



Java Essentials

Hoofdstuk 12

Eenvoudige klassen

**DE HOGESCHOOL
MET HET NETWERK**

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



Inhoud

1. Inleiding
2. Wrappers voor primitieve datatypes
3. Datums en tijden
4. Samenvatting



1. Inleiding

Doel:

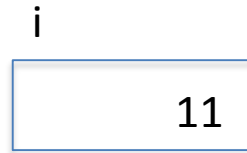
- eenvoudige klassen uit de Java API gebruiken
- Java API documentatie leren gebruiken



2. Wrappers voor primitieve datatypes

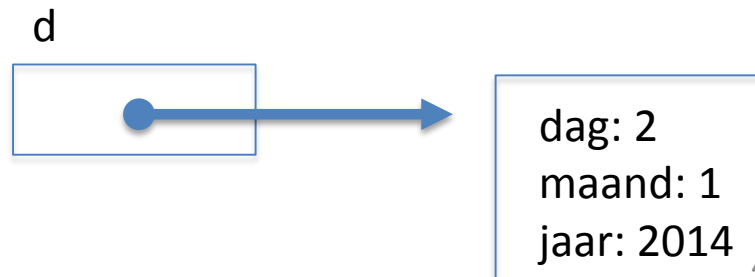
Primitieve variabele:

```
int i;  
i = 10;  
i++;  
System.out.println("de waarde van i " + i);
```



Referentievariabele:

```
Datum d; // declaratie van de variabele  
d = new Datum (2, 1, 2014); // aanmaak van de variabele  
d.printDatum(); // gebruik van de variabele
```



2.1 Wrapper-klassen

Voor elk primitief datatype bestaat er een object variant
= wrapperklasse

primitief datatype → wrapper class

vb `int`

kleine letter

vb `Integer`

hoofdletter

WrapperKlassen

Byte, Short, Integer, Long, Float, Double, Boolean, Character



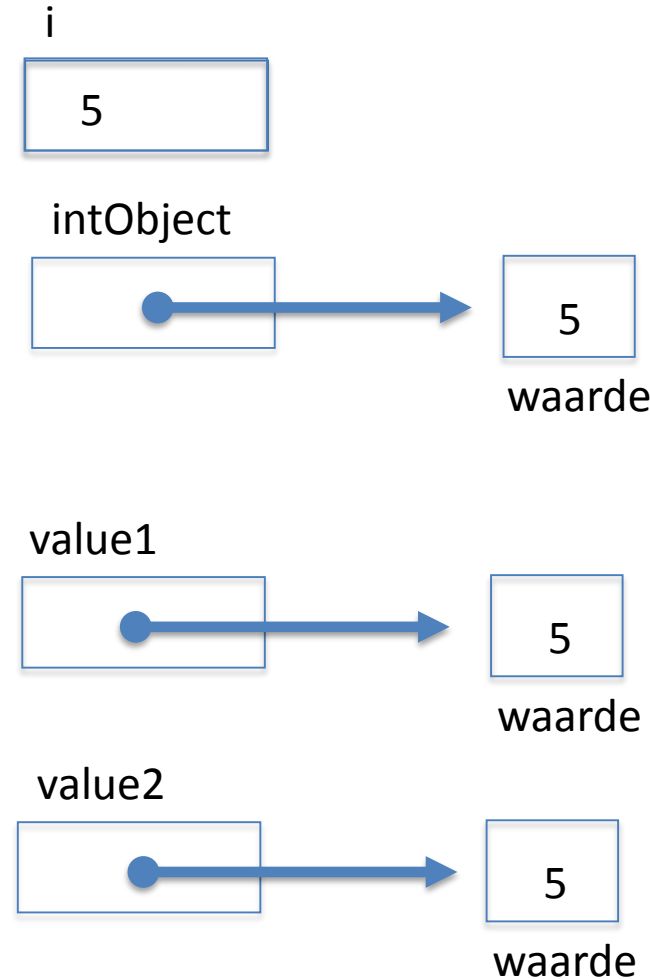
Voorbeeld:

```
int i = 88;  
Integer intObject = new Integer(5);  
System.out.println(intObject);  
i = intObject.intValue();  
System.out.println(i);
```

Output? 5
5

```
Integer value1 = new Integer(5);  
Integer value2 = new Integer(5);  
System.out.println(value1 == value2);  
System.out.println(value1.equals(value2));
```

Output? false
true



2.2 Autoboxing

autoboxing

primitief datatype → wrapper object

```
Integer intObject = 5;
```

auto-unboxing

wrapper object → primitief datatype

```
int intPrimitive = new Integer(6);
```

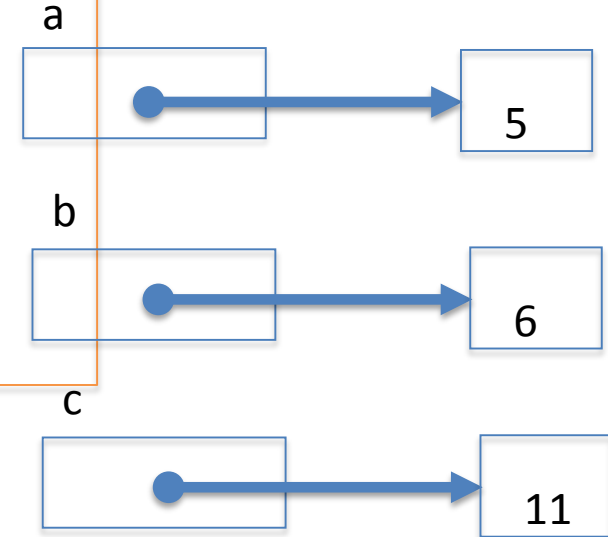
Omzettingen gebeuren *automatisch* door de compiler.

=> wrapper objecten en primitieve datatypes mogen door elkaar gebruikt worden



Voorbeelden

```
public class BadAutoboxing {  
    public static void main(String[] args) {  
        Integer a = 5;  
        Integer b = 6;  
        Integer c = a + b;  
        System.out.println(c);  
    }  
}
```



wanneer er veel gerekend moet worden
=> zeker niet met objecten werken!!!!

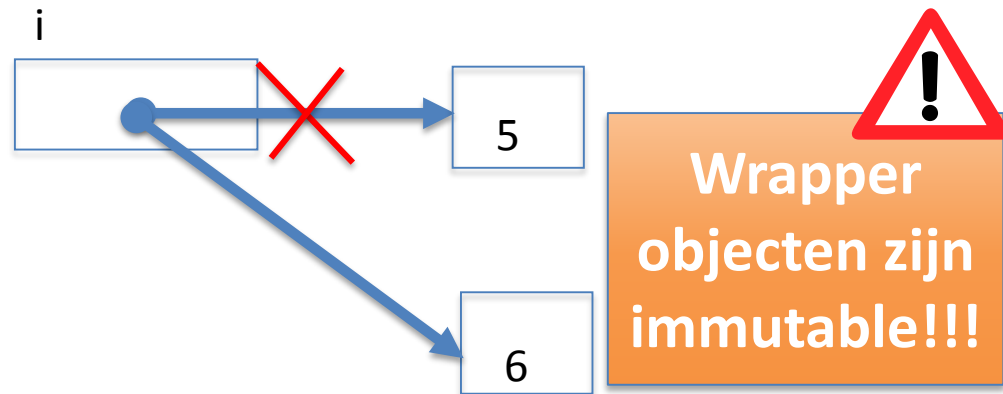

```
public class AutoboxingIncrement {  
    public static void main(String[] args) {  
        Integer i = 5;  
        i++;  
        System.out.println(i);  
    }  
}
```

Wat gebeurt er in het intern geheugen?

1. autoboxing

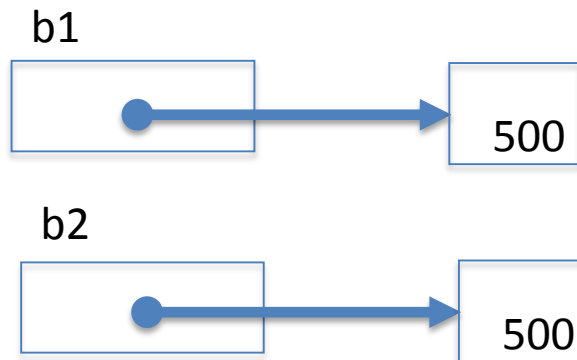
2. auto – unboxing

primitieve waarde verhogen
autoboxing



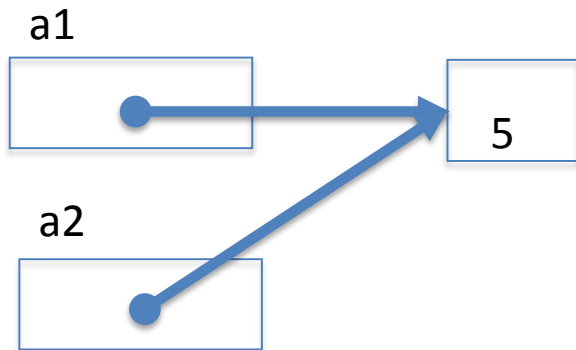
Autoboxing en de == en != operatoren

```
public class AutoboxingAssignement {  
    public static void main(String[] args) {  
        Integer b1 = 500;  
        Integer b2 = 500;  
  
        System.out.println(b1 == b2); // false  
        System.out.println(b1 != b2); // true  
    }  
}
```



Autoboxing en de == en != operatoren

```
public class AutoboxingAssignement {  
    public static void main(String[] args) {  
        Integer a1 = 5;  
        Integer a2 = 5;  
  
        System.out.println(a1 == a2); // true  
        System.out.println(a1 != a2); // false  
    }  
}
```



Getallen waarvan de waarde in 8 bits kan en booleans
=> gereserveerd stuk intern geheugen vergelijkbaar met canonical String area
Betere manier: equals methode!!!



Method overloading en autoboxing

```
public class AutoboxingOverloading {  
  
    public static void main(String[] args) {  
        method(5);  
    }  
  
    private static void method(long i) {  
        System.out.println("Long");  
    }  
  
    private static void method(Integer i) {  
        System.out.println("Integer");  
    }  
}
```

Output? long

Eerst steeds widening
Reden: compatibiliteit met
vorige Java versies



Method overloading en autoboxing

```
public class AutoboxingWidening {  
  
    public static void main(String[] args) {  
        int a = 5;  
        method(a);  
    }  
  
    private static void method(Long l) {  
        System.out.println("Long");  
    }  
}
```

Output? foutmelding

Autoboxing alleen naar
overeenkomstig wrapper type



2.3 static members: static variabelen

WrapperKlassen: Byte, Short, Integer, Long, Float, Double

MIN_VALUE minimale waarde

MAX_VALUE maximale waarde

WrapperKlassen: Float, Double

NaN resultaat van 0/0

NEGATIVE_INFINITY resultaat van getal(<0)/0

POSITIVE_INFINITY resultaat van getal (>0)/0

Zie Javadoc voor andere klasse (static) variabelen



2.3 static members : static methoden

- Conversie methoden String => primitief datatype

bv String -> int Integer.parseInt ()

 String -> float Float.parseFloat()



2.3 static members : static methoden

Andere static methoden:

- `Float.isNaN()` nagaan getal ongeldige waarde (0/0)
- `Float.isInfinite()` nagaan of de waarde oneindig is

```
System.out.println(Float.isInfinite( 5.0f / 0)); //true
```

Opgepast: er is ook een niet-static versie van de methode!

```
VB    Float f = new Float(5);  
      f = f / 0;  
      System.out.println(f.isInfinite()); //true
```

Zie Javadoc voor andere klasse (static) methoden



Opdracht 1:

Werken met de klasse Integer

- Open de API-documentatie van de klasse Integer.
- Maak een programma dat aan de gebruiker een waarde vraagt en deze inleest als een String.
- Zet de string-waarde om in een Integer-object.
- Tel bij dit object een waarde op en druk het resultaat af.
- Druk het aantal bits voor een integer af.
- Druk het aantal bytes voor een integer af.



Opdracht 2:

Wat is de output van onderstaand programma?

- Bekijk de uitvoer van Opdracht2.java.
- Gebruik de java doc om de // - lijnen van de nodige commentaar te voorzien.
- Voer het programma opnieuw uit met de waarde 253, -45 voor k. Ontleed de bekomen resultaten.



Opdracht 3:

De klasse BigInteger en BigDecimal.

- Open de API documentatie van de klasse BigInteger en de klasse BigDecimal.
- Schrijf een programma voor het berekenen van faculteit van een getal (bv? $5! = 5 \times 4 \times 3 \times 2 \times 1$).
Test je programma uit => wat merk je?
- Herschrijf je programma door gebruik te maken van de klasse BigInteger.



3. Datums en tijden

3.1 Inleiding

- Sinds Java 8 nieuwe klassen in package **java.time**
- Enkele moeilijkheden
 - 2 manieren om naar tijd te kijken
 - mens => jaren, maanden, dagen, uren, ...
 - computer => aantal tijdseenheden vanaf bepaald punt
 - Tijdszones => tijd verschilt op ieder punt op de aarde

Europe/London	Z	Greenwich
Europe/Brussels	+01:00	
America/New_York	-05:00	
 - Schrikkeljaren
 - Zomertijd



3.2 Computertijden: de klasse Instant

- Computertijd = aantal tijdseenheden voor of na een nulpunt
nulpunt = 1 januari 1970 om 00:00:00 (EPOCH)
tijdseenheid = 1 nanoseconde = 10^{-9} sec



- Klasse variabelen EPOCH, MAX, MIN
- Klasse Instant geen publieke constructor
=> onmogelijk om zelf instanties van klasse Instant aan te maken
- Statische methoden die instantie van klasse teruggeven
Bvb. `Instant now = Instant.now();`
- Methoden om bewerkingen te doen op een tijdstip (zie cursus, Javadoc)



```
import java.time.*;
```

```
public class InstantApp {
```

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

```
        System.out.println(Instant.EPOCH);
```

```
        System.out.println(Instant.MIN);
```

```
        System.out.println(Instant.MAX);
```

```
        Instant now = Instant.now();
```

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

```
        System.out.println(now.getEpochSecond());
```

```
        System.out.println(now.getNano());
```

```
        Instant earlier = now.minusSeconds(20);
```

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

```
        Instant later = now.plusSeconds(20);
```

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


```
        System.out.println(now.isAfter(earlier));
```

```
        System.out.println(now.isBefore(later));
```

```
    }
```

```
}
```

Greenwich



```
1970-01-01T00:00:00Z  
-10000000000-01-01T00:00:00Z  
+10000000000-12-  
31T23:59:59.999999999Z
```



Opdracht 4:

De klasse Instant

Maak een programma dat de gebruiker vraagt zijn naam in te geven.

Toon hoe lang dit geduurd heeft (in seconden).



3.3 Menselijke datums en tijden

3.3.1 Enumeraties voor weekdays en maanden

Enumeratie Month

- bevat 12 instanties voor de maanden: Month.JANUARY, ...
- interessante methoden (zie cursus, Javadoc)

...

Enumeratie DayOfWeek

- bevat 7 instanties voor de dagen: DayOfWeek.MONDAY, ...
- interessante methoden (zie cursus, Javadoc)



...

```
public class EnumMonthDay {
```

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

```
        Month maand = Month.FEBRUARY;
```

```
        DayOfWeek dag = DayOfWeek.FRIDAY;
```

```
        System.out.println(dag + " " + maand);
```

bevat lokale gegevens

```
        Locale lokaal = Locale.getDefault();
```

```
        System.out.println(maand.getDisplayName(TextStyle.FULL, lokaal));
```

```
        System.out.println(maand.getDisplayName(TextStyle.NARROW, lokaal));
```

```
        System.out.println(maand.getDisplayName(TextStyle.SHORT, lokaal));
```

```
        System.out.print("Het aantal dagen in deze maand ");
```

```
        System.out.println(maand.length(true));
```

```
        System.out.println("5 maanden verder is de maand " + maand.plus(5));
```

```
        System.out.print("3 dagen eerder is ");
```

```
        System.out.println(dag.minus(3).getDisplayName(TextStyle.FULL, lokaal));
```

```
    }
```

```
}
```

FRIDAY FEBRUARY

februari

F

feb

Het aantal dagen in deze maand 29

5 maanden verder is de maand JULY

3 dagen eerder is dinsdag



Opdracht 5:

Weekdagen en maanden

Maak een programma dat de gebruiker vraagt een weekdag in te geven (1-7) en een aantal dagen om erbij op te tellen.

Druk af welke dag van de week dit is.



3.3.2 Lokale datums en tijden

- Lokaal = geen rekening houdend met de tijdszones of het zomertijd
- Tijden \Rightarrow klasse `LocalTime`
Datums zonder tijden \Rightarrow klasse `LocalDate`
Datum en tijd \Rightarrow klasse `LocalDateTime`
- Deze klassen hebben geen constructors
 \Rightarrow statische methoden om instanties te bekomen

bvb methode `now()`
methoden `of()`



```

...
import java.time.*;

public class LocalDateTimeApp {
    public static void main(String args[]) {
        LocalDate nowDate = LocalDate.now();
        LocalTime nowTime = LocalTime.now();
        LocalDateTime nowDateTime = LocalDateTime.now();

        LocalDate otherDate = LocalDate.of(2015,6,23);
        LocalTime otherTime = LocalTime.of(10,25,2);
        LocalDateTime otherDateTime = LocalDateTime.of(otherDate,otherTime);

        System.out.println(nowDate);
        System.out.println(nowTime);
        System.out.println(nowDateTime);

        System.out.println(otherDate);
        System.out.println(otherTime);
        System.out.println(otherDateTime);
    }
}

```

```

2015-11-16
14:17:08.537
2015-11-16T14:17:08.537
2015-06-23
10:25:02
2015-06-23T10:25:02

```



Opdracht 6:

Weekdagen en maanden

Maak een programma dat van je geboortedatum het volgende afdrukt:

- De hoeveelste dag van het jaar het was
- De dag van de week
- Of dit al dan niet een schrikkeljaar was



3.3.3 Tijdzones

- Tijdzones
 1. info over regio/stad bvb Europe/Brussels
 2. info over verschil met standaard tijdzone UTC/Greenwich
- Klasse **Zoneld** tijdzone gebaseerd op regio/stad
- Klasse **ZoneOffset** tijdzone gebaseerd op verschil met standaard tijdzone.
Opm: Klasse ZoneOffset subklasse van klasse Zoneld
- Deze klassen hebben geen constructors
=> statische methoden om instanties te bekomen



```
...  
import java.time.*;  
  
public class TimeZone {  
  
    public static void main(String[] args) {  
        ZoneId zoneId = ZoneId.of("Europe/Brussels" );  
        System.out.println(zoneId);  
  
        ZoneId systemZoneId = ZoneId.systemDefault();  
        System.out.println(systemZoneId);  
  
        ZoneOffset timeZone = ZoneOffset.ofHours(2);  
        System.out.println(timeZone);  
    }  
}
```

Tijdszone op basis
van de systeemklok



Europe/Brussels
Europe/Paris
+02:00



3.3.4 Datums en tijden met tijdzones

- Klasse `ZonedDateTime` om datums/tijden te gebruiken rekening houdend met tijdzones.
- Geen constructors
=> statische methoden om instantie te bekomen
Bvb `of()` methoden
- Andere methoden zie Javadoc




```

...
import java.time.*;

public class ZonedDateTimeAppBis {

    public static void main(String[] args) {
        ZoneId londonZone = ZoneId.of("Europe/London");
        LocalDate datum = LocalDate.of(2016, Month.JANUARY, 5);
        LocalTime tijd = LocalTime.of( 20, 19, 30);

        ZonedDateTime nowBrussels = ZonedDateTime.now();
        ZonedDateTime nowLondon = ZonedDateTime.now(londonZone);
        ZonedDateTime dat = ZonedDateTime.of(datum, tijd, londonZone);

        System.out.println(nowBrussels);
        System.out.println(nowLondon);
        System.out.println(dat);
    }
}

```

```

2015-11-16T15:08:53.010+01:00[Europe/Paris]
2015-11-16T14:08:53.018Z[Europe/London]
2016-01-05T20:19:30Z[Europe/London]

```



Opdracht 7:

Tijden met tijdzones

Maak een programma dat de huidige tijd afdrukt in volgende steden

- je huidige plaats
- London (Europe/London)
- Sydney (Australia/Sydney)
- Los Angeles (America/Los_Angeles)
- UTC-4



3.4 Tijdsduur

Te gebruiken klassen

- Duration
tijdsverschillen tussen machinetijden (= instanties van de klasse Instant)
- Period
tijdsverschillen tussen “datebased-time” (= menselijk tijden)
- ChronoUnit: enumeratie
 - verschillende tijdseenheden: ChronoUnit.HOURS, ChronoUnit.DAYS, ...
 - tijdsverschil kan met methode between() omgezet worden in de gekozen tijdseenheid



...

```
public class DurationAppBis {  
    public static void main(String[] args) {  
        Instant now = Instant.now();  
        System.out.println("nu " + now);  
        Instant later = now.plusSeconds(500).plusMillis(2125698).plusNanos(456398);  
        System.out.println("xx seconde later " + later);  
  
        Duration duration = Duration.between(now, later);  
        System.out.println("duration: " + duration);  
  
        long milliseconds = ChronoUnit.MILLIS.between(now, later);  
        System.out.println("milliseconden: " + milliseconds);  
  
        LocalDate nowDate = LocalDate.now();  
        LocalDate thenDate = LocalDate.of(1980, 2, 15);  
  
        System.out.println("Datum1: " + nowDate);  
        System.out.println("Datum2: " + thenDate);  
  
        Period period = Period.between(thenDate, nowDate);  
        System.out.println("jaar: " + period.getYears() + " maanden: " + period.getMonths() +  
            " dagen: " + period.getDays());  
  
        long days = ChronoUnit.DAYS.between(thenDate, nowDate);  
        System.out.println("aantal dagen tussen 2 datums " + days);  
    }  
}
```

nu 2015-11-16T15:05:53.838Z

xx seconden later 2015-11-16T15:49:39.536456398Z

2625

2625698

Datum1: 2015-11-16

Datum2: 1980-02-15

jaar: 35 maanden:9 dagen: 1

aantal dagen tussen 2 datums 13058



Opdracht 8:

Tijdsduur

Maak een programma dat de periode berekent sinds je geboorte. Druk van deze periode het volgende af

- het aantal dagen
- het aantal maanden
- het aantal jaren
- Bereken je leeftijd uitgedrukt in maanden



3.5 Formattering van datums en tijden

- Conversie tussen tekst-voorstelling en Java datum/tijd-objecten
=> klasse `DateTimeFormatter`
- Conversie van tekenreeks naar datum/tijd-object (**parse**)
- Conversie van datum/tijd-object naar een tekenreeks (**format**)
- Conversie kan volgens eigen opgegeven patroon of volgens standaard patronen (statische constanten van de klasse).
Verdere uitleg tekens in patroon zie Javadoc.



```
public class FormatterApp {  
  
    public static void main(String[] args) {  
        DateTimeFormatter myFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");  
        DateTimeFormatter myFormatter2 = DateTimeFormatter.ofPattern("d/M/yy H:m:s");  
        DateTimeFormatter isoFormatter = DateTimeFormatter.ISO_LOCAL_DATE_TIME;  
  
        LocalDateTime dt1 = LocalDateTime.parse("03/08/1998 13:45:03",myFormatter1);  
  
        System.out.println(myFormatter1.format(dt1));  
        System.out.println(myFormatter2.format(dt1));  
        System.out.println(isoFormatter.format(dt1));  
    }  
}
```

03/08/1998 13:45:03

3/8/98 13:45:3

1998-08-03T13:45:03



```

public class FormatterApp {

    public static void main(String[] args) {
        DateTimeFormatter myFormatter1 = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
        DateTimeFormatter myFormatter2 = DateTimeFormatter.ofPattern("d/M/yy H:m:s");
        DateTimeFormatter isoFormatter = DateTimeFormatter.ISO_LOCAL_DATE_TIME;

        LocalDateTime dt1 = LocalDateTime.parse("03/08/1998 13:45:03", myFormatter2);

        System.out.println(myFormatter1.format(dt1));
        System.out.println(myFormatter2.format(dt1));
        System.out.println(isoFormatter.format(dt1));

    }
}

```

Exception in thread "main" java.time.format.DateTimeParseException: Text '03/08/1998 13:45:03' could not be parsed at index 8
 at java.time.format.DateTimeFormatter.parseResolved0(Unknown Source)
 at java.time.format.DateTimeFormatter.parse(Unknown Source)
 at java.time.LocalDateTime.parse(Unknown Source)
 at voorbeeldenDateTime.FormatterApp.main(FormatterApp.java:15)



- Afdrukken van datum/tijd-objecten
Gebruik maken van printf (zie ook hoofdstuk 6)

```
...  
public class DatumPrintf {  
  
    public static void main(String[] args) {  
        LocalDateTime datum = LocalDateTime.of(2016, 5, 1, 15, 45, 30);  
  
        System.out.format("%1$td/%1$tB/%1$ty %1$tH:%1$tM:%1$tS \n", datum);  
  
    }  
}
```

01/mei/16 15:45:30

3.5 Omzetting van en naar Date en Calendar

- Vóór Java 8 werd voor datums en tijden gebruik gemaakt van de klassen Date en Calendar.
- Uitgebreid met methoden om ze om te zetten naar objecten van de klasse Instant of ZonedDateTime



Opdracht 9:

Formatting

- Maak een programma dat de gebruiker vraagt een datum in te geven in het formaat DD/MM/YYYY.
- Zet de bekomen tekst om in een object van de klasse `LocalDate` en druk de inhoud af in het formaat YYYY-MM-DD.



4. Samenvatting

In dit hoofdstuk werden veel gebruikte klassen besproken

- Wrapperklassen
=> om primitieve datatypen te kunnen gebruiken als objecten
- Klassen voor Datum/Tijd
 - ✓ Computertijd => Instant
 - ✓ Datum (zonder tijdzone) => LocalDate
 - ✓ Tijd (zonder tijdzone) => LocalTime
 - ✓ Datum en Tijd (zonder tijdzone) => LocalDateTime
 - ✓ Datum en Tijd (met tijdzone) => ZonedDateTime
 - ✓ Tijdsduur (computertijd) => Duration
 - ✓ Tijdsduur (menselijke tijd) => Period
- enumeraties voor Datum/Tijd
=> Month, DayOfWeek, ChronoUnit

