Informatica

Leren programmeren met JAVA en NetBeans

Over JAVA en Rapid Application Development met NetBeans

Bert Van den Abbeele

Leren programmeren met JAVA en NetBeans

Over JAVA en Rapid Application Development met NetBeans...

door

Bert Van den Abbeele

Inhoudsopgave

Inleiding

In deze cursus leert u Java applicaties op te bouwen. We maken gebruik van de IDE (Integrated Development Environment) NetBeans. We wensen zo snel mogelijk een werkende applicatie te hebben. Hiervoor gebruiken we de formulier/label-ontwikkelaar van NetBeans, deze maakt RAD (Rapid Application Development) ook binnen Java mogelijk.

Tijdens het opbouwen van de applicaties en de oefeningen komen alle belangrijke aspecten van programmeren en van de programmeertaal Java aan bod.

Veel succes!

Bert Van den Abbeele

Probleemoplossend denken

Als we een programma maken moeten we denken aan het gegevensverwerkend proces.

De oplossing van een probleem kunnen we niet uit het hoofd bedenken. We moeten probleemgericht werken. Laten we eerst het probleem definiëren, dan een analyse maken en vervolgens een schema. Zo delen we ons probleem in stukken en maken we een aantal stroomdiagrammen, controlestructuren (bvb: Nassi Schneidermann diagrammen) of pseudocodes.

De stappen bij elke oefening zijn de volgende:

Compiler

De computer begrijpt enkel **machinetaal**. Elke processor wordt op een bepaalde manier aangestuurd om berekeningen te maken en met het geheugen te werken. Rechtstreeks in machinetaal schijven is bijna onmogelijk. Gelukkig bestaan er programma's die een voor de mens meer verstaanbare taal kan omzetten naar machinetaal. Zo'n programma heet een **compiler**.

Broncode ► Compiler voor Windows en x86-64 processor ► programma.exe ► Uitvoer

De tegenhanger van compileren is **interpreteren**. Hierbij wordt de code, lijn per lijn, ingelezen en vertaald voor de processor.

Broncode ► ► ► **Uitvoer**

Lijn per lijn...

Wanneer een executable (.exe bestand) gemaakt wordt door de compiler, gebeurt er vaak nog veel meer dan enkel het omzetten. De compiler zal eerst de **debugger** activeren die de broncode doorleest op mogelijke fouten.

De naam bugs stamt uit de tijd toen de computer nog geen gebruik maakte van chips, maar van elektronische schakelaars. Er waren toen ooit "bugs", namelijk kevers, die een programma door kortsluiting in de war bracht.

De debugger toont waar de fout optreedt en geeft hierbij goede raad. Ook wanneer de bugs verdwenen zijn zal je uiteindelijk aanpassingen moeten doorvoeren. Je kan je de vraag stellen of software ooit echt afgewerkt is. Het is zeer belangrijk dat je tussen de broncode **commentaar** plaatst, zodat je achteraf de code kan analyseren.

We vertelden reeds dat elke processor en elk besturingsysteem zijn eigen instructieset heeft. De compiler zorgt ervoor dat een programma slechts op een beperkt aantal computers zal werken. Een oplossing voor dit nadeel is het werken met een "*runtime library*". Hierbij wordt er niet naar een machinetaal maar naar een runtime-taal gecompileerd (=**intermediate language**). Wanneer je een runtime installeert, controleert deze je besturingsysteem en hardware. Het gecompileerde programma wordt via de runtime geoptimaliseerd voor het besturingsysteem en de software. Op deze manier moet de programmeur maar één versie van zijn programma maken.

Broncode > Compileer onafhankelijke intermediate language > Uitvoer via runtime library

Enkele voorbeelden:

- Java heeft zijn Virtual Machine "Write once, run anywhere."
- Common Runtime Library (CLR) van het (Microsoft) .NET Framework

De compiler die wij in deze cursus gebruiken is de **Java Development Kit (JDK)**. Deze is vrij te downloaden op http://java.sun.com

De compiler zelf heet **javac** en is terug te vinden in de installatiedirectory.

De programmeertaal JAVA

Java is een **objectgeörienteerde programmeertaa**l ontwikkeld door Sun Microsystems. Het belangrijkste kenmerk is zijn platformonafhankelijkheid via de Java virtual machine (JVM).

Objectgeörienteerd betekent dat we werken met **objecten**. Objecten worden gedefinieerd in **klassen**. Een klein programma kan slechts uit één klasse bestaan.

```
public class MijnNaam {
}
```

Hierboven definieerden we de publiek toegankelijke klasse “MijnNaam”.

De functionaliteit van ons progamma bevindt zich binnen de klasse in **methoden**. We kunnen methoden maken die berekeningen maken of die gegevens op het scherm weergeven.

Elk programma bestaat minstens uit één methode: de **main** methode. De main methode is het startpunt van het programma en daarom onmisbaar.

```
public static void main (String[] args) {
}
```

Hierboven definieerden we de publiek toegankelijke statische methode “Main”. De methode main geeft geen waarden terug, vandaar het woord **void**.

Elke programmeertaal beschikt over een **klassenbibliotheek**. Deze zijn hierachisch opgebouwd en verzorgen de functionaliteit binnen de programmeertaal. Het is ook mogelijk bibliotheken (**libraries**) toe te voegen.

Een consoleapplicatie maken

Stap 1: Schrijf onderstaande code in NotePad

```
public class MijnNaam {
     public static void main (String[] args) {
          System.out.println("Mijn naam!");
     }
}
```

Stap 2: Sla deze tekst op als mijnnaam.java! Sla dit op in de map waar de Java compiler aanwezig is, standaardlocatie: C:\jdk1.3.1_16\bin.

Stap 3: Ga naar de opdrachtprompt en compileer deze code met de JDK:
typ: javac MijnNaam.java
(Deze maakt MijnNaam.class aan!)

Stap 4: Start jouw programma met de JDK:
typ: java mijnnaam

Een applet maken

Stap 1: Schrijf onderstaande code in NotePad

```
import java.applet.Applet;
import java.awt.Graphics;

public class applet extends Applet {
     public void paint( Graphics g ) {
          g.drawString( "Een voorbeeld!", 25, 25 );
     }
}
```

Stap 2: Sla deze tekst op als applet.java! Sla dit op in de map waar de Java compiler aanwezig is, standaardlocatie: C:\jdk1.3.1_16\bin.

Stap 3: Ga naar de opdrachtprompt en compileer deze code met de JDK:
typ: javac applet.java
(Deze maakt applet.class aan!)

Stap 4: Maak een HTML pagina aan:

```
<HTML>
     <HEAD>
          <TITLE>Uw eerste applet</TITLE>
     </HEAD>
     <BODY>
          <APPLET CODE = "applet.class" width=130 height=40>
          </APPLET>
     </BODY>
</HTML>
```

Stap 5: Plaats applet.class bij het HTML document!

Stap 6: Kijk of uw eerste applet op de pagina wordt geladen!

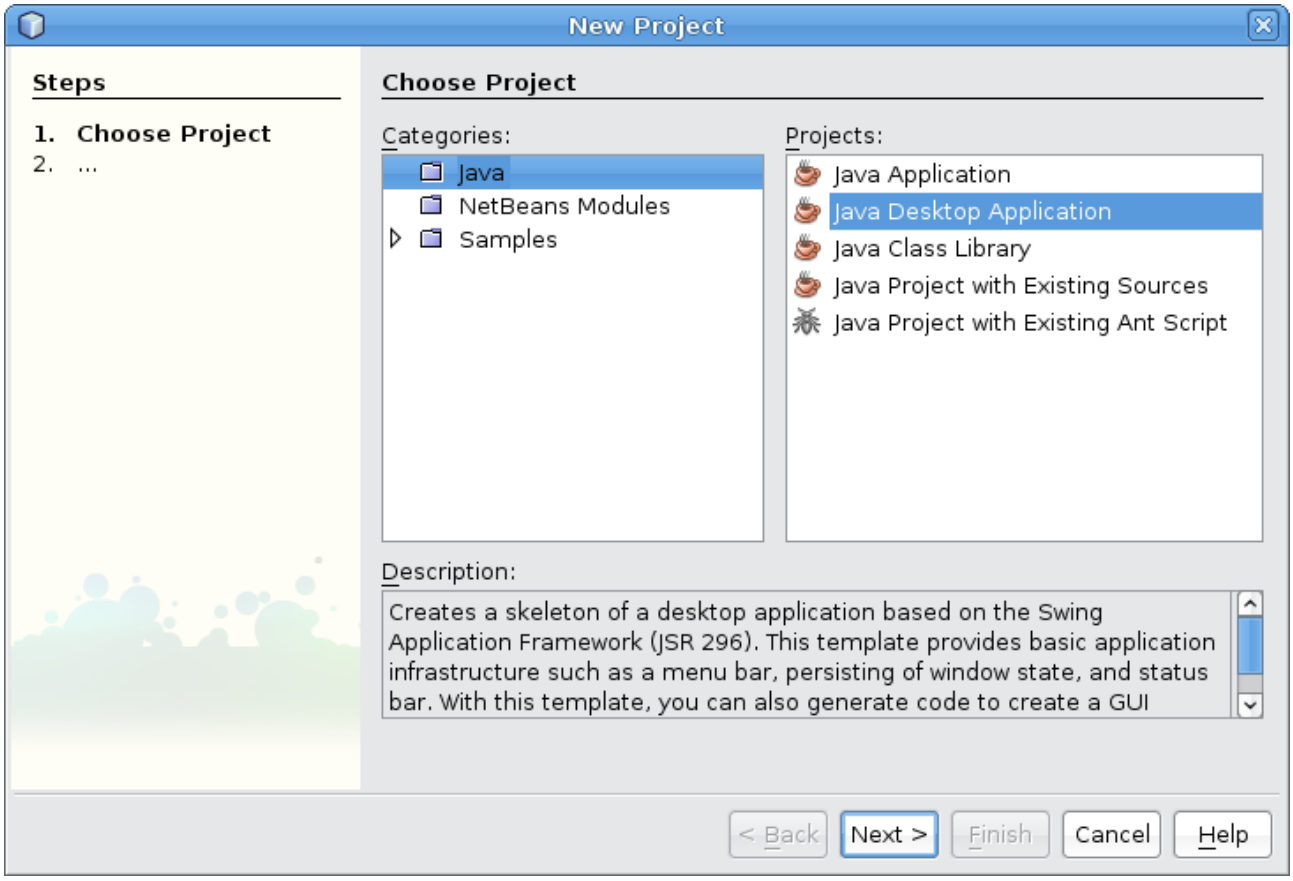
Tip: Alternatief voor Notepad

In plaats van Notepad zijn er vele teksteditoren die code inkleuren. Sommigen kunnen zelfs fouten detecteren (bvb.: niet afgeloten acolades). Een interessant alternatief voor Notepad is NotePad++. Deze simpele teksteditor maakt syntax kleuring, lijnnummering en inspringen mogelijk. Je kan deze downloaden http://notepad-plus.sourceforge.net/.

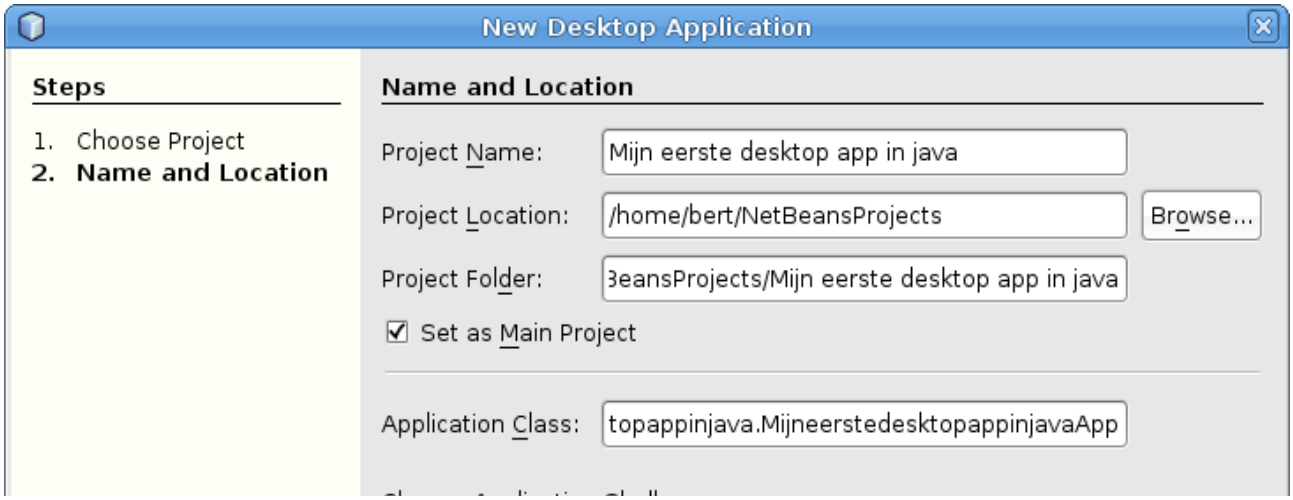
Mijn eerste java applicatie met NetBeans

We starten een nieuw project: File > New project...

Kies voor Java Desktop Application en vervolgens op Next >.

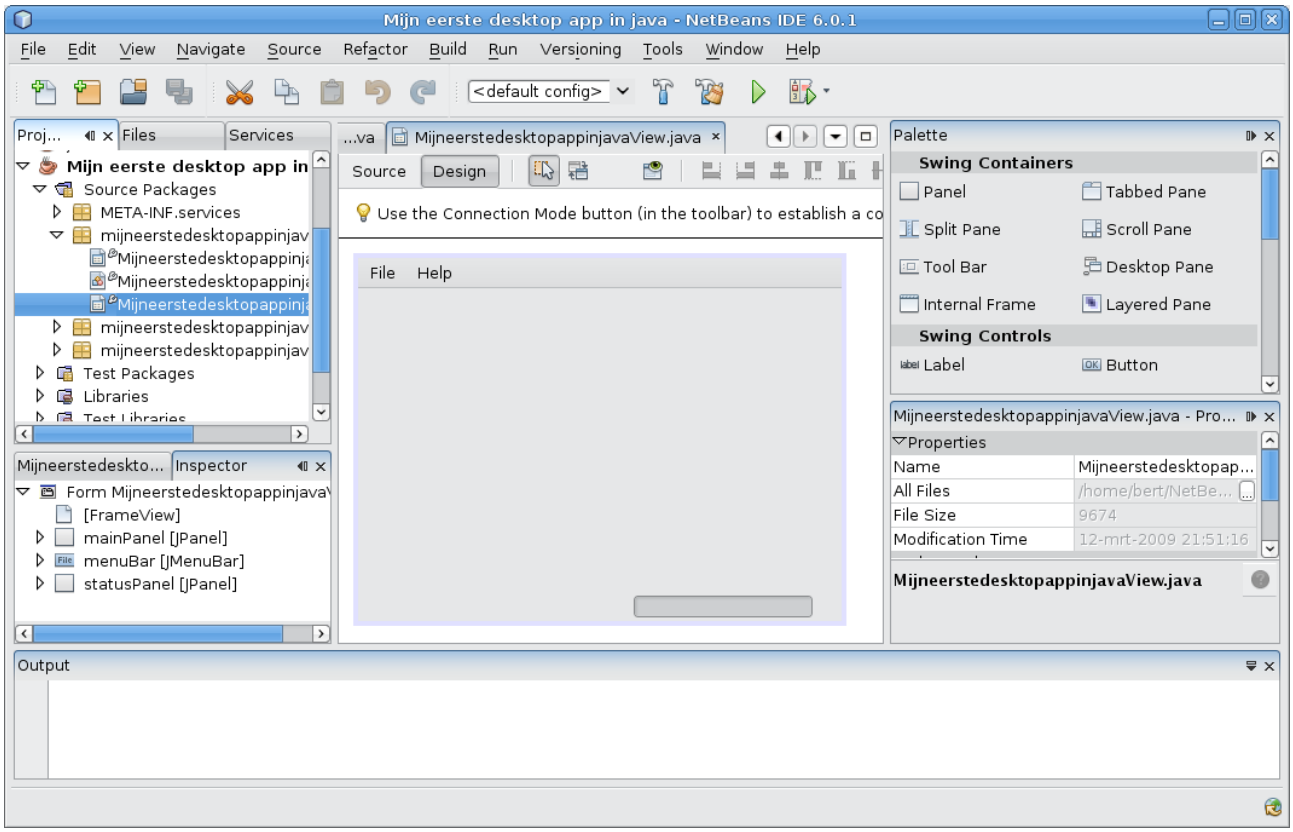


Geef je project een naam, locatie en kies basic application.



Klik op Finish en je applicatie wordt opgebouwd

We bestuderen de onderdelen van de IDE NetBeans. IDE staat voor Integrated Development Environment.



Zoals elk computerprogramma is er een titelbalk, menubalk en werkbalk aanwezig. De werkbalk bevat een zeer belangrijke knop.

De “run main project”-knop: 

Hiermee starten we onze applicatie. Er zijn ook knoppen aanwezig voor het opbouwen en debuggen van het project, deze komen later aan bod.

Het midden van het werkveld is opgedeeld in drie onderdelen. Links zijn onze bestanden terug te vinden. In het midden vinden we het actieve bestand terug. Rechts is het palet met besturingselementen en eigenschappen terug te vinden.

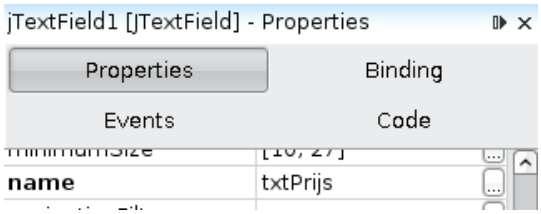
Onderaan het werkveld vinden we de balk output. Deze geeft ons informatie tijdens een run, build of debug. Dit is de plaats waar foutmeldingen tijdens het uitvoeren van het programma tevoorschijn komen.

Onze applicatie is een desktop applicatie. Daarom zijn er twee weergaven van ons bestand. We hebben de broncode- en de designweergave. De designweergave maakt het mogelijk besturingselementen op het formulier (paneel) te plaatsen en de eigenschappen ervan aan te passen. De source is de onderliggende code van ons programma. We kunnen tussen beide weergaven schakelen door de knoppen “Source” en “Design”.

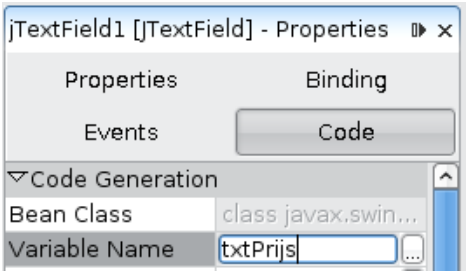
Besturingselementen toevoegen aan de applicatie

Sleep drie tekstvelden (**Text Field**), drie labels (**Label**) en één knop (**Button**) op het paneel. Versleep de elementen en gebruik de randen van objecten om deze te herschalen. Op de achtergrond wordt steeds code gegenereerd voor het paneel. Het kan zijn dat hierdoor de elementen van plaats veranderen, even herschikken is de oplossing.

We moeten vervolgens de **properties** van de elementen aanpassen. Bij elk element gaan we twee eigenschappen aanpassen: de naam en de tekst.



Geef de elementen een logische naam, laat de namen starten met een afkorting van hun type. De namen worden: lblPrijs, lblAantal en lblTotaal. Als tekst krijgen ze respectievelijk Prijs, Aantal en Totaal. De knop krijgt de naam btnBereken en als tekst Bereken. De tekstvelden krijgen geen tekst mee en als naam txtPrijs, txtAantal en txtTotaal.



Vervolgens gaan we van de properties naar de Code. Hier veranderen we voor elk element de naam van de **variabele** (Variable Name) door de naam van het element.

Druk op de “run main project”-knop: 

We wisselen naar de source-weergave: 

We zien een pakket met de naam van onze eerste toepassing. Onze applicaties importeert een aantal libraries waardoor het mogelijk wordt een formulier op te bouwen.

```
package mijneerstedesktopappinjava;

import org.jdesktop.application.Action;
import org.jdesktop.application.ResourceMap;
```

Hier start onze eerste klasse met dezelfde naam als ons pakket en project. Het erft alle functionaliteiten uit de geimporteerde bibliotheken.

```
public class MijneerstedesktopappinjavaView extends FrameView {

    public MijneerstedesktopappinjavaView(SingleFrameApplication app) {...}

    @Action
    public void showAboutBox() {...}
```

Hieronder komt de code die we gegenereerd hebben in de designweergave. Klik op het +-teken om de code te bekijken.

```
    Generated Code
```

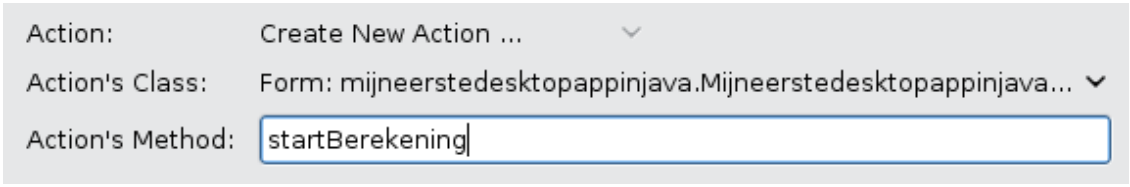
Vervolgens lees je de eigenschappen die we aangepast hebben. Deze namen zullen we straks nodig hebben om waarden uit te lezen en weer te geven in de elementen.

```
    // Variables declaration - do not modify
    private javax.swing.JButton btnBereken;
    private javax.swing.JLabel lblAantal;
    private javax.swing.JLabel lblPrijs;
    private javax.swing.JLabel lblTotaal;
    private javax.swing.JPanel mainPanel;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JProgressBar progressBar;
    private javax.swing.JLabel statusAnimationLabel;
    private javax.swing.JLabel statusMessageLabel;
    private javax.swing.JPanel statusPanel;
    private javax.swing.JTextField txtAantal;
    private javax.swing.JTextField txtPrijs;
    private javax.swing.JTextField txtTotaal;
    // End of variables declaration
```

Een actie toevoegen

We gaan een gebeurtenis toevoegen aan ons paneel. Wanneer iemand op een knop klikt is er een **event** op het paneel, aan dit event koppelen we een bepaalde actie. We keren terug naar de designweergave.

Dubbelklik op de knop berekenen. Kies voor Action: Create New Action... Geef de actie een logische naam. Klik vervolgens op OK.



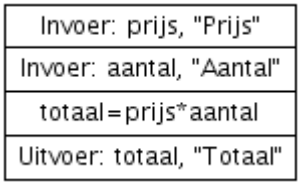
Het scherm verandert automatisch naar de sourceweergave. Er wordt een nieuwe methode aangemaakt binnen onze klasse. Hier kunnen we code binden aan onze knop.

```
@Action
public void startBerekening() {
}
```

Deze methode geeft geen waarde terug, vandaar void.

Variabelen declareren

Wanneer iemand op de knop drukt wordt de prijs en het aantal met elkaar vermenigvuldigd. Het resultaat wordt weergegeven in het tekstvak totaal.



Eerst gaan we variabelen aanmaken voor prijs, aantal en totaal. Een variabele kan verschillende **gegevenstypes** hebben. Twee belangrijke zijn numeriek en alfanumeriek. Binnen de categorie numerieke variabelen zijn er twee groepen: de **integere** (gehele) en **floats** (decimale) getallen.

integers

type	min	max	aantal bits
byte	-128	127	8
short	-32 768	32 767	16
int	-2 147 483 648	2 147 583 647	32
long	-9 223 372 036 854 775 808	9 223 375 036 854 775 807	64

floats

type	precisie	aantal bits
float	1,40129846432481707e-45 tot 3,40282346638528860e+38	64
double	4,94065645841246544e-324 tot 1,79769313486231570e+308	32

Binnen de alfanumerieke categorie zijn er ook twee mogelijkheden: **char** (karakter) of **string** (tekst). Er bestaat ook een variabele van het type **boolean**, deze variabele kent slechts twee toestanden true (waar) of false (vals). We hebben nood aan een float voor de prijs, een short voor het aantal en een float voor totaal.

```
public void startBerekening() {
     float prijs ;
     short aantal;
     float totaal;
}
```

Invoer

De volgende stap is het inlezen van de waarden uit het formulier. We kunnen alle objecten binnen onze applicatie bekijken door “**this.**” in te tikken. Vervolgens gaan we op zoek naar het tekstvak. We weten dat tekstvakken beginnen met txt. Het object tekstvak heeft een **methode** voor het uitlezen van zijn text, **getText()**. Op deze manier bekomen we *prijs=this.txtPrijs.getText()*. Deze instructie zal echter niet tot het gewenste resultaat leiden. We gaan tekst opslaan als een getal en dat kan je niet zomaar doen. Hiervoor moeten we een **typeconversie** doen.
We roepen **de klasse Float** op en zoeken de methode parseFloat. Op deze manier bekomen we *prijs=Float.parseFloat(this.txtPrijs.getText())*. Merk op dat de klasse Float waaruit we de methode getText() halen met een hoofdletter geschreven is, dit in tegenstelling tot het gegevenstype float.

```
     prijs=Float.parseFloat(this.txtPrijs.getText());
     aantal=Short.parseShort(this.txtAantal.getText());
```

Instructies

Vervolgens berekenen we het totaal.

```
     totaal=prijs*aantal;
```

Uitvoer

Tot slot gaan we het totaal weergeven op het formulier. Hiervoor roepen we de methode setText aan van ons tekstvak txtTotaal. Dit geeft ons *this.txtTotaal.setText(totaal)*. Deze instructie zal echter niet tot het gewenste resultaat leiden. De methode setText verwacht een string en we hebben de float totaal als **parameter** ingegeven. We moeten ook hier een **typeconversie** doen.

```
public void startBerekening() {
     float prijs ;
     short aantal;
     float totaal;
     prijs=Float.parseFloat(this.txtPrijs.getText());
     aantal=Short.parseShort(this.txtAantal.getText());
     totaal=prijs*aantal;
     this.txtTotaal.setText(String.valueOf(totaal));
}
```

Druk op de “run main project”-knop: 

Herhaling

Wat moet ik kennen?

- ✔ Rapid Application Development (RAD).
- ✔ Het gegevensverwerkend proces en probleemoplossend denken.
- ✔ De begrippen:compileren, debuggen en interpreteren.
- ✔ De begrippen: Klassen, methoden en parameters.
- ✔ De gegevenstypen.
- ✔ De begrippen: Java Development Kit (JDK) en Java Runtime Environment (JRE).

Wat moet ik kunnen?

- ✔ Een consoleapplicatie maken.
- ✔ Een applet maken.
- ✔ Een Netbeans Java Desktop applicatie maken met verschillende besturingselementen.
- ✔ Een actie toevoegen aan het besturingselement knop.
- ✔ In- en uitvoer verwerken met typeconversie (parsing).

Test je kennis

Vraag 1: Wat is het verschil tussen de Java Development Kit (JDK) en Java Runtime Environment (JRE).

.....

.....

Vraag 2: Wat is een methode?

.....

.....

Vraag 3: Waarom is typeconversie (parsing) noodzakelijk?

.....

.....

Vraag 4: Wat is het verschil tussen een applet, desktop- en consoleapplicatie?

.....

.....

Oefeningen

Oefening 1: Maak onderstaande consoleapplicatie en beantwoord de vragen.

```
public class Main {
     public static void main(String[] args) {
          String gegeven;
          Persoon.setNaam("Bert");
          gegeven=Persoon.getNaam();
          System.out.println(gegeven);
     }
}

class Persoon {
     static String strNaam;
     public static void setNaam(String ingave) {
          strNaam = ingave;
     }

     public static String getNaam() {
          return strNaam;
     }
}
```

Noteer de lijn waarop je het gevraagde terug kan vinden:

- De declaratie van de variabele:.....
- De methode die geen waarde terug geeft:.....
- De methode die een parameter nodig heeft:.....
- De methode waar ons programma mee start:.....
- De methode die tekst terug geeft:.....

Oefening 2: Maak een programma die de waarde van twee tekstvakken kan optellen, aftrekken, vermenigvuldigen,...

De sequentie

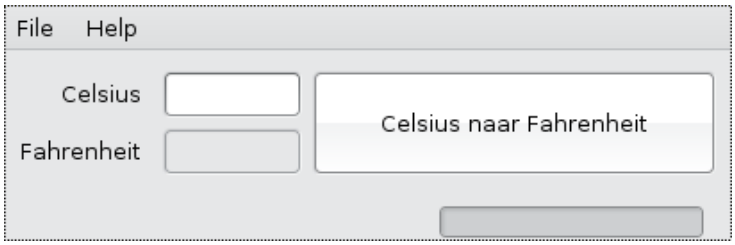
De **sequentie** is een opeenvolging van opdrachten. Deze worden stap per stap doorlopen. We willen de computer een aantal acties na elkaar laten uitvoeren zonder keuzes te maken of sprongen te nemen. We zullen ons probleem stapsgewijs analyseren.

Uiteindelijk zullen we een **algoritme** opstellen, dit is een opeenvolging bestaande uit verschillende stappen om ons doel te bereiken.

Oefening: Fahrenheit

Probleemstelling

Maak een programma die een aantal graden Celcius omzet naar Fahrenheit.



Palletobject	Property	Value
JLabel	Variable Name text	lblCelsius Celcius
JLabel	Variable Name text	lblFahrenheit Fahrenheit
JTextField	Variable Name text	txtCelsius
JTextField	Variable Name text enable	txtFahrenheit false
JButton	Variable Name text	btnRekenen Celcius naar Fahrenheit

Probleemanalyse	
Wat is gegeven?►	Fahrenheit = Celsius × 1.8 + 32
Wat moet nog ingevoerd worden?►	Een aantal graden Celsius.
Welke verwerking moet gebeuren?►	Omrekenen van Celsius naar Fahrenheit.
Wat met de uitvoer?►	Een aantal graden Fahrenheit.

Algoritme	
Invoer: celsius, "Graden Celsius?" fahrenheit=celsius*1.8+32 Uitvoer: fahrenheit, "Graden Fahrenheit = "	start celsius, fahrenheit: double INVOER: celsius fahrenheit=celsius*1.8+32 UITVOER: fahrenheit stop
Structuurdiagram	*Psuedocode*

Programma	
Operator	*Operator*
-	Negatief maken of aftrekken
*	Vermenigvuldigen
/	Deling
%	Modulo (rest bij deling)
+	Optellen

```
public void Bereken() {
     double celsius ;
     double fahrenheit;
     celsius=Float.parseFloat(this.txtCelsius.getText());
     fahrenheit=celsius*1.8+32;
     this.txtFahrenheit.setText(String.valueOf(fahrenheit));
}
```

Testen

Documenteren

Oefening: Het telraam

Probleemstelling

Maak een programma dat kan tellen. Er zijn twee knoppen en één label. Het label bevat steeds een zin met de huidige tellerwaarde.

Palletobject	Property	Value
JButton	Variable Name text	btnMeer Meer
JButton	Variable Name text	btnMinder Minder
JLabel	Variable Name text	lblTeller De teller staat op ...

Probleemanalyse	
Wat is gegeven?►	De teller start op 0
Wat moet nog ingevoerd worden?►	Een druk op de knop meer of minder

Welke verwerking moet gebeuren?►	Druk op knop meer = teller verhogen Druk op knop minder = teller verminderen
Wat met de uitvoer?►	De teller staat op ...

Algoritme

Structuurdiagram

```
teller: integer
teller=0
start Meer
        teller++
        UITVOER: “De teller staat op”, teller
stop
start Minder
        teller--
        UITVOER: “De teller staat op”, teller
stop
```

Psuedocode

Programma

Operator	*Operator*
+	Concatenatie (tekst samenvoegen)
++	Increment
--	decrement

```
int teller = 0;

@Action
public void Meer() {
     teller++;
     this.lblTeller.setText("De teller staat op
"+String.valueOf(teller));
}

@Action
     public void Minder() {
     teller--;
     this.lblTeller.setText("De teller staat op
"+String.valueOf(teller));
}
```

Java kan alle operatoren combineren met het isgelijkteken. We kunnen te teller verhogen door teller = teller + 1 maar ook verkort door teller+=1. Aangezien het hier een toename met 1 is kunnen we gebruik maken van de operator ++ en krijgen we teller++.

Testen

Documenteren

Herhaling

Wat moet ik kennen?

- ✔ De controlestructuur sequentie en zijn syntax..
- ✔ Operatoren gebruiken voor increment, decrement en concatenatie.

Wat moet ik kunnen?

- ✔ Een algoritme opstellen in NS-diagram of pseudocode.
- ✔ Tekst samenvoegen via het concatenatieteken.

Test je kennis

Vraag 1: Noteer in eigen woorden wat een algoritme is.

.....

.....

Vraag 2: Wat is concateneren?

.....

.....

Vraag 3: Wat gebeurt er wanneer we de opdracht “getal++” geven?

.....

.....

Oefeningen

Oefening 1: Maak een programma voor aan de kassa van de locale supermarkt. De kassierster geeft in wat de klant moet betalen en het ontvangen bedrag. Het programma berekent hoeveel geld we terug moeten krijgen.

Oefening 2: Maak een programma waarbij de enkele graden Celcius invoeren, na een druk op de knop weten we hoeveel graden Fahrenheit dit is. (Maak ook het omgekeerde mogelijk)

Oefening 3: Maak een programma voor een winkel. De gerant geeft de inkoopprijs in, en de winstmarge. Het programma geeft de verkoopprijs exclusief en inclusief 21% BTW.

De selectie

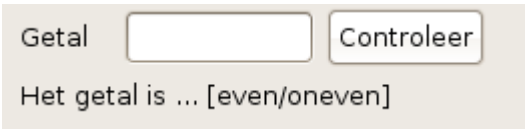
Wat is dit?

We spreken van een selectie, wanneer een actie pas wordt uitgevoerd als aan een bepaalde **voorwaarde** voldaan is. Een voorbeeld van een selectie in OpenOffice.org Calc is de functie ALS().

Oefening: Even of Oneven?

Probleemstelling

Geef een getal in en bepaal of dit getal even of oneven is.



Palletobject	Property	Value
JLabel	Variable Name text	lblGetal Getal
JTextField	Variable Name text	txtGetal
JButton	Variable Name text	BtnControleer Controleer
JLabel	Variable Name text	lblResultaat Het getal is ... [even/oneven]

Probleemanalyse

Wat is gegeven?►	Een even getal is deelbaar door 2.
Wat moet nog ingevoerd worden?►	Een getal.
Welke verwerking moet gebeuren?►	Bereken de rest na deling door 2. Is de rest 0 dan is het een even getal, zo niet dan is het een oneven getal. Modulus (mod) is een rekenkundige bewerking waarmee je op een eenvoudige en snelle manier de rest kan bekomen van een deling.
Wat met de uitvoer?►	Het antwoord: “even” of “oneven”

Algoritme


```
getal: integer
rest: double
antwoord: string
INVOER: getal
rest = getal % 2
IF rest==0 THEN
        antwoord=”even”
```

	ELSE
	antwoord=”oneven”
	ENDIF
	Uitvoer: antwoord

Structuurdiagram	*Psuedocode*

Hier worden twee verschillende acties ondernomen, we noemen dit een **tweevoudige selectie.** We spreken hier ook soms over de If…Then…Else of in het nederlands Als… Dan…Anders.

Programma

Operator	*Vergelijking*	*Operator*	*Vergelijking*
>	Groter dan	!	NOT
<	Kleiner dan	Voorbeeld: Als X==5, dan is X!=1 waar.	
= =	Gelijk aan	&&	AND
! =	Niet gelijk aan	Voorbeeld: Als X==5 dan is X>10 && X<100 vals.	
< =	Kleiner of gelijk aan	||	Short-circuit OR
> =	Groter of gelijk aan	Voorbeeld: Als X==5 dan is X>10 || X<100 waar.	
&	Logische AND		

```
public void Controleer() {
     int getal;
     getal = Integer.parseInt(this.txtGetal.getText());
     double rest;
     rest=getal%2;
     if (rest==0) {
          this.lblResultaat.setText("Even");
     }
     else {
          this.lblResultaat.setText("Oneven");
     }
}
```

Testen

Documenteren

Oefening: Kop of munt

Probleemstelling

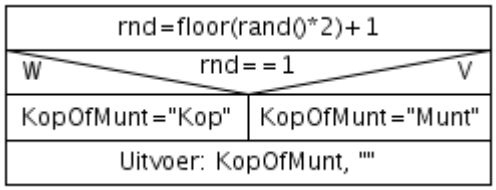
Maak een programma die kan tossen: Kop of Munt?

Palletobject	Property	Value
JTextField	Variable Name text	TxtTossen
JButton	Variable Name text	btnTossen Tossen!

Probleemanalyse

Wat is gegeven?►	Kop of munt?
Wat moet nog ingevoerd worden?►	-
Welke verwerking moet gebeuren?►	Willekeurig 1 (Kop) of 2 (Munt)...
Wat met de uitvoer?►	Kop of munt in tekstvak

Algoritme




```
rnd: double
rnd=Math.floor(Math.random()*2)+1
CASE rnd OF
        1 : KopOfMunt=”Kop”
        2 : KopOfMunt=”Munt”
ENDCASE
UITVOER: KopOfMunt
```

Psuedocode

In deze oefening hebben we een **willekeurig getal** nodig: 1 (Kop) of 2 (Munt). Hieronder lees je hoe we een willekeurig getal kunnen maken.

Math syntax	Bereik rnd
rnd=Math.random()	0,0...1 – 0,9...9
rnd=Math.random()*6	0,0...1 – 4,9...9
rnd=Math.floor(Math.random()*6)	0 – 4
rnd=Math.floor(Math.random()*6)+5	5 – 10

De algemene formule is dus:

rnd=Math.floor(Math.random()*HoogsteWaarde-LaatseWaarde+1)+LaagtseWaarde

In ons geval is dit:

rnd=Math.floor(Math.random()*2-1+1)+1

rnd=Math.floor(Math.random()*2)+1

Merk op dat de functie Math.random en Math.floor werken met rationele getallen. Java is zeer strict over het gegevenstype van variabelen. Indien we de bovenstaande code gebruiken moeten we de variabele rnd declareren als een double, omdat de Math-functie double terug geeft.

```
double rnd;
rnd=Math.floor(Math.random()*2)+1;
```

We slaan echter twee gehele getallen op in rnd. Willen we rnd toch als integer declareren dan moeten we het gegevenstype van de waarde die de Math-functie terug geeft aanpassen. Dit heet typecasting. Het **casten** gebeurt door het gegegevenstype tussen haken te plaatsen voor de expressie.

```
int rnd;
rnd=(int) Math.floor(Math.random()*2)+1;
```

In deze oefeningen hebben we opnieuw een keuze te maken. We hebben 1 (Kop) of 2 (Munt). We kunnen hiervoor een selectie opbouwen met de “if syntax” of gebruik maken voor de overzichtelijke “**select case** syntax”.

If select

```
if (rnd==1) {
        KopOfMunt="Kop";
} else if (rnd==2) {
        KopOfMunt="Munt";
} else {
        KopOfMunt="";
}
```

Case select

```
switch (rnd) {
        case 1:
                KopOfMunt = "Kop";
                break;
        case 2:
                KopOfMunt = "Munt";
                break;
        default:
                KopOfMunt="";
}
```

Programma

```
public void Tossen() {
     int rnd;
     String KopOfMunt;
     rnd=(int) Math.floor(Math.random()*2)+1;
     KopOfMunt="";
     switch (rnd) {
          case 1:
               KopOfMunt = "Kop";
               break;
     case 2:
               KopOfMunt = "Munt";
               break;
     default:
          KopOfMunt="";
}
this.txtTossen.setText(KopOfMunt);
}
```

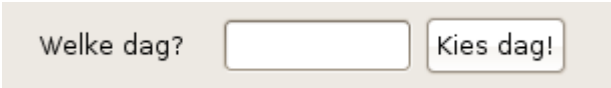
Testen

Documenteren

Oefening: Welke dag?

Probleemstelling

Je kan nooit beslissen op welke dag je huistaken gaat maken. Maak een programma die een willekeurige dag voor je kiest.

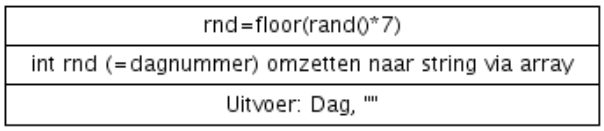


Palletobject	Property	Value
JLabel	Variable Name text	lblDag Welke dag?
JTextField	Variable Name text	txtDag
JButton	Variable Name text	btnDag Kies dag!

Probleemanalyse

Wat is gegeven?►	Een willekeurige dag
Wat moet nog ingevoerd worden?►	-
Welke verwerking moet gebeuren?►	Willekeurige dag kiezen (nummer 0 tot 6)
Wat met de uitvoer?►	De dag als tekst (maandag tot zondag)

Algoritme




```
rnd: int
rnd=Math.floor(Math.rand()*7)
arrDagen: array: String
Dag=arrDagen[rbd]
UITVOER: Dag
```

Structuurdiagram *Psuedocode*

We hebben 7 mogelijkheden, dus wensen we een willekeurig getal van 0 tot 6. We maken opnieuw gebruik van de functie: rnd=Math.floor(Math.random()*HoogsteWaarde-LaatseWaarde+1)+LaagtseWaarde. In ons geval: rnd=Math.floor(Math.random()*7)
Elk nummer staat voor een bepaalde dag. We moeten dit nummer kunnen omzetten naar een dag. De reeds gekende select case is bruikbaar maar er is een betere manier, namelijk werken met een **array**. Een array is een dimentionele variabele, een variabele met meerdere waarden. We voegen aan de variabele een index toe. Onder elke index kunnen we een nieuwe waarde (van hetzelfde gegevenstype) plaatsen.

```
switch (Dagen) {
        case 0:
                Dag = "maandag";
                break;
        case 1:
                KopOfMunt = "dinsdag";
                break;
        case 2:
                KopOfMunt = "dinsdag";
```

```
String[] arrDagen;
arrDagen = new String[7];
arrDagen[0] = "maandag";
arrDagen[1] = "dinsdag";
arrDagen[2] = "woensdag";
arrDagen[3] = "donderdag";
```

```
                break;
...
        case 6:
                KopOfMunt = "zondag";
                break;
}
```

select case

```
arrDagen[4] = "vrijdag";
arrDagen[5] = "zaterdag";
arrDagen[6] = "zondag";
```

array

Programma

```
public void WillekeurigeDag() {
      // We maken een array voor de dagen van de week.
      String[] arrDagen;
      arrDagen = new String[7];
      // We vullen de array met de dagen van de week.
      arrDagen[0] = "maandag";
      arrDagen[1] = "dinsdag";
      arrDagen[2] = "woensdag";
      arrDagen[3] = "donderdag";
      arrDagen[4] = "vrijdag";
      arrDagen[5] = "zaterdag";
      arrDagen[6] = "zondag";
      // Willekeurig getal tss 0 en 6 zoeken.;
      int rnd;
      //rnd=Integer.parseInt(willekeur);
      rnd=(int) Math.floor(Math.random()*7);
      // De waarde uit de array halen en terug geven aan de gebruiker.
      this.txtDag.setText(arrDagen[rnd]);
}
```

We merken dat we de code steeds structureren met tabs, dit maakt de code leesbaar. Om code gemakkelijker te begrijpen zie je hierboven **commentaar** tussen de code staan. Het documenteren en becommentariëren van code is zeer belangrijk. Er zijn twee manieren om code te noteren.

1. Code met één regel
   // Hierna komt de commentaar
2. Code met meerdere regels
   /* Hierna komt
   de meerlijnige code
   */

Deze syntax zorgt ervoor dat de regels genegeerd worden bij het uitvoeren van de code.

Testen

Documenteren

Herhaling

Wat moet ik kennen?

- ✔ De controlestructuur selectie en zijn syntax.
- ✔ De verschillende operatoren.
- ✔ Een select case en zijn syntax.
- ✔ Een array met zijn syntax.

Wat moet ik kunnen?

- ✔ De functie mod() gebruiken om een rest bij deling te berekeken.
- ✔ Een willekeurig getal genereren door de klasse Math te gebruiken.
- ✔ Commentaar toevoegen aan een programma.
- ✔ Het gegevenstype aanpassen via casting.

Test je kennis

Vraag 1: Wat is een array.

.....

Vraag 2: Waarom is het nodig te casten?

.....

Vraag 3: Wanneer maken we gebruik van een select case?

.....

Oefeningen

Oefening 1: Maak een programma die de vierkantswortel van een gegeven getal weergeeft.

Oefening 2: Maak een programma die weergeeft of een gegeven jaar een schrikkeljaar is.

Oefening 3: Maak een array aan met 10 willekeurige getallen. Laat deze willekeurige getallen zien aan de gebruiker.

Oefening 4: Maak een programma die de oppervlakte van een cirkel berekent.

Oefening 5: Maak een programma die controleert of een rekeningnummer geldig is.

Oefening 6: Maak een programma waarbij een persoon zijn grootte en zijn gewicht ingeeft. Het programma berekent je ideale gewicht en geeft commentaar (overgewicht, goed gewicht of ondergewicht).

De iteratie

Een **iteratie** is een **herhaling** in een programma. Er bestaan verschillende soorten herhalingen:

- **Begrensde herhaling**
  Een actie wordt een aantal keer herhaald. De begin en eindwaarde van de teller is op voorhand bepaald. Telkens de lus doorlopen wordt verhogen we de teller.

```
for (teller=startwaarde; teller<=eindwaarde; teller verhogen) {
        acties
}
```

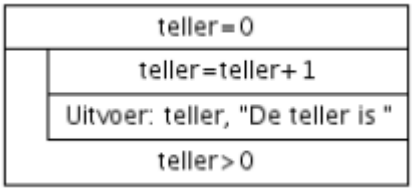
- **Voorwaardelijke herhaling met aanvangsvoorwaarde**
  Een voorwaarde wordt gecontroleerd alvorens de herhaling te starten. De herhaling blijft de acties uitvoeren tot de voorwaarde niet meer voldaan is.

```
while (voorwaarde) {
        acties
}
```

- **Voorwaardelijke herhaling met afbreekvoorwaarde**
  De acties worden steeds minstens één keer doorlopen. Op het einde van de acties wordt gecontroleerd of een voorwaarde voldaan is. Is de voorwaarde voldaan dan worden de acties opnieuw uitgevoerd. Is de voorwaarde niet voldaan dan wordt de lus niet uitgevoerd.

```
do {
opdrachtenblok
}
while (voorwaarde) ;
```

Wees steeds aandachtig wanneer je een herhaling gebruikt. Ooit al eens een programma gehad die niet meer reageerde? Misschien was dit programma in een **oneindige lus** terecht gekomen. De voorwaarde moet ooit voldaan zijn zodat de herhaling wordt gestopt.




```
int teller=0;
do {
        teller=teller+1
}
while (teller>0) ;
```

Structuurdiagram	*JAVA*

De teller verhogen of verlagen kan zoals hierboven via teller=teller+1. We kunnen deze ook in een verkorte notatie plaatsen.

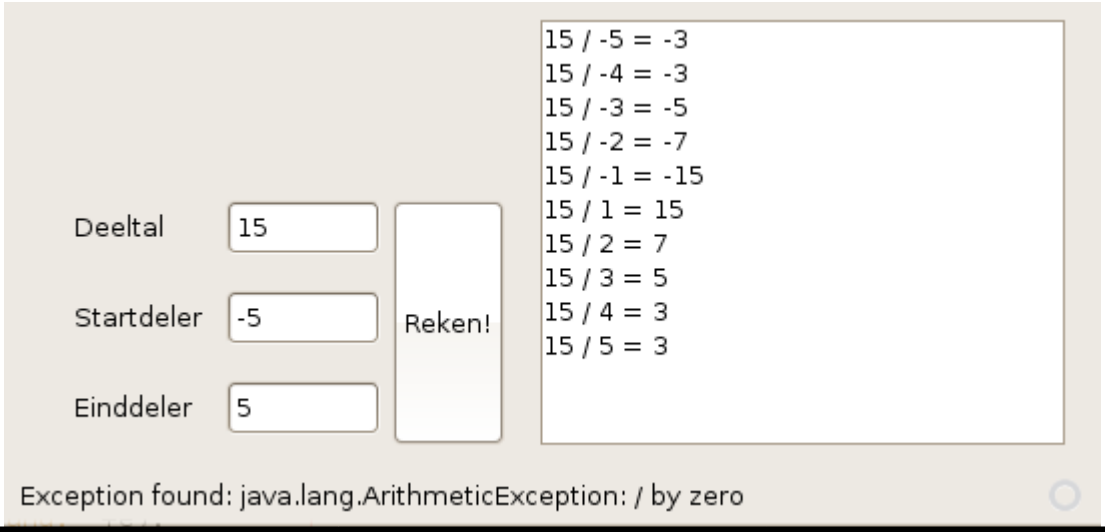
Voluit gescherven	*Verkorte notatie*
Teller = teller + 1	Teller = teller++
Teller = teller + 1	Teller++
Teller = teller + increment	Teller += increment

Deze verkortingen zijn ook met andere bewerkingen mogelijk.

Oefening: Quotiënten

Probleemstelling

Geef een overzicht van de deeltallen van deeltal X voor alle gehele delers vanaf een bepaalde startdeler tot einddeler.

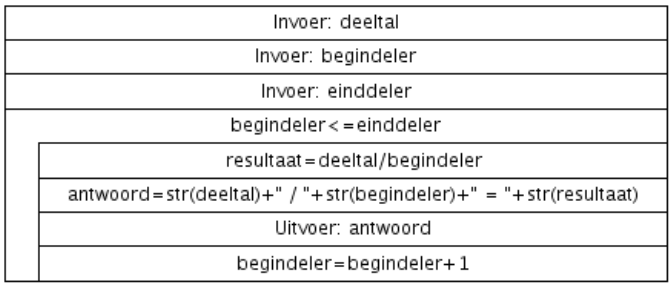


Palletobject	Property	Value
JLabel	Variable Name text	lblDeeltal Deeltal
JLabel	Variable Name text	lblStartDeler Startdeler
JLabel	Variable Name text	lblEindDeler Eindeler
JTextField	Variable Name text	txtDeeltal
JTextField	Variable Name text	txtBeginDeler
JTextField	Variable Name text	txtEindDeler
JButton	Variable Name text	btnReken Reken!
JTextArea	Variable Name text	txtResultaat

Probleemanalyse

Wat is gegeven?►	Een programma geeft delers weer
Wat moet nog ingevoerd worden?►	Deeltal, startdeler en einddeler
Welke verwerking moet gebeuren?►	Delen door (We gaan ook aandacht hebben voor het probleem: “delen door nul”.)
Wat met de uitvoer?►	Deeltal / Deler = Resultaat

Algoritme




```
INVOER: deeltal
INVOER: startdeler
INVOER: einddeler
FOR teller <= einddeler
        UITVOER:resultaat=deeltal/teller
ENDFOR
```

Structuurdiagram *Psuedocode*

Programma

In deze oefening is het mogelijk dat de gebruiker probeert te delen door nul. Dit kan voor problemen zorgen. Daarom gaan we de problemen of beter exceptions vangen. Hiervoor gebruiken we een try-catch.

```
Try {
        // Probeer dit uit te voeren... }
catch (Exception e) {
        // Vang uitzonderingen op in variabele e
        // Plaats de uitzondering in de statusbalk
        this.statusMessageLabel.setText("Exception found: "+e); }
```

Exception found: java.lang.ArithmeticException: / by zero

```
public void Delen() {
     int deeltal;
     int startdeler;
     int einddeler;
     int teller;
     int resultaat;
     deeltal=Integer.parseInt(this.txtDeeltal.getText());
     startdeler=Integer.parseInt(this.txtStartDeler.getText());
     einddeler=Integer.parseInt(this.txtEindDeler.getText());
     this.txtResultaat.setText("");
     for (teller=startdeler; teller<=einddeler;teller++) {
          try {
               resultaat=deeltal/teller;
               this.txtResultaat.setText(
               this.txtResultaat.getText()+String.valueOf(
               deeltal)+" / "+String.valueOf(teller)+" =
               "+String.valueOf(resultaat)+"\n");
          }
          catch (Exception e) {
               this.statusMessageLabel.setText("Exception found: "+e);
          }
     }
}
```

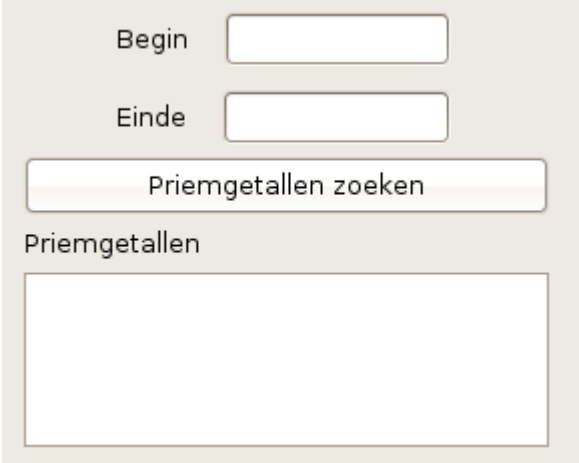
Testen

Documenteren

Oefening: Piemgetallen

Probleemstelling

Maak een programma dat priemgetallen zoekt.

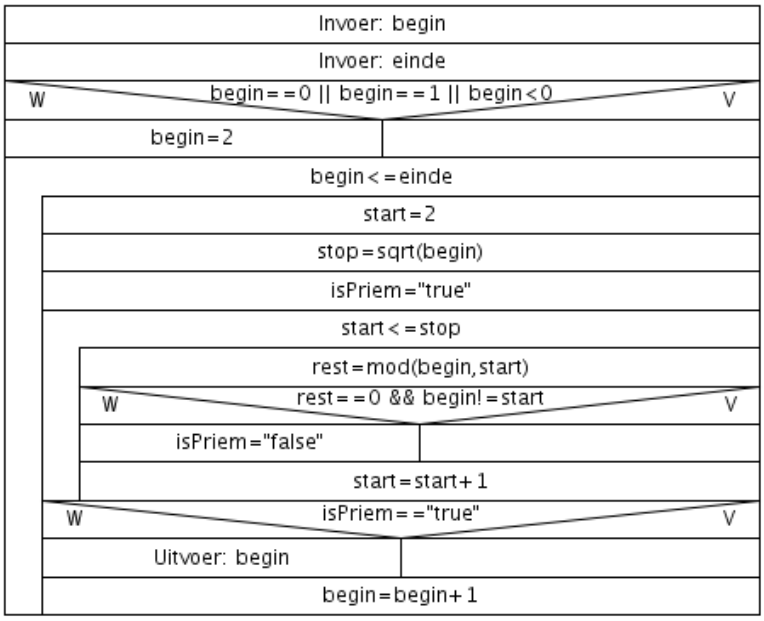


Palletobject	Property	Value
JLabel	Variable Name text	lblPriem Priemgetallen
JLabel	Variable Name text	lblBegin Begin
JLabel	Variable Name text	lblEinde Einde
JTextField	Variable Name text	txtBegin
JTextField	Variable Name text	txtEinde
JButton	Variable Name text	btnPriem Priemgetallen zoeken
JtextArea	Variable Name text	txtPriem

Probleemanalyse

Wat is gegeven?►	Priemgetallen zoeken...
Wat moet nog ingevoerd worden?►	Begin en eindpunt waartussen we de priemgetallen moeten vinden
Welke verwerking moet gebeuren?►	Is het priem?
Wat met de uitvoer?►	De priemgetallen onder elkaar weergeven.

Algoritme



Structuurdiagram

```
INVOER: begin
INVOER: einde
WHILE begin <= einde
        start = 2
        stop = sqrt(begin)
        isPriem = “true”
        WHILE start <= stop
        rest=begin%start
        IF rest==0 AND begin!=start
                isPriem=”false”
        ENDIF
        ENDWHILE
        IF isPriem==”true”
                UITVOER: begin
        ENDIF
ENDWHILE
```

Psuedocode

We voeren begin en eindpunt in. Aangezien negatieve getallen, nul en één niet priem zijn moeten we deze overslaan. Begin kan dus nooit lager zijn dan 2.
We maken een nieuwe methode of functieprocedure aan: public boolean isPriem(int x).Deze functie geeft true(1) of false(0) terug, vandaar het gegevenstype boolean. De functie isPriem neemt als argument een integer getal x.

We proberen in de functie het getal x te delen vanaf 2 tot de vierkantswortel van x. Indien de rest 0 is dan hebben we een deler gevonden. Deze deler mag alleen 1 of het getal zelf zijn. Het getal 1 controleren we niet maar het getal zelf wel. Indien we een getal vinden dat deelbaar is, moeten we enkel controleren of dit het getal zelf is. Indien we een deler vinden dat niet het getal zelf is dan is het getal geen priemgetal. Op het einde van de functie noteren we “return isPriem”, dit zorgt ervoor dat de waarde terug gegeven wordt.

Programma

```
public void priemgetallen() {
      // We maken het tekstveld leeg
      this.txtPriem.setText("");

      // Het begin en einde uitlezen

      int begin;

      int einde;

      begin=Integer.parseInt(this.txtBegin.getText());

      // De startwaarde kan nooit kleiner zijn dan
      // 2 (negatief, 0 en 1 zijn nooit priem)

            if ((begin==0) || (begin==1) || (begin<0)) {

                  begin=2;

            }
```

```
      einde=Integer.parseInt(this.txtEinde.getText());
      // Alle getallen van begin tot einde doorlopen
      // met stappen van 1 in begrensde herhaling
      int teller;
      for(teller=begin;teller<=einde;teller=teller+1) {
             if (isPriem(teller)==true) {
                  this.txtPriem.setText(this.txtPriem.getText()+
                  String.valueOf(teller)+"\n");
            }
      }
}


public boolean isPriem(int x) {
// Variabele aanmaken voor de terugkeerwaarde
boolean isPriem;
isPriem=true;
// We maken een teller aan, dit is tevens onze deler
// We delen de te testen waarde door alle getallen van 2
// We stoppen met testen bij de vierkanswortel
// van het te testen getal
int i;
int stop = (int) Math.sqrt(x);
// De teller doorlopen met een begrensde herhaling
for(i=2;i<=stop;i++) {
      // Alle delers testen, indien deelbaar
      // (en niet zichzelf) dan is het geen priemgetal
      // Indien deelbaar is de rest bij deling (modulus) 0
      double rest;
      rest = (x % i);
            // Indien rest 0 hebben we een deler gevonden
            // Het is geen priemgetal tenzij dat de deler
            // het getal zelf is
            if ((rest == 0) && (x != i)) {
                  isPriem=false;
            }
      }
return isPriem;
}
```

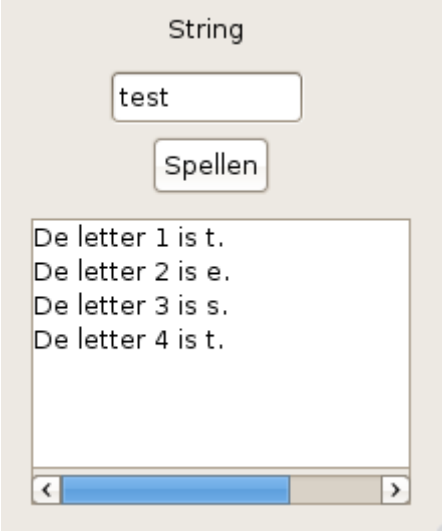
Testen

Documenteren

Oefening: Letter per letter

Probleemstelling

Maak een programma die een woord letter per letter weergeeft.



Palletobject	Property	Value
JLabel	Variable Name text	lblString String
JTextField	Variable Name text	txtString
JButton	Variable Name text	btnSpellen Spellen
JTextArea	Variable Name text	txtLetters

Probleemanalyse

Wat is gegeven?►	
Wat moet nog ingevoerd worden?►	Een woord
Welke verwerking moet gebeuren?►	Letter per letter weergeven (incl. nummering)
Wat met de uitvoer?►	De letters

Algoritme

Structuurdiagram

```
INVOER: tekst
Lengte = tekst.length()
nummer=0
DO
        letter=tekst.charAt(nummer)
        UITVOER: nummer en letter
        nummer++
WHILE huidigeLetter<Lengte
```

Psuedocode

Programma

Het gegevenstype String heeft een aantal methodes die we in deze oefening nodig hebben: charAt(int), length(),...

- tekst.length() telt het aantal tekens van een string terug. Bij tekst=”test” dan geeft tekst.length() 4 terug.
- tekst.charAt(3) geeft het karakter (char) terug op de derde plaats. Let wel op, we beginnen steeds te letten vanaf 0. Bij tekst=”test” dan heeft tekst.charAt(3) t terug.

We doorlopen letter per letter de string tot we de lengte bekomen hebben.

```
public void Spellen() {
     String tekst=this.txtString.getText();
     int huidigeLetter = 0;
     int lengteTekst=tekst.length();
     this.txtLetters.setText("");
     do {
          char letter;
          letter=tekst.charAt(huidigeLetter);
          this.txtLetters.setText(this.txtLetters.getText()
          +"De letter "+String.valueOf(huidigeLetter+1)+" is
          "+String.valueOf(letter)+".\n");
          huidigeLetter++;
     }
     while (huidigeLetter<lengteTekst);
}
```

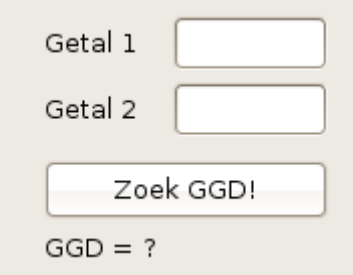
Testen

Documenteren

Oefening: Zoek de GGD...

Probleemstelling

Maak een programma die de grootst gemene deler weergeeft van twee getallen.



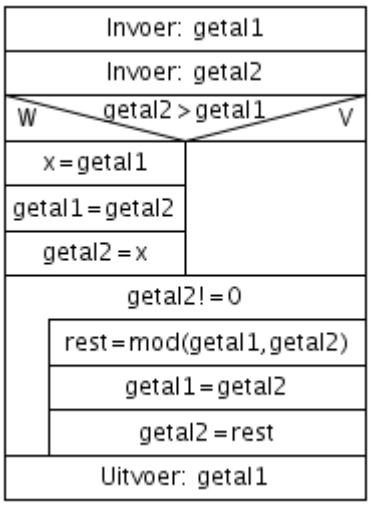
Palletobject	Property	Value
JLabel	Variable Name text	lblgetal1 Getal 1
JLabel	Variable Name text	lblGetal2 Getal 2
JLabel	Variable Name text	lblResultaat GGD = ?
JTextField	Variable Name text	txtGetal1
JTextField	Variable Name text	txtGetal2
JButton	Variable Name text	btnGGD Zoek GGD!

Probleemanalyse

Wat is gegeven?►	Grootst gemene deler
Wat moet nog ingevoerd worden?►	Twee gehele getallen
Welke verwerking moet gebeuren?►	Zoeken naar de GGD
Wat met de uitvoer?►	De GGD...

Algoritme

Eerst en vooral moeten we een algoritme zoeken voor het bepalen van de grootst gemene deler. Om de GGD te vinden gebruiken we de stelling van Euclides. Plaats steeds het grootste getal in de teller. Bepaal de rest bij deling. Indien de rest 0 is dan is de noemer de GGD. Indien we een rest hebben proberen we de noemer te delen door de gevonden rest. Opnieuw bepalen we de rest bij deling, indien 0 gevonden anders opnieuw de noemer delen door de gevonden rest.
Bijvoorbeeld: 48 / 18 geeft rest *12*, 18 / *12* geeft rest 6, 12 / 6 geeft rest 0 dus de GGD is 6



Structuurdiagram

```
INVOER: getal1
INVOER: getal2
IF getal2>getal1
        wissel geta 1 en getal2
ENDIF
WHILE getal2!=0
        rest=getal1%getal2
        getal1=getal2
        getal2=rest
ENDWHILE
UITVOER: getal1
```

Psuedocode

Programma

```
public void GGD() {
     int getal1;
     int getal2;
     getal1=Integer.parseInt(this.txtGetal1.getText());
     getal2=Integer.parseInt(this.txtGetal2.getText());
     int GGD;
     GGD=zoekGGD(getal1,getal2);
     this.lblResultaat.setText(String.valueOf(GGD));
}

int zoekGGD(int getal1,int getal2) {
     int rest;
     // Getal1 moet het grootste getal zijn (eventueel omwisselene)
     if (getal2>getal1) {
          int x;
          x=getal1;
          getal1=getal2;
          getal2=x;
     }
     // We delen de getallen tot zolang de rest niet 0 is
     while(getal2!=0){
          rest=getal1%getal2;
          getal1=getal2;
          getal2=rest;
     }
     return getal1;
}
```

Testen

Documenteren

Herhaling

Wat moet ik kennen?

- ✔ De controlestructuur iteratie en zijn syntax.
- ✔ De soorten herhalingen: begrensd, voorwaardelijk met aanvang- of afbraakvoorwaarde.
- ✔ De oneindige lus.

Wat moet ik kunnen?

- ✔ Uitzonderingen vangen met een try en catch.
- ✔ Een functieprocedures of methode maken met een return.

Test je kennis

Vraag 1: Hoe kan je een oneindige lus voorkomen?

.....

Vraag 2: Wat doet de actie “return var” in een functieprocedure of methode?

.....

Vraag 3: Hoe kan je voorkomen dat een programma blokkeert na delen door 0?

.....

Oefeningen

Oefening 1: Maak een programma die alle machten van een gegeven getal berekent tot een bepaald maximum.

Oefening 2: Maak een array met 100 willekeurige getallen en laat 10 waarden aan de gebruiker zien (0e, 10e, 20e, ...).

Oefening 3: Maak de rij van fibonacci aan 1 1 2 3 5 8 13 21 ...

$X_1 = 1$

$X_2 = 2$

$X_n = X_{n-2} + U_{n-1}$ voor $n > 2$

Oefening 4: Geef een getal in en geef al zijn delers weer.