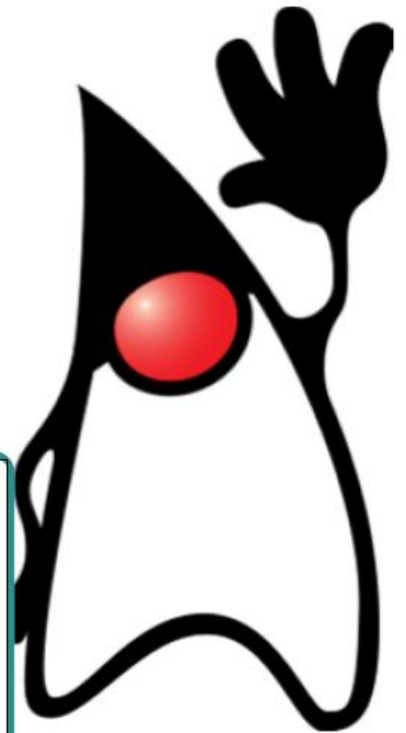


Methoden in Java - vervolg

Klasgroep	1EO-ICT
Opleiding	Bachelor Elektronica-ICT
Lokaal	Groot Auditorium
Tijdstip	maandag lestijd 3
Docent	Katja Verbeeck
Contact	katja.verbeeck@odisee.be
Handboek	hfst 3 - p 81 - p 100



Inhoud

- ✓ Methoden met parameters en return type
- ✓ Statische Methoden
- ✓ Methoden documenteren – javadoc tags : @param - @return - @ see
- ✓ Method overloading
- ✓ Sleutelwoorden return-break-continue
- ✓ Parameters van de main methode

Methoden met parameters en return type

Zelf methoden schrijven met parameters en returntypes

Voorbeelden methodedefinities

```
public int genereerGetal()  
public int increment(int getal1)  
public int berekenSom(int getal1, int getal2)  
public char geefEersteLetter (String s)  
public void setLeeftijd (int leeftijd)  
public boolean isOud(short leeftijd)
```

Een methode heeft steeds 1 functionaliteit te volbrengen :

Daarom :

- denk goed na over input (parameters) en return (wat moet de methode berekenen en teruggeven als resultaat).
- Splits in nuttige deelfunctionaliteiten als de methode te lang wordt !
- Wanneer de methode geen resultaat teruggeeft (bv omdat een dataveld van waarde veranderd moet worden, dan geef je **void** terug

genereerGetal

int

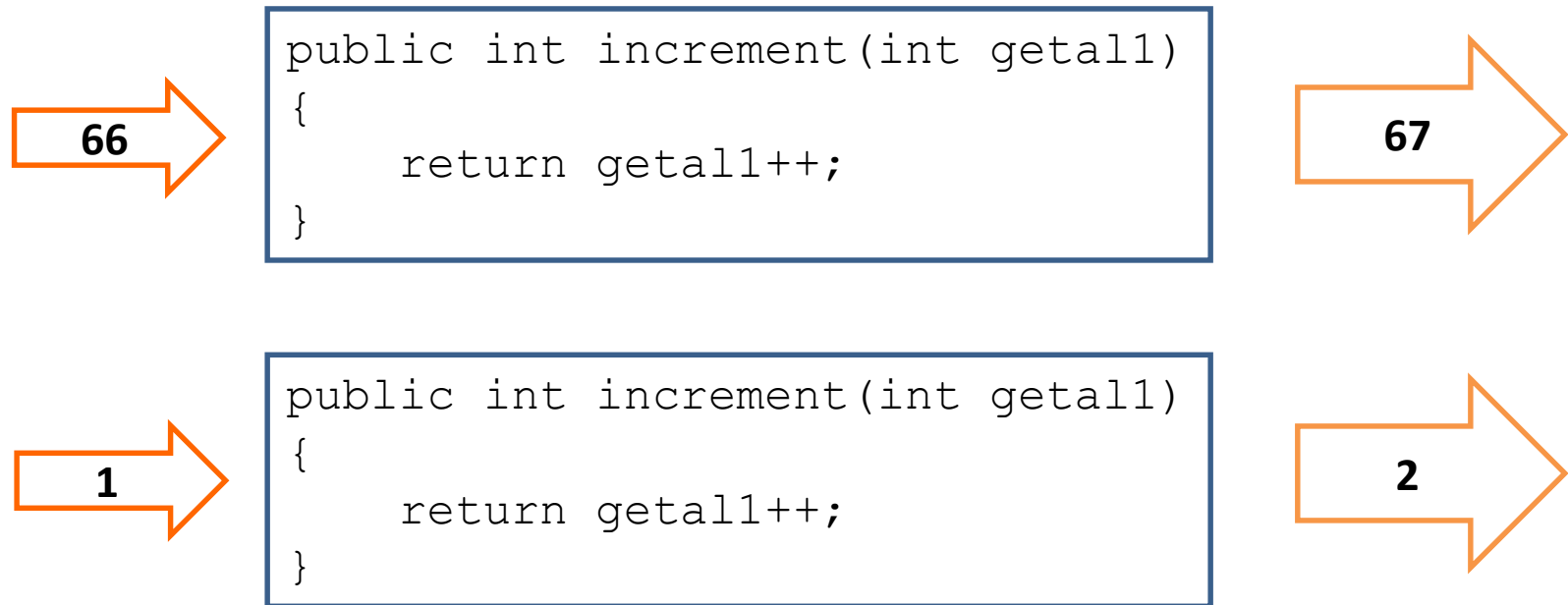
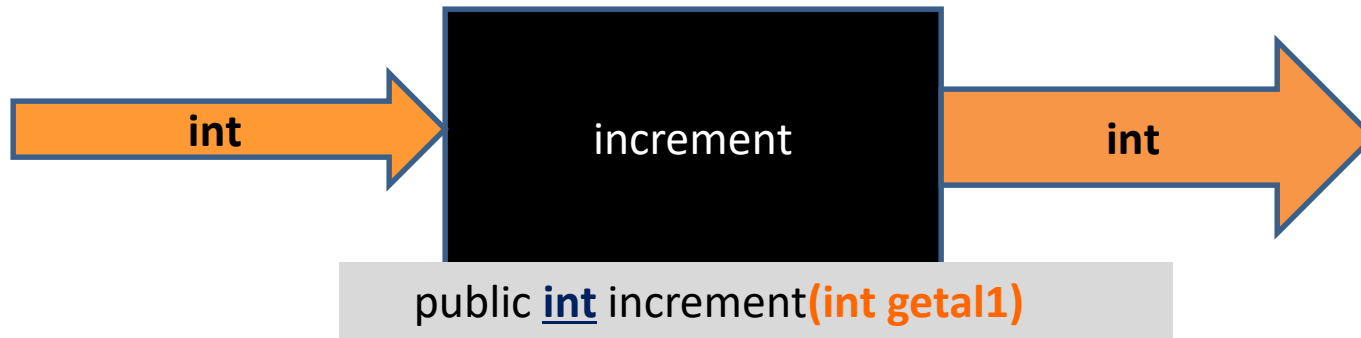
```
public int genereerGetal()
```

```
public int genereerGetal() {  
    return (int) Math.random();  
}
```

0

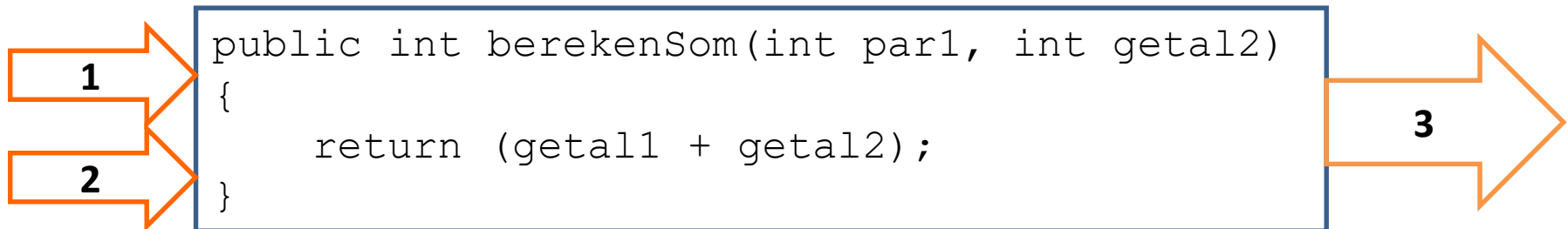
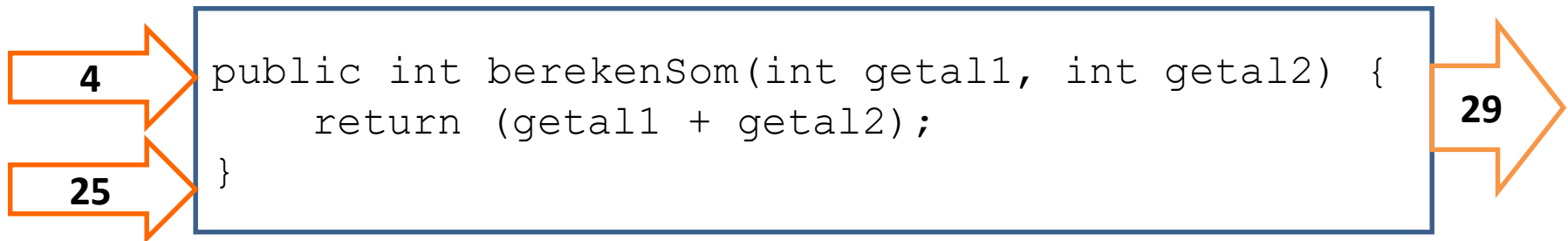
```
public int genereerGetal() {  
    return (int) (Math.random() * 10);  
}
```

9





```
public int berekenSom(int getal1, int getal2)
```



```
public char geefEersteLetter (String s)
```

```
public char geefEersteLetter(String s) {  
    return s.charAt(0);  
}
```

```
public void setLeeftijd (int leeftijd)
```

```
public void setLeeftijd(int leeftijd){  
    this.leeftijd = leeftijd; // dataveld leeftijd uit de klasse Persoon  
}
```

Hiermee duid je het
dataveld van het
huidige object aan
(zie verder semester2
Java OO)


```
public boolean isOud(short leeftijd)
```

```
public boolean isOud(short leeftijd){  
    boolean b;  
  
    if (leeftijd > 60) {  
        b = true;  
    }  
    else {  
        b = false;  
    }  
  
    return b;  
}
```

```
// beter  
public boolean isOud(short leeftijd) {  
    return (leeftijd > 60);  
}
```

Arrays als parameters en returntypes

```
public int geefSom(int[] getallen);
```

```
public int geefSom(int[] getallen) {  
    int som = 0;  
  
    for (int i = 0; i < getallen.length; i++) {  
        som += getallen[i];  
    }  
  
    return som;  
}
```

Arrays als parameters en returntypes

```
public int[] genereerGetallen(int aantal);
```

```
public int[] genereerGetallen(int aantal) {  
    int[] getallen = new int[aantal];  
  
    for (int i = 0; i < getallen.length; i++) {  
        getallen[i] = i + 1;  
    }  
  
    return getallen;  
}
```

Arrays als parameters en returntypes

```
public boolean zoek(int[] rij, char c);
```

```
public boolean zoek(char[] rij, char c) {  
    boolean gevonden = false;  
  
    for (int i = 0; i < rij.length; i++) {  
        if(rij[i] == c) {  
            gevonden = true;  
            break;  
        }  
    }  
  
    return gevonden;  
}
```

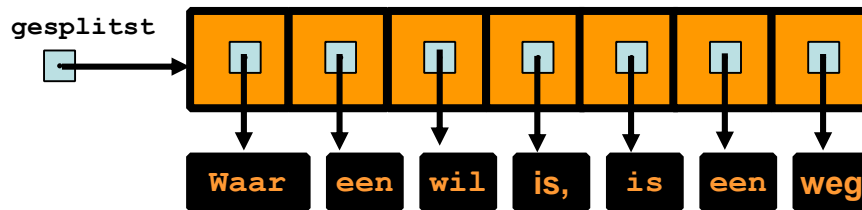
```
public boolean zoek(char[] rij, char c) {  
    for (int i = 0; i < rij.length; i++) {  
        if(rij[i] == c) {  
            return true;  
        }  
    }  
  
    return false;  
}
```

Arrays als parameters en returntypes

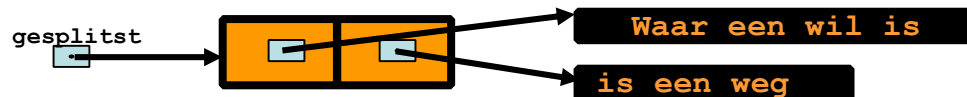
```
// methode split van de klasse String  
public String[] split(String s);
```

```
String mijnTekst = "Waar een wil is, is een weg";
```

```
String[] gesplitst = mijnTekst.split(" ");
```



```
String[] gesplitst = mijnTekst.split(",");
```



Algemeen : methoden met parameters en return type

```
public <returntype> methodeVanKlasseX(<parameterlijst,incl type>)
```

```
<returntype> resultaat;  
resultaat = <objectVanKlasseX>.methodeVanKlasseX(parameterwaarden);
```

klasse String

Methodedefinities:

```
public int indexOf(char c)
```

```
public char charAt(int i)
```

```
public String substring(int i,  
                        int j)
```

```
String obj = new String("hallo");
```

Oproepen:

```
int resultaat;  
resultaat = obj.indexOf('x');
```

```
char resultaat;  
resultaat = obj.charAt(2);
```

```
String resultaat;  
resultaat = obj.substring(1,4);
```

Statische methoden

Statische methoden

deze worden vooraf gegaan door het keyword **static** wanneer je een methode wil maken die **onafhankelijk** is **van de data members** in de klasse en die dus ook onafhankelijk is van gelijk welk object van die klasse dan kan je best de methode static maken. Je hoeft dan ook geen object aan te maken bij methode aanroep, je roept ze aan via de klasse!

Vb : de methoden in de klasse Math zijn static !

```
double getal = Math.random();
```


Voorbeelden statische methoden

Methoden van de klasse Math (statisch)

```
public static double random()  
public static double round(double x)  
public static double pow(double d1, double d2)
```

Methoden van de klasse Integer (statisch)

```
public static int parseInt(String s)  
public static String toString(int i)
```

Statische methoden in de main klasse

De main methode is zelf een statische methode. Wanneer je main methode te groot wordt kan je statische methoden schrijven in je main klasse. Zie bvb oef rijSorteer.java (labo9)

```
/**  
 * Je hebt een rij van 10 getallen. Sorteert de getallen van klein naar groot. Ga  
 * als volgt te werk: (a) zoek het kleinste getal van de 10 getallen (b)  
 * verwissel het kleinste getal met het eerste getal in de rij (c) zoek het  
 * kleinste getal van de overige 9 getallen (d) verwissel dat getal met het  
 * tweede getal in de rij (e) . . .  
 *  
 */
```

Statische methoden in de main klasse

```
public class RijSorteer {  
  
    public static void main(String[] args) {  
  
        int[] rij = {3, 1, 6, 2, 5, 10, 9, 8, 4, 7};  
  
        //afdrukken  
        drukRij(rij);  
        //sorteren  
        sorteerRij(rij);  
        //afdrukken  
        drukRij(rij);  
    }  
    ...  
}
```

Statische methoden in de main klasse

```
public static void drukRij(int[] rij) {  
    System.out.print("inhoud array : (");  
  
    int i;  
    for (i = 0; i < rij.length - 1; i++) {  
        System.out.print(rij[i] + ", ");  
    }  
  
    System.out.print(rij[i] + " )\n");  
}  
  
...
```

Statische methoden in de main klasse

```
public static void sorteerRij(int[] rij){
    int index;
    for (int i = 0; i < rij.length; i++) {
        int indexKleinste = i;

        //ga op zoek te gaan naar de index van het kleinste
        //element in de rij, startend vanaf een gegeven positie
        for (int j = i + 1; j < rij.length; j++) {
            if (rij[j] < rij[indexKleinste]) {
                indexKleinste = j;
            }
        }

        index = indexKleinste;

        if (i != index) {
            //verwissel i met index
            int temp = rij[i];
            rij[i] = rij[index];
            rij[index] = temp;
        }
    }
}

// klasse RijSorteer
```

Algemeen : statische methoden met parameters en return type

```
public static <returntype> methodeVanKlasseX(<parameterlijst,incl type>)
```

```
<returntype> resultaat;  
resultaat = NaamVanKlasseX.methodeVanKlasseX(parameters);
```

Definities:

```
klasse Math  
public static double pow(double x,  
double y)
```

```
klasse Integer  
public static int parseInt(String s)  
public static String toString(int i)
```

Oproepen:

```
double resultaat;  
resultaat = Math.pow(2,8);
```

```
int resultaat =  
    Integer.parseInt("123");  
String resultaat =  
    Integer.toString(123);
```

Hoe roep je volgende methoden op?

1) Uit klasse Math: **public static** double pow(double d1, double d2)

```
double macht = Math.pow(2,4);
```

*Formele parameters: d1,d2
Actuele parameters: 2,4*

2) Uit klasse String: **public** String substring(int i, int j)

```
String s = "Kermit";  
String sub = s.substring(2,4);
```

*Formele parameters: i,j
Actuele parameters: 2,4*

3) Uit klasse Math: **public static** double random()

```
double d = Math.random();
```

*Geen formele parameters
Dus ook geen actuele*

4) Uit klasse Integer: **public static** int parseInt(String s)

```
String tekst = "123";  
int i = Integer.parseInt(tekst);
```

*Formele parameter: s
Actuele parameter: tekst*

Javadoc -- tags :

@author - @version

@param - @return - @see

De klasse BinaireBewerking :

2 datavelden

```
package bewerkingen;
import java.util.Scanner;

/**
 * Documenteren van de klasse
 *
 * @author Katja Verbeeck
 * @version november 2016
 */

public class BinaireBewerking {
    /**
     * in dit veld wordt een eerste getal opgeslaan
     */
    int term1;
    /**
     * in dit veld wordt een tweede getal opgeslaan
     */
    int term2;
```

2 constructoren

```
/**
 * default constructor methode
 *
 */
public BinaireBewerking() {
    term1 = 0;
    term2 = 0;
}

/**
 * constructor methode
 * @param t1 de eerste term van de bewerking
 * @param t2 de tweede term van de bewerking
 */
public BinaireBewerking(int t1, int t2) {
    term1 = t1;
    term2 = t2;
}
```

1 methode

```
/**
 * Voert een gegeven rekenkundige bewerking uit op term1 en term2
 * @param operator De operator (+, -, *, / of %)
 * @return Het resultaat van de berekening
 */

public int berekenResultaat(char operator) {
    int res = -1;
    switch (operator){
        case '+': res = term1 + term2;
                break;
        case '-': res = term1 - term2;
                break;
        case '*': res = term1 * term2;
                break;
        case '/': res = term1 / term2;
                break;
        case '%': res = term1 % term2;
                break;
        default: break;
    }
    return res;
}
} -> einde van de klasse
```

De klasse Demo

```
package bewerkingen;
import java.util.Scanner;

public class BewerkingenDemo {
    /**
     * Documenteren van de hoofdmethode
     * @param args
     * @see #vraagBewerking
     * @see BinaireBewerking#berekenResultaat(char)
     * @see #druk
     */

    public static void main(String[] args) {
        BinaireBewerking b = new BinaireBewerking(7, -8);
        char operator;
        operator = vraagBewerking("Welke bewerking wil je uitvoeren? ");
        while(operator == '+' ||
                operator == '-' ||
                operator == '*' ||
                operator == '/' ||
                operator == '%'){
            int resultaat = b.berekenResultaat(operator);
            druk(b.term1 + " " + operator + " " + b.term2 + " = " + resultaat);
            operator = vraagBewerking("Welke bewerking wil je uitvoeren? ");
        };
        druk("Einde");
    }
}
```

Vervolg Demo

```
/**
 * Vraagt een bewerking
 * @param vraag de vraag
 * @return de ingelezen bewerking
 * @see Scanner#next()
 * @see String#charAt(int)
 */
public static char vraagBewerking(String vraag) {
    Scanner scan = new Scanner(System.in);
    System.out.print(vraag);
    char kar = scan.next().charAt(0);
    return kar;
}

/**
 * Drukt de gegeven info
 * @param info de af te drukken informatie
 * @see java.io.PrintStream#println(java.lang.String) */
public static void druk(String info) {
    System.out.println(info);
}
```



Typical forms for `@see package.class#member`

Referencing a member of the current class

`@see #field`
`@see #method(Type, Type,...)`
`@see #method(Type argname, Type argname,...)`
`@see #constructor(Type, Type,...)`
`@see #constructor(Type argname, Type argname,...)`

Referencing another class in the current or imported packages

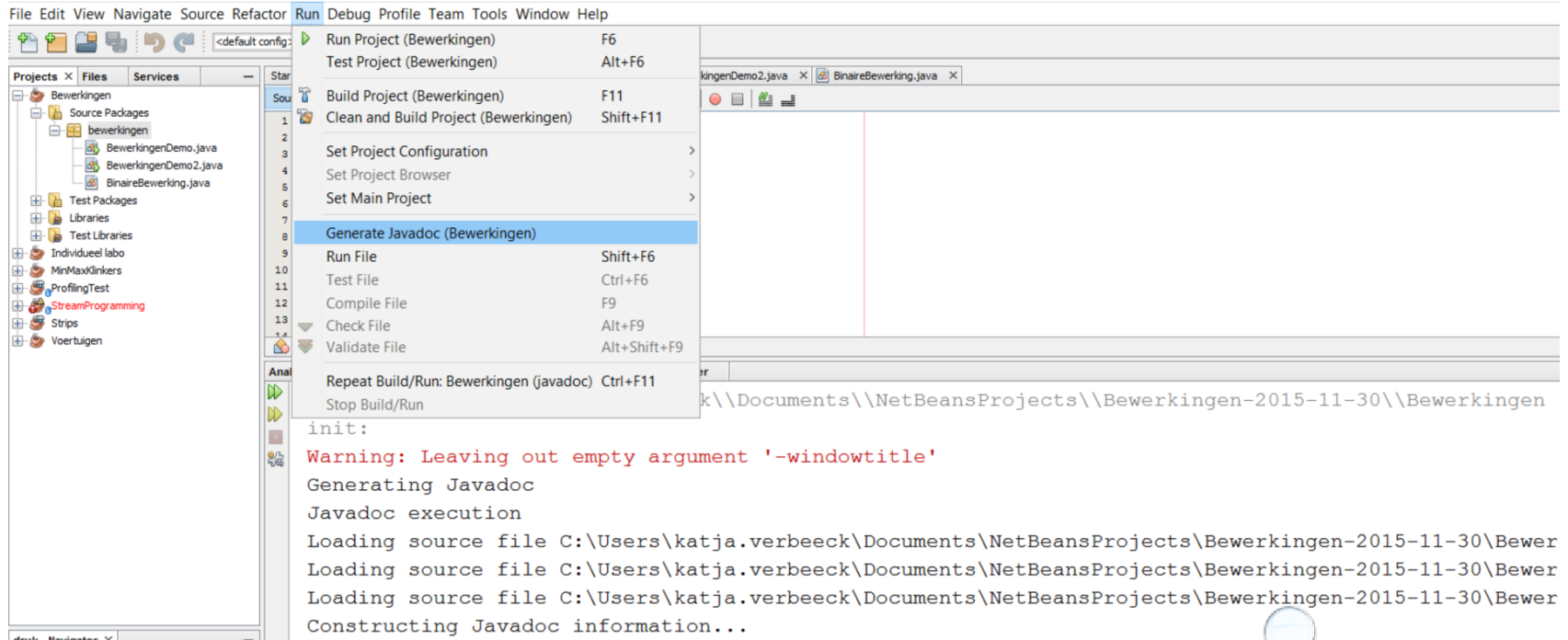
`@see Class#field`
`@see Class#method(Type, Type,...)`
`@see Class#method(Type argname, Type argname,...)`
`@see Class#constructor(Type, Type,...)`
`@see Class#constructor(Type argname, Type argname,...)`
`@see Class.NestedClass`
`@see Class`

Referencing an element in another package (fully qualified)

`@see package.Class#field`
`@see package.Class#method(Type, Type,...)`
`@see package.Class#method(Type argname, Type argname,...)`
`@see package.Class#constructor(Type, Type,...)`
`@see package.Class#constructor(Type argname, Type argname,...)`
`@see package.Class.NestedClass`
`@see package.Class`
`@see package`

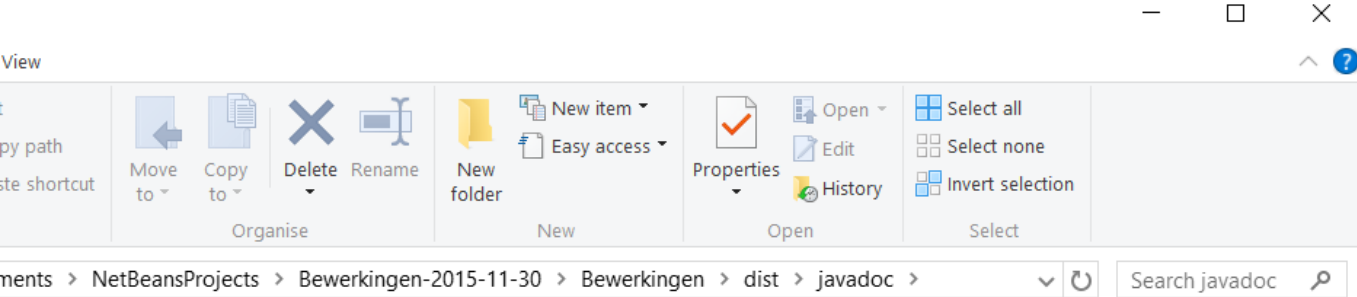
Generate javadoc

Bewerkingen - NetBeans IDE 8.2



The screenshot shows the NetBeans IDE 8.2 interface. The 'Run' menu is open, and 'Generate Javadoc (Bewerkingen)' is selected. The console output shows the following:

```
init:
Warning: Leaving out empty argument '-windowtitle'
Generating Javadoc
Javadoc execution
Loading source file C:\Users\katja.verbeeck\Documents\NetBeansProjects\Bewerkingen-2015-11-30\Bewer
Loading source file C:\Users\katja.verbeeck\Documents\NetBeansProjects\Bewerkingen-2015-11-30\Bewer
Loading source file C:\Users\katja.verbeeck\Documents\NetBeansProjects\Bewerkingen-2015-11-30\Bewer
Constructing Javadoc information...
```



Name

- bewerkingen
- index-files
- allclasses-frame.html
- allclasses-noframe.html
- constant-values.html
- deprecated-list.html
- help-doc.html
- index.html
- overview-tree.html
- package-list
- script.js
- stylesheet.css

De html files vind je terug in je directory dist van je Netbeans project

file:///C:/Users/katja.verbeeck/Documents/NetBeansProjects/Bewerkingen-2015-11-30/Bewerkingen/dist/javadoc/index.html

All Classes

- BewerkingenDemo
- BewerkingenDemo2
- BinaireBewerking

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

bewerkingen

Class BinaireBewerking

java.lang.Object
bewerkingen.BinaireBewerking

public class BinaireBewerking
extends java.lang.Object

Documenteren van de klasse

Version:
september 2014

Author:
Katja Verbeeck

Constructor Summary

Constructors

Constructor and Description
BinaireBewerking() default constructor methode
BinaireBewerking(int t1, int t2) constructor methode

Method Summary

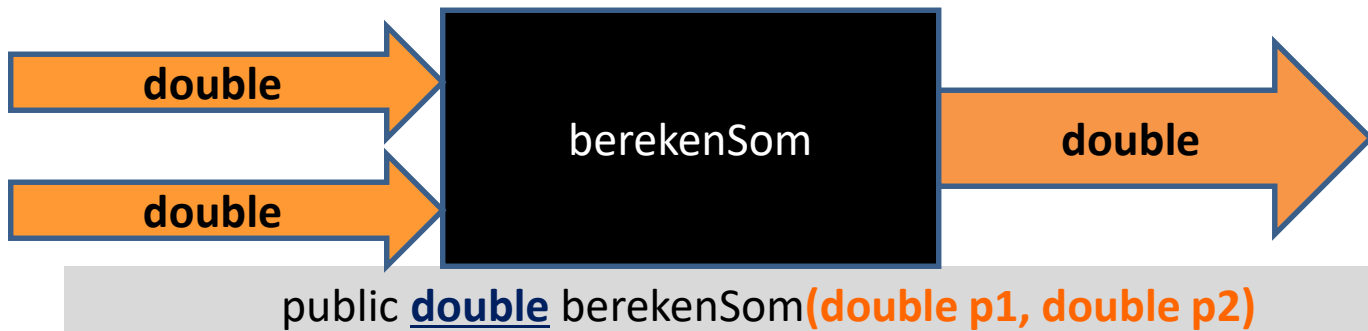
All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
int	berekenResultaat(char operator) Voert een gegeven rekenkundige bewerking uit op term1 en term2	

Method Overloading

In een programma kan je meerdere methode definities hebben met **dezelfde naam**, maar met een **andere parameterlijst**: → **Method Overloading**

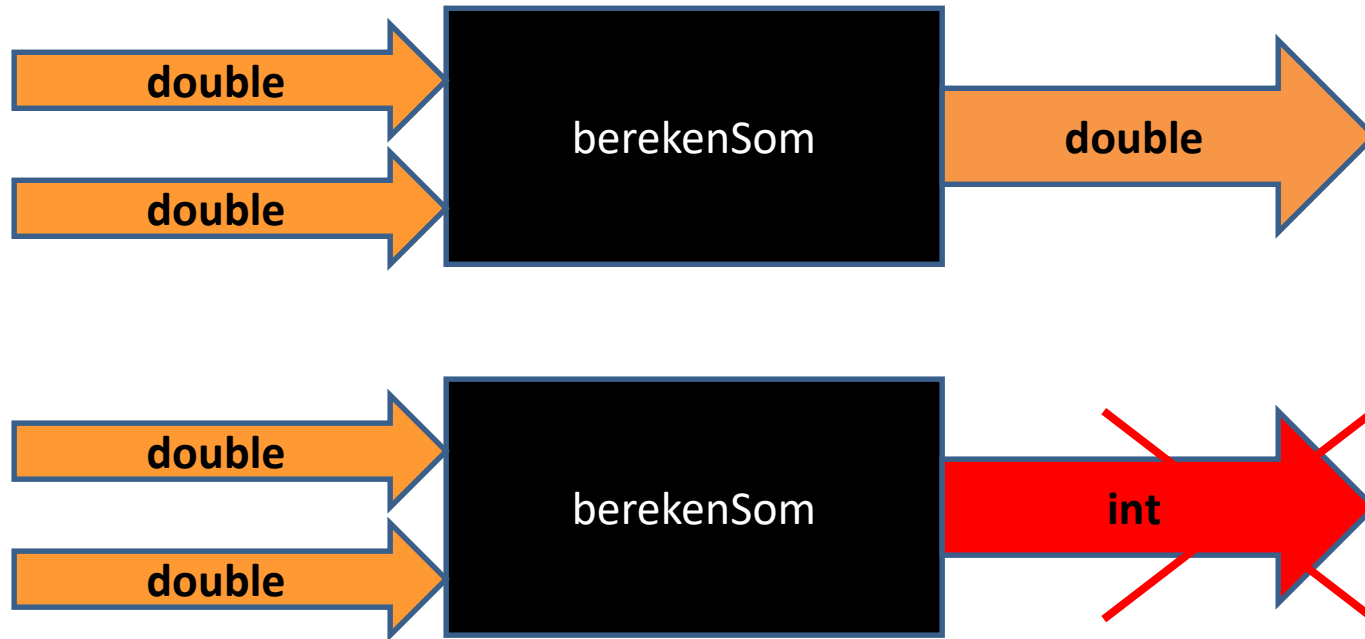


→ Minder of meer parameters



→ Parameter(s) van ander type

Opmerking: Meerdere methodes met **dezelfde parameterlijst**,
maar met een **verschillend return type** zijn niet toegestaan.



Opmerking: Enkel de naam van de formele parameters wijzigen kan ook niet...

```
public double berekenSom(double p1, double p2)  
public double berekenSom(double par1, double par2)
```

The code block is crossed out with a large red 'X'.

Voorbeeld van method overloading:

```
public double berekenMacht(double grondtal, int macht) {  
    return Math.pow(grondtal, macht);  
}  
  
public int berekenMacht(int grondtal, int macht) {  
    return (int)Math.pow(grondtal,macht);  
}
```

Afhankelijk van de actuele parameterwaarden wordt beslist welke methode er wordt opgeroepen:

```
int waarde = 25;  
double res = berekenMacht(waarde, 2);  
System.out.println("kwadraat = " + res);
```

→ De tweede methode wordt opgeroepen (de versie met 2 int-parameters)

Noot: Het resultaat daarvan (van type int) wordt hier toegekend aan een variabele van het type double (verbredende cast)

overloading in de Java API

bvb. in de klasse **String** vind je:

- 4 varianten voor de **indexOf** methode:
public int indexOf(**String** str)
public int indexOf(**String** str, **int** fromIndex)
public int indexOf(**int** ch)
public int indexOf(**int** ch, **int** fromIndex)
- 2 varianten voor de **lastIndexOf** methode
(zie Java API)
- 2 varianten voor de **split** methode
(zie Java API)
- 2 varianten voor de **startsWith** methode
(zie Java API)
- 9 varianten voor de **valueOf** methode
(zie Java API)

Sleutelwoorden

return-break-continue

```

public static void main(String[] args) {
    System.out.println("Geef getalwaarde: ");
    Scanner scan = new Scanner(System.in);
    int getal = scan.nextInt();

    for (int i = getal; i < Math.abs(getal*2); i++) {
        System.out.print(i);

        if (i < 0) {
            System.out.print(" ");
            continue;
        }
        else if (i > 0) {
            System.out.print("...");
            break;
        }
        else {
            System.out.print("---");
            return;
        }
    }

    System.out.println("!!!");
    scan.close();
}

```

Uitvoer bij ingave 5 ?

5...!!!

Uitvoer bij ingave -3 ?

-3 -2 -1 0---

Uitvoer bij ingave 0 ?

!!!

In dit laatste geval is de voorwaarde van de for loop niet voldaan en wordt de lus dus niet uitgevoerd

break in geneste lussen

```
public static void main(String[] args) {  
    for(int i=0; i< 3; i++){  
        System.out.println("Buitenste loop : waarde i : " + i);  
        System.out.print(" Binnenste loop : waarde t : ");  
  
        int t = 0;  
        while(t<100){  
            if (t==10) break;  
            System.out.print( t + " ")  
            t++;  
        }  
        System.out.println();  
    }  
    System.out.println("Einde van de loops");  
}
```

Buitenste loop : waarde i : 0
Binnenste loop : waarde t : 0 1 2 3 4 5 6 7 8 9
Buitenste loop : waarde i : 1
Binnenste loop : waarde t : 0 1 2 3 4 5 6 7 8 9
Buitenste loop : waarde i : 2
Binnenste loop : waarde t : 0 1 2 3 4 5 6 7 8 9
Einde van de loops

break in geneste lussen

```
public static void main(String[] args) {  
    one:  
    for(int i=0; i< 3; i++){  
        System.out.println("Buitenste loop : waarde i : ");  
        System.out.print(" Binnenste loop : waarde t : ");
```

```
        int t = 0;
```

```
        two:
```

```
        while(t<100) {  
            if (t==10) break one;  
            System.out.print( t + " ");  
            t++;  
        }
```

```
        System.out.println();  
    }
```

```
    System.out.println("Einde van de loops");
```

```
}
```

Buitenste loop : waarde i : 0

Binnenste loop : waarde t : 0 1 2 3 4 5 6 7 8 9 Einde van de loops

Parameter van de main methode

```
public static void main(String[] args) {  
    ➔ args.length  
    ➔ args[0], args[1], args[2], ...  
    ➔ conversie String naar primitief type:  
        * Integer.parseInt(String s)  
        * Double.parseDouble(String s)  
        * ...  
}
```

Parameter van de main methode

Herschrijf `DemoBewerkingen.java` zó dat het input krijgt via de `String[]` parameter van de main

Geef volgende uitvoer als het programma opgestart wordt zonder parameters:

```
Gebruik het programma als volgt:  
java.exe <klasse> <eerste getal> <operator> <...>  
  
Het programma zal de bewerking uitvoeren op de  
en het resultaat op het scherm drukken  
  
Druk op een toets om door te gaan. . .
```

Geef volgende uitvoer als het programma als volgt opgestart wordt:

java.exe Bewerkingen + 5 7

5 + 7 = 12

Demo run...

Parameters ingeven in Netbeans

NetBeans IDE 8.2

File Edit View Window Help Run Debug Profile Team Tools

Run Project (Bewerkingen) F6
Test Project (Bewerkingen) Alt+F6
Build Project (Bewerkingen) F11
Clean and Build Project (Bewerkingen) Shift+F11
Set Project Configuration > <default config>
Set Project Browser
Set Main Project
Generate Javadoc (Bewerkingen)
Run File Shift+F6
Test File Ctrl+F6
Compile File F9
Check File Alt+F9
Validate File Alt+Shift+F9
Repeat Build/Run: Bewerkingen (run) Ctrl+F11
Stop Build/Run

```
30  
31     if (args.length != 3) {  
32         System.out.println(" ");  
33         System.out.println(" ");  
34         System.out.println(" ");  
35         System.out.println(" ");  
36     }  
37     return;
```

Output - Bewerkingen (run) x

```
run:  
5 + 7 = 12  
BUILD SUCCESSFUL (total time: 0s)
```

Project Properties - Bewerkingen

Categories:

- Sources
- Libraries
- Build
 - Compiling
 - Packaging
 - Deployment
 - Documenting
- Run
- Application
 - Web Start
 - License Headers
 - Formatting
 - Hints

Configuration: <default config> New... Delete

Runtime Platform: Project Platform Manage Platforms...

Main Class: bewerkingen.BewerkingenDemo2 Browse...

Arguments: + 5 7

Working Directory: Browse...

VM Options: Customize...
(e.g. -Xms10m)

☐ Run with Java Web Start
(To run and debug the application with Java Web Start, first enable Java Web Start)

OK Cancel Help

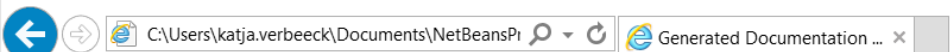
BewerkingenDemo2

```
public class BewerkingenDemo2 {  
    /**  
     * Het hoofdprogramma.  
     * De input voor het programma wordt als parameter meegegeven:  
     *  
     * @param args de input voor het programma :  
     * <ul>  
     * <li>Eerste woord is een operator  
     * <li>Tweede woord is een eerste getal  
     * <li>Derde woord is een tweede getal  
     * </ul>  
     *  
     * @see BinaireBewerking#berekenResultaat(char)  
     * @see #druk(java.lang.String)  
     *  
     */  
}
```

BewerkingenDemo2 : main

```
public static void main(String[] args) {  
    if (args.length != 3) {  
        System.out.println("Gebruik het programma als volgt: ");  
        System.out.println("java.exe <klasse>  
            <operator> <eerste getal> <tweede getal>");  
        System.out.println("Het programma zal de bewerking uitvoeren op  
            de twee getallen");  
        System.out.println("en het resultaat op het scherm drukken");  
        return;  
    }  
    char operator = args[0].charAt(0);  
    int getal1 = Integer.parseInt(args[1]);  
    int getal2 = Integer.parseInt(args[2]);  
  
    BinaireBewerking object = new BinaireBewerking(getal1, getal2);  
    int resultaat = object.berekenResultaat(operator);  
    druk(getal1 + " " + operator + " " + getal2 + " = " + resultaat);  
}
```

BewerkingenDemo2 : javadoc



C:\Users\katja.verbeeck\Documents\NetBeansPr... Generated Documentation ... x

All Classes

- BewerkingenDemo
- BewerkingenDemo2
- BinaireBewerking

BewerkingenDemo2

```
public BewerkingenDemo2 ()
```

Method Detail

main

```
public static void main(java.lang.String[] args)
```

Het hoofdprogramma. De input voor het programma wordt als parameter meegegeven:

Parameters:

args - de input voor het programma :

- Eerste woord is een operator
- Tweede woord is een eerste getal
- Derde woord is een tweede getal

See Also:

[BinaireBewerking.berekenResultaat\(char\), druk\(java.lang.String\)](#)

druk

```
public static void druk(java.lang.String info)
```

Drukt de gegeven info

Parameters:

info - de af te drukken informatie

See Also:

[PrintStream.println\(\)](#)

PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD