

# Velkommen til Introduktion til Programmering

---

- **Kurset har ca. 230 studerende fordelt på 10 øvelseshold**
  - 7 hold med nye studerende på datalogi bacheloren (DA1-DA7)
  - 2 hold med nye studerende på it bacheloren (IT1-IT2)
  - 1 hold med ældre studerende fra andre studieretninger (Hold 1)
- **Jeg hedder Kurt Jensen og er professor på Institut for Datalogi**
  - Jeg har undervist i "Introduktion til programmering" gennem rigtig mange år (med tilsammen 4.000 studerende)
  - Derudover har jeg i næsten 20 år været leder af instituttet
  - Det er jeg ikke længere, så nu kan jeg lave andre sjove og interessante ting som f.eks. at undervise jer
  - I kan Google mig ved at skrive "Kurt Jensen au"
- **Til at hjælpe mig har jeg 13 studenterinstruktorer**
  - Primært 2. og 3. års studerende på datalogi og it-produktudvikling



# Forelæsning Uge 1 – Mandag

- **Hvad er programmering?**
  - Program der kan løse Sudoku opgaver (eksempel)
  - Programmering og problemløsning (generelt)

- **Agenter og metoder**

- **UML specifikationssproget**

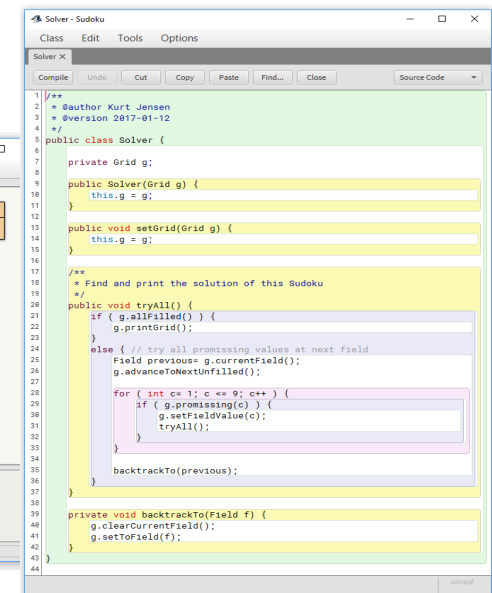
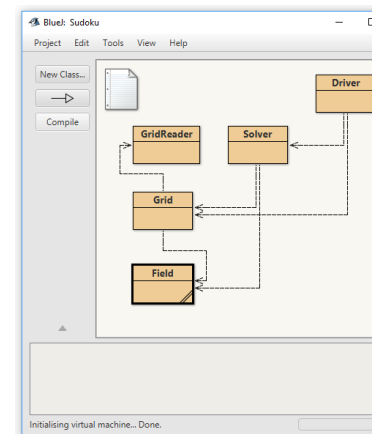
- Klassediagrammer
- Sekvensdiagrammer



- **Information om kurset**

- Hvad kan I forvente at lære?
- Undervisningsprincipper
- Demo af programmeringsomgivelser

- **Afleveringsopgave i uge 1**

A screenshot of the BlueJ IDE showing the source code for a Sudoku solver. The code is in a file named "Solver.java" and includes a class definition for "Solver". The code is as follows:

```
1 /**  
2  * @author Kurt Jensen  
3  * @version 2017-01-12  
4  */  
5 public class Solver {  
6  
7     private Grid g;  
8  
9     public Solver(Grid g) {  
10         this.g = g;  
11     }  
12  
13     public void setGrid(Grid g) {  
14         this.g = g;  
15     }  
16  
17     /**  
18     * Find and print the solution of this Sudoku  
19     */  
20     public void tryAll() {  
21         if (g.allFilled()) {  
22             g.printGrid();  
23         }  
24         else { // try all promising values at next field  
25             Field previous = g.currentField();  
26             g.advanceToNextUnfilled();  
27  
28             for (int c = 1; c <= 9; c++) {  
29                 if (g.promising(c)) {  
30                     g.setFieldValue(c);  
31                     tryAll();  
32                 }  
33             }  
34             backtrackTo(previous);  
35         }  
36     }  
37  
38     private void backtrackTo(Field f) {  
39         g.clearCurrentField();  
40         g.setToField(f);  
41     }  
42  
43 }  
44
```

# ● Program til at løse Sudoku opgaver

- **Opgaven er at udfylde de manglende felter, således at,**
  - hver af de 9 rækker
  - hver af de 9 søjler
  - hvert af de 9 kvadrater

**indeholder hvert af cifrene 1-9 præcis én gang**

Opgave

	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	



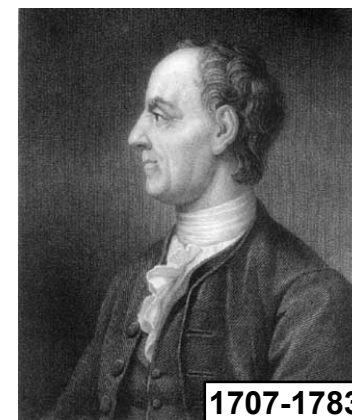
Løsning

9	6	3	1	7	4	2	5	8
1	7	8	3	2	5	6	4	9
2	5	4	6	8	9	7	3	1
8	2	1	4	3	7	5	9	6
4	9	6	8	5	2	3	1	7
7	3	5	9	6	1	8	2	4
5	8	9	7	1	3	4	6	2
3	1	7	2	4	6	9	8	5
6	4	2	5	9	8	1	7	3

# Lidt Sudoku historik

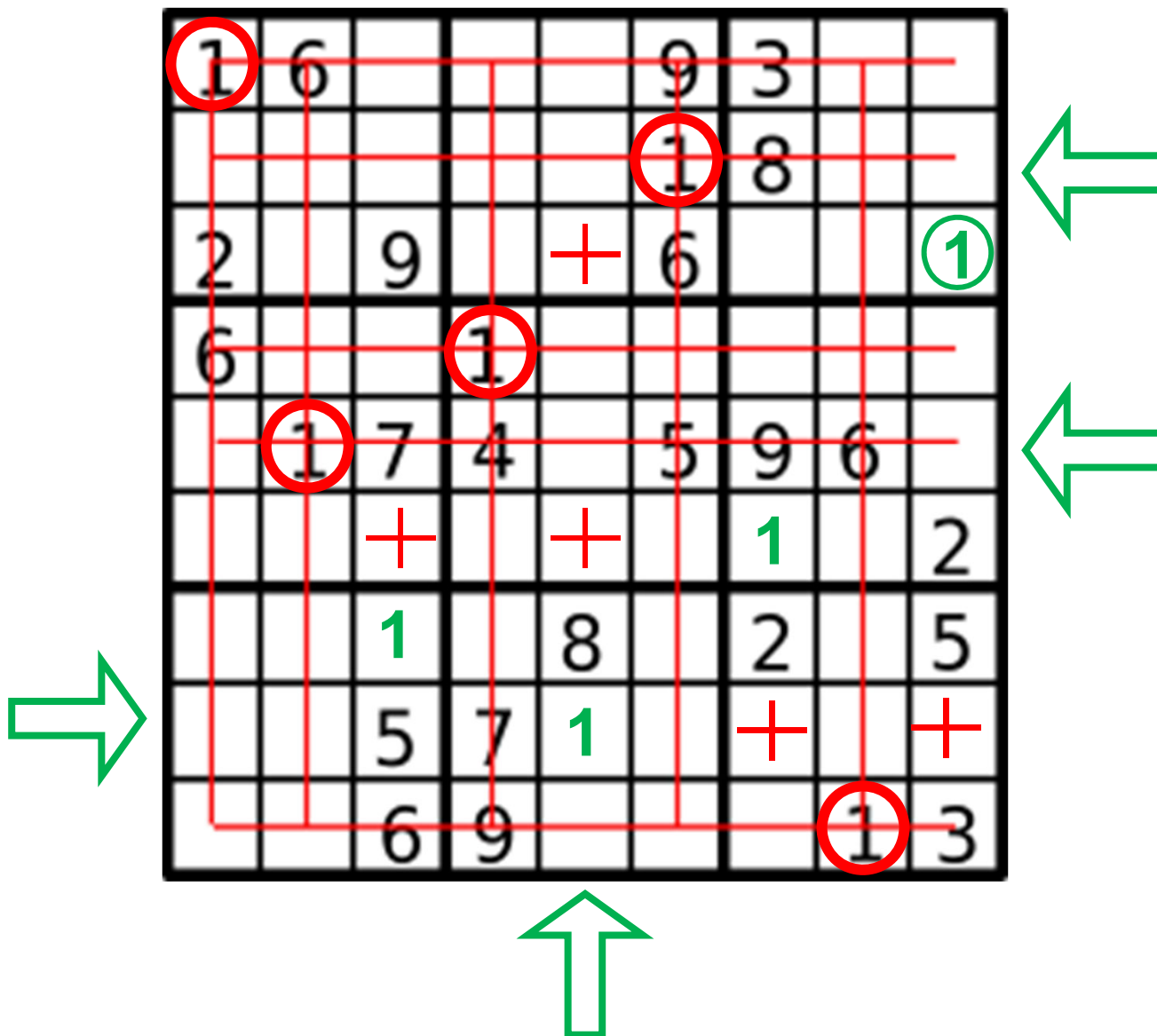
---

- **Sudoku er inspireret af latinske kvadrater**
  - Introduceret af schweizeren Leonhard Euler
  - En af de største matematikere i 17. hundredetallet
- **Sudoku blev enormt populær fra 1984 og frem**
  - Specielt i Japan, men også i resten af verden
  - "Sudoku" er en forkortelse af den japanske sætning "Suji wa dokushin ni kagiru" som betyder "**tallene må kun forekomme én gang**"
  - Mange danske aviser har stadig Sudoku opgaver
- **Sudoku og computere**
  - Sudoku opgaver kan **konstrueres** ved hjælp af computere
  - Her skal vi i stedet se på, hvordan Sudoku opgaver kan **løses** ved hjælp af computere – dvs. ved hjælp af programmering



# Strategi med udgangspunkt i ciffer

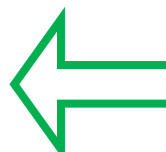
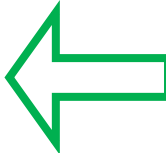
①



# Strategi med udgangspunkt i felt

---

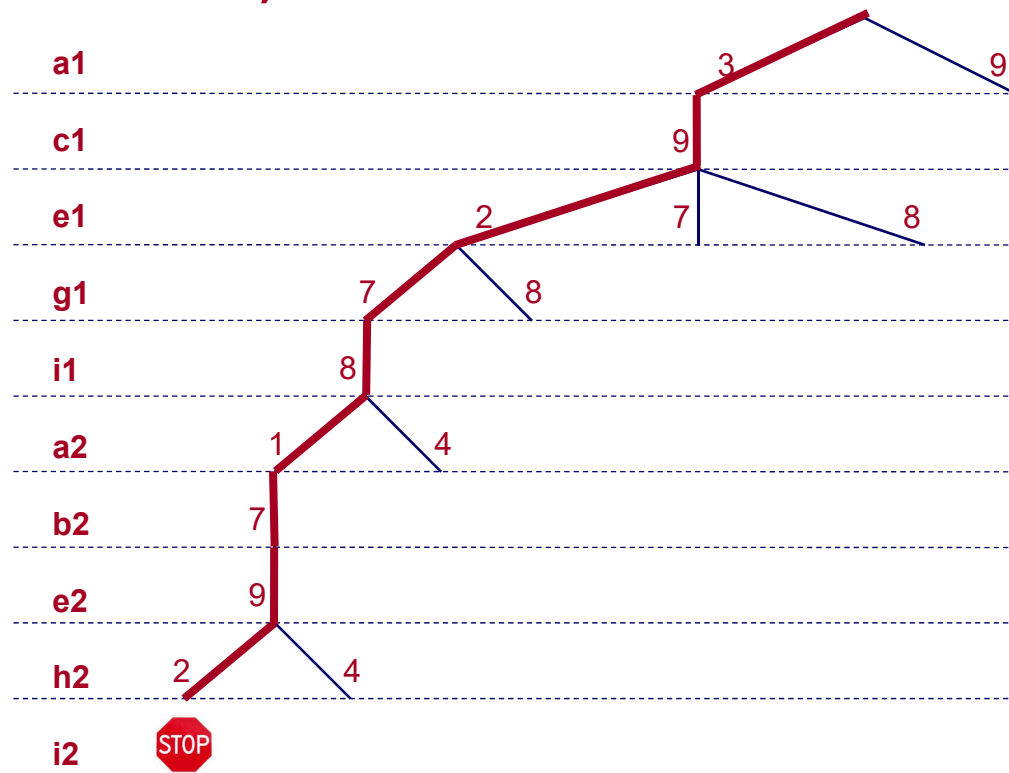
6	5	9	2	8	3	7	1	4
8	1	2		9	4			3
7	4	3	5	6	1	9		2
	2	1		4			7	
4	3	7		1		2		8/9
	6	8		2	7	4		1
1	9	6	8	7	2	3	4	5
3	7	4	1	5	9	8	2	6
2	8	5	4	3	6	1	9	



# Algoritme til løsning af Sudoku opgaver

- Systematisk afprøvning af alle muligheder (ved hjælp af strategi nummer 2)

	a	b	c	d	e	f	g	h	i
1	3	6	9	1	2	4	7	5	8
2	1	7	8	3	9	5	6	2	
3	2								1
4	8			4		7			6
5			6				3		
6	7			9		1			4
7	5								2
8			7	2		6	9		
9		4		5		8		7	

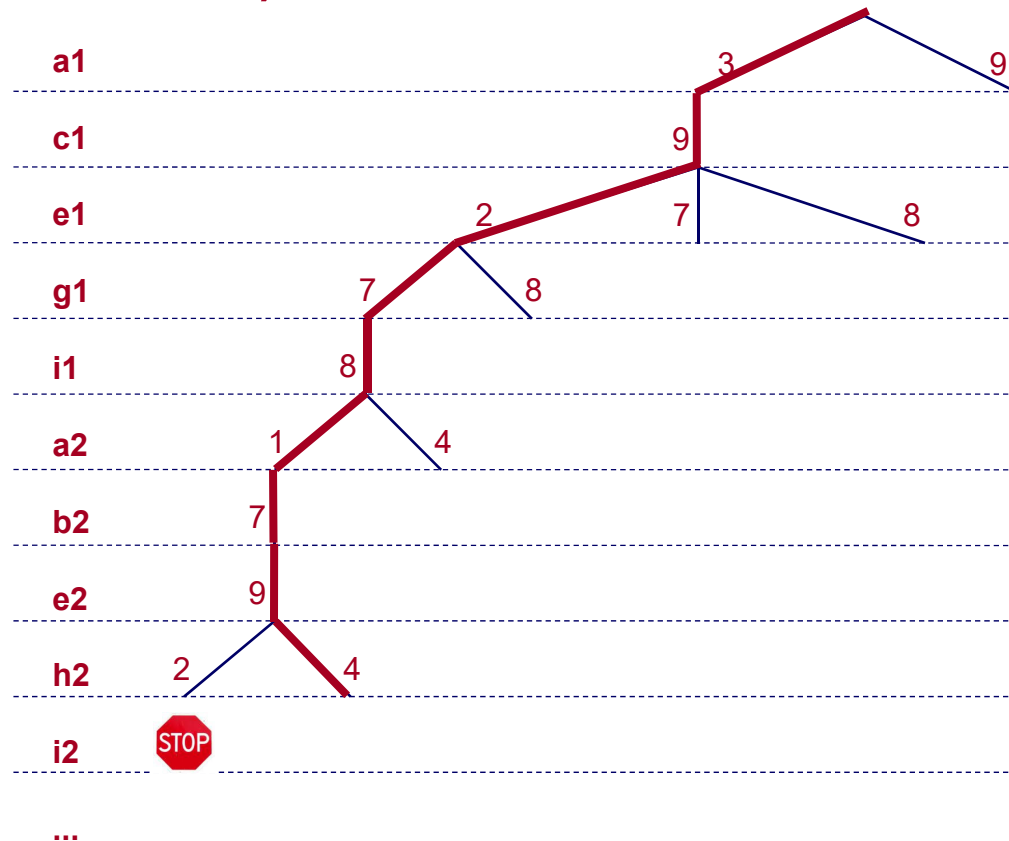


Vi kan ikke komme videre frem (vejen er blokeret)  
Vi må gå tilbage af den sti vi kom (indtil vi kan tage et andet vejvalg)  
Det kaldes backtracking

# Algoritme til løsning af Sudoku opgaver

- Afprøv systematisk alle muligheder (ved hjælp af strategi nummer 2)

	a	b	c	d	e	f	g	h	i
1	3	6	9	1	2	4	7	5	8
2	1	7	8	3	9	5	6	4	
3	2								1
4	8			4		7			6
5			6				3		
6	7			9		1			4
7	5								2
8			7	2		6	9		
9		4		5		8		7	

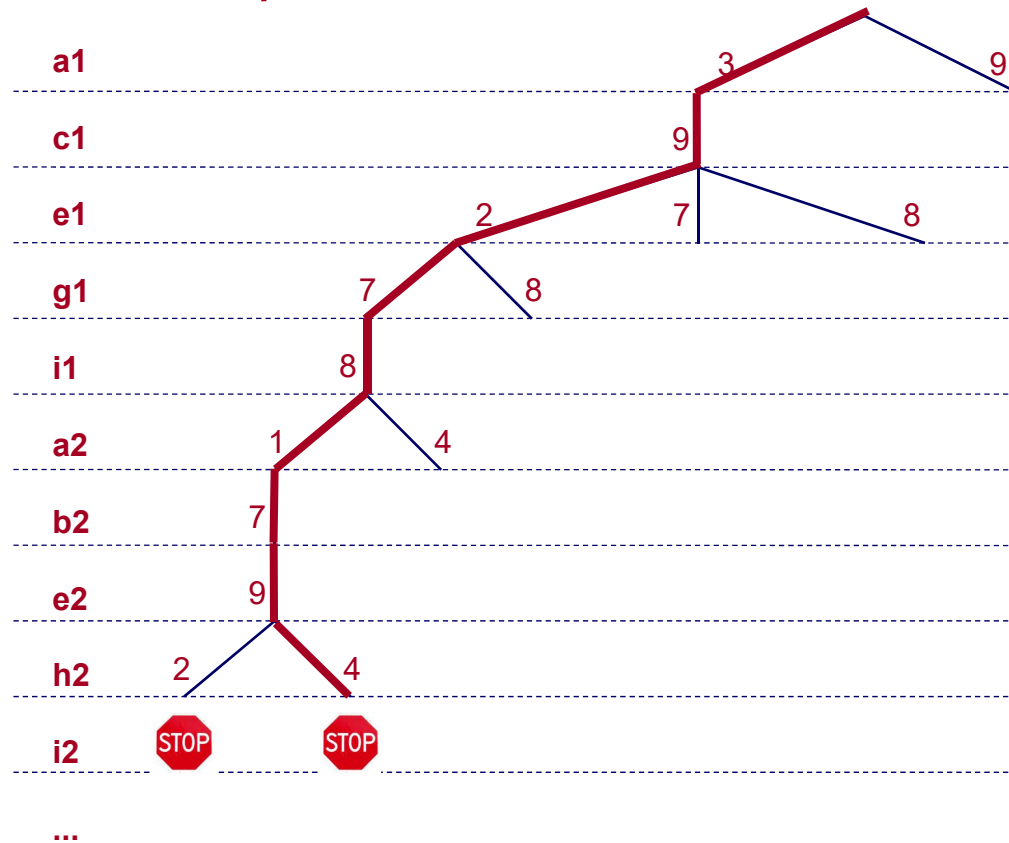




# Algoritme til løsning af Sudoku opgaver

- Afprøv systematisk alle muligheder (ved hjælp af strategi nummer 2)

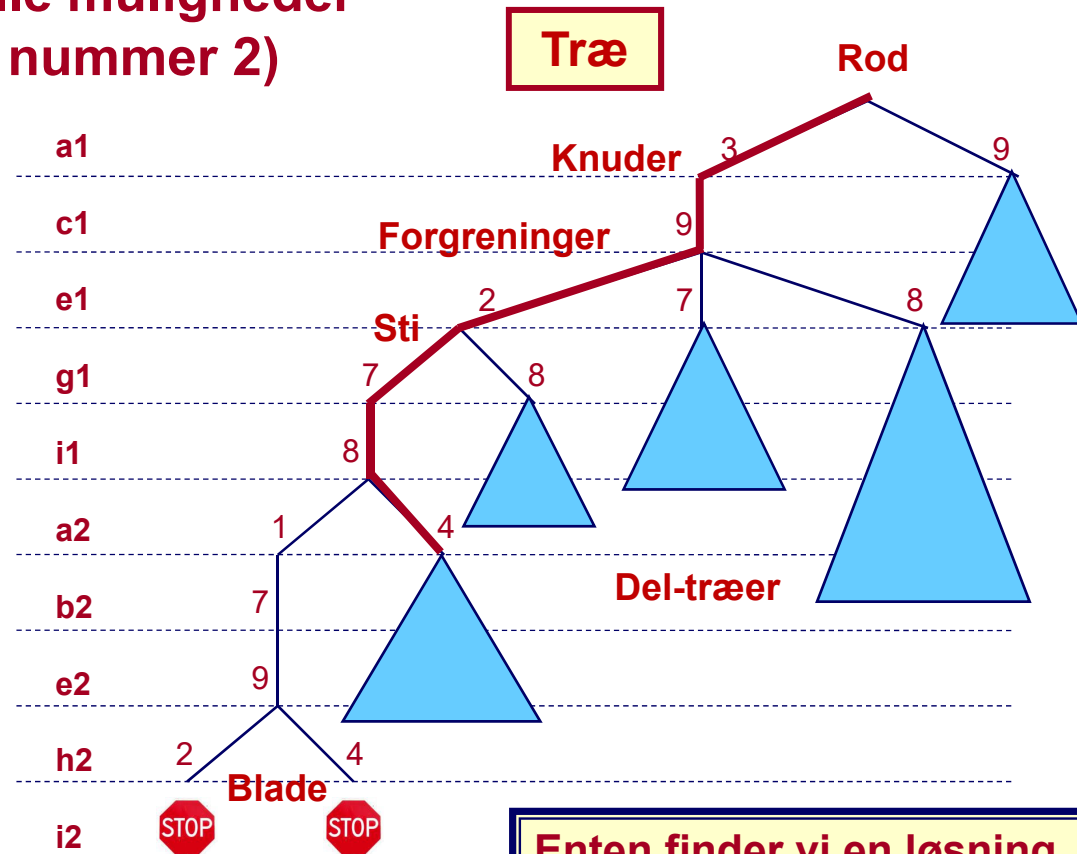
	a	b	c	d	e	f	g	h	i
1	3	6	9	1	2	4	7	5	8
2	1	7	8	3	9	5	6	4	
3	2								1
4	8			4		7			6
5			6				3		
6	7			9		1			4
7	5								2
8			7	2		6	9		
9		4		5		8		7	



# Algoritme til løsning af Sudoku opgaver

- Afprøv systematisk alle muligheder (ved hjælp af strategi nummer 2)

	a	b	c	d	e	f	g	h	i
1	3	6	9	1	2	4	7	5	8
2	4		8	3		5	6		
3	2								1
4	8			4		7			6
5			6				3		
6	7			9		1			4
7	5								2
8			7	2		6	9		
9		4		5		8		7	



Vores "vejvalg" udgør et træ  
Roden er foroven, forgreningerne  
i midten og bladene forneden



Enten finder vi en løsning  
(i en af de blå trekanter) eller  
også har vi vist, at der ikke  
findes en løsning

# Algoritmen – pseudokode – Java-kode

---

```
prøvAlleMuligheder() {                                // tryAll()
    HVIS alle felter er udfyldt {                       // allFilled()
        udskriv løsning                                // printGrid()
    }
    ELLERS {
        husk nuværende felt                            // previous = currentField()
        gå til næste tomme felt                        // advanceToNextField()
        FOR hvert ciffer c {
            HVIS c kan bruges {                         // promising(c)
                indsæt c i felt                         // setFieldValue(c)
                prøvAlleMuligheder()                   // tryAll()
            }
        }
        // backtrack
        fjern sidst indsatte ciffer                    // clearCurrentField()
        gå tilbage til forrige felt                    // setToField()
    }
}
```

## Rekursion

- Vi løser dele af problemet, hvorpå algoritmen kalder sig selv (på et simplere problem)
- Minder om induktion

# Java program – kan udføres af computer

---

```
public void tryAll() {
    if( g.allFilled() ) {
        g.printGrid();
    }
    else {
        // try all values at next field
        Field previous = g.currentField();
        g.advanceToNextField();

        for( int c = 1; c <= 9; c++ ) {
            if( g.promising(c) ) {
                g.setFieldValue(c);
                tryAll();
            }
        }

        // backtrack to previous field
        g.clearCurrentField();
        g.setToField(previous);
    }
}
```

**Demo**

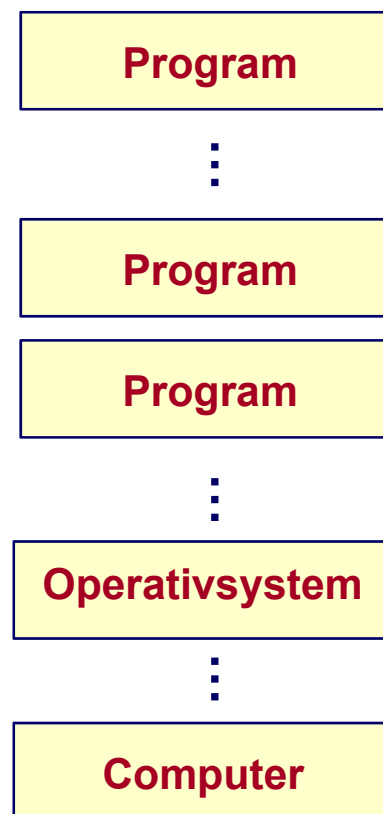
# Computerens styrker

---

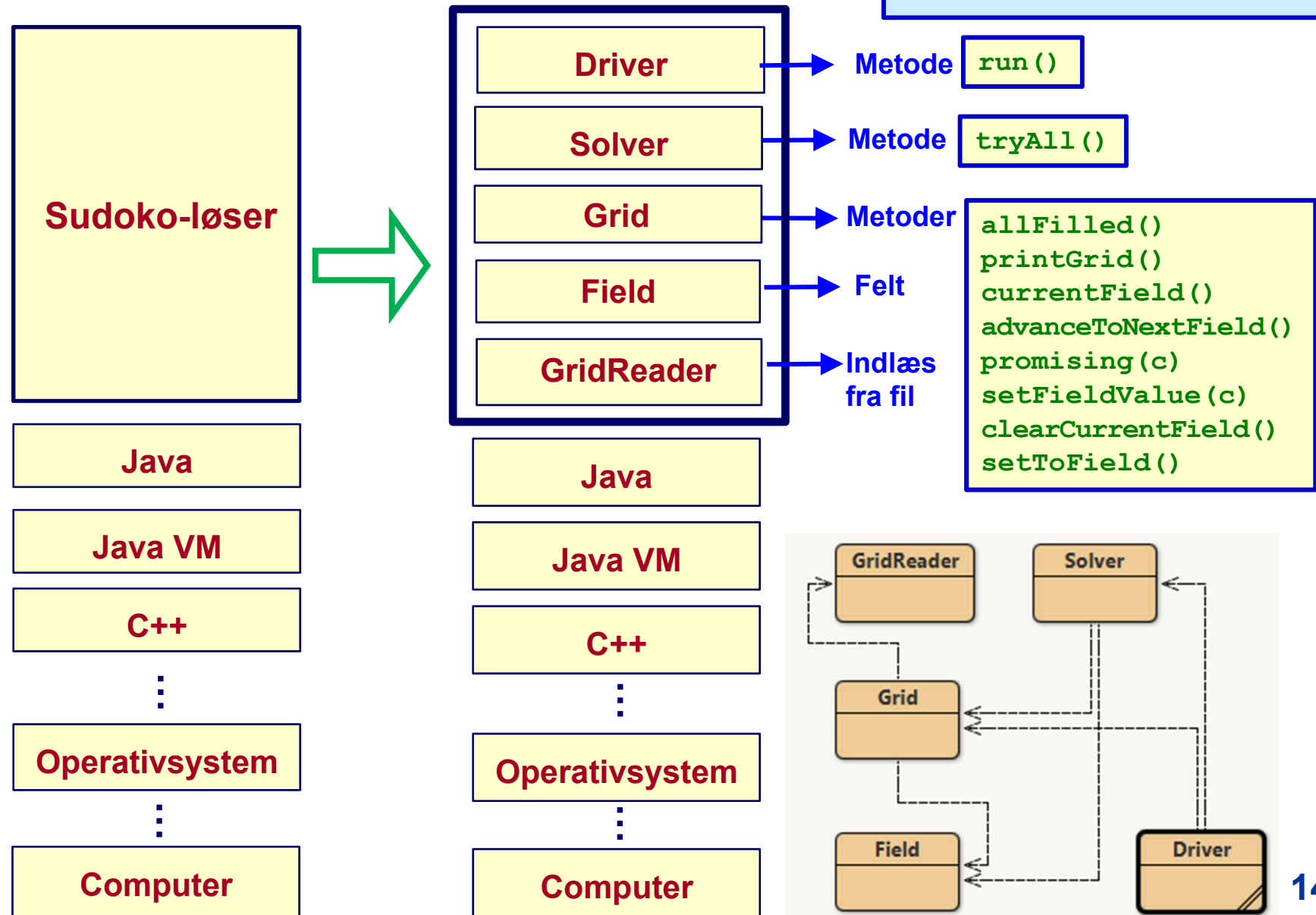
- **Computeren kan**
  - foretage beregninger lynhurtigt
  - lagre store datamængder
  - søge i store datamængder
  - bearbejde store datamængder
  - afsøge et stort antal muligheder og kombinationer
- **Laver ingen fejl**
  - hvis den er programmeret korrekt
- **En computer er en generel maskine, der kan programmeres til at gøre forskellige ting**
  - Computer + Sudoku-program = Sudoku-maskine
  - Computer + Skak-program = Skak-maskine

tekstbehandling, pengeautomat,  
Facebook, Google, Dropbox, iTunes,  
Windows, Linux, OS X, ...

**Lag på lag:**



# En Sudoku-maskine



# ● Agenter og metoder

---

- Hvis min bil går i stykker

- Jeg henvender mig på et autoværksted og forklarer dem hvad problemet er
- Jeg overlader bilen til værkstedet og får den senere tilbage i repareret stand



- Hvad har jeg gjort for at løse mit problem?

- Fundet en passende **agent** eller **serviceudbyder**
- Overbragt agenten en meddelelse om mit problem
- Det er herefter agentens ansvar at løse problemet på mine vegne
- Agenten har en **metode** til at løse problemet, men den behøver jeg ikke at kende til

# Agenter og metoder – blomsterhandel

---

- **Samme princip hvis jeg skal sende blomster til min farmor i Svendborg**
  - Jeg henvender mig til min lokale blomsterhandler med en meddelelse, der indeholder information om, hvilke blomster jeg ønsker, samt min farmors adresse, og så sker resten bag kulisserne uden min indblanding
  - Formodentlig ved at blomsterhandleren videregiver min meddelelse til en blomsterhandler i Svendborg, der sørger for at fremskaffe blomsterne, binde en buket og få dem sendt ud til min farmor





# Delegering til agenter

---

- **Der er forskellige slags agenter**
  - Hver type agent har sine metoder, som er specifikke for netop den service, vedkommende tilbyder
  - Havde jeg henvendt mig på autoværkstedet med mit blomsterproblem, ville de have svaret, at de ikke har nogen metode til at løse det problem
  - Omvendt kan blomsterhandleren ikke reparere biler
- **Løsning af problemet er agentens ansvar**
  - Agenter kan frit anvende en vilkårlig fremgangsmåde (metode) til at løse et problem
  - De skal blot levere en løsning på den type service, de tilbyder
  - Det giver stor fleksibilitet, at vi andre ikke blander os i agents måde at løse problemerne på

# Eksempler på agenter /serviceudbydere

---

- **Webserver**
  - Giver mulighed for at læse websider
- **Mail program**
  - Giver mulighed for at sende, modtage og opbevare mails
- **Messenger**
  - Giver mulighed for at sende, modtage og opbevare korte beskeder
- **Dropbox**
  - Giver mulighed for at opbevare og tilgå filer
- **Facebook**
  - Giver mulighed for at kommunikere med sine venner
- **Agenter /servere gør normalt ikke noget af sig selv**
  - De venter på, at brugerne beder dem om at gøre noget, og udfører så det, som de er blevet dem om
  - De kan dog også selv igangsætte handlingssekvenser (Facebook!)

# ● UML: Et grafisk specifikationssprog

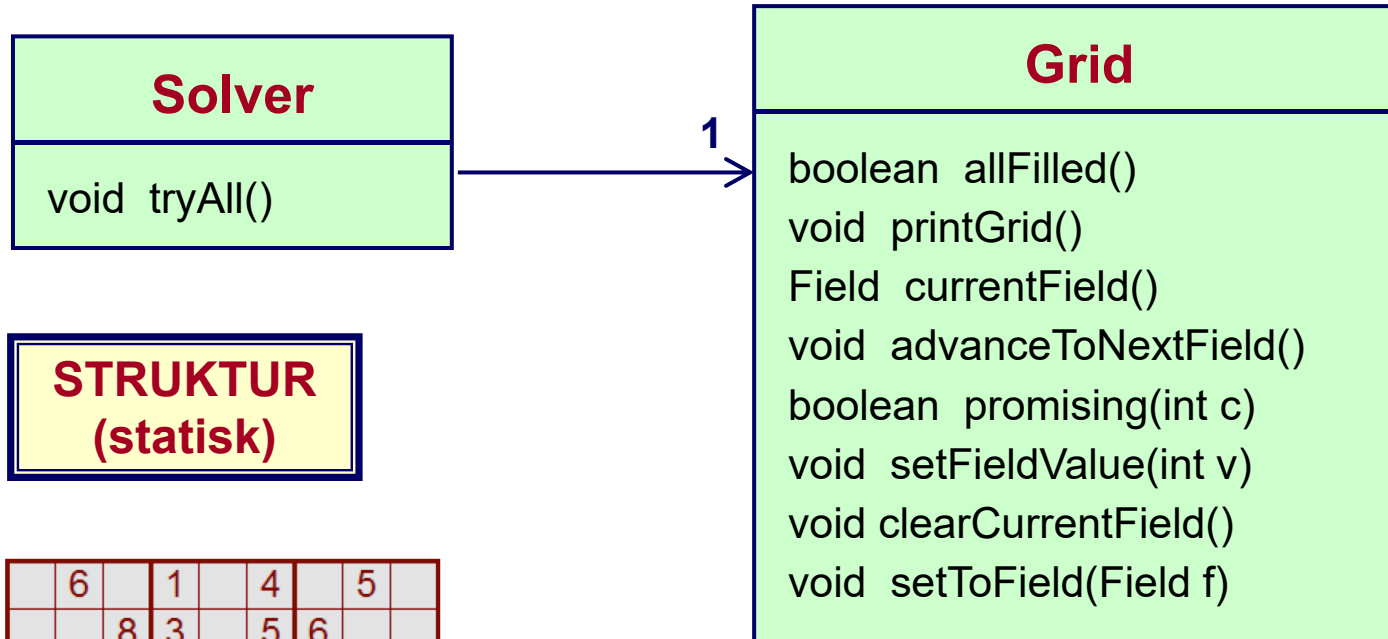
---



Industriel standard for  
specifikation af programmer

- **Mange forskellige diagramtyper**
  - **Klassediagrammer**
  - **Sekvensdiagrammer**
  - **Objektdiagrammer** (introduceres i en senere forelæsning)
  - ...

# Klassediagram for Sudoku løseren (uddrag)

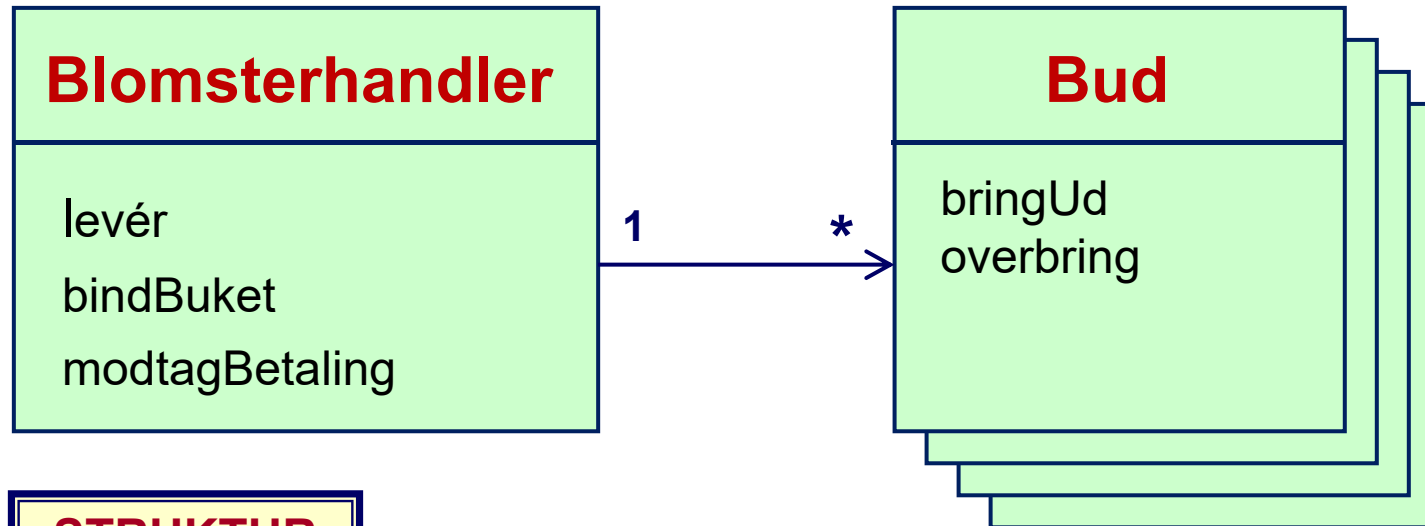


**STRUKTUR**  
(statisk)

	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

- Hver af de grønne kasser udgør en programdel (klasse)
- Pilen angiver, at Solver'en bruger faciliteter, som Grid'en stiller til rådighed
- 1-tallet angiver, at Solver'en anvender præcis én instans (udgave) af Grid'en
- Det totale klassediagram består af 5 grønne kasser (Driver, Solver, Grid, Field og GridReader)

# Klassediagram for blomsterhandel

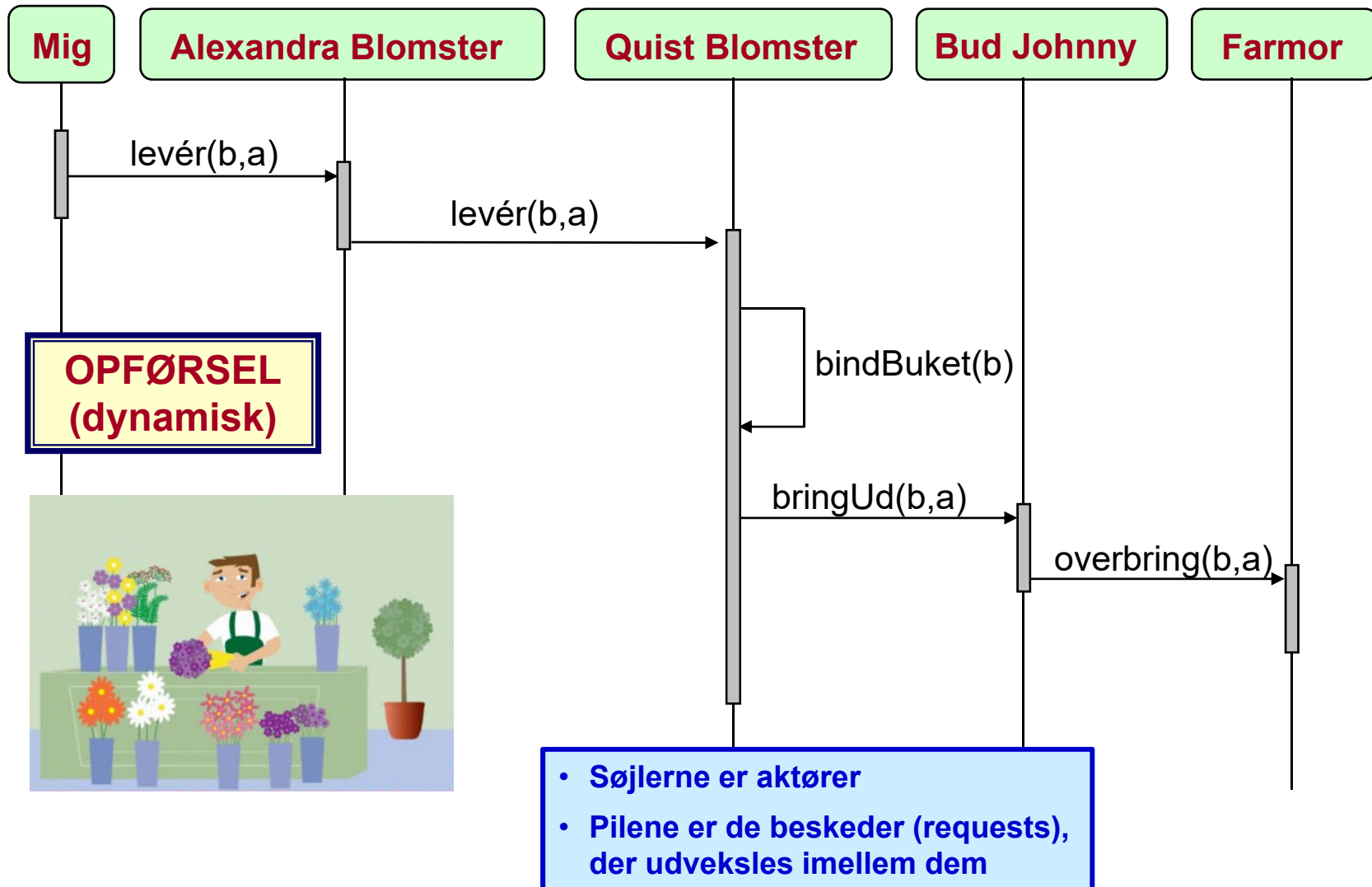


**STRUKTUR**  
(statisk)

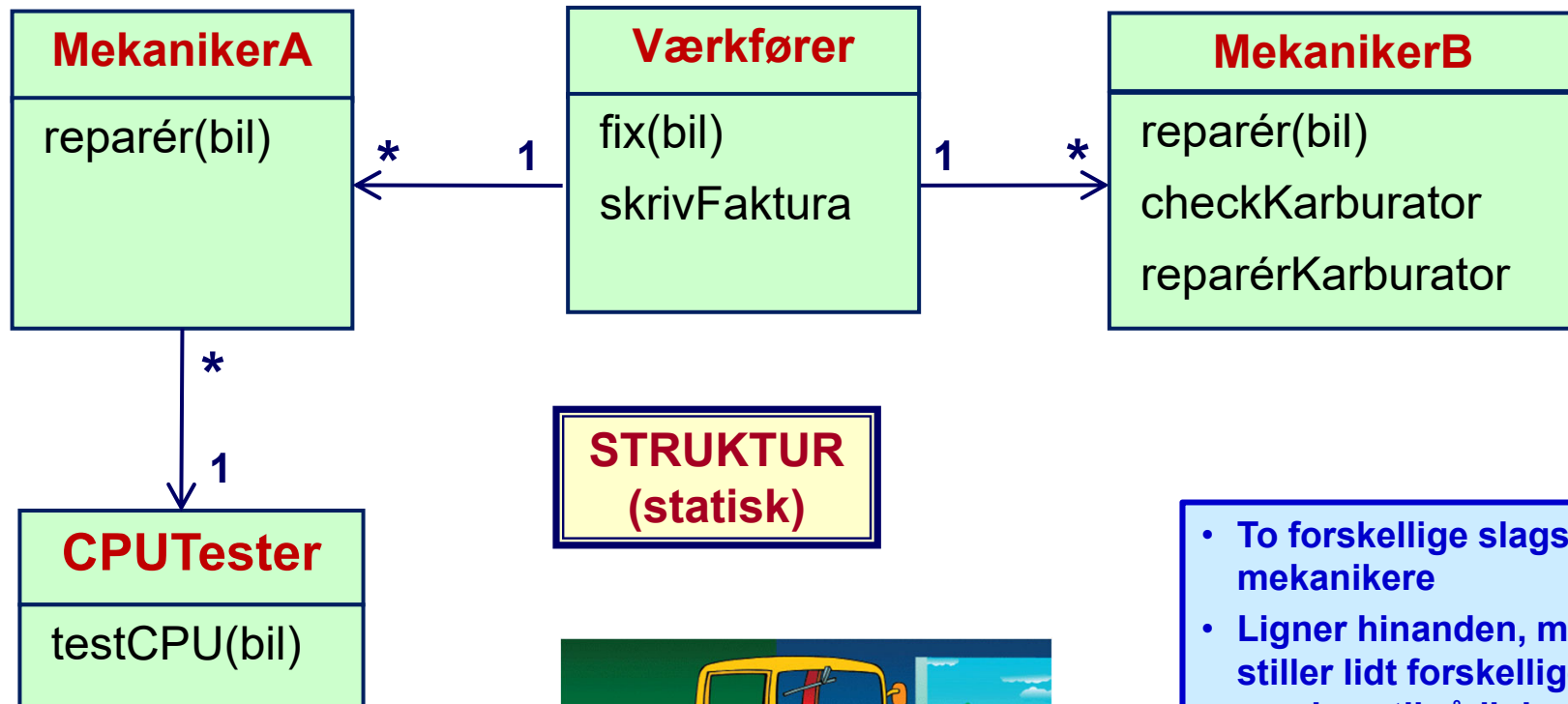


- 1-tallet angiver, at hvert Bud er tilknyttet præcis én Blomsterhandler
- Stjernen angiver, at Blomsterhandleren kan have flere Bude tilknyttet

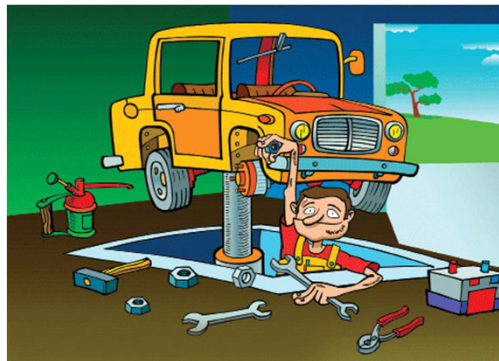
# Sekvensdiagram for blomsterhandel



# Klassediagram for autoværksted

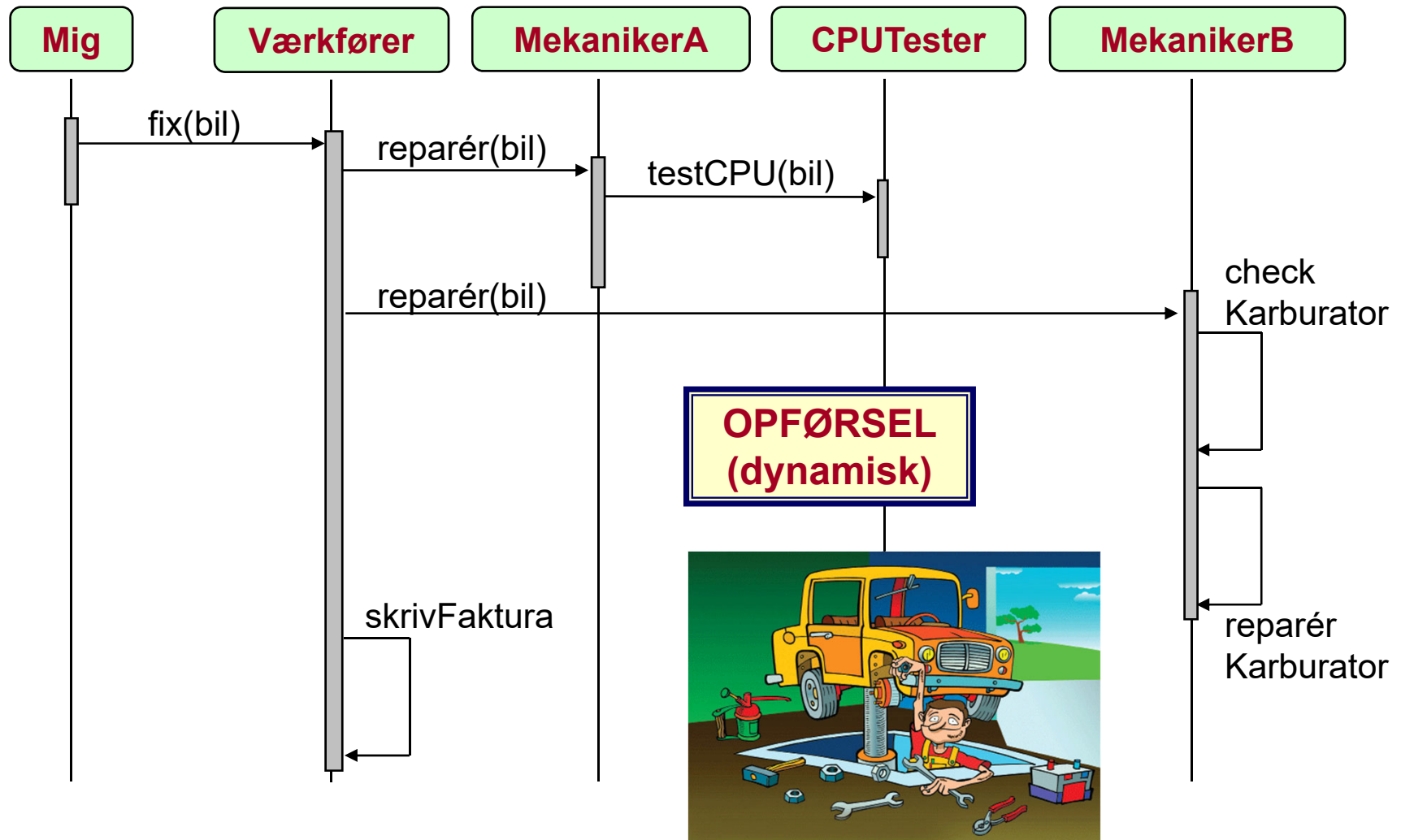


**STRUKTUR**  
(statisk)



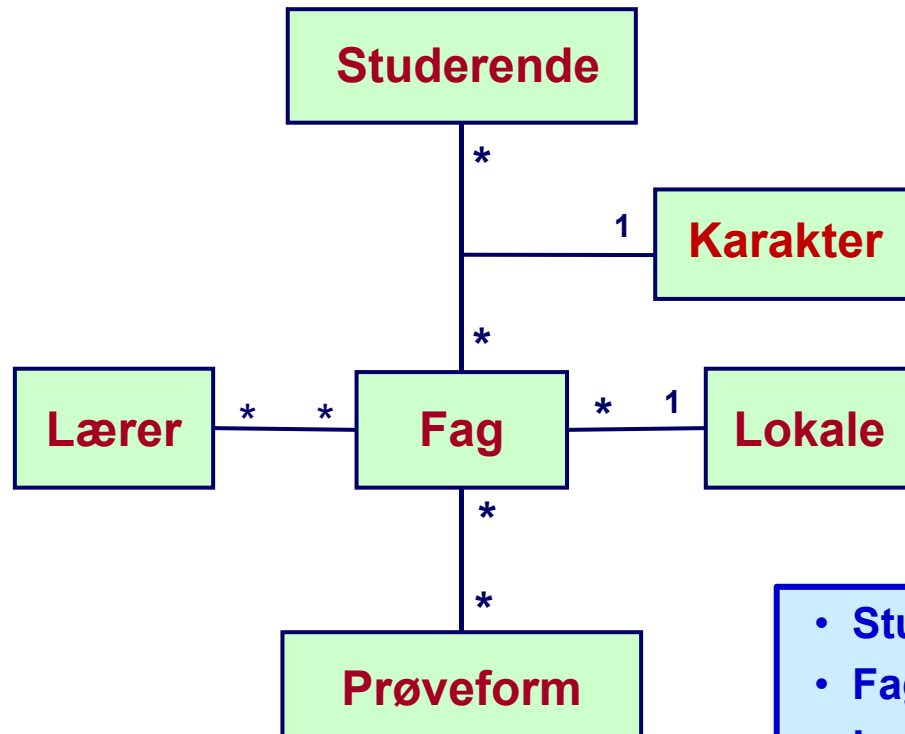
- To forskellige slags mekanikere
- Ligner hinanden, men stiller lidt forskellige services til rådighed

# Sekvensdiagram for autoværksted

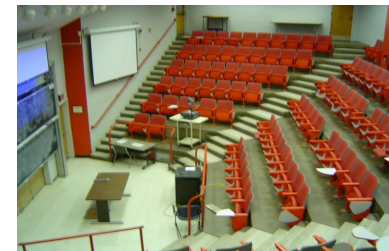




# Klassediagram for studieadministration



**STRUKTUR**  
(statisk)



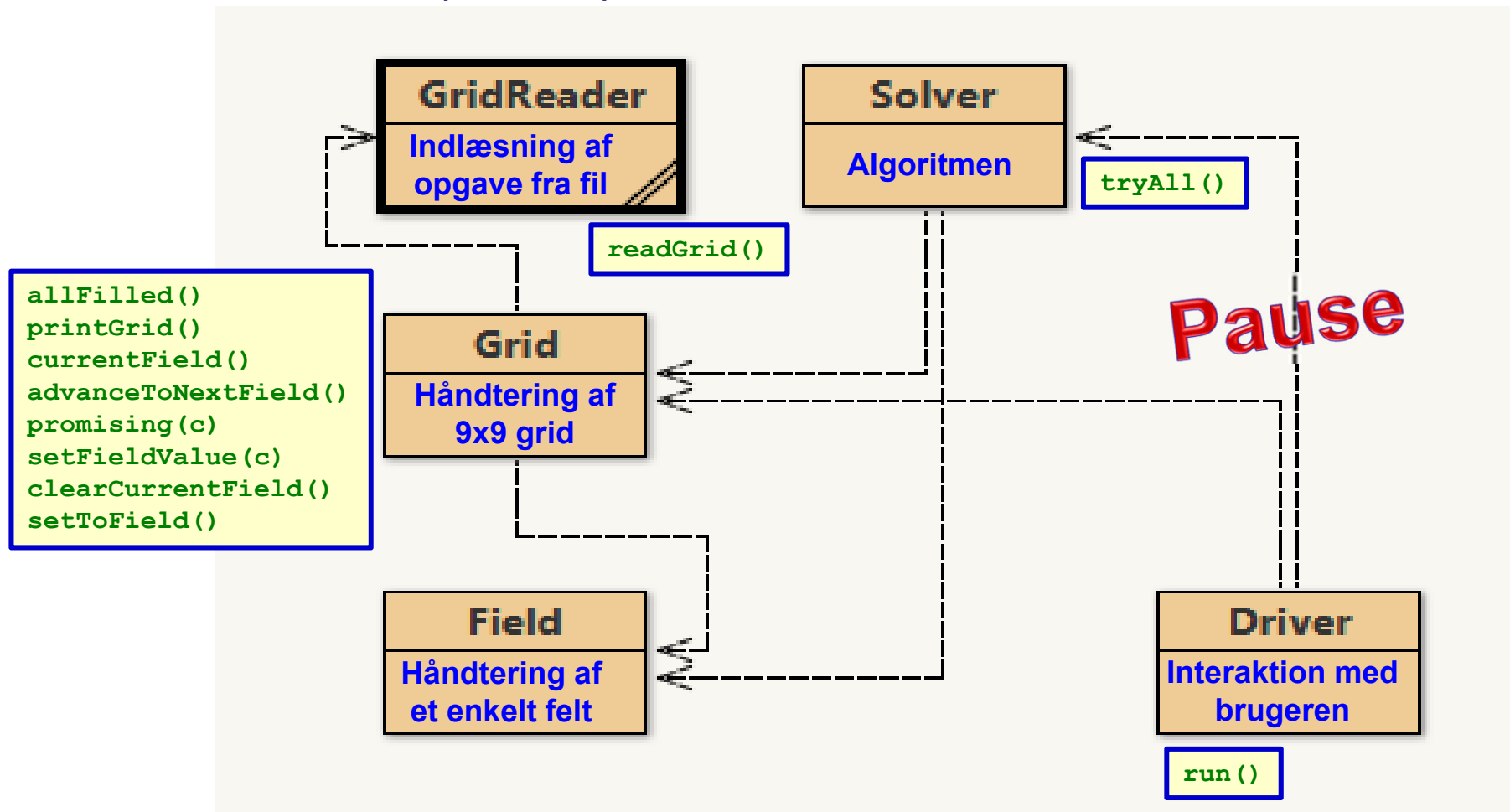
- **Studerende** (Rasmus, Stine, Søren, ...)
- **Fag** (Programmering, Calculus, ...)
- **Lærer** (Kurt Jensen, .....)
- **Lokale** (Aud. E, Aud. F)
- **Prøveform** (mundtlig, skriftlig, projekt, ...)
- **Karakter** (bestået, udeblevet, 7, ...)

↑  
Klasser  
(begreber)

↑  
Objekter  
(instanser af begreber)

# Klassediagram for Sudoku løseren

- **Kopieret fra BlueJ**
  - Fem forskellige klasser med hvert deres formål
  - I BlueJ er pilene stiplede



# ● Information om kurset

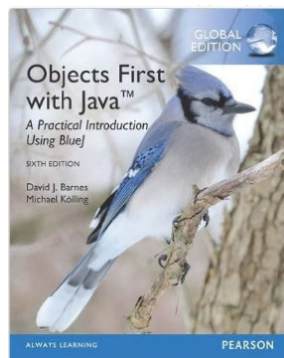
- **Objektorienteret programmering**

- Java er vores programmeringssprog
- BlueJ er vores programmeringsomgivelser (editor)
- Undervejs bruger vi kode produceret af andre (Javas klassebibliotek)

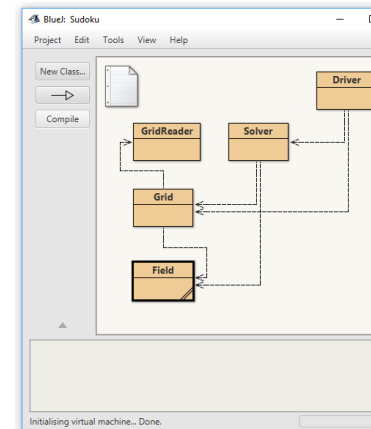


- **Modeldrevet programmering**

- Programmeringsopgaver tager udgangspunkt i simple objektorienterede modeller (primært klassediagrammer)
- UML diagrammerne er vores specifikationssprog
- Java er vores implementationssprog



**Blå skovskade**  
**Blue Jay**



```
1 /**
2  * @author Kurt Jensen
3  * @version 2017-01-12
4  */
5 public class Solver {
6
7
8
9     private Grid g;
10
11     public Solver(Grid g) {
12         this.g = g;
13     }
14
15     public void setGrid(Grid g) {
16         this.g = g;
17     }
18
19     /**
20      * Find and print the solution of this Sudoku
21      */
22     public void tryAll() {
23         if (g.allFilled()) {
24             g.printGrid();
25         }
26         else { // try all promising values at next field
27             Field previous = g.currentField();
28             g.advanceToNextUnfilled();
29             for (int c = 1; c <= 9; c++) {
30                 if (g.promising(c)) {
31                     g.setFieldValue(c);
32                     tryAll();
33                 }
34             }
35             backtrackTo(previous);
36         }
37     }
38
39     private void backtrackTo(Field f) {
40         g.clearCurrentField();
41         g.setToField(f);
42     }
43 }
44
```

# Programmering

---

- **Simpel programmering til husbehov**
  - I vil lære nogle grundliggende ting omkring programmering
  - Efter kurset vil I kunne lave simple programmer og forstå de vigtigste principper bag programmering
  - Men I bliver ikke verdensmestre i at programmere på 15 uger
  - Det kræver masser af træning – gennem flere år
- **Programmering kræver masser af praktisk øvelse**
  - I lærer ikke at programmere ved at læse bøger eller se videoer
  - I lærer det ved at **øve jer igen og igen**
  - Der er masser af basale ting, som skal sidde på rygmarven, og som I skal kunne gøre i søvne
  - Kan sammenlignes med **guitar / fodbold** – det bliver man ikke god til ved at læse om eller se på tv – man skal selv træne og træne
  - Derfor har dette kursus – som en studerende skrev i en evaluering – en **"latterlig mængde"** obligatoriske programmeringsopgaver

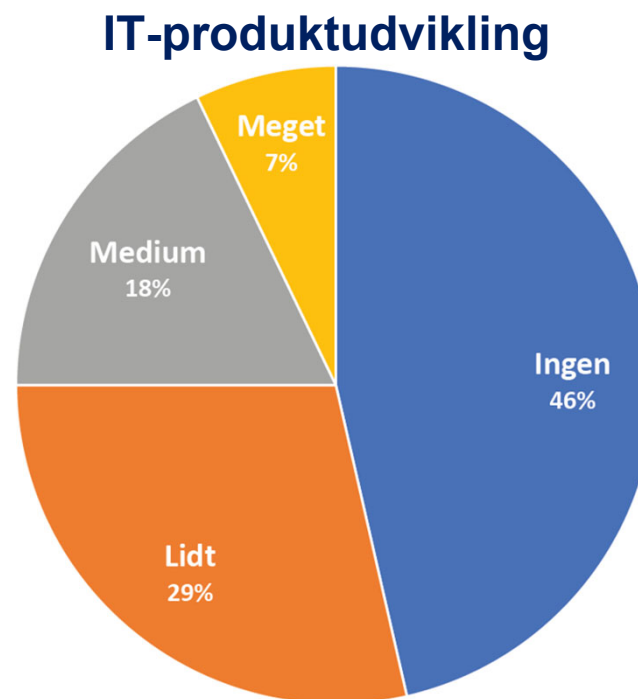
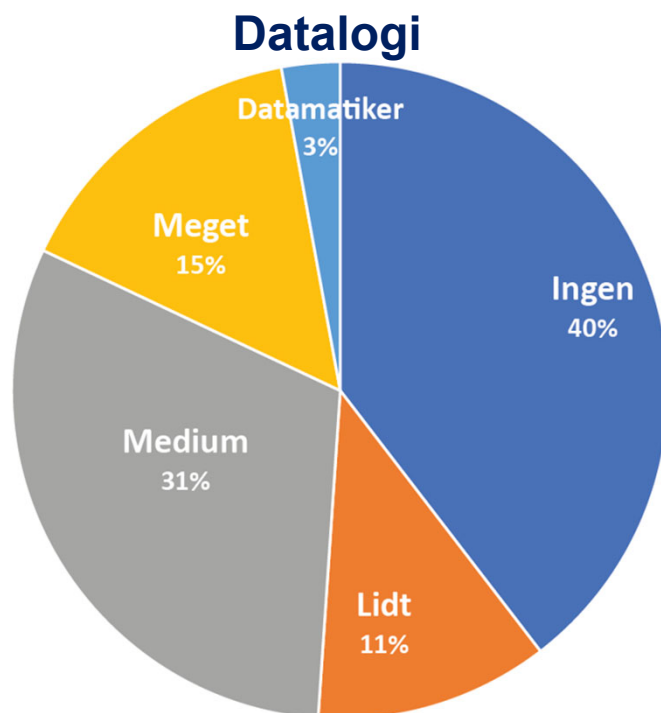
# Læringsmål

---

- Efter kurset vil I have kendskab til principper og teknikker for systematisk konstruktion af programmer, og I vil kunne
  - **anvende** et almindeligt programmeringssprog
  - **udvikle** velstrukturerede programmer og **afteste/debugge** disse
  - **forklare** arkitekturen af programmer, herunder nedarvning, abstrakte klasser og interfaces
  - **forklare** simple specifikationsmodeller og **realisere** disse i programmer
  - **anvende** standardklasser ved realisering af programmer

# Programmeringserfaring

- **Stor spredning med hensyn til programmeringserfaring**
  - For datalogi er det godt halvdelen, der har ingen eller lille erfaring
  - For it-produktudvikling er det tre fjerdedele



- **Det betyder, at nogle af jer vil synes, at det går langsomt her i starten**
  - Det er nødvendigt af hensyn til dem, der har ingen eller lille programmeringserfaring (mere end halvdelen af jer)

# Eksamen

---

- **15 minutters mundtlig prøve med ca. 15 minutters forberedelse**
  - 9 spørgsmål, der dækker kursets centrale emneområder
  - Eksaminanden forventes at demonstrere
    - Kendskab til de vigtigste begreber indenfor det trukne emneområde
    - Evne til at programmere i Java ved at præsentere små velvalgte programstumper indenfor emneområdet
    - Evne til at svare på simple spørgsmål inden for emneområdet, herunder relatere kursets afleveringsopgaver til emneområdet
- **I slutningen af uge 7 er der en køreprøve**
  - Praktisk prøve i programmering af 30 minutters varighed
- **I kursets anden halvdel er der et gennemgående projekt**
  - I skal konstruere et simpelt computerspil
  - Delaflevering hver uge (hvor I benytter de ting, der er gennemgået ugen før)
- **Køreprøve og projekt tæller med ved fastlæggelsen af endelig karakter**
  - Høje point kan trække en karakter op, mens lave point kan trække en karakter ned

# Aktiviteter på kurset

---

- **Forelæsninger**

- Giver overblik over begreber, principper og gennemgår eksempler
- Indeholder små quizzer, hvor I deltager aktivt
- Optages på video (forudsat at teknikken virker) og er således tilgængelige, hvis der er ting man vil have genopfrisket

- **Øvelser**

- Praktisk arbejde under vejledning af instruktør (ældre studerende)
  - Man arbejder primært med de obligatoriske afleveringsopgaver
  - Også mulighed for at stille spørgsmål til lærebog og videonoter

- **Hjemmearbejde**

- Læse kapitlerne i lærebogen
  - Herunder løse de **50-100 småopgaver**, der er i hvert kapitel
  - Det er **vigtigt**, at I løser opgaverne – I lærer kun at programmere ved at øve jer, og de fleste af opgaverne er små programmeringsopgaver
- Gennemse **videoerne** (ca. 75 i alt – af 5-10 minutters varighed)
  - Præsenterer vigtigt stof – integreret del af kurset
  - Næsten alle videoer er eksempler på "live programmering"
  - I kan stoppe (for at tænke jer om) eller gentage afsnit (som er vanskelige)



# Forelæsninger

---

- **Sprog**

- Bachelorkurser på det Naturvidenskabelige Fakultet (Natural Sciences) undervises på dansk (med mindre forelæseren ikke er dansktalende)
- Derfor vil jeg tale dansk, og mine slides vil være på dansk
- Mange fagudtryk og mange navne fra programmerne er på engelsk
- Sproget bliver derfor en (lidt uskøn) blanding af dansk og engelsk
- Det bliver I nødt til at leve med – det er typisk for vores fag

- **Forberedelse til forelæsningerne**

- Hovedformålet med forelæsningerne er at give jer et overblik over begreber og principper samt gennemgå udvalgte eksempler
- For de fleste af jer vil det herefter være lettere at læse lærebogen
- Nogle synes, at det er en fordel at læse i bogen før forelæsningerne
- Andre synes, at det er nemmere selv at gå i gang med lærebogen – uden at gå til forelæsningerne (eller nøjes med at se dem på video)
- Gør det, der fungerer bedst for jer (men ikke det der er nemmest)

- **Mine slides indeholder mange ting, som ikke er med i lærebogen**

**Obs!**

- Det er ting som bruges i opgaverne og er del af eksamenspensummet
- Som et minimum skal I derfor gennemgå forelæsningsslidsene

# Afleveringsopgaver

---

- **Programmering kræver masser af træning**
  - Derfor har kurset
    - 13 afleveringsopgaver og 5 quizzer i første halvdel
    - 7 afleveringsopgaver i anden halvdel
  - De to ugentlige øvelsesgange bruges primært til at arbejde med disse opgaver
  - De fleste af opgaverne før efterårsferien er forholdsvis små og kan løse på 30-60 minutter (under øvelserne)
  - Alle afleveringsopgaver er obligatoriske og skal godkendes af jeres instruktør for at I kan gå til køreprøven og den mundtlige eksamen
- **I begyndelsen vil instruktorerne ofte kræve genaflevering af opgaver med forholdsvis små fejl**
  - På den måde får vi hurtigere udryddet de værste unoder i jeres programmeringsstil
  - Genaflevering skal ske senest 1 uge efter den oprindelige afleveringsfrist
  - I kan normalt kun genaflevere fire gange i løbet af kursets første halvdel, så gør jer umage med at lave de enkelte afleveringer så gode som muligt

# Afleveringsopgaver (fortsat)

---

- **Deadline for alle afleveringsopgaver er mandag kl 12.00**
- **Pas på med, at I ikke kommer bagefter**
  - Det kan være meget svært at indhente igen
- **Sygdom og lignende**
  - Hvis I bliver syg i længere tid (eller af andre grunde ikke kan passe jeres studier), bør I **hurtigst muligt kontakte mig**, så vi kan lave en plan for, hvordan I får indhentet det forsømte
  - Det kan f.eks. ske i løbet af efterårsferien, hvis I har mulighed for det
- **Tilsvarende gælder selvfølgelig for de andre kurser, som I følger**
  - Der bør I også kontakte jeres forelæsere, hvis I af en eller anden grund kommer bagud

# Par-programmering

---

- **Ved øvelserne arbejdes i par (på 2 personer)**
  - Gælder også afleveringsopgaverne (bortset fra uge 5-6)
  - I må også gerne lave hjemmearbejde og forberedelse i par (eller i jeres læsegrupper)
- **Hvorfor skal I arbejde i par?**
  - Ved at arbejde i par hjælper I hinanden, så I ikke så let går i stå på grund af småproblemer
  - Det træner jer i at kunne arbejde sammen med andre, hvilket er en vigtig kompetence for programmører
  - Derudover er det en praktisk foranstaltning, således at instruktorerne kan nå at komme rundt på hele holdet – idet de så kun skal se og kommentere 12 besvarelser i stedet for 24
- **Par = 2 personer**
  - 1-mandsgrupper tillades dog, hvis der er særlige forhold (eller et ulige antal deltagere på øvelsesholdet)
  - 3-mandsgrupper tillades **aldrig** (så får man for lidt træning)

# Quizzer

---

- **I løbet af kursets første fem uger skal I løse fem små quizzer**
  - Quizzerne afprøver, om I har forstået de begreber, som introduceres ved forelæsningsne og i lærebogen
  - Hver quiz består af 12-16 spørgsmål og kan klares på 20-30 minutter
  - Quizzerne er **interaktive**. Så snart I har svaret på et spørgsmål, får I at vide om svaret er rigtigt eller forkert – og i sidstnævnte tilfælde ofte et vink til, hvad der skal rettes, for at svaret bliver rigtigt
  - Quizzerne løses **individuel** og er **obligatoriske afleveringsopgaver**, som skal afleveres inden den sædvanlige afleveringsfrist
- **Lav quizzerne sidst på ugen**
  - De bruger ofte stof, der bliver introduceret i ugens forelæsninger eller i de kapitler, som I skal læse

# Diskussionsforum

---

- **Kursets Brightspace side indeholder et diskussionsforum, der giver jer mulighed for at stille spørgsmål til forelæser og instruktører**
  - Det er den bedste og hurtigste måde at få hjælp på – når I ikke er til øvelser
  - Svaret kommer ofte inden for få timer/minutter (selv uden for normal arbejdstid)
- **For at få mest muligt ud af diskussionsforummet, er det vigtigt, at I er omhyggelige med at skrive jeres indlæg**
  - Giv jeres indlæg en velvalgt titel, som i få ord beskriver, hvad det drejer sig om – brug opgavenumre og tilsvarende "officielle" benævnelser, når I refererer til ting i kurset, f.eks. "BlueJ bogens opgave 4.12"
- **Når der svares på et indlæg, dannes der en tråd under det oprindelige indlæg**
  - Undervejs i diskussion kan man få lyst til at tage et andet emne op
  - I den situation bør man starte en ny tråd, fremfor at fortsætte i den gamle
  - På den måde bliver det lettere at finde relevant information på forummet

# Diskussionsforum (fortsat)

---

- **Jeres indlæg må ikke indeholde løsninger på hele opgaver**
  - Det duer ikke at sende 1-2 sider kode og spørge: "Er der nogen der kan se, hvorfor mit program ikke virker?"
  - I stedet skal I isolere problemet, hvilket er let, hvis I løser opgaverne I små skridt – således som vi anbefaler
  - I kan så nøjes med at kopiere nogle få kodelinjer og spørge, hvad der er galt i dem
  - Det gør det nemmere for os at svare på jeres spørgsmål – hvorfor svaret ofte kommer hurtigere
  - Husk at beskrive, hvad problemet er (oversætterfejl, runtime fejl, uventet resultat)
- **Hold jer endelig ikke tilbage med hensyn til at bruge diskussionsforummet**
  - Hvis der er noget, som I ikke kan finde ud af, er der sikkert en del andre i samme situation. De vil så få nytte af jeres spørgsmål og svaret herpå
  - I må også meget gerne selv svare på spørgsmål, som andre studerende sender til diskussionsforummet
  - Man kan poste anonymt, men det betyder blot at andre studerende ikke kan se, hvem I er, mens underviserne stadig kan

# Studiecafé

---

- **Stueetagen af Vannevar Bush bygningen  
(bygning 5343 i IT-Parken, Åbogade 34, ved Storcenter Nord)**
  - Lokalerne kan benyttes 24/7.
  - Uden for normal åbningstid kræver det dog, at man har anskaffet et adgangskort, så man kan komme ind
  - <http://studerende.au.dk/studier/fagportaler/datalogi/studiemiljoe/cs-studiecafe/> [Link](#)
- **Brug studiecaféen**
  - God måde at få struktureret jeres arbejdsdag på
  - Når I arbejder hjemme, bliver I let forstyrret af andre gøremål
- **På følgende tidspunkter, er der en instruktør fra kurset til stede**
  - Mandag 9-10, Tirsdag 8-10, Onsdag 10-12, Torsdag 10-12, Fredag 10-12
  - Kom tidligt. Instruktoren går, når der ikke er flere, der ønsker hjælp (så hvis du kommer i sidste øjeblik, risikerer du, at instruktoren er gået)
  - Bemandingen starter onsdag den 31.8 og fortsætter indtil kursets afslutning den 12. december



# Programmeringscafé

---

- **Tilbud til studerende, som ikke tidligere har programmeret (eller kun har programmeret en lille smule)**
  - 2-3 timer om ugen
  - Det er frivilligt, om man ønsker at deltage
- **Ledes af Magnus Madsen (tenure-track adjunkt)**
  - En time hvor man programmerer i fælleskab
    - Magnus programmerer på projektoren og de studerende skriver med på egen PC
    - Diskussion af problemstillinger undervejs
    - Spørgsmål til/fra de studerende
  - En time hvor hver studerende arbejder videre på programmet, mens Magnus går rundt og hjælper
- **De der deltog sidste år siger, at det var en særdeles stor hjælp for dem**
  - Caféen gjorde det meget letter at komme i gang med afleveringsopgaverne



# Programmeringscafé (fortsat)

---

- **Ingen forberedelse – tager kun den tid som caféen varer**
  - Intet nyt materiale – man får alt materiale via de almindelige forelæsninger, ved at læse bogen, se videoerne og deltage i øvelserne
- **Programmeringscaféen er et supplement, som forklarer de vigtigste principper i et langsommere tempo og med flere eksempler**
  - Hvis man har let ved at programmere og/eller man er super ambitiøs, så er programmeringscaféen ikke det rigtige sted at komme
  - I stedet kan man overveje at deltage i instituttets præ-talentforløb, som beskrives ved en senere forelæsning (se også [cs.au.dk/talent](https://cs.au.dk/talent))
- **Tid og sted for programmeringscaféen**
  - Mandag kl. 15.15-18.00 eller onsdag kl 15.15-18.00 (man deltager kun én af gangene)
  - Finder sted i Aabogade (i nærheden af Storcenter Nord)
- **Tilmelding er i princippet slut, men hvis du er interesseret kan du skynde dig at kontakte Andreas Birch Olsen <[abolsen@cs.au.dk](mailto:abolsen@cs.au.dk)>**



## brightspace.au.dk

Vigtige meddelelser +  
Diskussionsforum

Ugeoversigt for uge 1-7

Uge 1 (30. august - 5.  
september 2021)

Uge 2 (6. september - 12.  
september 2021)

Uge 3 (13. september -  
19. september 2021)

Uge 4 (20. september -  
26. september)

Uge 5 (27. september - 3.  
oktober)

Uge 6 (4. oktober - 10.

### Uge 1 (30. august - 5. september 2021)

Ugebrev 1 - Forum for uge 1-4

#### Forelæsninger

##### Mandag

Hvad er programmering?  
Objekt-orienteret modellering  
Information om kurset  
Slides

##### Torsdag

Objekters tilstand og opførsel  
Skabelse af objekter  
Iteration, selektion og  
parametrisering  
Slides

#### Øvelser

##### Første

Sammensætning af par  
BlueJ og AU e-mails  
Opgaverne fra Kap. 1 i BlueJ  
bogen

##### Anden

Raflebæger 1

##### Aflevering

Raflebæger 1 plus Quiz 1

#### Materialer

##### Kapitler

1. Objects and Classes (17 p)  
2.1-2.11. Class Definitions (22 p)  
Appendix A (bruges til opslag)

*Husk at løse opgaverne i BlueJ-  
bogen, mens I læser kapitlerne*

##### Videoer

1.1, 1.2, 1.3, 1.4, 2.1, 2.2  
BlueJ genveje

- I må meget gerne sætte et foto på jeres Brightspace profil
- Så lærer instruktorerne jer hurtigere at kende

# Brightspace

---

- **Pulse**

- Brightspace har en tilknyttet app, der hedder Pulse
- Den fungerer meget dårligt i forhold til komplekse websider, som vi har på dette kursus (formateringen forsvinder og det samme gør links)
- Jeg anbefaler derfor at tilgå kursets websider via en almindelig webbrowser på en bærbar/stationær computer (såsom Google Chrome, Safari, Microsoft Edge og Firefox)

# Undervisningsprincipper

---

- **I møder de samme begreber og teknikker mange gange gennem kurset (spiral-metoden)**
  - Introduktion ved forelæsning
  - Selvstudie via videonote og/eller bogkapitel
  - Praktisk træning ved en eller flere øvelsesgange
  - Repetition i senere forelæsning
  - Mere praktisk træning – osv.
- **Vær med fra start**
  - De første 3-4 uger kan være overvældende og svære – specielt, hvis man ikke har forudgående programmeringserfaring
  - Men hold ud og klø på – kommer I bagud i denne fase, er det vanskeligt at indhente
  - Der kommer ikke et tidspunkt, hvor vi skifter til noget ”helt andet”
  - Hvis I ikke forstår de ting vi arbejder med i uge 1-2, kan I heller ikke forstå det i uge 3-4

# Når I ikke kan få jeres kode til at virke

---

- **Ved øvelserne**
  1. Spørg dig selv
  2. Spørg din makker
  3. Spørg et andet par
  4. Kig i slides (og Java API'en)
  5. Spørg jeres instruktør
- **Uden for øvelserne kan I bruge diskussionsforummet**
  - I får som regel hurtigt svar
  - Svaret kan hjælpe mange andre
  - Man kan spørge (og svare) anonymt
  - I kan også gå i studiecaféen, hvor der ofte vil være andre studerende, som I kan spørge
- **Ved forelæsningerne**
  - Jeg kigger **ikke** på jeres detaljerede kode i pauserne
  - Det kan jeg simpelthen ikke nå
  - Men jeg svarer meget gerne på (næsten) alle andre spørgsmål

# Plagiering

---

- **Enhver form for plagiering er uacceptabelt og sidestilles med eksamenssnyd, som er en alvorlig forseelse**
  - Det er forbudt at kopiere andre studerendes afleveringsopgaver, og det samme er tilfældet for opgaver, som man finder på nettet eller andet steds
  - Det gælder både hele opgaver og dele af opgaver
  - Selv kopiering af små programdele (f.eks. en metode) opfattes som plagiering
  - Det er også plagiering at lade andre aflevere kopi af ens egen opgave
  - Man må ikke gøre sine opgavebesvarelser tilgængelige for personer uden for parret/læsegruppen (f.eks. via websider, github og lignende)
- **Kurset har nul-tolerance over for plagiering**
  - Studerende, der bliver grebet i plagiering, får ikke godkendt deres obligatoriske opgaver, og kan derfor **først komme til eksamen det efterfølgende år**
  - Det betyder, at man ikke består 1. års prøven og dermed må **forlade studiet**
- **Vi anvender en række automatiske tests til afsløring af plagiering**
  - Der testes både i forhold til opgaver fra tidligere år og i forhold til andre opgaver, der afleveres i år
  - **Lad være med at tage chancen** – vi opdager snyd hvert eneste år og konsekvenserne for de involverede er **voldsomme**

# Programmering er svært

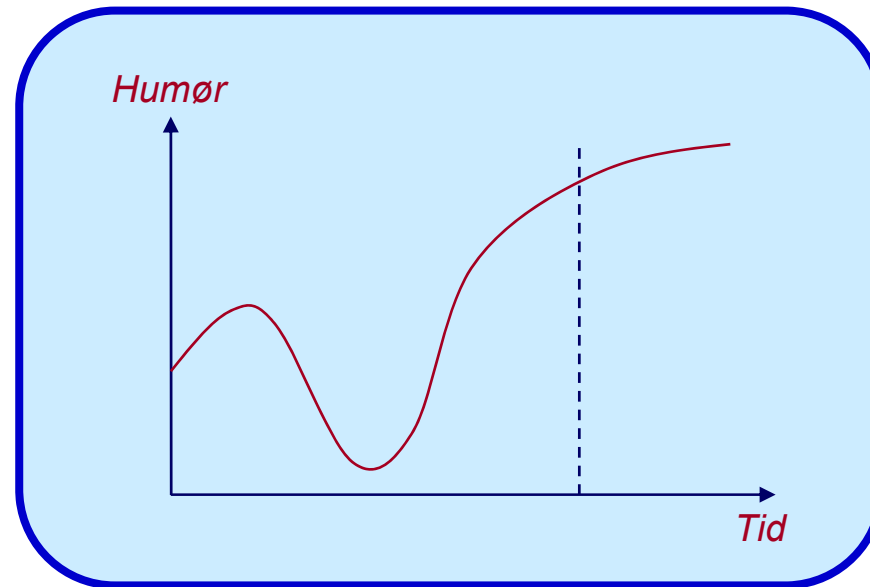
---

- **Programmering**

- Anderledes
- Ny tankegang

- **Faser**

- Motivation
- Begejstring
- **Tvivl?**
- Frustration
- Eksistentiel krise
- **Heureka!**
- Fascination
- Indsigt
- **Magt over teknologien**



- **Advarsel**

- Programmering er **sjovt** og **stærkt vanedannende**
- Når man først kommer godt i gang, kan det være svært at stoppe igen



# Øvelserne i de to første uger

---

- **Første øvelsesgang i uge 1**
  - Hjælp til installation af BlueJ inklusiv Java
  - Opgaverne fra Kapitel 1 i BlueJ bogen
  - I bør på forhånd kigge på så mange af disse opgaver som muligt
- **Anden øvelsesgang i uge 1**
  - Afleveringsopgave om Raflebæger (den ser vi på om et øjeblik)
- **Første øvelsesgang i uge 2**
  - Afleveringsopgave om studieteknik: Studievaner
  - Opgaverne fra Kapitel 2 og 3 i BlueJ bogen – Husk at forberede jer på dem
- **Anden øvelsesgang i uge 2**
  - Ny afleveringsopgave om Raflebæger, hvor terningerne nu kan have et vilkårligt antal sider
- **Efter de første to uger forventer vi, at I selv løser de 50-100 småopgaver, der er i hvert BlueJ kapitel – mens I læser kapitlet**
  - Nogle opgaver tjekker begreber, mens de fleste er små programmeringsopgaver
  - Det er **vigtigt** at I øver jer på disse – I lærer kun at programmere ved at øve jer
  - I skal også huske at gennemse de **videoer**, der hører til kapitlet

# Praktiske ting

---

- **Mails**

- Det er **VIGTIGT**, at I ser de mails, som jeg og instruktorerne sender
- Alle mails sendes til jeres officielle AU adresse
- Videre sendelse af mails bør etableres
- Se hvordan det gøres på  
<https://studerende.au.dk/it-support/mail/vejledninger-til-opsaetning-af-mail/>
- Hvis I har problemer, så spørg jeres instruktør og/eller medstuderende

- **I skal installere BlueJ inklusiv Java 8 JDK**

- Følg linket på den Brightspace side, der hedder "BlueJ + Java", som findes under "Information om kurset (inklusive nyttige links)"
- Hvis I har problemer, så spørg jeres instruktør og/eller medstuderende

- **Læs kursets Brightspace sider og følg med i de indlæg, der kommer på diskussionsforummet og under "Vigtige meddelelser"**

- Læs også gerne **ugebrevene**, som indeholder information om, hvordan man mest hensigtsmæssigt "angriber" ugens stof

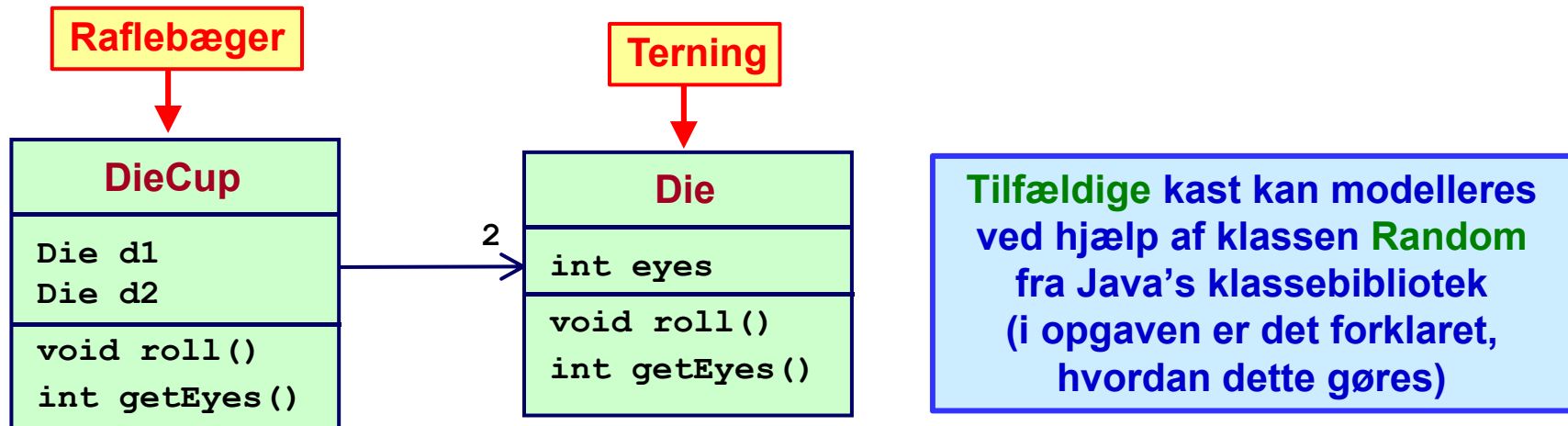
# Studiestartsprøve

---

- **Gælder alle nye bachelorstuderende**
  - Prøvens hovedformål er at identificere de studerende, der ikke har påbegyndt studiet, så de kan udmeldes inden de officielle optagelsestal opgøres
- **I begyndelsen af september vil I modtage en mail på jeres au-mailadresse**
  - Mailen indeholder et link til et spørgeskema, der handler om studievalg og studiestart
  - Det er **obligatorisk** at svare og på den måde vise, at I er studieaktive
  - Hvis I ikke svarer (inden for få dage) bliver I **automatisk frmeldt** jeres studie

# ● Afleveringsopgave: Raflebæger 1 (DieCup 1)

- I skal implementere et system med et raflebæger og to terninger



- **Terningen har to metoder:**
  - roll() repræsenterer et kast med terningen
  - getEyes() returnere det viste antal øjne (i sidste slag)
- **Raflebægeret indeholder to terninger og har to metoder:**
  - roll() repræsenterer et kast med de to terninger
  - getEyes() returnere det viste antal øjne (i sidste slag)

**Demo**



# ● Opsummering

---

- **Hvad er programmering?**
  - Program, der kan løse Sudoku opgaver (eksempel)
  - Programmering og problemløsning (generelt)
- **Agenter og metoder**
- **UML specifikationssproget**
  - Klassediagrammer
  - Sekvensdiagrammer
- **Information om kurset**
  - Hvad kan I forvente at lære?
  - Undervisningsprincipper
  - Masser af praktiske oplysninger
- **Afleveringsopgave: Raflebæger 1 (DieCup 1)**
  - Demo af BlueJ programmeringsomgivelsen

# Universitetsstudier er fuldtidsarbejde

---

- **Vi forventer, at I arbejder 45 timer pr uge, dvs. 15 timer pr kursus**
  - Svarer til en 37 timers arbejdsuge – når de eksamens- og undervisningsfrie perioder tages med i beregningen
  - En typisk arbejdsuge indeholder 4 timers forelæsning, 4 timers øvelser og 7 timers "hjemmearbejde" – alene, i par eller i jeres læsegruppe
  - Studerende med programmeringserfaring kan i **begyndelsen** klare kurset med lidt lavere belastning
- **En del studerende med programmeringserfaring undervurderer kurset og klarer sig derfor væsentligt dårligere til eksamen end de burde**
  - Undgå at falde i den faldgruppe
  - Det er for dumt at score en middelmådig karakter i et kursus, som man med en lidt bedre indsats burde klare sig godt i
- **Husk at begreber, brug af korrekt terminologi og pæn, velstruktureret programmeringsstil er vigtige**
  - Det ikke nok at kunne "hacke" noget kode sammen, der virker
  - Man skal også forstå og kunne forklare principperne bag koden

# Fast timeplan

- Som ny studerende kan det være en god ide at lave et fast arbejdschema, således at tingene ikke bare flyder

	MAN	TIR	ONS	TOR	FRE	LØR	SØN
8-9	TØ	fri	fri	fri	forelæsning	fri	fri
9-10	TØ	studiecafé	fri	TØ	forelæsning	fri	fri
10-11	studiecafé	studiecafé	læsegruppe	TØ	studiecafé	fri	fri
11-12	frokost	studiecafé	læsegruppe	frokost	studiecafé	fri	fri
12-13	forelæsning	studiecafé	frokost	studiecafé	frokost	fri	fri
13-14	forelæsning	frokost	TØ	studiecafé	studiecafé	fri	fri
14-15	forelæsning	studiecafé	TØ	forelæsning	læsegruppe	fri	fri
15-16	forelæsning	studiecafé	studiecafé	forelæsning	læsegruppe	fri	studér
16-17	fri	læsegruppe	studiecafé	fri	fri	fri	studér
17-18	fri	læsegruppe	fri	fri	fri	fri	studér
18-19	fri	fri	fri	fri	fri	fri	fri
19-20	fri	fri	fri	fri	fri	fri	fri
20-21	studér	fri	læsegruppe	studér	fri	fri	studér
21-22	studér	fri	læsegruppe	studér	fri	fri	studér

- Video om time management [Link](#)
- I uge 2 er der en studieteknikopgave om Studievaner

**Det var alt for nu.....**

**... spørgsmål**

---

