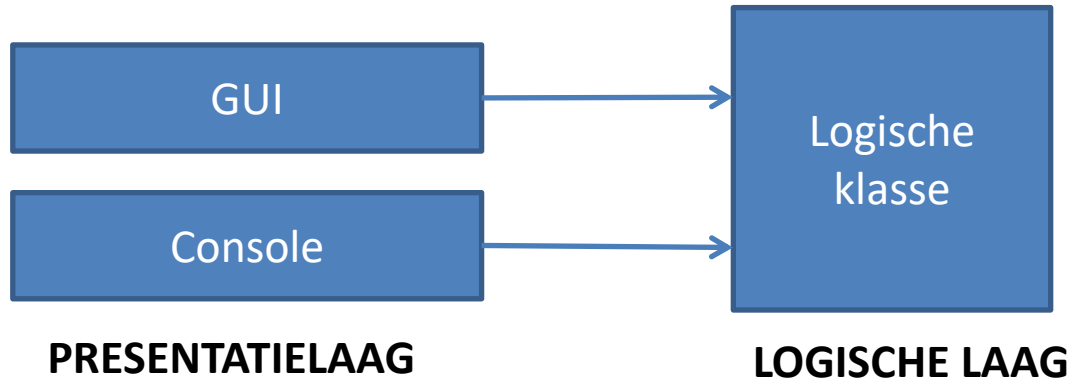


Les4

Grafische toepassingen in Netbeans IDE

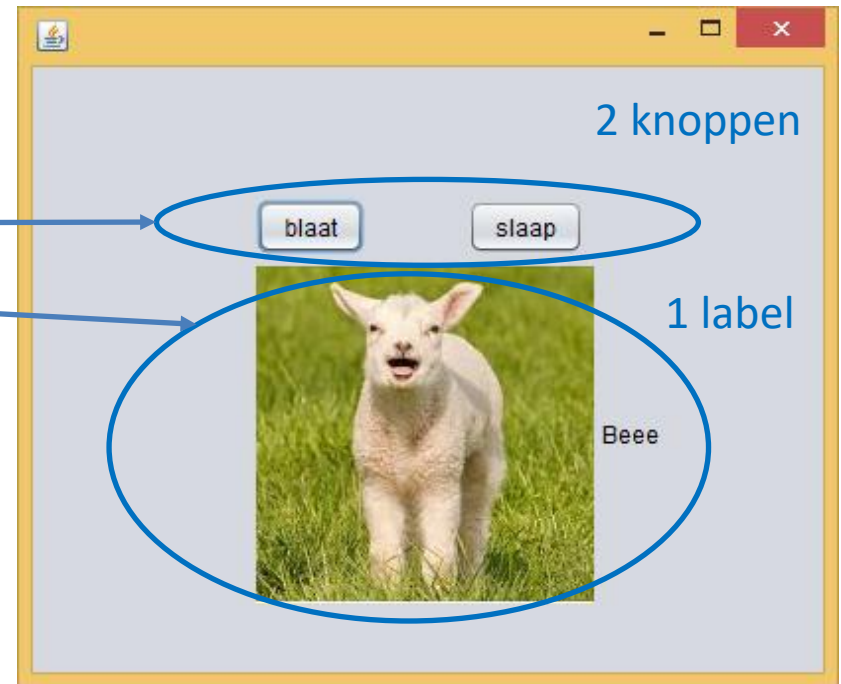
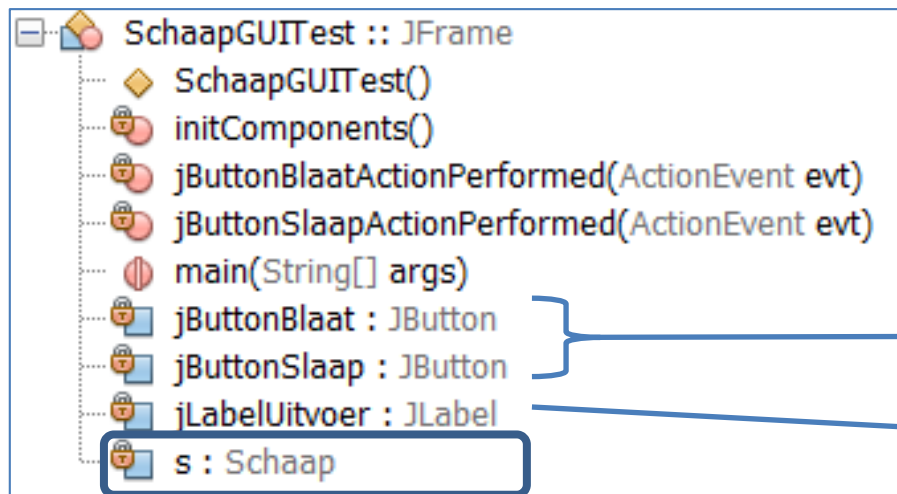
Samenspel van klassen



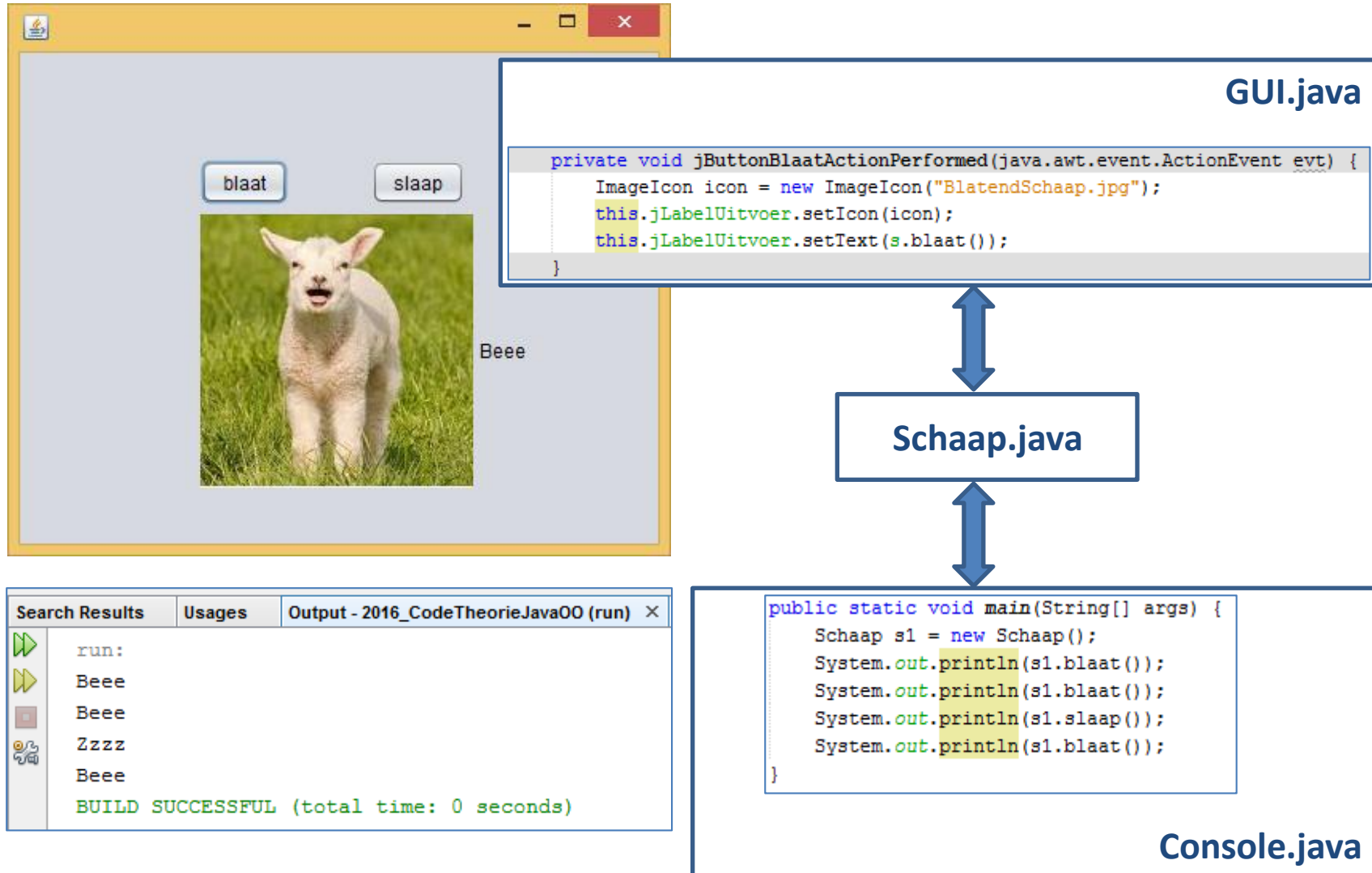
- Opsplitsen in afzonderlijke klassen
- Zoveel mogelijk code in logische klasse steken

=> Niet alleen de GUI, maar ook een console toepassing kan diezelfde logische klasse gebruiken

Opbouw van de grafisch venster



Presentatie versus Logica



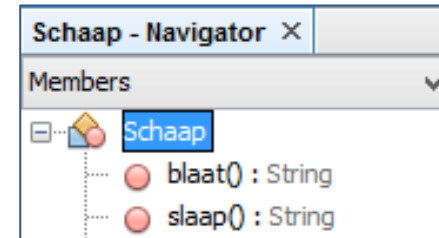
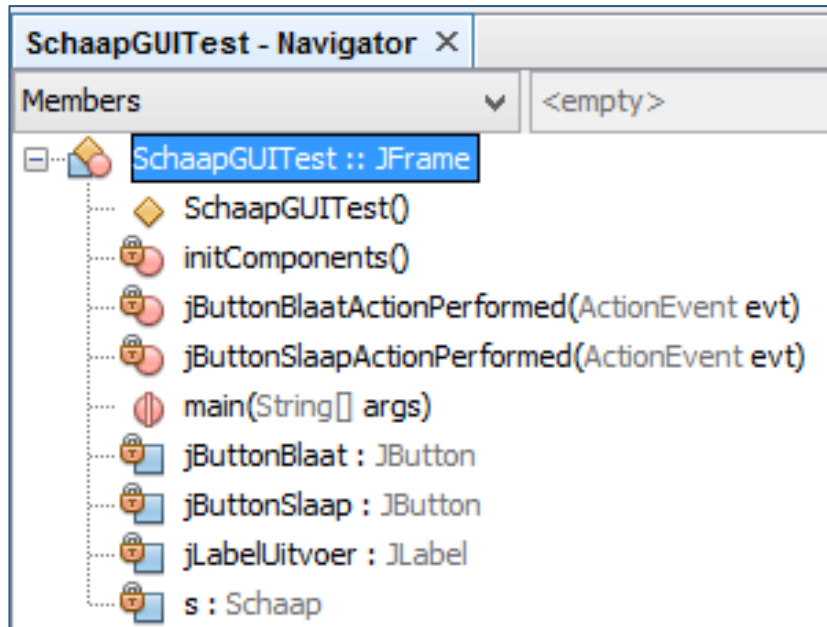
Niet printen in logische klasse!

Een grafische toepassing heeft geen consolevenster om naar te printen...

SchaapGUI.java

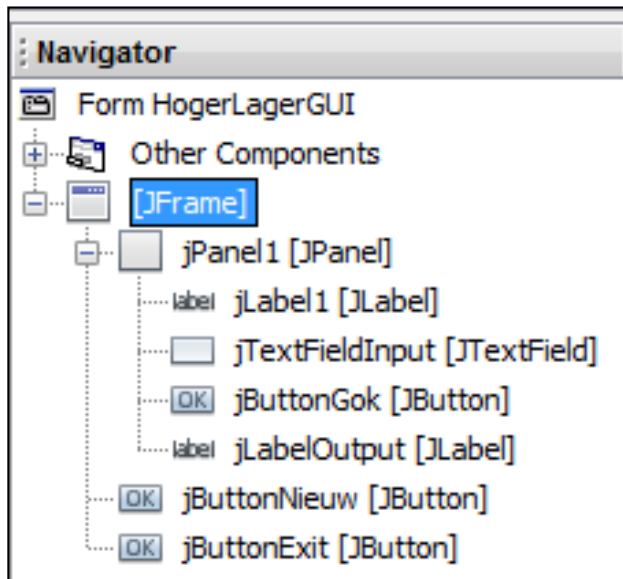


Schaap.java



```
public class Schaap {  
    public String slaap() {  
        return "Zzzz";  
    }  
  
    public String blaat() {  
        return "Beee";  
    }  
}
```

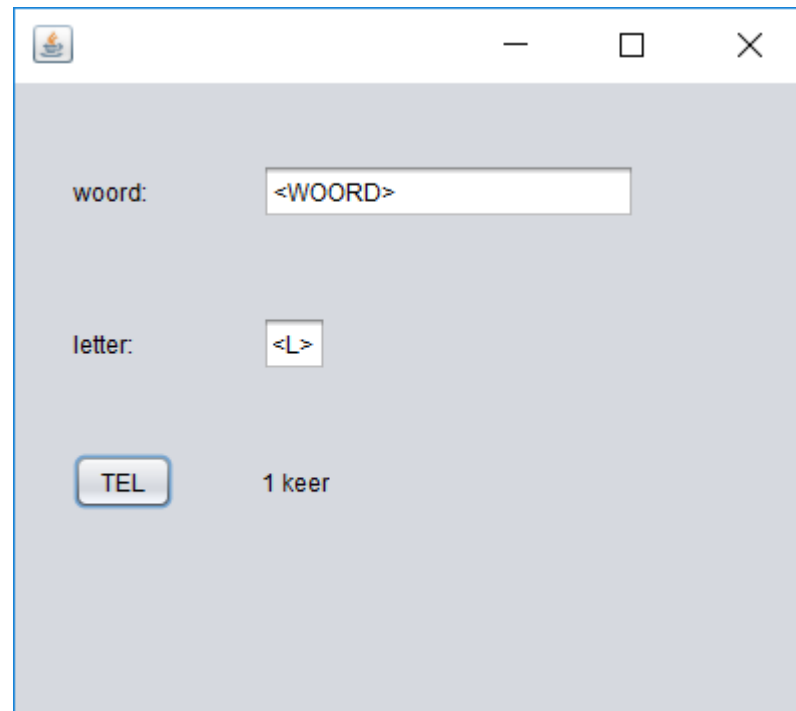
Opbouw van de grafische lay-out JFrame / JPanel



Weergave van info over 'logisch object'

- **Schaap** (icoontje en tekst als labelinfo)
- **Bewerkingen**: $12 * 8 = 96$ (dropdown voor operator ; numericUpDown voor getallen)
- **Punt** (afstand tussen 2 punten)
- **Cirkel – Vierkant** (oppervlakte en omtrek)
- **Breuk** (zie labo)

Intro: Zoek de fout

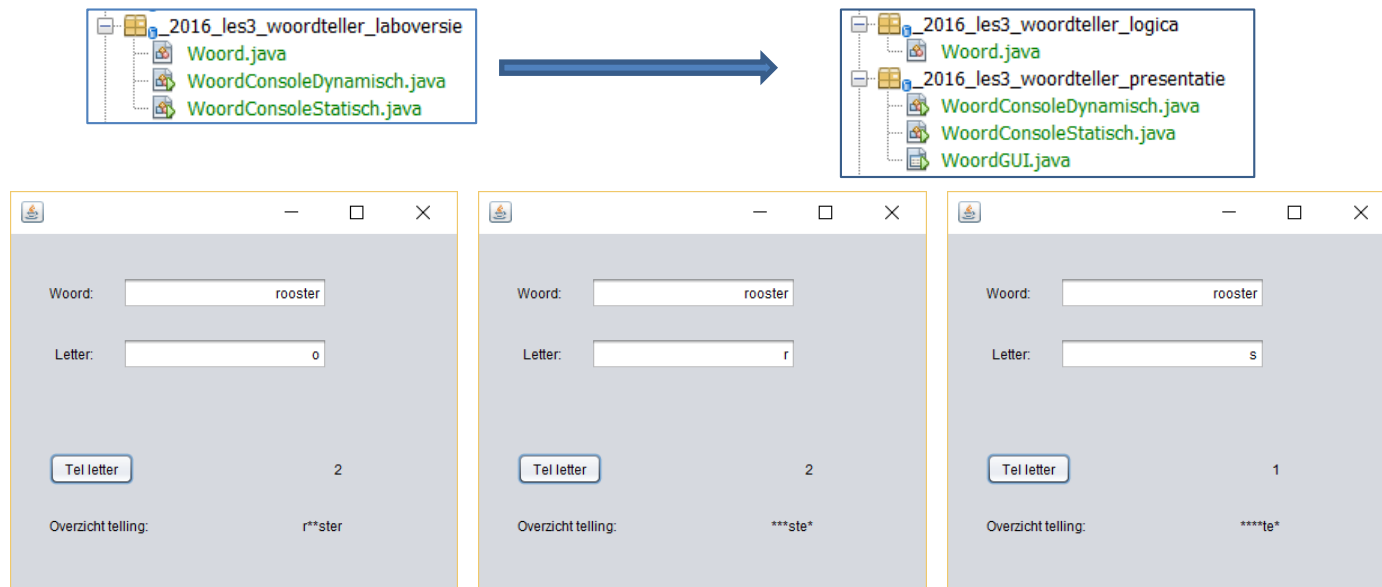


woord: <\WOORD>

letter: <L>

TEL 1 keer

Labo2-GUI variant (demo van scratch)



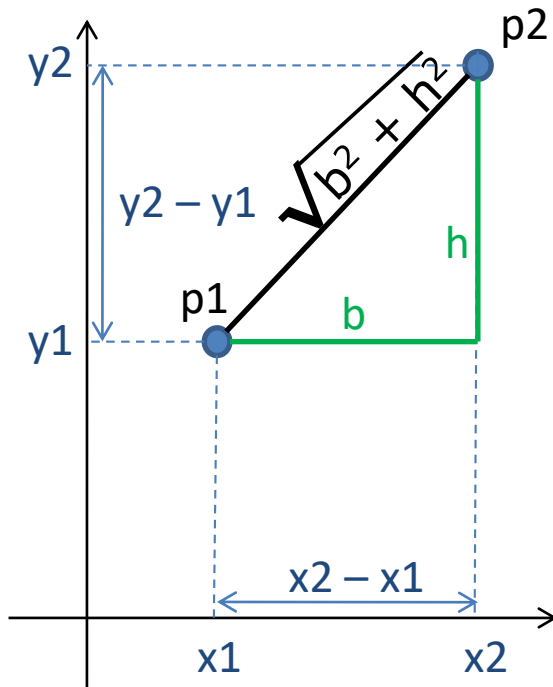
- GUI-componenten zijn ook objecten van klassen, nl. uit de Java Swing-bibliotheek!
- Naamgeving v/d componenten & refactoring
- Event handling & refactoring

Toepassing: Afstand tussen 2 punten

Zonder logische klasse

Met logische klasse Punt (statische approach)

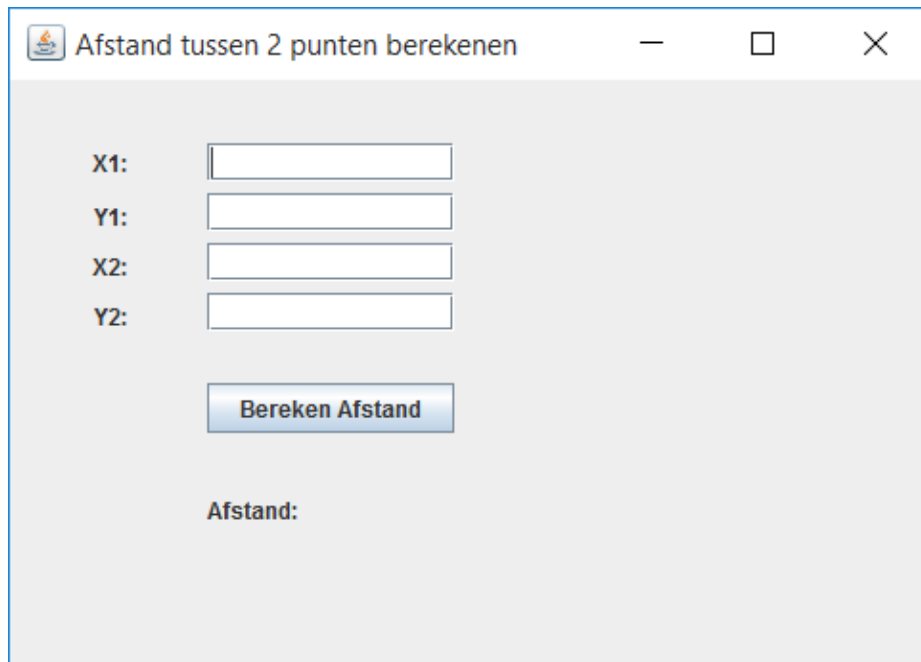
Met logische klasse Punt (OO-approach)



Stelling van Pythagoras

$$\begin{aligned} & \sqrt{b^2 + h^2} \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \end{aligned}$$

Grafische toepassing afstand tussen 2 punten



Afstand tussen 2 punten berekenen

X1:

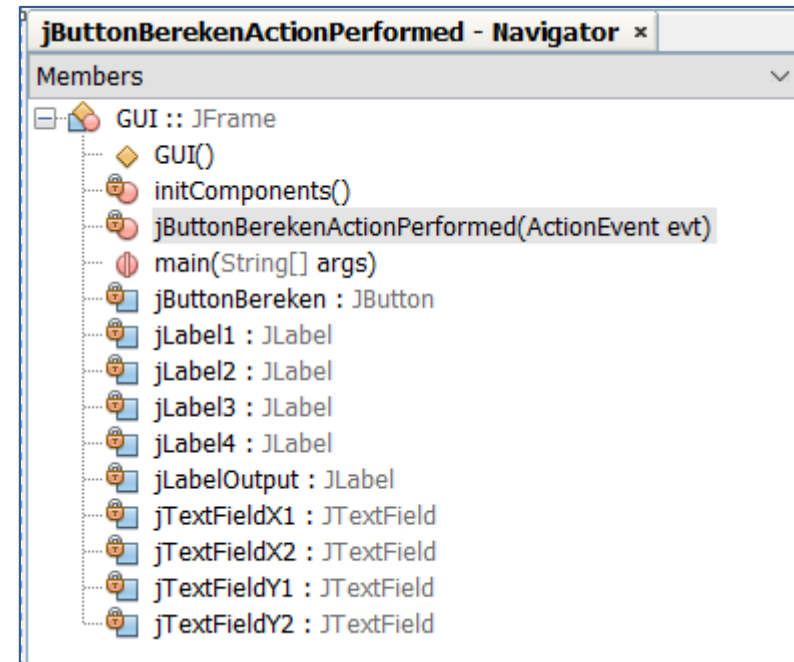
Y1:

X2:

Y2:

Bereken Afstand

Afstand:



$$\text{Formule} = \sqrt{(x2-x1)^2 + (y2-y1)^2}$$

Zonder logische klasse

$$= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
import java.text.DecimalFormat;
```

```
public class GUI {
```

```
...
```

```
private void jButtonBerekenActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    int x1 = Integer.parseInt(this.jTextFieldX1.getText());
```

```
    int y1 = Integer.parseInt(this.jTextFieldY1.getText());
```

```
    int x2 = Integer.parseInt(this.jTextFieldX2.getText());
```

```
    int y2 = Integer.parseInt(this.jTextFieldY2.getText());
```

```
    double afstand = Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
```

```
    DecimalFormat df = new DecimalFormat("0.00");
```

```
    this.jLabelOutput.setText("Afstand: " + df.format(afstand));
```

```
}
```

```
}
```

Logische klasse Punt (statische approach)

$$= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Punt

```
public static double berekenAfstand(int x1, int y1, int x2, int y2)
```

```
public class Punt {  
    public static double berekenAfstand(int x1, int y1, int x2, int y2) {  
        return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));  
    }  
}
```

Statische oproep vanuit GUI

```
package presentatie;
```

```
import java.text.DecimalFormat;
```

```
public class GUI {
```

```
    ...
```

```
    private void jButtonBerekenActionPerformed(java.awt.event.ActionEvent evt) {  
        int x1 = Integer.parseInt(this.jTextFieldX1.getText());  
        int y1 = Integer.parseInt(this.jTextFieldY1.getText());  
        int x2 = Integer.parseInt(this.jTextFieldX2.getText());  
        int y2 = Integer.parseInt(this.jTextFieldY2.getText());
```

```
        double afstand = Punt.berekenAfstand(x1, y1, x2, y2); //statische methode
```

```
        DecimalFormat df = new DecimalFormat("0.00");
```

```
        this.jLabelOutput.setText("Afstand: " + df.format(afstand));
```

```
    }
```

```
}
```

Statische oproep vanuit Console

```
package presentatie;
```

```
import logica.algemeen.Helper;
```

```
import logica.Punt;
```

```
public class Console {
```

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

```
        double res = Punt.berekenAfstand(1,2,3,4); //statische methode
```

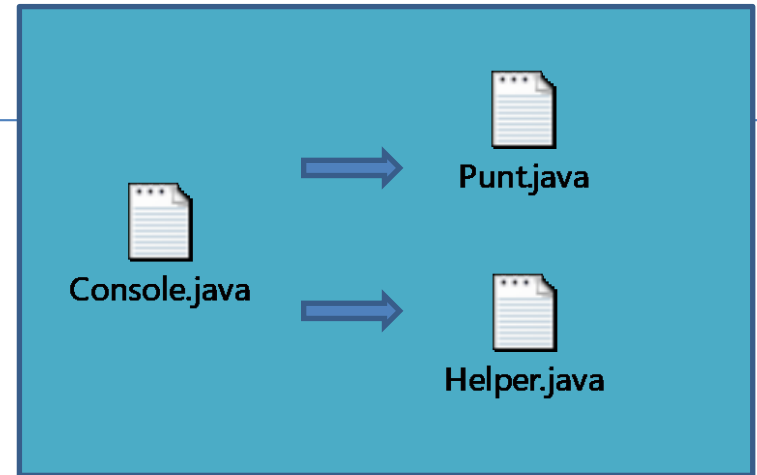
```
        System.out.println("afstand = " + res);
```

```
        System.out.println("afgerond = " + Helper.afronden(res, 2) );
```

```
        System.out.println("afgerond bis = " + Helper.afronden(res, "0.000") );
```

```
    }
```

```
}
```



Klasse Punt – OO-approach

```
...
public class GUI {
    ...
    private void jButtonBerekenActionPerformed(java.awt.event.ActionEvent evt) {
        int x1 = Integer.parseInt(this.jTextFieldX1.getText());
        int y1 = Integer.parseInt(this.jTextFieldY1.getText());
        int x2 = Integer.parseInt(this.jTextFieldX2.getText());
        int y2 = Integer.parseInt(this.jTextFieldY2.getText());

        Punt p1 = new Punt(x1,y1);
        Punt p2 = new Punt(x2,y2);

        double afstand = p1.berekenAfstand(p2); //object methode

        DecimalFormat df = new DecimalFormat("0.00");
        this.jLabelOutput.setText("Afstand: " + df.format(afstand));
    }
}
```

Analoog voor Console toepassing

```
package presentatie;
```

```
import logica.algemeen.Helper;  
import logica.Punt;
```

```
public class Console {  
    public static void main(String[] args) {  
        Punt p1 = new Punt(1,2);  
        Punt p2 = new Punt(3,4);
```

```
        double afstand = p1.berekenAfstand(p2); //object methode
```

```
        System.out.println("Afstand: " + afstand);
```

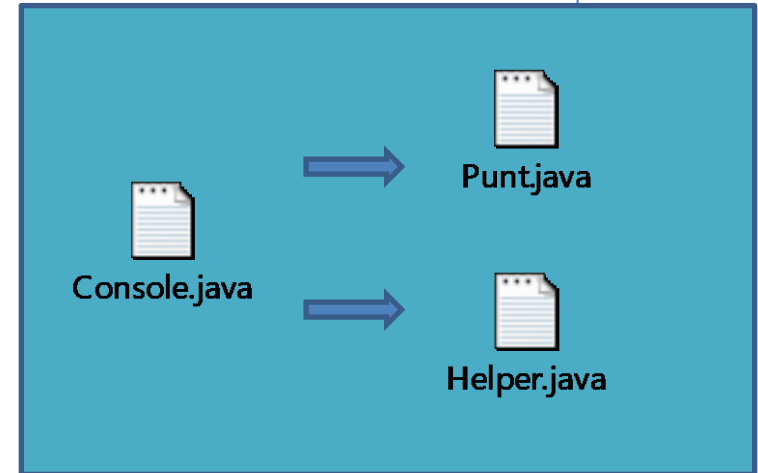
```
        //afronden via statische method uit Helper klasse
```

```
        System.out.println("afgerond = " + Helper.afronden(res, 2) );
```

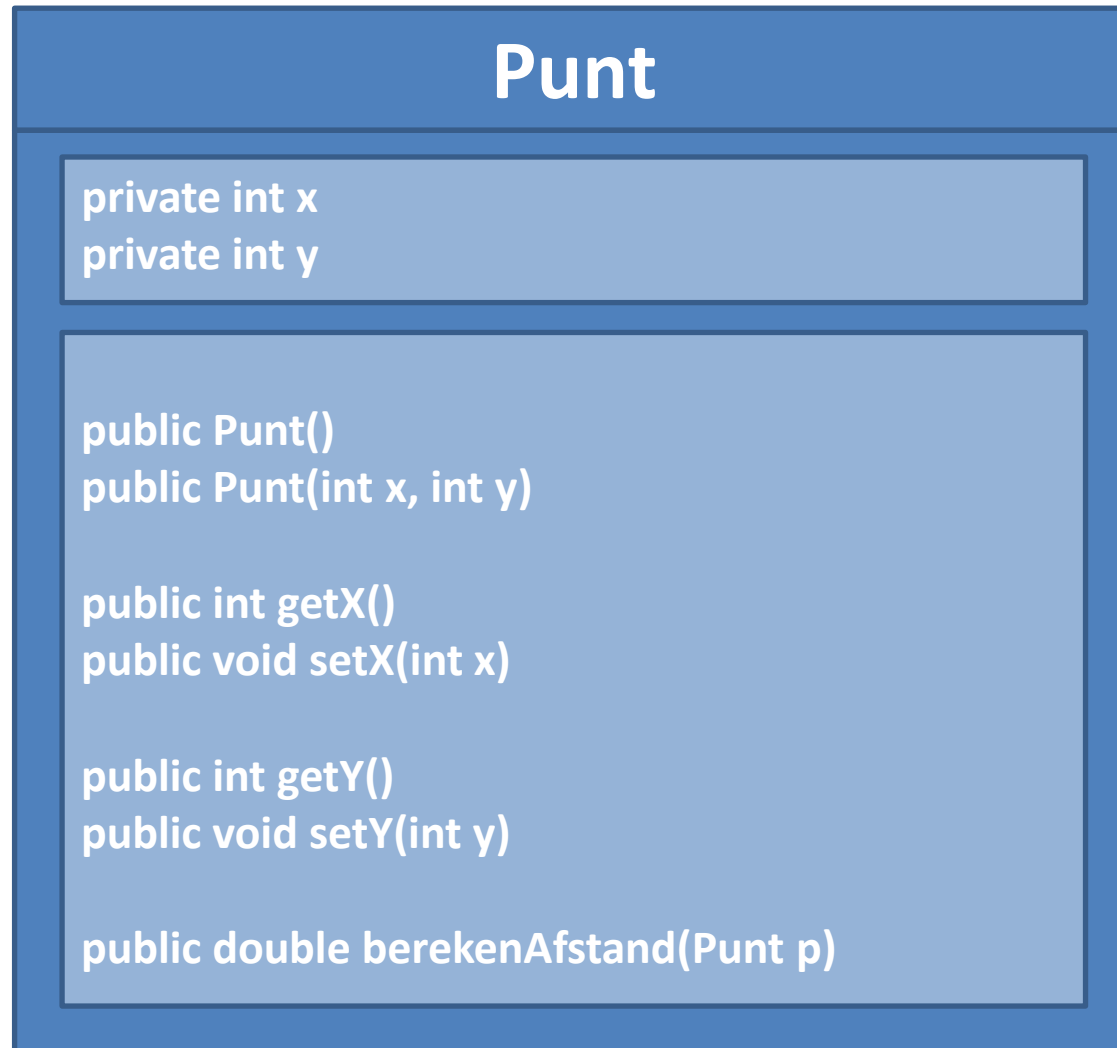
```
        System.out.println("afgerond bis = " + Helper.afronden(res, "0.000") );
```

```
    }
```

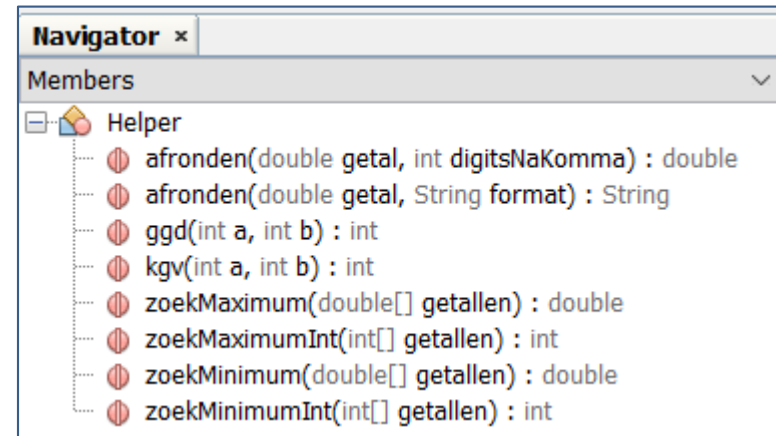
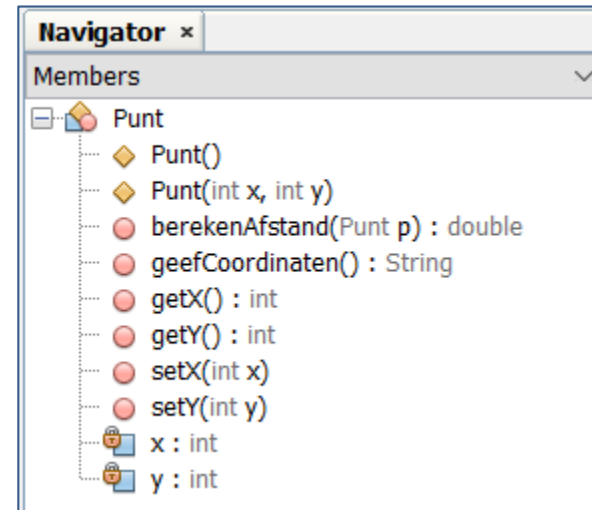
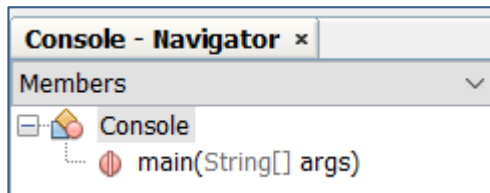
```
}
```



Klassendiagramma



Netbeans – Navigator view



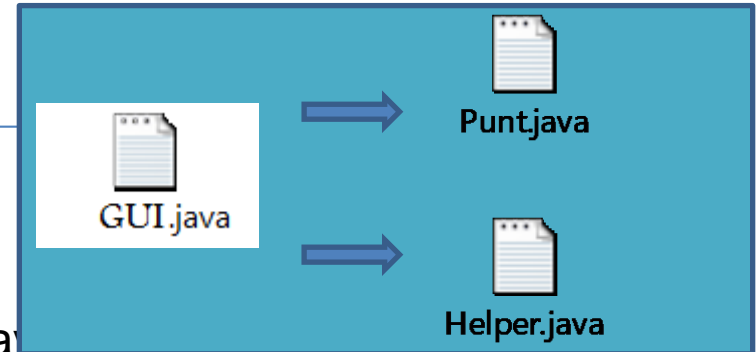
Samenspel van klassen

DEBUGGING SESSIE

```
...
public class GUI {
    ...
    private void jButtonBerekenActionPerformed(java
        int x1 = Integer.parseInt(this.jTextFieldX1.getText());
        int y1 = Integer.parseInt(this.jTextFieldY1.getText());
        int x2 = Integer.parseInt(this.jTextFieldX2.getText());
        int y2 = Integer.parseInt(this.jTextFieldY2.getText());

        Punt p1 = new Punt(x1, y1);
        Punt p2 = new Punt(x2, y2);

        double afstand = p1.berekenAfstand(p2);
        this.jLabelOutput.setText("Afstand: " + Helper.afronden(afstand, "0.00") );
    }
}
```



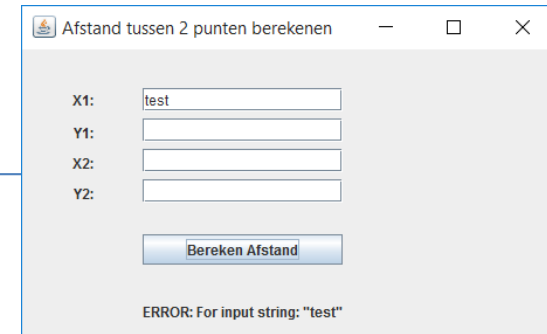
Grafische variant met input controle

```
...
public class GUI {
    ...
    private void jButtonBerekenActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            int x1 = Integer.parseInt(this.jTextFieldX1.getText());
            int y1 = Integer.parseInt(this.jTextFieldY1.getText());
            int x2 = Integer.parseInt(this.jTextFieldX2.getText());
            int y2 = Integer.parseInt(this.jTextFieldY2.getText());

            double afstand = new Punt(x1, y1).berekenAfstand(new Punt(x2, y2));

            DecimalFormat df = new DecimalFormat("0.00");
            this.jLabelOutput.setText("Afstand: " + df.format(afstand));

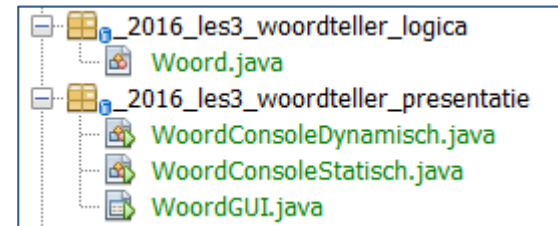
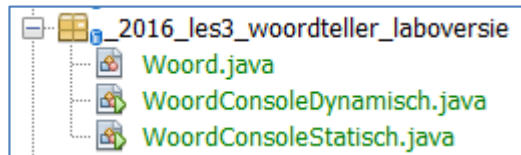
        } catch (Exception e) {
            this.jLabelOutput.setText("ERROR: " + e.getMessage());
        }
    }
}
```



Nabeschouwing

- Variabele van type Punt (logische klasse) is hier lokaal gedeclareerd in de methode waar de logische afhandeling gebeurt. **Elke klik op de knop is een nieuwe logische vraag die beantwoord moet worden**
- Dit is niet altijd aangewezen: Als **meerdere klikken eenzelfde logisch gegeven beïnvloeden**, dan zal het logische object als `veld gedefinieerd` moeten worden in de presentatieklasse . Zie volgend toepassingsvoorbeeld.

Labo2-GUI variant



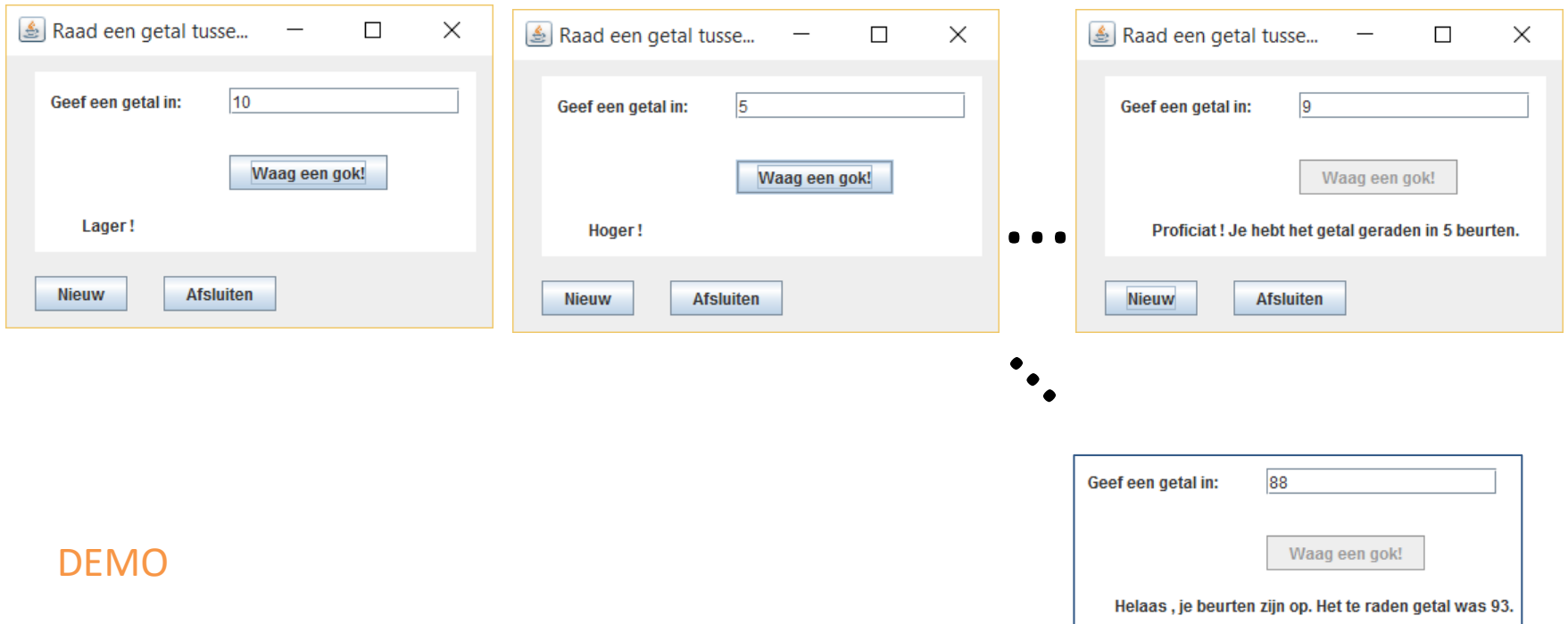
Initial state of the GUI application window. The 'Woord' field contains 'rooster' and the 'Letter' field contains 'o'. The 'Tel letter' button is highlighted, and the count is 2. The 'Overzicht telling:' field shows 'r**ster'.

State of the GUI application window after clicking 'Tel letter' with 'o'. The 'Letter' field now contains 'r', the count is 2, and the 'Overzicht telling:' field shows '***ste*'.

State of the GUI application window after clicking 'Tel letter' with 'r'. The 'Letter' field now contains 's', the count is 1, and the 'Overzicht telling:' field shows '*****e*'.

- GUI-componenten zijn ook objecten van klassen, nl. uit de Java Swing-bibliotheek!
- Naamgeving v/d componenten & refactoring
- Event handling & refactoring

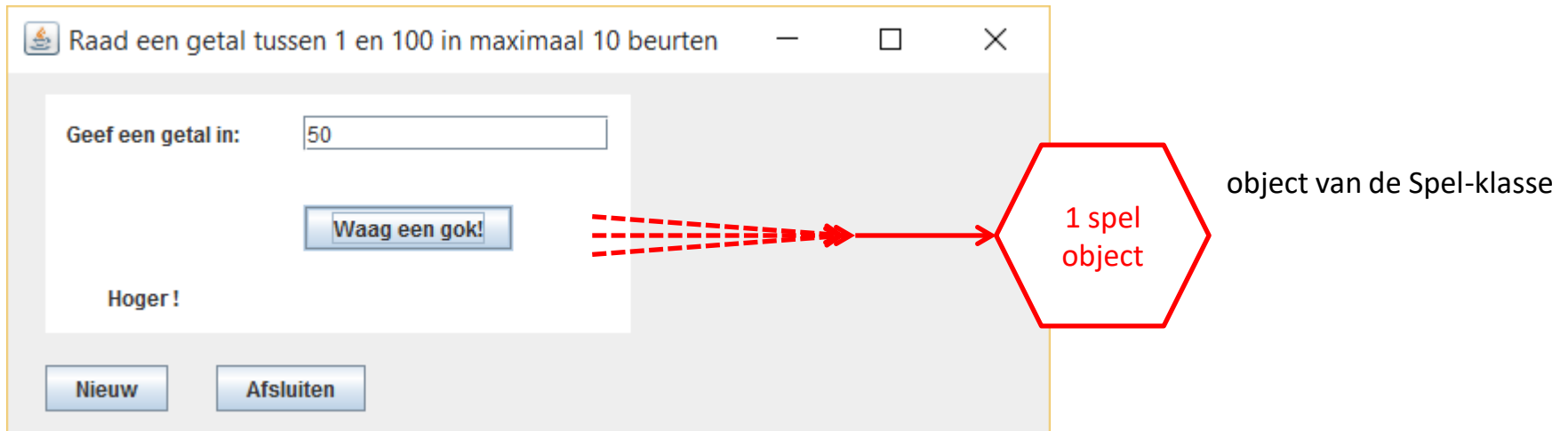
Grafisch Hoger-Lager spel

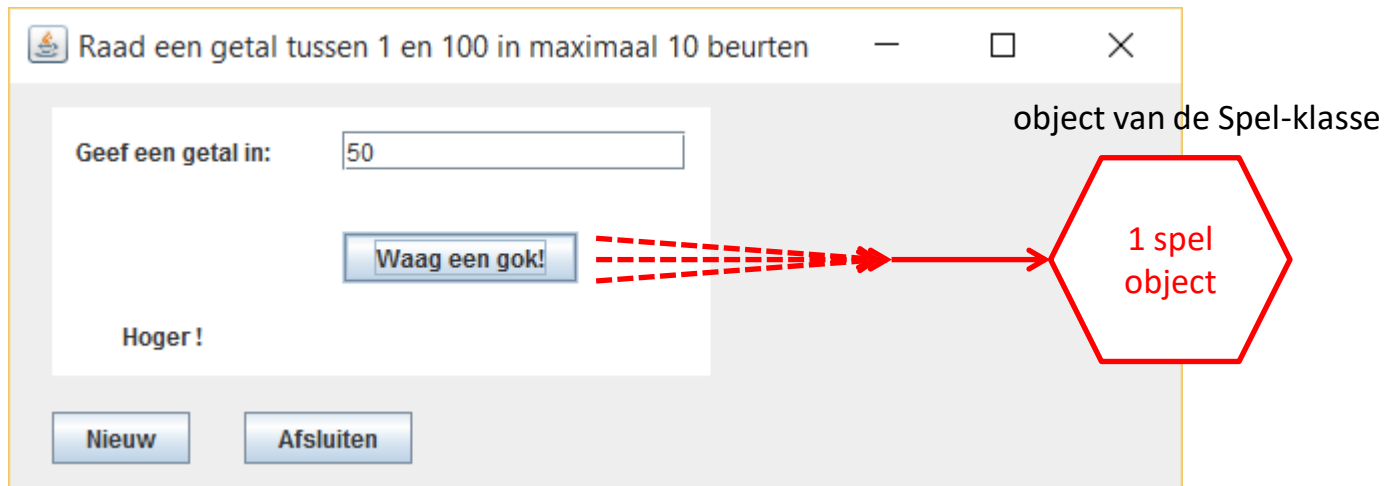


DEMO

GUI ontwerp

Als meerdere klikken eenzelfde logisch gegeven beïnvloeden, dan zal het logische object als veld gedefinieerd moeten worden in de presentatieklasse .





Bij het **opstarten van de toepassing** wordt een Spel-object gecreëerd (i.e. gedeclareerd en geïnitieerd). Initialisatie van een Spel-object omvat het genereren van een nieuw te raden getal binnen bereik [MIN,MAX]. Bij initialisatie wordt ook bepaald hoeveel beurten de speler krijgt om het spel uit te spelen.

Doorheen het drukken op de knop **‘Waag een gok!’** verandert de ‘toestand’ van hét spel dat je aan het spelen bent: Het aantal reeds gespeelde beurten van het spel vermeerderd telkens met één.

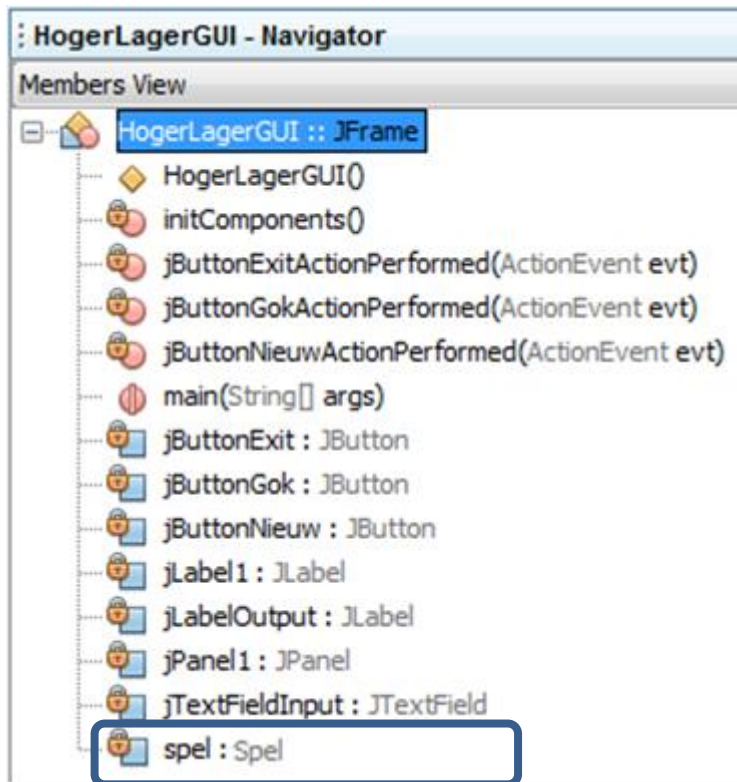
Druk je op de knop **‘Nieuw’**, dan zal je een nieuw spel starten. Dit kan je doen

- door een nieuw Spel-object te creëren en te initialiseren
- of door het bestaande spel-object zijn toestand te resetten

Met de knop **‘Afsluiten’** wordt de toepassing beëindigd

Het centrale Spel-object

Waar en hoe wordt het Spel-object gedeclareerd en geïnitialiseerd?



- **spel** is een variabele van de logische klasse **Spel**
- Dit **spel**-object is gedefinieerd als een dataveld in de presentatieklasse **HoyerLagerGUI**
- De toegang tot het **spel**-object is **private**
- 'spel' is een variabele van het **referentietype**

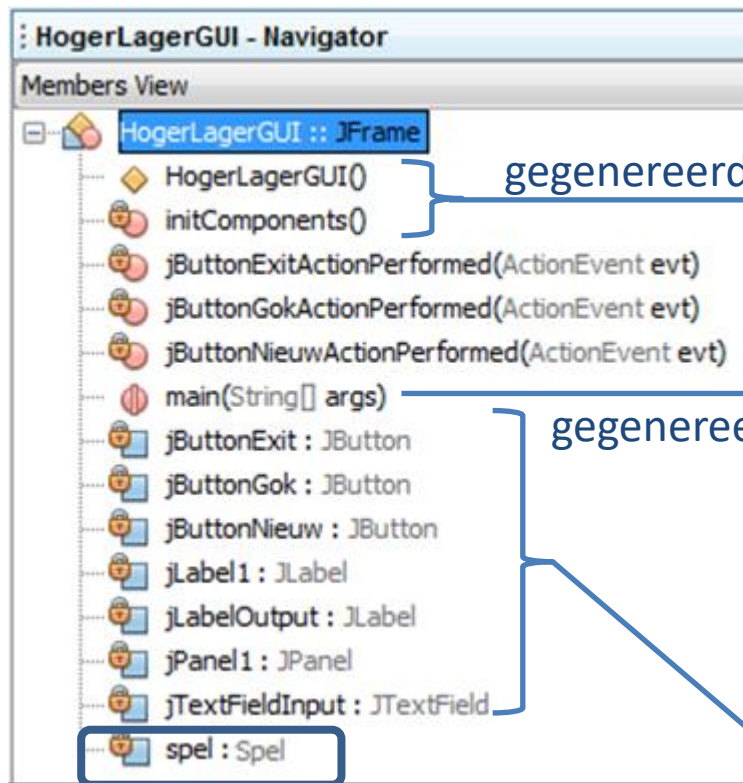
```
public class HoyerLagerGUI extends javax.swing.JFrame {  
    private Spel spel;  
}
```

- ➔ De variabele '**spel**' moet naar een effectief object van de logische klasse **Spel** refereren
- ➔ Via dit concrete '**spel**'-object zal je bijgevolg de logica kunnen afhandelen

Initialisatie v/h logisch object in de constructor

```
public class HogerLagerGUI extends javax.swing.JFrame {  
    private static final int MIN = 1;  
    private static final int MAX = 100;  
    private static final int AANTAL_BEURTEN = 10;  
  
    private Spel spel;  
  
    public HogerLagerGUI() {  
        initComponents();  
        this.setTitle("Raad een getal tussen " + MIN + " en " + MAX  
            + " in maximaal " + AANTAL_BEURTEN + " beurten");  
  
        //maak het Spel object aan  
        spel = new Spel(MIN, MAX, AANTAL_BEURTEN);  
    }  
}
```

Algemene opbouw/werking van de GUI



```
/** Creates new form HoyerLager */
public HoyerLagerGUI() {
    initComponents();
}

/**...*/
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() { ... } // </editor-fold>
```

```
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {
            new HoyerLagerGUI().setVisible(true);
        }

    });
}
```

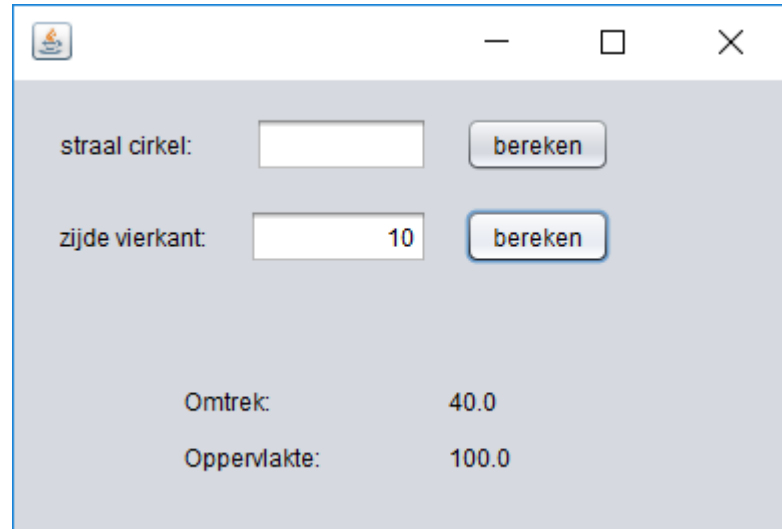
```
// Variables declaration - do not modify
private javax.swing.JButton jButtonExit;
private javax.swing.JButton jButtonGok;
private javax.swing.JButton jButtonNieuw;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabelOutput;
private javax.swing.JPanel jPanel1;
private javax.swing.JTextField jTextFieldInput;
// End of variables declaration
```

Toepassing: Cirkel - Vierkant



A Java Swing window titled "Toepassing: Cirkel - Vierkant" with a standard Mac OS X title bar. The window has a light gray background. It contains two input fields: "straal cirkel:" with the value "5" and "zijde vierkant:" which is empty. Each input field has a "bereken" button next to it. At the bottom, the calculated values are displayed: "Omtrek: 31.42" and "Oppervlakte: 78.54".

straal cirkel:	<input type="text" value="5"/>	<input type="button" value="bereken"/>
zijde vierkant:	<input type="text"/>	<input type="button" value="bereken"/>
Omtrek:	31.42	
Oppervlakte:	78.54	



A Java Swing window titled "Toepassing: Cirkel - Vierkant" with a standard Mac OS X title bar. The window has a light gray background. It contains two input fields: "straal cirkel:" which is empty and "zijde vierkant:" with the value "10". Each input field has a "bereken" button next to it. At the bottom, the calculated values are displayed: "Omtrek: 40.0" and "Oppervlakte: 100.0".

straal cirkel:	<input type="text"/>	<input type="button" value="bereken"/>
zijde vierkant:	<input type="text" value="10"/>	<input type="button" value="bereken"/>
Omtrek:	40.0	
Oppervlakte:	100.0	

Opdracht:

Ontwerp de relevante 'logische' klassen om bovenstaande functionaliteit mogelijk te maken. Geef klassendiagramma.

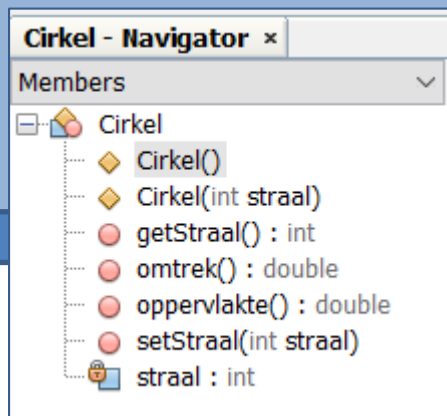
Cirkel

```
private int straal
```

```
public Cirkel()  
public Cirkel(int straal)
```

```
public int getStraal()  
public void setStraal(int straal)
```

```
public double oppervlakte()  
public double omtrek()
```



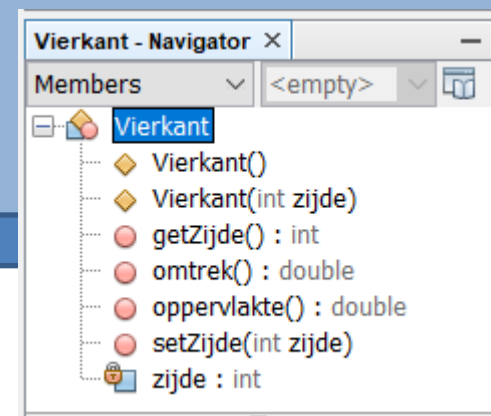
Vierkant

```
private int zijde
```

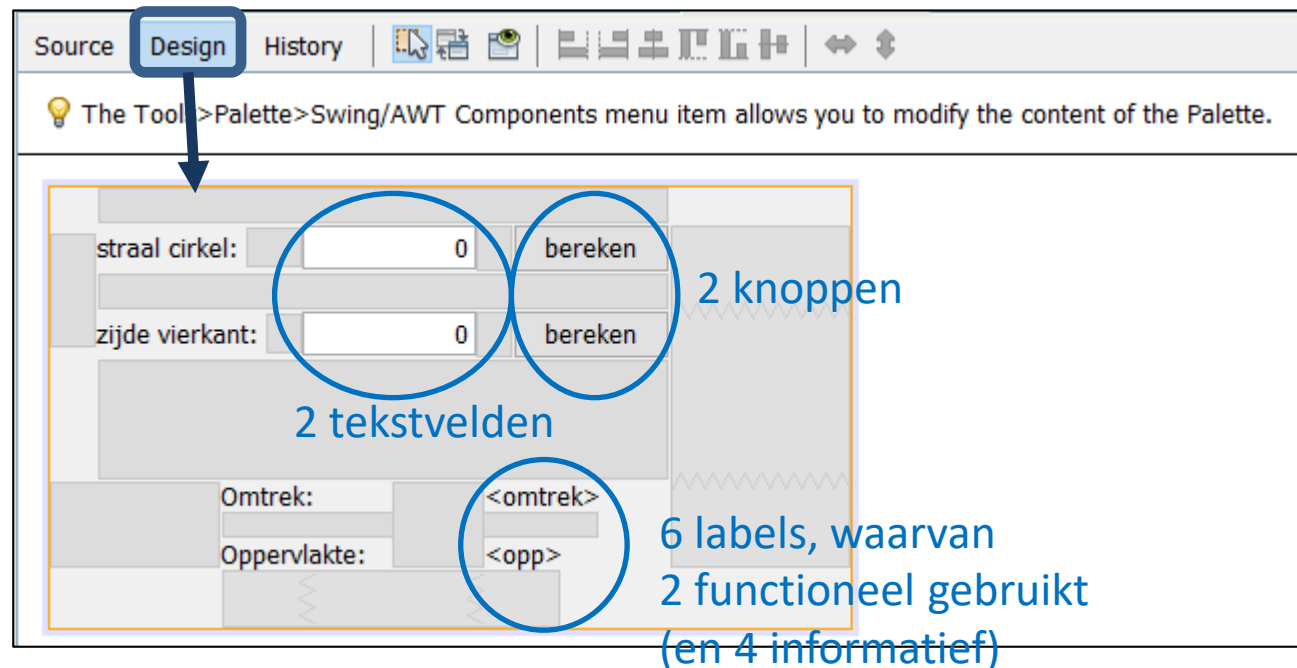
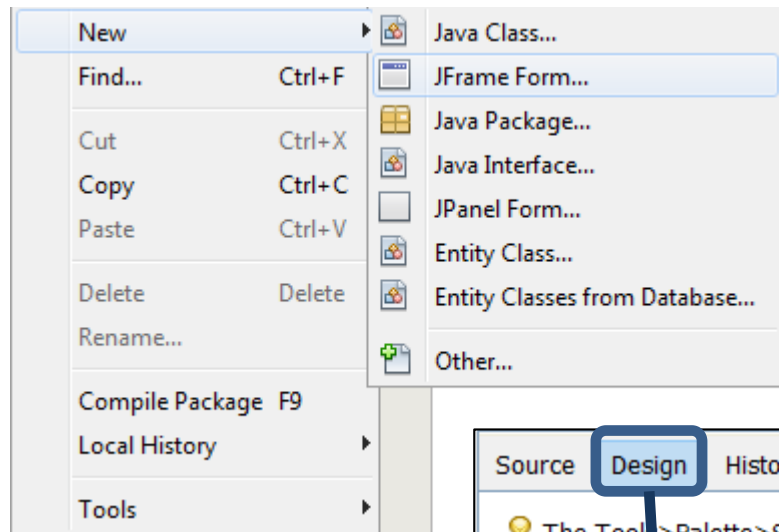
```
public Vierkant()  
public Vierkant(int zijde)
```

```
public int getZijde()  
public void setZijde(int zijde)
```

```
public double oppervlakte()  
public double omtrek()
```

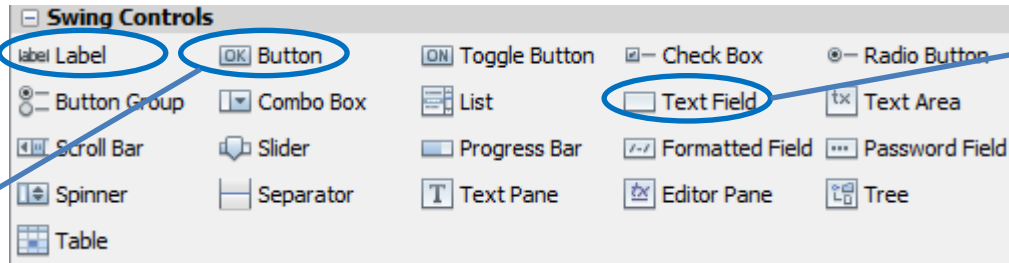


Grafische toepassing maken met de Netbeans IDE



Grafische componenten & Standaard event handling

jLabelWaardeOmtrek
jLabelWaardeOpp



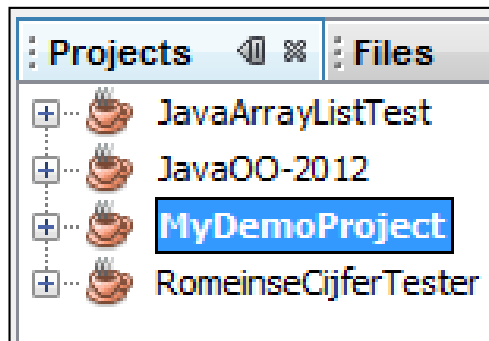
jTextFieldStraal
jTextFieldZijde

```
private void jButtonBerekenCirkelActionPerformed(java.awt.event.ActionEvent evt) {  
    int straal = Integer.parseInt(this.jTextFieldStraal.getText());  
    Cirkel c = new Cirkel(straal);  
    this.jLabelWaardeOmtrek.setText(Double.toString(helper.afRonden(c.omtrek(), 2)));  
    this.jLabelWaardeOpp.setText(Double.toString(helper.afRonden(c.oppervlakte(), 2)));  
}
```

```
private void jButtonBerekenVierkantActionPerformed(java.awt.event.ActionEvent evt) {  
    int zijde = Integer.parseInt(this.jTextFieldZijde.getText());  
    Vierkant v = new Vierkant(zijde);  
    this.jLabelWaardeOmtrek.setText(Double.toString(helper.afRonden(v.omtrek(), 2)));  
    this.jLabelWaardeOpp.setText(Double.toString(helper.afRonden(v.oppervlakte(), 2)));  
}
```

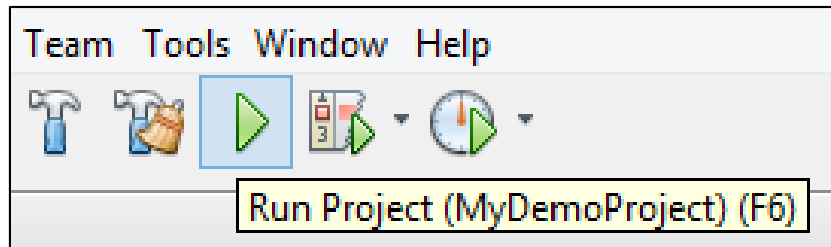

Netbeans-IDE: structuur

- Meerdere projecten

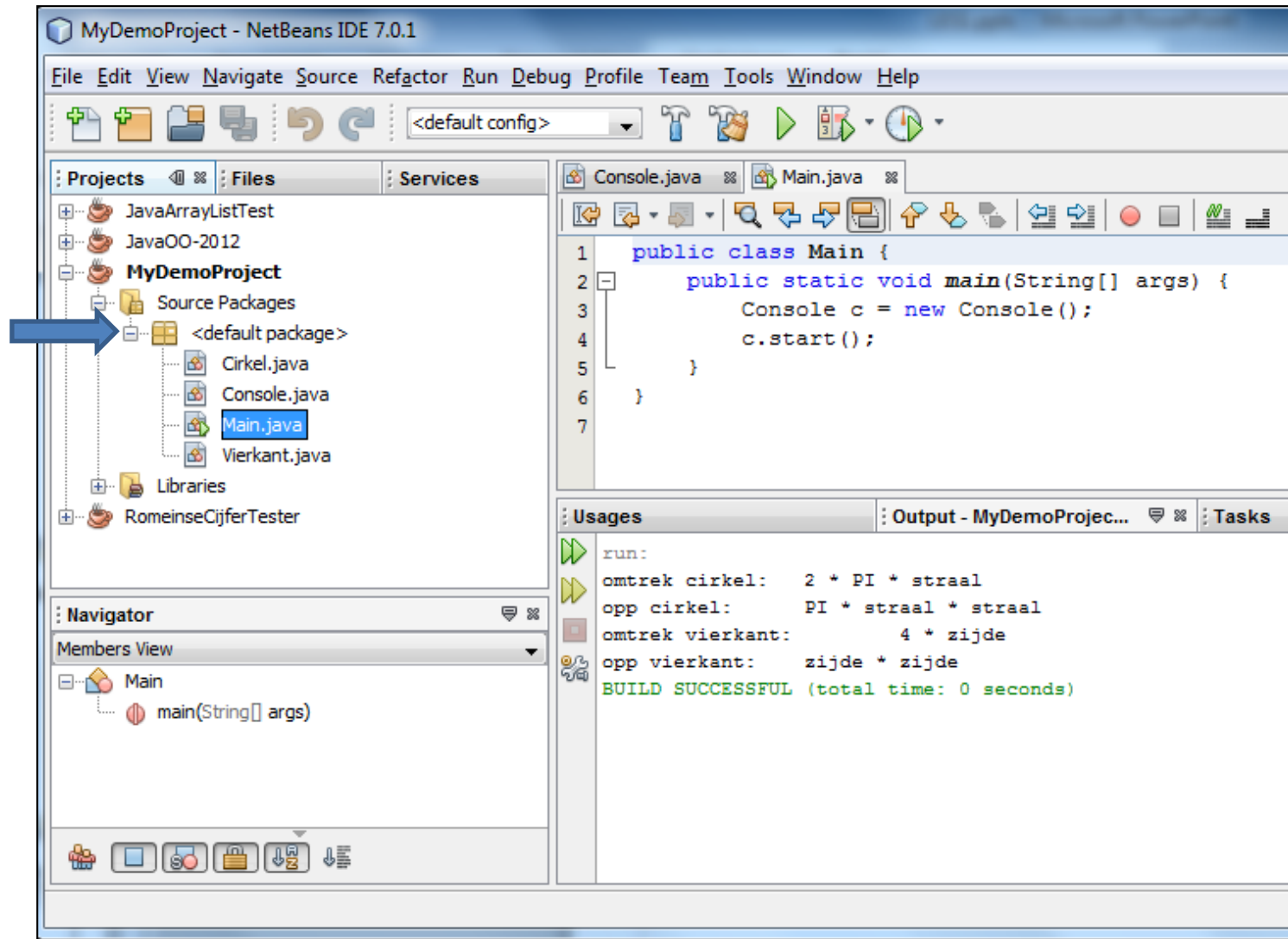


File > New Project > Java Application

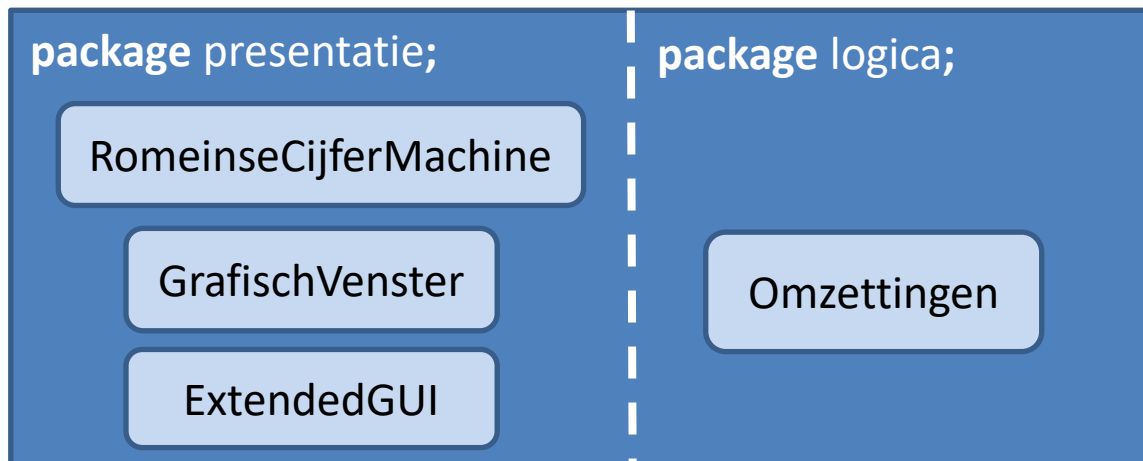
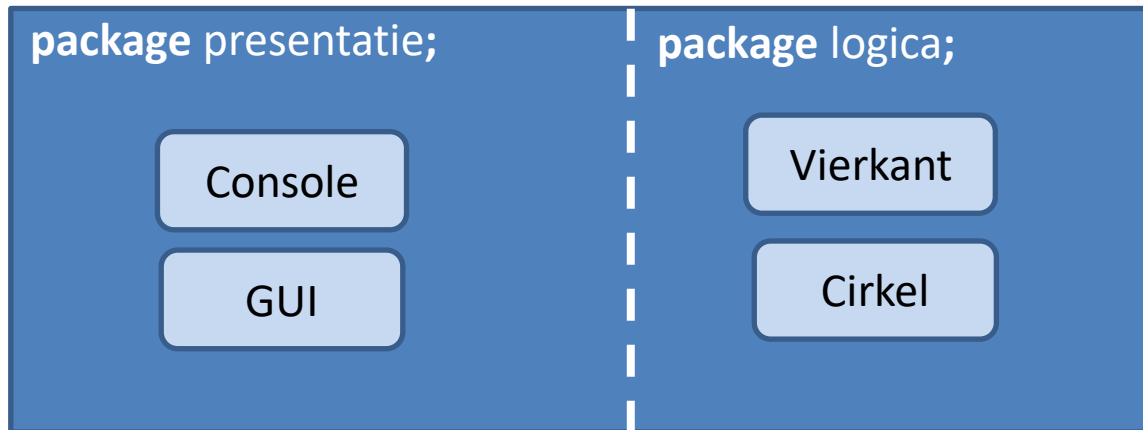
- Selecteer project en kies in menu:



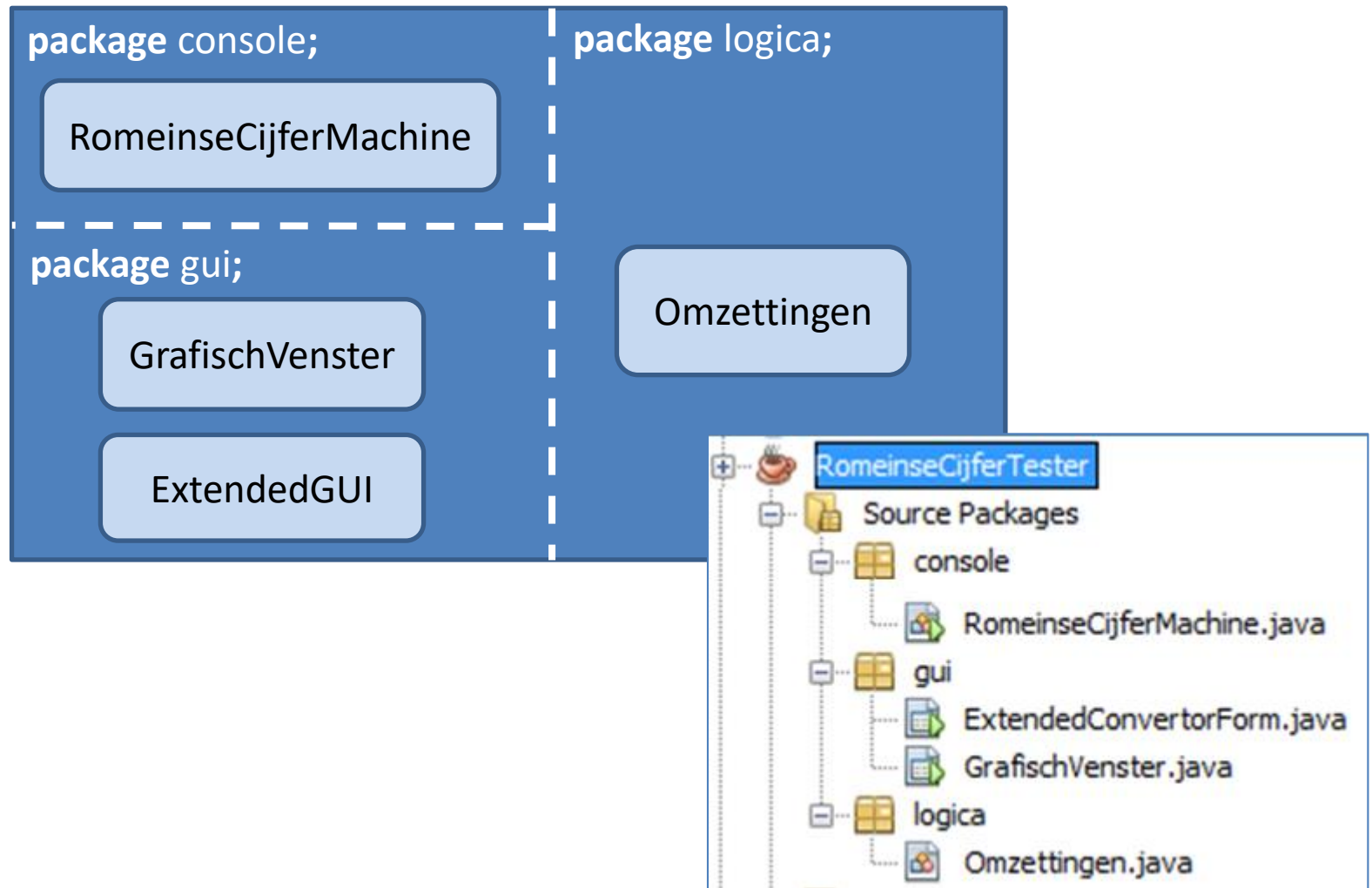
Alles in eenzelfde (default) package...



Beter: Code organiseren in Java packages

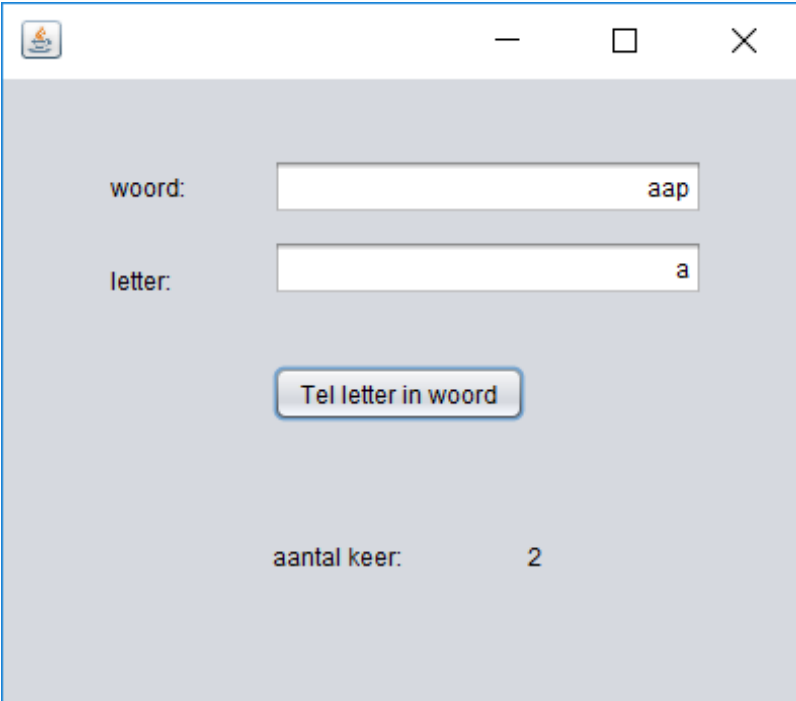


Beter: Code organiseren in Java packages



Meerdere programma's in 1 project (bv. Console & Grafische versie v/e toep)

```
run:  
Geef een woord: aap  
Geef een letter: a  
De letter a komt 2 keer voor in woord aap  
Geef een letter: |
```

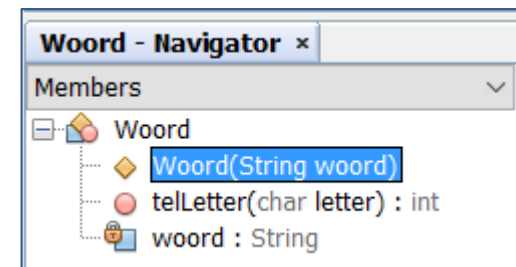


woord: aap

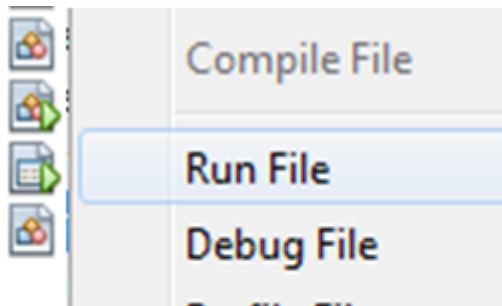
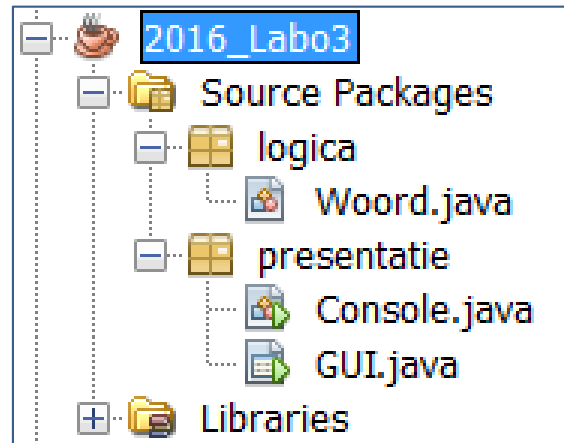
letter: a

Tel letter in woord

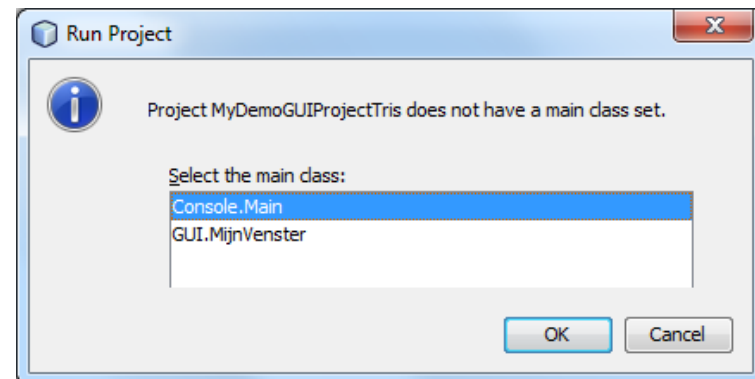
aantal keer: 2



Meerdere programma's in 1 project (bv. Console & Grafische versie v/e toep)

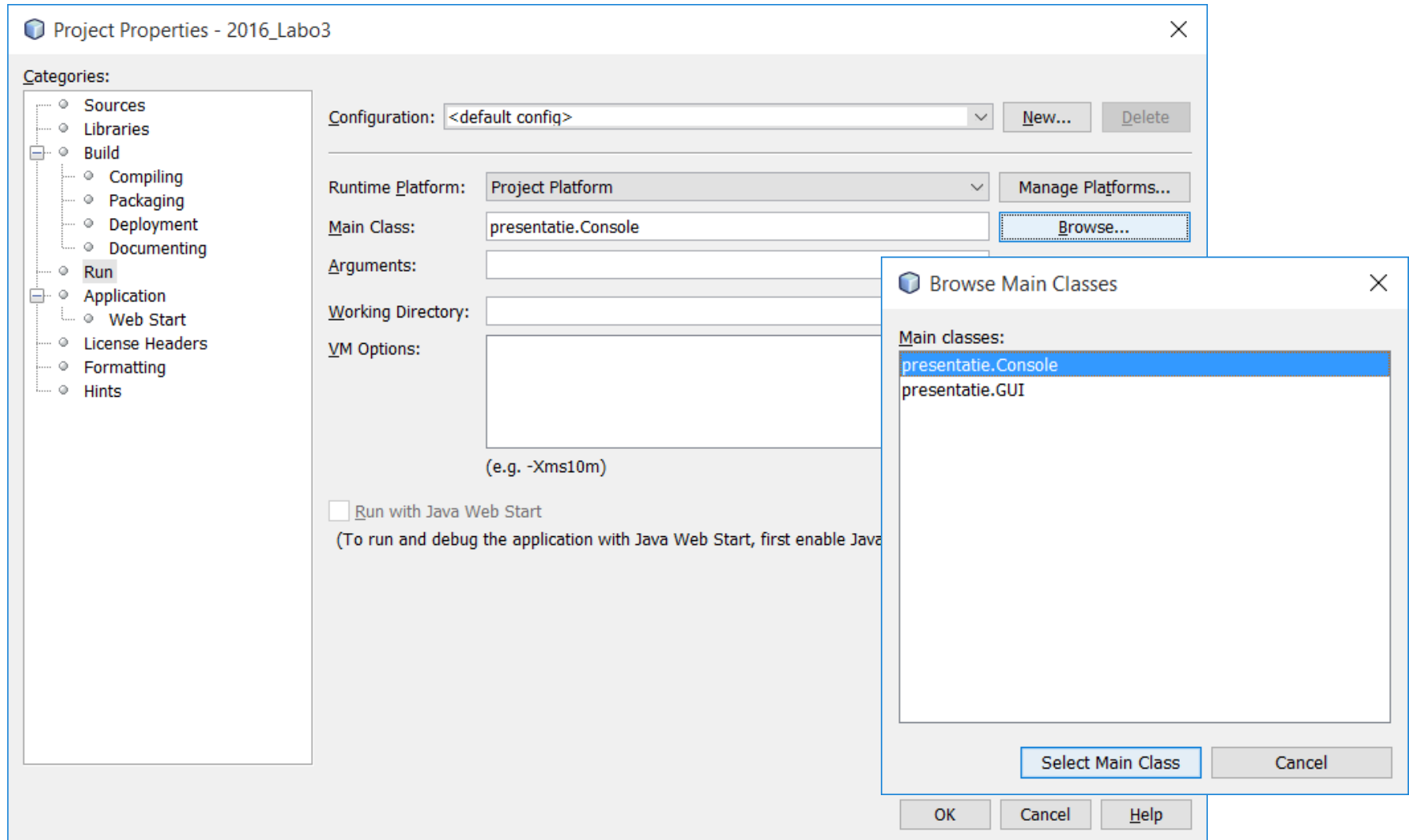


Selecteer gewenste programma
om te 'runnen' of te 'debuggen'



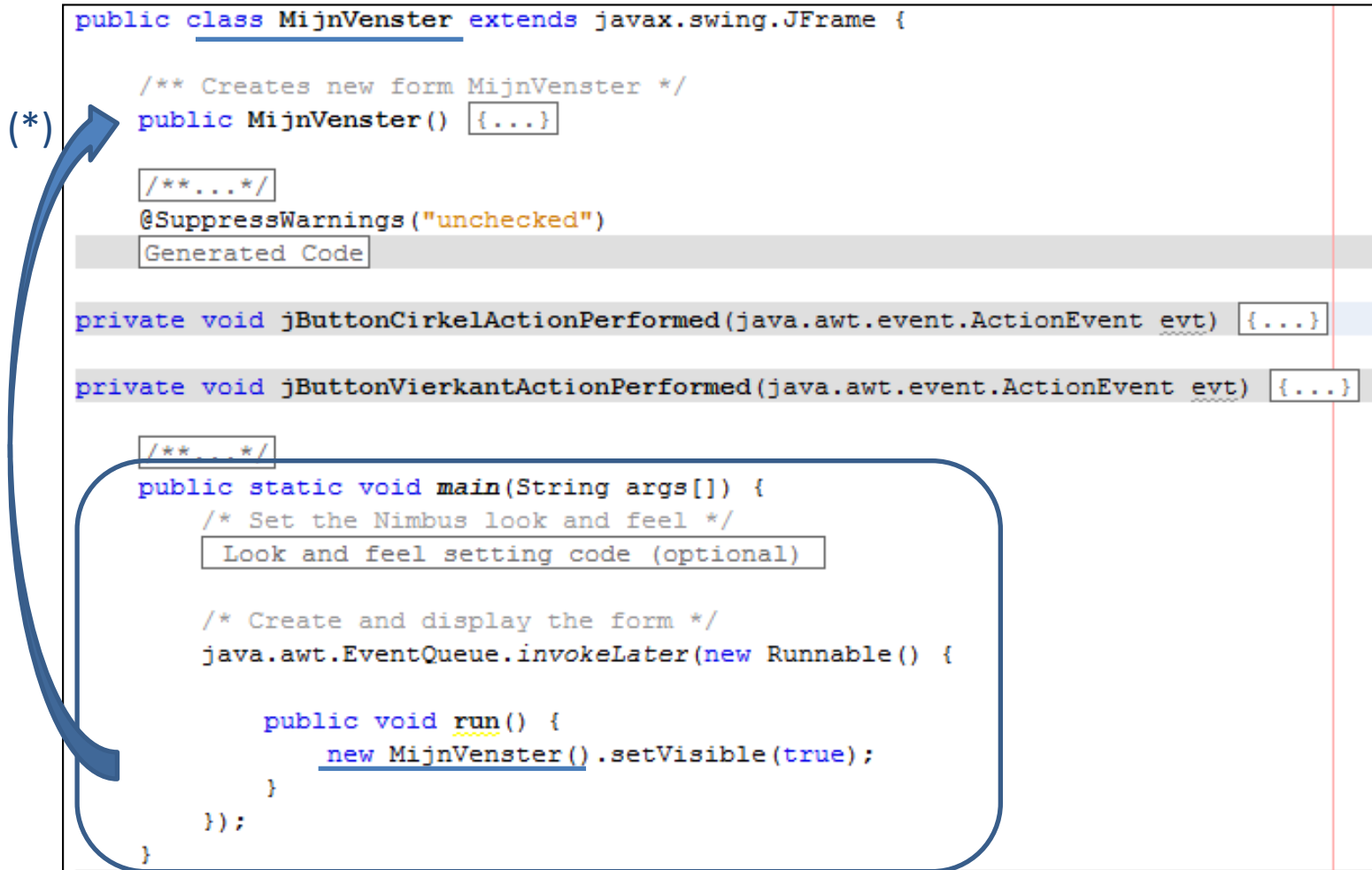
Of stel in als 'main class'

Andere 'main class' instellen via properties-venster van het project



Elk grafisch programma heeft zijn 'main' methode

Project > new JFrame form



```
public class MijnVenster extends javax.swing.JFrame {

    /** Creates new form MijnVenster */
    public MijnVenster() {...}

    /**...*/
    @SuppressWarnings("unchecked")
    Generated Code

    private void jButtonCirkelActionPerformed(java.awt.event.ActionEvent evt) {...}

    private void jButtonVierkantActionPerformed(java.awt.event.ActionEvent evt) {...}

    /**...*/
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new MijnVenster().setVisible(true);
            }
        });
    }
}
```


Import van nodige packages

```
public class Console {  
    public static void main(String[] args) {  
        Punt p1 = new Punt(1,2);  
        Punt p2 = new Punt(3,4);  
    }  
}
```



```
import logica.Punt;  
  
public class Console {  
    public static void main(String[] args) {  
        Punt p1 = new Punt(1,2);  
        Punt p2 = new Punt(3,4);  
    }  
}
```

Navigate	
Show Javadoc	Alt+F1
Find Usages	Alt+F7
Call Hierarchy	
Insert Code...	Alt+Insert
Fix Imports	Ctrl+Shift+I

import van 'meerdere' klassen van andere package

```
1 package les1.formules.console;
2
3 import logica.Cirkel;
4 import logica.Vierkant;
5
6 public class Console {
7
8     public void start() {
9         Cirkel c = new Cirkel();
10        System.out.println(c.geefFormules());
11
12        Vierkant v = new Vierkant();
13        System.out.println(v.geefFormules());
14    }
15 }
```

```
1 package les1.formules.console;
2
3 import logica.*;
4
5 public class Console {
6
7     public void start() {
8         Cirkel c = new Cirkel();
9        System.out.println(c.geefFormules());
10
11        Vierkant v = new Vierkant();
12        System.out.println(v.geefFormules());
13    }
14 }
```

Zichtbaarheid van methoden

Documenteren van methoden

```
private void jButtonVierkantActionPerformed(java.awt.event.ActionEvent evt) {  
    this.jTextAreaFormules.setText(new Vierkant().g);  
}
```

geefFormules () String
getClass () Class<?>

Vierkant

```
public String geefFormules ()
```

Javadoc not found. Either Javadoc documentati

public <>private javadoc

```
/**
 * Een eenvoudige Vierkant-klasse
 * @author kristien.vanassche
 */
public class Vierkant {
    private String geefFormuleOmtrek() {
        return "omtrek = 4 * zijde";
    }

    private String geefFormuleOppervlakte() {
        return "oppervlakte = zijde * zijde";
    }

    /**
     * Geeft formule terug voor omtrek en oppervlakte van een vierkant
     * @return formules (opp & omtrek) in tekstvorm
     */
    public String geefFormules() {
        return "vierkant:\r\n\t" + geefFormuleOmtrek() + "\r\n\t" + geefFormuleOppervlakte();
    }
}
```

Java documentatie

The screenshot shows an IDE with a Java file. The code includes two event listener methods and a main method. A tooltip for the `geefFormules()` method is displayed over the code. The tooltip shows the method signature and a description of its purpose.

```
private void jButtonCirkelActionPerformed(java.awt.event.ActionEvent evt) {  
    this.jTextAreaFormules.setText(new Cirkel().geefFormules());  
}  
  
private void jButtonVierkantActionPerformed(java.awt.event.ActionEvent evt) {  
    this.jTextAreaFormules.setText(new Vierkant().geefFormules());  
}  
  
/**  
 * @param args the command line arguments  
 */  
public static void main(String args[]) {  
    /* Set the Nimbus look and feel */  
    /* Look and feel setting code (optional) */  
}
```

geefFormules() String

- toString() String
- equals(Object obj) boolean
- getClass() Class<?>
- hashCode() int
- notify() void
- notifyAll() void
- wait() void
- wait(long timeout) void
- wait(long timeout, int nanos) void

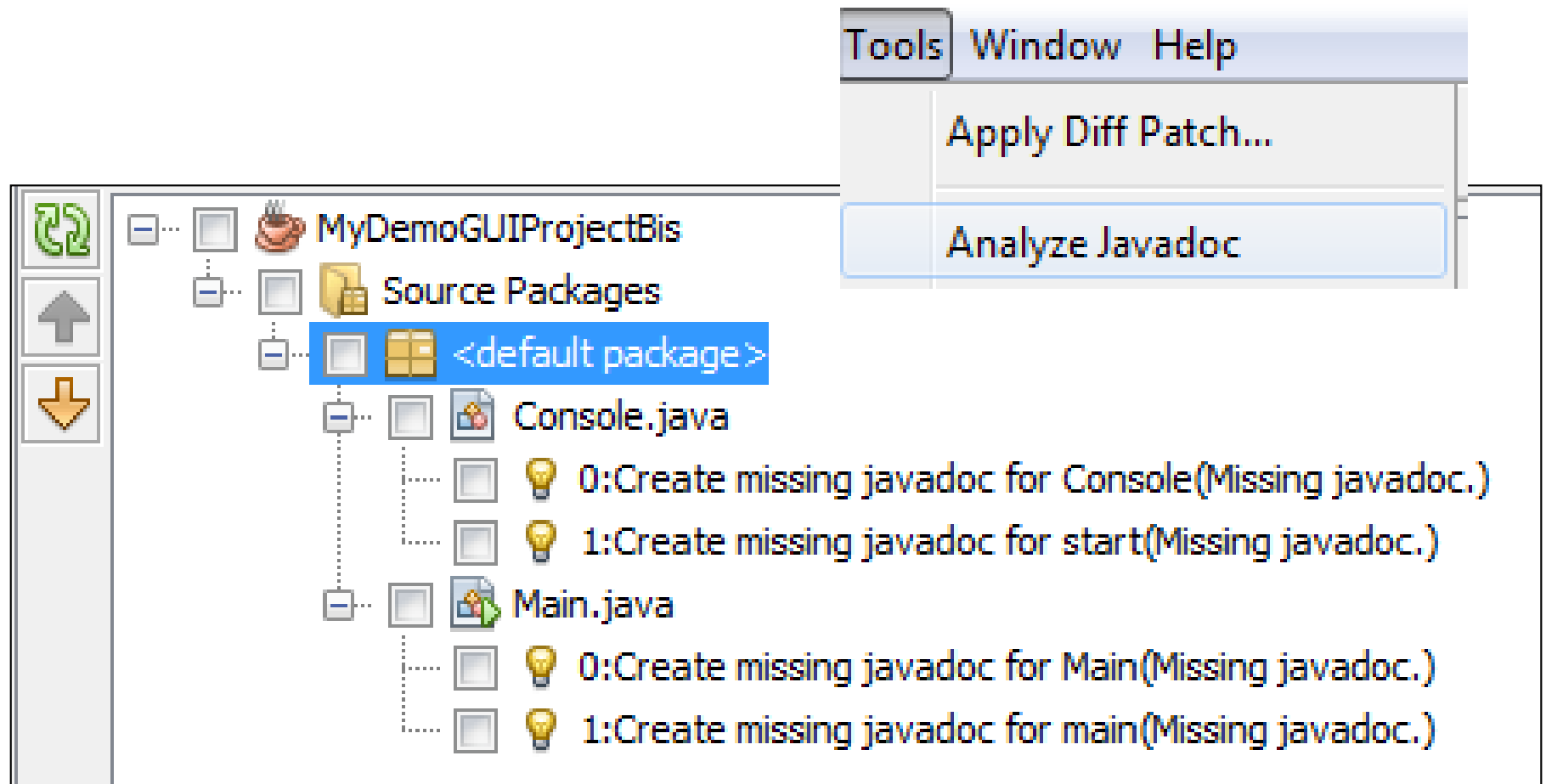
Vierkant

```
public String geefFormules()  
  
Geeft formule terug voor omtrek en oppervlakte van een vierkant
```

Returns:

formules (opp & omtrek) in tekstvorm

Analyse Javadoc

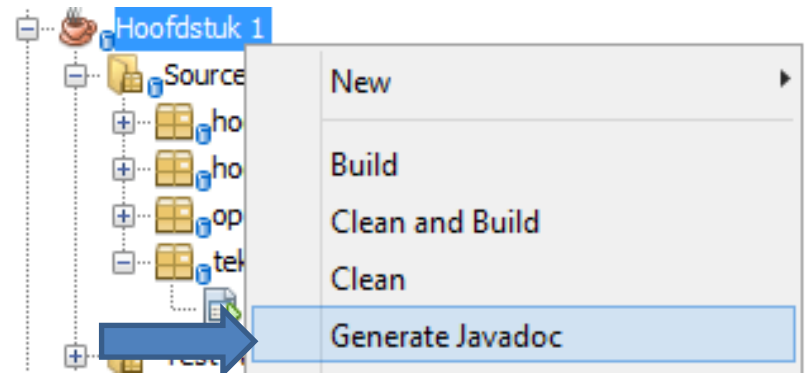
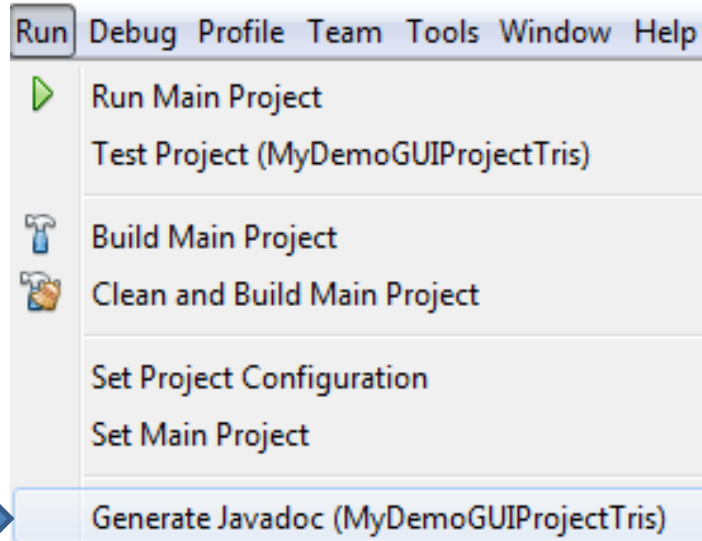


Noot: Controle op publieke methoden, maar niet op private

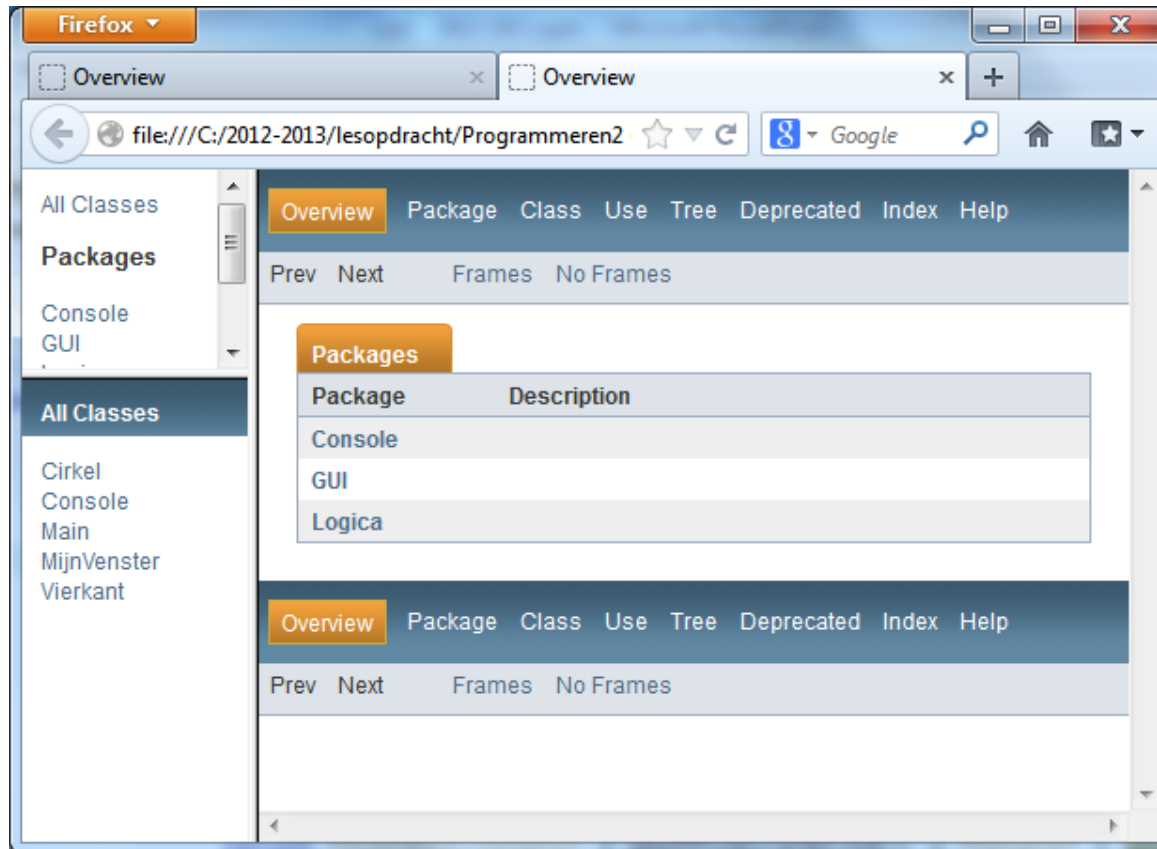
Generate Javadoc

Run menu
-> generate javadoc

project selecteren
-> generate javadoc



Javadoc resultaat



Javadoc-ondersteuning

The screenshot shows an IDE with a code editor on the left and a Javadoc window on the right. The code editor contains the following Java code:

```
System.out.print("Geef een woord in: ");
inputWoord = sc.next();

while (!inputWoord.equals("")) {
    woord = new Woord(inputWoord);

    System.out.print("Geef een letter in: ");
    inputLetter = sc.next();

    while (inputLetter != "" && inputLetter.length() == 1) {
        int aantal = woord.
        System.out.println(aantal + " keer " + inputLetter + " komt voor in " + woord);
        System.out.print("Geef een letter in: ");
    }
}
```

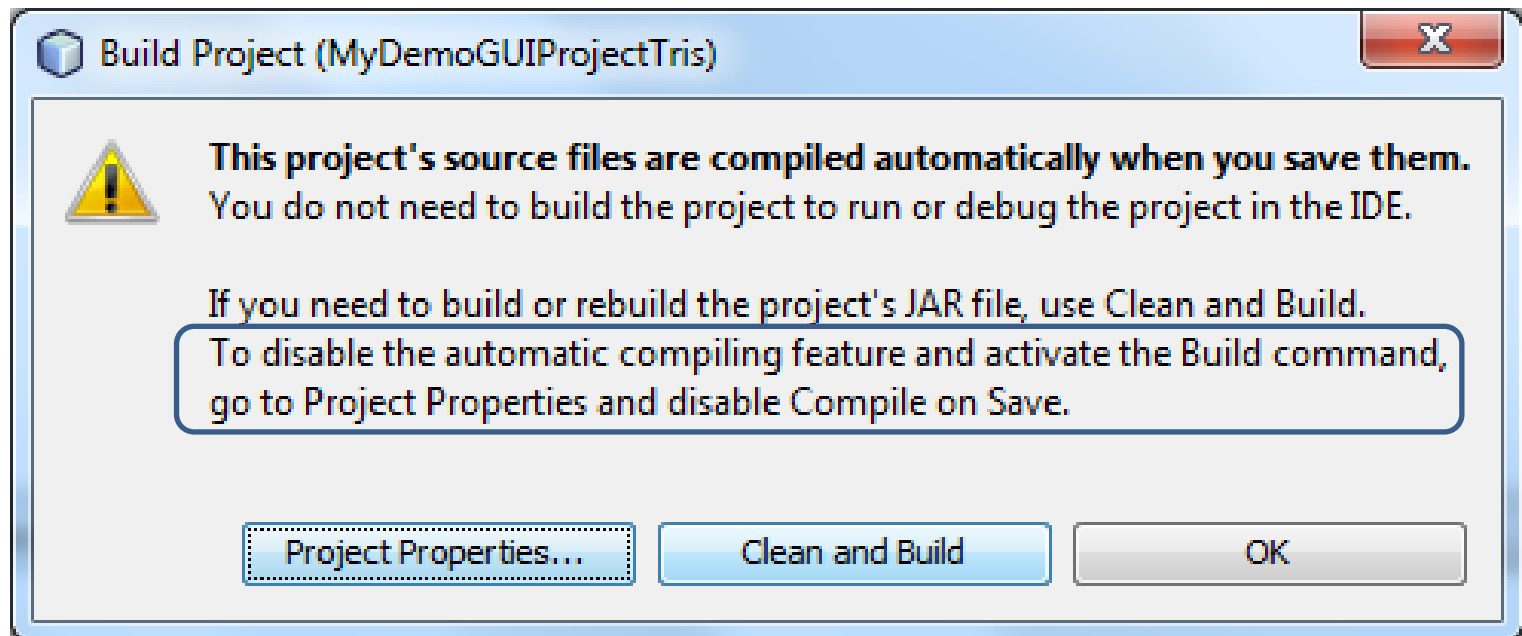
The Javadoc window displays the documentation for the `lab01_00.Woord` class. It includes the following information:

- Class:** `lab01_00.Woord`
- Method:** `public int telLetter(char letter)`
- Description:** telt hoeveel keer een opgegeven letter voorkomt in het gestockeerde woord
- Parameters:**
 - `letter` - de opgegeven letter
- Returns:** het aantal keer dat de opgegeven letter voorkomt in het woord

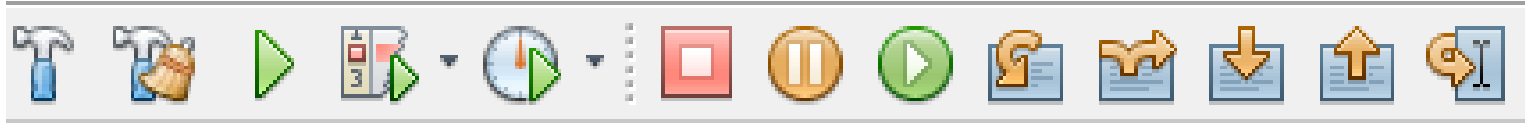
Below the Javadoc window, a small table shows the return type and parameter for the selected method:

Return Type	Parameter
<code>int</code>	<code>hashCode()</code>
<code>int</code>	<code>telLetter(char letter)</code>
<code>String</code>	<code>woord</code>

Automatische compilatie




Netbeans: Krachtig debuggen



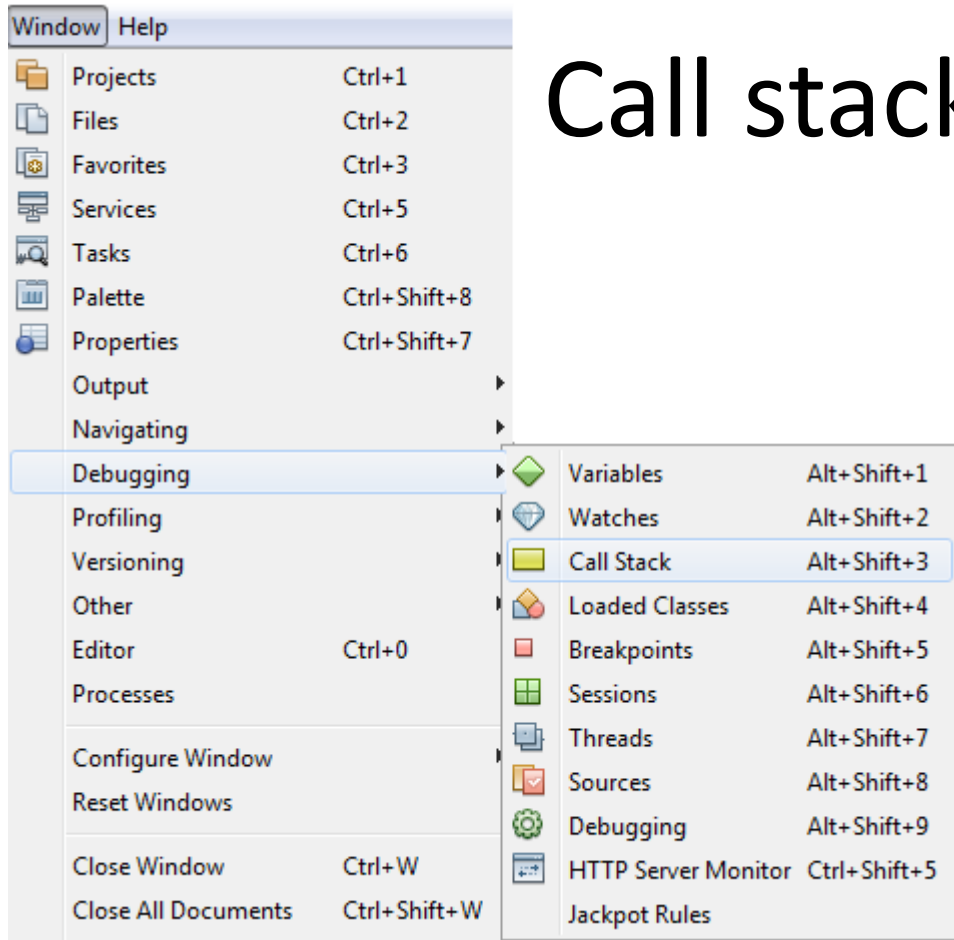
- F6 (**run**) – F11 (**build**)
- F7 (**step into**) - CTRL F7 (**step out**) DEMO:
F7 --> JAVA API --> ctrl F7
- F8 (**step over**)
- **Run to cursor**
- Watch local variables
- Set breakpoint (ook @runtime !)
- View stack trace

Lokale variabelen consulteren

ut	Search Results	Analyzer	Javadoc	Tasks	Variables %	
Name		Type			Value	
[-]	◆ this	MijnVenster			...	#1150
[+]	◆ jButtonCirkel	JButton			...	#1392
[+]	◆ jButtonVierkant	JButton			...	#1393
[+]	◆ jLabelFormules	JLabel			...	#1396
[+]	◆ jScrollPane1	JScrollPane			...	#1157
[+]	◆ jTextAreaFormules	JTextArea			...	#1378
[-]	Static				...	
	◆ teller	int			...	1

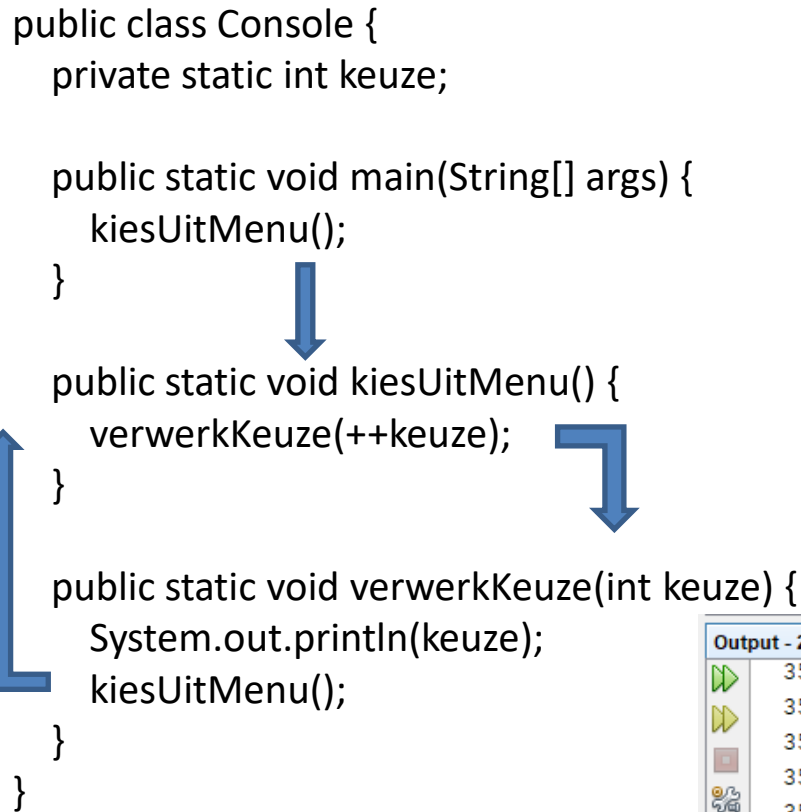


Call stack venster



Opgelet: Circulair hoppen geeft StackOverflowError

```
public class Console {  
    private static int keuze;  
  
    public static void main(String[] args) {  
        kiesUitMenu();  
    }  
  
    public static void kiesUitMenu() {  
        verwerkKeuze(++keuze);  
    }  
  
    public static void verwerkKeuze(int keuze) {  
        System.out.println(keuze);  
        kiesUitMenu();  
    }  
}
```



Output	Test Results	Variables	Call Stack ×
Name			
Console.kiesUitMenu:15			
Console.verwerkKeuze:20			
Console.kiesUitMenu:15			
Console.verwerkKeuze:20			
Console.kiesUitMenu:15			
Console.verwerkKeuze:20			
Console.kiesUitMenu:15			
Console.verwerkKeuze:20			
Console.kiesUitMenu:15			
Console.verwerkKeuze:20			
Console.kiesUitMenu:15			
Console.main:11			

Output - 2019_CodeTheorieJava00

```
3517  
3518  
3519  
3520  
3521  
3522  
3523  
3524
```

```
Exception in thread "main" java.lang.StackOverflowError  
    at java.nio.CharBuffer.arrayOffset(CharBuffer.java:1021)  
    at sun.nio.cs.UTF_8.updatePositions(UTF_8.java:77)  
    at sun.nio.cs.UTF_8.access$200(UTF_8.java:57)  
    at sun.nio.cs.UTF_8$Encoder.encodeArrayLoop(UTF_8.java:636)  
    at sun.nio.cs.UTF_8$Encoder.encodeLoop(UTF_8.java:691)  
    at java.nio.charset.CharsetEncoder.encode(CharsetEncoder.java:570)
```

Opmerking:

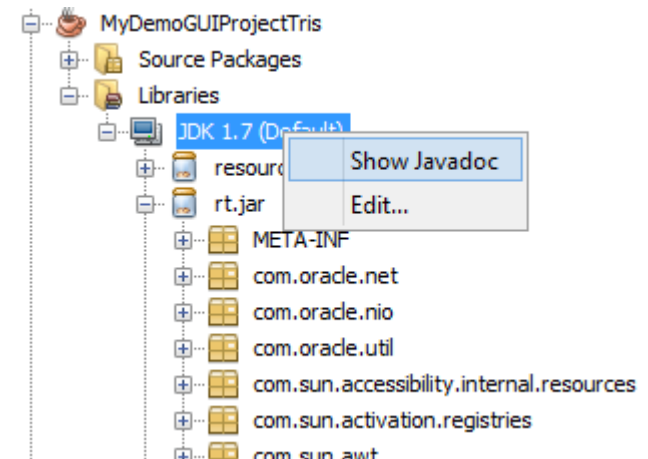
Deze exception kán je niet opvangen

Netbeans: Krachtig coderen

- Code completion (**CTRL SPATIE**)
- Shortcuts: *sop*, *psvm*, ...
- **this.** >> selectie via popup (veld, methode)
- **this.j** >> selectie Swing component
- **this.jButt** >> selectie Swing button component
- Fix imports (CTRL-SHIFT-I)
- Format source (ALT-SHIFT-F)
- GUI drag & drop (WYSIWYG)
- compilatie-aanwijzingen tijdens typen
- Dubbel-klik op foutmeldingen >> plaats van fout

Netbeans: Krachtig documenteren

- Javadoc schrijven (demo)
- Javadoc raadplegen
 - In Navigator venster over signatuur hoveren
 - In Editor venster op methode-oproep klikken
- Javadoc html genereren
- UML Klassendiagramma (plug-in)
- Bibliotheken (show Javadoc)
 - JDK 1.8 (default)
 - CodeLibrary.jar



Kruispunt – Verkeerslicht – Auto's

```
/**
 * Met deze methode wordt de verkeersdoorstroom van een kruispunt gesimuleerd
 * @param aantalAutos Het aantal auto's dat het kruispunt moet oversteken
 * @param vertrektijdAuto De tijd die een auto nodig heeft om het kruispunt over te steken
 * @throws InterruptedException
 */
public void run(int aantalAutos, int vertrektijdAuto) throws InterruptedException {
    initAutos(aantalAutos);

    //eigenlijk testscenario
    while (true) {
        runGroen(vertrektijdAuto);
        runOranje();
        runRood();
    }
}
```

run - Navigator X

Members <empty>

- Kruispunt
 - autos : Auto[]
 - licht : Verkeerslicht
 - curAutoAanLicht : int
 - setVerkeerslicht(int tijdRood, int tijdOranje, int tijdGroen)
 - run(int aantalAutos, int vertrektijdAuto)**
 - initAutos(int aantalAutos)
 - runGroen(int vertrektijdAuto)
 - runOranje()
 - runRood()

[2016_les3.MijnAutoVerkeerslichtConsoleProject.Kruispunt](#)

public void run(int aantalAutos, int vertrektijdAuto) throws [InterruptedException](#)

Met deze methode wordt de verkeersdoorstroom van een kruispunt gesimuleerd

Parameters:

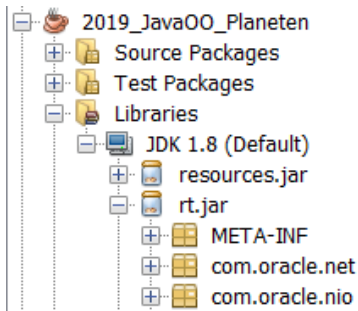
- aantalAutos - Het aantal auto's dat het kruispunt moet oversteken
- vertrektijdAuto - De tijd die een auto nodig heeft om het kruispunt over te steken

Throws:

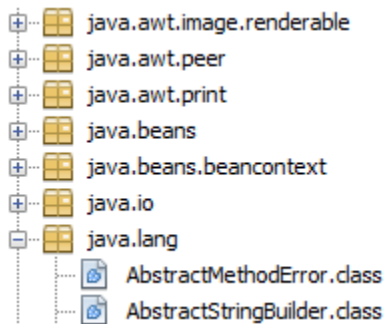
[InterruptedException](#)

Press Ctrl+F1 to enlarge

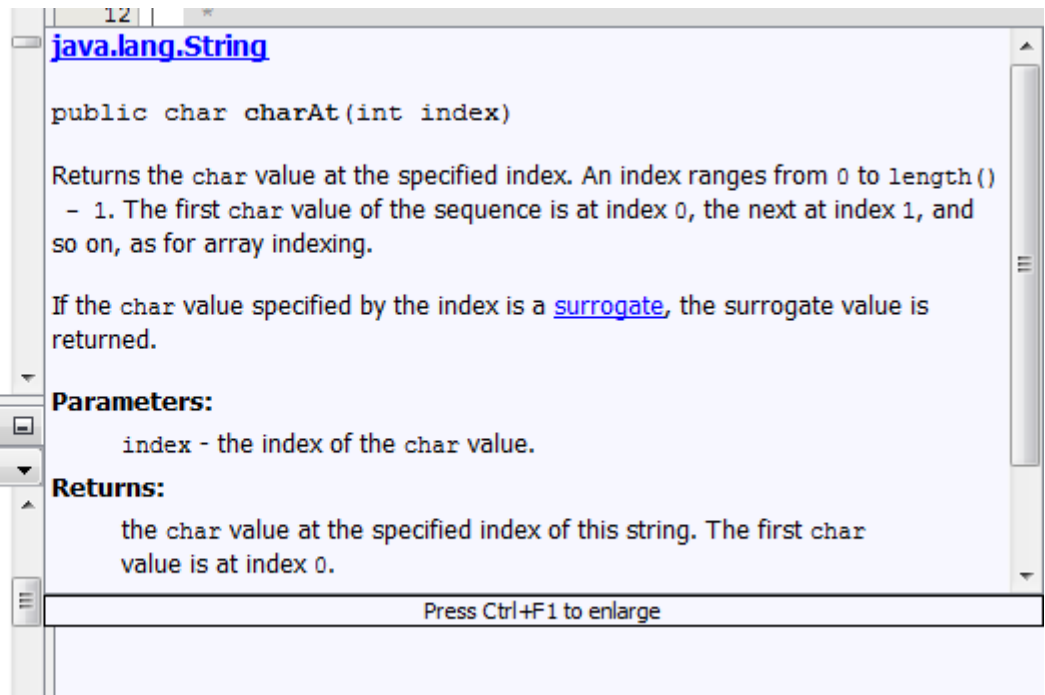
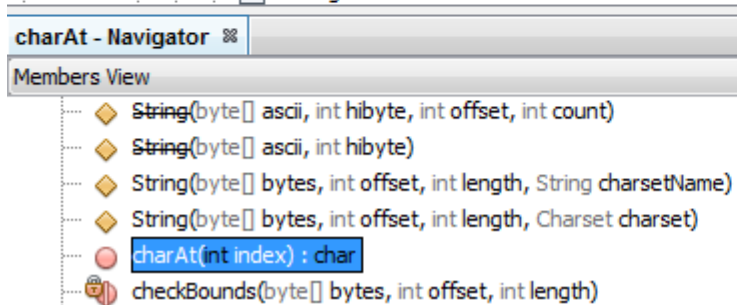
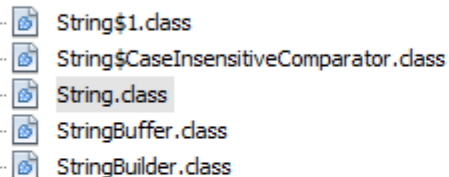
JDK 1.8 (Default) klassenbibliotheek



...



...

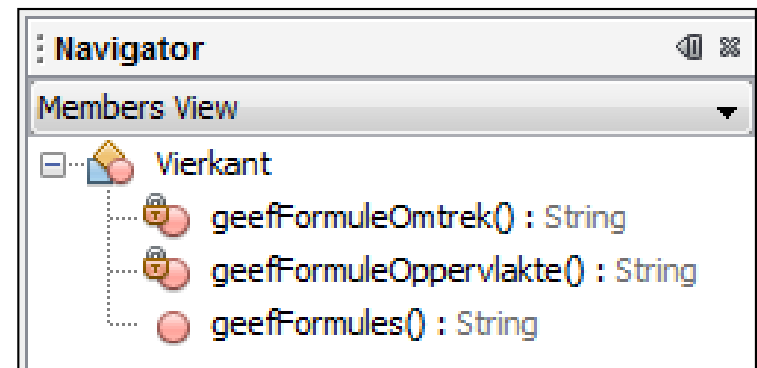


Netbeans: Krachtig structureren

- Meerdere projecten
- Meerdere packages (package import + definitie)
- Meerdere klassen
- Scheiden presentatie en logica
 - Console versie + **logicaX**
 - Grafische versie + **logicaX**
 - Uitgebreide grafische versie + **logicaX**
- Hergebruik van code (private methoden, Single Responsibility (i.e. 'S' van de SOLID principles))

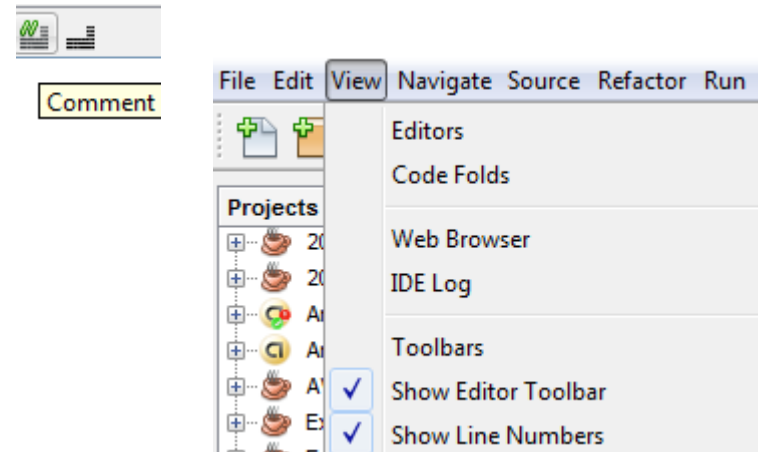
Netbeans: Makkelijk in onderhoud

- Easy Refactoring
- Naamconventie Swing-componenten
 - JButtonPlus, JButtonMin, JButtonMaal
 - jTextFieldVoornaam, jTextFieldFamiliennaam
- Overzicht in Navigator venster
 - Signaturen
 - Jump to code
 - Javadococumentatie



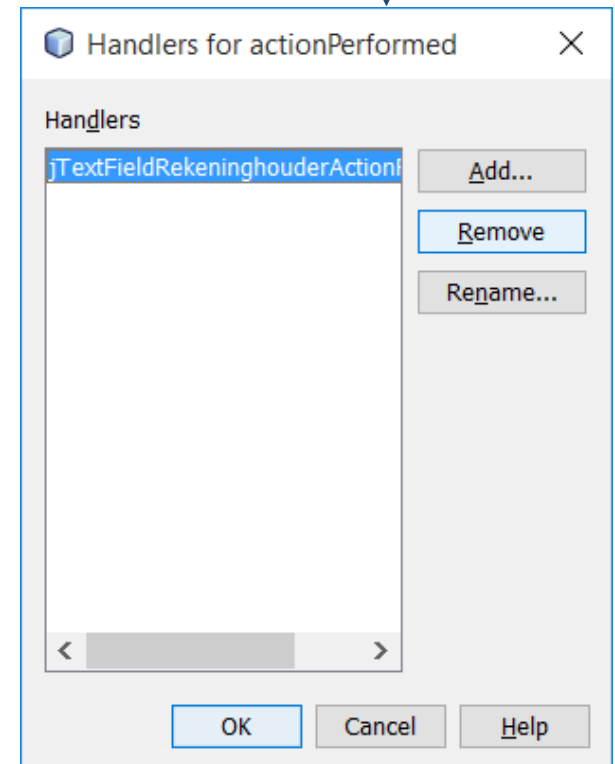
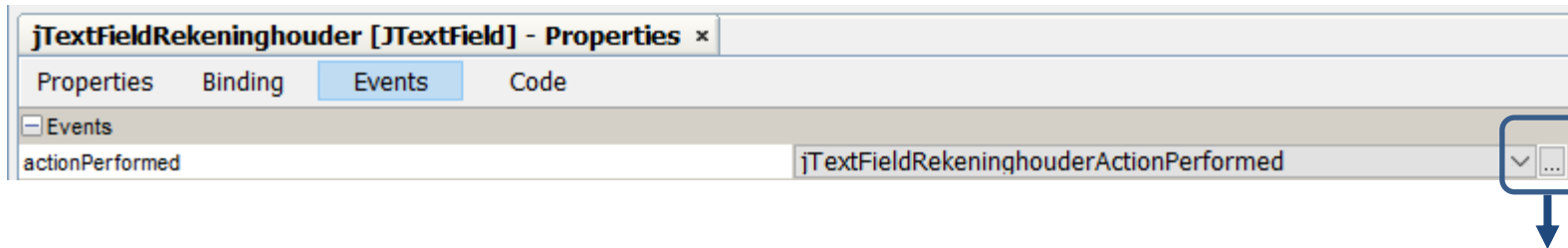
Netbeans: Allerlei

- Comment / Uncomment
- Show Line Numbers
- Reset Windows
- Een event verwijderen
- Rename GUI component => related event methode
- (ver-)slepen van bestanden naar (andere) package
- package naamconventie: [be.ikdoeict.util](#)



Event handler methode verwijderen

```
private void jTextFieldRekeninghouderActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```



Besluit: Netbeans

- Krachtig debuggen
- Krachtig coderen
- Krachtig documenteren
- Krachtig structureren
- Makkelijk in onderhoud
- Allerlei