Les3

Excepties (bis)

Enumeraties (bis)

GUI toepassingen

Vorige les:

```
package presentatie;

import logica.*;

public class Console {
    psvm(…) {
            Auto a = new Auto(…);

            try {
                a.doeIets(…);
            } catch (IllegalArgumentException e) {
                S.o.p( e.getMessage() );
            }
    }
}
```

```
package logica;

class Auto {
    //private velden

    //constructoren

    //methoden
    public void doeIets(…) {
            throw new IllegalArgumentException("error info");
    }
}
```

Excepties (bis)

Unchecked excepties

Checked excepties

Beslis zelf of je een **unchecked exception** wil opvangen

```
try {
    String s = null;
    char c = s.charAt(0);
} catch (NullPointerException e) {
        System.err.println("ERROR – Null referentie !! ");
}
```

```
try {
    int[] rij = new int[3];
    rij[5] = 1;
} catch (IndexOutOfBoundsException e) {
        System.err.println("ERROR - Index out of bound !! ");
}
```

```
try {
    java.math.BigInteger bi = new BigInteger("100");
    java.lang.Number number = (Number) bi;
    java.math.BigDecimal bd = (BigDecimal) number;
} catch (ClassCastException e) {
        System.err.println("ERROR - Class cast !! ");
}
```

Heeft te maken met overerving, zie later

```java
    public static void main(String[] args) {
        String s = null;
        char c = s.charAt(0);
    }
}
```

ut × Test Results

Debugger Console × 2019_CodeTheorieJavaOO (run) ×

```
run:
Exception in thread "main" java.lang.NullPointerException
        at _2016_les3_UncheckedExceptions.NullPointerExceptionVoorbeeldA.main(NullPointerExceptionVoorbeeldA.java:10
C:\Users\kristien.vanassche\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 2 seconds)
```

```java
    public static void main(String[] args) {
        try {
            String s = null;
            char c = s.charAt(0);
        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }
        System.out.println("Code gaat hier verder");
    }
}
```

ut ×

Debugger Console × 2019_CodeTheorieJavaOO (run) ×

```
run:
Error: null
Code gaat hier verder
BUILD SUCCESSFUL (total time: 1 second)
```

Soorten excepties

"Unchecked" excepties

java.lang
Class NullPointerException

java.lang
Class IndexOutOfBoundsException

java.lang
Class ClassCastException

java.lang
Class NumberFormatException

java.lang
Class IllegalArgumentException

java.util
Class InputMismatchException

Afhandelen MOET NIET (maar mag altijd)

"Checked" excepties

java.io
Class IOException

java.io
Class FileNotFoundException

java.net
Class UnknownHostException

java.lang
Class InterruptedException

Afhandelen MOET!

java.lang

Class Integer

Integer (Java Platform SE 6)

parseInt

```
public static int parseInt(String s)
                    throws NumberFormatException
```

Parses the string argument as a signed decimal integer. The characters in the string must
negative value. The resulting integer value is returned, exactly as if the argument and the

Parameters:

`s` - a `String` containing the `int` representation to be parsed

Returns:

the integer value represented by the argument in decimal.

Throws:

`NumberFormatException` - if the string does not contain a parsable integer.

Unchecked exceptions moet je niet op te vangen

```
public void testParse() {
	int i = Integer.parseInt("Hallo");
}
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "Hallo"
		at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)
		at java.lang.Integer.parseInt(Integer.java:449)
		at presentatie.Console.testParse(Console.java:24)
		at presentatie.Console.main(Console.java:16)
```

Unchecked exceptions kan je best toch opvangen

```
public void testParse() {
        try {
                int i = Integer.parseInt("Hallo");
        }
        catch (NumberFormatException e) {
                System.err.println("ERROR - Number format !! ");
        }

        //code gaat hier verder
}
```

...en naar wens afhandelen

```
public void parse3() {
    boolean ok = false;
    Scanner sc = new Scanner(System.in);
    int getal = -1;

    while (!ok) {
      try {
        System.out.println("Geef geheel getal: ");
        getal = Integer.parseInt(sc.next());
        ok = true;
      } catch (NumberFormatException e) {
        System.err.println("ERROR - Number format !! ");
      }
    }

    //input ok => code gaat hier verder
    System.out.println("Verwerk getal " + getal);
  }
```

```
Geef geheel getal: abc
ERROR – Number format!!
Geef geheel getal: XYZ
ERROR – Number format!!
Geef geheel getal: 123
Verwerk getal 123
```

Scanner-methoden: close(), next() en nextInt()

java.util

Class Scanner

next

`public String next()`

Finds and returns the next complete token from this scanner. A complete token is preceded and followed by input that matches the delimiter pattern. This method may block while waiting for input to scan, even if a previous invocation of `hasNext()` returned `true`.

Specified by:

`next` in interface `Iterator<String>`

Returns:

the next token

Throws:

- `NoSuchElementException` - if no more tokens are available
- `IllegalStateException` - if this scanner is closed

See Also:

`Iterator`

close

`public void close()`

Closes this scanner.

If this scanner has not yet been closed then if its underlying `readable` also implements the `Closeable` interface then the readable's `close` method will be invoked. If this scanner is already closed then invoking this method will have no effect.

Attempting to perform search operations after a scanner has been closed will result in an `IllegalStateException`.

Specified by:

`close` in interface `Closeable`

Specified by:

`close` in interface `AutoCloseable`

nextInt

`public int nextInt()`

Scans the next token of the input as an `int`.

An invocation of this method of the form `nextInt()` behaves in exactly the same way as the invocation `nextInt(radix)`, where `radix` is the default radix of this scanner.

Returns:

the `int` scanned from the input

Throws:

- `InputMismatchException` - if the next token does not match the *Integer* regular expression, or is out of range
- `NoSuchElementException` - if input is exhausted
- `IllegalStateException` - if this scanner is closed

…naar wens afhandelen

```
public void parse3() {
    boolean ok = false;
    Scanner sc = new Scanner(System.in);
    int getal = -1;

    while (!ok) {
        try {
            System.out.println("Geef geheel getal: ");
            getal = sc.nextInt();
            ok = true;
        } catch (InputMismatchException e) {
            System.err.println("ERROR – Input mismatch !! ");
        }
    }

    //input ok => code gaat hier verder
    System.out.println("Verwerk getal " + getal);
  }
```

```
Geef geheel getal: abc
ERROR – Input mismatch!!
Geef geheel getal: XYZ
ERROR – Input mismatch!!
Geef geheel getal: 123
Verwerk getal
```

Alle fouten algemeen opvangen en verder gaan

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int waarde = -1;

        try {
            waarde = vraagGetal("schoenmaat", sc);
        } catch (Exception e) {
            System.out.println("ERROR“ + e.getMessage() );
        }

        System.out.println("waarde is: " + waarde);
}
```

```
private static int vraagGetal(String info, Scanner sc) {
    System.out.println("Geef " + info);
    return sc.nextInt();
}
```

Wat als gebruiker "**Hallo**" intikt ?

```
Geef je schoenmaat
hallo
ERROR: null
```

De diverse fouten specifiek opvangen en verder gaan

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int waarde = -1;

        try {
            waarde = vraagGetal("schoenmaat", sc);
        } catch (InputMismatchException e) {
            System.out.println("ERROR: input mismatch");
        } catch (NoSuchElementException e) {
            System.out.println("ERROR: no such element");
        } catch (IllegalStateException e) {
            System.out.println("ERROR: illegal state");
        }

        System.out.println("waarde is: " + waarde);
}
```

```
private static int vraagGetal(String info, Scanner sc) {
    System.out.println("Geef " + info);
    return sc.nextInt();
}
```

Wat als gebruiker "**Hallo**" intikt ?

```
Geef schoenmaat
hallo
ERROR: input mismatch
```

Fout opvangen en niet verder gaan

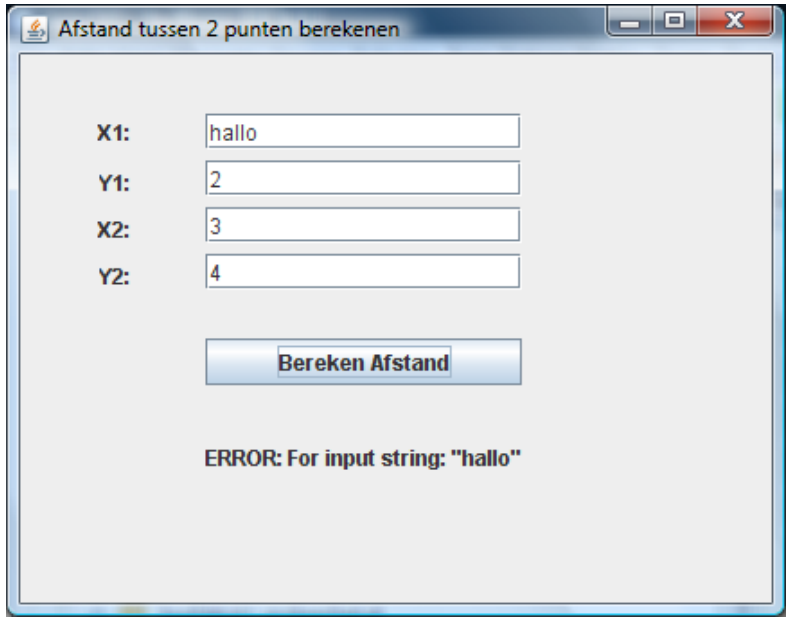
```
public Adres(String straat, int nr, int postcode, String gemeente) {
        if (straat == null || gemeente == null || nr < 0 || postcode <= 999 || postcode >= 9999) {
                throw new IllegalArgumentException("ongeldig adres");
        }
        …
}
```

```
public void test() {
        Scanner sc = new Scanner(System.in);

        try {
                String straat = sc.nextLine();
                int nr  = sc.nextInt();
                int postcode = sc.nextInt();
                String gemeente = sc.nextLine();

                Adres a = new Adres(straat, nr, postcode, gemeente);

                //Ga enkel verder als ALLE INPUT OK

        } catch (Exception e) {
                System.out.println("ERROR: " + e.getMessage() );
        }
}
```

Help de gebruiker om fouten te vermijden (vb. Console)

```
public void test() {
        …
        int nr = -1;
        System.out.println("Geef straatnummer:");
        do {
            try {
                nr  = sc.nextInt();
            } catch (Exception e) {…}
        } while (nr < 0);

        int postcode = -1;
        System.out.println("Geef postcode:");
        do {
            try {
                postcode  = sc.nextInt();
            }  catch (Exception e) {…}
        } while (postcode < 999 || postcode > 9999);
        …
}
```

Help de gebruiker om fouten te vermijden (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 = Math.sqrt(Math.pow(x2 – x1, 2) + Math.pow(y2 – y1, 2));
        this.jLabelOutput.setText("Afstand: " + afstand );
    } catch (Exception e) {
        this.jLabelOutput.setText("ERROR: " + e.getMessage() );
    }
}
```

Exceptie opwerpen (typisch in logische klasse)

```
//in constructor
public Persoon(String naam, String voornaam, int leeftijd, boolean geslacht) {
        if (leeftijd < 0) {
            throw new IllegalArgumentException("negatieve leeftijd is niet mogelijk");
        }

        this.naam = naam;
        this.voornaam = voornaam;
        this.leeftijd = leeftijd;
        this.geslacht = geslacht;

}
```

```
//in setter
public void setPostcode(int postcode) {
        if (postcode > 999 && postcode <= 9999) {
            this.postcode = postcode;
        }
        else {
            throw new IllegalArgumentException("Ongeldige postcode");
        }
    }
```

Excepties opvangen (typisch in presentatie klasse)

```
public void test() {
   try {
      System.out.println("Geef leeftijd");
      int leeftijd = sc.nextInt();
      Persoon persoon = new Persoon("Florimont", "Rosa", leeftijd, true);
      …
      System.out.println("Geef postcode");
      int code= sc.nextInt();
      adres.setPostcode(code);
  }
  catch (Exception e) {
      System.out.println("Oei, een " + e.getMessage() );
  }
}
```

Mogelijke foutmelding op scherm:

- Oei, een **java.util.InputMismatchException** (gegenereerd door sc.nextInt())
- Oei, een **java.lang.IllegalArgumentException: negatieve leeftijd is niet mogelijk** (zie eerdere dia)
- Oei, een **java.lang.IllegalArgumentException: Ongeldige postcode** (zie eerdere dia)

Checked exceptions

java.io

Class IOException

java.io

Class FileNotFoundException

java.net

Class UnknownHostException

java.lang

Class InterruptedException

Let op: de algemene Exception klasse is een "checked" exception

Algemeen Exception-object genereren

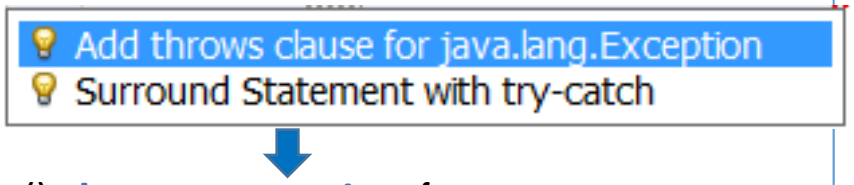
```
public class Data {
    private String data;

    public Data(String data) {
        this.data = data;
    }

    public boolean analyseer() {
        if (data == null || data.equals("")) {
            throw new Exception("Er kon geen data gevonden worden");
        }

        return true;
    }
}
```

```
unreported exception Exception; must be caught or declared to be thrown
----
(Alt-Enter shows hints)
```

```
public class AlgemeneExceptieOpwerpen {
    public static void main(String[] args) {
        boolean res = new Data("").analyseer();
    }
}
```

Algemene exception afhandelen

```
public class Data {
   private String data;

   public Data(String data) {
      this.data = data;
   }

   public boolean analyseer() throws Exception {
      if(data == null || data.equals("")) {
         throw new Exception("Er kon geen data gevonden worden");
      }

      return true;
   }
}
```




```
public class AlgemeneExceptieOpwerpen {
   public static void main(String[] args) {
      boolean res = new Data("").analyseer();
   }
}
```

Algemene exception afhandelen

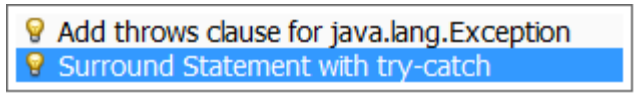
```
public class Data {
   private String data;

   public Data(String data) {
      this.data = data;
   }

   public boolean analyseer() throws Exception {
      if(data == null || data.equals("")) {
         throw new Exception("Er kon geen data gevonden worden");
      }

      return true;
   }
}
```

```
public class AlgemeneExceptieOpwerpen {
   public static void main(String[] args) {
      boolean res = new Data("").analyseer();
   }
}
```

unreported exception Exception; must be caught or declared to be thrown
---
(Alt-Enter shows hints)

Algemene exception afhandelen

```
public class Data {
   private String data;

   public Data(String data) {
      this.data = data;
   }

   public boolean analyseer() throws Exception {
      if(data == null || data.equals("")) {
         throw new Exception("Er kon geen data gevonden worden");
      }

      return true;
   }
}
```




```
public static void main(String[] args) {
      try {
         boolean res = new Data("").analyseer();
      } catch (Exception ex) {
         System.err.println(ex.getMessage());
      }
   }
}
```

Checked exceptions MOET je afhandelen

TextFile

read

```
public static java.lang.String read(java.lang.String fileName)
                                throws java.io.IOException
```

Leest alle lijnen van het opgegeven bestand. Bij leesproblemen wordt een IOException opgegooid.

Parameters:

`fileName` - de (absolute of relatieve) locatie van het bestand

Returns:

de tekst van het gelezen tekstbestand (alle lijnen als 1 grote string)

Throws:

`java.io.IOException`

Zoniet compiler error: unreported exception java.io.IOException;

must be caught or **declared to be thrown**

Must be caught…

```
public String testLees() {
        String content = "";

        try {
                content = TextFile.read("adressen.txt");
         }
         catch (IOException e) {
                System.err.println("ERROR – IO !! ");
         }

        return content;
}
```

… or declared to be thrown

```
public static String testLees() throws java.io.IOException {
        return TextFile.read("adressen.txt");
}
```

Noot: Je kan zo de afhandeling van een exceptie naar oproepende methode brengen, of naar diens oproepende methode. In extremis handelt geen enkele methode de exceptie af (niet aan te raden)

```
public static void start() throws java.io.IOException {
        String content = testLees();
}
```

```
public static void main(String[] args) throws java.io.IOException {
        start();
}
```

Voorbeeld: Thread.sleep(…)

```
unreported exception InterruptedException; must be caught or declared to be thrown

Thread.sleep called in loop
----
(Alt-Enter shows hints)
```

```
Thread.sleep(1000);
```

Thread.sleep(…)

java.lang

Class Thread

sleep

```
public static void sleep(long millis)
                  throws InterruptedException
```

Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers. The thread does not lose ownership of any monitors.

Parameters:

`millis` - the length of time to sleep in milliseconds

Throws:

unchecked exception → `IllegalArgumentException` - if the value of `millis` is negative

checked exception → `InterruptedException` - if any thread has interrupted the current thread. The *interrupted status* of the current thread is cleared when this exception is thrown.

Toepassing met Thread.sleep

13 auto's staan in de file. Laat **elke seconde** een auto wegrijden.

```
public void start() throws InterruptedException {
        Auto[] autos = new Auto[13];

        for (int i = 0; i < autos.length; i++) {
                autos[i] = new Auto();
        }

        for (Auto auto : autos) {
            Thread.sleep(1000);
            auto.rij();
            System.out.println("auto rijdt weg");
        }
    }
```

```
try {
        scen.start();
} catch (InterruptedException e) {
        System.out.println("FOUT: " + e.getMessage() );
}
```

Netbeans Demo: _2016_les3.MijnAutoVerkeerslichtConsoleProject

BESLUIT: Soorten excepties

Unchecked excepties

java.lang
Class NullPointerException

java.lang
Class IndexOutOfBoundsException

java.lang
Class ClassCastException

java.lang
Class NumberFormatException

java.lang
Class IllegalArgumentException

java.util
Class InputMismatchException

Checked excepties

java.io
Class IOException

java.io
Class FileNotFoundException

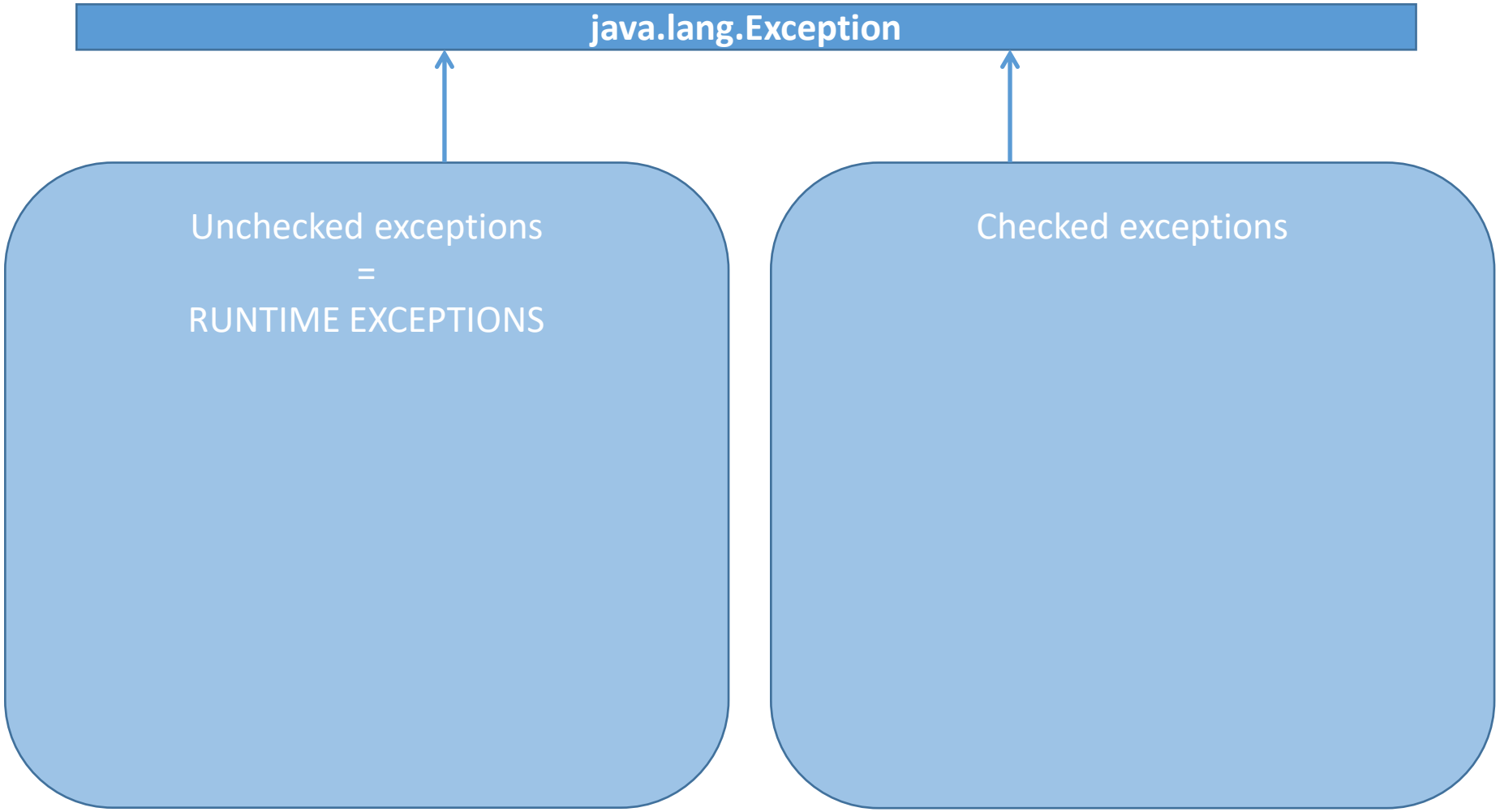
java.net
Class UnknownHostException

java.lang
Class InterruptedException

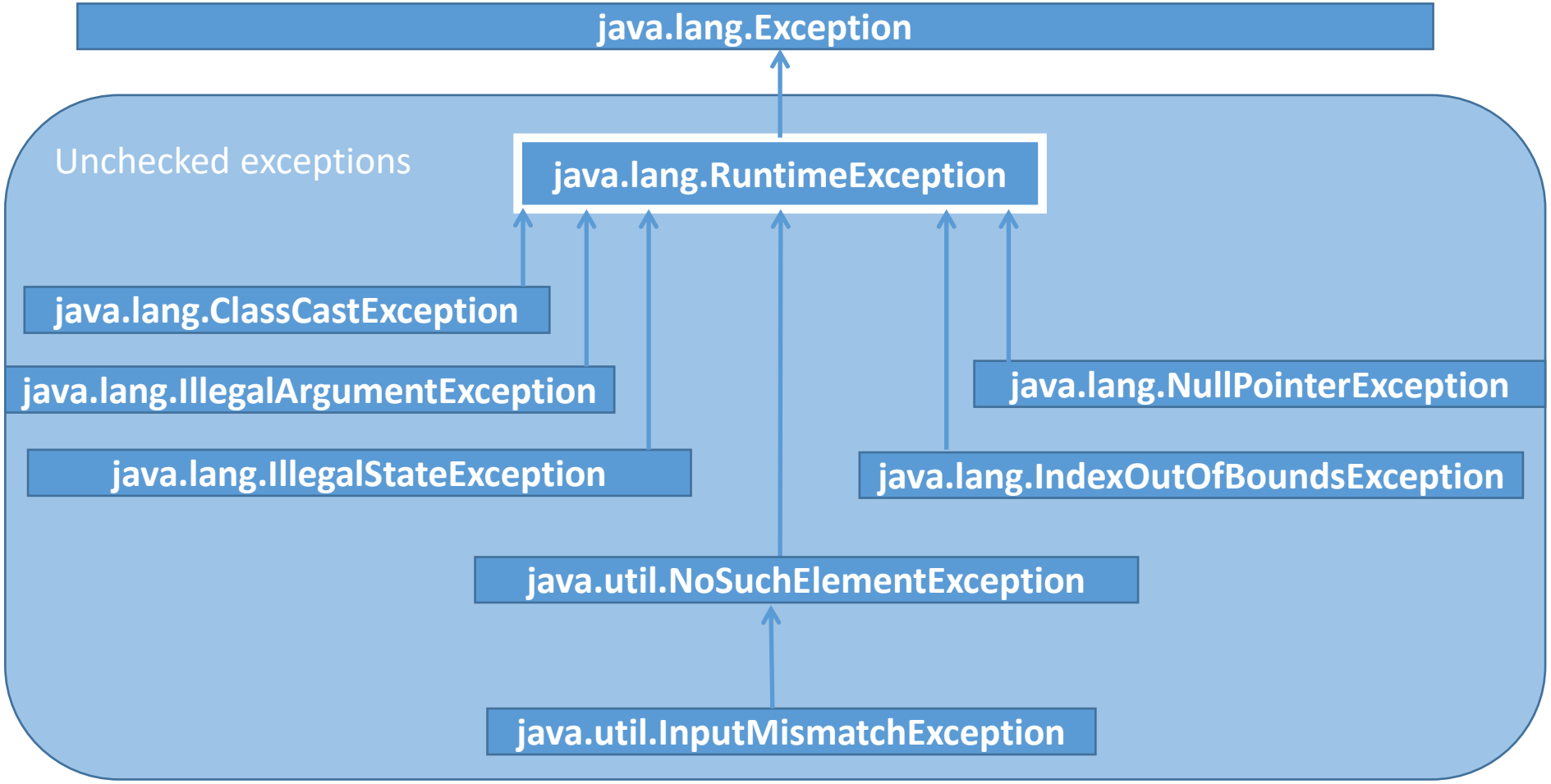
Runtime exceptions
➔ Not required to check

Alle andere exceptions
➔ Required to check
➔ **try/catch** blok óf **throws** clausule in declaratie van oproepende code!

Hiërarchie van excepties



Hiërarchie van excepties
(zie verder Hoofdstuk overerving)



Enumeraties (bis)

Enumeraties

Gealloceerd in statische ruimte

```
public enum Geslacht {
    MAN, VROUW
}
```

```
public enum Getalstelsel {
    DECIMAAL, BINAIR , HEXADECIMAAL , OCTAAL
}
```

```
public enum Weekdag {
    MAANDAG, DINSDAG, WOENSDAG, DONDERDAG, VRIJDAG
}
```

```
public enum Kleur {
    ZWART, BRUIN, PAARS
}
```

LocalDate.of

https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html#of-int-java.time.Month-int-

Output - 2019_CodeTheorieJavaOO (run) | Javadoc

java.time.LocalDate

```
public static LocalDate of(int year, Month month, int dayOfMonth)
```

Obtains an instance of LocalDate from a year, month and day.

This returns a LocalDate with the specified year, month and day-of-month. The day must be valid for the year and month, otherwise an exception will be thrown.

Parameters:

- year - the year to represent, from MIN_YEAR to MAX_YEAR
- month - the month-of-year to represent, not null
- dayOfMonth - the day-of-month to represent, from 1 to 31

Returns:

the local date, not null

Throws:

DateTimeException - if the value of any field is out of range, or if the day-of-month is invalid for the month-year

```java
import java.time.LocalDate;
import java.time.Month;

public class DemoLocalDate {
    public static void main(String[] args) {
        LocalDate valentine = LocalDate.of(2019, Month.FEBRUARY, 14);
        System.out.println("Valentine is " + valentine);

        LocalDate today = LocalDate.now();
        System.out.println("Today is " + today);
        LocalDate nextMonth = LocalDate.of(2019, Month.JANUARY, today.getDayOfMonth()+20);
        System.out.println(nextMonth);
    }
}
```

ut ✕ | Search Results

Debugger Console ✕ | Debugger Console ✕ | 2019_CodeTheorieJavaOO (run) ✕

```
run:
Valentine is 2019-02-14
Today is 2019-02-24
Exception in thread "main" java.time.DateTimeException: Invalid value for DayOfMonth (valid values 1 - 28/31): 44
        at java.time.temporal.ValueRange.checkValidValue(ValueRange.java:311)
        at java.time.temporal.ChronoField.checkValidValue(ChronoField.java:703)
        at java.time.LocalDate.of(LocalDate.java:248)
        at _2016_les1_LocalDate.DemoLocalDate.main(DemoLocalDate.java:13)
C:\Users\kristien.vanassche\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 2 seconds)
```

Month.of

```java
private String toMaand(int i) {
    return Month.of(i + 1).name().substring(0, 3) + "'" + (this.JAAR % 100);
}
```

```
Month.
APRIL                                   Month
AUGUST                                  Month
DECEMBER                                Month
FEBRUARY                                Month
JANUARY                                 Month
JULY                                    Month
JUNE                                    Month
MARCH                                   Month
MAY                                     Month
NOVEMBER                                Month
OCTOBER                                 Month
SEPTEMBER                               Month
from(TemporalAccessor temporal)         Month
of(int month)                           Month
valueOf(String arg0)                    Month
valueOf(Class<T> enumType, String name) T
values()                                Month[]
```

Statische methode

of

`public static Month of(int month)`

Obtains an instance of `Month` from an `int` value.

`Month` is an enum representing the 12 months of the year. This factory allows the enum to be obtained from the `int` value. The `int` value follows the ISO-8601 standard, from 1 (January) to 12 (December).

Parameters:

`month` - the `month`-of-year to represent, from 1 (January) to 12 (December)

Returns:

the `month`-of-year, not null

Throws:

`DateTimeException` - if the `month`-of-year is invalid

Niet-statische methode

name

`public final String name()`

Returns the name of this enum constant, exactly as declared in its enum declaration. **Most programmers should use the `toString()` method in preference to this one, as the toString method may return a more user-friendly name.** This method is designed primarily for use in specialized situations where correctness depends on getting the exact name, which will not vary from release to release.

Returns:

the name of this enum constant

```java
public class Getal {
    private int waarde;

    public Getal(int i) {
        this.waarde = i;
    }

    public String vertaal(Getalstelsel stelsel) {
        switch(stelsel) {
            case BINAIR:
                return Integer.toBinaryString(waarde);
            case DECIMAAL:
                return Integer.toString(waarde);
            case HEXADECIMAAL:
                return Integer.toHexString(waarde);
            case OCTAAL:
                return Integer.toOctalString(waarde);
            default:
                return "";
        }
    }
}
```

```java
public enum Getalstelsel {
    DECIMAAL, BINAIR , HEXADECIMAAL , OCTAAL
}
```

```java
public class Console {
    public static void main(String[] args) {
        Getal getal = new Getal(123);
        System.out.println(getal.vertaal(Getalstelsel.BINAIR));
        System.out.println(getal.vertaal(Getalstelsel.DECIMAAL));
        System.out.println(getal.vertaal(Getalstelsel.HEXADECIMAAL));
        System.out.println(getal.vertaal(Getalstelsel.OCTAAL));
    }
}
```

```
1111011
123
7b
173
```

Gebruik enumeratie in UML-Klassendiagramma


```
public enum Geslacht {
    MAN, VROUW
}
```

```
public class Persoon {
    private String naam;
    private String voornaam;
    private int leeftijd;
    private Geslacht geslacht;

    public Persoon() {
        this.naam = ""
        this.voornaam = "";
    }

     public Persoon(...) {
        ...
    }

    public String getVolledigeNaam() {
        return this.naam + " " + this.voornaam;
    }

    public int getLeeftijd() {
        return this.leeftijd;
    }

    public Geslacht getGeslacht() {
        return this.geslacht;
    }
}
```

```
//in Console testprogramma:
Persoon p = new Persoon();
 System.out.println(p.getVolledigeNaam() + ";" + p.getLeeftijd() +  ";" + p.getGeslacht());
//Wat is de uitvoer ?
```

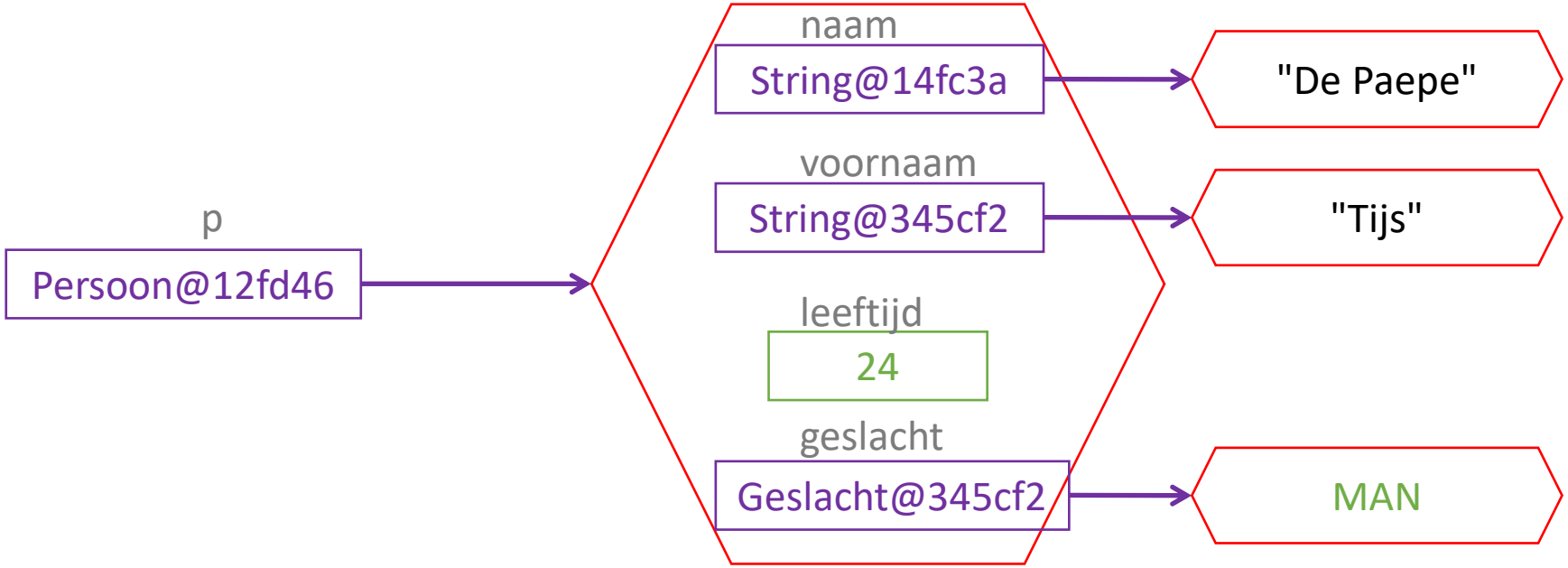
;0;null

Name	Type	Value
naam	String	""
voornaam	String	""
leeftijd	int	0
geslacht		null

```
//in Console testprogramma:
Persoon p = new Persoon("De Paepe", "Tijs", 24, Geslacht.MAN);
 System.out.println(p.getVolledigeNaam() + ";" + p.getLeeftijd() +  ";" + p.getGeslacht());
//Wat is de uitvoer ?
```

naam	String	"De Paepe"
voornaam	String	"Tijs"
leeftijd	int	24
geslacht	Geslacht	"MAN"

De Paepe Tijs;24;MAN



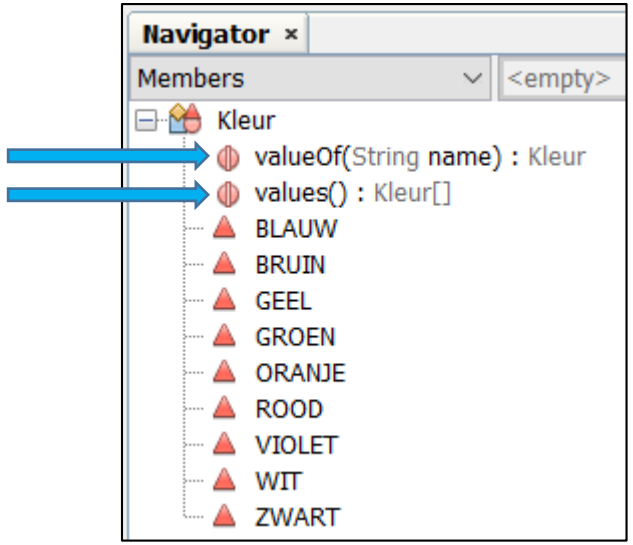
Enumeraties in aparte package onderbrengen

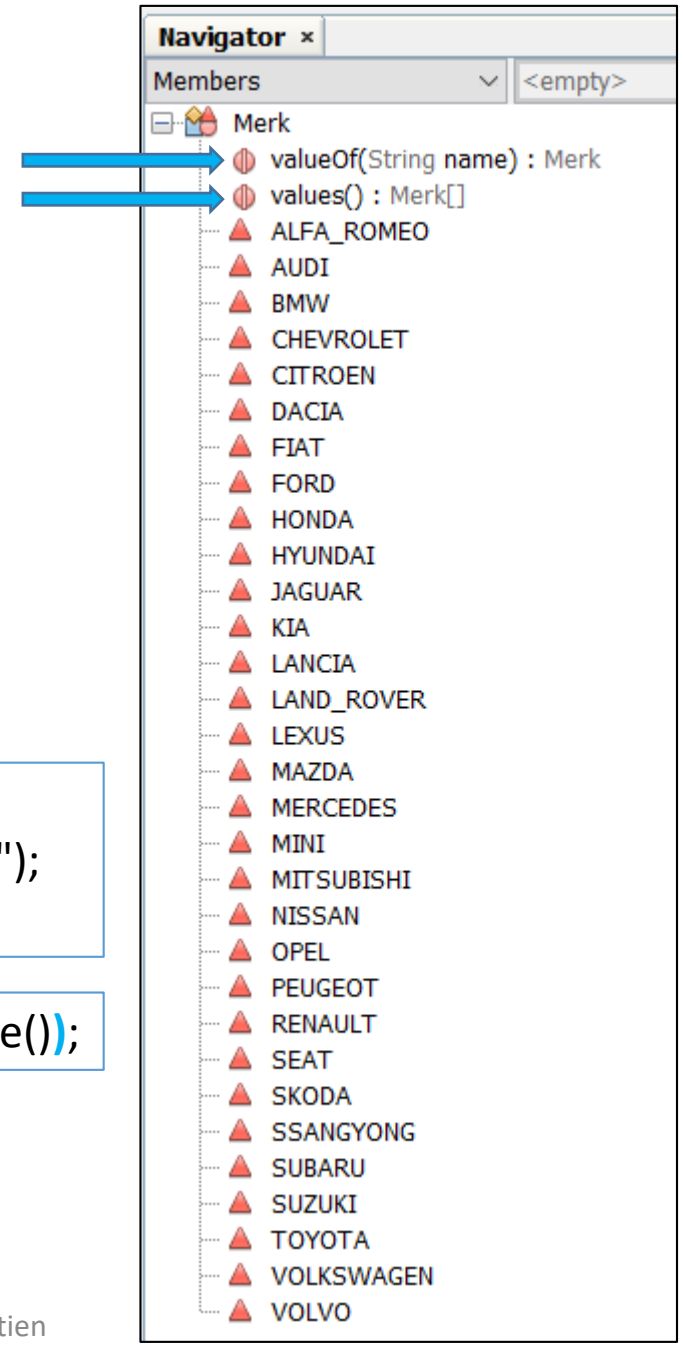
Kleur.java

```
package logica.enumeraties;

public enum Kleur {
    ROOD, WIT, ZWART , GEEL , BLAUW, GROEN , BRUIN , ORANJE, VIOLET
}
```

Merk.java

```
package logica.enumeraties;

public enum Merk {
    ALFA_ROMEO, AUDI, BMW, CHEVROLET, CITROEN, DACIA, FIAT, FORD, HONDA,
    HYUNDAI, JAGUAR, KIA, LANCIA, LAND_ROVER, LEXUS, MAZDA, MERCEDES, MINI,
    MITSUBISHI, NISSAN, OPEL, PEUGEOT, RENAULT, SEAT, SKODA, SSANGYONG,
    SUBARU, SUZUKI, TOYOTA, VOLKSWAGEN, VOLVO
}
```






```
for (Kleur k : Kleur.values()) {
        System.out.print(k + " ");
}
```

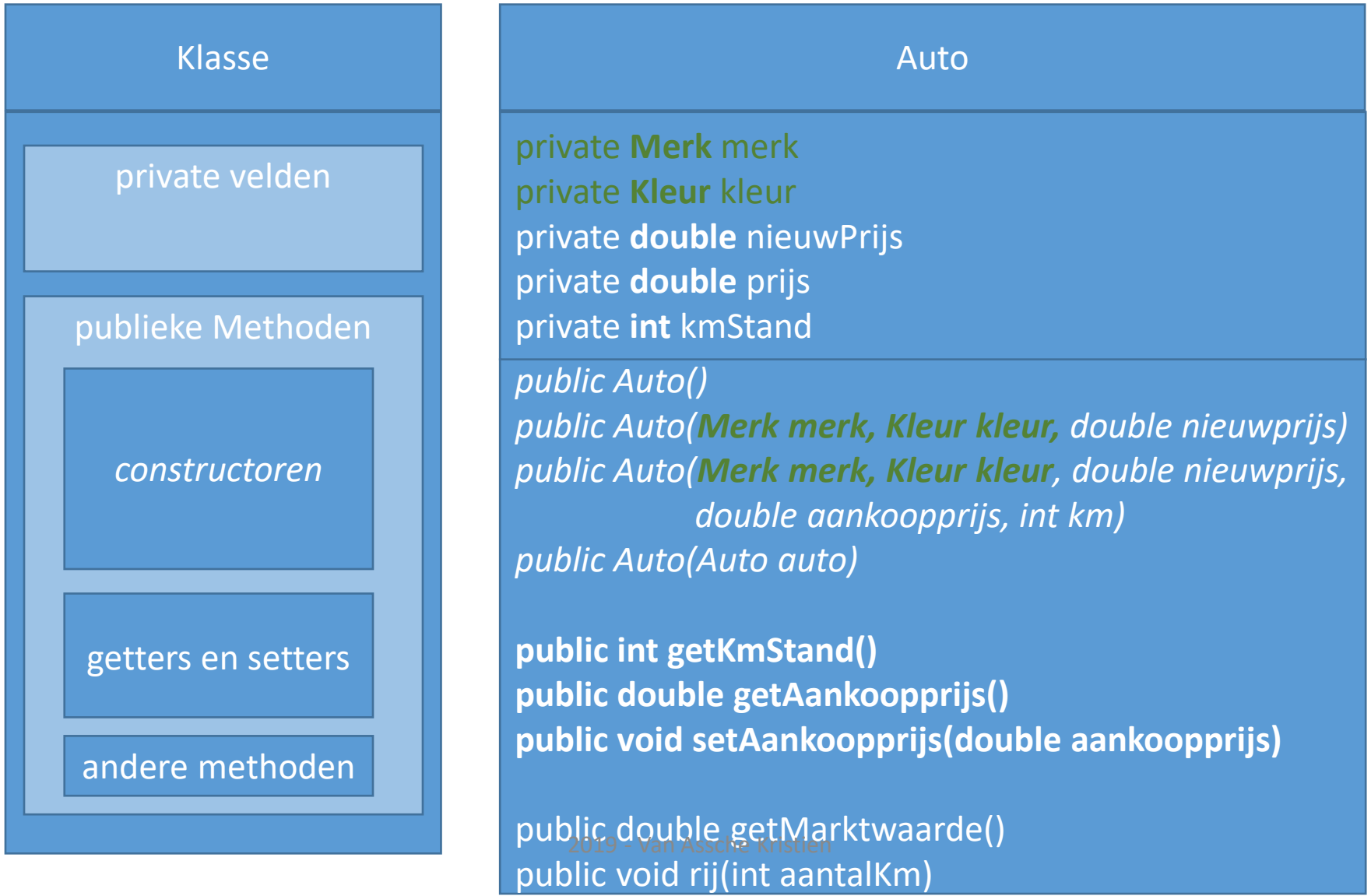
```
Kleur kleur = Kleur.valueOf(sc.nextLine().toUpperCase());
```

Input van de gebruiker

```
Geef merk:
lada
FOUT:No enum constant logica.enumeraties.Merk.LADA

Geef kleur (ROOD WIT ZWART GEEL BLAUW GROEN BRUIN ORANJE VIOLET):
rood

Geef nieuwprijs:
20000

Geef aankoopprijs:
15000

Geef kmStand:
25000
```

```
public Auto kiesJeAuto() {
    Auto a = null;
    Merk merk = null;
    try {
        System.out.println("Geef merk: ");
        merk = Merk.valueOf(sc.nextLine().toUpperCase());
    } catch (Exception e) {
        System.out.println("FOUT:" + e.getMessage());
    }

    try {
        System.out.print("Geef kleur ( ");
        for (Kleur k : Kleur.values()) {
            System.out.print(k + " ");
        }
        System.out.println("): ");
        Kleur kleur = Kleur.valueOf(sc.nextLine().toUpperCase());

        System.out.println("Geef nieuwprijs: ");
        double prijs = sc.nextDouble();
        …

        a = new Auto(merk, kleur, prijs, aankoopprijs, kmStand);
    } catch (Exception e) {
        System.out.println("FOUT:" + e.getMessage());
    }

    return a;
}
```

Auto-voorbeeld met enumeraties



Constructor overloading

```
public Auto(Merk merk, Kleur kleur, double nieuwprijs, double aankoopprijs, int km) {
        if (this.nieuwprijs < 0 || aankoopprijs < 0) {
            throw new IllegalArgumentException("prijzen kunnen niet negatief zijn");
        }
        else if (aankoopprijs > nieuwprijs) {
            throw new IllegalArgumentException(
                                "aankoopprijs kan niet groter zijn dan nieuwwaarde");
        }
        else if (km < 0) {
            throw new IllegalArgumentException("kilometerstand kan niet negatief zijn");
        }

        this.merk = merk;
        this.kleur = kleur;
        this.nieuwprijs = nieuwprijs;
        this.aankoopprijs = aankoopprijs;
        this.kmStand = km;
}
```

```
public Auto(Auto a) {
    this.merk = a.merk;
    this.kleur = a.kleur;
    this.nieuwprijs = a.nieuwprijs;
    this.aankoopprijs = a.aankoopprijs;
    this.kmStand = a.kmStand;
}
```

```
public Auto(Merk merk, Kleur kleur, double nieuwprijs) {
        this(merk, kleur, nieuwprijs, nieuwprijs, 0);
}
```

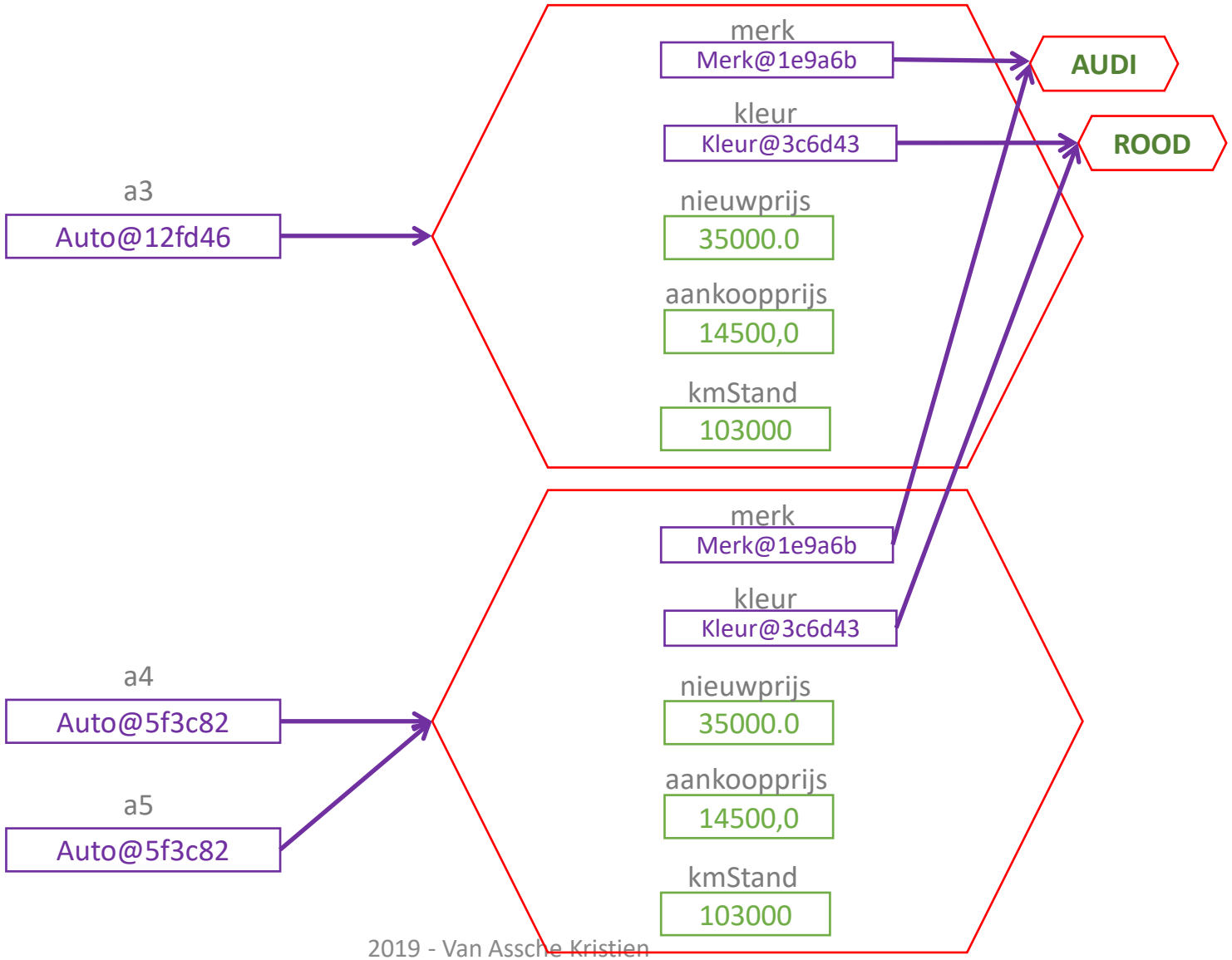
Cf. Oefening Auto

```
public class Console {
    public static void main(String[] args) {
        Auto a1 = new Auto();
        Auto a2 = new Auto(Merk.BMW, Kleur.WIT, 30000);
        Auto a3 = new Auto(Merk.AUDI, Kleur.ROOD, 35000, 14500, 103000);
        Auto a4 = new Auto(a3);
        Auto a5 = a4;

        a4.rij(1000);

        //data opvragen
        int km = a4.getKmStand();
        double aankoopprijs = a4.getAankoopprijs();
        double marktwaarde = a4.getMarktwaarde();

        km = a3.getKmStand();
        km = a5.getKmStand();
    }
}
```

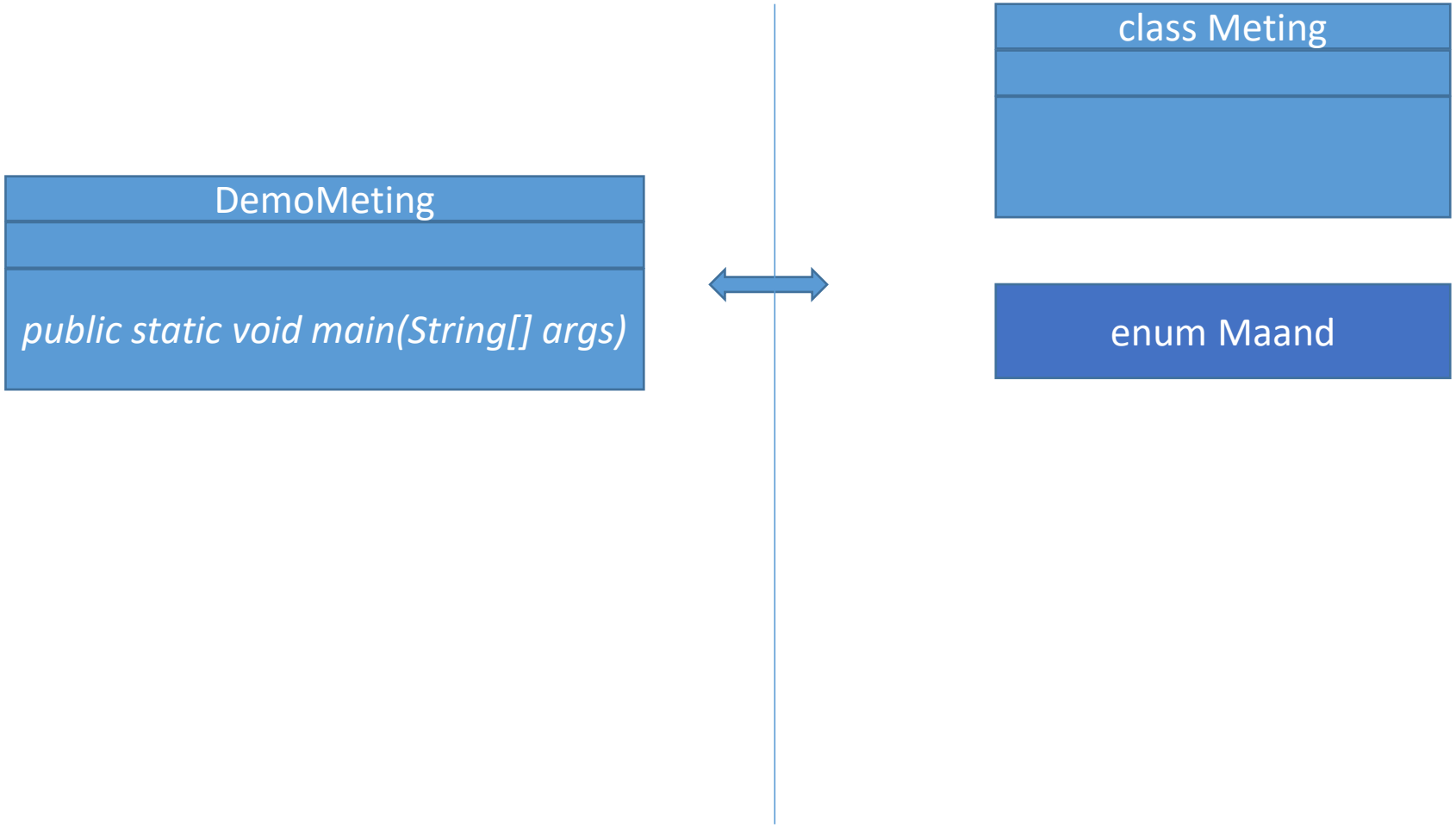
```
//104.000 km
//14.500€
// cf. implementatie
```

```
//103.000 km
//104.000 km
```

```
Auto a3 = new Auto(Merk.AUDI, Kleur.ROOD, 35000, 14500, 103000);
Auto a4 = new Auto(a3);
Auto a5 = a4;
```



Update Labo1: Samenspel met meerdere types



Update Labo1 >> eigen enumeratie

```
public enum Maand {
    JANUARI, FEBRUARI, MAART, APRIL, MEI, JUNI, JULI, AUGUSTUS, SEPTEMBER, OKTOBER, NOVEMBER, DECEMBER
}
```

```
public static void drukWaarden(double[][] waarden) {
    Maand[] maanden = Maand.values();

    for (int i = 0; i < maanden.length; i++) {
        System.out.printf("%15s", maanden[i] + ": ");

        for (double elt : waarden[i]) {
            System.out.print(elt + "\t");
        }

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

```
Output - 2019_CodeTheorieJavaOO (run)
       JANUARI: -13.0    -4.0    -9.5    3.5     -4.0    1.5     -11.5   -8
      FEBRUARI: -6.5     4.0     -6.5    7.0     2.0     -1.5    -2.0    6.
         MAART: 11.0     3.0     10.5    6.0     12.0    2.5     14.5    6.
         APRIL: 6.0      8.5     12.5    14.5    8.5     16.0    11.5    12
           MEI: 16.5     11.5    24.5    22.5    9.0     12.5    22.5    11
          JUNI: 16.5     19.0    27.0    24.0    27.0    10.5    25.0    29
          JULI: 21.5     23.5    12.5    15.5    14.5    11.0    10.5    18
     AUGUSTUS: 7.0       8.5     20.0    5.0     21.5    17.0    17.0    7.
    SEPTEMBER: 4.5       10.0    4.0     18.0    17.5    6.0     0.0     6.
      OKTOBER: 5.0       8.0     11.5    12.5    -1.5    5.0     3.5     5.
     NOVEMBER: 3.5       0.5     -9.0    7.5     5.0     8.5     4.5     1.
     DECEMBER: -4.0      -7.0    -10.0   -6.5    -2.5    -14.0   -14.5   -3
BUILD SUCCESSFUL (total time: 1 second)
```

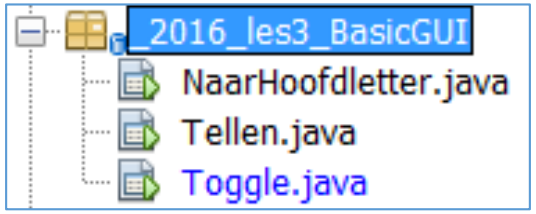
GUI

uitgebreide demo van scratch

Grafische toepassingen

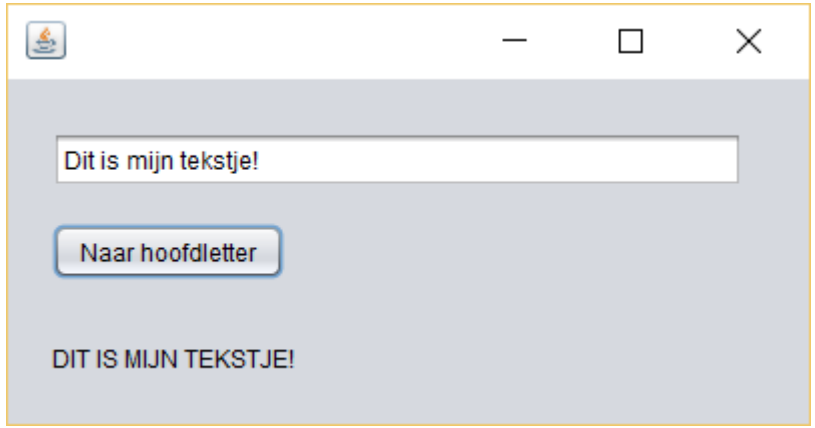
Na druk op KNOP1

Grafische toepassingen enkele mini-voorbeeldjes



Zonder afzonderlijke logische klasse

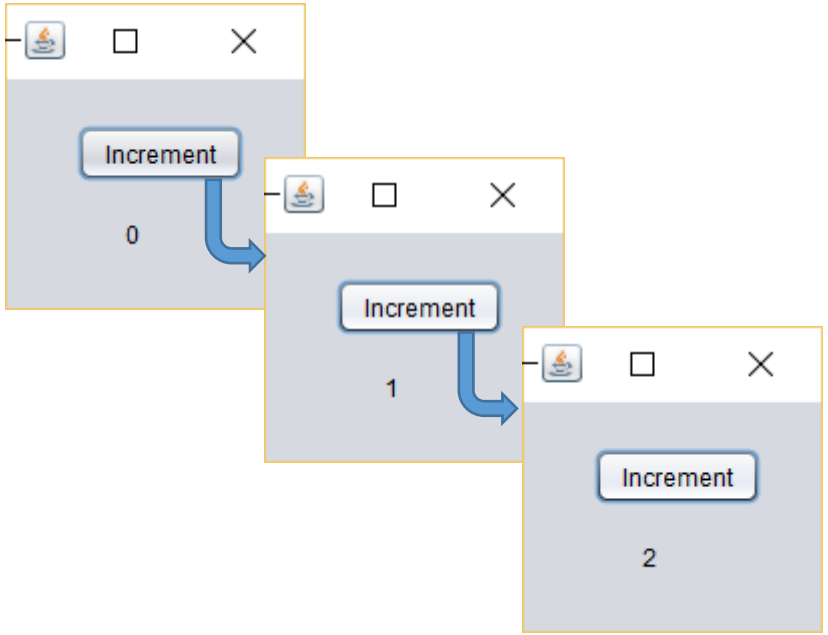

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    if (this.jButton1.getText().equals("TRUE")) {
        this.jButton1.setText("FALSE");
    }
    else {
        this.jButton1.setText("TRUE");
    }
}
```




```
private void jButtonZetOmActionPerformed(java.awt.event.ActionEvent evt) {
    String tekst = this.jTextFieldInput.getText();
    this.jLabelOutput.setText(tekst.toUpperCase());
}
```

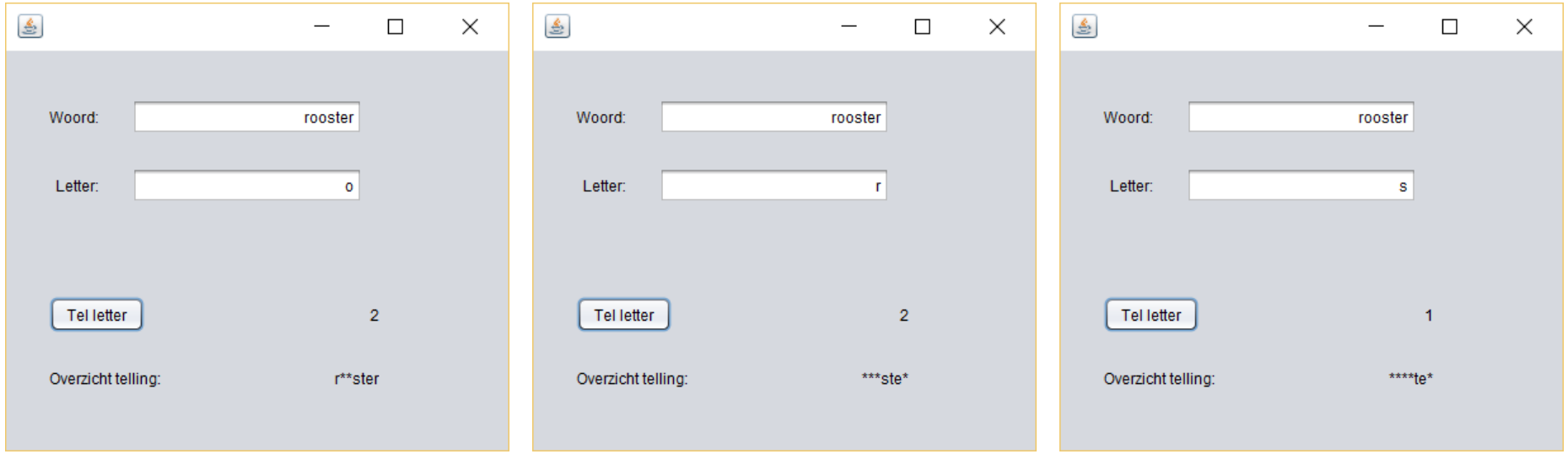
```
    // Variables declaration - do not modify
    private javax.swing.JButton jButtonZetOm;
    private javax.swing.JLabel jLabelOutput;
    private javax.swing.JTextField jTextFieldInput;
    // End of variables declaration
```

Zonder afzonderlijke logische klasse – globaal veld in de grafische klasse




```
public class Tellen extends javax.swing.JFrame {
    private int teller;

    private void jButtonTelActionPerformed(java.awt.event.ActionEvent evt) {
        this.jLabelTeller.setText("" + teller++);
    }
}
```

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