Opdracht JSF32, week 6

In deze opdracht experimenteren we met:

* Java Bytecode

Om goed met bytecode te kunnen werken, is het noodzakelijk dat je een overzicht hebt van wat het is, en hoe het gebruikt wordt. Lees hiertoe eerst het artikel:

JAVA bytecode: understanding bytecode makes you a better programmer <http://www.mastercorp.free.fr/Ing1/Cours/Java/java_lesson1/doc/Articles/Decompilation/Java%20bytecode.htm>.

Voor een overzicht van alle bytecode opcodes zie:

<http://en.wikibooks.org/wiki/Java_Programming/Byte_Code>

Voor de echte diepgraver: De exacte JVM specificaties zijn te vinden in:

<http://docs.oracle.com/javase/specs/jvms/se5.0/html/Instructions2.doc.html>

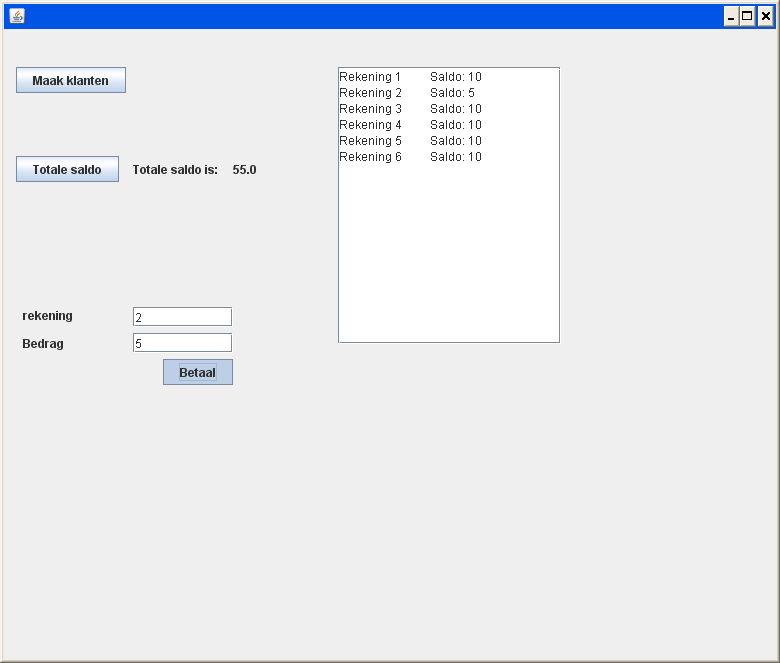
Verder heb je een bytecode editor nodig. gebruik hiervoor JBE (<http://www.cs.ioc.ee/~ando/jbe/>). Start deze door het runnen van de batchfile 'jbe.bat'.

Indien je de classfiles te ver verminkt hebt, kun je de originele .class files weer terugzetten vanuit de file 'build\_origineel.zip'.

# De opdracht

Gegeven is een applicatie OS\_ByteCodeOpgave, die je kunt runnen door het uitvoeren van de batchfile OS\_ByteCodeOpgave.bat.

Je ziet dat de sourcecode niet volledig meegeleverd is. Wel staan alle class files tot je beschikking.

Na start van de applicatie zie je het volgende window: 

Deze GUI stelt een proof of concept applicatie voor een bank voor.

Een bank bestaat uit klanten, en elke klant heeft één rekening. Na het drukken op de button 'Maak klanten' worden er 6 klanten aangemaakt, die ieder een bonus van 10 op hun rekening krijgen.

Achter het label: 'Totale saldo is:' wordt het totale saldo van alle rekeningen van de bank weergegeven, na het drukken op button 'Totale saldo'.

In de textarea rechts wordt per rekening het aanwezige saldo aangegeven.

Onderaan kun je voor één rekening een betaling uitvoeren, waardoor het saldo wijzigt.

**Opdracht 1**

In deze opdracht gaan we bytecode lezen en interpreteren.

a. Zoek in de classfiles (dus niet in de source code) uit, welke methodes aangeroepen worden bij:

het berekenen en tonen van het totale saldo (geef classfile en regelnummers in de classfile)

Bank.class Regelnummer: 46

het betalen voor een rekening (geef ook weer classfile en regelnummers in de classfile)

Bank.class Regelnummer: 49

b. In de BankFrame$x.class files worden ook klassen opgeslagen, die binnen de BankFrame klasse gebruikt worden.   
In welke regels van de source-code van BankFrame.java worden deze klassen gedefinieerd?

98, 134, 169 (bankframe$4 niet)

c. Welke attributen (Fields) en methoden heeft de Klant klasse (geef naam en type)?   
Methode <init> hoeft niet; dat is de constructor.

Attributen:

naam : String, mijnRekening : Rekening

Methoden:

voegRekeningToe : void, mijnSaldo : int, getRekening : Rekening, verlaagSaldo : void

Reconstrueer voor elke methode de originele java-code:

methode voegRekeningToe:

0 aload\_0 // zet this op stack

1 aload\_1 // zet 1e parameter op stack

2 putfield #3 <os\_bytecodeopgave/Klant/mijnRekening Los\_bytecodeopgave/Rekening;>

// ken 1e parameter toe aan veld mijnRekening van this (this.mijnRekening)

5 return // keer terug uit methode

Dus de java-code was:

public void voegRekeningToe(Rekening rek) {

this.rekening = rek;

}

methode mijnSaldo

0 aload\_0 //Zet this op stack

1 getfield os\_bytecodeopgave/Klant/mijnRekening Los\_bytecodeopgave/Rekening; //Haal mijnRekening op

4 invokevirtual os\_bytecodeopgave/Rekening/getSaldo()I //Haal het saldo van mijnRekening op

7 ireturn //Return het saldo als int

public int mijnSaldo()

{

return mijnRekening.getSaldo();

}

methode getRekening

0 new os\_bytecodeopgave/Rekening //Maak nieuw Rekening aan

3 dup //

4 aload\_0 //Zet this op stack

5 getfield os\_bytecodeopgave/Klant/mijnRekening Los\_bytecodeopgave/Rekening; //Haal mijnRekening op

8 invokevirtual os\_bytecodeopgave/Rekening/getNummer()I //Haal het rekeningnummer van mijnRekening op

11 invokespecial os\_bytecodeopgave/Rekening/<init>(I)V //Initialize de nieuw aangemaakte Rekening met het opgehaalde rekeningnummer

14 astore\_1 //Store het rekeningnummer

15 aload\_1 //Zet 1e parameter op de stack

16 aload\_0 //Zet this op de stack

17 getfield os\_bytecodeopgave/Klant/mijnRekening Los\_bytecodeopgave/Rekening; //Haal mijnRekening op

20 invokevirtual os\_bytecodeopgave/Rekening/getSaldo()I //Haal het saldo van mijnRekening op

23 invokevirtual os\_bytecodeopgave/Rekening/changeSaldo(I)V //Pas het saldo aan

26 aload\_1 //Zet de 1e parameter op de stack

27 areturn //Return de nieuwe Rekening

public Rekening getRekening()

{

Rekening rekening = new Rekening(mijnRekening.getNummer());

rekening.changeSaldo(mijnRekening.getSaldo());

return rekening;

}

methode verlaagSaldo

0 aload\_0 //Zet this in de stack

1 getfield os\_bytecodeopgave/Klant/mijnRekening Los\_bytecodeopgave/Rekening; //Haal mijnRekening op

4 iload\_1 //Haal de eerste parameter op (int)

5 invokestatic java/lang/Math/abs(I)I //Neem de absolute waarde hiervan

8 ineg //Maak het bedrag negatief

9 invokevirtual os\_bytecodeopgave/Rekening/changeSaldo(I)V //Pas het saldo aan met de gegeven parameter (int)

12 return //Keer terug uit de methode

public void verlaagSaldo(int saldo)

{

mijnRekening.changeSaldo(Math.abs(saldo));

}

**Opdracht 2**

In deze opgave gaan we de bonus bij het openen van een rekening eens misbruiken.

Zoek waar in de classfiles de rekeningen aangemaakt worden, en in welke methode de bonus op de rekening gezet wordt.

Pas deze methode aan, zodat een hogere bonus gegeven wordt. Toon dit met een hogere bonus van 100 en van 1000.

*Bonus van 100 werkt. Bij het wijzigen van de bonus van 1000 wordt de volgende error gegeven:*

*“Error: Instruction requires an argument of type byte at line 6, instruction "bipush".” Dit komt doordat ‘bipush’ slechts 1 byte kan meegeven, terwijl 1000 meer dan 1 bytes bevat. De oplossing hiervoor is om ‘sipush’ te gebruiken, omdat dit 2 bytes kan meegeven.*

**Opdracht 3**

In deze opdracht gaan we ervoor zorgen, dat klanten onbeperkt kunnen opnemen van hun rekening, zonder dat hun saldo verandert. Pas hiertoe methode verlaagSaldo van de Klant klasse aan. Toon aan dat je applicatie werkt door hem te runnen.

*We hebben “invokevirtual os\_bytecodeopgave/Rekening/changeSaldo(I)V” weggehaald.*

**Opdracht 4**

Hetzelfde als opdracht 3, maar nu alleen voor rekening 4. Rekening 4 mag dus niet wijzigen bij betalen, de andere rekeningen wel. Breng weer een verandering aan in verlaagSaldo van Klant.

Gebruik hiervoor een opcode die vergelijkt met een constante integer waarde (4). Deze integer waarde moet eerst op de stack gepusht worden.

verlaagSaldo methode

**aload\_0** //Zet this op stack

**getfield os\_bytecodeopgave/Klant/mijnRekening Los\_bytecodeopgave/Rekening;** //Get rekening

**invokevirtual os\_bytecodeopgave/Rekening/getNummer()I** //Zet rekeningnummer op stack

**bipush 4** //Zet getal 4 op stack

**if\_icmpeq 12** //Vergelijk de laatste integers op de stack en if equals dan ga door naar regel 12 (return)

aload\_0

getfield os\_bytecodeopgave/Klant/mijnRekening Los\_bytecodeopgave/Rekening;

iload\_1

invokestatic java/lang/Math/abs(I)I

ineg

invokevirtual os\_bytecodeopgave/Rekening/changeSaldo(I)V

return

**\*Dikgedrukt is toegevoegd**