# Informatica

## Les 2

## bibliotheekklassen - Leibniz - digitaal

Jan Lemeire

**Informatica 2e semester**

*februari – mei 2022*

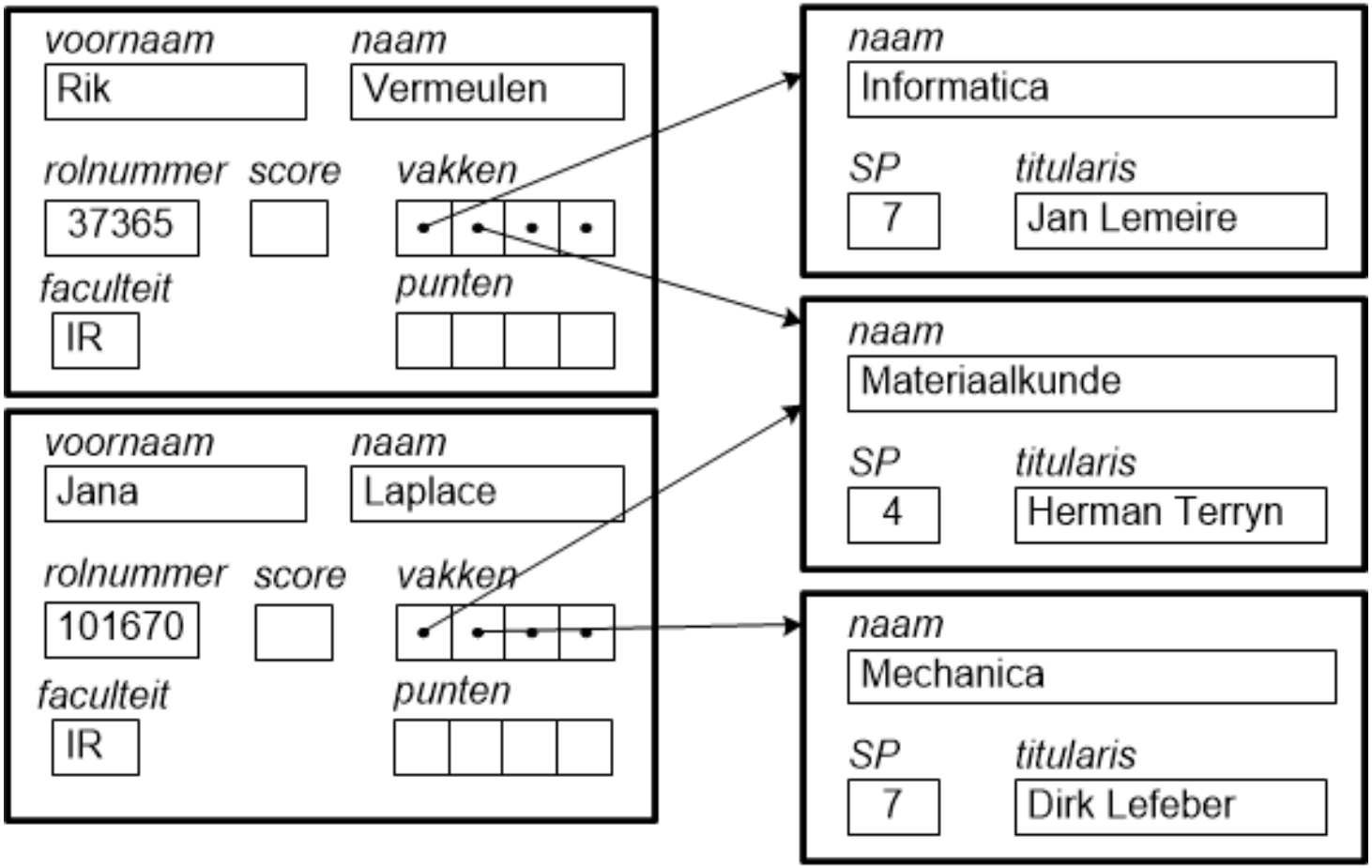


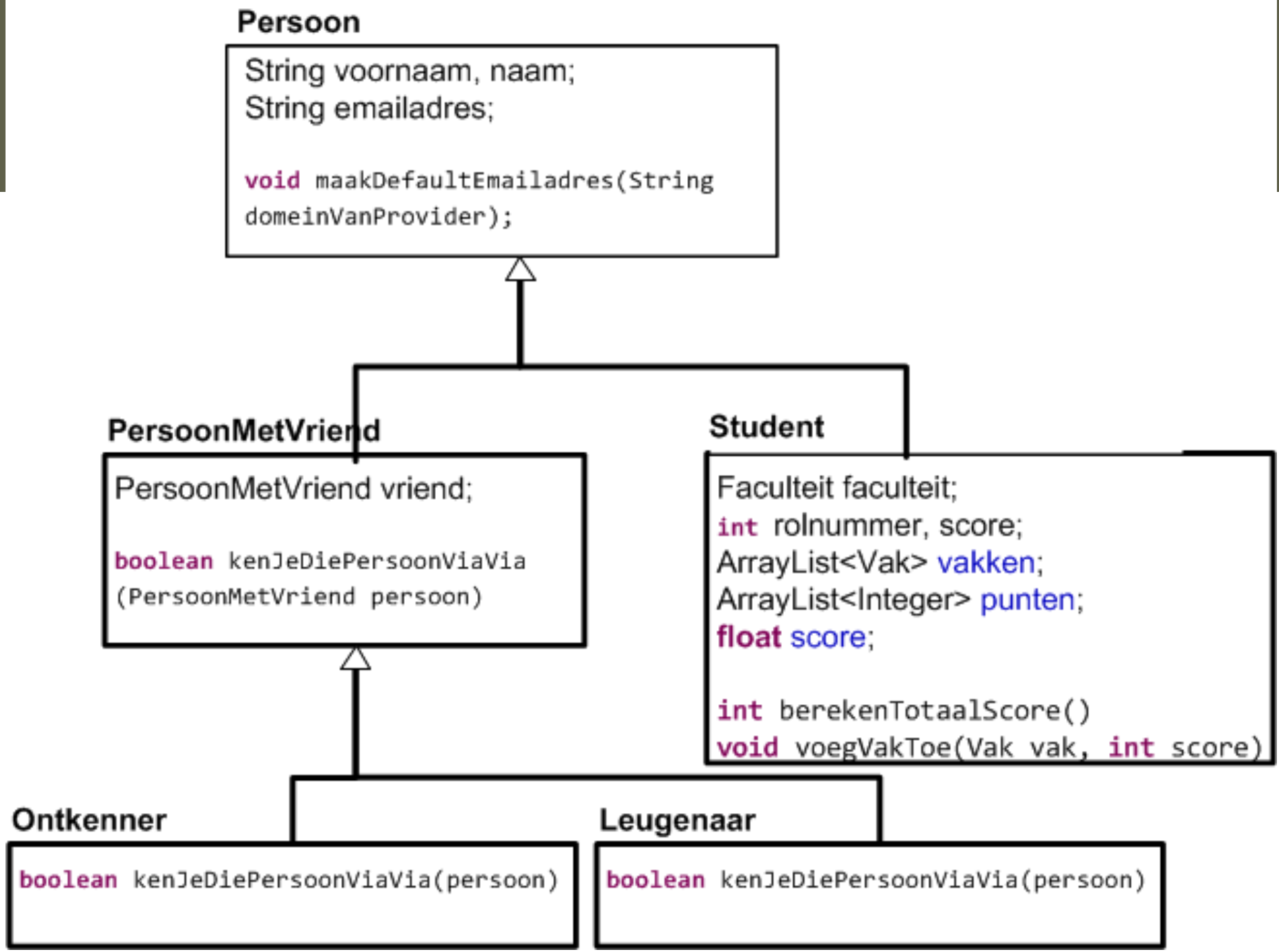
VRIJE UNIVERSITEIT BRUSSEL

# Vandaag

1. Object-georiënteerd programmeren
2. Oefening
3. Bibliotheekklassen
4. Goocheltruc
5. Deel III: hoofdstuk 0
6. Deel III: hoofdstuk 1
7. Binaire representatie

# Object-georiënteerd programmeren


voornaam
Rik
naam
Vermeulen
rolnummer
37365
score
vakken
faculteit
IR
punten
voornaam
Jana
naam
Laplace
rolnummer
101670
score
vakken
faculteit
IR
punten
naam
Informatica
SP
7
titularis
Jan Lemeire
naam
Materiaalkunde
SP
4
titularis
Herman Terryn
naam
Mechanica
SP
7
titularis
Dirk Lefeber


Persoon
String voornaam, naam;
String emailadres;
void maakDefaultEmailadres(String domeinVanProvider);
PersoonMetVriend
PersoonMetVriend vriend;
boolean kenJeDiePersoonViaVia (PersoonMetVriend persoon)
Student
Faculteit faculteit;
int rolnummer, score;
ArrayList<Vak> vakken;
ArrayList<Integer> punten;
float score;
int berekenTotaalScore()
void voegVakToe(Vak vak, int score)
Ontkenner
boolean kenJeDiePersoonViaVia(persoon)
Leugenaar
boolean kenJeDiePersoonViaVia(persoon)

```java
public class Persoon {

    //========================== ATTRIBUTEN ============================//
    String voornaam, naam;
    String emailadres;

    //========================== CONSTRUCTORS ==========================//
    Persoon(String voornaam, String naam){
        this.voornaam = voornaam;
        this.naam = naam;
    }
    Persoon(String voornaam, String naam, String emailadres){
        this.voornaam = voornaam;
        this.naam = naam;
        this.emailadres = emailadres;
    }

    //========================== METHODES ==============================//
    public void maakDefaultEmailadres(String domeinVanProvider){
        emailadres = voornaam+"."+naam+"@"+domeinVanProvider;
    }

    public String toString(){
        return voornaam+" "+naam+(emailadres==null?"": " ["+emailadres+"]");
    }
}
```

p. 21

```
public class Student extends Persoon
{
        enum Faculteit {IR, WE, GF, LK, LW, ES, RC, PE};

        int rolnummer;
        Faculteit faculteit = Faculteit.IR;
        ArrayList<Vak> vakken;
        Map<Vak, Integer> punten;
        float score;

        Student(String voornaam, String naam, int rolnummer, Faculteit fac){
                super(voornaam, naam);
                this.rolnummer=rolnummer;
                this.faculteit = fac;
                vakken = new ArrayList<Vak>();
                punten = new HashMap<Vak, Integer>();
        }
}
public class Vak {
        String naam, titularis;
        int SP;
        Vak(String naam, String titularis, int SP){
                this.naam = naam;
                this.titularis = titularis;
                this.SP = SP;
        }
}
```

Map

p. 33

```
public static void main(String[] args) {
    Student rik = new Student("Rik", "Vermeulen", 37365, Faculteit.IR);
    Student jana = new Student("Jana", "Laplace", 101670, Faculteit.WE);

    Vak informatica = new Vak("Informatica", "Jan Lemeire", 7);
    Vak materiaalkunde = new Vak("Materiaalkunde", "Herman Terryn", 4);
    Vak mechanica = new Vak("Mechanica", "Dirk Lefeber", 7);

    rik.vakken.add(informatica);
    rik.punten.put(informatica, 14);
    rik.vakken.add(materiaalkunde);
    rik.punten.put(materiaalkunde, 12);
    rik.vakken.add(mechanica);
    rik.punten.put(mechanica, 17);

    jana.vakken.add(informatica);
    jana.punten.put(informatica, 16);
    jana.vakken.add(mechanica);
    jana.punten.put(mechanica, 13);
}
```

# Berekening score

Deze methode staat in de klasse Persoon en heeft zo toegang tot attributen `score` en `punten`. *Anders moet je het object er voor zetten met een punt.*

```
float berekenTotaalScore(){
    score=0; // score is gedefinieerd als attribuut van de klasse
    int totaalSP=0; // dit is een lokale variabele van de methode
    for(Vak vak: vakken){
        score += punten.get(vak) * vak.SP;
        totaalSP += vak.SP;
    }
    score /= totaalSP;
    return score;
}
```

for-lus

```
float scoreRik = rik.berekenTotaalScore();
float scoreJana = jana.berekenTotaalScore();
System.out.println(rik+" behaalde "+scoreRik+"/20");
System.out.println(jana+" behaalde "+scoreJana+"/20");
```

# Oefening

```java
public class ObjectOefening {

    /** PROGRAMMA */
    public static void main(String[] args) {

        KlasseA a1 = new KlasseA(10);
        KlasseA a2 = new KlasseA(2, 8);

        System.out.println("a1 = "+a1);
        System.out.println("a2 = "+a2);

        KlasseB b1 = new KlasseB();
        System.out.println("b1 = "+b1);

        a1.f(3);
        a2.f(3);

        System.out.println("a1 = "+a1);
        System.out.println("a2 = "+a2);

        b1.g(a2);
        System.out.println("b1 = "+b1);
    }
}
```

```java
class KlasseA {
    int x, y;
    KlasseA(int x){
        this.x=x;
        this.y=0;
    }
    KlasseA(int x, int y){
        this.x=x;
        this.y=y;
    }
    void f(int factor){
        x = factor * x;
        y = y / factor;
    }
    public String toString(){
        return "A [x="+x+";y="+y+"]";
    }
}
class KlasseB {
    int z = 5; // default waarde

    void g(KlasseA a){
        z = z * a.x + a.y;
    }
    public String toString(){
        return "B [z="+z+"]";
    }
}
```

# Javabibliotheek-klassen

# Scanner

- stelt je in staat om String te ontleden
- verwerken gebruikersinvoer en/of tekstbestanden

```
import java.util.Scanner;
public class ScannerExample {

	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);

		System.out.print("Geef een invoer:");
		double num = scanner.nextDouble();

		System.out.println("De vierkantswortel van " + num +
"is: " + Math.sqrt(num));
		scanner.close();
	}
}
```

# Hoe?

1. Maak een object via 1 van de constructors
2. Gebruik methoden op object
3. Eventueel afsluiten

**Gebruik online documentatie**

# Constructors

## Constructor Summary

**Scanner**(**InputStream** source)
Constructs a new Scanner that produces values scanned from the specified input stream.

**Scanner**(**Path** source)
Constructs a new Scanner that produces values scanned from the specified file.

**Scanner**(**String** source)
Constructs a new Scanner that produces values scanned from the specified string.

| Method Summary | |
|---|---|
| void | **close**()<br>Closes this scanner. |
| **String** | **findInLine**(**String** pattern)<br>Attempts to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters. |
| **String** | **findWithinHorizon**(**Pattern** pattern, int horizon)<br>Attempts to find the next occurrence of the specified pattern. |
| boolean | **hasNext**()<br>Returns true if this scanner has another token in its input. |
| boolean | **hasNext**(**String** pattern)<br>Returns true if the next token matches the pattern constructed from the specified string. |
| boolean | **hasNextBoolean**()<br>Returns true if the next token in this scanner's input can be interpreted as a boolean value using a case insensitive pattern created from the string "true\|false". |
| boolean | **hasNextDouble**()<br>Returns true if the next token in this scanner's input can be interpreted as a double value using the **nextDouble()** method. |
| boolean | **hasNextFloat**()<br>Returns true if the next token in this scanner's input can be interpreted as a float value using the **nextFloat()** method. |
| boolean | **hasNextInt**()<br>Returns true if the next token in this scanner's input can be interpreted as an int value in the default radix using the **nextInt()** method. |
| boolean | **hasNextLine**()<br>Returns true if there is another line in the input of this scanner. |
| boolean | **hasNextLong**()<br>Returns true if the next token in this scanner's input can be interpreted as a long value in the default radix using the **nextLong()** method. |
| **String** | **next**()<br>Finds and returns the next complete token from this scanner. |
| boolean | **nextBoolean**()<br>Scans the next token of the input into a boolean value and returns that value. |
| double | **nextDouble**()<br>Scans the next token of the input as a double. |
| float | **nextFloat**()<br>Scans the next token of the input as a float. |

# 1.2.2 Random

- Pseudowillekeurig getal, want computer is een *deterministisch systeem*
- Gebaseerd op een algoritme die een reeks van schijnbaar-total-willekeurige getallen genereert

```
nSeed = 5323 # initial starting seed
def PRNG():
    global nSeed

    # Take the current seed and generate a new value from it
    nSeed = 8253729 * nSeed + 2396403

    # Take the seed and return a value between 0 and 32767
    return nSeed  % 32767
```

for-lus

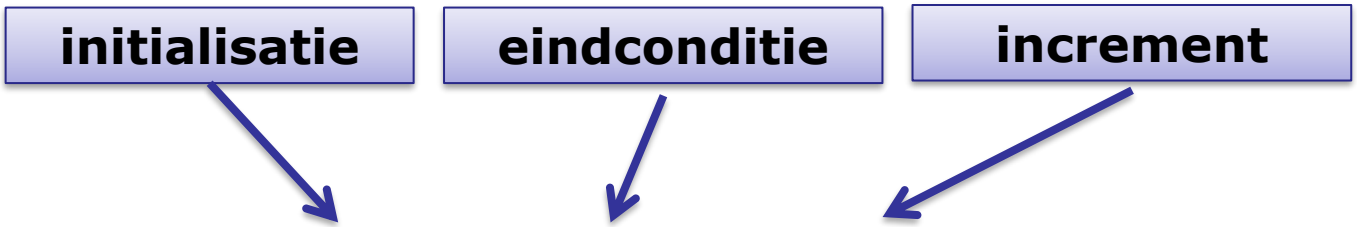
p. 15-16

```
import java.util.Random;

public class Randomtest {
   public static void main(String[] args) {
// de seed wordt automatisch bepaald (op basis van de huidige tijd)
        Random rand = new Random();
        for (int i=0; i<8; i++)
                System.out.print(rand.nextInt(1000) + ", ");
        System.out.println();

        rand = new Random(42); // vaste seed
        for (int i=0; i<8; i++)
             System.out.print(rand.nextInt(1000) + ", ");
        System.out.println();

        rand = new Random(42); // vaste seed (dezelfde!)
        for (int i=0; i<8; i++)
              System.out.print(rand.nextInt(1000) + ", ");
        System.out.println();
   }
}
```

Uitvoer:

```
200, 593, 612, 198, 427, 419, 276, 790,
130, 763, 248, 884, 970, 525, 505, 918,
130, 763, 248, 884, 970, 525, 505, 918,
```

# Java's for-lus




```
for(int i=0;i<100;i+=2){
      System.out.print(i+", ");
}
```

$$i \mathrel{+}= 2 \quad \equiv \quad i = i + 2$$

$$i{+}{+} \quad \equiv \quad i \mathrel{+}= 1 \quad \equiv \quad i = i + 1$$

# Constructor

## Constructor Summary

**Random**()
Creates a new random number generator.

**Random**(long seed)
Creates a new random number generator using a single long seed.

## Method Summary

| | |
|---|---|
| boolean | **nextBoolean**()<br>Returns the next pseudorandom, uniformly distributed boolean value from this random number generator's sequence. |
| void | **nextBytes**(byte[] bytes)<br>Generates random bytes and places them into a user-supplied byte array. |
| double | **nextDouble**()<br>Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence. |
| float | **nextFloat**()<br>Returns the next pseudorandom, uniformly distributed float value between 0.0 and 1.0 from this random number generator's sequence. |
| double | **nextGaussian**()<br>Returns the next pseudorandom, Gaussian ("normally") distributed double value with mean 0.0 and standard deviation 1.0 from this random number generator's sequence. |
| int | **nextInt**()<br>Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence. |
| int | **nextInt**(int bound)<br>Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence. |
| long | **nextLong**()<br>Returns the next pseudorandom, uniformly distributed long value from this random number generator's sequence. |
| void | **setSeed**(long seed)<br>Sets the seed of this random number generator using a single long seed. |

# 1.2.3 De ArrayList

- Opslaan van elementen

**≈ de Python-lijst**

# ArrayList vs Python list

| Array | ArrayList | Python list | Python tuple |
|---|---|---|---|
| **Arrays zien we volgende keer** | *Flexibele grootte* | *Flexibele grootte* | *Vaste grootte* |
| | *Initiële grootte facultatief* | *Niet nodig* | *Bij initialisatie* |
| | `ArrayList<Integer> arrayList = new ArrayList<Integer>();` | `list=[]` | *Creatie en initialisatie samen* |
| | *Initialisatie niet mogelijk* | `list=[1, 2, 3]` | `tuple = (1, 2, 3, 4, 5 )` |
| | `x = arrayList.get(2);` | `x=list[2]` | `x=tuple[2]` |
| | `arrayList.set(1, 5);` | `list[1]=5` | *Veranderen niet mogelijk* |
| | `arrayList.add(7);` | `list.append(7)` | *Niet mogelijk* |
| | `arrayList.size()` | `len(list)` | `len(tuple)` |

# Aanmaken

## Constructor Summary

**ArrayList()**

**Constructs an empty list with an initial capacity of ten.**

**ArrayList(int initialCapacity)**

**Constructs an empty list with the specified initial capacity.**

```
ArrayList<Integer> array = new ArrayList<Integer>();
```

- Bij het aanmaken specifieer je het type
  - Type is dus in feite een 'parameter'

# Java Generics

- Je maakt een klasse, maar laat het exacte type nog onbepaald
  - Gebruiker geeft het type mee bij constructie
- Moet wel een klasse zijn
  - Elk primitief type heeft een klassevariant

| Primitief type | Klasse |
|---|---|
| `int` | `Integer` |
| `float` | `Float` |
| `double` | `Double` |
| `char` | `Char` |
| `boolean` | `Boolean` |

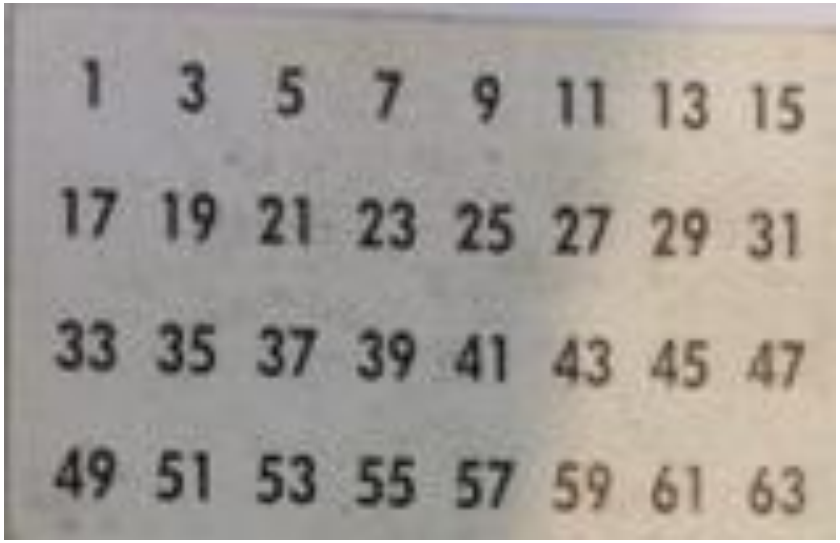
## Method Summary

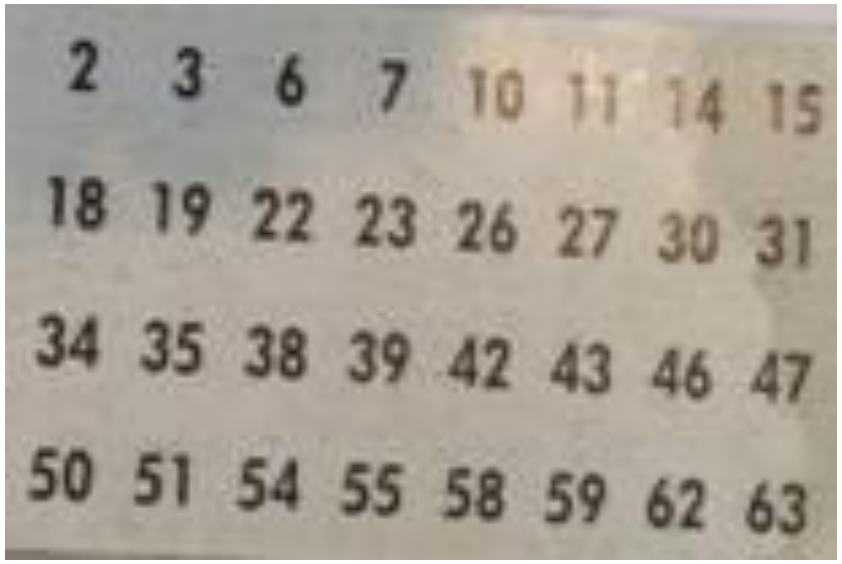
| | |
|---|---|
| boolean | add(E e)<br>Appends the specified element to the end of this list. |
| void | add(int index, E element)<br>Inserts the specified element at the specified position in this list. |
| boolean | contains(Object o)<br>Returns true if this list contains the specified element. |
| E | get(int index)<br>Returns the element at the specified position in this list. |
| int | indexOf(Object o)<br>Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. |
| boolean | isEmpty()<br>Returns true if this list contains no elements. |
| E | remove(int index)<br>Removes the element at the specified position in this list. |
| boolean | remove(Object o)<br>Removes the first occurrence of the specified element from this list, if it is present. |
| E | set(int index, E element)<br>Replaces the element at the specified position in this list with the specified element. |
| int | size() Returns the number of elements in this list. |

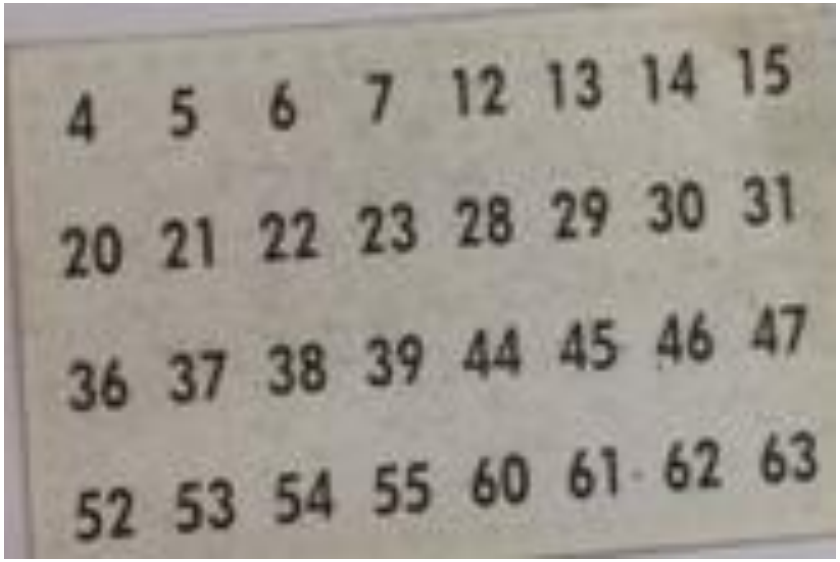
# KIES EEN GEHEIM GETAL TUSSEN 1 & 64

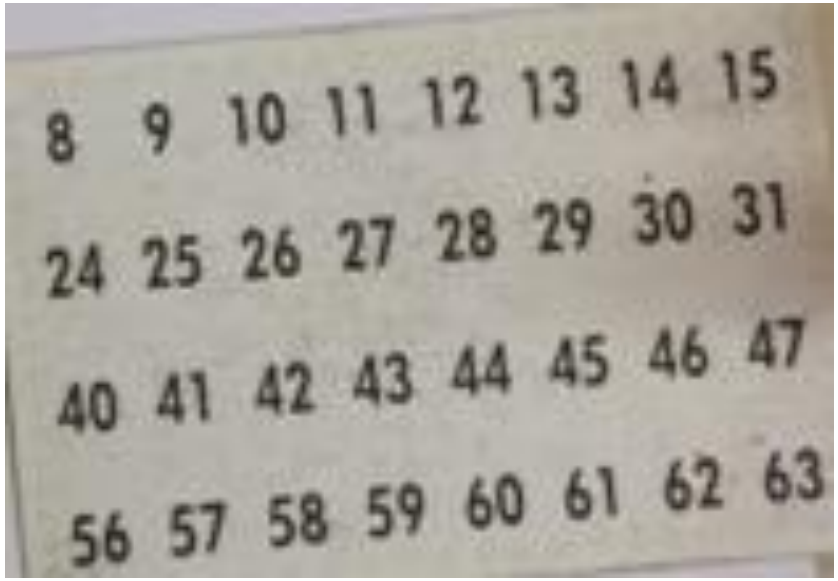
6 kaartjes – 6 ja/nee vragen

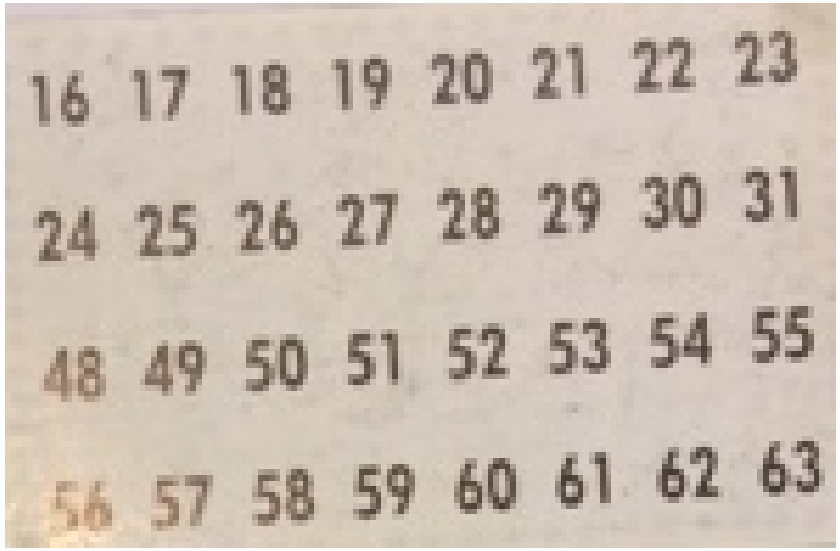
*Benjamin Jadot, 1e bachelor industrieel ingenieur, 2016*

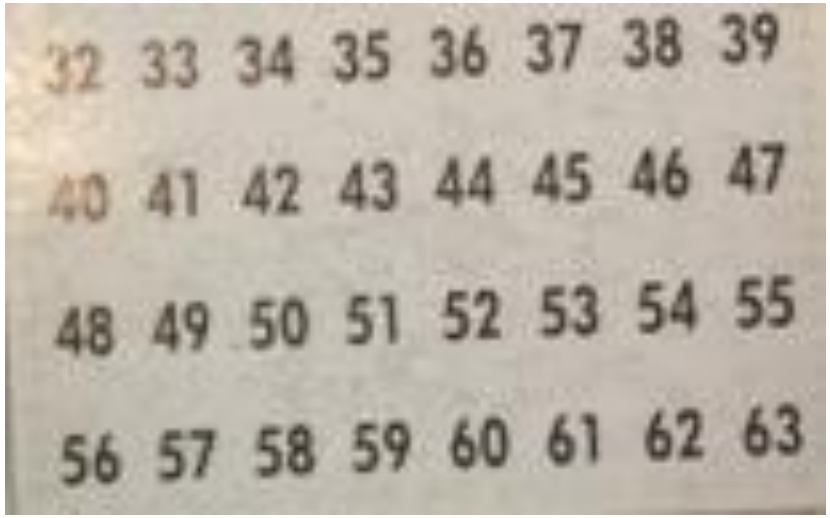

1 3 5 7 9 11 13 15
17 19 21 23 25 27 29 31
33 35 37 39 41 43 45 47
49 51 53 55 57 59 61 63


2 3 6 7 10 11 14 15
18 19 22 23 26 27 30 31
34 35 38 39 42 43 46 47
50 51 54 55 58 59 62 63


4 5 6 7 12 13 14 15
20 21 22 23 28 29 30 31
36 37 38 39 44 45 46 47
52 53 54 55 60 61 62 63


8 9 10 11 12 13 14 15
24 25 26 27 28 29 30 31
40 41 42 43 44 45 46 47
56 57 58 59 60 61 62 63


16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63


32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63

# Leibniz

$$\frac{\partial f}{\partial x}$$

**1646 – 1716**

# Calculemus!

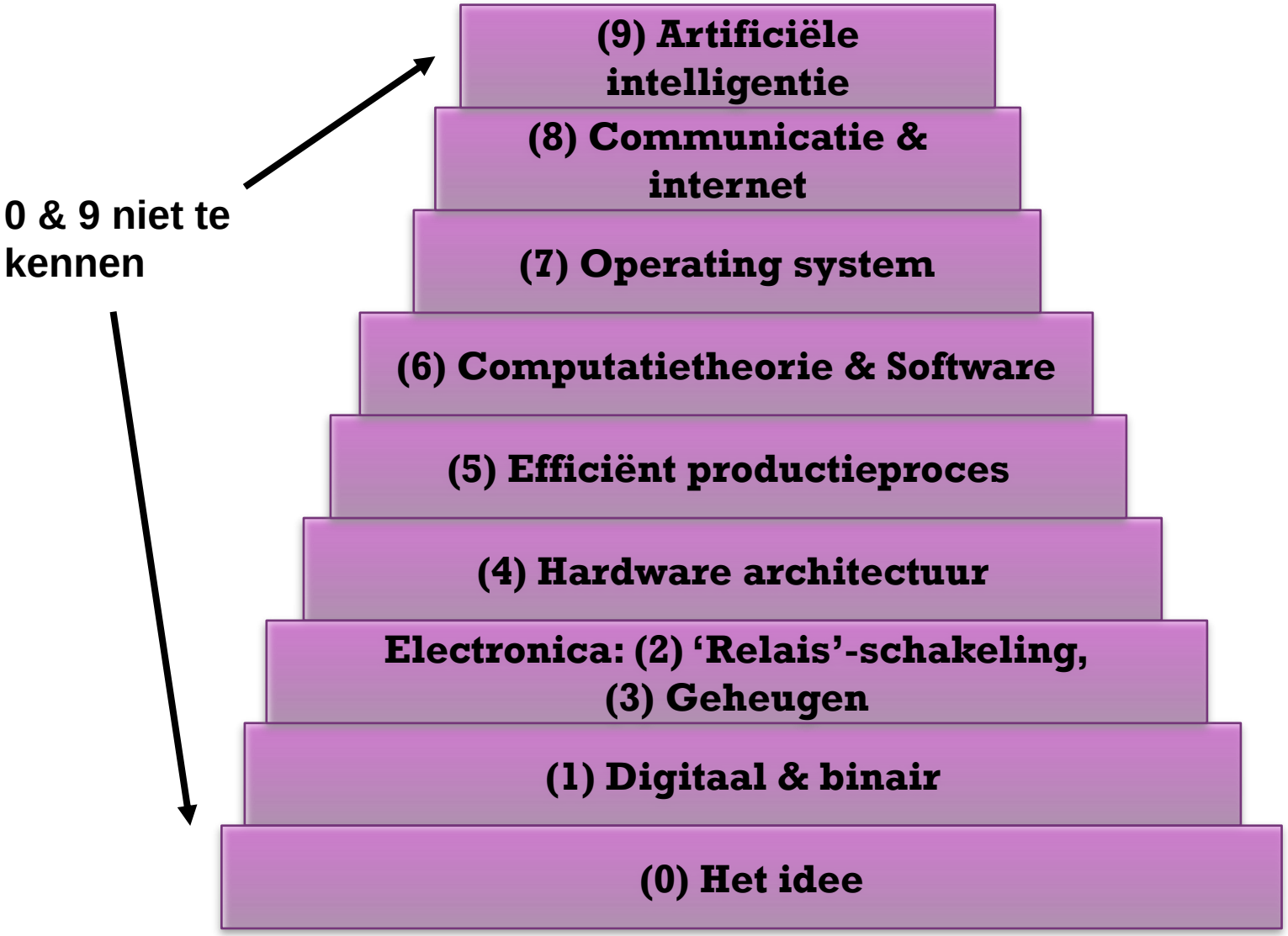
# Berechnen wir!

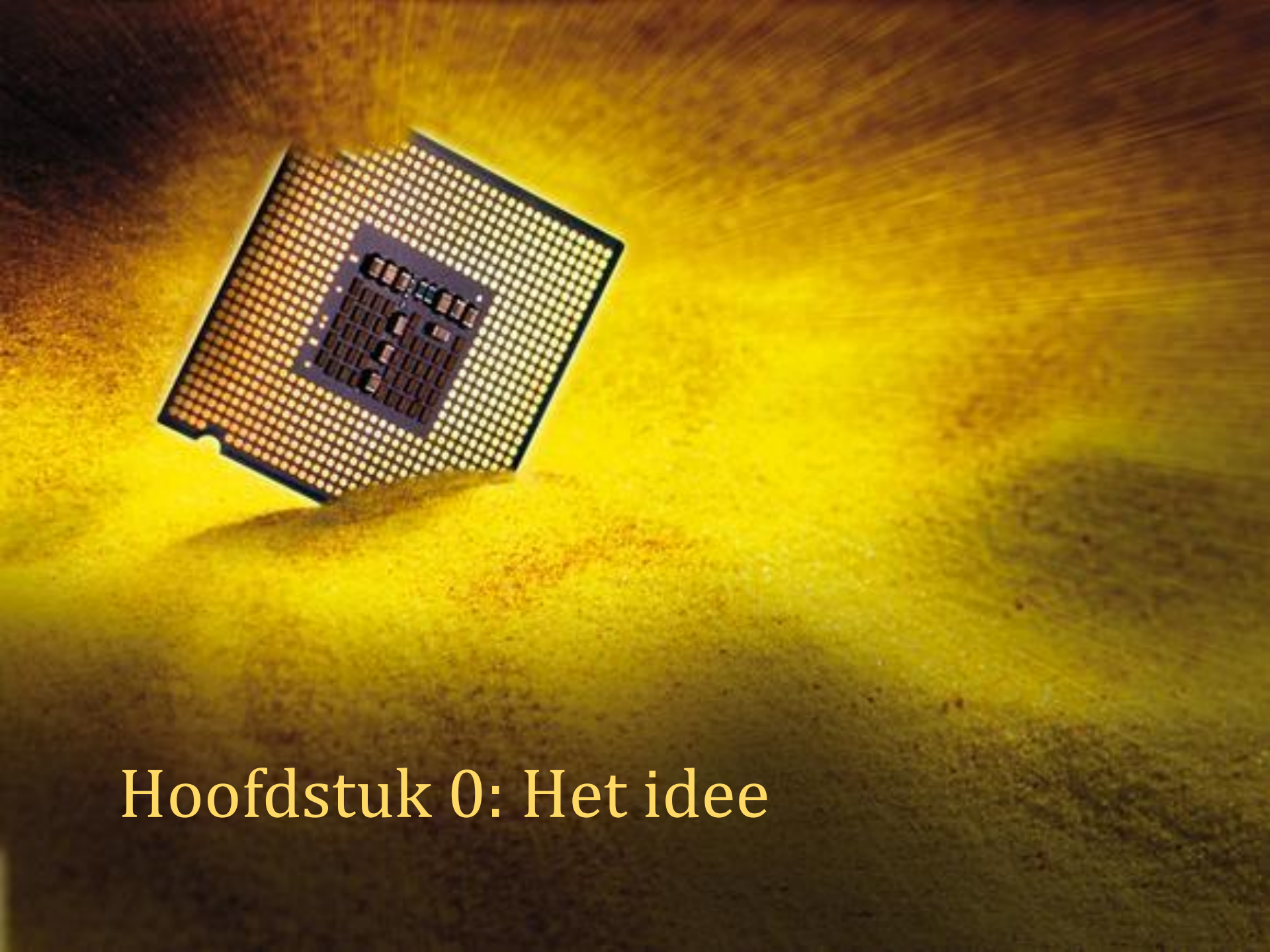
# Leibniz

**1646 – 1716**

- Droomde van de "**Calculus ratiocinator**"
  - Een logisch denkend apparaat

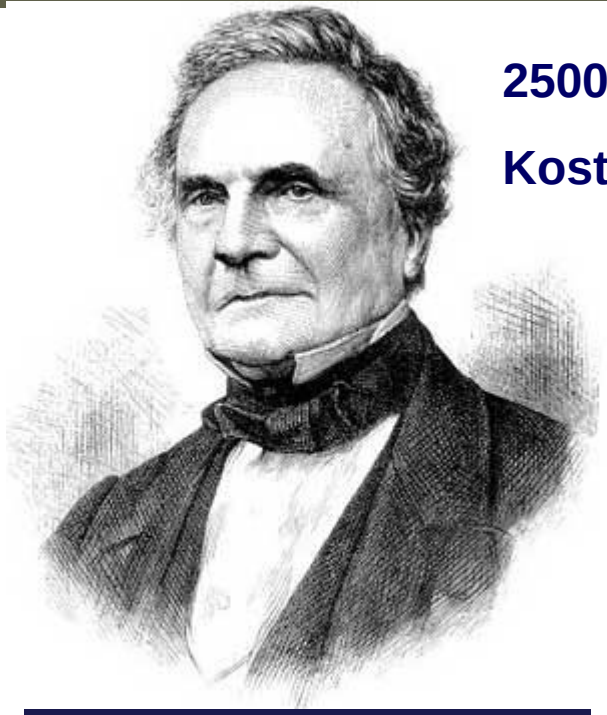
# Waarmaken van Leibniz's droom





# Hoofdstuk 0: Het idee

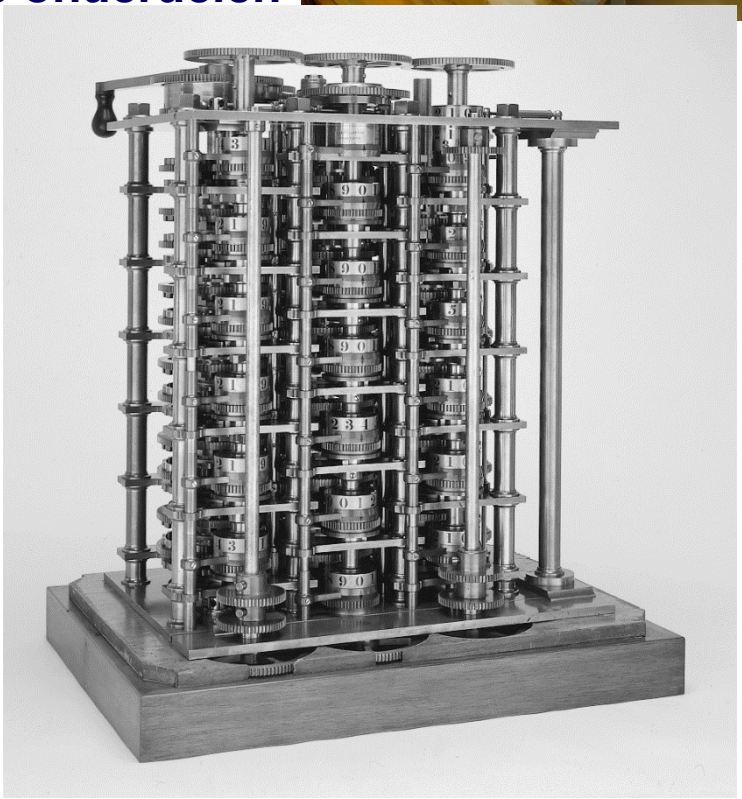
# Eerste computer, mechanisch



**Charles Babbage**
**1791-1871**

**25000 mechanische onderdelen**

**Kostprijs: 17470 £**



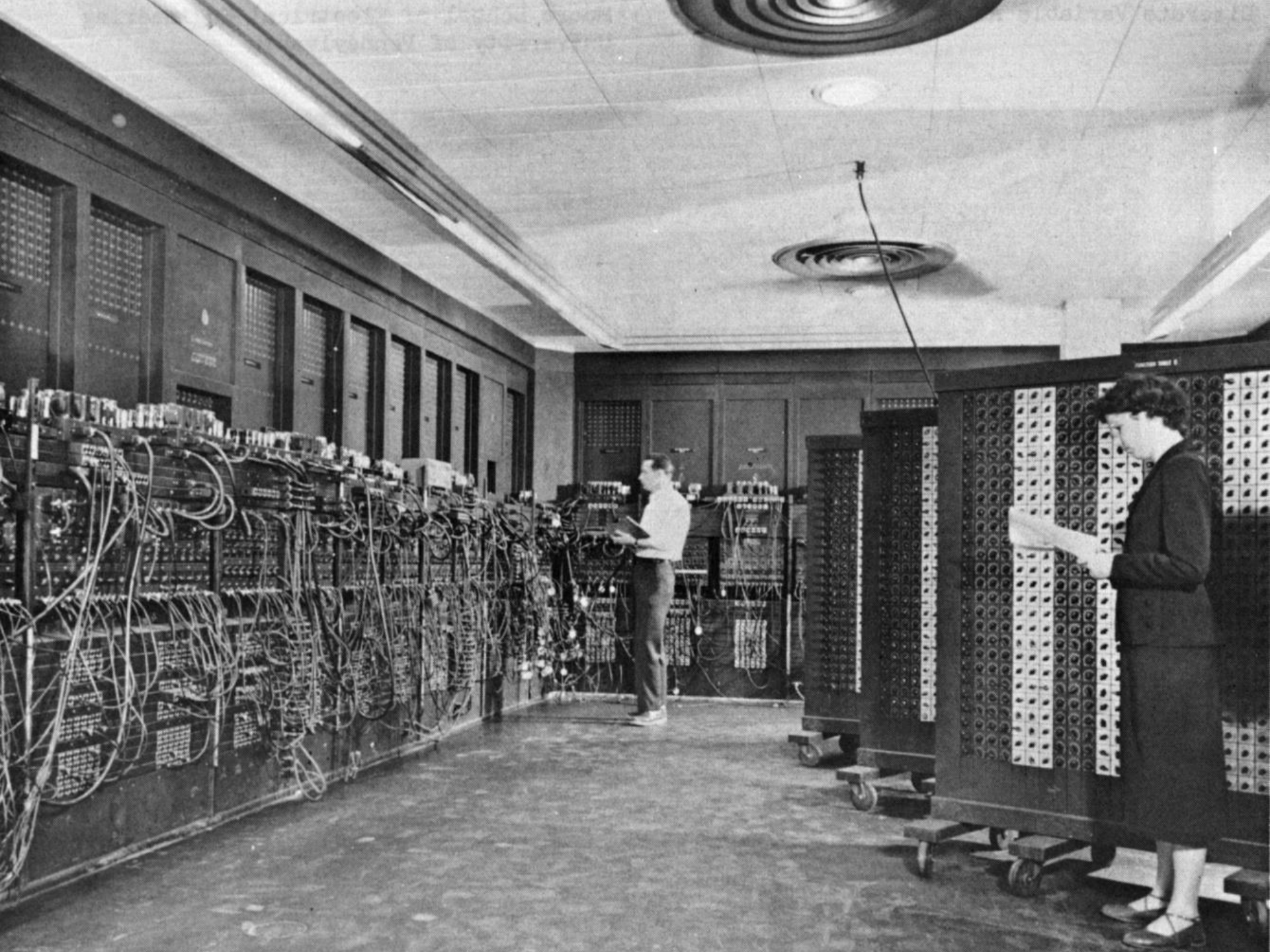
# De eerste computer-wetenschapper & visionair

**Ada Lovelace**
**1815 – 1852**

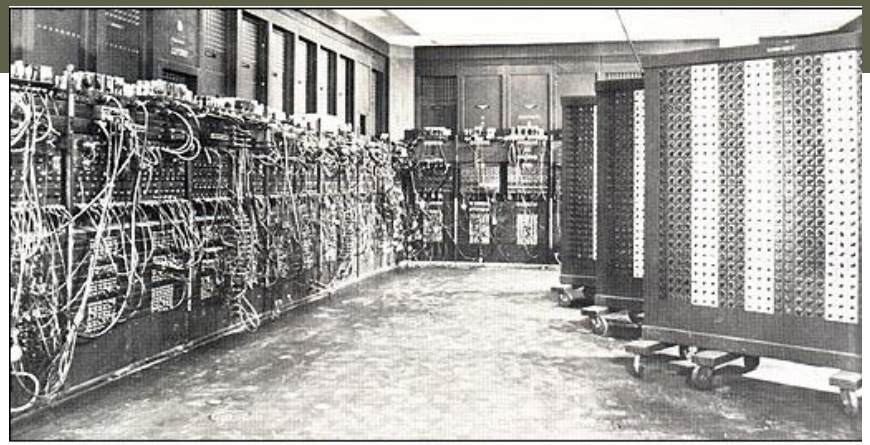
- “Poëtische wetenschap”.
- Ziet kracht van Babbage’s machine
- Machine kan ook met symbolen om => mentale processen!
- Beschrijft wat software en een algoritme is: *stap-voor-stap bewerkingen doen om tot de oplossing te komen*.
- Voorziet het inzetten van herbruikbare functies.
- Schrijft het eerste programma bestemd voor de universele computer van Charles Babbage.
- Vraagt zich af  of "machines ooit zullen kunnen denken?" Ze dacht van niet.

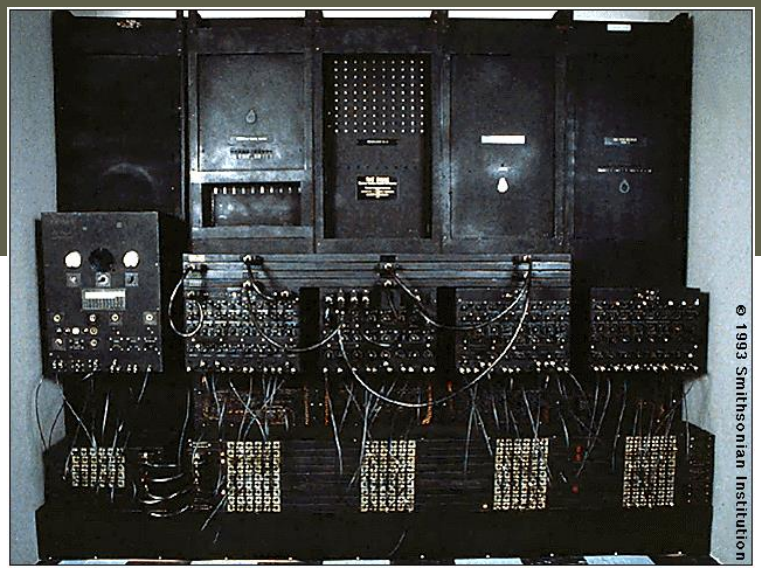
Ada ving een glimp van de toekomst...

*In de 1980s is de programmeertaal ‘Ada’ ter ere van haar gemaakt*

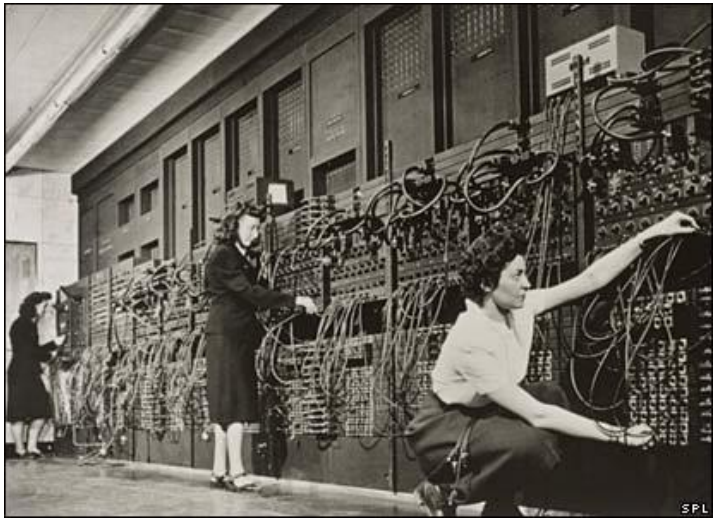


# ENIAC





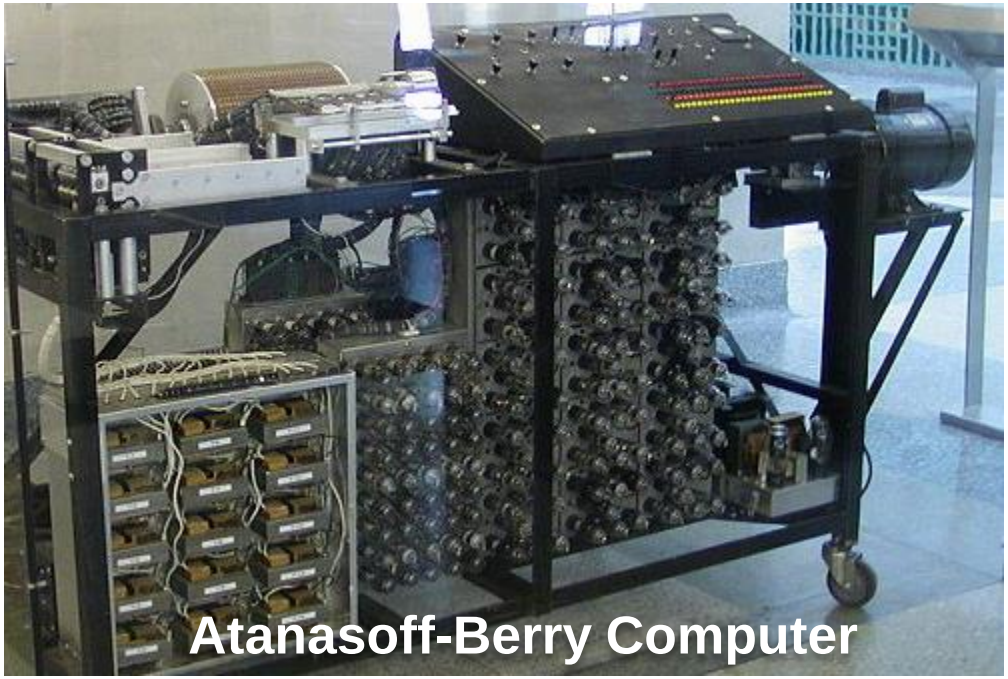

Eerste computer: WOII




**John Mauchly and John Eckert, 1945**

**John Atanasoff**
**1903 – 1995**

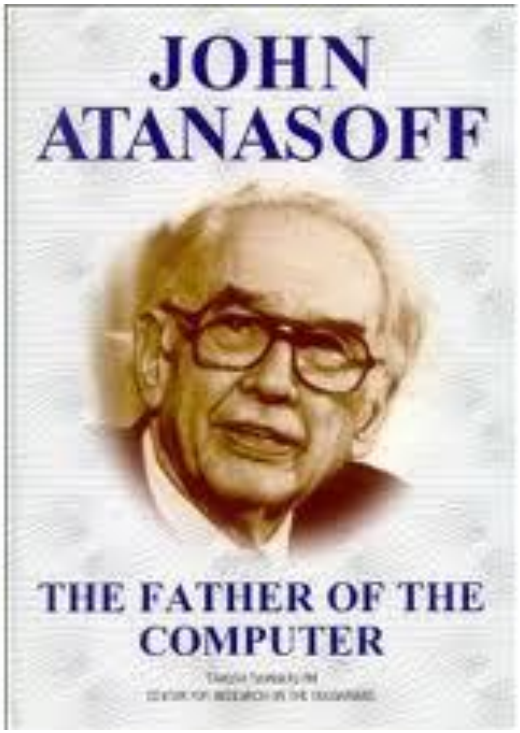
**Clifford Berry**
**1918 – 1963**



**Atanasoff-Berry Computer**

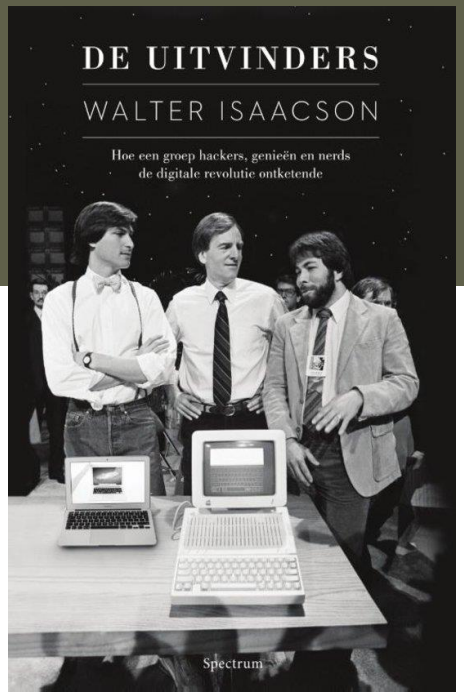
# Honeywell v. Sperry Rand case, 1973

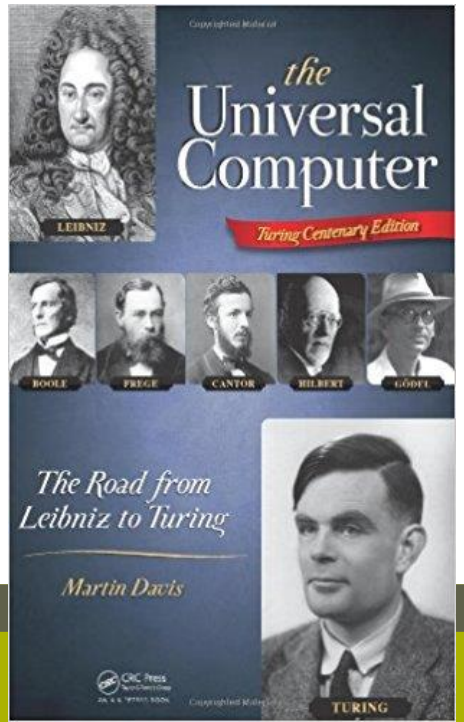

Honeywell v. Sperry Rand case, 1973


JOHN
ATANASOFF
THE FATHER OF THE
COMPUTER

# Deel III: referenties

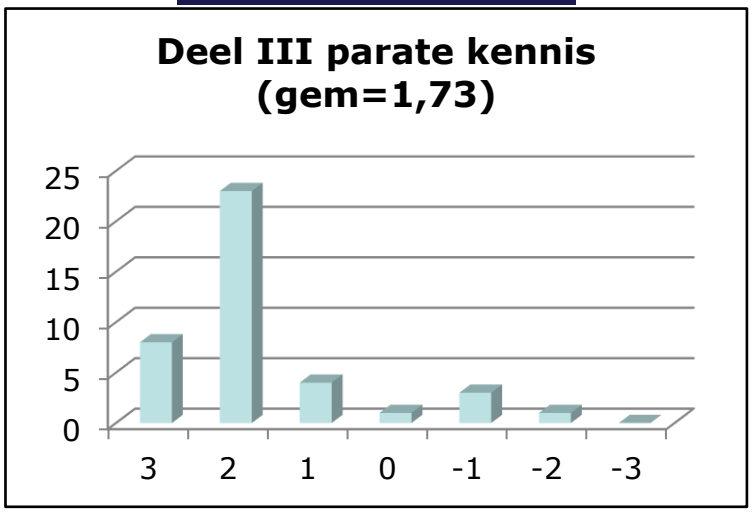
- "The Innovators: How a Group of Hackers, Geniuses and Geeks Created the Digital Revolution", by Walter Isaacson, 2014. *Aanwezig in de VUB-bibliotheek.*
  - In het Nederlands: "De uitvinders, hoe een groep hackers, genieën en nerds de digitale revolutie ontketende."
- "The Universal Computer. The Road from Leibniz to Turing", by Martin Davis, 2000. *Aanwezig in de VUB-bibliotheek.*
- Links op parallel.vub.ac.be



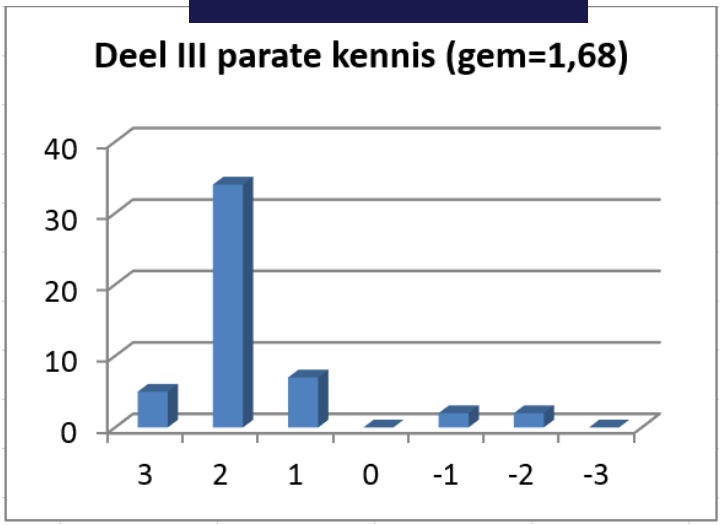



# Deel III: parate kennis

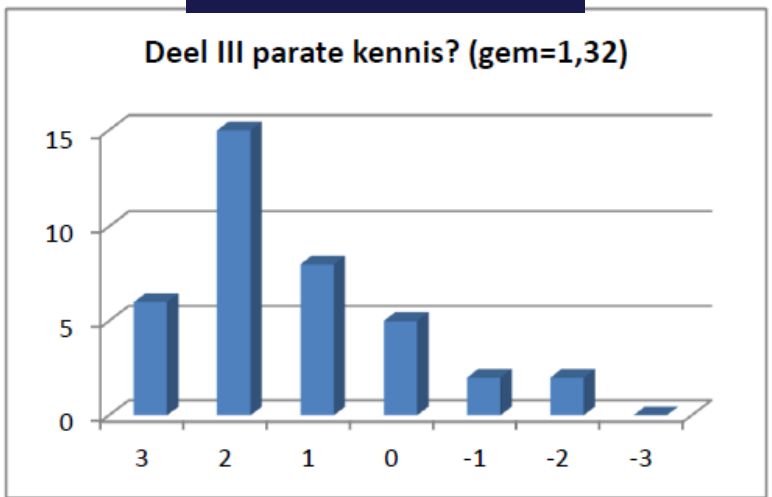
## 2012 – 2013



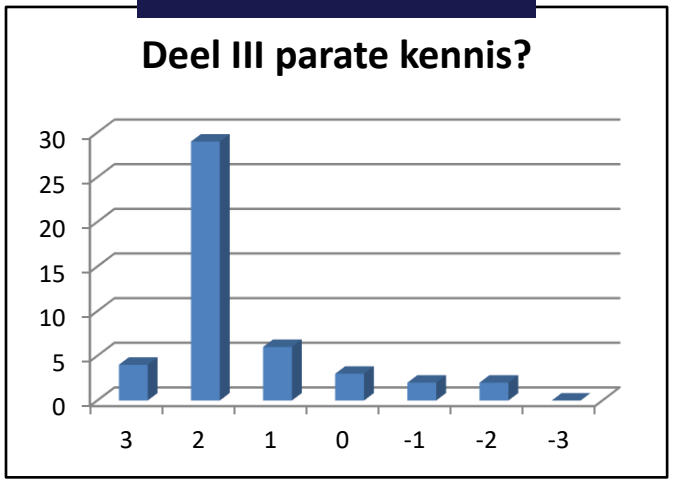

## 2013 – 2014




## 2014 – 2015




## 2015 – 2016



# Hoofdstuk 1: Van analoog naar digitaal

# Analoog rekenen

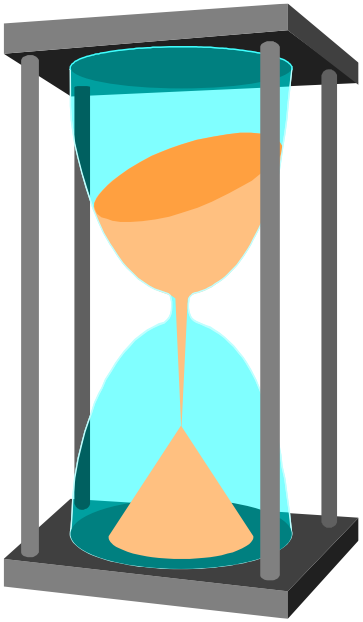
- Gebruik makend van fysische grootheden
  - Cf Babbage
- **Analoge electronica**

  **http://www.chem.uoa.gr/applets/appletopamps/appl_opamps2.html**
  **http://www.falstad.com/circuit/** (***bekijken we later***)
- De rekenlat of 'slide rule' was lange tijd hèt rekeninstrument van de ingenieur

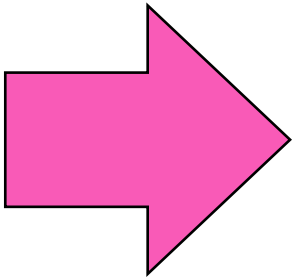
**http://www.antiquark.com/sliderule/sim/n909es/virtual-n909-es.html**

# Digitale Technologie

*Informatie wordt geëncodeerd als ‘digits’*

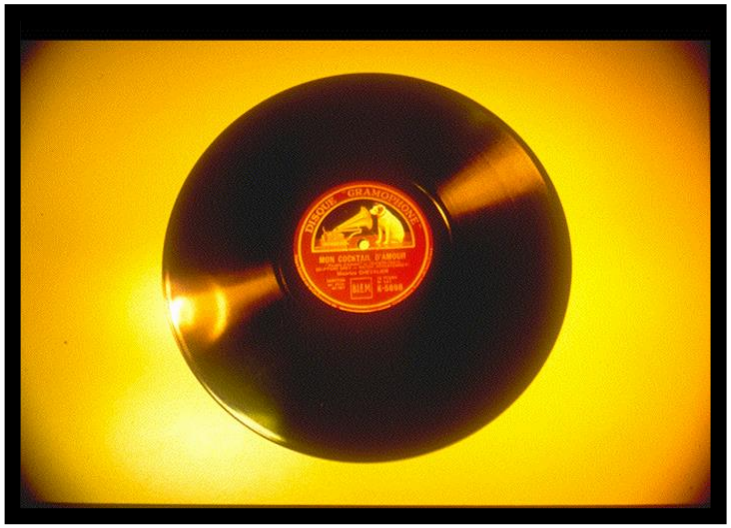


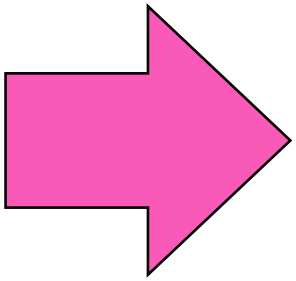
**Analoog**

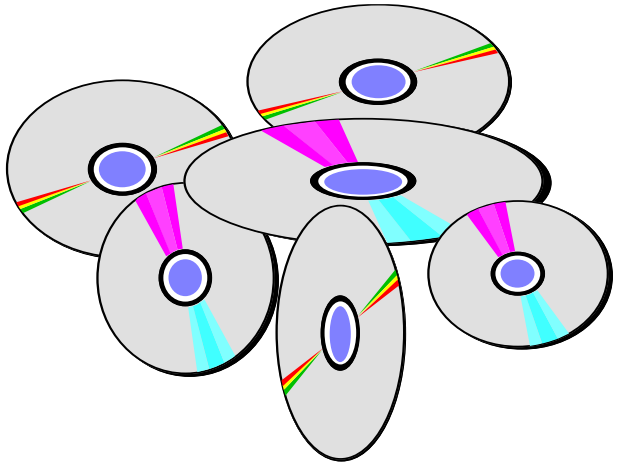



**Digitaal**

# Muziek







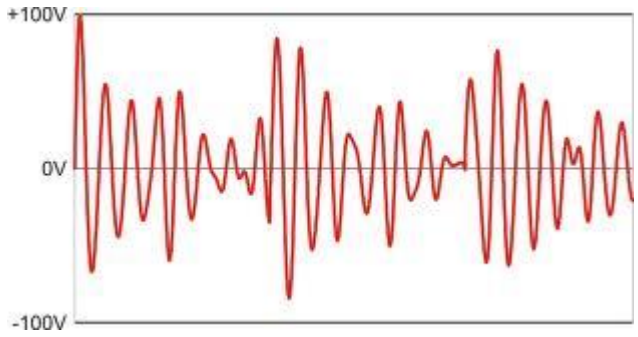
**Analoog**

**Digitaal**

informatie gecodeerd door middel van een continu veranderlijke fysische grootheid

# Muziek

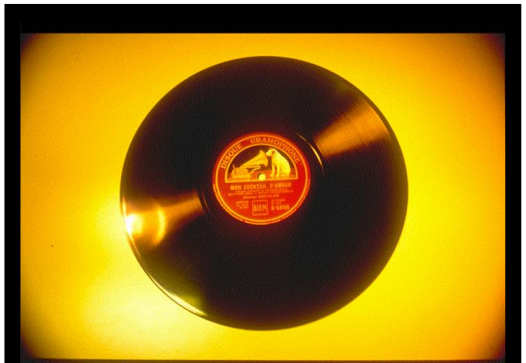
**Mens hoort tussen 20 en 20,000 Hz**




```
-096
+057
+164
+210
+219
+216
+165
-003
-117
-183
-138
-067
```

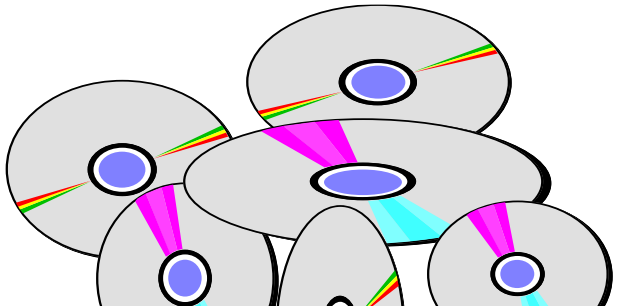
informatie gecodeerd door middel van een continu-veranderlijke fysische grootheid

## Analoog

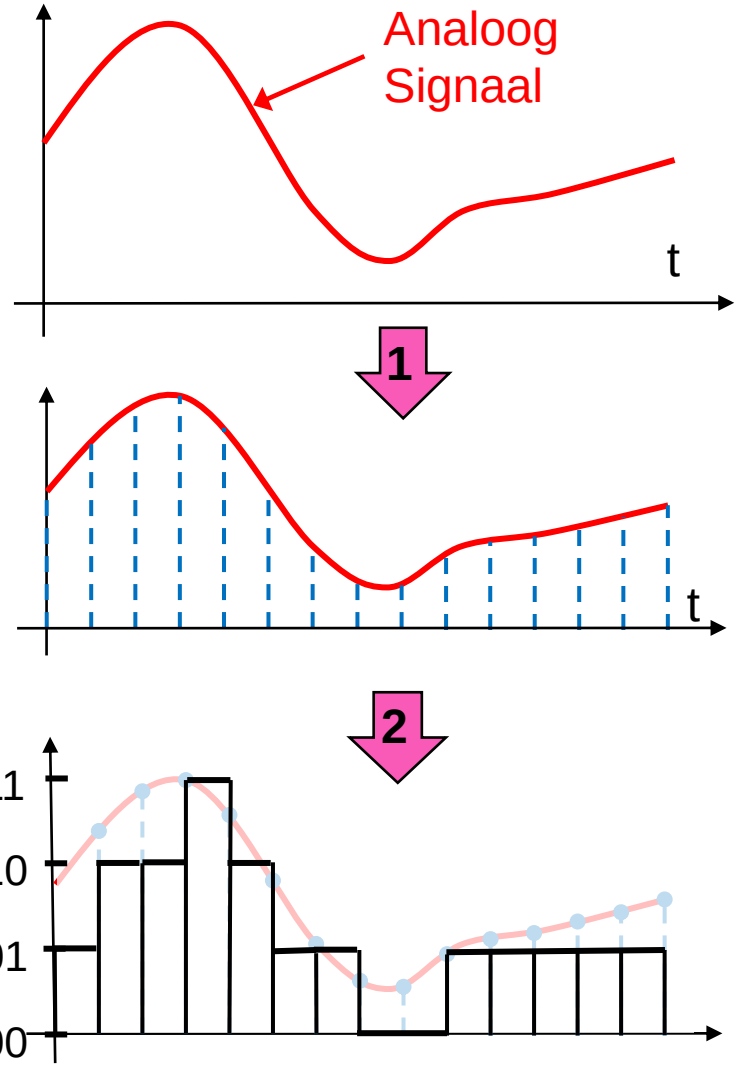


## Digitaal (CD)

**(44100 metingen/s)**



# Digitaliseren

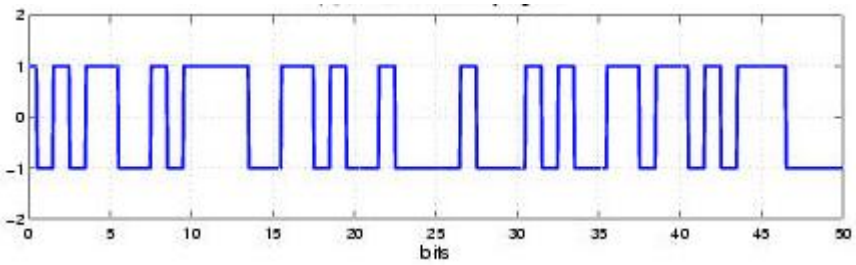



**(1) Discretizeren**:
*samplen* in de tijd

**(2) Quantizatie**: de amplitude in elk samplepunt voorstellen als een binair getal

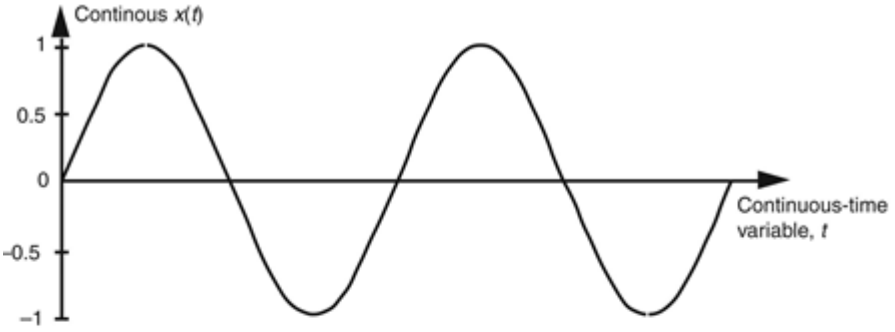
**(3)** Alle bits achter elkaar

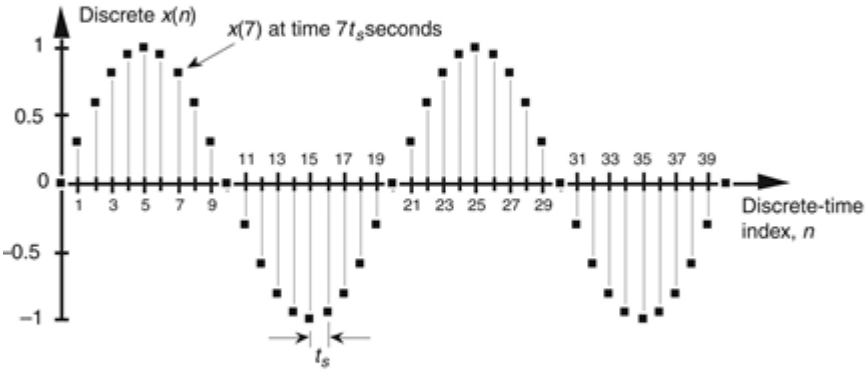
**Digitaal signaal**

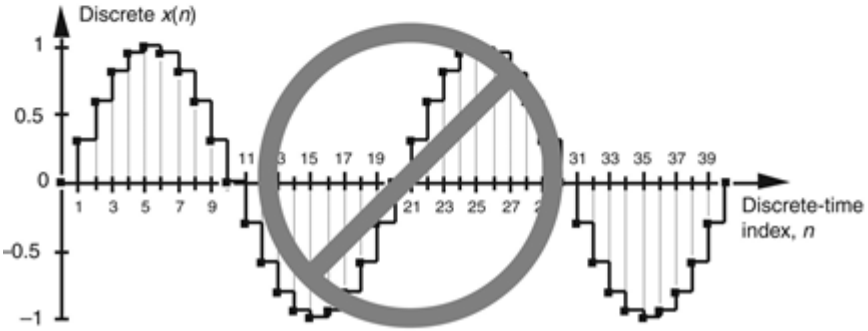



Digitaliseren van een analoog signaal (bvb muziek) gebeurt door op geregelde tijdstippen (discretizeren) de amplitude van het signaal weg te schrijven als een aantal bits (quantizeren). Alle bits achter elkaar geven het digitaal signaal.

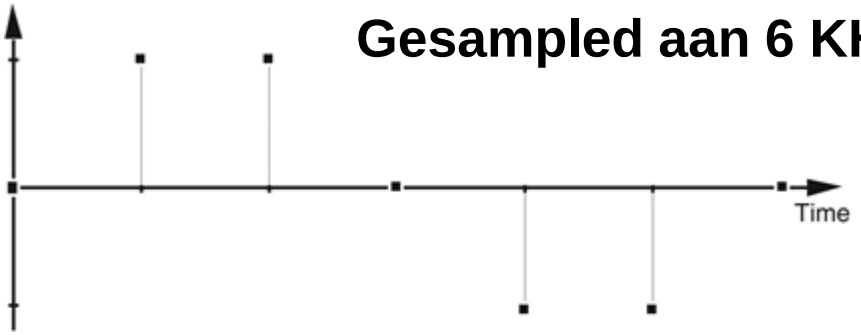
# Reconstructie van analoog signaal

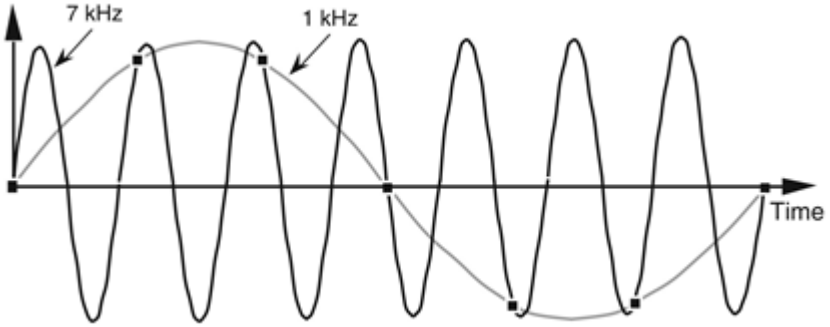


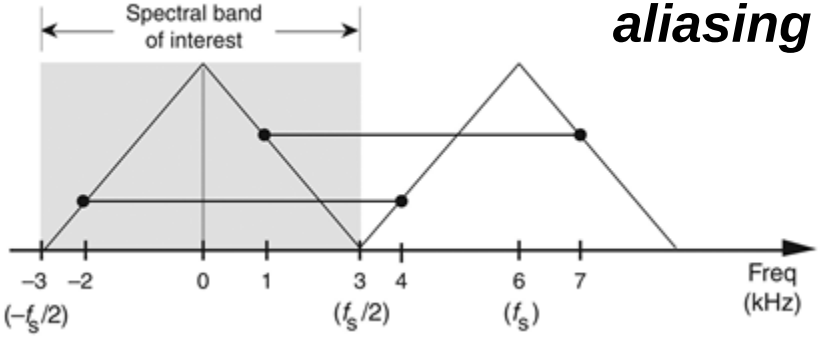




**Gesampled aan 6 KHz**

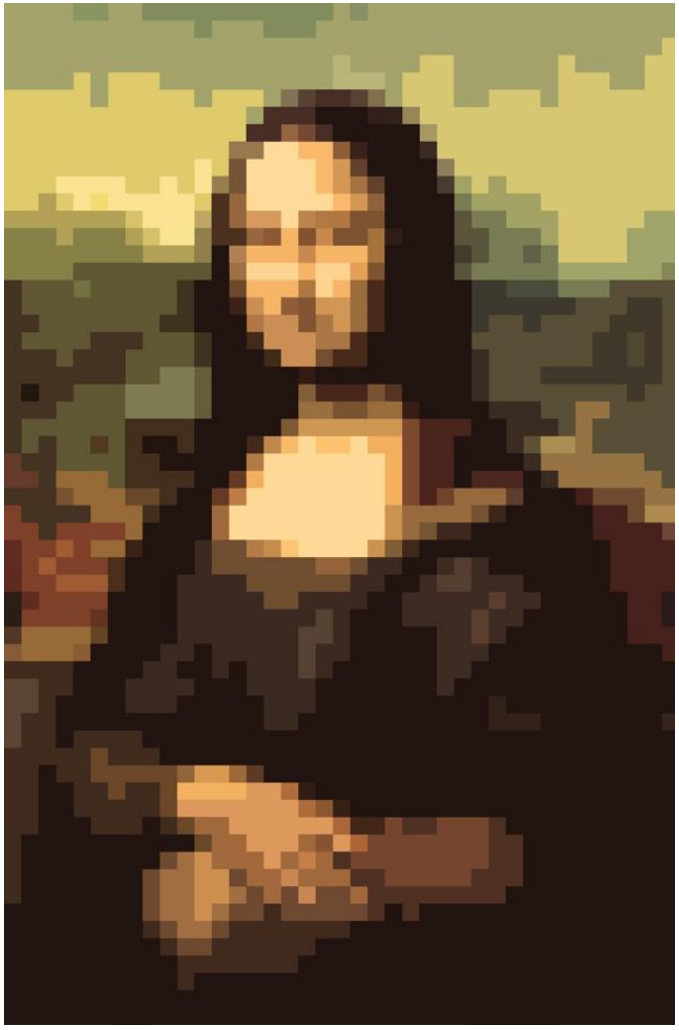



***aliasing***



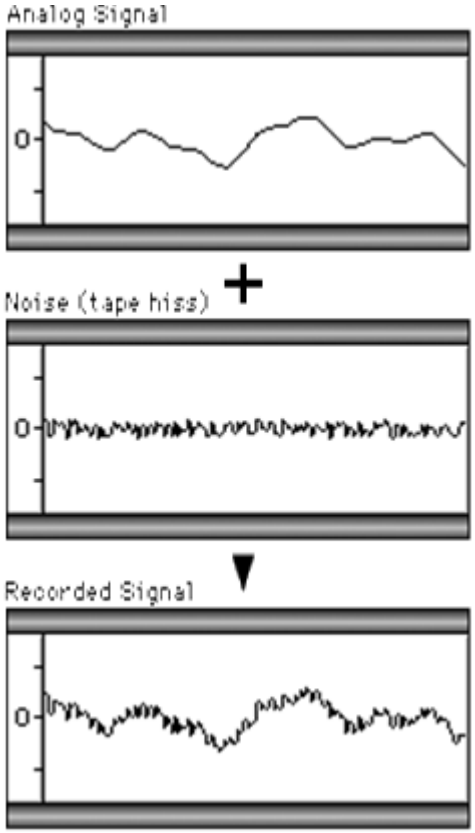
# Digitalisatie van beelden

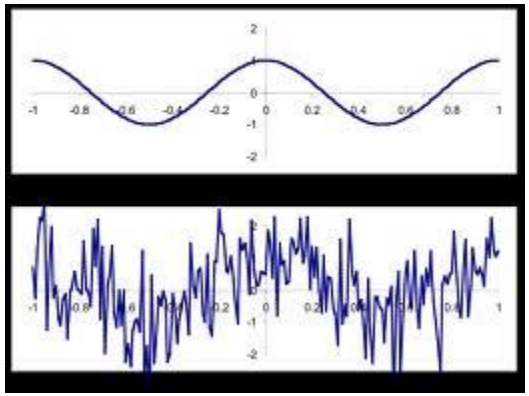


voor elk punt (pixel) de kleur opslaan
=> **bitmap**

# Effect van signaal-vervormingen of ruis (noise)

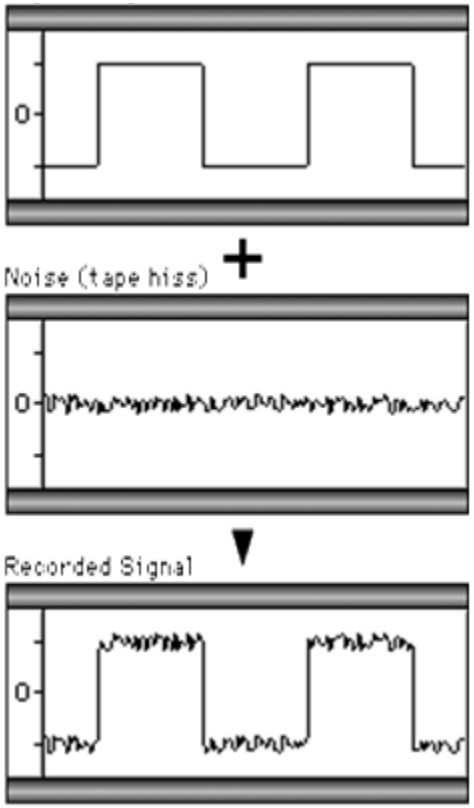
## Analoog signaal



**Informatieverlies**

## Digitaal signaal




**Informatie nog te herkennen**

# Waarom digitaal?

- Unambiguë signalen, immuun voor ruis.
  - Als ruis onder een aanvaardbaar niveau is kan men steeds nog het origineel signaal herkennen. Het is immers een 0 of een 1.
- Perfecte copieën kunnen gemaakt worden.
- Bewerkingen zonder informatieverlies
  - *Digitale signaalverwerking is mogelijk!*
- Simpel, gemakkelijk te maken.
- ➡ Digitale componenten zijn goedkoop, klein, betrouwbaar en men kan er miljoenen op een klein gebied plaatsen.

Alles dat voorgesteld kan worden door één of ander signaal/patroon, kan voorgesteld worden door bits.

# Go digital! Go binary!

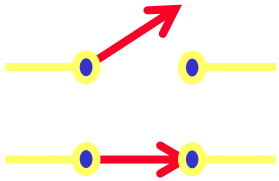
- Van analoog naar digitaal en binair…
- Leibniz had er al aan gedacht (zoals te zien is in zijn nota’s)
- Ook was dit de basis van de computer van Atanasoff en Berry

# Waarom binair?

- *De ultieme essentie van informatie: 0 of 1 (to be or not to be)*
- In digitale toestellen gebruikt men meestal het binaire stelsel omdat binaire cijfers eenvoudig kunnen voorgesteld worden door fysische grootheden zoals een positieve of een negatieve spanning, een magnetisch veld in de ene of de andere richting of een open of gesloten schakelaar.
- Eenvoudige reconstructie: groter of kleiner dan een drempelwaarde (*threshold*) bepaalt de bitwaarde

# Binaire representatie

- Een binair getal (bit) kan voorgesteld worden door 2 voltages die gegeven kunnen worden door een switch:
  - Waarde 0 = 0 Volt = switch open
  - Waarde 1 = 5 Volt = switch gesloten



- Een getal van *n* bits kan $2^n$ waarden aannemen
  - 2 bits : 4 combinaties 00 01 10 11
  - 3 bits : 8 combinaties 000 001 010 011 100 101 110 111
  - 8 bits (= 1 byte) 256 combinaties
  - 16 bits: 65 536 combinaties
  - 32 bits: 4 294 967 296 combinaties

$2^{10} \approx 1000$
$2^{20} \approx 10^6$
$2^{30} \approx 10^9$

# Java basic datatypes

| Type | Grootte | Minimum | Maximum | Precisie |
|---|---|---|---|---|
| byte | 8 bits | -128 | 127 | |
| short | 16 bits | -32 768 | 32 767 | |
| int | 32 bits | $-2^{31}$ | $2^{31}$ -1 | |
| long | 64 bits | $-2^{63}$ | $2^{63}$ -1 | |
| char | 16 bits | 0 | $2^{16}$ - 1 | |
| float | 32 bits | $-3.4 \times 10^{38}$ | $3.4 \times 10^{38}$ | 6-9 cijfers |
| double | 64 bits | $-1.7 \times 10^{308}$ | $1.7 \times 10^{308}$ | 15-17 cijfers |

# Voorstelling van karakters

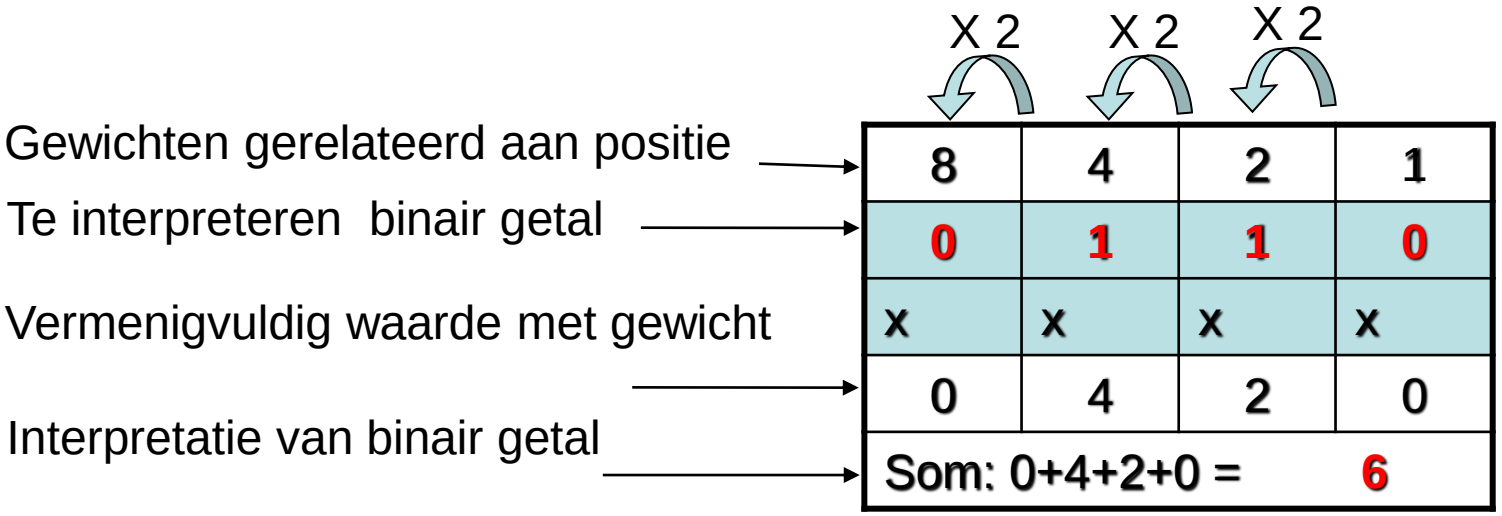
De huidige computers gebruiken gestandaardiseerde tabellen waarin de alfanumerieke tekens die op het toetsenbord staan geassocieerd worden met binaire getallen. De meestgebruikte is de ASCII tabel.

De 128 tekens van hiernaast kunnen voorgesteld worden met 7 bits ($2^7$=128)

**TABLE 3**
**ASCII CHARACTER CODES (DECIMAL)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | Ctrl-@ | 32 | Space | 64 | @ | 96 | ` |
| 1 | Ctrl-A | 33 | ! | 65 | A | 97 | a |
| 2 | Ctrl-B | 34 | " | 66 | B | 98 | b |
| 3 | Ctrl-C | 35 | # | 67 | C | 99 | c |
| 4 | Ctrl-D | 36 | $ | 68 | D | 100 | d |
| 5 | Ctrl-E | 37 | % | 69 | E | 101 | e |
| 6 | Ctrl-F | 38 | & | 70 | F | 102 | f |
| 7 | Ctrl-G | 39 | ' | 71 | G | 103 | g |
| 8 | Backspace | 40 | ( | 72 | H | 104 | h |
| 9 | Tab | 41 | ) | 73 | I | 105 | i |
| 10 | Ctrl-J | 42 | * | 74 | J | 106 | j |
| 11 | Ctrl-K | 43 | + | 75 | K | 107 | k |
| 12 | Ctrl-L | 44 | , | 76 | L | 108 | l |
| 13 | Return | 45 | - | 77 | M | 109 | m |
| 14 | Ctrl-N | 46 | . | 78 | N | 110 | n |
| 15 | Ctrl-O | 47 | / | 79 | O | 111 | o |
| 16 | Ctrl-P | 48 | 0 | 80 | P | 112 | p |
| 17 | Ctrl-Q | 49 | 1 | 81 | Q | 113 | q |
| 18 | Ctrl-R | 50 | 2 | 82 | R | 114 | r |
| 19 | Ctrl-S | 51 | 3 | 83 | S | 115 | s |
| 20 | Ctrl-T | 52 | 4 | 84 | T | 116 | t |
| 21 | Ctrl-U | 53 | 5 | 85 | U | 117 | u |
| 22 | Ctrl-V | 54 | 6 | 86 | V | 118 | v |
| 23 | Ctrl-W | 55 | 7 | 87 | W | 119 | w |
| 24 | Ctrl-X | 56 | 8 | 88 | X | 120 | x |
| 25 | Ctrl-Y | 57 | 9 | 89 | Y | 121 | y |
| 26 | Ctrl-Z | 58 | : | 90 | Z | 122 | z |
| 27 | Escape | 59 | ; | 91 | [ | 123 | { |
| 28 | Ctrl-\ | 60 | < | 92 | \ | 124 | \| |
| 29 | Ctrl-] | 61 | = | 93 | ] | 125 | } |
| 30 | Ctrl-^ | 62 | > | 94 | ^ | 126 | ~ |
| 31 | Ctrl-_ | 63 | ? | 95 | _ | 127 | Delet |

# Binair talstelsel




| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | Decimale Interpretatie |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 130 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 85 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 41 |
| $a_7$ | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | $\sum_{i=0}^{n} a_i \cdot 2^i$ |

# Enkele oefeningen

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | Decimale Interpretatie |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | |

Opgave 1: Bereken de decimale interpretatie van de bovenstaande binaire getallen.
Opgave 2: Zet om naar binair: 6, 17, 31, 255
Opgave 3: Zet uw geboortedatum om in binair formaat (dag, maand en jaar apart).
Opgave 4: Hoeveel decimale getallen kunnen er voorgesteld worden aan de hand van een 10-bit binair woord?

# Grootte-ordes

| | | Bytes | |
|---|---|---|---|
| $10^3$ | Kilo | kB | $\approx 2^{10} = 1024$ |
| $10^6$ | Mega | MB | $\approx 2^{20}$ |
| $10^9$ | Giga | GB | $\approx 2^{30}$ |
| $10^{12}$ | Tera | TB | $\approx 2^{40}$ |
| $10^{15}$ | Peta | PB | $\approx 2^{50}$ |
| $10^{18}$ | Exa | EB | $\approx 2^{60}$ |

Vergeet nooit meer: $2^{10} \approx 1000$

# Binair rekenen?

- 1 input => 1 output
  - Maar 1 niet-triviale functie: NOT
- 2 inputs => 1 output

| | 0 | 1 |
|---|---|---|
| **0** | ? | ? |
| **1** | ? | ? |

**$2^4$ = 16 mogelijkheden**

- Genoeg voor een basis: AND, OR, XOR, NOT

*Alle functies (met meer inputs) zijn samen te stellen uit deze!!*

# 2 inputs => 1 output: 16 mogelijkheden

| | 0 | 1 |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 0 | 0 |

| | 0 | 1 |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 0 | 1 |

| | 0 | 1 |
|---|---|---|
| **0** | 1 | 1 |
| **1** | 0 | 0 |

| | 0 | 1 |
|---|---|---|
| **0** | 1 | 1 |
| **1** | 1 | 1 |

| | 0 | 1 |
|---|---|---|
| **0** | 1 | 0 |
| **1** | 1 | 0 |

| | 0 | 1 |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 1 | 1 |

**triviaal**

| | 0 | 1 |
|---|---|---|
| **0** | 1 | 0 |
| **1** | 0 | 1 |

| | 0 | 1 |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 0 |

**exclusive or (xor)**

**and**

| | 0 | 1 |
|---|---|---|
| **0** | 1 | 0 |
| **1** | 0 | 0 |

| | 0 | 1 |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 0 | 1 |

| | 0 | 1 |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 1 | 0 |

| | 0 | 1 |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 0 | 0 |

| | 0 | 1 |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 1 |

| | 0 | 1 |
|---|---|---|
| **0** | 1 | 1 |
| **1** | 1 | 0 |

| | 0 | 1 |
|---|---|---|
| **0** | 1 | 1 |
| **1** | 0 | 1 |

| | 0 | 1 |
|---|---|---|
| **0** | 1 | 0 |
| **1** | 1 | 1 |

**or**

# basisoperaties op bits en bytes

**And**

| a \ b | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

**Or**

| a \ b | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

**Exclusive or**

| a \ b | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| **a=60** | | **0** | **0** | **1** | **1** | **1** | **1** | **0** | **0** |
|---|---|---|---|---|---|---|---|---|---|
| **b=13** | | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **1** |
| **~a** | **Not** | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| **a & b** | **And** | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **a \| b** | **Or** | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| **a ^ b** | **Exclusive or** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| **a << 2** | **Shift left** | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| **a >> 2** | **Shift right** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Zie programma **`BitwiseOperators`** in package **`voorbeelden`**

# Programma van vorige tabel

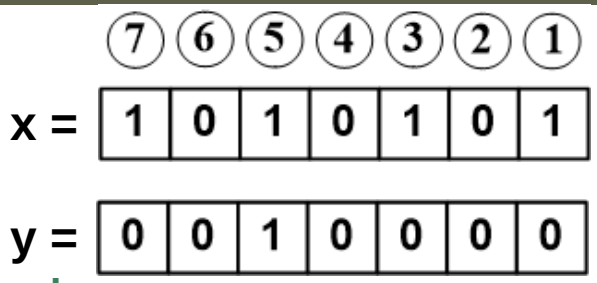
Zie programma **BitwiseOperators** in package **voorbeelden**

```
public class BitwiseOperators {
    public static void main(String[] args) {
        int a = Integer.parseInt("00001101", 2); //2= basis van talstelse
        int b = 13; /* 13 = 0000 1101 */        /* a = 0011 1100 = 60 */
        int c = 0;

        c = a & b;          /* 12 = 0000 1100  and */
        System.out.println("a & b = " + c );

        c = a | b;          /* 61 = 0011 1101  or */
        System.out.println("a | b = " + c );

        c = a ^ b;          /* 49 = 0011 0001  exclusive or (xor) */
        System.out.println("a ^ b = " + c );

        c = ~a;             /*-61 = 1100 0011  not */
        System.out.println("~a = " + c );

        c = a << 2;         /* 240 = 1111 0000  shift left */
        System.out.println("a << 2 = " + c );

        c = a >> 2;         /* 15 = 1111        shift right */
        System.out.println("a >> 2  = " + c );
    }
}
```

# Enkele toepassingen

Gegeven: integers x & i

```
int x = 85, i = 4;
```




1. "Is de (i+1)-de bit van x gelijk aan 1?"

```
int y = 1 << i; // y wordt een masker genoemd
if ((x & y) > 0)
   System.out.println("De "+(i+1)+"de bit van" +x+" is 1.");
else
    System.out.println("De "+(i+1)+"de bit van" + x +" is 0.");
```

2. "Zet de (i+1)-de bit van x op 1"

```
i = 1; // we willen 2e bit op 1 zetten
y = 1 << i; // een masker met een 1 op plaats i
x |= y; // is identiek aan x = x | y;
System.out.println("x = "+x+": "+(i+1)+ "de bit is nu 1.");
```

3. "Wat zijn de waarden van de 4e tot en met de 6e bit?"

```
int z = (x >> 3) & 7; // 7 is hier ook een masker: 111
System.out.println("Bits 4 tot en met 6 van "
+Integer.toBinaryString(x)+" zijn "+Integer.toBinaryString(z));
```

# Hexadecimaal talstelsel

Om de notatie van binaire getallen te verkorten wordt ook de ***hexadecimale notatie*** gebruikt. Een **hexadecimaal cijfer** beschrijft 4 bits en wordt voorgesteld door 0, 1, ..., 9, A, ..., F.

| Binair | Hex. |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |

| Binair | Hex. |
|---|---|
| 1000 | 8 |
| 1001 | 9 |
| 1010 | A |
| 1011 | B |
| 1100 | C |
| 1101 | D |
| 1110 | E |
| 1111 | F |

Vb: omzetting van 1110001101

| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| 3 | | | 8 | | | D | | | |

Om aan te duiden dat een getal in het hexadecimaal talstelsel is (bvb tijdens programmeren), voegen we de prefix '0x' toe: 0x38D.

Naar tiendelig talstelsel:

$0x38D = 3*16^2 + 8*16^1 + 13*16^0 = 3*256 + 8*16 + 13 = 869$

*0x11 is dus gelijk aan 17*