Een LOI-opleiding is **méér dan leerstof alleen**

Bedankt voor het aanvragen van deze proefles. Hierin laten we een stukje van de leerstof zien, om een indruk te geven van de opzet en het niveau.

Maar een opleiding van de LOI is veel meer dan leerstof alleen. Je persoonlijke docent, examentrainingen, de online leeromgeving met onder meer de innovatieve kennistrainer en online hoorcollege's; het zijn allemaal belangrijke voordelen van de LOI. Voordelen waarmee je sneller en gemakkelijker je uiteindelijke doel bereikt: een erkend diploma.

Voordat je bij de leerstof aankomt, vertellen we nog iets meer over deze belangrijke onderdelen. En heb je nog vragen, aarzel dan niet om de afdeling studievoorlichting te bellen. Zij kunnen je ook van dienst zijn bij het bepalen van het juiste niveau, vakgebied of opleiding. Bel 071-545 1234.

Flexibel studeren, thuis of klassikaal

Thuis studeren; zelfstandig maar zeker niet alleen

Wil je zelf bepalen waar, wanneer en hoe snel je studeert? En alle ruimte omde opleiding te combineren met een baan en druk privéleven? Het kan met een thuisstudie van de LOI. De studie-methode is ontwikkeld om flexibel én doelgericht te studeren. Maar zeker niet alleen, want je krijgt alle docentenbegeleiding en kunt online contactleggen met medestudenten.

Klassikaal studeren op een manier die past bij deze tijd

De klassikale opleidingen van de LOI bieden structuur én voldoende flexibiliteitom ze te combineren met een baan en druk privéleven. Je neemt elke 2 tot 3 weken deel aan een klassikale bijeenkomst waarbij je kiest uit dag, avond- en zaterdagprogramma's. Je werkt in kleine groepen onder leiding van topdocenten uit de praktijk. En studeert daarnaast flexibel via de online leeromgeving.

Professionele begeleiding tijdens je opleiding

Bij de LOI krijg je tijdens je gehele opleiding begeleiding van professionals met ruime ervaring in het vakgebied. Topdocenten, die via de online leeromgeving bereikbaar zijn voor extra uitleg en vragen en bij klassikale opleidingen de bijeenkomsten verzorgen. Je kunt dus altijd rekenen op de kennis en ervaring van een vakspecialist. Bij de volledige MBO- en HBO-opleidingen heb je bovendien een persoonlijke coach en/of loopbaanbegeleider, die je begeleidt van start tot diploma.

Online leeromgeving met effectieve studietools

De online leeromgeving, LOI Campus, geeft op elk moment inzicht in je opleiding en vorderingen. Met onder andere een agenda, examengegevens, cijferoverzicht en contactmogelijkheden met docenten en medestudenten. Daarnaast is de leeromgeving hét platform om effectief te leren met online studietools.

Unieke opbouw van de lesstof

Alleen bij de LOI ontvang je speciaal ontwikkeld lesmateriaal, waardoor het bijzonder geschikt is om zelfstandig te bestuderen. Het lesmateriaal is toegankelijk geschreven, overzichtelijk en zo opgebouwd dat je je kennis steeds verder verdiept. Bovendien bevat het verschillende elementen die studeren makkelijker maken.

Opbouw van het lesmateriaal

Inleiding
De inleiding vertelt je wat je in het komende hoofdstuk kunt verwachten.

Trefwoorden
In de kantlijn staan trefwoorden. Hiermee kun je een bepaald onderwerp makkelijk en snel terugvinden.

Oefenopgaven
In de leerstof kom je regelmatig oefenopgaven tegen. Met deze opgaven test je of je de opgedane kennis kunt toepassen. Aan het eind van het hoofdstuk zijn de antwoorden opgenomen. Kun je de oefenopgaven niet goed uitwerken, dan betekent dit dat je het bijbehorende deel van de leerstof nog eens moet doornemen.

Parate-kennisvragen
Aan het einde van het hoofdstuk vind je de parate-kennisvragen. Hiermee kun je testen of je de leerstof voldoende kent. De vragen zijn genummerd. Het antwoord kun je snel terugvinden dankzij de genummerde verwijsrondjes in de kantlijn.

Inzendopgaven
Als je de antwoorden op de parate-kennisvragen weet en de oefenopgaven goed kunt uitwerken, ga je verder met de inzendopgaven. Deze stuur je via de online leeromgeving ter beoordeling naar je docent. Binnen enkele dagen heb je een uitgebreid antwoord terug.

Neem nú een kijkje in de opleiding van je keuze, scroll snel door!

Hoofdstuk 1

Introductie tot programmeren

Computers zijn dom. Althans, ze kunnen alleen opdrachten uitvoeren, die mensen ze opdragen om uit te voeren. Een gebruiker geeft bijvoorbeeld door middel van het klikken op een muis aan wat hij verlangt van de computer. Daarvoor heeft een programmeur een vertaling gemaakt zodat de computer weet welke opdrachten uitgevoerd moeten worden oftewel wat de gebruiker bedoelt met die muisklik. Daarnaast voeren computers vele bewerkingen uit waar geen menselijke input aan te pas komt, maar ook deze moeten allemaal worden geprogrammeerd. In de begintijd van de computer, globaal vanaf de jaren vijftig van de afgelopen eeuw, moest dit letterlijk met enen en nullen aan de computer worden uitgelegd. Gelukkig hoeven tegenwoordig programmeurs niet meer in nullen en enen te programmeren. Ze kunnen gebruikmaken van een groot scala aan programmeertalen die hun hierbij helpen. Nu verschillen deze programmeertalen op het eerste gezicht vaak veel van elkaar, maar toch is er een gezamenlijke basis. Als een programmeur deze basis kent, kan hij zich relatief snel de basis van een willekeurige programmeertaal eigen maken.
Na dit hoofdstuk hebt u basiskennis van de opbouw van een programma. U hebt geleerd wat een IDE is en u hebt een globaal idee van de wereld van de programmeertalen. Ten slotte zult u in de taal Java een eerste programma hebben geschreven.

Een stukje programmeergeschiedenis

Voor de elektronische computer waren er mechanische computers. Ondanks vele discussies en het feit dat winnaars meestal de geschiedenis schrijven, weten we inmiddels dat de eerste werkende computer de Z3 was. De Z3 is in 1941 in Berlijn gebouwd door prof. dr. Conrad Zuse. Tevens ontwikkelde de heer Zuse de eerste programmeertaal ”Plankalküll” geheten.

① In de loop der jaren ontwikkelde de programmeertalen zich van eerstegeneratie- tot vierdegeneratietalen. Het onderscheid tussen de generaties zit met name in het gebruik van

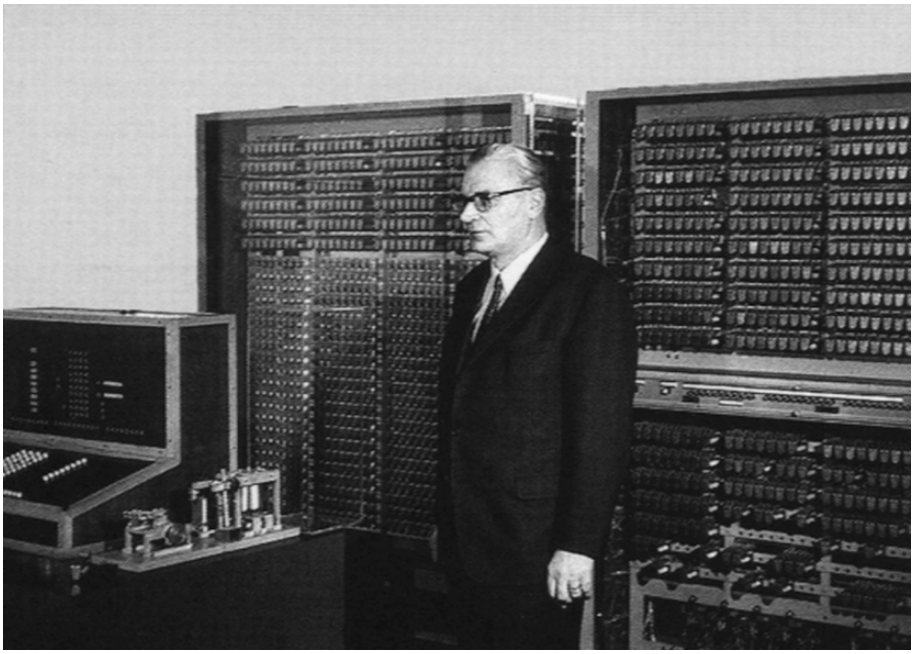
de menselijke taal bij het geven van instructies aan een computer.

Machinetaal

Bij eerstegeneratietalen, ook wel machinetalen genoemd, worden geen menselijke woorden gebruikt. De programmeur programmeert binair (alleen enen en nullen). Dit is namelijk het enige wat een computer feitelijk begrijpt: uit of aan, nul of een, wel of geen elektrisch stroompje. Het zal duidelijk zijn dat code schrijven zoals 01010011 10110111 niet alleen erg foutgevoelig, maar ook erg moeilijk is.

Assembleertaal

Een volgende generatie programmeertalen maakt al gebruik van een vertaalprogramma waarmee heel simpele commando's gegeven kunnen worden door het op een bepaalde wijze vullen van geheugenplaatsen. Dit worden tweedegeneratie- of assembleertalen genoemd.



Afb. 1. Conrad Zuse bij zijn in de 70'er jaren herbouwde Z3.

Derdegeneratietaal

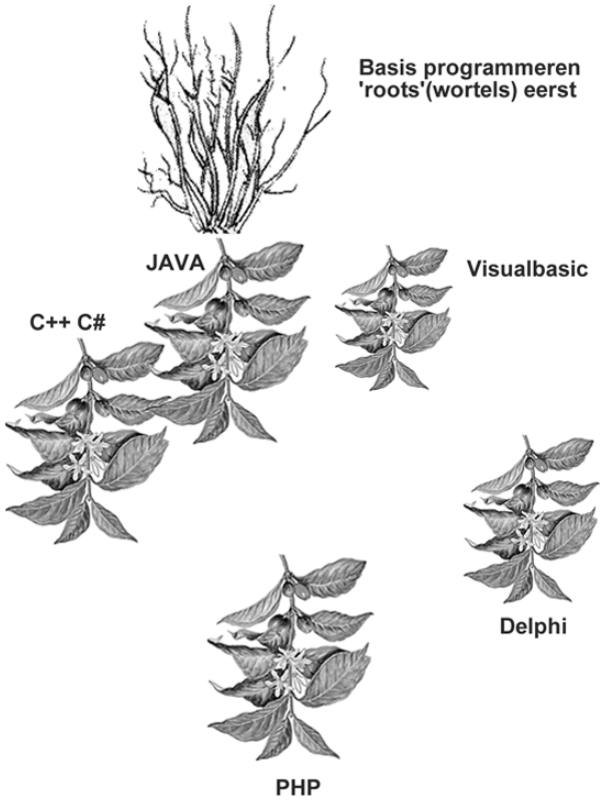
De volgende generatie talen richt zich meer op de programmeur dan op de machine. De vertaalprogramma's worden complexer, zodat een programmeur meer menselijke taal kan gebruiken met het gebruik van termen als begin, end, writeln, for en next.
In deze tijd ontstaan de eerste programmeertalen die ook geschikt zijn voor "thuisgebruik". Voorbeelden hiervan zijn BASIC, Pascal, COBOL, FORTRAN, dBASE en Clipper.

Vierde-generatietaal

Talen die gerekend worden tot de vierdegeneratietalen zijn niet zo zeer programmeertalen als wel hulpmiddelen voor het ontwikkelen van toepassingen. Bijna alle programmeertalen in de derde en vierde generatie gebruiken dezelfde basis. Ondanks het feit dat we in deze module Java gebruiken voor het aanleren van de basis van het programmeren, is deze basis in veel andere talen nagenoeg gelijk. De programmeer-

structuren die u binnen deze cursus leert, kunt u zo ook in andere talen toepassen. Zodra u een programmeertaal goed beheerst, is het relatief eenvoudig om over te stappen op een andere taal. De wortels van iedere moderne programmeertaal zijn gelijk. In deze basiscursus behandelen we deze ”roots”. De plant die u er daarna op ent, kan iedere andere taal zijn. Zo heeft elke taal een manier om aan te geven dat een bepaalde actie een bepaald aantal keer herhaald moet worden, maar de termen die gebruikt worden kunnen per taal verschillen.

② Het grote verschil tussen de diverse programmeertalen zit meer in de bibliotheken dan in de techniek. In een programmabibliotheek of API zitten de uitbreidingen van een specifieke taal. Dit maakt een taal specifiek geschikt voor een bepaald soort toepassingen. Een programmeur die zeer ervaren is in een bepaalde taal, heeft dus een enorme bibliotheekkennis van deze taal. Deze kennis koestert hij en dit is dan ook de reden dat men niet graag van taal wisselt. Bij het wisselen van een taal, moet de programmeur weer een andere bibliotheek leren. In deze module zal, zoals al opgemerkt is, de nadruk liggen op de gezamenlijke basis van de programmeertalen, waardoor u in staat zult zijn deze snel te doorzien.



Afb. 2. Roots first, na een gemeenschappelijke basis kan de volgende stap in iedere taal gezet worden.

Java en C#: een nieuwe generatie

Java en C# zijn twee veel gebruikte programmeertalen die een nieuwe soort programmeertalen vertegenwoordigen. Om dit technisch enigszins beter uit te kunnen leggen, moet een andere onderverdeling van programmeertalen worden toegelicht.

③ Men kan programmeertalen namelijk ook indelen in gecompileerde en geïnterpreteerde talen.

Gecompileerde taal

Bij een gecompileerde taal, wordt de door de programmeur geschreven broncode door een zogeheten compiler omgezet in machinecode, die rechtstreeks door de computer verwerkt kan worden. Een compiler is dus een computerprogramma dat de broncode kan omzetten in code die de computer begrijpt. Deze omgezette versie is dan het uiteindelijke programma dat telkens wordt uitgevoerd bij het opstarten van het programma. Voor allerlei talen en besturingssystemen zijn er verschillende compilers. Als u een computerprogramma schrijft in de taal C, en u wilt het gebruiken voor zowel Windows als Linux, hebt u twee verschillende compilers nodig. Voor iedere compiler dient u als programmaontwikkelaar afzonderlijk te betalen. Om een computerprogramma geschreven in zo'n soort taal geschikt te maken voor algemeen gebruik kost dus veel moeite en geld. Het voordeel van gecompileerde programma's is dat ze over het algemeen sneller zijn dan geïnterpreteerde talen. Voorbeelden van gecompileerde talen zijn: C, C++, Pascal, Delphi, VisualBasic, Java.

Geïnterpreteerde taal

Bij de geïnterpreteerde taal is de broncode gelijk aan de code die wordt uitgevoerd. Er wordt dus geen vertaalprogramma gebruikt zoals een compiler. Een computerprogramma voor een geïnterpreteerde taal kan daarom in een eenvoudige teksteditor, zoals notepad, geschreven worden. Voor de geïnterpreteerde taal is een server nodig, die het programma uitvoert. Het programma is een tekstbestand, dat regel voor regel door de server wordt gelezen en uitgevoerd door de tekst te vertalen naar voor de computer begrijpelijke instructies. Het tekstbestand is dus het uiteindelijke programma dat elke keer opnieuw gelezen wordt bij het opstarten van het programma. Het voordeel van de geïnterpreteerde taal is dat het op ieder platform zonder aanpassingen kan draaien. Geïnterpreteerde programma's zijn over het algemeen trager dan gecompileerde programma's. Voorbeelden van geïnterpreteerde talen zijn: PHP, HTML, XML, Java.

Als oplettende lezer hebt u opgemerkt dat Java bij beide categorieën wordt genoemd. Java is dan in zekere zin ook een taal apart. Java-broncode wordt gecompileerd tot machinecode,

die vervolgens wordt geïnterpreteerd door een Virtual Machine.

JVM Java Virtual Machine (4)

De Java Virtual Machine (JVM) is als het ware de tweede helft van het programma. Het programma vormt samen met de JVM een functioneel geheel. De JVM bevat als het ware de code die nodig is om het programma te laten draaien op het actuele besturingssysteem. Bij het compileren wordt dus de vertaalslag gemaakt van de code zoals door de programmeur is geschreven naar een code die door de JVM begrepen wordt. De JVM maakt vervolgens de vertaling van die code naar een code die een computer(platform) begrijpt.
Voor ieder computerplatform (Windows, Linux, MacOS, Solaris, BSD etc) is een JVM ontwikkeld. Zodra op een besturingssysteem een JVM is geïnstalleerd, kan het ieder Java-programma draaien. Dus nadat u uw eerste grote programma in Java hebt geschreven en gecompileerd, werkt het programma op ieder besturingssysteem!
In de meeste besturingssystemen is de JVM standaard geïntegreerd. Door een conflict tussen SUN, de ontwikkelaar van Java, en Microsoft, bevat het Windows-besturingssysteem niet standaard de JVM. Gebruikers dienen de JVM apart te downloaden en te installeren. Vaak gebeurt dit min of meer automatisch, door het bezoeken van een website die Java-technologie gebruikt. De gebruiker krijgt de vraag of Java geïnstalleerd moet worden, klikt op ja en de installatie wordt voltooid.
De ontwikkelaar heeft als voordeel dat hij het programma maar één keer hoeft te compileren. De gebruiker heeft als voordeel dat een Java-programma de snelheid heeft van een gecompileerde taal en de flexibiliteit heeft van een geïnterpreteerde taal. Deze techniek heeft zichzelf zo bewezen dat Microsoft in de ontwikkeling van C# grote stukken van Java heeft overgenomen. Java was ten opzichte van zijn voorgangers een taal apart en een volgende ontwikkeling in het programmeren. Talen zoals Java en C# noemen we bytecodetalen.

Bytecodetaal (5)

Een bytecodetaal is dan een taal die eerst wordt gecompileerd tot bytecode en vervolgens wordt geïnterpreteerd. Ook door de keuzes die Microsoft heeft gemaakt met c# en dot NET lijkt de bytecodetechniek het momenteel te gaan winnen. De dot NET-techniek is de variant van Microsoft op de SUN Java Virtual Machine.

Oefenopgave 1

In de volgende tabel staan een aantal programmeertalen in de eerste kolom. In de tweede, derde en vierde kolom, staat de manier waarop deze talen uitgevoerd worden. Zet bij iedere

taal in de juiste kolom een kruis. Maak gebruik van internet voor het vinden van de antwoorden.

Programmeer-taal	Interpretatie	Compilatie	Bytecode
Perl Python ASP Basic Assembly JSP Fortran Ruby JavaScript			

Installeren en configureren van Eclipse en Java

Leren is vaak ook proberen en dit geldt zeker ook voor programmeren.
Daarom zullen we in deze module ook daadwerkelijk wat kleine programma's gaan programmeren. Hier is uiteraard een programmeeromgeving voor nodig.

Eclipse

In deze module maken we gebruik van Eclipse, de OpenSource IDE van IBM.

IDE (6) Een IDE is een grafische omgeving, om computerprogramma's te schrijven en te compileren. Met Eclipse beschikt u over een forse gereedschapskist, waarmee u Java programma's kunt ontwikkelen. Het downloaden en installeren van Java en Eclipse gaat in drie stappen:

- De software van Java en Eclipse downloaden.
- Java installeren.
- Eclipse installeren.

Deze stappen worden achtereenvolgens in deze paragraaf doorgenomen. Het is van belang dat u eerst Java installeert en pas daarna Eclipse.

Het downloaden van Java en Eclipse

Op LOI Campus vindt u de volgende bestanden:

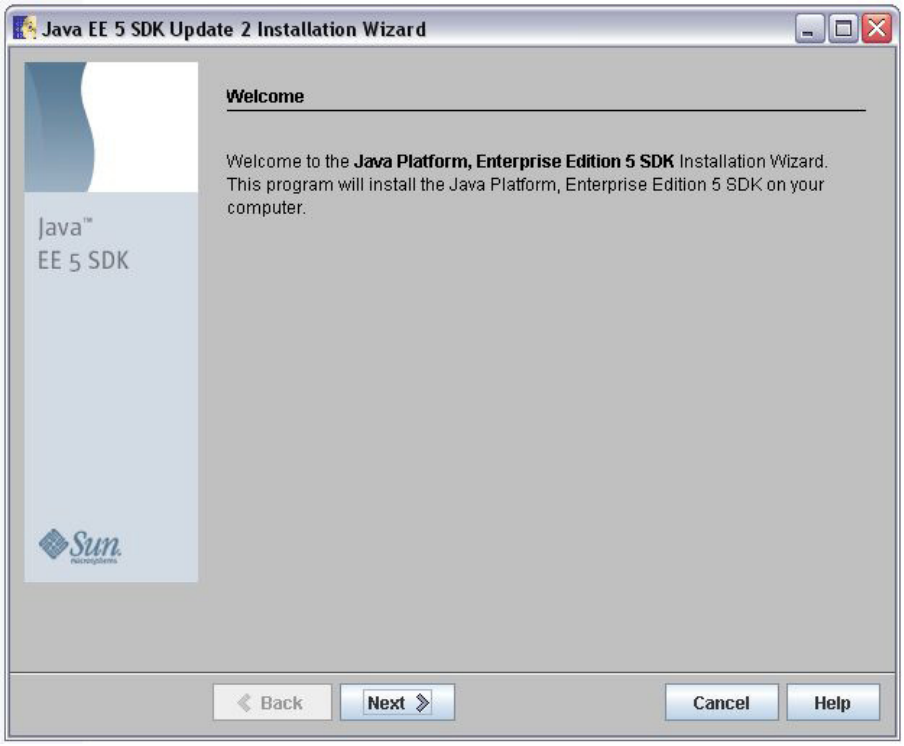
- Java_ee_sdk-5_02-windows.exe
- Eclipse-SDK-3.2.2-win32.zip

Login op Campus, en navigeer naar Start > Downloaden. De bestanden zijn opgenomen onder Java Programmeren.

U hebt beide bestanden nodig.

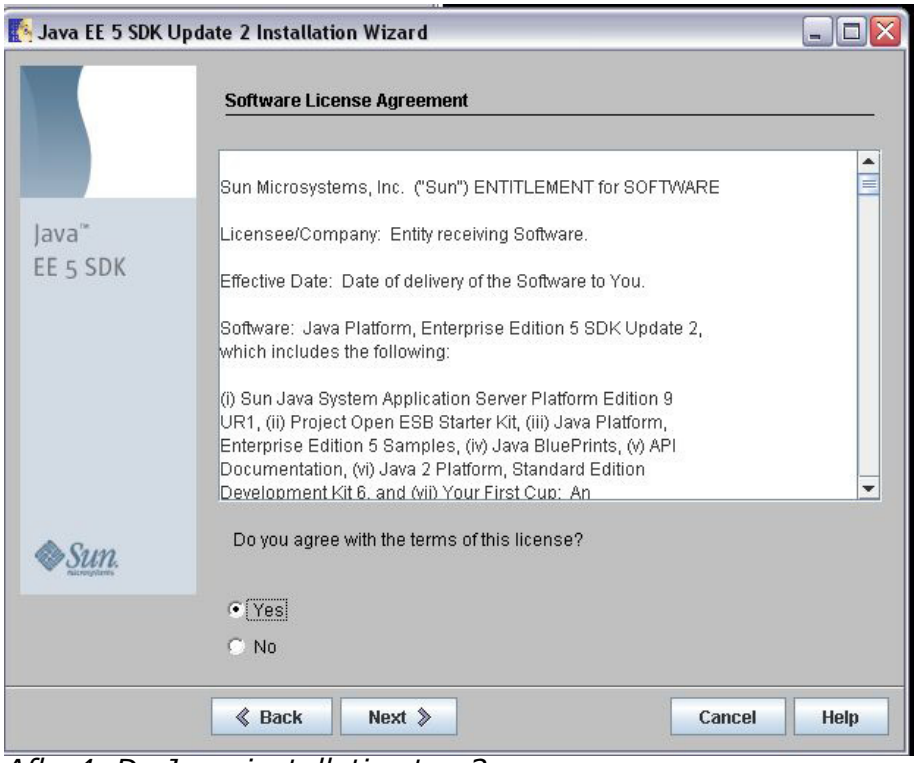
Java installeren

Dubbelklik op het bestand ”java_ee_sdk-5_02-windows.exe”, een scherm opent zich, zoals zichtbaar in afbeelding 3.



Afb.3. Java-installatie-openingsscherm.

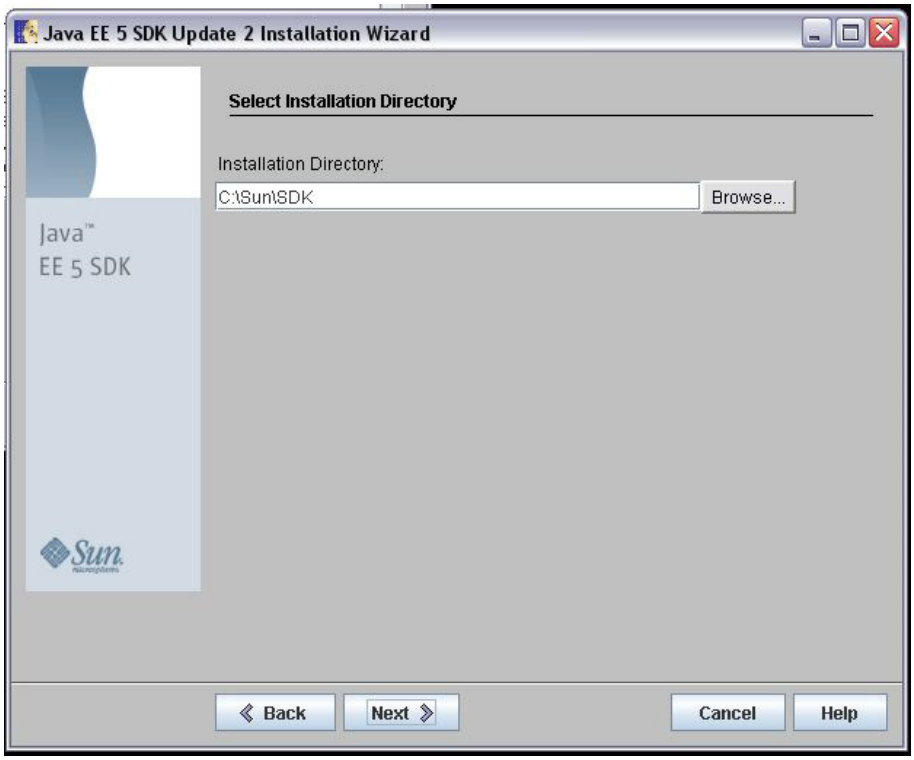
Klik op Next, een scherm zoals zichtbaar in afbeelding 4 opent zich.



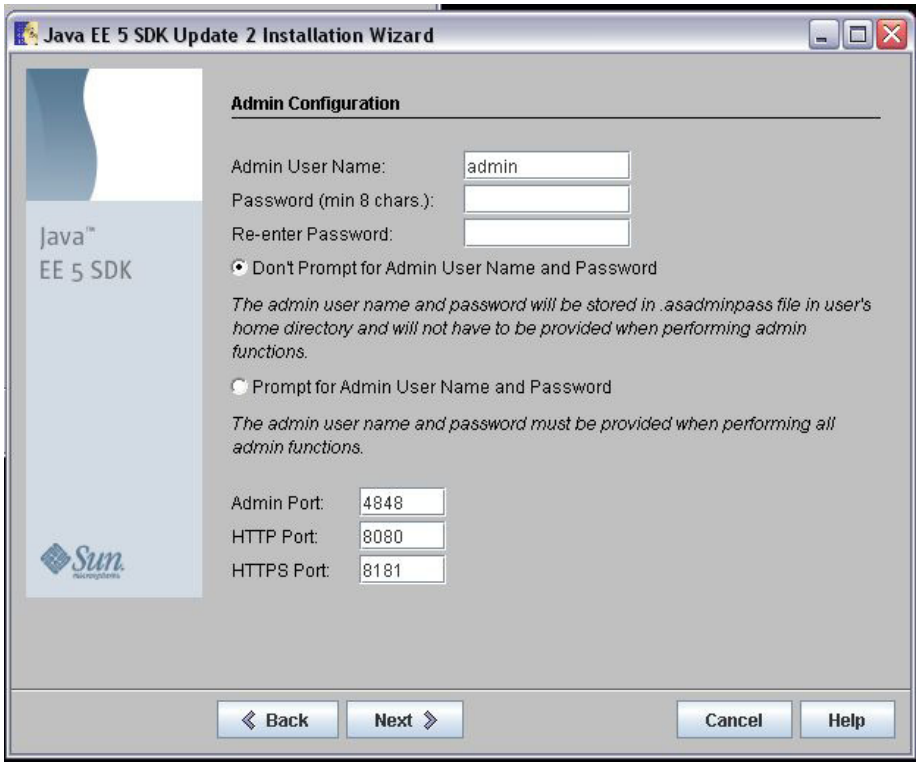
Afb. 4. De Java-installatie stap 2.

Kies ”Yes” en klik op Next, een scherm zoals zichtbaar in afbeelding 5 opent zich.

Kies een installatiedirectory. Als u een directory kiest die niet bestaat, verschijnt er nog een pop-up na het klikken op Next. Deze pop-up is in deze paragraaf verder weggelaten. Klik op Next, een scherm zoals zichtbaar in afbeelding 6 opent zich.

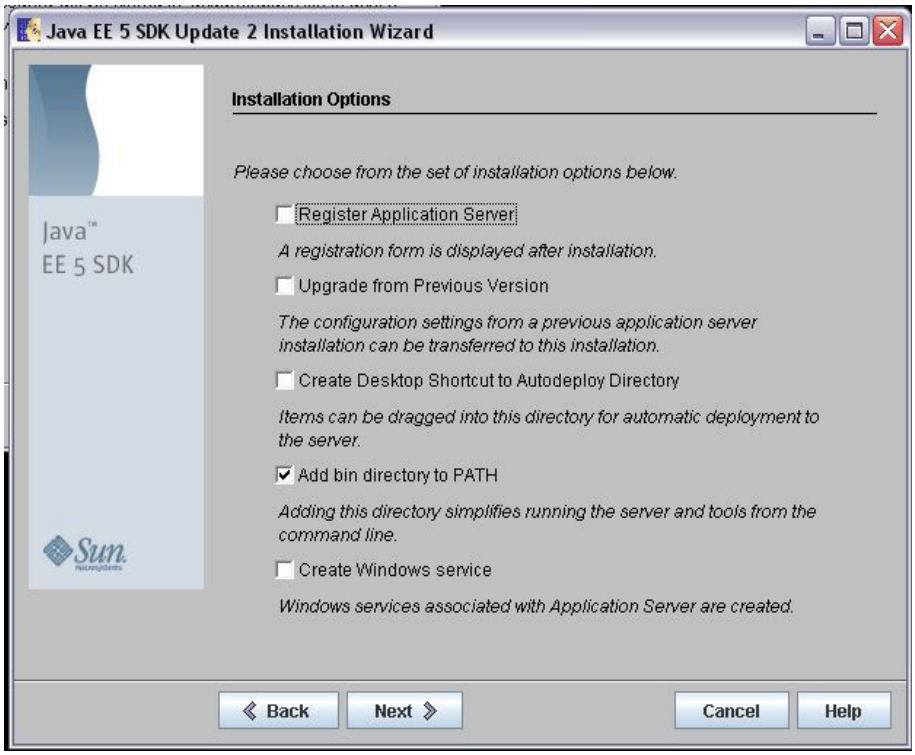


Afb. 5. De Java-installatie stap 3.

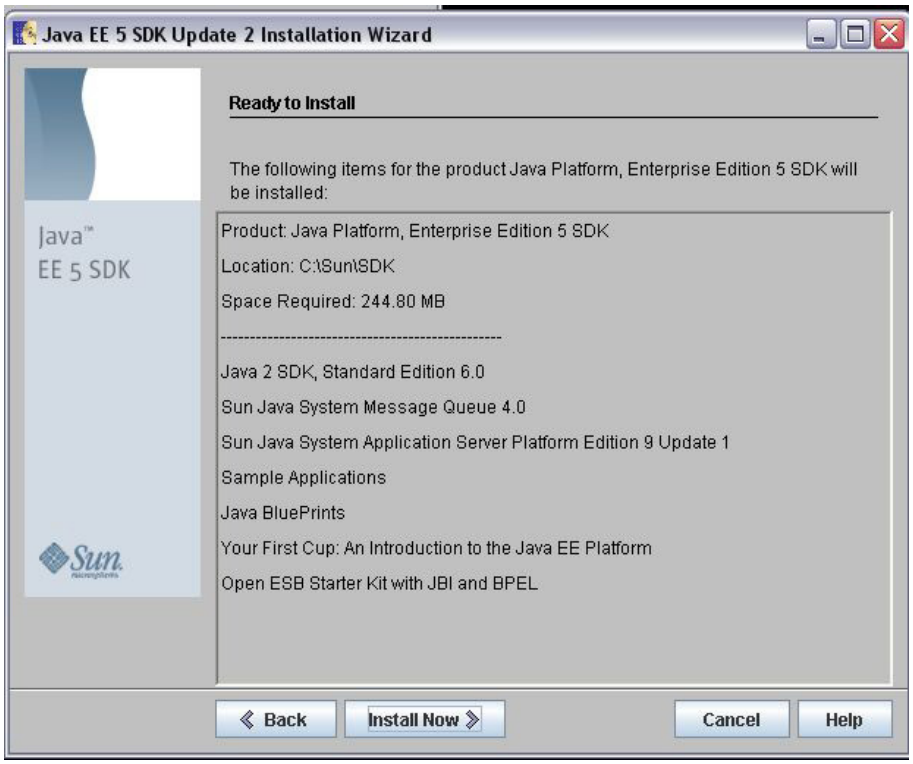


Afb. 6. De Java-installatie stap 4.

Vul tweemaal hetzelfde admin password in en klik op Next, een scherm zoals zichtbaar in afbeelding 7 opent zich.



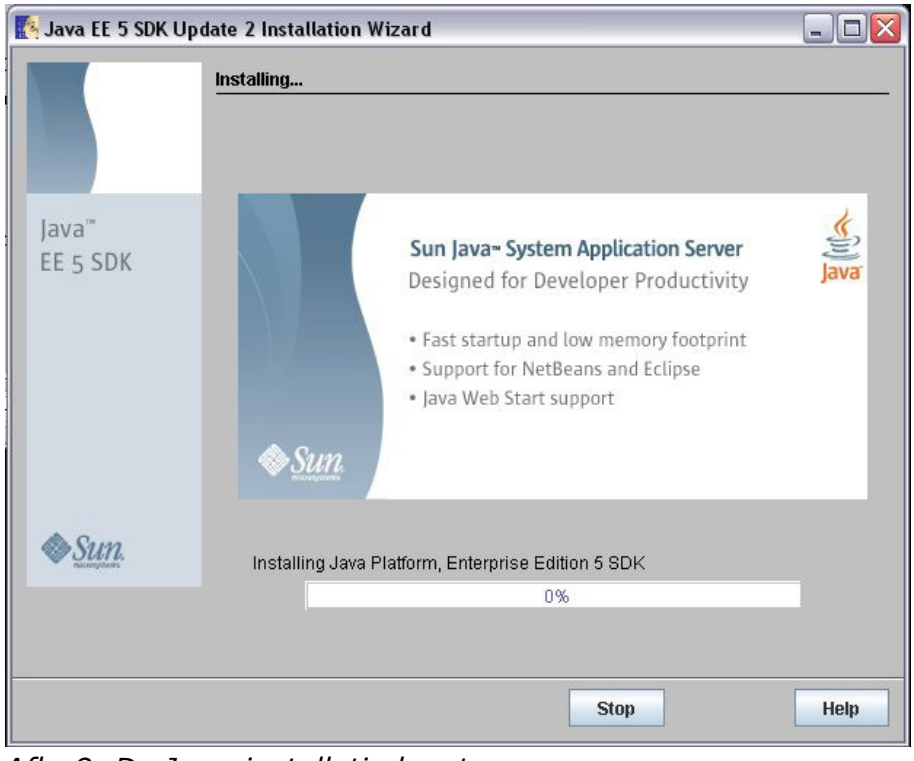
Afb. 7. De Java-installatie stap 5.



Afb. 8. De Java-installatie stap 6.

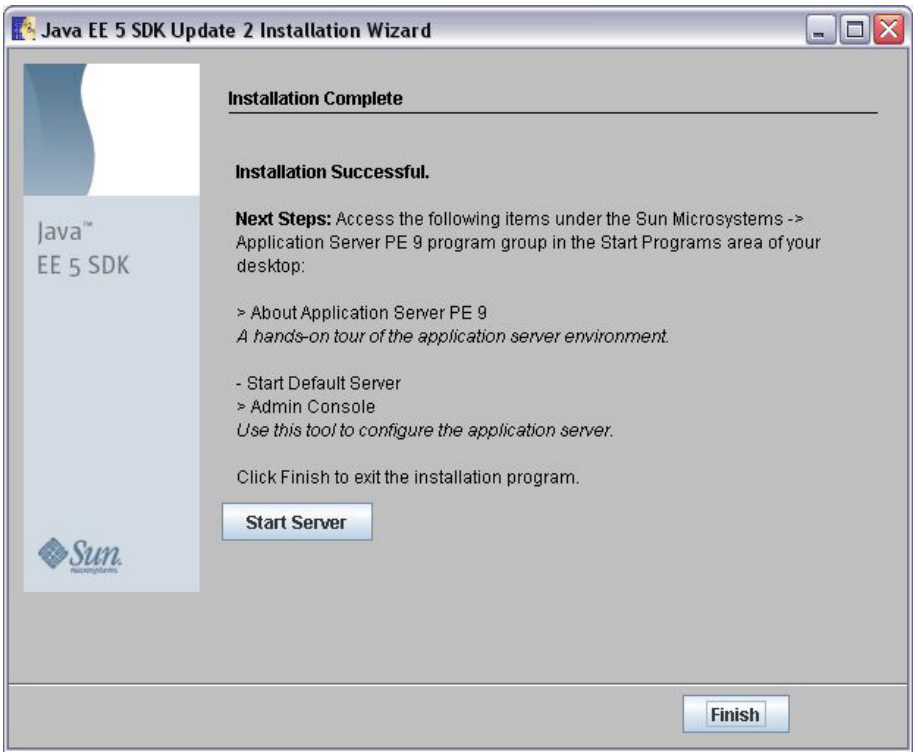
Zorg dat er alleen een vinkje staat bij de optie ”Add bin directory to PATH”. Klik op Next, een scherm zoals zichtbaar in afbeelding 8 opent zich.

Klik op Install Now. Een scherm zoals zichtbaar in afbeelding 9 opent zich.



Afb. 9. De Java-installatie loopt.

Na verloop van tijd opent er een scherm zoals zichtbaar in afbeelding 10.



Afb. 10. De Java-installatie is klaar.

Klik op de knop Finish. U hoeft de server niet te starten.

Eclipse installeren

De installatie van Eclipse dient *na* de installatie van Java te gebeuren. Eclipse is zelf een Java-programma en kan zonder Java niet werken.
Pak de zipfile ”eclipse-SDK-3.2.2-win32.zip” uit op een voor u handige plaats, bijvoorbeeld c:\Eclipse. U kunt natuurlijk ook een andere directory kiezen. In Windows Vista zit echter een beveiliging, waardoor u Eclipse niet in de directory Program Files kunt plaatsen. In andere directories werkt Eclipse prima.
Navigeer naar de uitgepakte directory en dubbelklik op het programma startup.jar, een afbeelding zoals zichtbaar in afbeelding 11 opent zich.

Afb.11. De installatie van Eclipse.

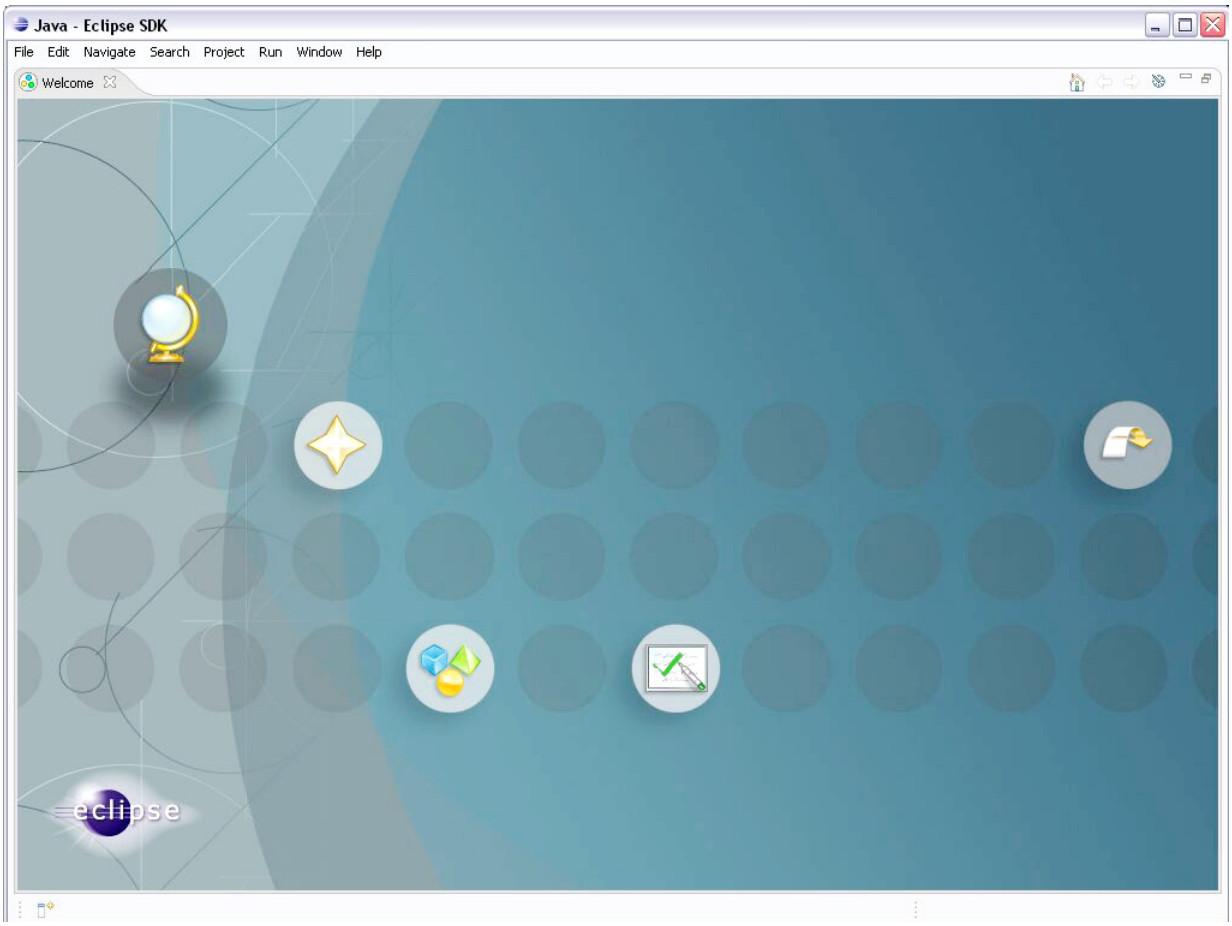
In deze stap stelt u de locatie van uw workspace in. In de workspace bewaart u alle programmeerprojecten die u met Eclipse geschreven hebt. Tijdens LOI-programmeeropleidingen moet u regelmatig gedeeltes van uw workspace zippen en insturen als uitwerking van een opdracht. Klik na het ingeven van de workspace op OK, een scherm zoals zichtbaar in afbeelding 12 opent zich.

Naast Eclipse zijn er voor Java nog diverse andere IDE's. Zo kennen we voor Java ook Visual Cafe, JBuilder en Netbeans. Voor iedere programmeertaal is er een heel scala aan IDE's. Het is vaak een kwestie van persoonlijke voorkeur welke IDE gebruikt wordt.

Oefenopgave 2
In deze oefenopgave gaan we een nieuw project aanmaken, met een nieuwe class in Eclipse. Hierna zijn we klaar voor het schrijven van het eerste programma. Een project is een manier om programmaontwerpen die bij elkaar horen, gezamenlijk op te slaan. Per hoofdstuk maken we in deze cursus een project aan, zodat alle oefenopgaven van dat hoofdstuk bij elkaar staan. In de "echte" programmeerwereld, wordt het project gebruikt om één of meer applicaties die bij elkaar horen, op te slaan.

- Start Eclipse
- Klik op het menu Window – Open Perspective – Java.
- Klik op het menu File – New – Project.
- Kies Java Project.
- Noem het project Hoofdstuk1.
- Klik op Finish.
- Klik op het menu File – New – Class.
- Gebruik als Package name Hoofdstuk 1.

- Geef de class de naam HalloWereld.
- Selecteer ”public static void main (String[] args).
- Deselecteer ”Inherited abstract methods”.
- Klik op Finish.



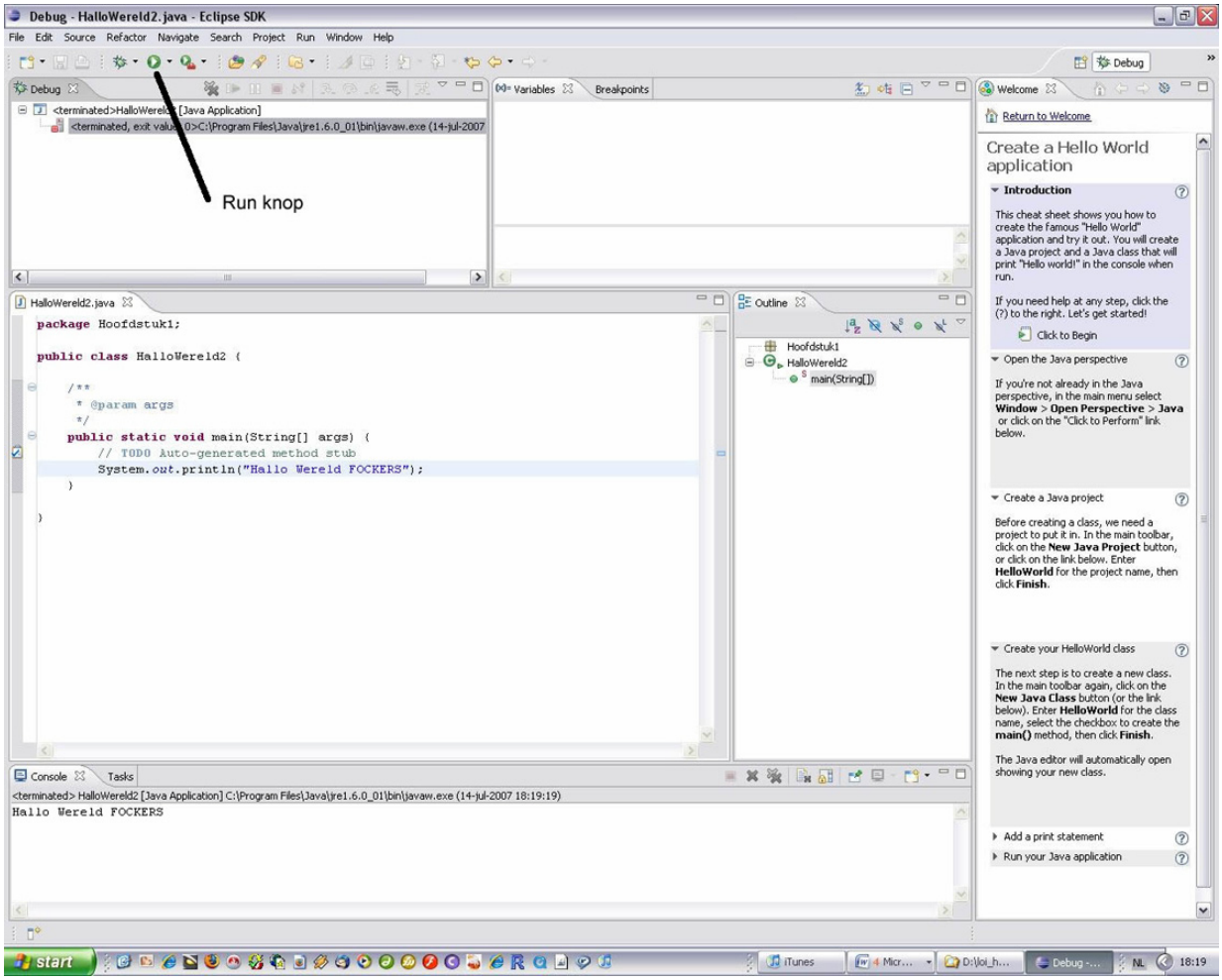
Afb.12. De Eclipse-installatie is gereed.

Het ”Hallo Wereld”-programma

In de meeste programmeerboeken en -cursussen wordt gebruikgemaakt van het ”Hallo Wereld”-programma. Dit is een eenvoudig programma dat een regel tekst afdrukt op het scherm. Het ”Hallo Wereld”-programma heeft als doel de eerste drempel tot het programmeren te nemen. Door dit programma in een vroeg stadium uit te voeren, heb je als beginnende programmeur meteen iets concreets in handen. In Oefenopgave 3 gaan we ons eerste Java-programma schrijven.

Oefenopgave 3
Start Eclipse, indien Eclipse na oefenopgave 2 niet meer gebruikt is, start hij automatisch het juiste project op. Indien dit niet het geval is, dien je via het file-menu het hoofdstuk 1-project alsnog te openen.

Plaats tussen de accolades ({ }) van de main method (public static void main (String[] args)) de volgende regel:
System.out.println(”Hallo Wereld”);
Klik op Window – Open Perspective – Other – Debug.
Klik op de run knop (zichtbaar in afbeelding 13). Eclipse vraagt vervolgens welk programma het moet starten, kies hier voor ”HalloWereld”.



Afb. 13. Resultaat van het runnen van het ”Hallo Wereld”-programma.

Anatomie van een programma

In deze paragraaf gaan we kijken naar de basis anatomie oftewel de structuur van een programma. Naast de onderdelen die hier behandeld zullen worden, zijn er ook elementen in een programma die noodzakelijk zijn om een werkend programma te krijgen, maar die niet van belang zijn om het basisconcept van het programmeren onder de knie te krijgen. Er zal dus geen aandacht worden besteed aan de noodzakelijke zogeheten keywords als public, static en void.

Het programma HalloWereld (HW), dat we in oefenopgave 3 gemaakt hebben, dient als voorbeeld voor het uitleggen van de structuur van een programma. Boven in het HW-programma staat een package-declaratie.

Package

De package wordt voornamelijk gebruikt voor grotere programma's, wanneer een programma uit verschillende classes oftewel hoofdonderdelen bestaat. Een package is een manier om aan te geven dat verschillende stukken code bij elkaar horen. Zoals u kunt zien, heeft het package standaard dezelfde naam als het project. Hoewel ze formeel niet helemaal gelijk aan elkaar zijn, maakt dat voor onze opdrachten niets uit, omdat niet gewerkt wordt met programma's bestaande uit verschillende classes en alleen daar is dit onderscheid relevant.

Commentaar

Onder de package-declaratie staat commentaar. Commentaar komt voor in verschillende vormen. Zo kent Java lijncommentaar, dat begint met //. Dit geeft aan dat de rest van de regel commentaar en geen programmacode is. Daarnaast kent Java paragraafcommentaar, dat ingesloten ligt tussen /* en */. Hierbij is alles dat binnen deze tekens staat commentaar en geen programmacode. Ten slotte kent Java javadoc-commentaar, dat ingesloten ligt tussen /** en */.

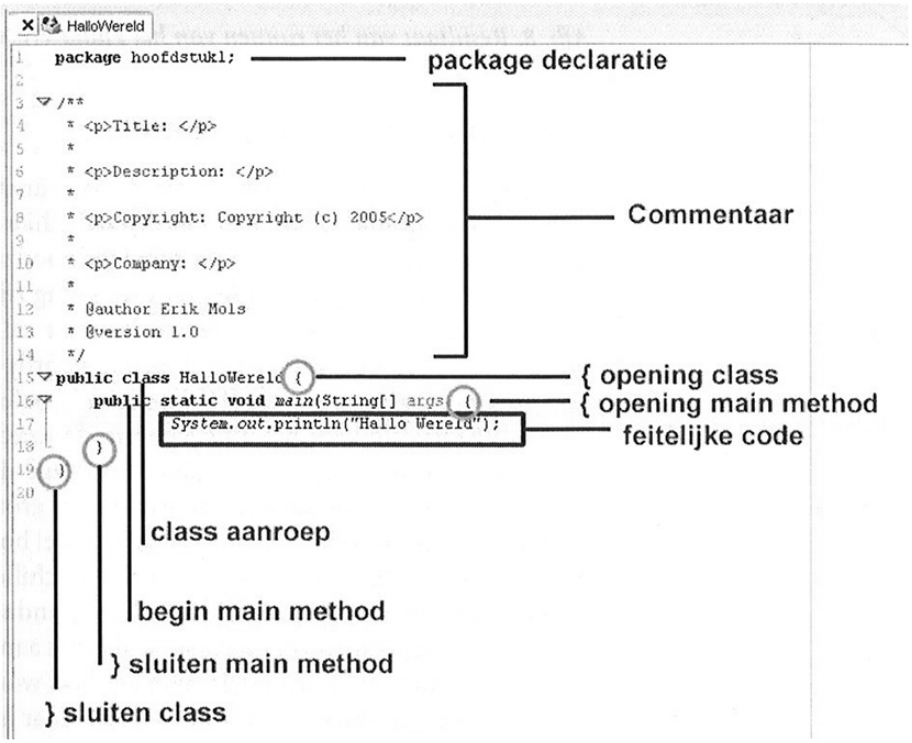
⑦ Commentaar is een wezenlijk onderdeel van ieder programma. Als een programmeur een stuk code zonder commentaar heeft geschreven en hij moet een jaar later een wijziging aanbrengen, kan hij uit zijn eigen code geen wijs meer. Naast het feit dat u uw eigen code niet meer begrijpt, kan het nog gebeuren dat anderen uw code ook niet begrijpen. Een moderne programmeur werkt over het algemeen in een team gezamenlijk aan een applicatie en dus is commentaar essentieel om tot een eindresultaat te komen.

In de anatomie van een programma is het belangrijk te weten dat een gesloten classonderdeel tussen accolades staat{ }. Als een programma, zoals in onze voorbeelden, slechts uit één klasse bestaat, betekent dit dus dat deze accolades het programma omsluiten. In principe bestaat een programma overigens uit minimaal één class en één method.

Class

Een class is een hoofdonderdeel van een programma en dus als het programma maar uit één class bestaat ook gelijk het hele programma. Binnen een class kunnen vervolgens één of meer methods voorkomen. In andere programmeertalen worden methods ook wel function, procedure of routine genoemd. Een method is een onderling samenhangende programmacode die een bepaalde actie teweegbrengt. Zo kan een method bijvoorbeeld alle priemgetallen onder de duizend weergeven of een rapport aanmaken op basis van bepaalde gegevens. Het is dus een eenheid in een programma. Elke class heeft één MAIN method.

MAIN method ⑧ De MAIN method is de plaats waar het programma gestart wordt. Vanuit hier worden eventuele andere methods aangeroepen. Net als alle andere methods is de MAIN method een onderdeel van een class, daarom staat de MAIN method binnen de accolades van de class HalloWereld.



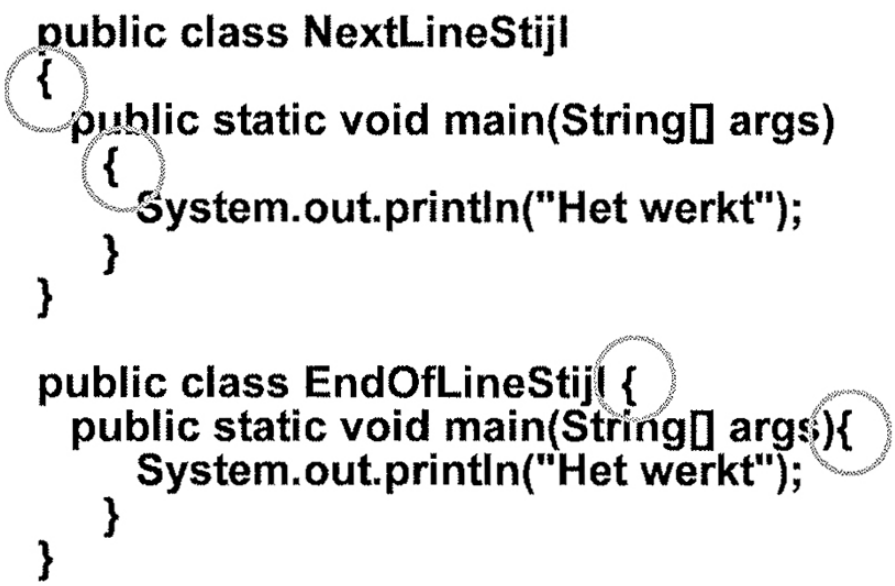
Afb. 14. Anatomie van een Java-programma.

Afbeelding 14 geeft de opbouw van een Java-programma. Boven de class staat commentaar, dat geen functie voor de werking van het programma heeft. De class HalloWereld omsluit vervolgens met zijn accolades zowel de MAIN method als de ”feitelijke code”. De MAIN method omsluit met zijn accolades de ”feitelijke code”, die dus in de MAIN method staat.

Statement De ”feitelijke code” noemen we een statement. Dit is de feitelijke opdracht die de computer dient uit te voeren. Statements eindigen altijd met een ; .

Blok ⑨ Een gedeelte van een programma, dat tussen accolades staat, noemen we een blok. Om het geheel voor programmeurs extra lastig te maken zijn er twee blokstijlen die gebruikt worden. We kennen de ”end of line”-stijl en de ”next line”-stijl waarbij de plaats van de blokhaken afwijkend is. Beginnende programmeurs maken met de ”end of line”-stijl minder fouten. Veel programmeurs vinden echter de ”next line”-stijl netter. In deze module maken we gebruik van de ”end of line”-stijl. De keuze voor een stijl heeft op geen enkele wijze invloed op het functioneren van een programma. De stijlkeuze is vaak een

persoonlijke voorkeur van de programmeur of een onderlinge afspraak binnen een groep programmeurs.



Afb. 15. De next line-stijl heeft een andere positie voor de openende blokhaken.

Een veel gehoorde vraag van mensen die voor het eerst met programmeren in aanraking komen, is: waarom? Waarom moet er een ; aan het eind van een statement en is dit bij een andere programmeertaal juist weer een punt? Het simpele antwoord is: afspraken. Zoals bij elke taal moet onderling worden afgesproken welk woord, begrip en teken wat betekent. Anders kunnen we elkaar niet verstaan. Als de ene persoon met het woord ”vet” dik bedoelt en de andere persoon ”cool” of ”gaaf”, kunnen behoorlijke spraakverwarringen ontstaan. Bij programmeren luistert dit extra nauw, omdat een computer precies uitvoert wat hem opgedragen is. Niets meer en niets minder. Waar een mens misschien nog denkt er staat ”woter”, maar gezien de rest van de tekst zal de ander wel ”water” bedoelen, kan een computer dit niet. De kleinste fout kan dus leiden tot het falen van een programma. De regels accepteren, doorgronden en volgen is daarom eigenlijk de enige manier om programmeren te leren.

Oefenopgave 4

Welke van de volgende drie programma's is anatomisch juist?

a. Code voorbeeld 1

```
package hoofdstuk1

/**
 * ?p?Title: ?/p?
 *
 * ?p?Description: ?/p?
 *
 * ?p?Copyright: Copyright (c) 2005?/p?
 *
 * ?p?Company: ?/p?
 *
 */
public class OefenOpgave4 {
    public static void main(String[] args) {
        System.out.println("Oefenopgave 4");
}
```

b. Code voorbeeld 2

```
package hoofdstuk1

/**
 * <p>Title: </p>
 *
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) 2005</p>
 *
 * <p>Company: </p>
 */
public class OefenOpgave4 {
    public static void main(String[] args)
        System.out.println("Oefenopgave 4");
    }
}
```

c. Code voorbeeld 3

```
package hoofdstuk1

/**
 * <p>Title: </p>
 *
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) 2005</p>
 *
 * <p>Company: </p>
 */
public class OefenOpgave4 {
    public static void main(String[] args) {
        System.out.println("Oefenopgave 4");
    }
}
```

Samenvatting

De eerste programmeertaal stamt uit 1941. Programmeertalen kunnen verdeeld worden in vier generaties. Het onderscheid tussen de generaties zit met name in het gebruik van de menselijke taal bij het geven van instructies. Een andere indeling tussen programmeertalen is die tussen geïnterpreteerde talen en gecompileerde talen en bytecodetalen. Van de laatste is Java een voorbeeld. Bij een gecompileerde taal, wordt de door de programmeur geschreven broncode door een zogeheten compiler omgezet in machinecode. Bij de geïnterpreteerde taal is de broncode gelijk aan de code die wordt uitgevoerd. Een IDE is een grafische omgeving, om computerprogramma's te schrijven en te compileren. Een programma wordt vooral uit blokken opgebouwd. Een blok is te herkennen aan de accolades waar het tussenstaat. Ieder programma heeft ten minste één class en één method.

Parate-kennisvragen

1. Wat is het grote verschil tussen de diverse generatietalen?
2. Wat is het belangrijkste verschil voor de programmeur tussen de diverse vierdegeneratietalen?
3. Welke indeling in programmeertalen kent men naast de indeling in generaties?
4. Wat is de functie van de Java Virtual Machine?
5. Wat zijn bytecodetalen?
6. Wat is een IDE?
7. Waarom is commentaar in een programma zo belangrijk?
8. Welke method is als het ware de startknop van een applicatie?
9. Hoe kunnen we een blokcode herkennen?

Uitwerking van de oefenopgaven

Oefenopgave 1

Programmeer-taal	Interpretatie	Compilatie	Bytecode
Perl	X		
Python			X
ASP	X		
Basic	X		
Assembly		X	
JSP	X		
Fortran		X	
Ruby	X		
JavaScript	X		

Oefenopgave 2
Van deze opgave is geen uitwerking. De opgave is een stappenplan. Het eindresultaat is zichtbaar in afbeelding 7.

Oefenopgave 3
Van deze opgave is geen uitwerking. De opgave is een stappenplan. Het eindresultaat is zichtbaar in afbeelding 8.

Oefenopgave 4
Antwoord c is juist. Zowel code a als b hebben een constructiefout in de accolades.

Overtuigd?

Ben jij ook overtuigd van de voordelen van een LOI-opleiding? Wacht dan niet langer en schrijf je in voor de opleiding van je keuze. Doe het vandaag nog! Dan heb je je lesmateriaal al binnen een paar dagen in huis.

Vragen?

Heb je nog vragen over de opleiding van je keuze of over studeren bij de LOI? Bel dan even. Ons telefoonnummer is (071) 545 1234. Een heel team professionele adviseurs zit voor je klaar. We zijn bereikbaar op werkdagen van 08.00 tot 20.00 uur en op zaterdag van 10.00 tot 14.00 uur.