bij het maken van de sequentiediagrammen streven we naar een zo goed mogelijke weergave van de werkelijkheid. Elk sequentiediagram moet in ieder geval technisch correct zijn. Ook is het sequentiediagram bedoeld om de communicatie met de klant te ondersteunen. Het vaststellen dat de verstrekte informatie onjuist of onvolledig is, levert voldoende materiaal op om nieuwe vragen aan de gebruiker te stellen. Ook het maken van sequentiediagrammen gebeurt dus iteratief.

Hieronder volgen drie voorbeelden:

Voorbeeld 1

Bij elk bankkantoor staat de geldautomaat de hele dag te wachten op een klant. D klant kan zijn of haar pas invoeren, waarna de automaat zal vragen om de pincode die bij de kaart en rekening van de gebruiker hoort. Nadat de pincode gevalideerd is en de gebruiker toegang is verleend tot het systeem, wordt een menu getoond met twee mogelijkheden; “Geld opnemen” en “Saldo tonen”. Als de gebruiker daarna kiest voor “Saldo tonen” wordt een aantal opties getoond met verschillende bedragen. Na de keuze van een bedrag volgt het uitwerpen van de pas en daarna het uitwerpen van het gekozen bedrag. Ten slotte keert het apparaat terug in de wachttoestand.

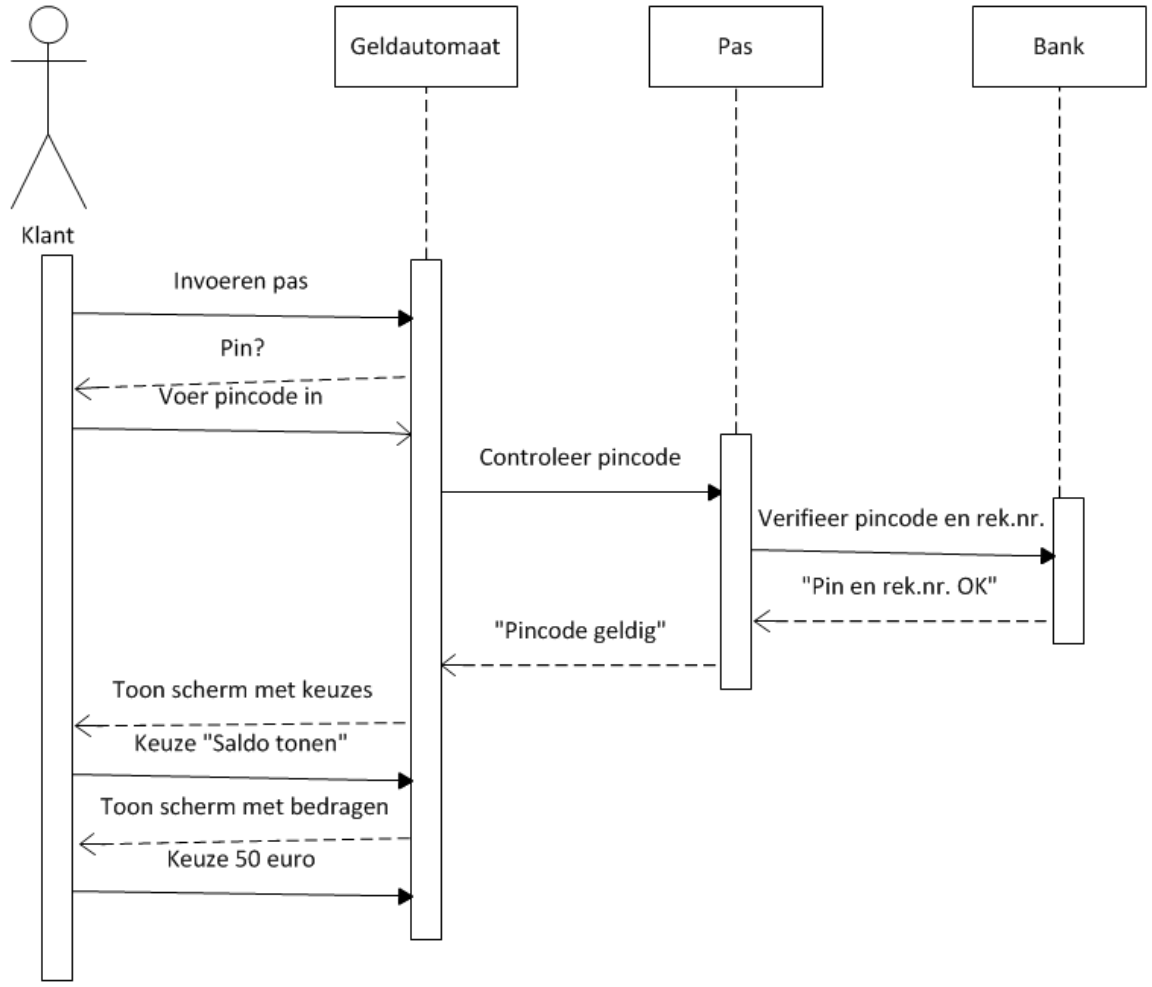


Fig. 5-9 Sequentiediagram met deel van een geldopname bij een geldautomaat.

Voorbeeld 2

Een manager wil een overzicht hebben van alle projecten die in uitvoering zijn. Hij wil weten wie er bij betrokken zijn, hoeveel uur er aan gewerkt is en aan welke producten. Hieronder zie je twee mogelijke uitwerkingen.

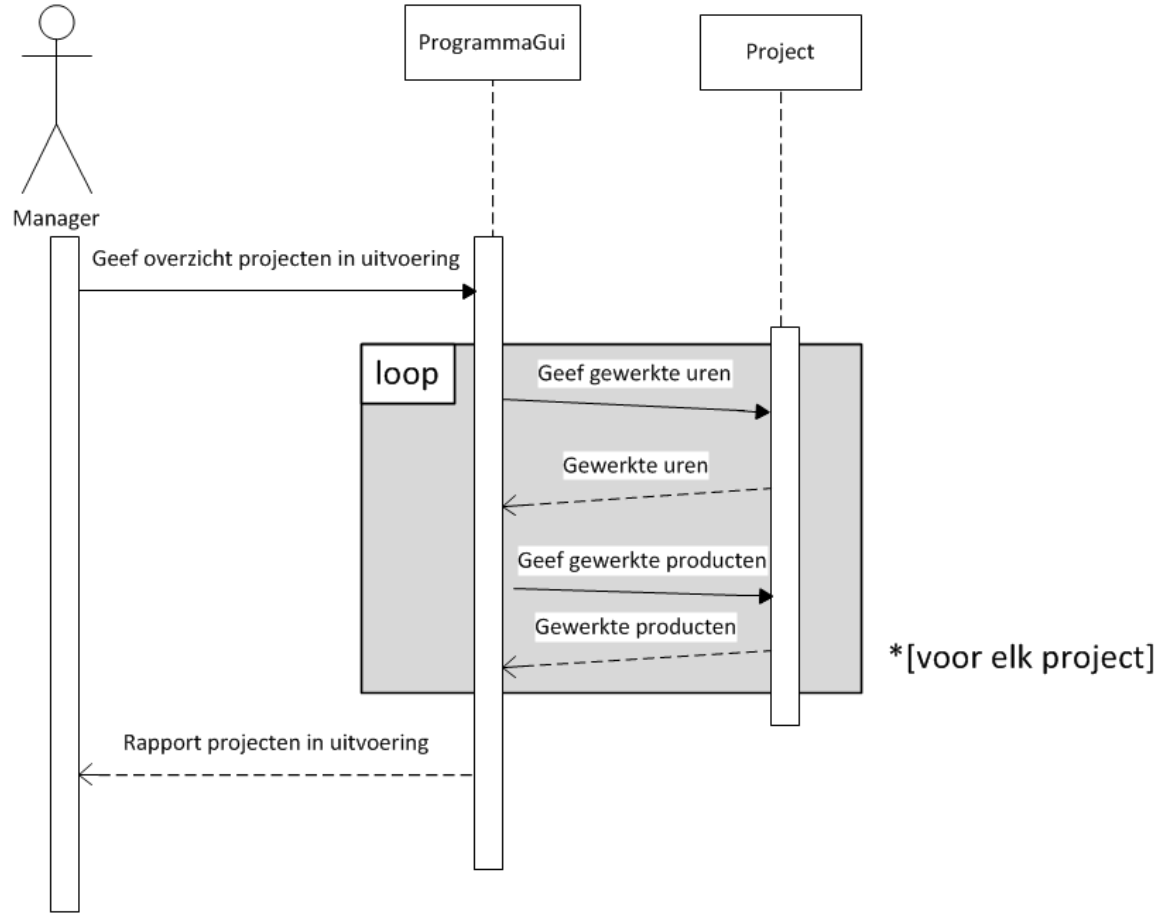


Fig. 5-10 Overzicht van projecten, met een loop grof weergegeven

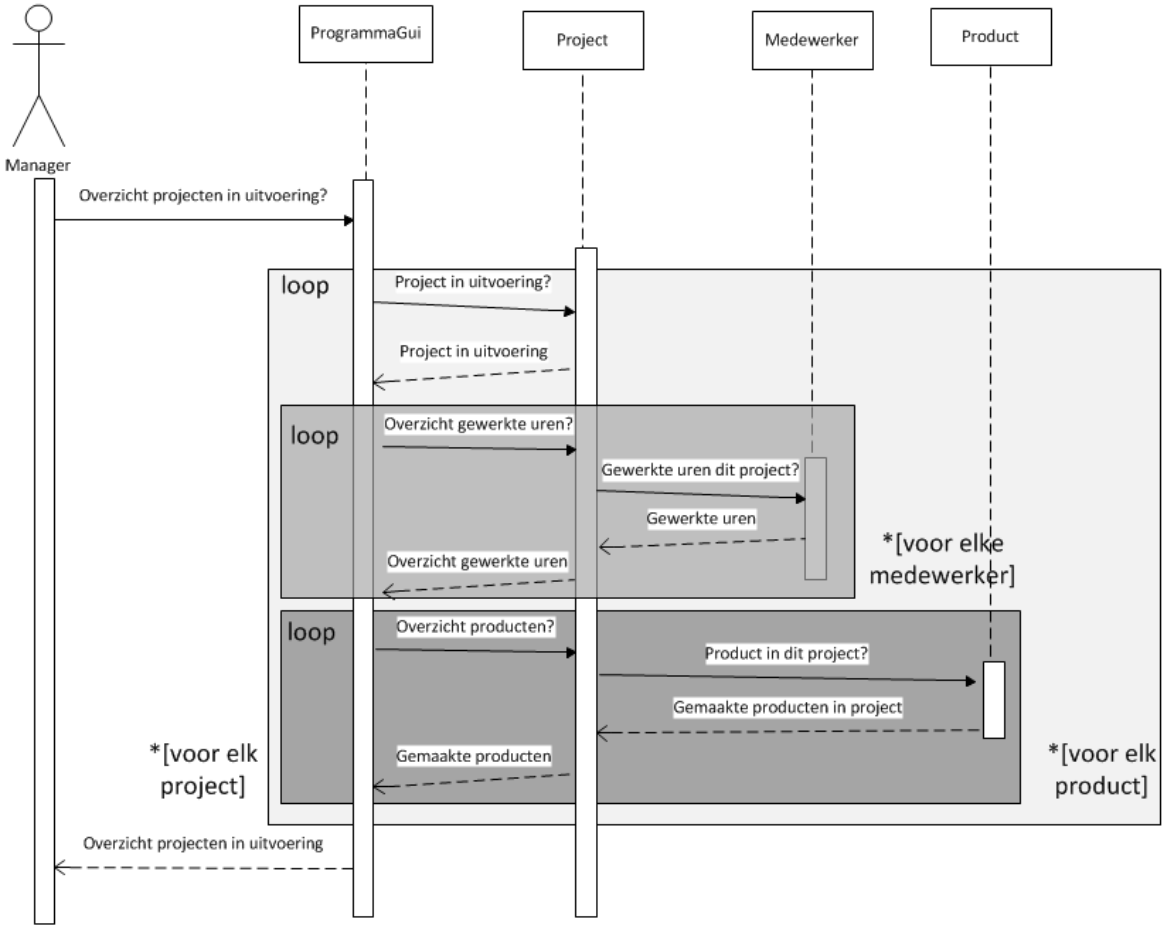


Fig. 5-11 Overzicht van projecten, meerdere loops gedetailleerd weergegeven

Voorbeeld 3

Een bioscoop heeft een aantal zalen waar films worden getoond. Tot een uur voor de aanvang van een filmvertoning kan er door een klant een film gereserveerd worden. Uiteraard kan dit alleen indien er nog plaatsen voor de filmvertoning beschikbaar zijn. Een scenario: De heer Jansen belt op en vraagt of er nog plaatsen beschikbaar zijn voor een bepaalde film. De vraag van de bioscoopmedewerker is welke vertoning de heer Jansen wil zien, dat wil zeggen de datum en het eventuele tijdstip. De klant geeft aan dat het om de donderdagavondvoorstelling gaat. De heer Jansen wil twee plaatsen reserveren. De medeweker geeft aan dat er nog plaatsen beschikbaar zijn op rij 3 en vraagt of hij daar mee akkoord kan gaan. De heer Jansen bevestigt dit en vraagt hoe laat de gereserveerde kaarten moeten worden afgehaald. De medewerker reserveert de kaarten in het systeem en vertelt de heer Jansen dat de gereserveerde kaarten uiterlijk een half uur voor het begin van de voorstelling afgehaald moeten worden. De heer Jansen vraagt wat het verschuldigde bedrag is. Dit wordt door de medewerker doorgegeven, waarna het gesprek wordt beëindigd.

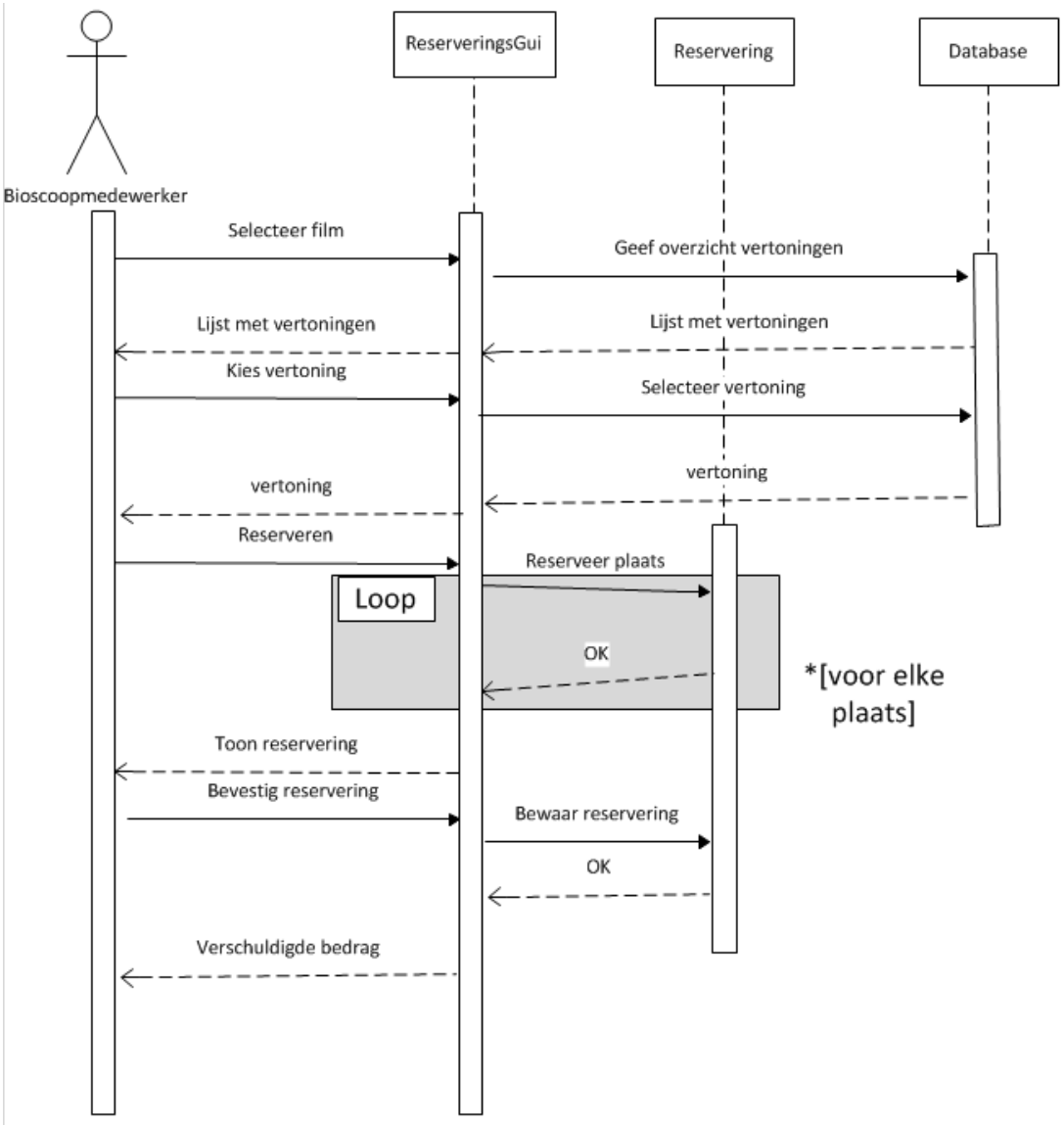


Fig. 5-12 Reserveren van kaartjes voor een film in een bioscoop

5.5 Opdrachten

Opdracht 5.5.1

Zie bijlage 1. (Achteraan in deze reader.)
Wat vind je van deze uitwerking van de volgende context?
Chantal: "Bij een klantgesprek moet ik de mogelijkheid hebben om een bestelling in te voeren. Ik selecteer de klant en vervolgens de gewenste producten. Bij elk geselecteerd product voer ik het gewenste aantal in. Nadat alle producten zijn ingevoerd, kan ik de order bevestigend afsluiten. Het systeem toont een bevestiging van de opslag van de zojuist geplaatste order."

Opdracht 5.5.2

Nadat een gebruiker een munt inwerpt in een snoepautomaat, verschijnt op het scherm van de automaat de waarde van de ingeworpen munt. Pas nadat het bedrag verschijnt, kan de volgende handeling uitgevoerd worden. Teken het sequentiediagram.

Opdracht 5.5.3

Een leraar wil inloggen op het netwerk. Na het invoeren van naam en wachtwoord op het invulscherm, zoekt de netwerkinterface contact met de database met inloggegevens.Blijkt de combinatie naam en wachtwoord te bestaan, dan geeft de database de besturing terug aan de netwerkinterface en de netwerkinterface geeft de toegang tot het netwerk vrij aan de leraar. Teken het sequentiediagram.

Opdracht 5.5.4

Bij het opruimen van de harde schijf besluiten we om alle bestanden die een extensie .mt3 hebben via de Windows Verkenner één voor één te wissen. Teken een sequentiediagram, voeg de herhaling toe en vermeld de 'guard'-conditie.

Opdracht 5.5.5

Maak een sequentiediagram van het invoeren van een klant uitgaande van de template van opdracht 4 van hoofdstuk 2.

Opdracht 5.5.6

Jan: „Voordat ik naar een fabrikant toe ga, wil ik eerst een overzicht van voorgaande gesprekken met de contactpersonen bij deze fabrikant. Ook wil ik een overzicht van de laatst gehanteerde inkoopprijzen”.

Maak hiervan een sequentiediagram.

Opdracht 5.5.7

Maak in Visio sequentiediagrammen van de hoofdfuncties van de volgende context. (Zie De Fietsfabriek blz. 19 en 20.)

Index

Versiebeheer:

Versienummer	Datum	Wijziging
1.0	22-10-2011	Figuurverwijzingen hfdst. 3 verbeterd. Afbeelding objectdiagram aangepast.
2.0	10-01-2012	Fouten eruit gehaald. Aanvullingen gedaan. Opgaven beter genummerd.
3.0	6-9-2015	Alles m.b.t. presentatie van objecten eruit gehaald. Meer afgestemd op toets en Barroc-IT

Bijlage 1 Behorend bij opdracht 5.5.1