Voorwoord

Waarover gaat dit boek?

Als je aan een timmerman vraagt of hij een tafel wil maken, zal zijn reactie waarschijnlijk zijn: waarvoor wil je de tafel gebruiken, wat voor tafel moet het zijn? Een keukentafel, eettafel, salontafel, bijzettafel? Moet de tafel rond zijn of vierkant of een andere vorm hebben? En van welke afmeting moet de tafel zijn? Hoe hoog? Van welk materiaal gemaakt? Uitsluitend hout? Welk houtsoort dan?

Hoeveel poten moet de tafel hebben en wat moet de vorm van de poten zijn? Op welke manier moeten ze bevestigd worden? En hoe zit het met de afwerking van de tafel?

Dit zijn veel vragen en er zijn er veel meer. In dit geval gaat het alleen nog maar om een tafel, een begrip waarvan iedereen de functie kent.

Als een klant aan een programmeur vraagt een programma te schrijven, rijzen er minstens zoveel vragen. Het inventariseren van die vragen en het verkrijgen van de antwoorden erop is een belangrijke fase in de ontwikkeling van een programma. Zonder die fase kun je niet weten welk programma je moet schrijven, net zomin als een timmerman zomaar weet welke tafel hij moet maken.

Het uitschrijven van zogeheten *use cases* is een belangrijk hulpmiddel om achter de verschillende functies van een applicatie te komen. In het volgende hoofdstuk kun je daar meer over lezen.

Timmerlieden werken vaak met tekeningen om de hele constructie of een deel daarvan goed te kunnen begrijpen en te maken. Op dezelfde manier werkt een programmeur met tekeningen van de applicatie: zogeheten *UML-diagrammen*. UML staat voor Unified Modeling Language; het woord language duidt erop dat het om een taal gaat. UML is geen gewone taal, maar een *beeldtaal* waarmee je met behulp van verschillende soorten diagrammen telkens andere aspecten van een applicatie kunt verduidelijken. Een use case kun je vaak goed in beeld brengen met een *activiteitendiagram*.

Andere veelgebruikte UML-diagrammen zijn *klassendiagrammen*, die in hoofdstuk 3 en 4 uitvoerig aan bod komen. Een ander soort UML-diagram is een *sequentiediagram*, waarover je in hoofdstuk 5 kunt lezen.

Als je een product maakt, moet je het *testen* om te zien of het aan alle eisen voldoet, of het goed in elkaar zit en goed functioneert. Dat geldt voor een tafel en zeker voor een applicatie. Applicaties kunnen heel omvangrijk en complex worden. Het is belangrijk al vroeg in de ontwikkeling met testen te beginnen, zodat fouten snel aan het licht komen. Over testen gaat hoofdstuk 6.

Een tafel is een relatief eenvoudig product en de constructie ervan is overzichtelijk. Het maken van software is in het algemeen geen proces dat in een rechte lijn van begin naar eind loopt. Terwijl de applicatie groeit, groeit meestal ook het inzicht van de programmeur. In hoofdstuk 7 kun je een voorbeeld van zo'n ontwikkelproces vinden.

Veel programmeerproblemen staan niet op zichzelf maar blijken bij de ontwikkeling van meer applicaties voor te komen. Met enige moeite kun je daarin patronen herkennen, en ook de manier waarop je de software inricht om die problemen op te lossen kan een patroon vormen, een zogeheten *ontwerppatroon* (design pattern). Hoofdstuk 8 en 9 gaan over ontwerppatronen.

Het laatste hoofdstuk gaat over *persistentie*, de opslag van de gegevens waar een toepassing gebruik van maakt.

Welke voorkennis moet je hebben?

Om dit boek met succes te kunnen begrijpen, moet je een beetje kennis hebben van Java. Het is handig als je weet wat een klasse is, wat een object en wat het verschil is tussen de twee. Het is handig als je weet dat een object gegevens kan bewaren in zijn attributen en dat je door het aanroepen van een methode een object iets kunt laten doen.

Ik ga ervan uit dat je een basiscursus Java hebt gevolgd en wel eens een Java-programma hebt geschreven. Van begrippen als polymorfie en dynamische binding heb je misschien wel eens gehoord, maar als je ze niet precies kunt plaatsen, is dat niet erg, in hoofdstuk 4 worden ze uitgelegd.

UML en broncode

In dit boek leer je wat UML is aan de hand van praktische voorbeelden. In de meeste boeken over UML komt geen regel code voor. Dat is merkwaardig omdat UML uiteindelijk juist een hulpmiddel is bij het maken van programmacode. Ik besteed in dit boek dan ook veel aandacht aan de omzetting van UML-diagrammen naar broncode.

Supplementair materiaal

Op de begeleidende website, www.pearsoneducation.nl/laan, vind je de broncode uit dit boek. Voor docenten is er extra materiaal beschikbaar. Neem hiervoor contact op met de uitgever via docent@pearson.com.

Dankwoord

Graag wil ik de meelezers Clem Cornelis en Arie Toet bedanken voor hun opbouwende commentaar en nuttige ideeën en aanbevelingen. Zij hebben een kwalitatieve bijdrage geleverd aan de realisatie van dit boek.

Gertjan Laan
Oostzaan, maart 2007