Informatica

2e semester: les 9

Software, Sorteren en OS

Jan Lemeire

Informatica 2e semester

februari – mei 2023

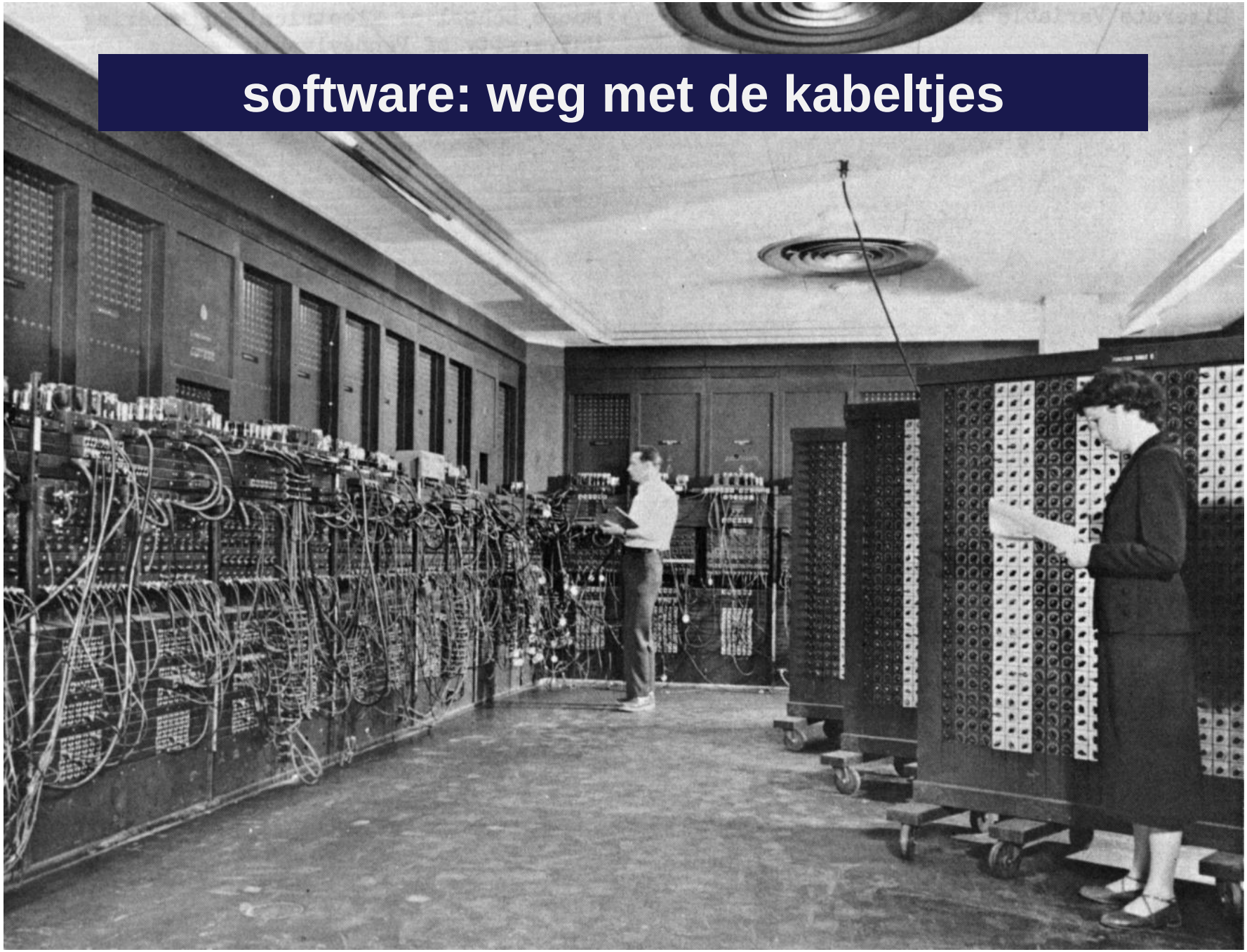
VRIJE UNIVERSITEIT BRUSSEL

Vandaag

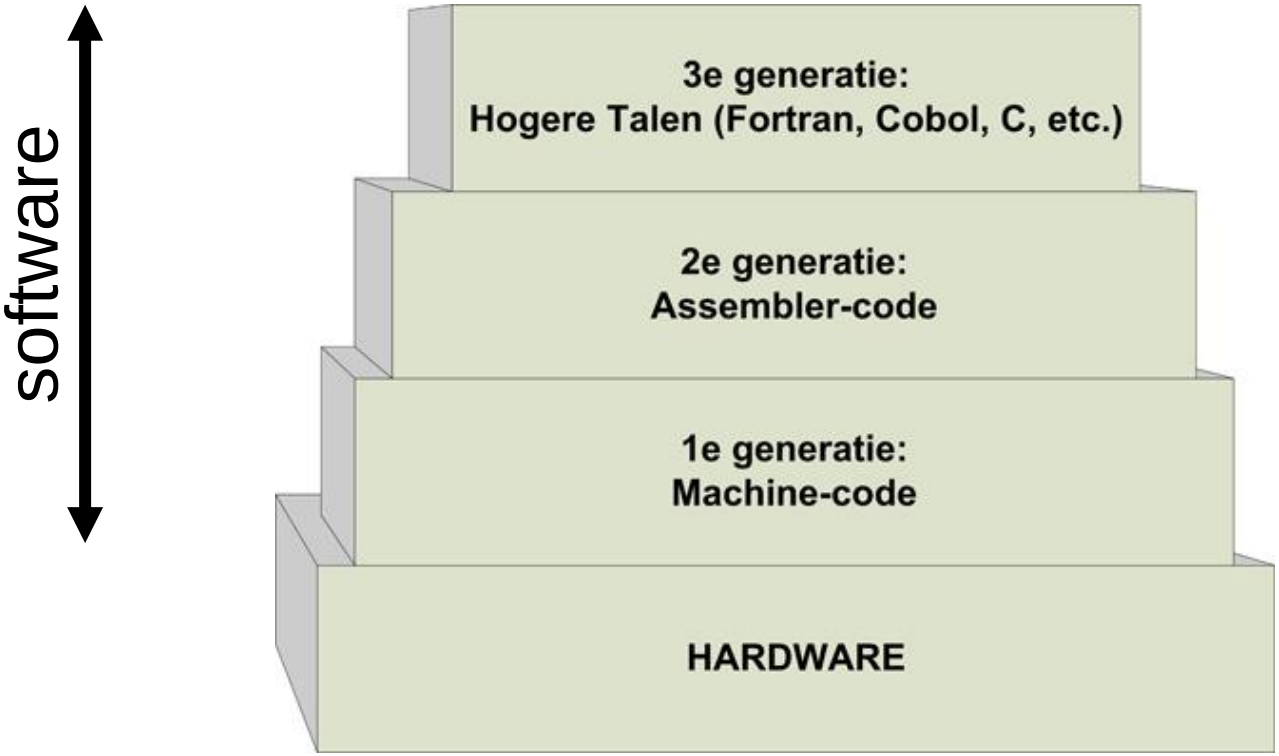
1. Software (deel III)
2. Sorteren
3. Het gebruik van static
4. Highscore server
5. Besturingssystemen (deel III)

Hoofdstuk 6b: Software

software: weg met de kabeltjes

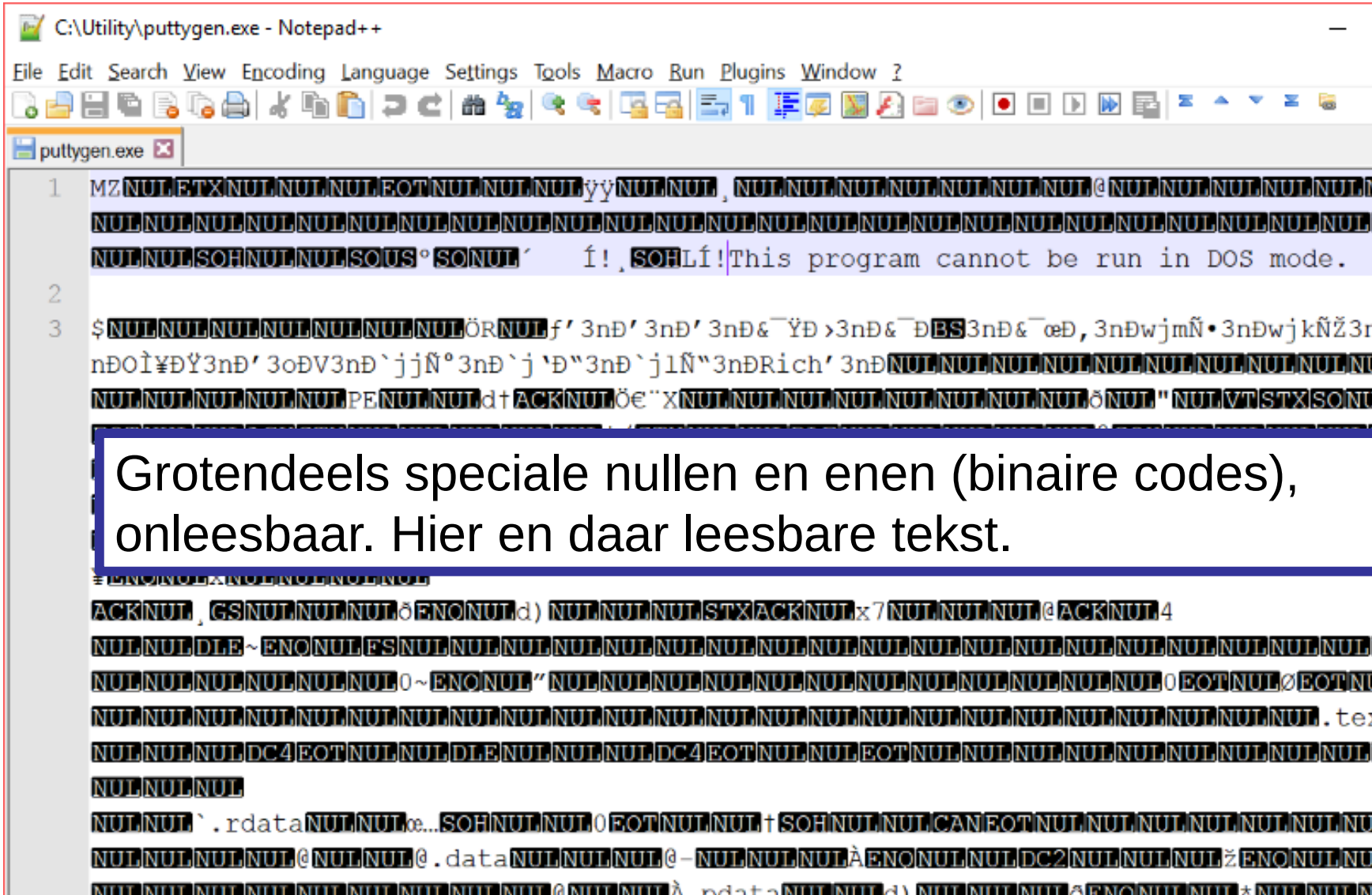


Programmeertalen

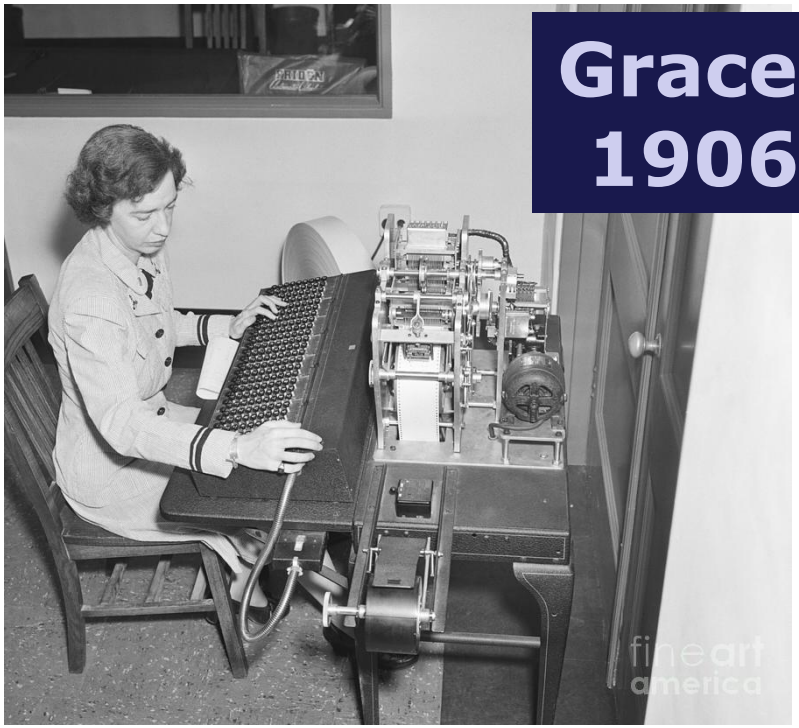


Een processor biedt een aantal instructies aan die hij begrijpt en kan uitvoeren. Deze instructies zijn binair en noemt men *machinecode*. Voor de leesbaarheid biedt *assembler* mnemonische afkortingen aan voor de binaire instructies, alsook variabelen en labels van codelijnen. De assembler zet alle afkortingen om naar binaire machinecode die dan uitgevoerd kunnen worden.

Executable = Machinetaal



Machine-onafhankelijk programmeren



Grace Hopper
1906 - 1992

Ontwikkeling programmeertalen die op elke machine uitgevoerd kunnen worden. De programma's worden daarna omgezet in specifieke machinetaal. Grace Hopper was hiervoor de grondlegger met de taal COBOL, die nog steeds voorkomt!

Voorbeeldprogramma (zie hfst 4)

```
Scanner scanner = new Scanner(System.in);
System.out.print("Geef een getal:");
int x = scanner.nextInt();
int m = x;
int t = 0;
do {
     System.out.println(m);
     m = m * x;
     t++;
} while (m < 1000 && t < 20);
```

In assembler (javacode vanaf lijn 3)

```
     lw  $t0, $s0       # load word
     mv  $t1, $t0        # move value
     mv  $t2, 0
l1:  sw  $t0, $s1       # store word
     mul $t1, $t1, $t0 # multiply
     add $t2, $t2, 1   # addition
     lt  $t3, $t1, 1000 # less than
     lt  $t4, $t2, 20   # less than
     and $t5, $t3, $t4 # and
     beq $t5, 1, l1    # jump to l1 if equal
```

Lezen van toetsenbord en schrijven naar scherm gebeurt door te lezen en schrijven naar geheugenadressen $s0 en $s1. Variabelen x, m en t staan in registers $t0, $t1 en $t2. Voor de conditie hebben we 3 extra registers nodig ($t3, $t4 en $t5). ‘l1’ is een label van een instructie, om er later naar toe te kunnen springen. Resultaat van een bewerking komt in eerste operand.

Machinetaal bestaat uit 3 soorten instructies

Hoofdstuk 4

1. Berekeningen en logische bewerkingen
   - Berekeningen: optellen, vermenigvuldigen, bitwise operaties
   - Logisch: vergelijken, and/or/xor/not
2. Controle-instructies
   - De program counter wordt verhoogd/verlaagd: het programma gaat dus niet verder met de volgende instructie, maar springt naar een andere positie
3. Lezen en schrijven van gegevens
   - Van en naar geheugen
   - Van input en naar output

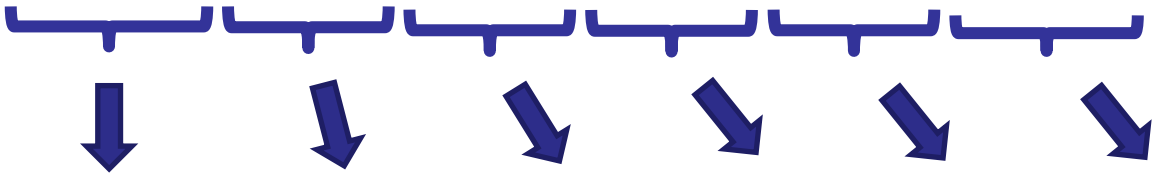
Machinetaal (binair)

assembler

```
add $t0, $s1, $s2
```

machinetaal

00000010001100100100000000100000



000000	10001	10010	01000	00000	100000
special	$s1	$s2	$t0	0	add
0	17	18	8	0	32

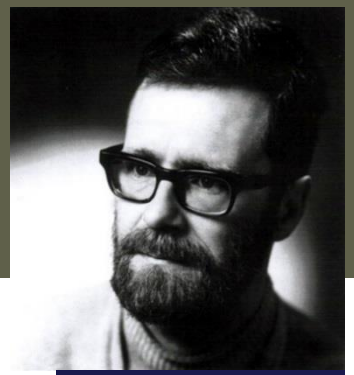
1 instructies = 32 bits met specifieke betekenis

3e generatietaal

- Onafhankelijk van de hardware
- ≈ beschrijving van het algoritme
  - Gebruik van variabelen ipv geheugenadressen
  - Geen JUMP, enkel for/while/if/switch
    - Jumps leidt tot spagetticode, je mag van overal naar overal springen
  - Functies/methodes mogelijk
- Omzetting naar assembler: *compileren*
  - Checkt je code op syntaxfouten!

Programmeren in assembler is niet produktief. Het is redelijk omslachtig, je maakt snel fouten en de code is heel onleesbaar. Daarom heeft men programmeertalen ontwikkeld die minder code vergen, meer aansluiten bij algoritmes, meer gestructureerd zijn, leesbaarder en minder foutgevoelig. De Nederlander Edgser Dijkstra was het voornaamste genie achter de 3e generatietalen.

Edsger Dijkstra (NL)

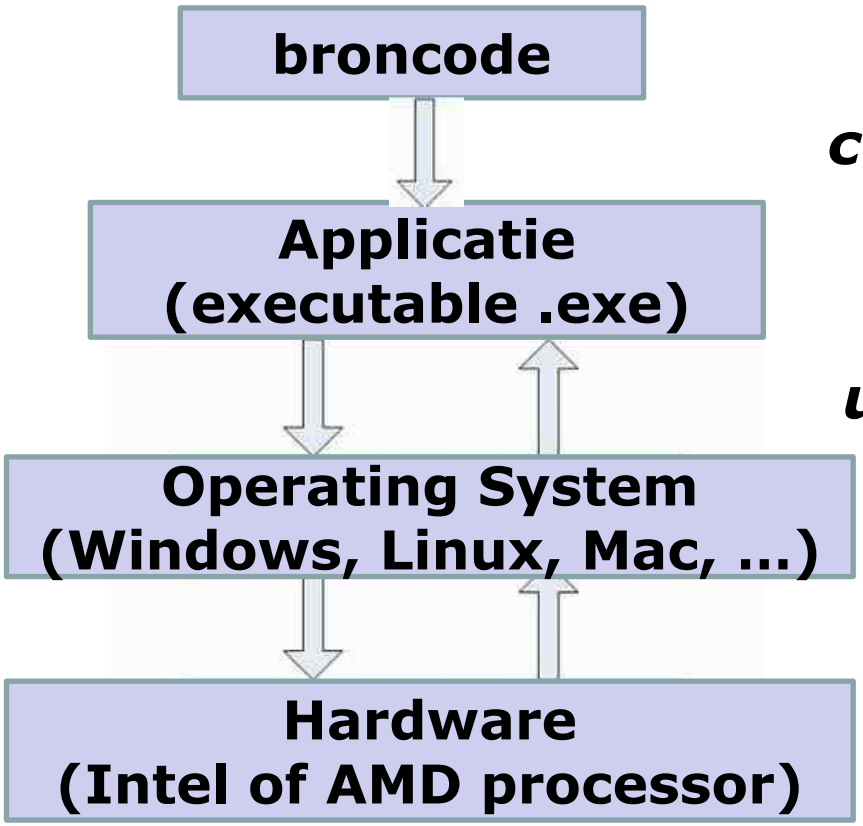


1930 – 2002

Welke constructies moet een programmeertaal minimaal bevatten?

- **While/for**
  - **Goto** is onnodig. In oude talen kon je zeggen ‘spring naar lijn 111’, naar elke willekeurige lijn van je programma. Dit leidde echter tot onoverzichtelijke ‘spaghetti’-code. Dijkstra toonde aan dat een while voldoende is
- **Functies**
  - In het begin waren er om praktische redenen veel beperkingen op het gebruik van functies.
- **Recursie**
  - Hier was veel tegenkanting tegen, omdat het praktisch tamelijk ‘duur’ is (vergt veel van de processor)

Programma => computer



Hogere taal

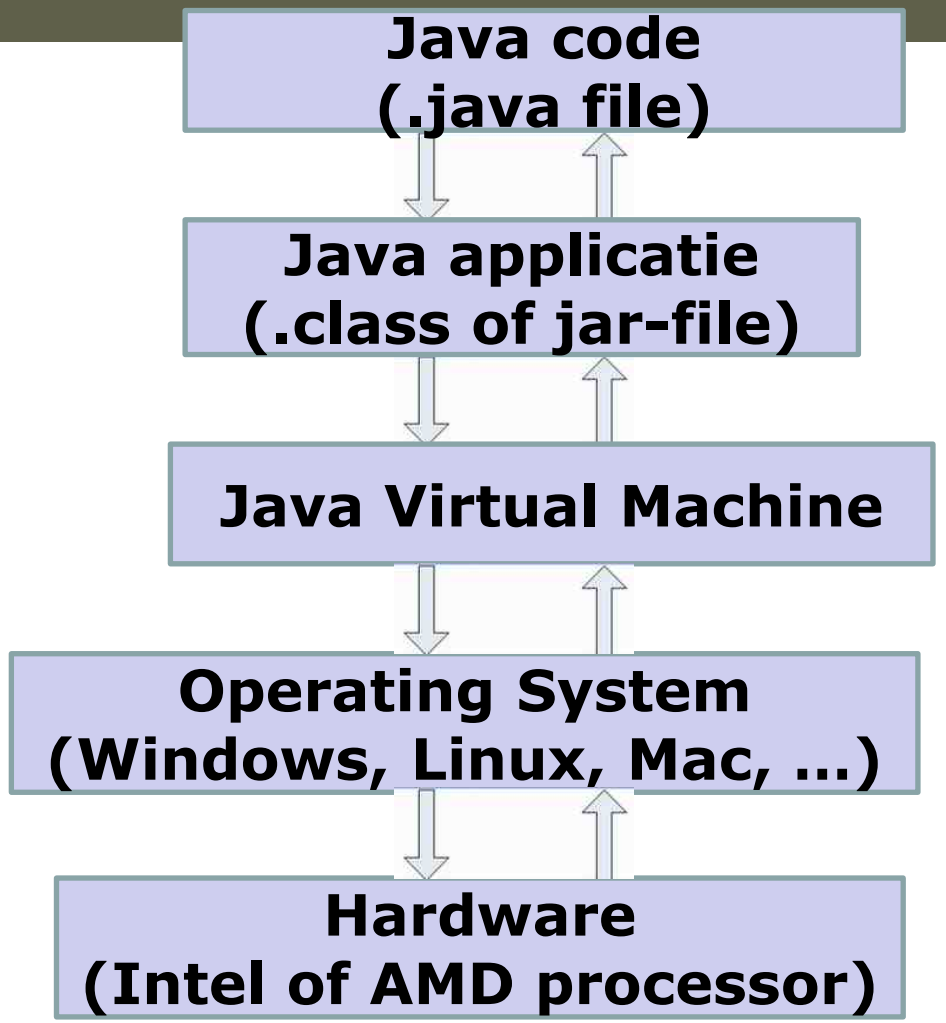
compileren

Machinetaal

uitvoeren

Nieuwe compilatie nodig voor elk systeem (operating system + hardware)!

Java: via virtuele machine



compileren

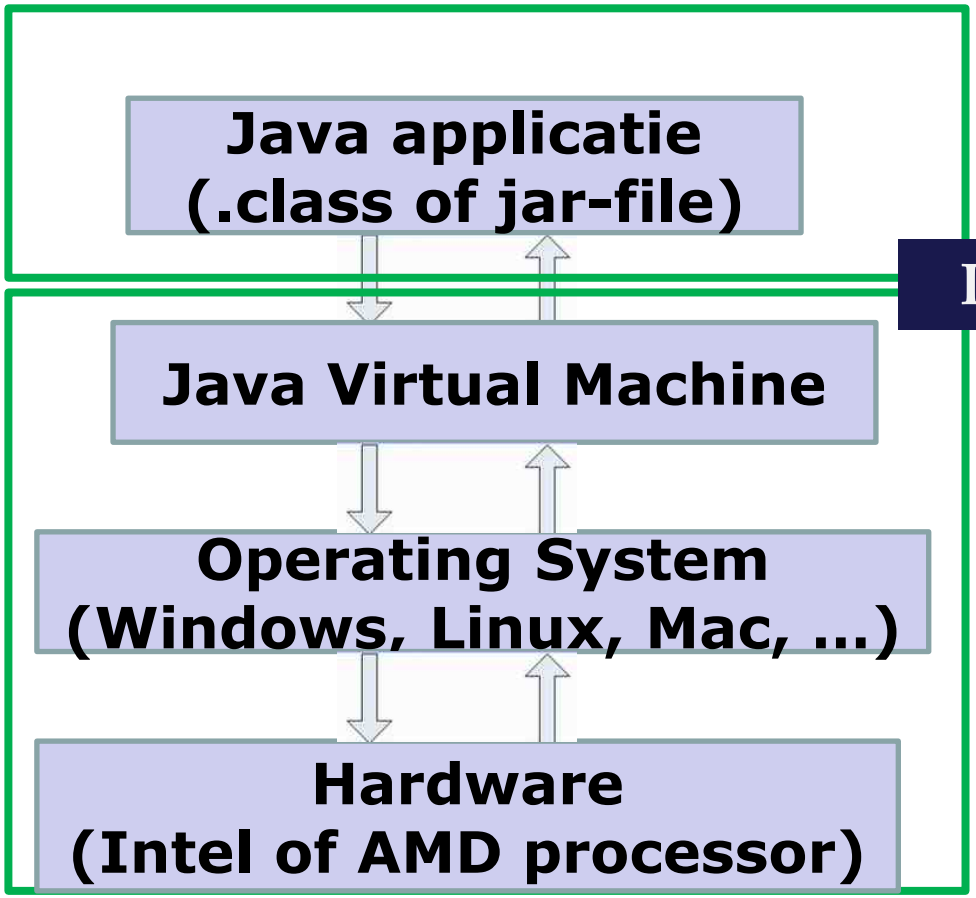
Specifieke tussentaal die iets hoger is dan assembler

uitvoeren

Tijdens het uitvoeren wordt instructies van de class-files omgezet naar machinetaal

Javacode werkt op elk operating system!
Je moet wel de virtuele machine installeren

Ideaal voor internet



Jou code staat op **webserver**

Via virtuele machine runt hij op elke **lokale machine ('client')**

Doordat Javacode via een virtuele machine (die wel specifiek is voor elk systeem) tijdens het uitvoeren wordt omgezet naar machinetaal, kan je programma overal uitgevoerd worden.

Python & Matlab

- ***Geïnterpreteerde talen***
- Je code wordt tijdens het uitvoeren omgezet naar machinetaal.

Je moet Python/Matlab dus geïnstalleerd hebben

- Javacode wordt wèl gecompileerd, *java-files* worden omgezet naar *class-files*.
  - Javacode wordt dus gecontroleerd op fouten. Maar Python en Matlab niet, waardoor je programma zal crashen bij fouten met de types van variabelen bijvoorbeeld
  - Java Virtual Machine gebruikt een **Just-in-time compiler** om toch optimale machinecode te verkrijgen
    - Pypy is een just-in-time compiler voor Python

Snelheid matrixmultiplicatie

Product van twee 200x200 matrices

	runtime (ms)	faster?
C (executable)	8,4	
java	28	0,30
python	3100	0,0027
python met numpy library functie	11,5	0,73
Python met just-in-time compiler pypy	82	0,10
matlab	172	0,049
matlab library functie	0,713	11,8
gpu (intel)	0,375	22,4
gpu (AMD tahiti)	0,161	52,2
gpu (nvidia geforce GTX 650)	0,077	109,1

https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

4e generatietaal

- High-level specification languages: programmeren *wat* je wilt, niet *hoe* het moet gebeuren
  - Vb: GUI aanmaken met een tool
    - Te integreren in Eclipse
      - **Google Windowbuilder Pro**
      - **Visual Editor**

(Nog) niet echt succesvol

Enkel voor specifieke applicaties (zoals GUIs)

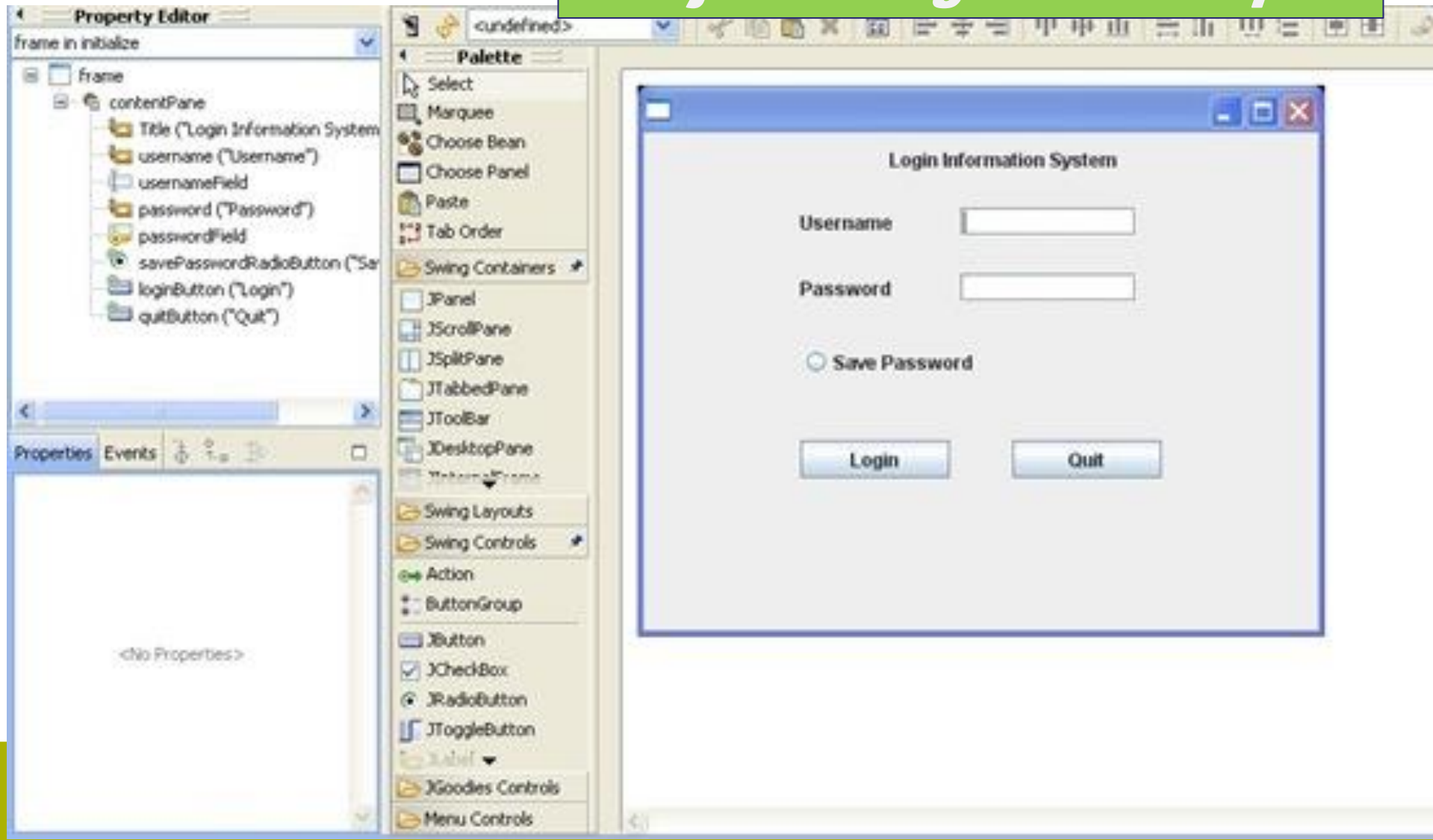
- Mijn mening: nog geen artificiële intelligentie

=> computer kan nog niet begrijpen wat je wilt

Een GUI aanmaken kan via een tool waarbij je niets moet programmeren. Je bouwt als het ware je applicatie door aan te geven ***hoe*** deze er uit moet zien. Deze methode werkt echter nog niet voor algemene applicaties…

Tool om GUIs te maken

Kan je toevoegen aan Eclipse!

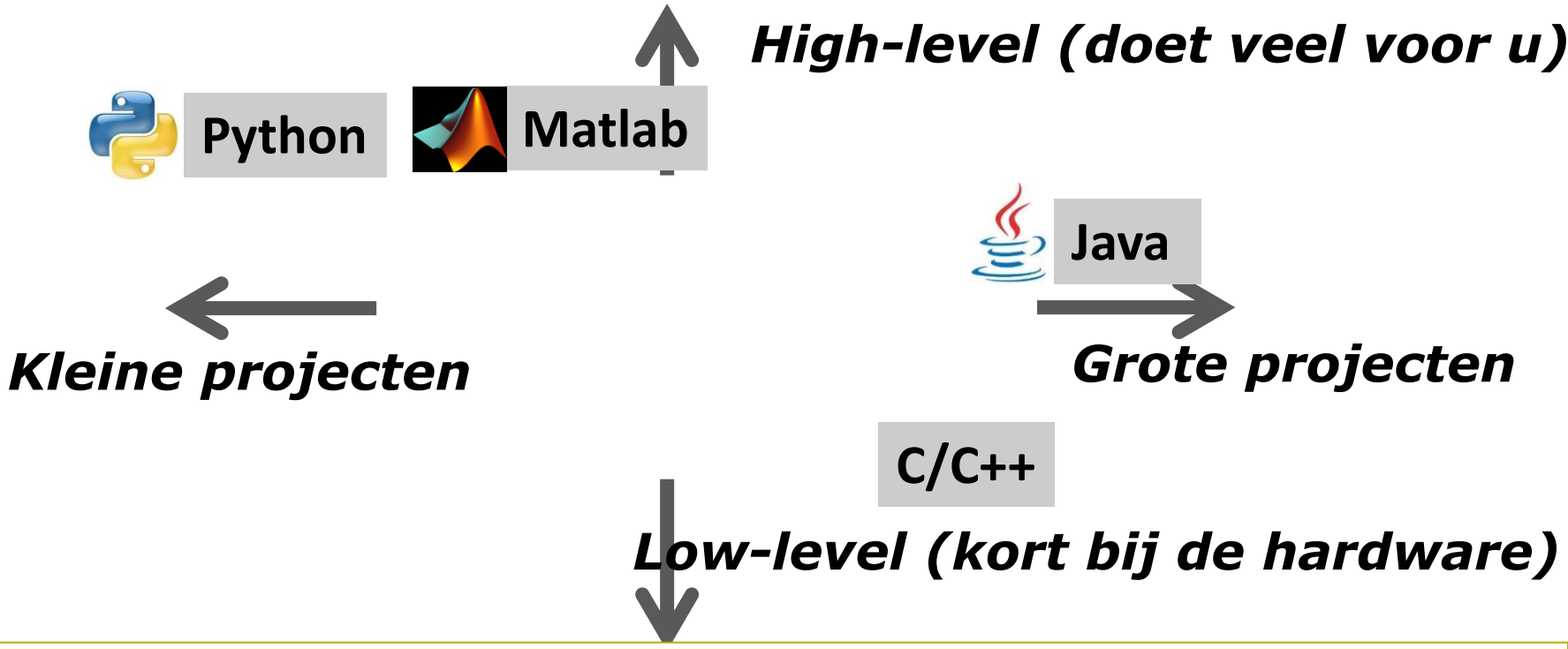


Object-georiënteerde talen

- Worden als 3e generatietalen aanzien
- Toch is de code-organisatie fundamenteel verschillend
  - De code is opgesplitst in klasses ipv functies.
  - Vergt een andere manier van denken
  - Is dè uitdaging van dit semester, zowel voor jullie als voor ons (lesgevers)

Voor ingenieur: welke taal?

moet handig en efficiënt zijn, geschikt voor de taak



Zonder al te diep in te gaan over wat nu de beste taal is (een eeuwige steeds-hoog-oplopende discussie onder informatici), in dit schema een kleine poging tot conclusie. C/C++ zullen velen later nog zien, het is een veel-gebruikte taal die kort bij de hardware staat, daarvoor uitermate geschikt voor het programmeren van systemen (chips, robots, kritische applicaties, …)

Wat met je Javaproject?

- **Executable jar-file**
  - ⇒ ***Eclipse: Export => Java => Runnable JAR file***
  - ⇒ ***Je kiest klasse met de main() die uitgevoerd moet worden***
- **App for Android**
  - **Android = java**
  - **Mogelijk, maar nog iets te moeilijk voor 1e bachelors (vooral heel wat techniciteiten, XML-files enzo)**

Hoofdstuk 8
Sorteren

Sorteren

- Van

51	03	24	86	45	30	27	63	96	50	10

- Naar

03	10	24	27	30	45	50	51	63	86	96

- Toepassingen:
  - Woordenboek
  - Googleresultaten
  - mailbox
  - Database
  - …

1. Selection Sort

Idee: zoek kleinste, dan tweede kleinste, enzovoorts

Step	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
0	51	*03*	24	86	45	30	27	63	96	50	10
1	**03**	51	24	86	45	30	27	63	96	50	*10*
2	**03**	**10**	*24*	86	45	30	27	63	96	50	51
3	**03**	**10**	**24**	86	45	30	*27*	63	96	50	51
4	**03**	**10**	**24**	**27**	45	*30*	86	63	96	50	51
5	**03**	**10**	**24**	**27**	**30**	*45*	86	63	96	50	51
6	**03**	**10**	**24**	**27**	**30**	**45**	86	63	96	*50*	51
7	**03**	**10**	**24**	**27**	**30**	**45**	**50**	63	96	86	*51*
8	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	96	86	*63*
9	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	**63**	*86*	96
10	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	**63**	**86**	*96*
11	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	**63**	**86**	**96**

```
public static void selectionSort(int[] array){
        aantalVergelijkingen = 0;
        aantalKopies = 0;
        // we selecteren telkens het kleinste element
        for(int i = 0; i< array.length-1;i++){ // laatste is niet
nodig
            int minIndex = indexMinimumVanaf(array, i);
            swap(array, i, minIndex);
            if (PRINT_TUSSEN_RESULTATEN)
                System.out.println(" > ["+i+"]
"+Arrays.toString(array));
        }
}
public static int indexMinimumVanaf(int[] array, int vanaf){
        int min = array[vanaf];
        int minIndex = vanaf;
        for(int i=vanaf+1;i<array.length; i++){ // vanaf + 1
                if (array[i] < min){
                        min = array[i];
                        minIndex = i;
                }
                aantalVergelijkingen++;
        }
        return minIndex;
```

```
/** swaps elements i and j from the array  */
private static void swap(int[] array, int i, int j){
      if (i != j){
            int tmp = array[i];
            array[i] = array[j];
            array[j] = tmp;
            aantalKopies+=3;
      }
}
```

Om de waarden van 2 variabelen te wisselen heb je altijd een tijdelijke (temporary=tmp) variabele nodig.

`x=y; y=x;` zal niet werken…

Performantie Selection Sort: n^2

Idee: zoek kleinste, dan tweede kleinste, enzovoorts

Step	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
0	51	*03*	24	86	45	30	27	63	96	50	10
1	**03**	51	24	86	45	30	27	63	96	50	*10*
2	**03**	**10**	*24*	86	45	30	27	63	96	50	51
3	**03**	**10**	**24**	86	45	30	*27*	63	96	50	51
4	**03**	**10**	**24**	**27**	45	*30*	86	63	96	50	51
5	**03**	**10**	**24**	**27**	**30**	*45*	86	63	96	50	51
6	**03**	**10**	**24**	**27**	**30**	**45**	86	63	96	*50*	51
7	**03**	**10**	**24**	**27**	**30**	**45**	**50**	63	96	86	*51*
8	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	96	86	*63*
9	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	**63**	*86*	96
10	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	**63**	**86**	*96*
11	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	**63**	**86**	**96**

***n* stappen**

***n/2* operaties**

Performantie selection sort

De basisoperaties: vergelijkingen en kopies

- Aantal kopies ***≈ 3.n = O(n)***
- Aantal vergelijkingen

$$(n-1)+(n-2)+\ldots+1=\sum_{i=1}^{n-1} n-i$$

$$=\sum_{i=1}^{n-1} i=(n-1).(\frac{1+(n-1)}{2})=\frac{n^2-n}{2}=\mathrm{O}(n^2)$$

2. Bubble Sort

Idee: 'bubbel' kleinste-tot-dan-toe naar boven

Step	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
0	51	*03*	24	86	45	30	27	63	96	50	10
1	***03***	51	*10*	24	86	45	30	27	63	96	50
2	**03**	***10***	51	24	*27*	86	45	30	*50*	63	96
3	**03**	**10**	***24***	51	27	*30*	86	45	50	63	96
4	**03**	**10**	**24**	***27***	51	30	*45*	86	50	63	96
5	**03**	**10**	**24**	**27**	***30***	51	*45*	*50*	86	63	96
6	**03**	**10**	**24**	**27**	**30**	***45***	51	50	*63*	86	96
7	**03**	**10**	**24**	**27**	**30**	**45**	***50***	51	63	86	96
8	**03**	**10**	**24**	**27**	**30**	**45**	**50**	**51**	63	86	96

```
public static void bubbleSort(int[] array){
      aantalVergelijkingen = 0;
      aantalKopies = 0;
      boolean sorted;
      int i=0;
      if (PRINT_TUSSEN_RESULTATEN)
       System.out.println(" > ["+i+"] "+Arrays.toString(array));
      do {
            sorted = true;
            for(int j=array.length-1; j>i; j--){
                  if (array[j] < array[j-1]){
                        swap(array, j, j-1);
                        sorted = false;
                  }
                  aantalVergelijkingen++;
            }
            i++; // weten dat het i'de element op zijn plaats
staat
            if (PRINT_TUSSEN_RESULTATEN)
            System.out.println(" > ["+i+"] "+Arrays.toString(array));
      } while(!sorted);
}
```

Performantie Bubble Sort

- Aantal vergelijkingen
- **Worst case**: evenveel als selection sort $O(n^2)$
  - Bvb in omgekeerde volgorde
- **Best case**?
  - Als array reeds gesorteerd is $O(n)$
  - Slechts enkelen niet op zijn plaats

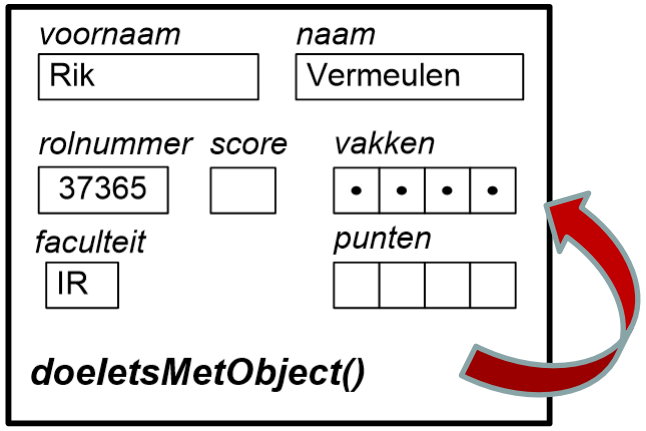
“Static” in java

denken in objecten

static vs non-static

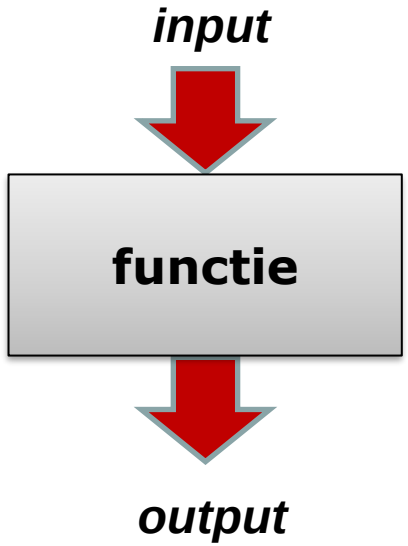
Niet in cursus
Hoort bij boek 1, p. 53 1.6.1

Gewone methode



elk hun eigen attribuutwaarden
Methode kan attributen bekijken of veranderen

statische methode



Methode gebruikt enkel input om output te berekenen.
Kan wel statische variabelen gebruiken.

Handige van static

- Je hebt geen referentie naar het object nodig
  - Referenties naar object: zie pagina 16-17 van de cursus
  - Hoofdstuk 1 kan nu van pas komen!
- Statische methode Math.sin(x) kan je overal oproepen
  - Als niet-statisch: je moet een Math-object aanmaken
- Statische variabele kan je overal oproepen:
  - Een statische variabele hangt niet vast aan een object, maar aan de klasse. Je hebt 1 variabele voor de klasse.
  - aantalIteraties (zie pagina 36 van de cursus)

Probleem met static

- Vanuit object kan je static dingen oproepen,
- maar niet omgekeerd: **vanuit static kan je geen objectattributen of methodes oproepen van een gewoon object!!**
  - Compiler (Eclipse) zal een fout geven

  Cannot make a static reference to the non-static method…
  - Voorbeeld: statische methode roept gewone methode op
  - Dan moet je alles *static* maken… en loopt het fout
  - (Of je moet statische referenties bij gaan houden…)
- Niet doen dus

Als je dit tegen komt is het foutief gebruik van static

Wanneer static te gebruiken?

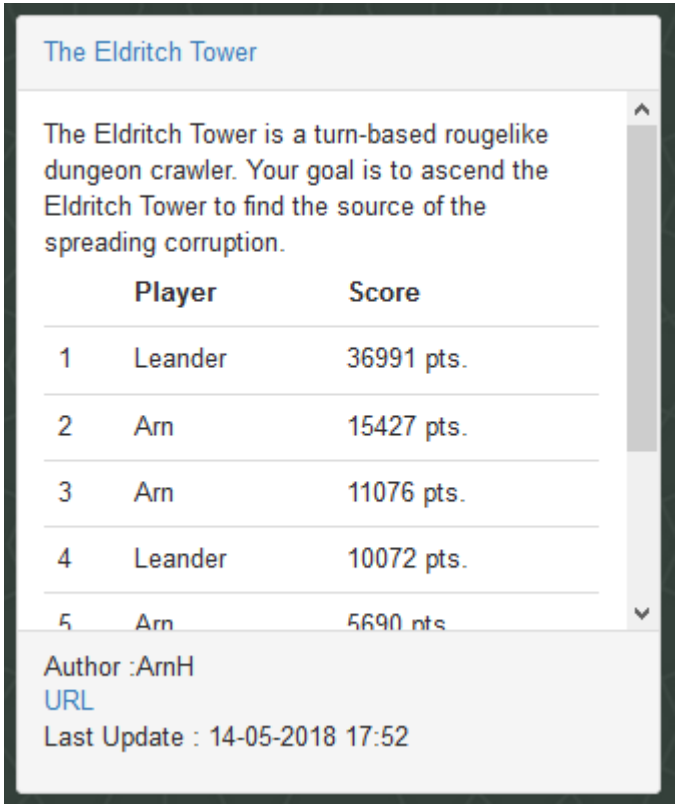
Zo weinig mogelijk…

- **Algemene functies**, die niets aan een object veranderen
  - Voorbeeld: berekenen gemiddelde, hoek ofzo
  - Als je een object wilt veranderen gaat het fout (oproepen niet-statische variabele
- **Algemene, globale variabelen**
  - 1 waarde die duidelijk algemeen moet zijn over alle objecten heen
  - Voorbeeld: teller, ‘mode’ van je GUI
- Als je functie **een 2e waarde** moet teruggeven
  - Kan dus niet in
  - Zie cursus p. 26 (aantalIteraties) en hoofdstuk over sorteren

Highscore server

https://rapptor.vub.ac.be/gamemanager/

Bijhouden highscores op server



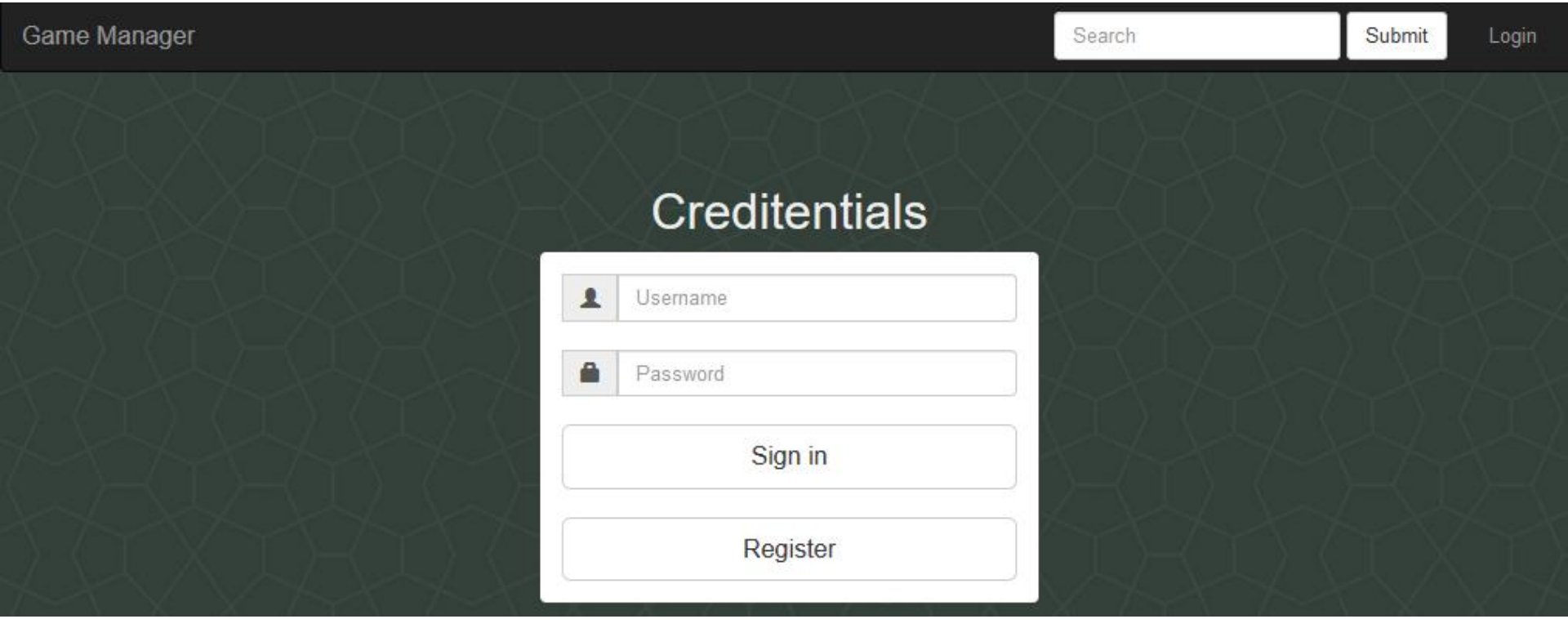
- Je kan de high scores tonen in je spel
- Of op parallel website
- Of via link http://rapptor.vub.ac.be/gamemanager/static/game/XXX

met XXX het nummer van uw spel

- Of als html-code in een website:

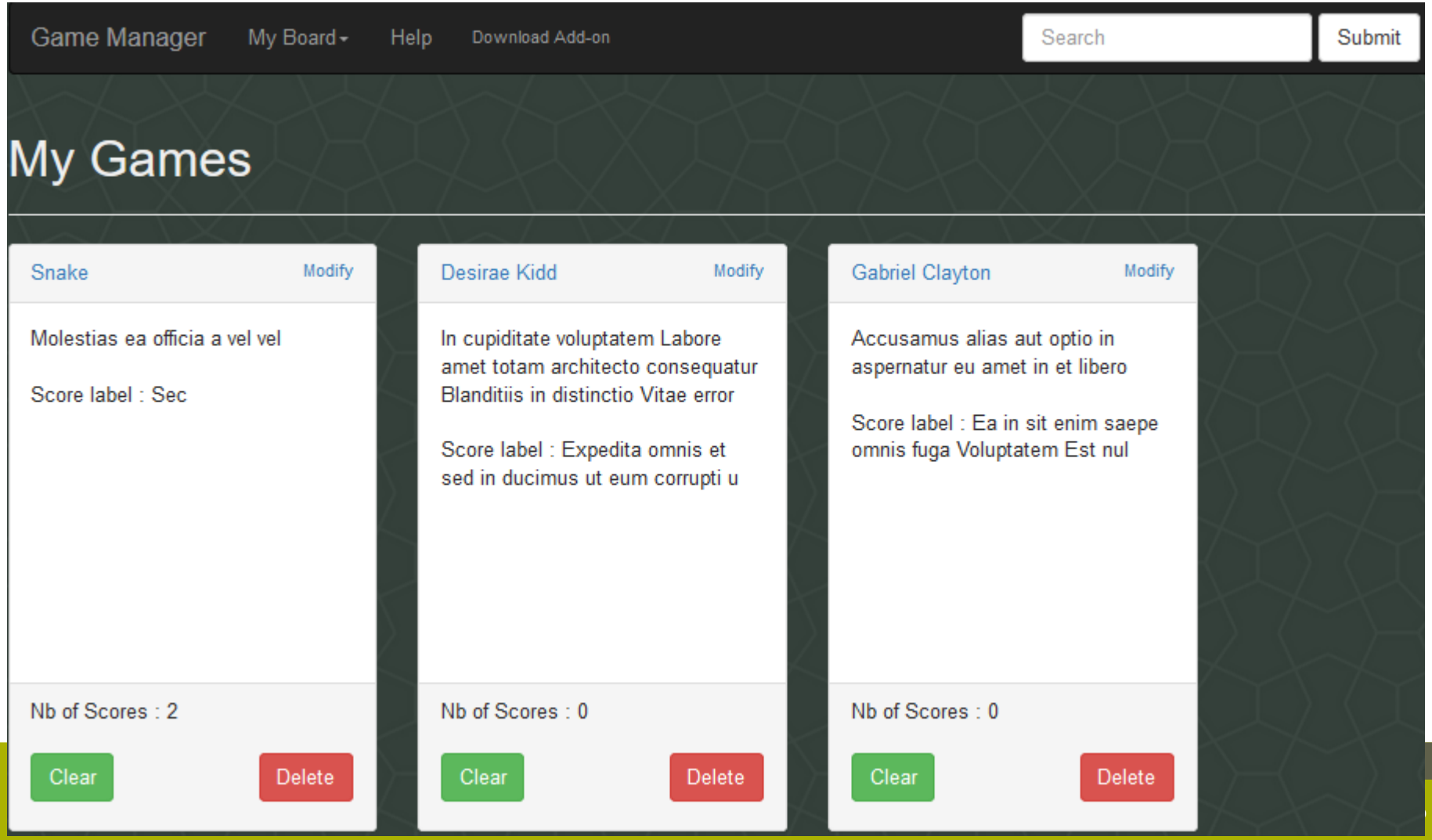
```
<iframe width="500" height="315"
src=http://rapptor.vub.ac.be/gamemanager/static/game/XXX frameborder="0"></iframe>
```

Stap 1: aanmaken gebruiker

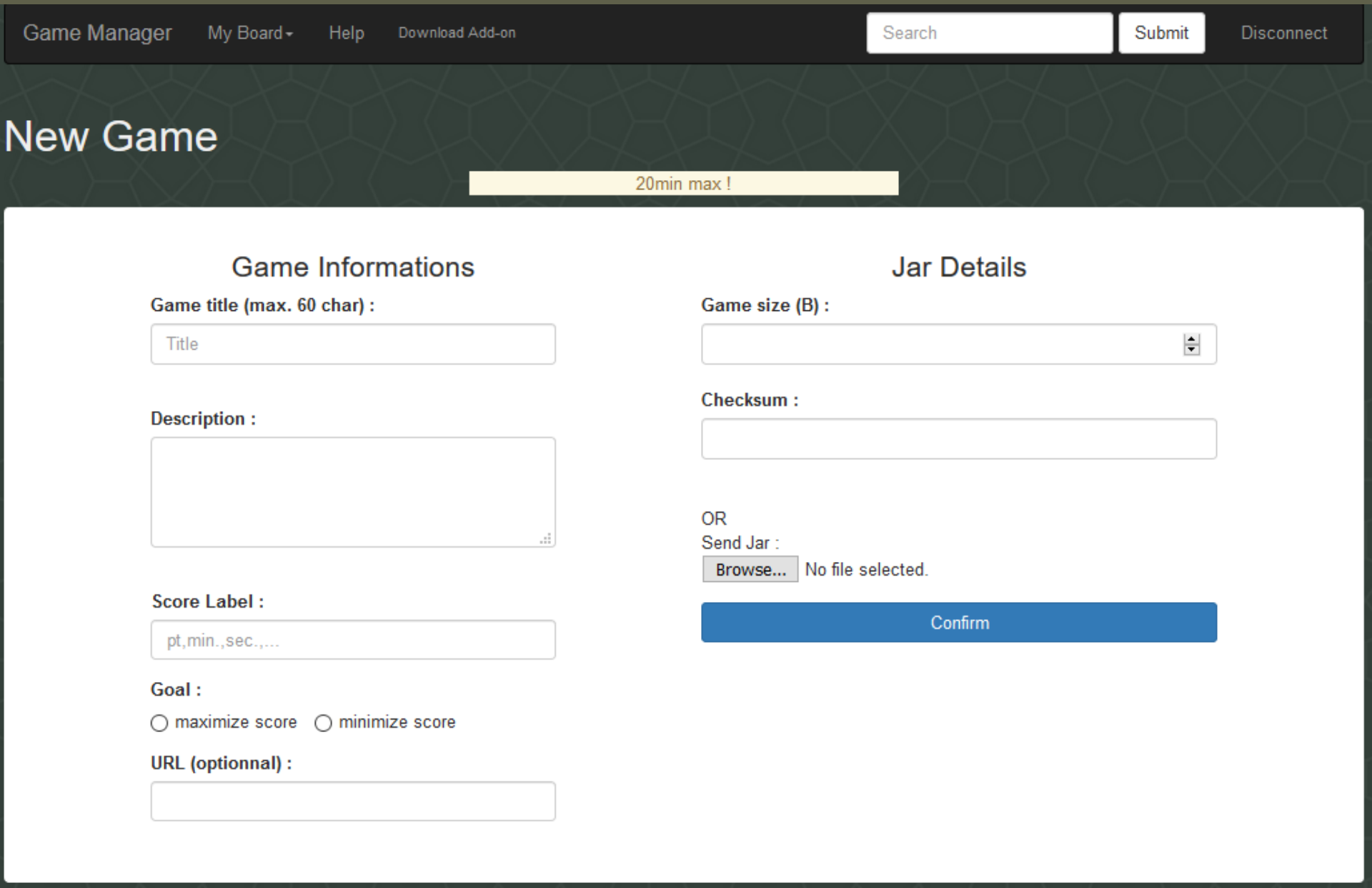


- Code nodig: krijg je nog via mail

Inloggen


Game Manager
My Board
Help
Download Add-on
Search
Submit
My Games
Snake
Modify
Molestias ea officia a vel vel
Score label : Sec
Nb of Scores : 2
Clear
Delete
Desirae Kidd
Modify
In cupiditate voluptatem Labore amet totam architecto consequatur Blanditiis in distinctio Vitae error
Score label : Expedita omnis et sed in ducimus ut eum corrupti u
Nb of Scores : 0
Clear
Delete
Gabriel Clayton
Modify
Accusamus alias aut optio in aspernatur eu amet in et libero
Score label : Ea in sit enim saepe omnis fuga Voluptatem Est nul
Nb of Scores : 0
Clear
Delete

Stap 2: aanmaken game (Submit game)



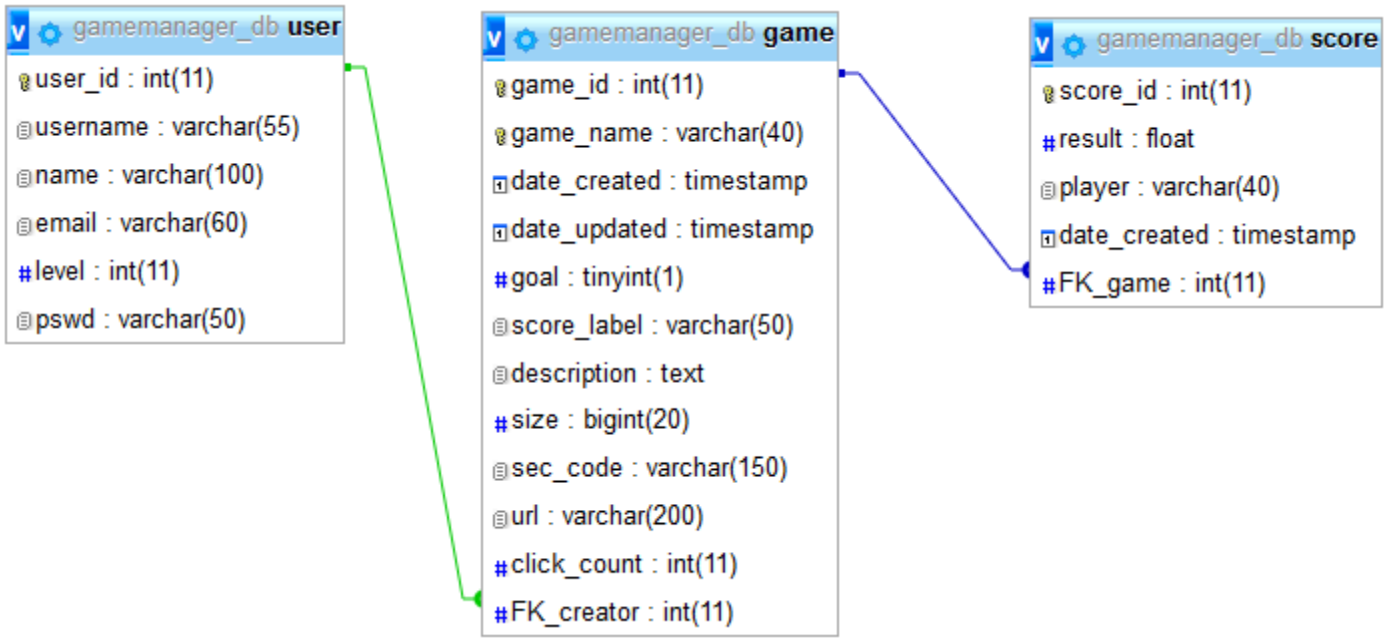
Stap 3: toevoegen aan java

- GM.jar toevoegen aan project (zie documentatie)
- API (Application Programming Interface)
  - Class **ApiBoard** in package `import gm.ApiBoard;`
  - `List<Score> getResults(String gameName)`
  - `String addResult(float result, String playerName, String gameName)`
-
```
public class Score {
    public String score;
    public String date;
    public String player;
}
```
=> democode is bijgeleverd

Stap 4: Security

- Bij verzenden van resultaten
  - Gm.jar zal getal berekenen en meesturen
    - Als project in Eclipse: aantal bytes (Game size)
    - Als jar-file: checksum (=hashfunctie uit de encryptie; zie laatste les)
  - Wordt gecontroleerd als gespecifieerd bij de Game Information op website (stap 2)
    - Als je ze allebei op nul zet wordt het niet gecontroleerd
    - Als getallen niet overeenkomen: resultaat wordt niet weggeschreven
  - Bereken juiste waarde:
    - Game size met `ApiBoard.getProjectInfo()`
    - Checksum: op website bij game information kan je jar-file uploaden

Achter de schermen: database



- Lijkt op opslaan van objecten van 3 klassen (user, game, score)

Hoofdstuk 7: Operating Systems

Bedrijfscomputers

Initieel waren er voornamelijk bedrijfscomputers

- Centrale computer
  - "mainframe" systemen
  - Voornaamste fabrikant: **IBM**

Interactiviteit via *Terminal*

- Scherm van 24 lijnen van 80 tekens groot

Gebruikers werken op de centrale computer via ‘domme’ terminals die enkel het scherm tonen en de input van de gebruiker (via toetsenbord) doorgeven.

Toepassingen mainframe

- Gegevens (bvb boekhouding) van banken, bedrijven, winkels etc
- Gegevens worden bijgehouden in *database*
  - *Database = gestructureerd bijhouden van gegevens*
- Eigenschappen mainframe:
  - *Betrouwbaarheid (heel belangrijk)*
  - *Robuust (crasht bijna nooit)*
  - *Veiligheid (security) van gegevens (bvb bankgegevens)*

IBM gaat voor Personal Computer

- Computer voor “thuis”
- Kan op eigen kracht werken (niet geconnecteerd met centrale computer)
- De PC is geboren!
- IBM: op dat moment het grootste informaticabedrijf
  - Concentreert zich op hardware

De PC ontketent een nieuwe revolutie in de informatica…

Microsoft Albuquerque Group, December 7, 1978. Top row: Steve Wood, Bob Wallace, Jim Lane. Middle row: Bob O'Rear, Bob Greenberg, Marc McDonald, Gordon Letwin. Front row: Bill Gates, Andrea Lewis, Marla Wood, Paul Allen. Missing from photograph are Ric Weiland and Miriam Lubow. Photo courtesy Microsoft Archives.

IBM heeft Operating System nodig

- IBM gaat langs bij Bill Gates en zijn ‘hippie’-vrienden
- Zitten thuis te programmeren
- **MicroSoft** is geboren
- Steken DOS (Disc Operating System) in elkaar
- Nog steeds terug te vinden in Windows
  - Cmd-window
  - Programma’s start je met commando (en eventueel argumenten – dit zijn de “String[] args” van de main)

- *Commando-based:*

Je geeft commando's

In de terminal of shell


```
                        Welcome to MS-DOS 2011

For a list of simple commands, please type HELP
>HELP

LOC - Displays the location of the currently operating program
OPEN <directory> - Opens and displays contents of the directory entered
EXECUTE <directory/filename> - Executes a file within that directory
REBOOT - Reboots current session (Confirmation will be required to prevent
accidental reboots)
SYS - Displays system properties
>RUN: person .EXE

What was that? I can't understand you.
>realityconfig

Attempting to configure your reality...
Enter password:
>^C to abort.

Incorrect password. Exiting function.
>SYS

Main Harddrive: Q:/
Space Used: 105246/6152374 KB
Processing Power: 10 GH
RAM: 3000000 KB
>OPEN GAMES

What are you, stupid? GAMES isn't a directory! Try Q:/!
>EXECUTE adventure.EXE

Try adding a directory to that, before you seem any less intelligent.
>_
```

Bill Gates wordt rijkste man ter wereld

0.1 →

Macht ligt vanaf nu bij software en vooral het besturingsysteem

1984

IBM maakt historische vergissing door op hardware te blijven mikken
Ze mist de softwareboot compleet…

IBM

Lancering PC

Microsoft wint het pleit

IBM heeft definitief hegemonie verloren

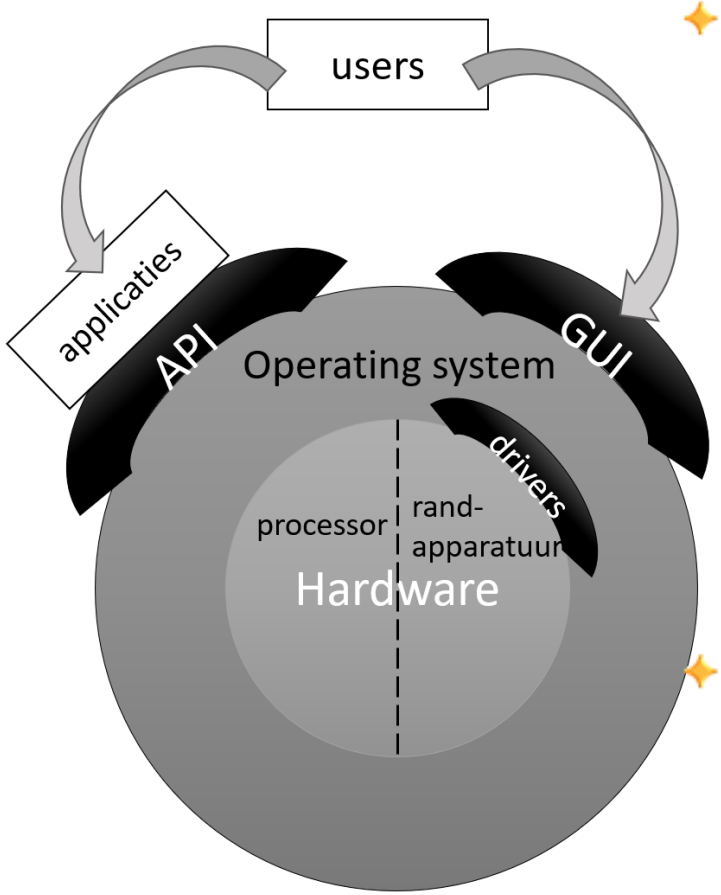
Besturingsysteem

- = Operating System (OS)
- OS regelt en organiseert de computer
- OS wordt van harde schijf gestart bij het *booten*
  - BIOS zorgt hiervoor

Een besturingssysteem (in het Engels operating system of afgekort OS) is een programma (meestal een geheel van samenwerkende programma's) dat na het opstarten van een computer in het geheugen geladen wordt en alle mogelijkheden van de computer aan de gebruiker. Het OS biedt ook de functionaliteiten aan om andere programma's - applicaties genoemd - uit te voeren.

Application Programming Interface (API)

- OS verstopt de details van de hardware voor de gebruiker en voor applicaties dmv een API



- Deze worden op een uniforme wijze aan de applicaties aangeboden. De API abstraheert de toegang tot de verschillende randapparatuur, zonder OS moet elk programma zelf instaan voor het aansturen van randapparatuur (zoals printer, beeldschermen, harde schijf). Een gebruikersprogramma is enkel afhankelijk van het OS, niet van de randapparatuur.
- Het communiceren van het OS met het randapparaat gebeurt via een *driver* (die wel specifiek is voor elke apparaat zoals printer).

Interactiviteit

- Geen interactiviteit = batch programma
  - Programma en gegevens worden op voorhand klaargemaakt
  - Tijdens de uitvoering kan je niet interageren met het programma
  - Resultaten worden op het einde als geheel gepresenteerd
- Commando-gebaseerd
  - Cf DOS, linux shell
  - De commando's worden 'geïnterpreteerd' en het programma gestart
  - Je kan een lijst van commando's doorgeven ('batch')
- Grafische User-Interface (GUI)
  - Windows, muis, toetsenbord, touch screen

Hoofdtaken OS (vervolg)

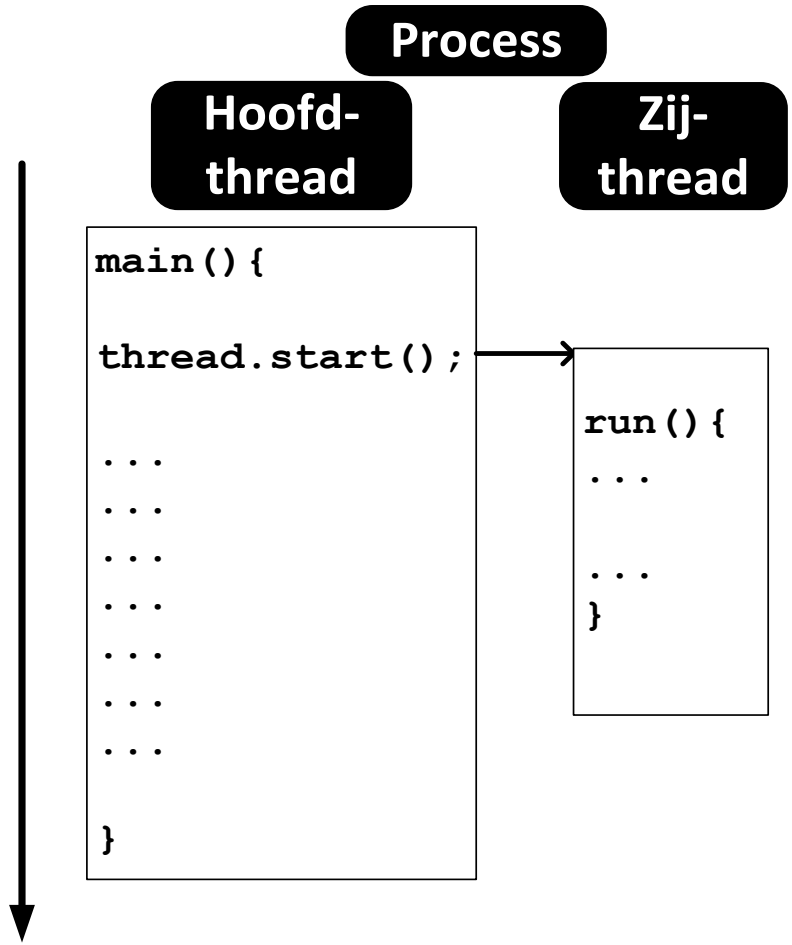
- Verdelen van toegang tot systeembronnen (RAM-geheugen, opslag, printer etc.) tussen actieve programma's
  - Elk programma krijgt een deel van het werkgeheugen toegewezen (java: standaard 64MB, je kan dit anders instellen)
  - OS voorkomt dat programma buiten zijn deel gegevens kan lezen of schrijven (beveiliging!)
- Aanbieden gegevens (files) en applicaties aan gebruiker
  - OS beheert het filesysteem (georganiseerd in een boomstructuur dmv folders)
- Verdelen van processortijd over de aktieve programma’s
  - Zie verder

Task Manager

- Windows: start via Control-Alt-Delete
- Toont actieve *applicaties* en *processen*, alsook processorgebruik
  - Applicaties: van gebruiker
  - Processen: naast de processen van de applicaties, ook processen en achtergrondprocessen ('services') van operating system en applicaties
- Operating System verdeelt cycles van processor (CPU) over de verschillende processen (*process scheduling*)

Op een modern OS kan je meerdere programma's tegelijk draaien. Het uitvoeren van een programma resulteert in een proces. Een achtergrondproces is in feite ook een gewoon programma, maar ze is niet zichtbaar voor de gebruiker. De achtergrondprocessen zorgen voor het beheer van het systeem of bieden services aan (zoals het sharen van je (muziek-)files en het checken van je mailbox).

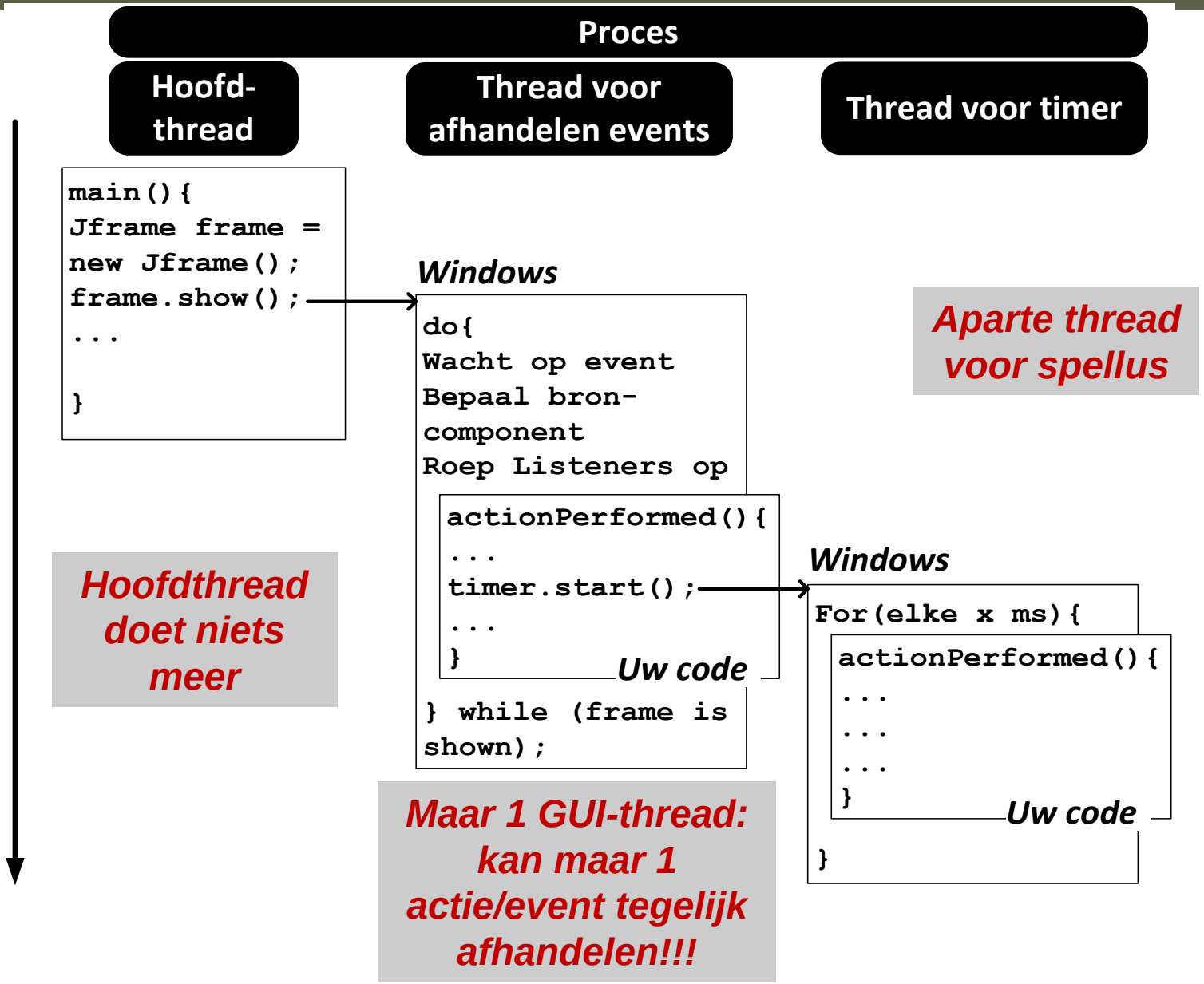
Threads van processen



Elk programma dat uitgevoerd wordt komt overeen met 1 onafhankelijk *proces*, maar elk proces kan bestaan uit meerdere *threads*. Elke thread voert een sequentie van instructies uit gespecifieerd door de code. Een sequentie kan je zien als een ‘draad’, vandaar de benaming. Vele moderne toepassingen zijn zelf opgebouwd uit een aantal threads die simultaan uitgevoerd worden. Het tekstverwerkingsprogramma *Word* bv. gebruikt verschillende threads om teksten en figuren op het scherm te tonen, zodanig dat wanneer men snel door een tekst wenst te lopen men niet hoeft te wachten op het tekenen van alle figuren. Een andere thread zal tegelijkertijd je taalfouten opsporen (de *spelling checker*). Threads van eenzelfde proces werken dus met dezelfde gegevens, terwijl elk proces zijn eigen gegevens heeft (word document, excel sheet, mail, chat, …).

Threads van je project

Boek I: p 50



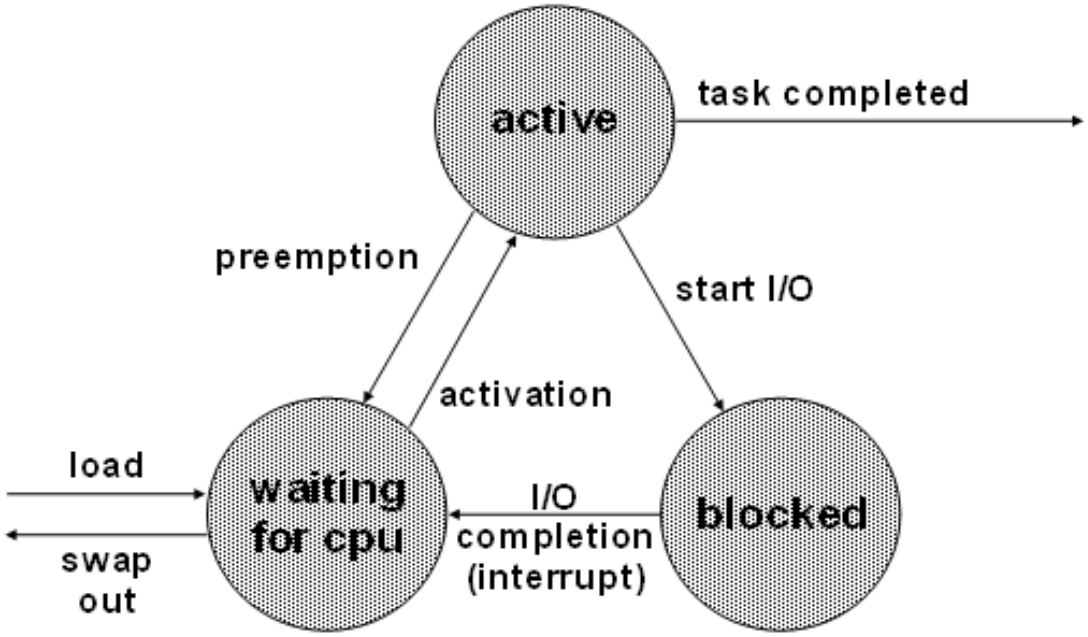
Process scheduler

De *process-scheduler* is het deel van het besturingssysteem dat op elk ogenblik bepaalt welk van de programma's toegewezen wordt aan de centrale verwerkingseenheid (CPU). Het hart van de computer, de processor of CPU, kan immers maar 1 programma tegelijk uitvoeren. De processor is immers opgebouwd volgens de *Von Neumann-architectuur*. Karakteristiek voor deze architectuur is dat hij één programma stap-voor-stap uitvoert. De CPU zal om beurten de programma's uitvoeren.

Moderne computers hebben intussen meerdere processorcores (dual core, quadcore, …). Dat zijn dan in feite 2, respectievelijk 4 onafhankelijke processoren die elk één proces tegelijkertijd kunnen uitvoeren.

De *process-scheduler* zal per core de processortijd verdelen over de lopende processen en threads: elk proces/thread krijgt een periode ('time slice') toegekend.

Toestanden van een proces



De lopende processen, kunnen zich, op elk ogenblik in drie toestanden bevinden:

1) *actief*: de centrale verwerkingseenheid (CPU) is aan het proces aan het werken; dwz. dat instructies van dat programma opgehaald worden door de stuureenheid en worden uitgevoerd.

2) *geblokkeerd*: een in- of uitvoeroperatie (Input/Output of I/O) is aan de gang en het proces moet wachten tot het einde ervan;

3) *wachtend*: het proces zou kunnen uitgevoerd worden, maar de centrale verwerkingseenheid is niet beschikbaar, ze is instructies aan het uitvoeren van een ander proces. Het proces moet wachten tot de proces-scheduler processorcycles ter beschikking stelt.

Overgang van 1 proces naar een ander

- Als het actieve proces op I/O (input/output) moet wachten
  - Dikwijls wordt bij I/O het OS geactiveerd, omdat die de I/O beheert (bvb toegang tot files controleert)
- Of als de toegekende 'time slice' die een proces toegekend krijgt op is
  - Bij starten van een proces wordt ook een timer gestart die na de time slice een *interrupt* geeft
  - *Interrupt* zorgt dat de processor met het actieve proces stopt en de scheduler van het OS wordt gereactiveerd. Deze beslist welk proces nu een time slice toegekend krijgt.
- Overgang naar een nieuw proces*: context switch*
  - de staat van het oude proces wordt opgeslagen en dat van het nieuwe wordt geladen

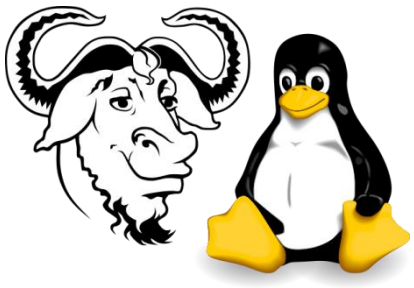
Interrupts

Met een *Interrupt* (‘onderbreking’) wordt een lopend programma onderbroken. Deze is voorzien in de hardware van de processor. Je kan immers niet verwachten dat een lopend programma zelf zal checken of hij verder mag gaan. Ook kan het OS er niet van uitgaan dat elk programma in een redelijke tijd stopt. Het OS moet op elk moment een programma kunnen onderbreken. Anders zou een ‘oneindige lus’ volledig beslag leggen op je processor zonder dat je er iets aan kan doen (behalve dan de computer herstarten).

Let op: het OS is ook niets meer dan een programma. Er is geen ‘big brother’ in je computer die toekijkt wat de processor doet. Controle over het actieve proces wordt verkregen door *interrupts.* Processoren bevatten een hardwareschakeling waardoor een programma kan onderbroken worden door middel van een *interrupt*. Vervolgens bepaalt de *interrupt handler* welk programma uitgevoerd moet worden. Aan de hand van de oorsprong van de interrupt veroorzaakt deze een sprong naar het gewenste programma. Als er bijvoorbeeld input binnenkomt (muisklik, internetbericht). Of voor de controle door het OS. Stel dat een programma een file wil openen, dan wordt het OS even geactiveerd om te controleren of het programma dat wel mag. Als de file nog open is in een ander programma bvb, wordt de toegang geweigerd.

Unix & Linux

- UNIX operating system: zoals mainframe initieel bestemd voor bedrijfscomputers
  - User moet inloggen
  - Heeft eigen files op *server* (in zijn *home*)
  - Enkel administrator kan dingen aan systeem veranderen
- Linux: Open Source-versie van UNIX
  - Open Source volgt de GNU-regels: de code mag vrij (gratis) gebruikt worden zolang er geen geld voor gevraagd wordt
- *Apple* gebruikt nu ook een Unix-versie
- *Android* van Google is java op Linux



Conclusie: in de consumentenmarkt is het Windows of Linux.

Strategie: open versus gesloten

- **Microsoft's Windows**:
  - Open besturingssysteem
  - Iedereen mag er software voor ontwikkelen
    - Geeft andere bedrijven kansen
  - Microsoft concentreerde op besturingssysteem & software
    - niet op hardware en niet op alle software
- **Apple**:
  - Hield en houdt controle over het hele systeem, software & hardware
    - Werkte initieel tegen hun (eind '90 bijna failliet)
    - Via "apps" kan je software aanbieden
  - Pakt nu succesvol uit met 'totaalproducten'
  - Gebruiksgemak, stijl en design steeds prioritair

Analoog bij OS voor smartphones: Android versus Apple OS