# Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2023, gehouden op donderdag 30 maart 2023 jl. en georganiseerd door NHL Stenden Hogeschool). Bij elkaar bijna 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats op donderdag 27 maart 2025 in Zwolle en wordt dan georganiseerd door Hogeschool Windesheim. Kijk op www.nioc2025.nl voor meer informatie.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

# Java in het inleidend programmeeronderwijs; terugblik en reflectie

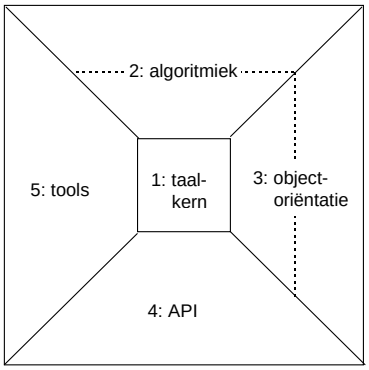
Peter Kluit, Marleen Sint en Frank Wester

## 1 Inleiding

Een jaar of drie geleden wilden wij als docenten van de Open Universiteit en de Technische Universiteit Delft het inleidend programmeeronderwijs vernieuwen. Een belangrijke doelstelling daarbij was om in elk geval de instapcursus aantrekkelijk te maken voor een grotere doelgroep dan alleen informaticastudenten. Verder wilden we zo snel mogelijk beginnen met object-georiënteerd programmeren. En last but not least: we wilden dat studenten programmeren weer leuk gingen vinden, iets wat in de laatste jaren daarvoor steeds minder het geval bleek.
We dachten deze doelstellingen te kunnen verwezenlijken door gebruik te maken van de (toen nog nieuwe) taal Java en van een visuele programmeeromgeving waarin op eenvoudige wijze programma's ontwikkeld kunnen worden met een grafische gebruikersinterface; zie voor onze argumentatie Wester (1996) en Wester e.a. (1997).
Uitgaande van deze vrij onconventionele aanpak besloten de twee instellingen samen drie nieuwe cursussen te ontwikkelen: Visueel programmeren met Java, Objectgeoriënteerd programmeren met Java en Datastructuren en algoritmen. Nu de ontwikkeling van deze cursussen vrijwel is afgerond en de nodige ervaring met het materiaal is opgedaan in het onderwijs aan eigen studenten (en bij veel andere onderwijsinstellingen), is het een goed moment om terug te blikken en na te gaan wat we zoal geleerd hebben en wat we wel en niet hebben kunnen waarmaken van onze ambities.
Terugkijkend op het proces van de ontwikkeling van deze cursussen, blijkt dat we voortdurend hebben moeten kiezen wat we wel en vooral ook wat we niet in het onderwijs wilden opnemen en daarnaast in welke volgorde we de gekozen onderwerpen wilden aanbieden. Ten opzichte van de oude situatie bleek het aanbod van wat je studenten kunt onderwijzen veel groter te zijn dan vroeger.

## 2 Een kader

Het gebruik van Java als taal voor inleidend programmeeronderwijs dwingt tot keuzes. De taal zelf is veel rijker dan een taal als bijvoorbeeld Pascal. Daarbij komen dan nog de noodzaak van het uitleggen van objectgeoriënteerde concepten en technieken, de enorm grote Java-programmabibliotheek (beter bekend als API) en de fraaie mogelijkheden die visuele programmeeromgevingen bieden.




Om deze keuzes beter te kunnen beschrijven en visualiseren, gebruiken we een kader als getoond in figuur 1. We maken onderscheid tussen vijf aspecten die in het programmeeronderwijs aan bod kunnen komen: de taal Java zelf, algoritmiek, objectoriëntatie, de API en tools. We omschrijven nu eerst elk van deze vijf deelgebieden. Zoals uit die beschrijvingen zal blijken, omvat een deelgebied het maximaal haalbare binnen *inleidend* programmeeronderwijs op dat gebied, en dus niet het totaal van hetgeen er op dat deelgebied te leren valt.

Figuur 1: Visualisatie van de vijf deelgebieden van het gebruikte kader

### *2.1 De taalkern van Java*

Met deze taalkern bedoelen we het kunnen omgaan met de taalconstructies van Java zoals bijvoorbeeld beschreven in het standaardwerk van Arnold en Gosling (1996). Ook aspecten als het gehanteerde geheugenmodel, typering en parametermechanismen horen hier bij. Uiteraard zijn er directe raakvlakken met objectoriëntatie (alle constructies die direct met klassen en instanties te maken hebben) en algoritmiek (constructies voor keuze- en herhalingsopdrachten e.d.). In dit kader rekenen we de taalconstructies als zodanig tot de taalkern, terwijl we de vaardigheid om er mee om te gaan tot de deelgebieden rekenen.

Bij een taal als Pascal was het goed mogelijk om alle taalelementen in een inleidende cursus te behandelen. De taal Java biedt echter duidelijk meer mogelijkheden: zo zal je in een inleidende cursus niet toe kunnen komen aan het programmeren met behulp van threads. Ook exception handling, interfaces en abstracte klassen komen al gauw in een vervolgcursus terecht.

*2.2 Algoritmiek*

Bij het onderwijs in algoritmiek onderscheiden we twee niveaus (zie stippellijn in figuur 1):
a het leren ontwerpen en implementeren van relatief eenvoudige algoritmen door gebruik te maken van de taalconstructies voor herhalingen, keuzes, arrays, e.d.
b het leren kennen, begrijpen en toepassen van de bekende datastructuren (lijsten, grafen, bomen, …) en de algoritmen die op deze datastructuren werken.
Het eerste niveau is een must voor iedereen die eenvoudige programma's wil kunnen maken. De onderwerpen uit het tweede niveau zijn in het algemeen vooral belangrijk voor studenten informatica. Een interessante vraag is of je het onderwerp recursie tot het eerste of tot het tweede niveau moet rekenen.

*2.3 Objectoriëntatie*

Ook bij onderwijs in objectoriëntatie (OO) maken we een onderscheid in twee niveaus (zie stippellijn):
a Studenten moeten elementaire objectgeoriënteerde begrippen leren en kunnen toepassen. In eerste instantie gaat het om basisbegrippen als: wat is een klasse, een instantie, een attribuut, een methode, een methode-aanroep en hoe werkt overerving (inclusief dynamische binding). Daarnaast moet bij studenten ook enig begrip worden ontwikkeld van objectgeoriënteerd modelleren. Zo zul je in het inleidende onderwijs zo mogelijk wat aandacht willen besteden aan een modelleertaal, waarbij op dit moment de keuze voor UML (Unified Modeling Language) voor de hand ligt.
b Ervaren OO-programmeurs maken vaak gebruik van z.g. ontwerppatronen (design patterns). Het zijn oplossingen voor vaak voorkomende ontwerpbeslissingen waarmee (grote) OO-programma's een meer transparante structuur kunnen krijgen. Een standaardwerk dat een overzicht biedt van dergelijke ontwerppatronen is dat van Gamma e.a. (1995). Het is niet mogelijk om veel standaardpatronen al in het inleidend programmeeronderwijs te behandelen, maar als je kiest voor OO-onderwijs kun je het ook niet laten liggen. Ook een goed begrip van de constructie van klassehiërarchieën en het goed gebruiken van interfaces rekenen we tot dit hogere niveau.

*2.4 API*

De eerste versie van de JDK (Java Development Kit) had nog een standaard API (Application Programming Interface) bestaande uit 8 packages (java.applet, java.awt, java.image, java.peer, java.io, java.lang, java.net, java.util) met minder dan 230 klassen en interfaces. Bij JDK1.1 was dit aantal gegroeid tot 22 packages en bij de huidige versie Java 2 gaat het om een nog veel groter aantal. Veel van deze packages en klassen maken het mogelijk om snel leuke en zinvolle toepassingen te maken wat studenten motiverend vinden en waardoor je als docent al snel te horen krijgt: waarom doen we niets met de image (of security, rmi, beans, …) packages? Een gerichte keuze wat je wel en niet wilt gebruiken in je onderwijs, is lastig. Belangrijk is wel dat studenten wordt geleerd zelf hun weg te vinden in de wirwar van API-klassen en -interfaces.

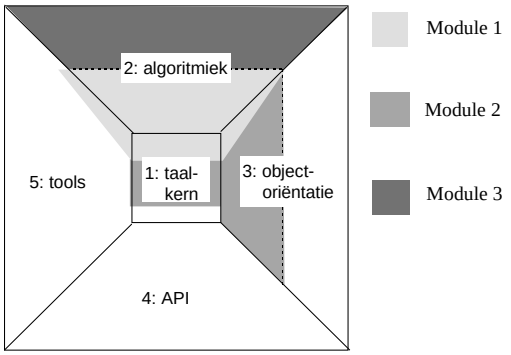
*2.5 Tools*

Moderne programmeeromgevingen bevatten fraaie tools die het programmeren sneller en aangenamer kunnen maken. Behalve het feit dat, zeker voor het maken van gebruikers-interfaces, veel minder code hoeft te worden ingetypt, kan de omgeving gebruikt worden om gemakkelijk toegang te krijgen tot de API, om eenvoudiger overzicht te houden over de ontwikkelde klassen binnen een project, enzovoorts. Daar tegenover staat overigens dat het goed gebruiken van een tool vereist dat je het probleem begrijpt waar het tool je bij helpt. Met name onervaren studenten hebben soms moeite te begrijpen dat de omgeving het projectbeheer voor hen doet, omdat ze het probleem projectbeheer niet kennen.
Ook andere tools en technieken kunnen binnen het onderwijs aan bod komen, zoals het inpassen van Java-applets in web-pagina's, tools voor het koppelen van databases aan programma's, het presenteren van databasetabellen en het gebruik van JAR's (om Java-klassen te kunnen

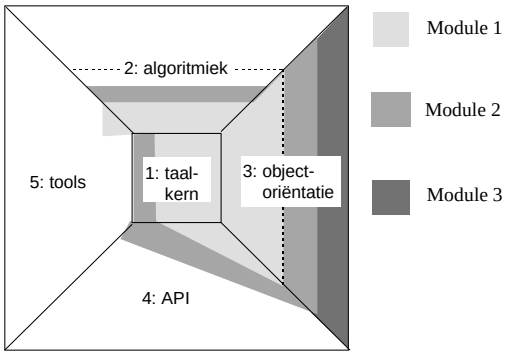
archiveren en versturen). Studenten vinden dit vaak leuk, maar het kost allemaal tijd en kan ten koste gaan van andere, meer fundamentele aspecten van het programmeeronderwijs.

**3 Mogelijke trajecten**

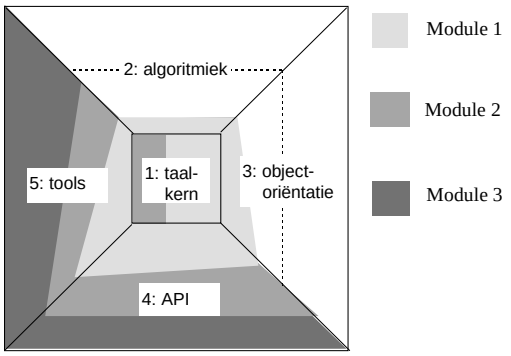
De figuren 2 tot en met 5 tonen in vogelvlucht vier mogelijke Java-trajecten, allemaal gebaseerd op drie inleidende vakken van elk drie studiepunten. Alle trajecten samen vullen het hele gebied. De aanduidingen in de figuren zijn zuiver kwalitatief; onze bedoeling reikt niet verder dan het uitzetten van een paar mogelijke richtingen en het aangeven van de valkuilen die zich daarbij voordoen.



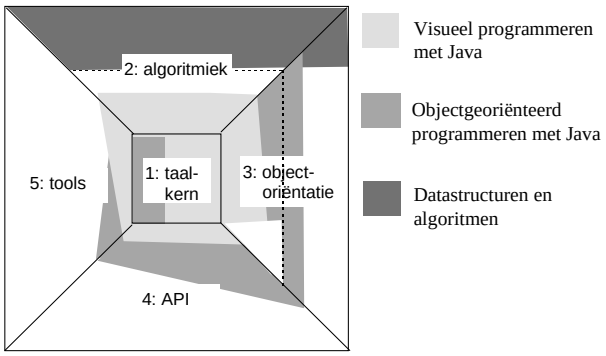

Figuur 2: Klassiek traject




Figuur 3: OO-traject




Figuur 4: Fun-traject




Figuur 5: Compromis-traject

*3.1 Klassiek traject*

De eerste inspanningen op het gebied van Java-onderwijs waren veelal directe afgeleiden van het oudere, procedureel gerichte onderwijs. In een extreme vorm kan een dergelijk traject er uitzien als getoond in figuur 2.

De basiscursus beperkt zich tot statische, niet-objectgeoriënteerde Java-programma's; in dat kader worden de nodige algoritmische vaardigheden bijgebracht en de bijbehorende taalconstructies van Java behandeld. In een vervolgcursus kunnen dan de objectgeoriënteerde concepten aan de orde komen, en in een derde cursus tenslotte datastructuren en algoritmen (als gekozen wordt voor Java, dan zal de behandeling van datastructuren logischerwijs steunen op objectoriëntatie). Een deel van de taalkern (denk aan threads) zal mogelijk in het geheel niet behandeld worden.

Dit traject zal zelden in deze extreme vorm worden aangetroffen. Het vormt echter wel een belangrijke referentie, omdat het niet meer stof behandelt dan een oude trits van CS1/CS2 gevolgd door een als gevorderd beschouwd OO-vak. In een iets minder extreme vorm is het boek Java Gently (Bishop, 1998) een goed voorbeeld van deze benadering (zonder het derde vak).

Het grote nadeel van dit traject is, dat het weinig gebruik maakt van de specifieke mogelijkheden van Java en daarom voor studenten niet bijzonder aantrekkelijk is.

*3.2 OO-traject*

In het traject getoond in figuur 3, worden de basisbegrippen van objectoriëntatie snel en volledig behandeld. In de vervolgcursussen ligt de nadruk veel meer op gevorderde OO-concepten dan op algoritmiek, de API of tools. Dit traject heeft als nadeel dat het relatief weinig aandacht besteedt aan basis-algoritmische vaardigheden. In de praktijk heeft men daar, zeker op universiteiten, niet zoveel last van; de meeste studenten hebben al eens geprogrammeerd (bij de TU Delft bijvoorbeeld geldt dat voor 80 % van de eerstejaars) en beschikken daardoor toch al over de nodige handigheid. Studenten die echt van de grond af aan moeten beginnen, kunnen bij een dergelijk traject echter gemakkelijk uit de boot vallen, zeker omdat de gevorderde OO-concepten vrij zware kost vormen.

### *3.3 Fun-traject*

Een dergelijk traject is er vooral op gericht om de studenten zoveel mogelijk van Java te laten zien; de nadruk ligt dan sterk op het gebruik van veel API-packages en tools. Dit zal vrijwel altijd gaan ten koste van een echt begrip van algoritmiek en ook van OO. Studenten die zichzelf op deze wijze in Java hebben geschoold, zullen typisch een leuke applet van het net halen en daar vervolgens net zo lang aan knutselen tot deze aan hun behoefte is aangepast, vaak zonder veel benul van de achterliggende principes. Het zal duidelijk zijn dat wij geen pleitbezorgers zijn van dit traject. Studenten (en dan vooral degenen die al aardig kunnen programmeren en leuke dingen willen maken) oefenen echter een sterke druk in deze richting uit. Wil een cursus voor hen aantrekkelijk zijn, dan zal in elk geval voor een deel aan hun behoefte tegemoet gekomen moeten worden.

### *3.4 Compromis-traject*

In de praktijk zal er gezocht worden naar een compromis. Figuur 5 toont het compromis uit onze cursussen. De eerste twee cursussen proberen zoveel mogelijk recht te doen aan Java en objectoriëntatie; de derde cursus is een klassieke cursus datastructuren en algoritmen, maar dan geïllustreerd aan de hand van een OO-taal. Een dergelijk compromis heeft veel voordelen. Er komen voldoende fundamentele zaken aan bod, en er wordt net genoeg aan de API en tools gedaan om de studenten enthousiast te maken. Uit de manier waarop we de figuur nu getekend hebben, blijkt echter ook het grootste nadeel: er is wat weinig oefening in het schrijven van eenvoudige algoritmen en het maken van eenvoudige OO-ontwerpen. Vooral bij studenten zonder voorkennis op het gebied van programmeren schieten deze vaardigheden dan ook tekort.

## 4 Drie cursussen

In deze paragraaf geven we een kort overzicht van de inhoud van onze drie cursussen.

### *4.1 Visueel programmeren met Java*

Over de inhoud van Visueel programmeren met Java hebben wij reeds eerder geschreven (Wester, 1997; Kluit 1998). De cursus was (zeker bij verschijnen) in twee opzichten vernieuwend: er wordt vanaf het begin gebruik gemaakt van een visuele omgeving, en er wordt vanaf het begin objectgeoriënteerd gewerkt. Deze opzet wordt mogelijk gemaakt door een onconventionele opbouw. In een eerste blok (35 % van de studielast) leren de studenten omgaan met de visuele omgeving en leren ze de kernbegrippen van objectoriëntatie. Ze passen hun kennis toe middels het schrijven van eenvoudige event handlers, waarin berichten naar (door de omgeving gecreëerde) objecten worden gestuurd. Pas in het tweede blok gaan de studenten zelf klassen definiëren en volledige methoden schrijven. In dit blok worden ook de representatie van objecten in het geheugen en de gevolgen daarvan voor toekenning en parameter-overdracht behandeld. De helft van dit blok is gewijd aan het bijbrengen van basisvaardigheden op het gebied van algoritmiek en de bijbehorende taalconstructies.
In vrijwel elk hoofdstuk worden naast het behandelen van de theorie een of meer Java-applets gemaakt. In het algemeen is gekozen voor toepassingen die een grote doelgroep aanspreken zoals programma's voor het berekenen van een annuïteitenhypotheek of het bepalen van inkomstenbelasting, het omrekenen van vreemde valuta's, en het construeren van magische vierkanten.

### *4.2 Objectgeoriënteerd programmeren met Java*

In de tweede cursus wordt vrijwel alles van de taalkern behandeld wat in de eerste cursus geen plaats had gekregen: abstracte klassen, interfaces, applicaties, exception handling en threads

(synchronisatieproblemen worden echter alleen aangestipt). Aan algoritmiek wordt in deze cursus geen expliciete aandacht meer besteed. Op OO-gebied wordt wel heel wat gedaan.
De cursus begint met een inleiding in objectgeoriënteerd ontwerpen, met enkele diagram-technieken uit UML (Unified Modeling Language). Daarna volgt een grondige behandeling van overerving en dynamische binding. Er wordt één ontwerppatroon behandeld, en wel MVC (klasse Observable en interface Observer). Op het gebied van de API besteedt de cursus aandacht aan graphics en containers (dat lag voor de hand wegens het werken met een visuele omgeving), aan event handling, aan I/O met behulp van tekstbestanden en aan de koppeling met een database (dit laatste op uitdrukkelijk verzoek van veel studenten). Op het gebied van tools wordt alleen de debugger van Visual Café behandeld.
In de loop van de cursus blijkt bovendien een bepaald programmeerpatroon steeds terug te komen (bij graphics, bij threads, bij observable en observer, bij event handling): de applicatie-programmeur herdefinieert of implementeert een methode, maar roept deze nooit aan; dit gebeurt (dankzij dynamische binding) vanuit de betreffende API. Op deze manier konden we de API gebruiken om een belangrijk OO-principe te laten zien, dat we *programmeren per contract* hebben genoemd. Dit concept loopt als een rode draad door de cursus.
Ook in deze vervolgcursus is bij de keuze van toepassingen bewust rekening gehouden met een brede doelgroep. Zo krijgen studenten o.a. te maken met programma's voor het tekenen van figuren, met een girorekening en met een spelletje met een grafisch interface waarin een held een aantal vijanden moet verslaan. In deze laatste toepassing wordt veel van de geleerde theorie toegepast.

### *4.3 Datastructuren en algoritmen*

Het vak dat in de Amerikaanse programmeercurricula bekend staat als Computer Science 2 (CS2), komt in onze curricula pas als derde vak aan bod. In deze cursus, die anders dan de twee eerste cursusssen vooral bedoeld is voor informaticastudenten, worden de bekende datastructuren behandeld, met de standaardalgoritmen die op deze datastructuren werken. Daarnaast worden sorteeralgoritmen, en algemene algoritmische technieken als verdeel-en-heers-technieken, greedy algoritmen en dynamisch programmeren behandeld. Van alle algoritmen wordt ook de complexiteit nader bekeken. Omdat recursie nog niet in de eerste twee cursussen is behandeld, is hier apart tijd voor ingeruimd.
Voor deze cursussen wordt gebruik gemaakt van een bestaand Engelstalig tekstboek. In Delft is het vorige cursusjaar gewerkt met Weiss (1998), dat nu vervangen is door Bailey (1998).
Bij de Open Universiteit is gekozen voor het boek van Goodrich & Tamassia (1998) waar een werkboek (Wester e.a., 1999) bij is ontwikkeld. Het boek heeft een sterk objectgeoriënteerde benadering (inclusief de behandeling en toepassing van enkele eenvoudige ontwerppatronen) en sluit in die zin goed aan op de twee eerste cursussen. Algoritmen worden lang niet altijd in Java gepresenteerd; vaak is een presentatie in de vorm van pseudocode illustratiever.
Het aantal boeken over D&A dat gebaseerd is op Java, is in het afgelopen jaar sterk gegroeid; een bespreking van een aantal van deze boeken is te vinden in het JavaWorld-artikel van Vanhelsuwé (1998).

## 5 Ervaringen

De eerste ervaringen met de cursus Visueel programmeren met Java zijn gepubliceerd in Kluit e.a. (1998). We vatten deze kort samen en voegen de recent verkregen ervaringen met de vervolgcursussen er aan toe.

### *5.1 Ervaringen aan de Open Universiteit*

Bij de Open Universiteit maken de eerste twee cursussen deel uit van de propedeuse; de D&A-cursus is geplaatst in het basisdoctoraal van de informatica-opleiding. Behalve aan opleidings-studenten is de cursus Visueel programmeren met Java ook in grote aantallen verkocht aan studenten die niet een volledige opleiding ambiëren: sinds november '97 in totaal aan ongeveer 1700 studenten. De vervolgcursus is er sinds januari '99 en de cursus Datastructuren en algoritmen komt pas in het najaar beschikbaar. Voor de opleidingsstudenten is er voor de eerste twee cursussen een aanvullend practicum. Alle cursussen worden bij de OU via zelfstudie gedaan met aanvullende groepsbegeleiding voor de opleidingsstudenten; voor alle studenten is er ondersteuning middels een website per cursus met bijbehorende nieuwsgroep waar veel gebruik van wordt gemaakt.

Zowel Visueel programmeren met Java als Objectgoriënteerd programmeren met Java worden door de studenten ervaren als erg leuk en zinvol, maar ook vrij moeilijk en tijdrovend. Aan het eind van deze twee cursussen schieten met name bij studenten zonder voorafgaande programmeerervaring, de algoritmische vaardigheid en de vaardigheid in het opstellen van een eenvoudig OO-ontwerp tekort. Gezien de positionering van deze twee cursussen in het kader (zie paragraaf 3.4) is dit achteraf eigenlijk niet verbazingwekkend. Wel zou verbetering kunnen optreden door onervaren studenten een vorm van prakticumbegeleiding aan te bieden; juist zij slagen er vaak niet in om praktische opdrachten zelfstandig tot een goed einde te brengen.
De ervaringen met de cursus Datatstructuren en algoritmen beperken zich tot een groep proefstudenten. Zij vonden de cursus boeiend, leuk en pittig en waarderen het tekstboek en de gekozen werkvorm (huiswerkopgaven). Wel geven ze aan dat ze meer Java hadden verwacht en dat ze behoefte hebben aan een practicum (dat wordt voorzien) dat de kennis van Objectgeoriënteerd programmeren en Datastructuren en algoritmen integreert.

*5.2 Ervaringen in Delft*
Bij de opleiding informatica aan de TU Delft worden de cursussen gebruikt in het eerste jaar van de opleiding informatica. De cursus Visueel programmeren met Java werd voor het eerst gebruikt in september 1997, de cursus Objectgeoriënteerd programmeren met Java in november 1998. De cursus Datastructuren en algoritmen gaat in Delft in april-mei 1999 voor het eerst in deze vorm gegeven worden. We beperken ons hier dus tot de evaluatie van de eerste twee cursussen.
Beide cursussen werden geëvalueerd met behulp van de uniforme cursusevaluatie van de TU Delft. Enerzijds maakt dat vergelijking met andere vakken mogelijk, anderzijds kunnen op deze wijze geen vragen specifiek over dit vak worden gesteld.
Met betrekking tot waardering voor het vak als geheel en voor de kwaliteit van het materiaal scoren beide cursussen erg goed in verhouding tot andere vakken binnen de opleiding en binnen de TU. Toch is er een niet onaanzienlijke groep mensen die het gevoel hebben over onvoldoende voorkennis te beschikken, die het vak moeilijk vinden en de studielast te hoog. Omdat ook studenten zonder programmeerervaring kunnen uitgroeien tot waardevolle informatici, zoeken wij naar maatregelen om in de toekomst juist deze groep beter te begeleiden.
Het practicum bij de tweede cursus zorgde voor problemen. We vroegen de studenten (onder meer) zelf een ontwerp te maken voor een niet al te eenvoudig programma, en dat vervolgens te programmeren. Bijna de helft van de studenten slaagde er niet in dit ontwerp te maken, of durfde er zelfs niet aan te beginnen. Degenen die het ontwerp wel konden maken, slaagden er als regel ook in het te implementeren. Wij hebben de mogelijkheden van studenten om zelf actief te ontwerpen in deze fase kennelijk overschat.

*5.3 Ervaringen elders*
De cursus Visueel programmeren met Java wordt ook aan veel andere opleidingen (hbo en universiteit) gebruikt; in '98 werden meer dan 1700 cursussen verkocht aan circa 15 opleidingen. Een aantal van deze opleidingen gebruikt ook de vervolgcursus of overweegt dit te gaan doen. Begin april j.l. vond in Utrecht een bijeenkomst plaats van docenten die de cursus Visueel programmeren met Java gebruiken, of dat van plan zijn. Doel van deze bijeenkomst was het uitwisselen van ervaringen. Opvallend is dat veel docenten aanvankelijk enthousiast zijn over de cursus, maar na de eerste maal gebruiken wat sceptischer worden. Het materiaal blijkt niet voor elke groep studenten even geschikt.
Een van de opleidingen meldde een sterke correlatie met de vooropleiding: studenten met vwo deden het duidelijk beter dan anderen. Daar waar de cursus als bijvak wordt gebruikt voor niet-informatici (en vooral: studenten in een niet-exacte richting) laat de motivatie soms te wensen over. Studenten dreigen te verdwalen in het eerste blok, waar veel onderwerpen (Java, programmeertechnieken, OO-concepten, programmeeromgeving) min of meer kaleidoscopisch aan de orde komen. Het is in deze fase belangrijk om hoofd- en bijzaken te kunnen onderscheiden. Het tweede blok bevat vervolgens een aantal fundamentele zaken (het zelf maken van een klasse, en vooral de representatie in het geheugen van objecten) die voor sommige studenten te diep gaan.

De cursus Visueel programmeren met Java maakt ook deel uit van het z.g. CODI-programma dat een groep geselecteerde docenten uit het middelbaar onderwijs de mogelijkheid biedt om een

eerstegraadsbevoegdheid voor informatica te behalen. Vanuit deze groep studenten kwam aanzienlijke kritiek op de cursus. Het merendeel ervaart de studielast als veel meer dan aangegeven. Een aantal cursisten had problemen met de vereiste wiskundige voorkennis. Dit is opmerkelijk, omdat we tijdens het schrijven van de cursus bewust hebben getracht de afhankelijkheid van wiskunde te beperken.
Cursisten van de CODI worden geacht binnenkort op hun eigen school onderwijs in informatica te geven. We mogen dan ook aannemen dat ze zeer gemotiveerd zijn om het vak grondig te beheersen.

De cursus is kennelijk niet voor elke doelgroep even goed bruikbaar. Bij de TU Delft is vwo met wiskunde B vereist, bij de Open Universiteit (voor deze cursus) tenminste havo met wiskunde. Hbo-studenten hebben (als regel) havo, echter niet noodzakelijk met wiskunde. Cursisten van de CODI hebben tenminste een lerarenopleiding achter de rug, maar niet noodzakelijk in een exact vak. Programmeren is in onze cursussen niet een 'vrij formele tak van wiskunde' (Edsger Dijkstra), maar het vereist wel een zekere vaardigheid in het omgaan met abstractie.

## 6 Conclusies

Terugkijkend op het project hebben we het gevoel dat we onze doelstellingen voor een belangrijk deel hebben waargemaakt:
– Het is gezien de grote belangstelling voor de instapcursus Visueel programmeren met Java gelukt om deze cursus aantrekkelijk te maken voor een veel grotere doelgroep dan informatica-studenten alleen. Afgezien van de negatieve geluiden vanuit de CODI-groep en van een groep studenten met een zwakkere achtergrond en motivatie, wordt er positief over de cursus geoordeeld.
– Ook het direct beginnen met objectgeoriënteerd programmeren en het daarbij gebruik maken van een visuele programmeeromgeving blijkt goed mogelijk te zijn. De zorg van ons (en anderen) of dit wel zou aanslaan bij studenten, is ongegrond gebleken.
– Wat evident gelukt is, is dat studenten programmeren in Java en met een omgeving als Visual Café heel leuk vinden. Zeker de evaluatieresultaten bij de vervolgcursus Objectgeoriënteerd programmeren met Java zijn wat dat punt betreft uiterst positief.
Toch hebben we ook onze zorgen over het nieuwe programmeercurriculum. De belangrijkste is dat we te weinig aandacht hebben geschonken aan het oefenen met elementaire algoritmen. In principe is alle stof hiervoor behandeld en ook wel begrepen, maar de vaardigheid van studenten op dit punt ligt duidelijk lager dan vroeger in het oude curiculum, dat veel algoritmischer van aard was. Extra oefeningen in een practicum liggen hier voor de hand. In wat mindere mate geldt onze zorg ook wel met betrekking tot de vaardigheid die studenten hebben met objectgeoriënteerd modelleren. Ook hier zou aanvullende oefening geen kwaad kunnen.
Een eindconclusie zou kunnen zijn dat het (te) ambitieus gebleken is om binnen drie vakken van 3 studiepunten volleerde OO-programmeurs op te leiden. Wil je een grote groep studenten het gevoel geven dat ze het echt kunnen, dan zul je meer ruimte moeten creëren in je curriculum.

## 7 Literatuur

Arnold, K. en Gosling, J., (1996) *The Java Programming Language*. Addison Wesley.
Bailey, D.A., (1998) *Java Structures, Data Structures in Java for the Principled Programmer*, McGraw-Hill, www.mhhe.com/engcs/compsci/bailey/.
Bishop, J., (1998) *Java Gently, Programming Principles Explained*. Addison Wesley, http://cseng.aw.com/bookdetail.qry?ISBN=0-201-34297-9.
Gamma, E., Helm, R., Johnson, R. en Vlissides J., (1995) *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison Wesley, http://cseng.awl.com/bookdetail.qry?ISBN=0-201-63361-2.
Goodrich, M.T., en Tamassia, R., (1998) *Data Structures and Algorithms in Java*. John Wiley & Sons, www.cs.brown.edu/courses/cs016/book/.
Kluit, P.G., Sint, H.J. en Wester, F.J., (1998) Visual programming with Java; evaluation of an introductory programming course. In *Proceedings Integrating Technology into Computer Science Education (ITiCSE) 98*, Dublin, Ireland, blz. 143-147, ACM, www.acm.org/pubs/citations/proceedings/cse/282991/p143-kluit/.
Sint, H.J., Kluit, P.G., Nolet, C.A., van den Eijnde, J.P.H.W., (1999) *Objectgeoriënteerd programmeren met Java*. Open Universiteit, Heerlen.

Vanhelsuwé, L., (1998) Data structures and algorithms: A comparative slice and dice -- er, review -- of 5 Java DS&A books, *JavaWorld*, www.javaworld.com/javaworld/jw-06-1998/jw-06-bookreview.html.
Vanhelsuwé, L., (1999) In search of the best Java book for beginners; A comparative review of 7 intro-programming language books , *JavaWorld*, www.javaworld.com/javaworld/jw-02-1999/jw-02-bookreview.html.
Weiss, M.A., (1998) *Data Structures & Problem Solving Using Java*. Addison Wesley.
Wester, F.J., (1996) *Java in het inleidend programmeeronderwijs*, TINFON, 7(2), blz. 79-81.
Wester, F.J., Sint, H.J. en Kluit, P.G., (1997) Visual programming with Java; an alternative approach to introductory programming. In *Proceedings Integrating Technology into Computer Science Education (ITiCSE) 97*, Uppsala, Sweden, blz. 57-58, ACM, www.acm.org/pubs/citations/proceedings/cse/268819/p57-wester/.
Wester, F.J., Sint, H.J., Kluit, P.G., Nolet, C.A., et al., (1998) *Visueel programmeren met Java*. Open Universiteit, Heerlen.
Wester, F.J., te Boekhorst, M.J., Witsiers-Voglet, M., (1999) *Werkboek Datastructuren en algoritmen*. Open Universiteit, Heerlen.

**Java-websites Open Universiteit**
Veel informatie over de cursussen Visueel programmeren met Java, Objectgeoriënteerd programmeren met Java (en binnenkort ook Datastructuren en algoritmen) is te vinden op de website www.ou.nl/java. Behalve informatie over inhoud en opbouw van de cursussen zijn ook rondleidingen met Java-applets uit de verschillende cursussen te bekijken.
Voor docenten die met het cursusmateriaal werken wordt momenteel een website met discussiegroep opgezet voor het uitwisselen van informatie en onderwijsmateriaal.

**Auteurs**
Dr. P.G. Kluit is universitair docent bij de faculteit ITS van de Technische Universiteit Delft. Hij is betrokken bij het onderwijs in programmeren en in software engineering. E-mail: P.G.Kluit@twi.tudelft.nl
Drs. H.J. Sint is docent informatica bij de Open Universiteit Nederland. Zij had de cursusteam-leiding van Visueel programmeren met Java (met Frank Wester) en van Objectgeoriënteerd programmeren met Java. E-mail: Marleen.Sint@ouh.nl
Ir. F.J. Wester is senior-docent software-systemen bij de Open Universiteit Nederland. Hij was leider van het project Inleidend programmeeronderwijs en cursusteamleider van de OU-cursus Datastructuren en algoritmen. E-mail: Frank.Wester@ouh.nl