Informatica

les 1

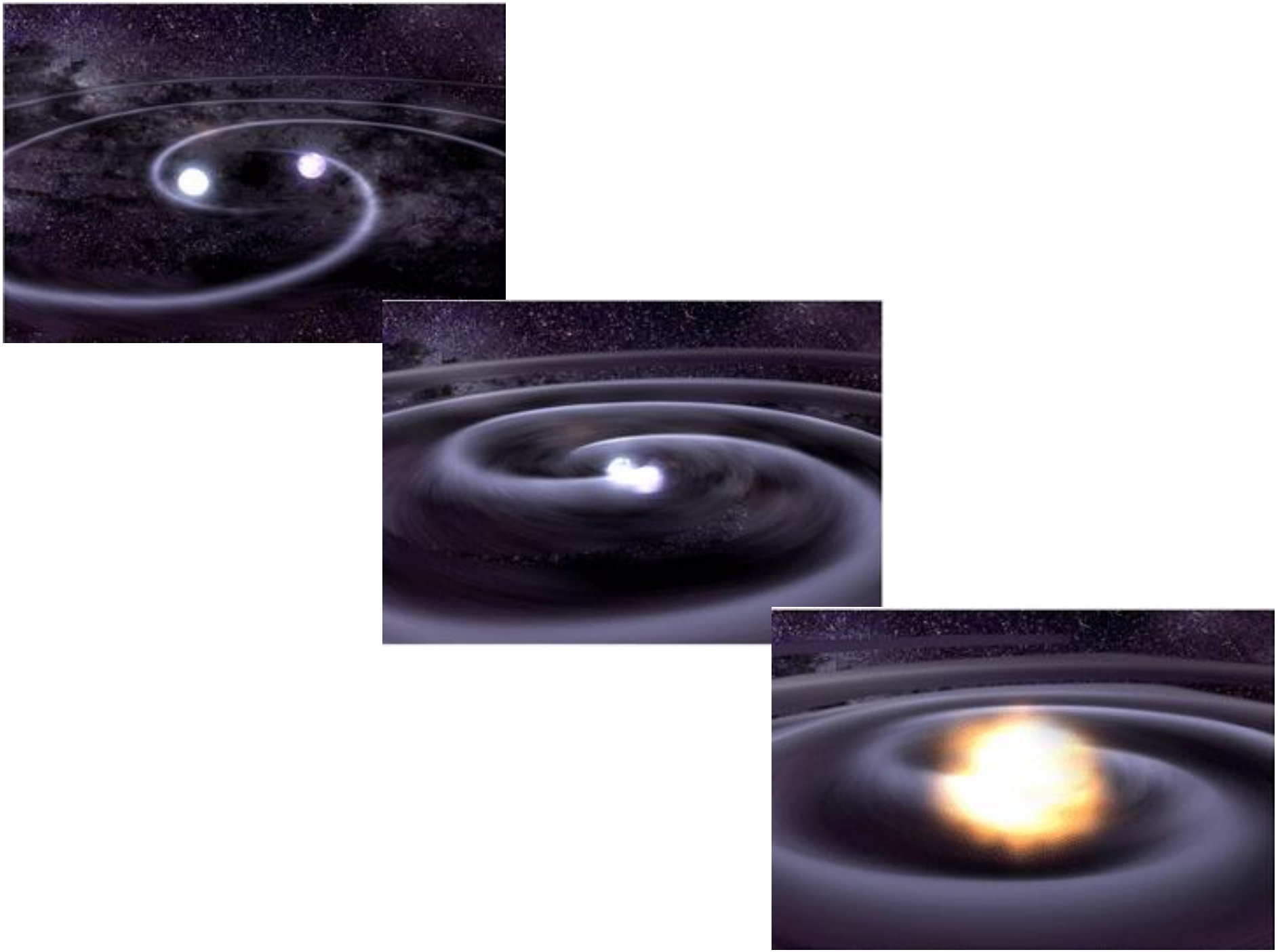
Java versus Python

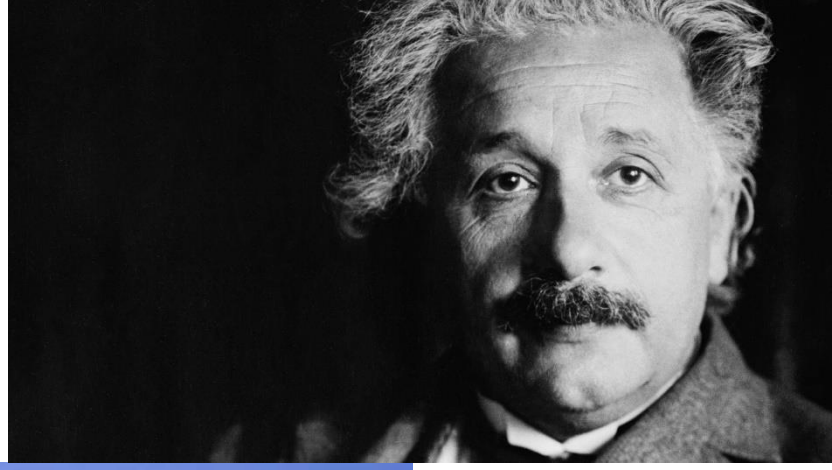
Jan Lemeire

Informatica IR 2e semester

februari – mei 2023

VRIJE UNIVERSITEIT BRUSSEL



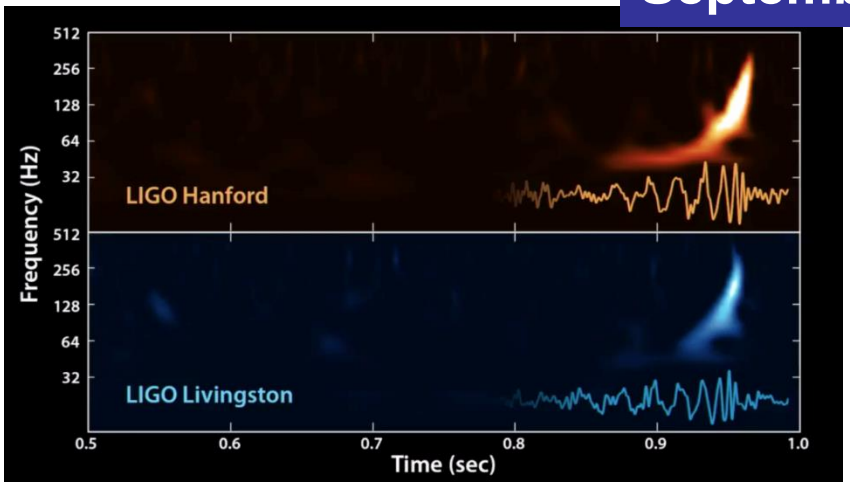


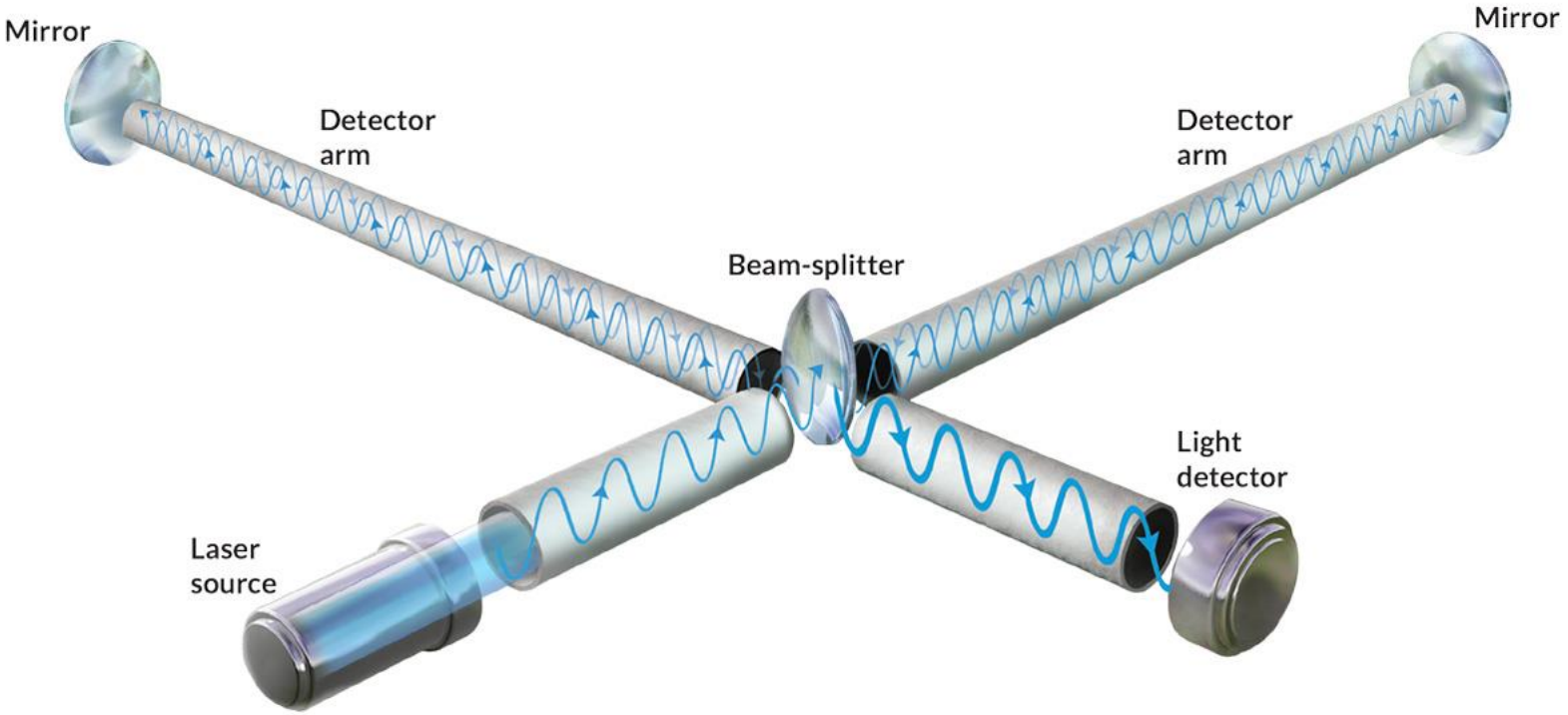
LIGO

LIGO

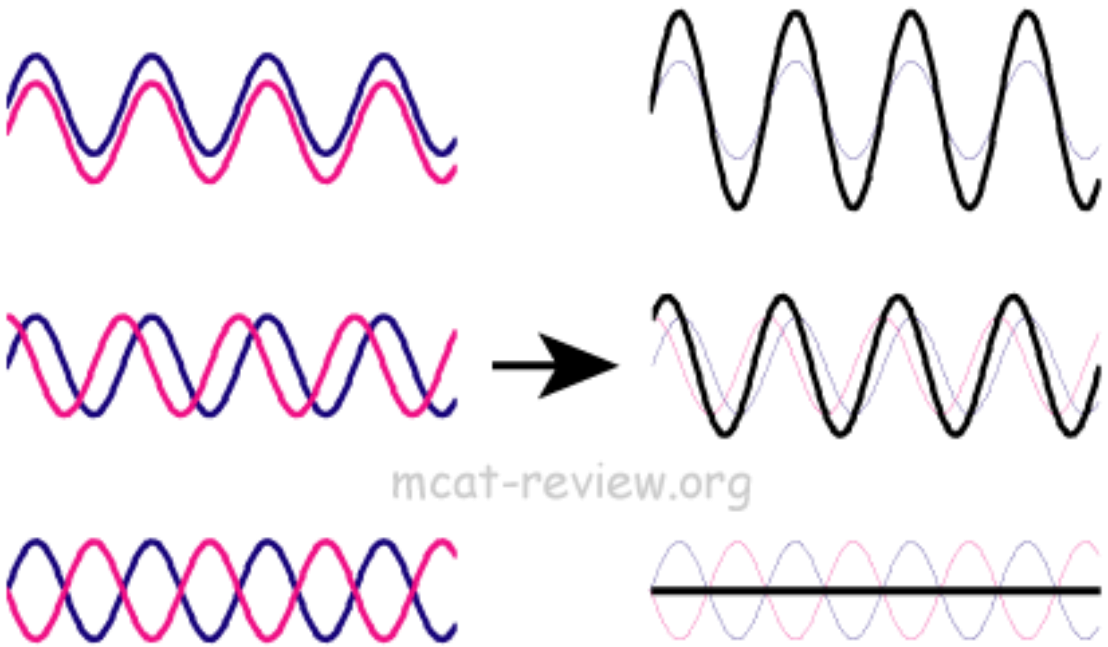
The First Observation of Gravitational Waves

September 14, 2015





Superpositie van signalen



In faze

In tegenfaze

UPDATE: August 14, 2017
Detection of a gravitional wave by 3 detectors, also by the European Virgo.
1.8 billion years ago, collision of black holes of 25 and 31 times the mass of the sun

14 augustus 2017 om 12.30.43 uur Belgische tijd.
Het verschijnsel is voor het eerst tegelijk waargenomen door drie detectoren, en twee soorten van detectoren: de twee LIGO's in de VS en de Europese Virgo.
Voor de Europese Virgo was de waarneming een primeur. Virgo, die een uitgebreide update heeft gekregen en nu de Advanced Virgo heet, was nog maar twee weken opnieuw bezig met het verzamelen van wetenschappelijke gegevens.
De botsing vond plaats op zowat 1,8 miljard lichtjaar afstand.
GW170814, zoals de nieuw ontdekte zwaartekrachtgolf is gedoopt (Gravitational Wave en de datum), is veroorzaakt in de laatste ogenblikken van de versmelting van twee zwarte gaten die 31 en 25 keer zo zwaar waren als de zon

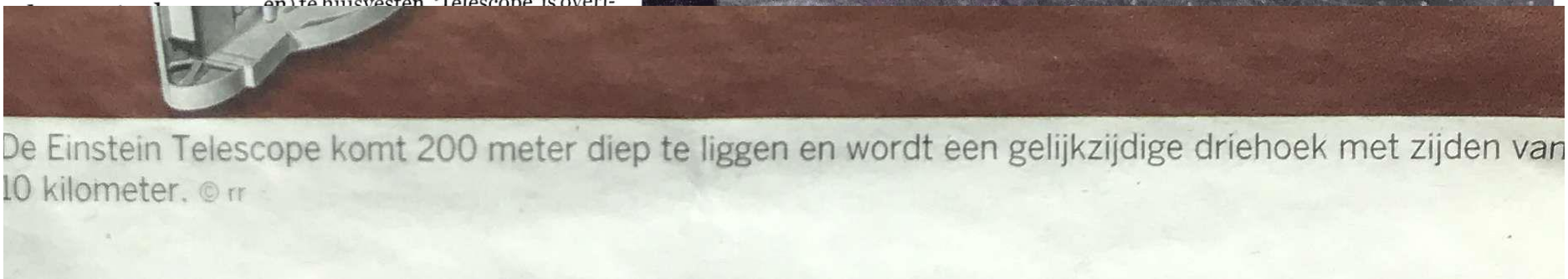
Albert Einstein had in 1915 geopperd dat het heelal bestaat uit ruimte en tijd, en dat die één geheel vormen.
Na een heftige gebeurtenis kan die "ruimtetijd" trillen. De schokgolven, zwaartekrachtgolven, gaan dan door het heelal als rimpelingen na een steen in een vijver. Bij zo'n rimpeling rekt de ruimte iets uit of krimpt zij iets.
Hoe groter de massa en hoe sneller de beweging van een massarijk object, des te sterker de zwaartekrachtgolf.

ok Vlaanderen zet zijn schouders onder de Einstein Telescope

Drielandenregio wil 'kosmisch afluistercentrum'

t zuiden van

gaten en gravitatiegolven voortvloei-
en) te huisvesten. 'Telescope' is overi-



De Einstein Telescope komt 200 meter diep te liggen en wordt een gelijkzijdige driehoek met zijden van 10 kilometer. © rr


EVGA
GeForce GTX 260
CORE 216


2010
350 Million triangles/second
3 Billion transistors GPU
1995
5.000 triangles/second
800.000 transistors GPU
2016
14.000 Million triangles/second
15 Billion transistors GPU
OGIO
HELMET


[Beta - MikeTheSempai]
70m


B
SYNCHRONIZE

DESALNIETTEMIN

VALERIE DROEVEN

We zitten met de poepers

'De buurtpolitie' speelt wel vaker een gebaggd spel met fictie en realiteit

Gamers verliezen strijd tegen bitcoin en artificiële intelligentie

Goldrush op de grafische chips

Gamers zijn boos. Door het succes van bitcoin en artificiële intelligentie raken zij niet meer aan grafische kaarten. De prijs ervan is met 30 procent gestegen en sommige winkels zouden maar één kaart per klant verkopen.

DOMINIQUE DECKMYN, ILLUSTRATIE JEROEN MURRÉ

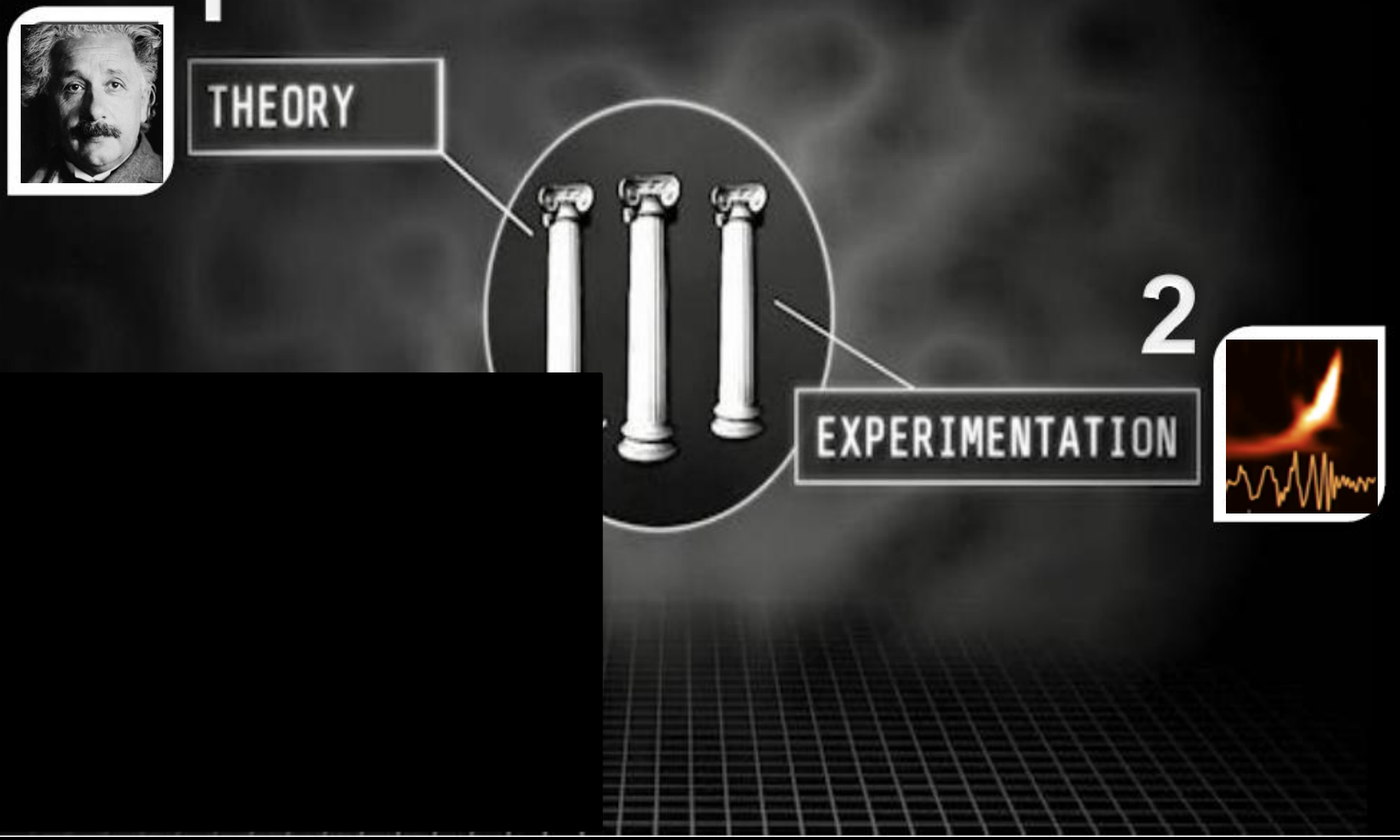
1.000 euro

Om de beste grafische kaarten wordt nu haast letterlijk gevochten

Cryptomunt

The third pillar of the scientific world:

computational science

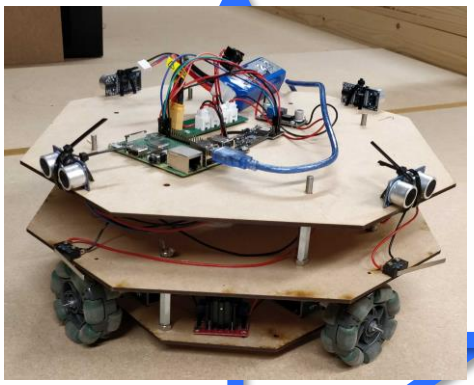

1
THEORY
2
EXPERIMENTATION

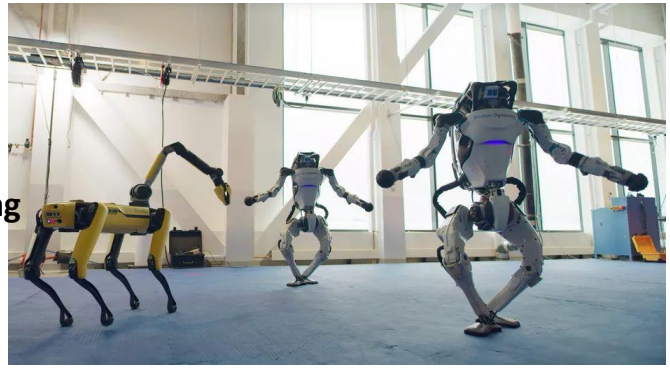
Informatica = tool

Jan Lemeire (jan.lemeire@vub.be)

- Burgerlijk ingenieur, 1994, VUB
  + bijkomende masters in de computerwetenschappen (1995)
- Werkte 4 jaar in de privé, voor 2 IT-consultancybedrijven
- 2000-2007: doctoreerde aan de VUB als assistent
  - Gaf oefeningen informatica
- Sinds 2008: professor aan VUB
  - Vak 'parallel systems' in de masters
  - Sinds 2011: titularis 'Informatica' eerste bachelors
- Sinds oktober 2013: geeft ook les aan *industrieel ingenieurs*
  - Bachelors: Informatica en Basiselektronica
  - Masters: Computerarchitectuur
- **Onderzoeksdomeinen**:
  - Software om robots intelligenter en autonomer te maken
  - Zelf-lerende robots zoals opgroeiende kinderen

ROBOTS SLIMMER MAKEN





Gen

Learning / adaptability

Self-learning

Unsupervised learning

One-shot learning

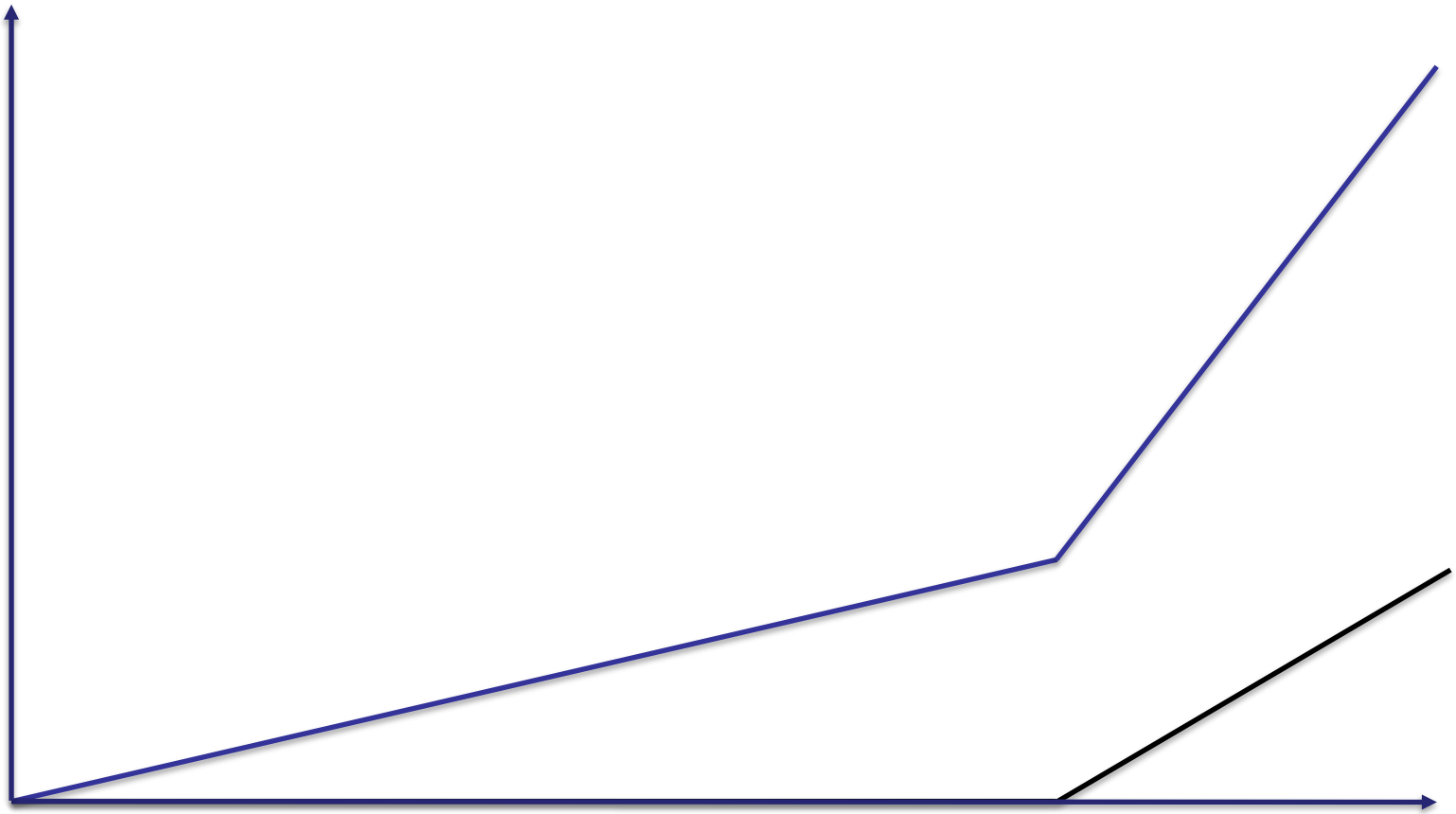
Supervised learning

Complexity

Eerste semester

Python: procedureel programmeren

Universiteit zonder voorkennis



1e semester

Basis-programmeervaardigheden

While – for
Lijsten/arrays
Functies
Recursie

CURSUS I

Java & Object-georiënteerd programmeren

Regels
Organisatie van de code
Gebruik klasses

Datastructuren

Arrays – ArrayLists
Stack, Queue, Linked lists
Trees - Maps

CURSUS II

Algoritmen

Basisalgoritmen
Slimme algoritmen
Sorteren

Beheers het gebruik van bibliotheken:

- Programmeervaardigheden
- Klassestructuur
- Voorbeelden
- Documentatie

PROJECT

GUI-klassen

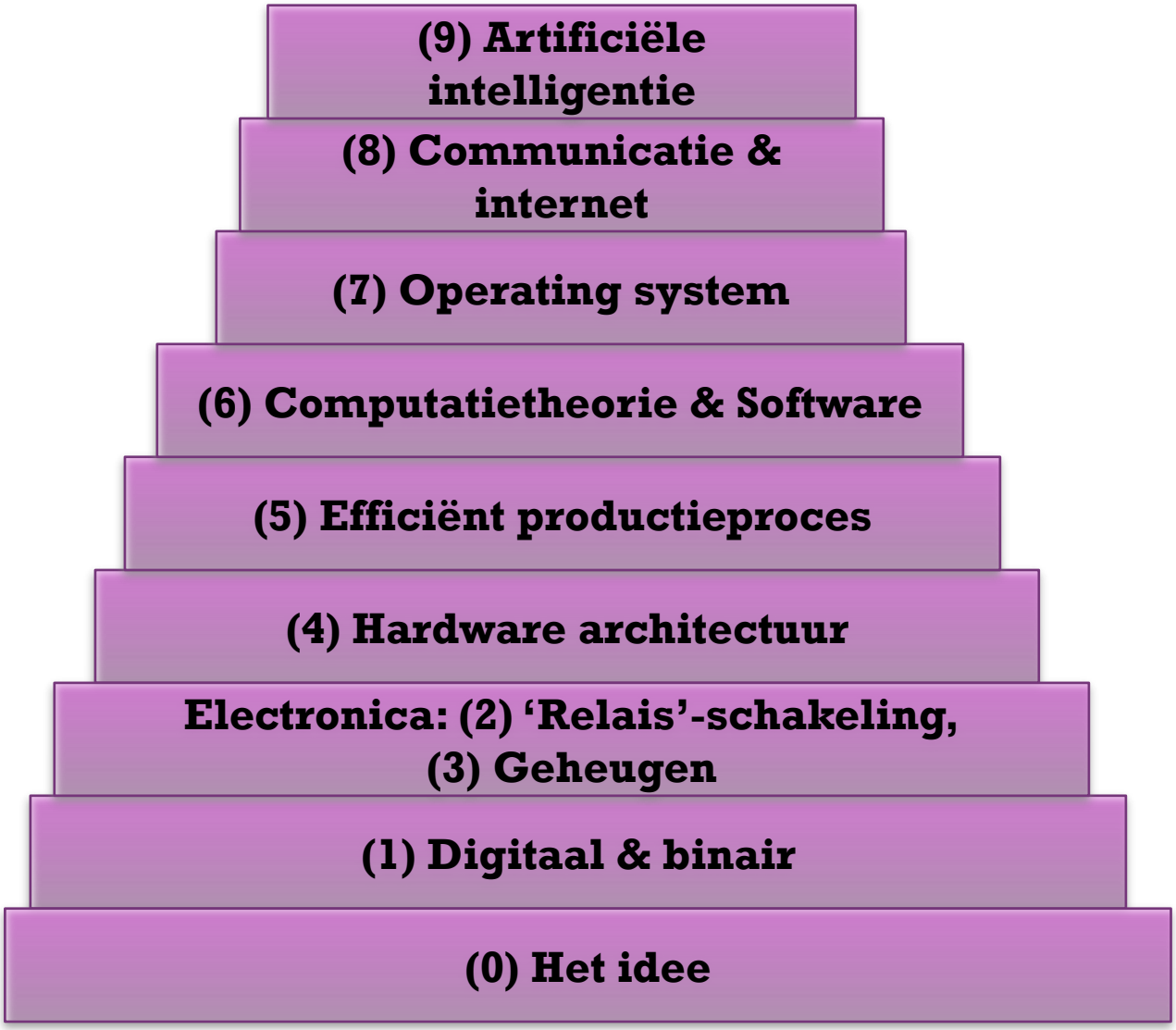
JFrames – Jpanels – Jbuttons - …
Events – Eventlisteners - mouse
Paint, layout, JApplet

Utility-klassen

Timer, Thread, Random, File, …

CURSUS III: technologie, historiek en economische aspecten

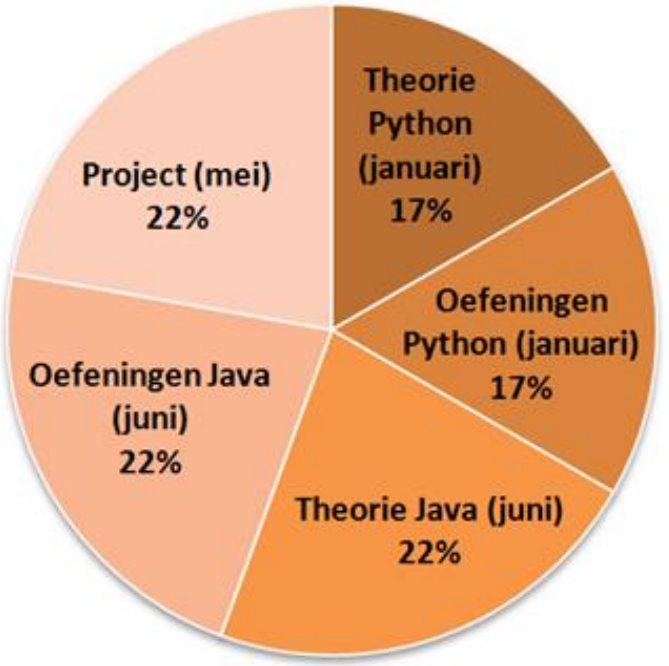
Informatica deel III: technologie, historiek en economische aspecten



2e semester

- Beschikbaar op canvas, website en vubtiek:
  - Cursustekst deel I
  - Cursustekst deel II & deel III
- Slides: de te kennen leerstof
  - alles staat in ook in cursusteksten, indien niet: staat duidelijk aangegeven in slides
- ***http://parallel.vub.ac.be*:**

  Code & alle info



2e zit en vrijstellingen

Zie ook parallel.vub.ac.be

- Indien **10/20** (afgerond gemiddelde) op je eindcijfer ben je geslaagd.
- Indien niet: **vanaf 12/20** heb je een (officieuze) **vrijstelling** voor één van de 5 onderdelen: Python theorie, Python oefeningen, Java project, Java theorie of Java oefeningen. De vrijstelling geldt van 1e naar 2e zit, maar ook van 1 jaar op een ander.
  - Er wordt niet afgerond.
- In juni bieden we de mogelijkheid om facultatief je punten van het 1e semester te verbeteren, indien je een 7, 8 of 9 behaalde in het eerste semester (gemiddelde over theorie en oefeningen) en je slaagt voor je mondeling examen met minstens een 12/20.

Deel I

- 4 pagina's referentie java
- Java's spelregels: *van buiten kennen*
- Pijlers van object-georiënteerde programmeertalen: via voorbeelden kunnen uitleggen

TIOBE Index for February 2022

1	3	▲	Python	15.33%	+4.47%
2	1	▼	C	14.08%	-2.26%
3	2	▼	Java	12.13%	+0.84%
4	4		C++	8.01%	+1.13%
5	5		C#	5.37%	+0.93%
6	6		Visual Basic	5.23%	+0.90%
7	7		JavaScript	1.83%	-0.45%
8	8		PHP	1.79%	+0.04%
9	10	▲	Assembly language	1.60%	-0.06%
10	9	▼	SQL	1.55%	-0.18%
11	13	▲	Go	1.23%	-0.05%
12	15	▲	Swift	1.18%	+0.04%
13	11	▼	R	1.11%	-0.45%
14	16	▲	MATLAB	1.03%	-0.03%

- http://www.tiobe.com/tiobe_index

Java versus Python

p. 8

1. Object-georiënteerde taal.
2. public static void main
3. Puntkomma's en accolades
4. System.out.println
5. Typeren: sterk en statisch
6. Arrays en ArrayLists
7. Verschil tussen letters en woorden
8. De for-lus
9. Varia

Python2java

Hetzelfde in Python

```
'Functions for working with temperatures.'

from math import sqrt

def to_celsius(t):
    'Convert Fahrenheit to Celsius.'
    return (t - 32.0) * 5.0 / 9.0

x = 100
x = x / 3
x = sqrt(x)

y = to_celsius(x)

print str(x)+" Fahrenheit = "+str(y)+" Celsius"
```

Java bestaat louter uit klasses

Statements eindigen met ';'

File zelfde naam als klasse

2 manieren om commentaar te geven

JavaVoorbeeld.java

```
// een klein voorbeeld met de belangrijkste java-elementen

import java.lang.Math;

public class JavaVoorbeeld {

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

        // declaratie en initialisatie van x
        int x = 100;

        // enkele berekeningen
        x = x / 3;

        x = (int) Math.sqrt(x);

        // omzetting naar Celsius
        double y = toCelsius(x);

        // printen
        System.out.println(x+" Fahrenheit ="+y+"Celsius");

    }

    /** Convert Fahrenheit to Celsius */
    public static double toCelsius(double gradenFahrenheit){
        return (gradenFahrenheit - 32) * 5 / 9;
    }

}
```

Het programma begint met de **main**. De eerste lijn moet er zo uit zien.

Declaratie van variabelen: type aangeven

Functie van Math-klasse, een java-klasse met handige mathematische functies.

sqrt geeft een double terug, met (int) zet je om ('casting') naar integer.

Oproep van zelfgemaakte functie.

Printen van gegevens

Een functie duidt je aan met static

Parameter van methode

Type van teruggeefwaarde

Commentaar die gebruikt wordt voor documentatie

Niet in cursus

Rekenen met integers!

Python code	Waarde x	Java code	Waarde x
x = 100	100	int x = 100;	100
x = x / 3	33.33333 (a)	x = x / 3;	33 (b)
x = sqrt(x)	5.773502	x = (int) Math.sqrt(x);	5 (c)

(a) Alhoewel x eerst als integer werd opgeslagen, wordt het type verandert naar float om zo de cijfers achter de komma weer te geven. (Enkel in Python 3, niet in Python 2)

(b) In de meeste programmeertalen blijft x hetzelfde type behouden. Bij integerbewerkingen wordt alles na de komma weg gegooid.

(c) De sqrt-functie neem een float/double als input. X wordt automatisch omgezet naar double (typeverruiming). De teruggeefwaarde van sqrt is een double, maar x is een integer. Omdat het hier over typeverarming gaat moet je expliciet aangeven dat je het type verandert en akkoord bent met verliezen van alle cijfers achter de komma.

Niet in cursus

Rekenen met integers!

Java code	Waarde x	
int x = 100; x = x * 2 / 3;	(a) x = ?	**x = 66**
int x = 100; x = x * (2 / 3);	(b) x = ?	**x = 0**
int x = 100; x = 2 / 3 * x;	(c) x = ?	**x = 0**
double x = 2 / 3;	(d) x = ?	**x = 0.0**
double x = 2.0 / 3;	(e) x = ?	**x = 0.66667**

- Integers: alles naar de komma verdwijnt!
- (d) de omzetting naar double gebeurt pas na de berekening!
  (e) is een oplossing.

Wat berekent *GHI*?”

```java
import java.util.Scanner;

public class GHI {

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

        System.out.println("Geef 2 getallen: ");

        Scanner scanner = new Scanner(System.in); // om input in te lezen
        int a = scanner.nextInt();
        int b = scanner.nextInt();

        int ghi = berekenGHI(a, b);
        System.out.println("Het resultaat na het ingeven van "+a+" en "+b+" is "+ghi);
    }

    public static int berekenGHI(int x, int y){
        while (x != y){
            if (x > y)
                x = x-y;
            else
                y = y-x;
        }
        return x;
    }
}
```

Hetzelfde in python

```
def berekenGHI(x, y):
    while x != y:
        if x > y:
            x = x-y
        else:
            y = y-x
    return x

s = input("Geef 2 getallen: ")
a = int(s)
s = input()
b = int(s)
ggd = berekenGHI(a, b)
print("GGD van", a, "en", b, "is", ggd)
```

Getypeerde parameters

```
public static int berekenGrootstGemeneDeler(int x, int y)
```

- Java: Enkel op te roepen met integers
- In python: ook reëel getal mogelijk! Elk type, dus ook string…
  - Parameters *4.5* & *5.5* werkt
  - Parameters *4.5* en *5.1* werkt het niet: gaat oneindig door, x en y worden nooit gelijk

    => NOK!

Type checking nuttig

- Je weet als gebruiker wat mee te geven
  - In python: input documenteren
- Geruststelling, je kan 'niets fouts' doen
  - Hier: negatieve waarden geeft fouten: checken
  - In Python: steeds extra checks doen op type
- Parameters kunnen ingewikkelde objecten zijn

Pythonvoorbeeld audio

```
sample_rate, signal = scipy.io.read("mymusic.wav")
freq = scipy.fft(signal))
```

Inlezen van audio-file en omzetten naar frequentiespectrum.

Gemakkelijk, maar hoe zien de gegevens er uit? Wat zijn de types? ***Niet duidelijk.***

Functie: maar 1 teruggeefwaarde

- Java, zoals de meeste programmeertalen
- Functie
  - kan maar 1 waarde teruggeven
  - Voorbeeld: celsius-conversie, ggd
- Indien meerdere waarden: ‘truken’ nodig

Voorlopige conclusie

Python

- Snel, voor kleine programma's
- High-level, handige basisfunctionaliteiten
  - **Cf Matlab**

Java

- Voor grote programma's en bibliotheken, te delen met andere programmeurs
- Iets meer low-level
- Meest-gebruikte industriële taal

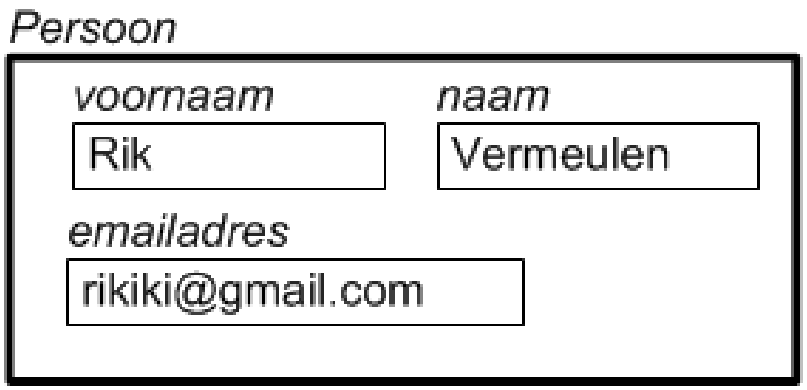
Java versus Python

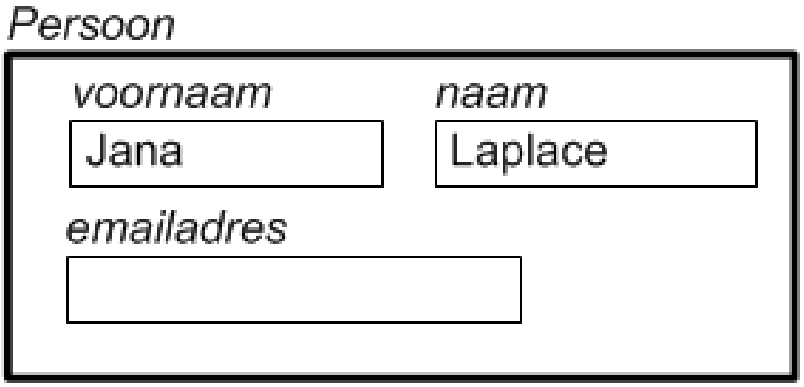
1. Object-georiënteerde taal.
2. public static void main
3. Puntkomma's en accolades
4. System.out.println
5. Typeren: sterk en statisch
6. Arrays en ArrayLists
7. Verschil tussen letters en woorden
8. De for-lus
9. Varia

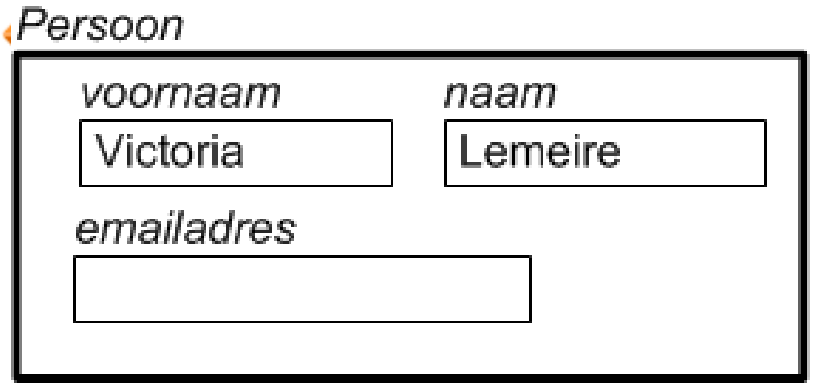
Doelstellingen object-georiënteerde talen

- Een project zo structureel mogelijk opbouwen met objecten
- Code opdelen in logische componenten, modulariteit
- Code makkelijker te **hergebruiken** en **uit te breiden**

Objecten


Persoon
voornaam
Rik
naam
Vermeulen
emailadres
rikiki@gmail.com


Persoon
voornaam
Jana
naam
Laplace
emailadres


Persoon
voornaam
Victoria
naam
Lemeire
emailadres

```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+"]");
    }
}
```

Ternary operator: … ? … : …

p. 16

Persoonobjecten

```
public class PersoonMain {

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

        Persoon rik = new Persoon("Rik", "Vermeulen", "rikiki@gmail.com");
        Persoon jana = new Persoon("Jana", "Laplace");
        Persoon victoria = new Persoon("Victoria", "Lemeire");

        victoria.voornaam = "Vicky";
        jana.maakDefaultEmailadres("hotmail.com");
        System.out.println("Adres aangemaakt: "+jana.emailadres);

        System.out.println("Rik: "+rik);
        System.out.println("Jana: "+jana);
        System.out.println("Victoria: "+victoria);
    }
}
```

Objecten

- Gedefinieerd met een **klasse** (*type object*)
- Heeft **attributen**
  - De eigenschappen van het object
- Aangemaakt met een **constructor**
  - Object is *instantiatie* van een welbepaalde klasse
- Heeft **methodes**
  - Methode heeft toegang tot de attributen, kan deze veranderen

Overerving

Een student is een persoon

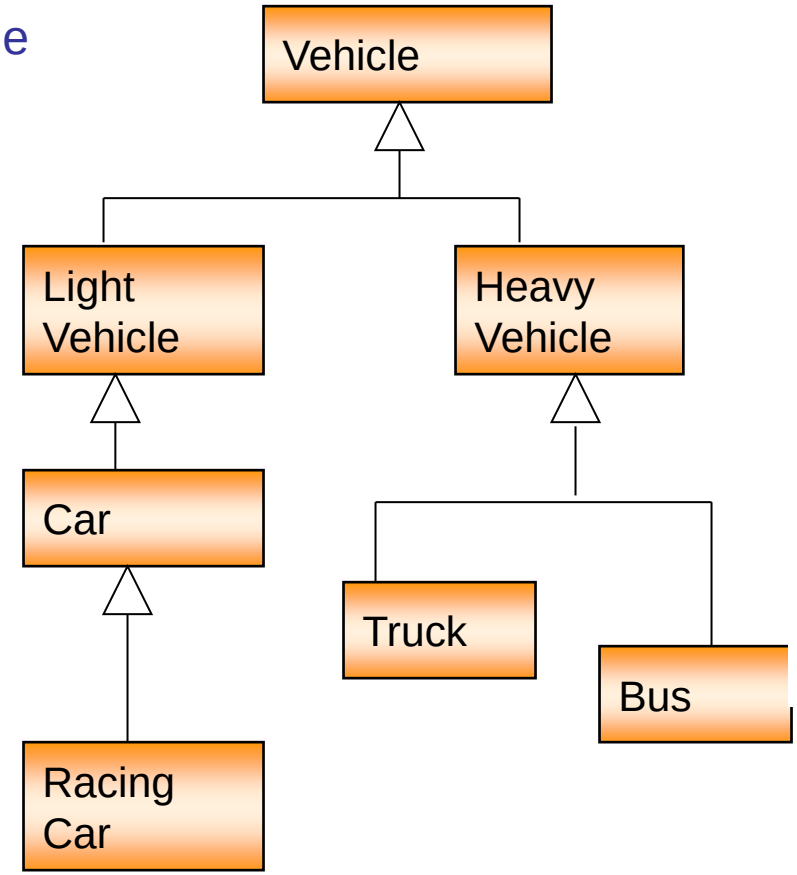
```
public class Student extends Persoon{
        enum Faculteit {IR, WE, GF, LK, LW, ES, RC, PE};

        int rolnummer;
        Faculteit faculteit = Faculteit.IR; // default waarde voor
alle nieuwe objecten

        Student(String voornaam, String naam, int
rolnummer){
                super(voornaam, naam);
                this.rolnummer=rolnummer;
        }
}
```

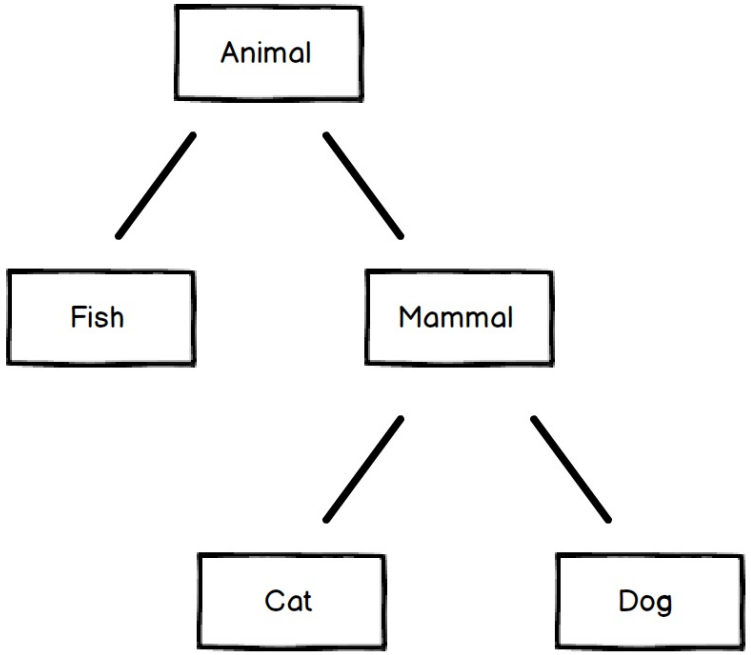
Overerving (*inheritance*)

- Subklasse erft alle attributen en methodes over van moederklasse
- Subklasse kan attributen toevoegen
- Subklasse kan methodes toevoegen of methodes *overschrijven*
- Constructor van subklasse moet een constructor van moederklasse oproepen (superconstructor)

Overerving leidt tot een klassehiërarchie

Generalizatie





Specializatie

Pijlers van object-georiënteerde programmeertalen

I. Encapsulatie

- **2.3 ArrayList p. 12**
- **3.1 Stapel-datastructuur p. 13**
- **6.2 Java's LinkedList p. 56**

II. Overerving (inheritance)

- 1.1.3 Studentvoorbeeld p. 20
- 1.3 Vriendenvoorbeeld p. 36
- 1.4.1 MyPanel p. 41
- 1.4.3 PainComponent overschrijven p. 44
- **4.3 FunctieMetAfgeleide-interface p. 25**

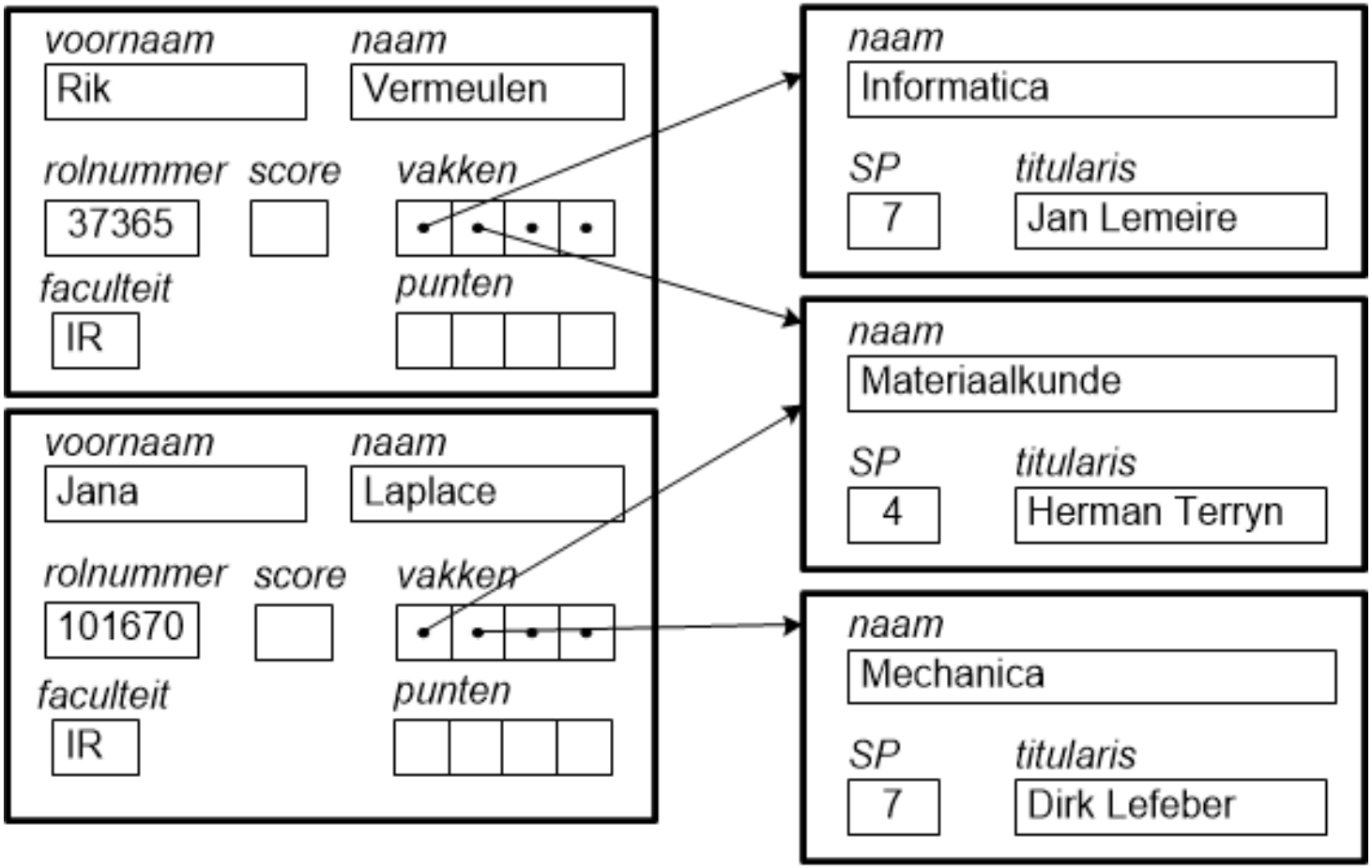
III. Polymorfisme en abstractie

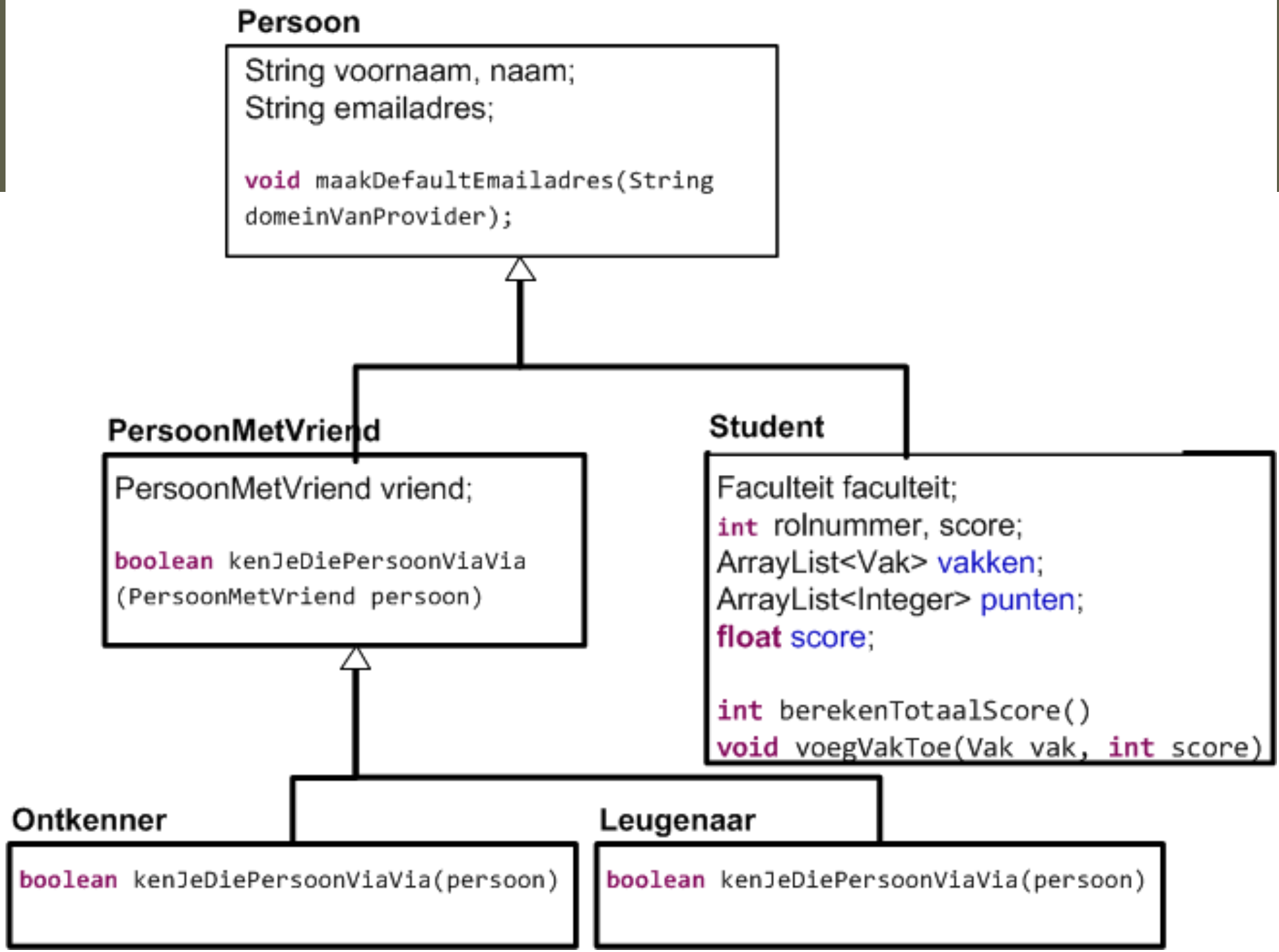
- 1.2.4 Set p. 31
- 1.2.5 Map p. 33
- 1.4.2 EventListener p. 43
- 1.6.4 Abstracte klassen p. 58
- **4.2 Functie-interface p. 22**
- **5.2.2 Backtracking & Breadth-first p. 32**
- **Addendum bij hoofdstuk 5 (zie website, is optioneel)**
  - **Abstract zoekalgoritme**
  - **Vergelijking van zoekalgoritmes**

Super-constructor

- Het aanmaken van een Student-object vergt ook het aanmaken van een Persoon-object
- Op de eerste lijn van de Student-constructor roep je een constructor van Persoon op met
  `super(voornaam, naam);`

Objectattributen


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

        int rolnummer;
        Faculteit faculteit = Faculteit.IR; // default waarde voor alle nieuwe objecten
        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;
        }
}
```

```
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);
}
```