# Plan van Aanpak zij-instroom programma voor JTech Ordina

#### Aanleiding en achtergrond

Ordina kent twee groepen instromers in JTech (Java technologie). Voor beide groepen is het in eerste instantie van belang om zo snel mogelijk voor klanten inzetbaar te zijn. Belangrijk onderdeel daarvoor is het verkrijgen van het OCA (Oracle Certified Associate) programma voor java programmeurs. Klanten hechten hier veel belang aan.

Daarnaast is voor werkelijke inzetbaarheid bij klanten een aanzienlijke lijst nodig van skills en ervaring met tooling en frameworks rondom Java nodig.

Ordina kent twee typen instromers:

1. Instromers zonder IT achtergrond
2. Instromers met IT achtergrond, bijvoorbeeld HBO-ICT

## Scope

Dit document richt zich in eerste instantie op het bijscholingsprogramma voor Java dat door zij-instromers binnen de eerste 12 weken wordt doorlopen. Bedoeling is om dit nog uit te breiden naar het fast-lane programma dat later in het eerste jaar volgt.

Op dit moment wordt voor beide typen zij-instromers in een opleiding voorzien die gericht is op het verkrijgen van het OCA certificaat. De huidige opzet waarin beide groepen direct al aan het voorbereidingsprogramma voor OCA deelnemen voorziet echter in een aantal zaken niet of te weinig. Problemen die zich op dit moment voordoen zijn de volgende:

1. Het programma voor personen zonder een IT achtergrond is in deze vorm niet goed geschikt omdat ze basisinzichten en ervaring missen op het gebied van programmeren wanneer ze daar direct aan zouden beginnen.
2. Het programma voor het OCA is “te droog” en alleen gericht op de vele taalelementen van Java. Simpel gezegd, je weet waar alle knoppen en de versnellingspook zitten maar daarmee kun je nog niet rijden. Het OCA programma is bijzonder uitgebreid wat Java taaldetails betreft maar geeft geen houvast voor wat betreft het “begrijpen” of “toepassen”. Instromers zonder IT achtergrond lopen daardoor vast en kunnen de kennis die ze tot zich nemen niet of niet goed plaatsen. Het voelt als kauwen op zand.

Naast het OCA certificaat is er ook nog het OCP certificaat (Oracle Certified Professional). Dit is gericht op professionals binnen Java. De bedoeling is om binnen één jaar instromers via een fast lane programma klaar te stomen voor het behalen van dit certificaat.

## Voorgestelde opzet

De nieuwe opzet voor zij-instromers is dat er een voorprogramma komt van ongeveer vier weken voor de zij-instromers zonder IT achtergrond. Binnen deze vier weken is het de bedoeling dat:

* de zij-instromers die beschikken over een IT achtergrond hun OCA certificaat behalen
* De zij-instromers zonder IT achtergrond (net) voldoende basis-skills en inzichten hebben om aan projecten deel te nemen tezamen met instromers die wel een IT achtergrond hebben.

Vervolgens komen beide trajecten tezamen en doorlopen beide typen instromers hetzelfde programma:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Groep | 4 weken | | 8 weken | 4 weken |  |
| Met IT achtergrond | OCA training Masters | Behalen  OCA | Gezamelijke projecten | Inzetbaar voor klant | |
| Zonder IT achtergrond | Voorprogramma | | Gezamelijke projecten | OCA training | Behalen OCA |

## Inhoud van het 4-weeks voorprogramma

Het voorprogramma is erop gericht om de instromers zonder IT achtergrond zodanig “ op stoom” krijgen dat ze mee kunnen draaien in een projectgroep van 4 man waarbij er minimaal één persoon een IT achtergrond heeft. Doel van het complete voorprogramma is voor hen dat ze minder moeite zullen hebben om

Vragen die zich hierbij voordoen:

* Welke werkomgeving/tools gebruiken we?
* Wat krijgen de instromers “cadeau” en wat moeten ze zelf bedenken of toevoegen?
* Wel of geen front end gebruiken?
* Waar moeten de cases op gericht zijn? Wat is het beoogde resultaat?

Belangrijke randvoorwaarden:

* Tijd is beperkt, voorkennis is beperkt
* Er is begeleiding op deze instromers beschikbaar binnen deze periode door een Java docent met ervaring binnen de Java stack
* De docent maakt gebruik van een reader waarin de opdrachten staan uitgewerkt
* Er is een reader beschikbaar voor de studenten waarin de opdrachten staan
* Studenten maken gebruik van een vooraf ingerichte laptop met daarop linux (mogelijk via een virtueel image). Binnen deze installatie is Maven en een Java SDK voorgeïnstalleerd. De opdrachten zijn voorgeïnstalleerd (folder genaamd Opdrachten met daarin de java projecten in de juiste configuratie). Reden hiervoor is dat het niet de bedoeling is dat de studenten hiermee teveel tijd kwijt zijn. Bij sommige opdrachten is het juist de bedoeling dat er een goed maven project wordt ingericht, binnen die folder zijn in dat geval alleen de begin sources aanwezig. Voor het tekenen van UML is er een geschikt tekentool aanwezig. Voor het schrijven van use cases is er een geschikte wordprocessor aanwezig.

## Randvoorwaarden en bepalende omstandigheden

Vier weken is een korte tijd. De instromers zijn vermoedelijk uit beroepen als fysiotherapeut, bioloog, omgeschoolde administratieve kracht. Elementaire computerkennis en affiniteit met software wordt wel verondersteld. De instromer heeft al eens iets geprogrammeerd, bijvoorbeeld in Basic, kan met Windows en met browsers omgaan en met een tekstverwerker.

Van Java zelf wordt geen voorkennis verondersteld, wel van een procedule taal. Omdat het voorprogramma nieuw is en op zich al tot extra kosten leidt voor Ordina wordt verondersteld dat er voor het voorprogramma geen aparte docent is maar wel collega’s uit JTech beschikbaar zijn om op afroep af en toe de instromer wat op weg te helpen.

Leerdoelen   
  
Het beheersen van de volgende elementen is voor instromers zonder IT achtergrond een voorwaarde.

* inzicht in het uitwerken van een probleem in een algoritme (al dan niet met UML, use cases, basale data)
* Omzetten van het algoritme in Java programmacode: input/output, testen, simpel front end, data: tekst files, eenvoudige collections
* Enige ervaring in de tooling op basisniveau basisniveau, build tool (bijv IntelliJ, maven)

Enige beheersing van genoemde onderdelen vormen dan ook de waterscheiding tussen de eerste en de tweede groep.

Hieronder staat een lijstje van leerdoelen die gewenst zijn om deze groep op stoom te krijgen met daarachter het middel en de geschatte tijdsduur. De totale tijdsduur is uiteraard 4 weken, de voorgestelde verdeling van de beschikbare tijd over de lesdoelen staat in de laatste kolom:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Leerdoel | Lesactiviteiten | Tools | Gedetailleerde activiteiten | Verdeling qua tijd |
| 1. Ontwerpen: Kunnen omzetten van een probleem in een algoritme | Schrijven/uitbreiden van use cases, vaststellen van technisch ontwerp | Tekstverwerker  Tekentool | Use case schrijven Algoritme bedenken  Technische benodigdheden op een rijtje zetten | 20% |
| 1. Programmeren: Kunnen omzetten van algoritme naar java code | Case omzetten naar programmacode | omgeving ingerichte projecten  maven (command line)  GIT  Browser | Java programmeren invoer/uitvoer bekijken  Fouten leren opsporen | 60% |
| 1. Kunnen testen van de eigen programmacode | Testen van de eigen programmacode | JUnit tests | Java programmeren (JUnit, maven aanschoppen)  Invoer/uitvoer vergelijken met testcases | 15% |
| 1. Om kunnen gaan met simpele data | Gebruik van de meest simpele Java collections en bestanden | Maakt deel uit van 1., 2. En 3. | Maakt deel uit van 1, 2 en 3 | Maakt deel uit van 1, 2 en 3 |
| 1. basis van versiebeheer kunnen gebruiken | Basis van versiebeheer (GIT) en samenwerken op 1 repository | GIT | Versies ophalen en naar repo pushen, samenwerken binnen GIT | 5% |

## Leerdoelen

De leerdoelen zullen gezien de eerder genoemde randvoorwaarden zeer beperkt moeten worden gesteld. Om deze groep op een aantrekkelijke en effectieve manier “aan het werk” te zetten zie ik als voorwaarde vooral uit te gaan van voorbeelden en een zo kant en klaar mogelijke plek om hun werk te doen zonder (in dit stadium) onnodige afleidingen.

Voor de volledigheid hieronder een plaatje dat schematisch aangeeft waar in de leerpiramide het voorprogramma moet worden geplaatst versus de leerinspanning voor OCA:

|  |
| --- |
| Voorprogramma |
| Afbeelding met tekst  Automatisch gegenereerde beschrijving |

|  |
| --- |
| OCA certificaat programma |
| Afbeelding met tekst  Automatisch gegenereerde beschrijving |

De piramides zijn zeer verschillend en moeten dat ook zijn. Zonder begrijpen en toepassen kom je niet door de grote hoeveelheid “weet” kennis die noodzakelijk is voor het behalen van het OCA certificaat.

## Werkomgeving/tools

* Een vooraf ingerichte werkomgeving (zodat ze niet of zeer beperkt zelf tools moeten installeren en de omgeving moeten inrichten)
* Uitgaan van het leren via voorbeelden.

Deels uitgewerkte cases (dus ook voorzien van deels kant en klare programmacode) die ze moeten aanvullen. Op deze manier worden ze op weg geholpen en kunnen ze zien “hoe iets moet” alvorens zelf iets te proberen.

Een mogelijke wijze om hier rekening mee te houden is bijvoorbeeld om ze te voorzien van een laptop met een virtueel image van een Linux versie met daarop een kant en klare ingerichte omgeving die gericht is op dit voortraject, waarop de voorbeelden aanwezig zijn.

Welke aspecten gaan we raken?  
  
Het OCA traject dat uiteindelijk volgt raakt werkelijk ongeveer alle denkbare uithoeken van de taal Java. Het is absoluut ondoenlijk en ook niet verstandig om zij-instromers van een voorprogramma te voorzien waarbij al deze aspecten worden geraakt. Zeker het voorprogramma van de zij-instromers zonder IT achtergrond zal zeer beperkt van opzet moeten zijn. Doel is om ze vertrouwd te maken met de omgeving, ze een aantal voorbeelden te laten uitwerken en daardoor vertrouwd te maken met de werking van Java en van het omzetten van een probleem met een werkend programma en om ze begrippen te leren die daarbij van belang zijn.

Front end of geen front end? Gebruik van een IDE of geen IDE?  
  
Een java applicatie is doorgaans voorzien van een front end en een back end en werkt doorgaans via een controller die beide delen met elkaar verbindt. Het front-end is daarbij geen java maar een framework dat een webpagina genereert. Het back end is een datasource doorgaans verbonden aan een relationele database.

Zowel front end als back end zijn dus geen Java: het back end is bijvoorbeeld JPA die de verbinding vormt met een relationele database, het front end bijvoorbeeld javascript met een framework als Thymeleaf of AngularJS dat webpagina’s genereert en de applicatie hiermee verbindt. Dit zijn allemaal elementen die complex zijn en daardoor het gevaar opleveren dat ze een sta-in-de-weg zijn voor onze primaire doelen.

Hoewel het aantrekkelijk is dat beginnende javanen Java leren kennen binnen het ecosysteem waar Java geplaatst is voor een klant van Ordina mogen deze technieken niet tijd of energie opsouperen voor de beginnende instromers.

Front end = Terminal  
Java vanaf de Command line

Voor de eerste week of weken gebruiken we cases waarbij alleen vanaf de command line gebruikt wordt. Er is geen front end of de case zorgt voor een simpel “menu” dat telkens op het scherm in de vorm van tekst wordt afgedrukt. Invoer vindt plaats achter door het Java programma afgedrukte prompts, uitvoer uiteraard naar hetzelfde terminal scherm.

Frond end = kant en klare webservice  
Er is een webservice die de instromer “cadeau” krijgt. Zijn programma wordt gevoed door een webpagina dat verbonden is aan de webservice, de webservice stuurt de invoer door aan zijn Java programma. De uitvoer van zijn java programma gaat naar de webservice die dit weer naar het scherm stuurt. Het hele front end en de functionaliteit om de in/uitvoer van de instromer met het front end uit te wisselen krijgt de instromer cadeau.

Wat ziet de instromer hiervan?  
- Door een maven goal aan te roepen doen vanaf de command line wordt zowel zijn programma als het front end gecompileerd en gerund. Dit is een vast commando dat moet worden ingetypt.

* Door in een browser naar de URL te gaan voor de opdracht wordt het scherm getoond en kan de gebruiker het eigen Java programma testen.
* De uitvoer wordt op het scherm getoond, logging komt in het terminal venster terecht.

Wat zijn nu de voor- en nadelen van deze opzet?

**Voordeel**: er komt een aansprekende applicatie uit die een en ander toont wat het programma in Java tot stand brengt. Voordeel is een representatieve Java applicatie waarin zowel een front-end als een backend aanwezig is. Het leidt de aandacht van de leerdoelen wel wat af maar maakt de ervaring een stuk rijker.

**Nadeel** is een (beperkte) hoeveelheid extra moeite. De inspanning van de instromer op dit punt mag niet teveel tijd opsouperen: vandaar dat het front end grotendeels cadeau wordt gegeven. Relatief nadeel is dat er enige begeleiding door een collega javaan waarschijnlijk wel nodig is bij het voor het eerst gebruik maken van dit type opdrachten.

Het front end in deze opzet kan nog in twee smaken worden aangeleverd:

* 1. Per opdracht, kant en klaar front end
* 2. Generieke oplossing waarbij de webservice start door een html pagina met javascript in te lezen en over de eigen startpagina heen te kopiëren – dank voor het idee van deze opzet gaat naar Leon Pocornie. Voordeel van deze oplossing is dat de instromer nog in beperkte mate de pagina kan aanpassen zonder zich teveel in front end technieken te hoeven verdiepen.

Stiekum heb ik tevens de omgeving grotendeels beschreven. Hieronder een korte beschouwing van de tooling:

Tooling

Iedere opdracht is voorzien van een vooraf ingerichte folderstructuur en tevens de pom.xml configuratiebestand voor het laden van plugins en libraries voor het betreffende probleem en daarnaast een application.properties bestand.  
Een zo generiek mogelijke opzet kan ervoor zorgen dat deze structuur grotendeels per opdracht kan worden gekopiëerd.

Maven  
Het compileren, testen en runnen van het programma gebeurt met Maven. Maven is voorgeïnstalleerd op de laptop van de instromer. Configuratie gebeurt geheel met de bestanden (application.properties en pom.xml).

Jar bestand  
Het runnen van (apart) het jar bestand maakt deel van minstens één van de opdrachten, liefst van een paar opdrachten. Het is niet moeilijk en geeft een goede basis.

Libraries

Het importeren van de benodigde libraries wordt gedemonstreerd binnen de java code. Benodigde libraries en het feit dat die in de pom.xml te vinden zijn wordt gedemonstreerd bij een aantal van de opdrachten. De uiteindelijke pom.xml krijgen de studenten wel cadeau.

Internet  
Via internet zijn er heel veel voorbeelden beschikbaar die via een youtube filmpje zijn te volgen. Ik stel voor om voor enkele opdrachten ook hier gebruik van te maken binnen de context van waar de student zich op dat moment bevindt. Bijvoorbeeld maven en command line. Het levert een fijne afwisseling, een menselijke stem en het gevoel dat de student zelf in staat is om een aantal dingen ook zelf op te zoeken.

GIT   
Omdat het niet zo moeilijk is om de basisbeginselen van het samenwerken in één project met gedeelde sources bij te brengen stellen wij ons voor dat ook GIT gebruikt wordt binnen de introductiecursus. Vanaf de command line. GIT is voorgeïnstalleerd en voorgeconfigureerd met een gemeenschappelijke repository binnen github.

Tekstverwerker  
Een platte tekstverwerker naar keuze

IntelliJ (of andere IDE)  
Geen Eclipse maar in plaats daarvan IntelliJ roept men hier in koor. Ik sluit mij daarbij aan: Eclipse heeft een zeker voor beginners heel lastige GUI, IntelliJ levert minstens dezelfde functionaliteit. Hoewel gezegd kan worden dat er “teveel wordt geholpen” (voor wat betreft de Java syntax en mogelijke programmeerfouten) is het zeker voor absolute beginners wel fijn om via de IntelliJ IDE ondersteuning te krijgen bij het programmeren. Voordeel: helpt bij het programmeren, libraries die ontbreken en dergelijke worden aangegeven, mogelijkheid tot door het programma heen stappen, IDE geeft ook meer overzicht over de folderstructuur – Nadeel: wel zelf de syntax ook leren svp, niet alleen maar leunen op IntelliJ. Nadeel: leren van interface van IntelliJ kost ook weer kostbare tijd. Zelf de folderstructuur inrichten geeft ook inzicht hoe maven en Java eigenlijk werken.

# Opdrachten voor de 4weekse intro

Het is waarschijnlijk handig om een vaste structuur te hanteren voor het opzetten van de opdrachten zodat bij iedere opdracht de volgende metagegevens te vinden zijn, zoals:

* Leerdoel
* Benodigdheden
* Invoer
* Uitvoer
* Processing
* Uitleg
* Tijdsduur in uren
* Uitdagingen voor de student
* Succescriteria

De benodigden zijn de tools die nodig zijn (terminal, GIT, maven, enz) en de bestanden die nodig zijn. Dat laatste item mag een verwijzing zijn naar een vooraf ingerichte folder met daarin de startcode van het webproject.

Invoer, uitvoer en processing beschrijven op hoofdlijnen wat het programma doet als het klaar is. Uitleg is hier een heel korte beschrijving van het doel van de opdracht. Uitdagingen voor de student zijn de hoepels waar de student doorheen moet springen om tot het einddoel te komen. Succescriteria zijn de “nakijk” criteria aan de hand van deze criteria kan de student zelf vaststellen of hij de opdracht goed heeft volbracht.

#### Outline van de vier weken

Eerste week hieronder is uitgewerkt. Voor de latere weken volgt de opbouw uit de cases.

### Week 1

Onderwerpen: Leren kennen van de Java omgeving, leren omgaan met de tools. Simpele in-en uitvoer, compileren en uitvoeren via de command line.  
Simpele applicatie ontwikkelen. Variabelen en elementaire datatypen. Simpel object.

Vertrouwd raken met het werken met objecten.  
Alle belangrijkste java besturingselementen: while loop, if statement, enkele belangrijke standaardbibliotheken.

Hoe schrijf ik een use case

Intro GIT

### Week 2

Intro IntelliJ

Import van libraries, opzoeken van veel gebruikte Java libraries.

Object orientatie.

Lijsten van objecten maken en gebruiken.

JUnit om te testen. Try Catch

Hoe foutscenario’s toe te voegen aan de use cases

Hoe ontwerp en schrijf ik een test

### Week 3

Simpel front end (command line front end)  
Simpel back end (bestanden lezen en schrijven)

Case huishoudboekje uitwerken

Javadoc inzien om verder gebruik te maken van libraries (IntelliJ)  
Javadoc genereren

### Week 4

Hoe schrijf ik een integratietest

Front end met Angular

Webservice controller

Backend verder uitwerken.   
Java gebruiken in samenhang met Angular front-end

Verder met het back-end dat we in week 3 hebben uitgewerkt.

Schermen tonen/testen op verschillende manieren.

Case over huishoudboekje nu uitwerken en gebruiken als webapplicatie.

# Cases Week 1

|  |  |  |
| --- | --- | --- |
| Case 1.1 | HelloWorld | |
| Doel | Intro Java en simpel programma compileren | |
| Benodigdheden | | Java jdk, maven, terminal, startprogramma case 1 folder |
| Invoer | | geen |
| Uitvoer | | Hello world zinnetje |
| Processing | | Geen |
| Uitleg | | Student moet programma kunnen compileren en runnen, begrijpen hoe de code werkt |
| Tijdsduur in uren | | 1 uur |
| Uitdagingen voor de student | | Compileer commando weten te gebruiken |
| Succescriteria | | Uitvoer staat op het scherm |

|  |  |  |
| --- | --- | --- |
| Case 1.2 | Invoer en uitvoer command line | |
| Doel | Programma gebruiken dat invoer ontvangt van de command line en uitvoer naar terminal produceert | |
| Benodigdheden | | Java jdk, maven, terminal, startprogramma case 1.2 folder |
| Invoer | | Naam invoeren |
| Uitvoer | | Ingevoerde naam van de student |
| Processing | | Geen |
| Uitleg | | Student moet programma kunnen compileren en runnen, begrijpen hoe de code werkt |
| Tijdsduur in uren | | 1 uur |
| Uitdagingen voor de student | | Begrijpen hoe in- en uitvoer werkt. |
| Succescriteria | | Na invoer van de naam staat naam op het scherm “hallo Simon”. |

.. enzovoorts.