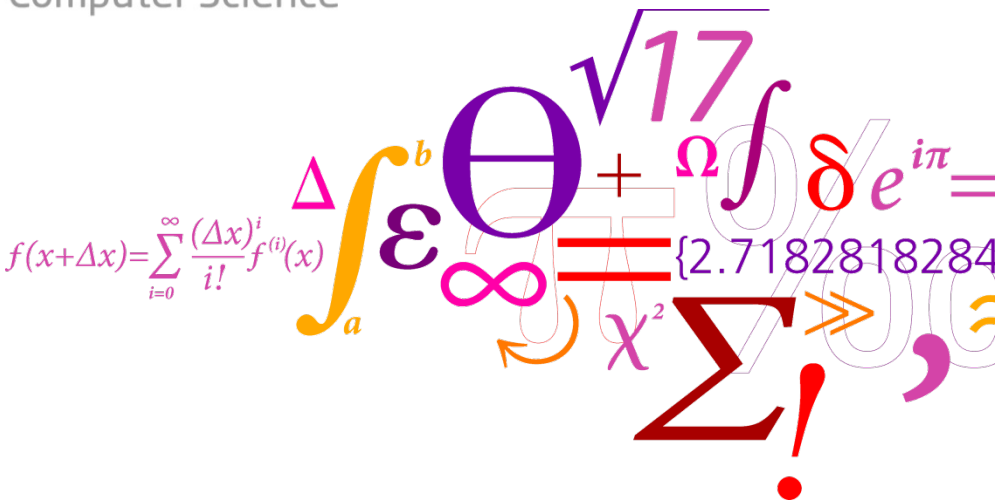
Projekt i software-udvikling (02362)
Forår 2020

Ekkart Kindler

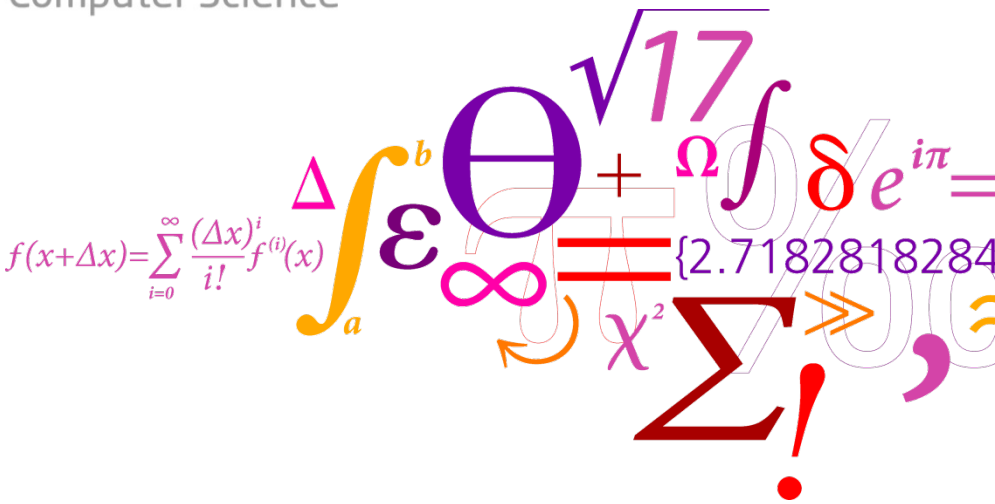
DTU Compute
Department of Applied Mathematics and Computer Science

Bemærk: Kun slides 2-3 og 10-14 er nye. Alle andre slides blev præsenter allerede!



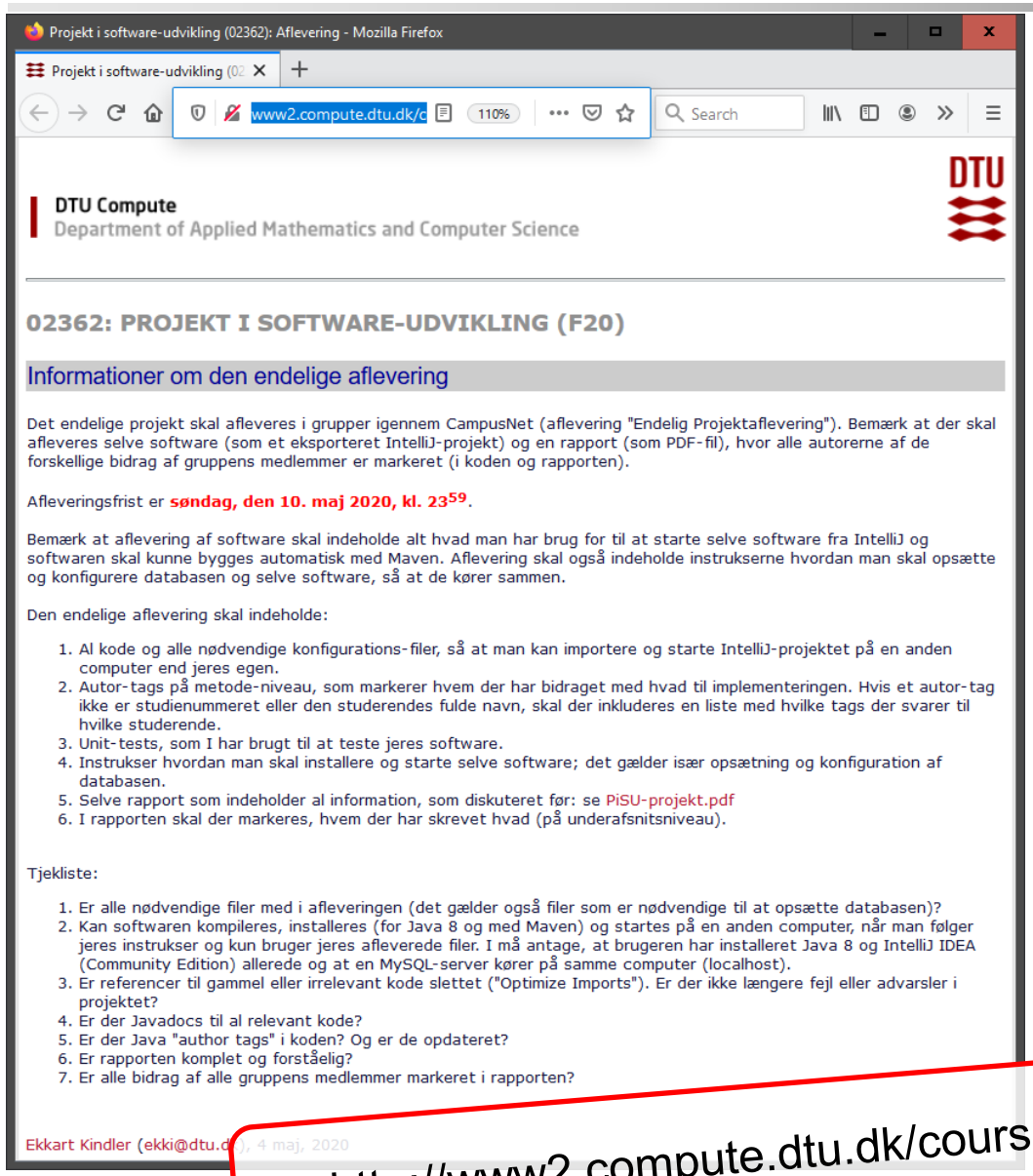
X. Projekt: Aflevering og Eksamen

DTU Compute
Department of Applied Mathematics and Computer Science



Info og Tjeklist

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Projekt i software-udvikling (02362): Aflevering - Mozilla Firefox

Projekt i software-udvikling (02

www2.compute.dtu.dk/c 110% Search

DTU Compute
Department of Applied Mathematics and Computer Science

02362: PROJEKT I SOFTWARE-UDVIKLING (F20)

Informationer om den endelige aflevering

Det endelige projekt skal afleveres i grupper igennem CampusNet (aflevering "Endelig Projektaflevering"). Bemærk at der skal afleveres selve software (som et eksporteret IntelliJ-projekt) og en rapport (som PDF-fil), hvor alle autorerne af de forskellige bidrag af gruppens medlemmer er markeret (i koden og rapporten).

Afleveringsfrist er **søndag, den 10. maj 2020, kl. 23^{59}**.

Bemærk at aflevering af software skal indeholde alt hvad man har brug for til at starte selve software fra IntelliJ og softwaren skal kunne bygges automatisk med Maven. Aflevering skal også indeholde instrukserne hvordan man skal opsætte og konfigurere databasen og selve software, så at de kører sammen.

Den endelige aflevering skal indeholde:

1. Al kode og alle nødvendige konfigurations-filer, så at man kan importere og starte IntelliJ-projektet på en anden computer end jeres egen.
2. Autor-tags på metode-niveau, som markerer hvem der har bidraget med hvad til implementeringen. Hvis et autor-tag ikke er studienummeret eller den studerendes fulde navn, skal der inkluderes en liste med hvilke tags der svarer til hvilke studerende.
3. Unit-tests, som I har brugt til at teste jeres software.
4. Instrukser hvordan man skal installere og starte selve software; det gælder især opsætning og konfiguration af databasen.
5. Selve rapport som indeholder al information, som diskuteret før: se PiSU-projekt.pdf
6. I rapporten skal der markeres, hvem der har skrevet hvad (på underafsnitsniveau).

Tjekliste:

1. Er alle nødvendige filer med i afleveringen (det gælder også filer som er nødvendige til at opsætte databasen)?
2. Kan softwaren kompileres, installeres (for Java 8 og med Maven) og startes på en anden computer, når man følger jeres instrukser og kun bruger jeres afleverede filer. I må antage, at brugeren har installeret Java 8 og IntelliJ IDEA (Community Edition) allerede og at en MySQL-server kører på samme computer (localhost).
3. Er referencer til gammel eller irrelevant kode slettet ("Optimize Imports"). Er der ikke længere fejl eller advarsler i projektet?
4. Er der Javadocs til al relevant kode?
5. Er der Java "author tags" i koden? Og er de opdateret?
6. Er rapporten komplet og forståelig?
7. Er alle bidrag af alle gruppens medlemmer markeret i rapporten?

Ekkart Kindler (ekki@dtu.d), 4 maj, 2020

Se http://www2.compute.dtu.dk/courses/02362/f20/aflevering.shtml

Rapport (indhold)

Se PiSU-projekt.pdf og også eksemplet (materiale til uge 12)!

- Introduktion
  - Kort overblik over projektet og hovedpunkter, som opgaven indebærer
  - Overblik over selve rapporten og hvordan den skal læses
- Problembeskrivelse
  - Kontekst og baggrund
  - Overblik over hovedfunktioner (i grupper) med prioritering og argumentation for denne (kan fx bruge MoSCoW-kategorier: Must, Should, Could, Won't)
  - Husk også at nævne, hvilke data, der skal gemmes permanent (persistence)
  - Hvad kan I bygge videre på (fx GUI og kode i fik fra Ekkart v0.0.2a - v0.6.2a)

Rapport (indhold)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Kravspecifikation (Analyse)
  - Beskrivelse af krav fra brugerens perspektiv
  - Beskriv funktioner og use-cases (som brugeren opfatter dem)
  - Beskriv de relevante informationer som domænemodel:
    - Klassediagrammer
    - Aktivitetsdiagrammer
    - evt. tilstandsdiagrammer

Husk: Alle diagrammer skal forklares i teksten!

  - Beskriv ikke-funktionelle krav: fx brugbarhed (usability), vedligeholdbarhed (maintainability), …
  - Evtl. UC-diagram med beskrivelse af de vigtigste use-cases

Rapport (indhold)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Database- og softwaredesign

  Overblik over de vigtigste komponenter, som inkluderer, også tilkobling a databasen: arkitektur

  - Software design: hovedkomponenter af software og deres sammenspil

    → Især MVC, DAL, …

    - Klassediagrammer (indeholder især de vigtigste model-, kontroller- og view-klasser)
    - Sekvensdiagrammer som viser samspil mellem vigtige klasser/komponenter
  - Database design → se DB kursus

Rapport (indhold)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Implementering
  - Hvordan er designet bliver implementeret (men kun nogle interessante udtræk, ikke hele koden)
  - Dette omfatter software og database
  - Herunder kan I også diskutere evt. mangler af jeres implementering
- Udviklingsproces og test
  - Udviklingsproces og brugte værktøjer
  - Diskussion af JUnit-test (automatisk)
  - Diskussion af acceptance-test (manuelle test, fx use-cases)

Rapport (indhold)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Det skal være muligt at bruge softwaren uden anden information!

- Håndbog
  - Hvordan installerer man selve software
  - Hvordan starter man softwaren
  - Hvordan bruger man software (den eksisterende GUI er I ikke nødt til at diskutere med alle detaljer); evt. med nogle screenshots
- Konklusion
  - Opsamling af hele resultatet
  - Nogle afsluttende bemærkninger

Rapport (indhold)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Referencer
  - Litteratur og
  - Websider I har bugt
- Appendiks (evt.)
  - Nogle relevante diagrammer som ikke kunne være med i hoveddelen
  - Komplet ordliste / glossar

Hus at sige: Hvem der har skrevet hvad!

Gruppepræsentation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

15 minutter

Den røde tråd: Why, What, How, Demo

- Why: Indledning & Motivation

og måske "what" på meget højt abstraktionsniveau.

- What: Hvad har i udviklet fra brugerens synsvinkel (Analyse)
  - Hvilke hoved-usecases og -funktioner omfatter jeres software
  - GUI (hvis der er noget specielt hos jer)
  - Tekniske perspektiv af det: domænemodel (klassediagrammer, tilstandsdiagrammer, aktivitetsdiagrammer)

Why, what, how: gerne med nogle Powerpoint-slides

Gruppepræsentation

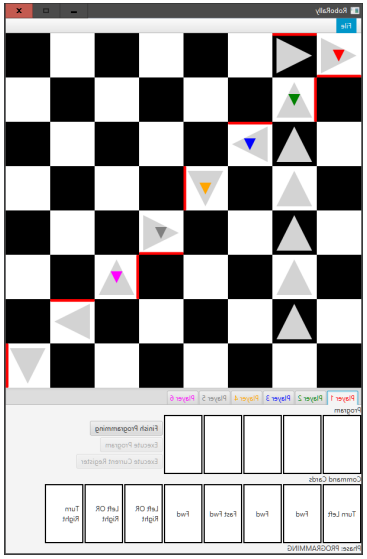
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- How: Hvordan er softwaren realiseret
  - Arkitektur & Design
  - Nogle få implementeringsdetaljer

Især: MVC og DAL

Et kort overblik af jeres IntelliJ-projekt og måske database (MySQL workbench, etc.)

- Demo: Vis hvad jeres software kan



Tip: Gem et spil i en spænende situation, så at I kan vise situationer, som kun sker senere hen i spillet (checkpoint, vinde, skubbe, pit, …)

Hvert foredrag skal have en indledning og afslutning (afrunding)

“Deltagere”

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

I skal tale til

- en med ikke så meget teknisk forstand (CEO), som skal forstå hvad softwaren kan gøre
- og end med mere teknisk forstand (CTO), som I skal overbevise at det I gøre giver mening og I har styr på jeres design

så at begge to får noget ud af det

Individuelle eksaminer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

15 minutter

I skulle kunne svare på, hvordan de forskellige emner fra forelæsningen kan findes I jeres analyse, design og implementering.

Det gælder især:

- Domænemodel og diagrammer
- MVC
- Detaljer af jeres
  - controllere: app- og spillelogik
  - DAO / filer
- Exceptions
- Generics

Hver studerende skal bringe en computer med softwareprojektet installeret i IntelliJ, så at I kan pege på tingene i implementeringen i jeres IDE.

I skal også have rapporten på jeres computere, så at vi fx. kan spørge om jeres diagrammer i rapporten

Tidsplan

Se http://www2.compute.dtu.dk/courses/02362/f20/eksamen.shtml

Mandag, 18. maj

9^{30} - 12^{00}: Group A

13^{30} - 16^{15}: Group B

Tirsdag, 19. maj

9^{30} - 12^{15}: Group C

13^{30} - 16^{15}: Group D

Projekt i software-udvikling

www2.compute.dtu.dk/

110%

DTU Compute
Department of Applied Mathematics and Computer Science

02362: PROJEKT I SOFTWARE-UDVIKLING (F20)

Eksamen

Eksaminer til kursus "Projekt i software-udvikling" (02362) finder sted mandag, den 18. maj og tirsdag, den 19. maj 2020! Eksaminer foregår på Zoom — linket til Zoom-mødet bliver bekendtgjort via en besked på CampusNet (Inside).

Formatet er at der først ville være en gruppepræsentation på ca. 15 minutter med en Powerpoint-præsentation og en live-demo af projektet (hvor alle gruppemedlemmer skal være en aktiv del af). Bagefter er der en eksamination af de enkelte gruppemedlemmer (på ca. 15 minutter hvert). Hvert studerende skal have gruppens udviklede software installeret på dens egen computer og kunne køre den og vise den i IntelliJ.

Der ville også være lidt mere information om eksamen under sidste forelæsning i uge 13.

For at gennemføre eksamenen skal hver studerende have installeret

- Zoom, samt et web-kamera og et mikrofon,
- IntelliJ med en kørbar version af gruppens software (som inkluderer også en kørende database) og
- mulighed for at vise en præsentation (enten Powerpoint eller PDF) på dens computer,

Desuden skal hver studerende vise billedlegitimation (studiekort eller pas) på en måde, så bedømmerne kan se det. Endvidere skal den studerende med sit web-kamera vise rummet vedkommende sidder i, for at bedømmerne kan sikre sig, at den studerende er alene i rummet. Desuden henviser DTU på DTU's æreskodeks og nogle generelle ting vedr. online-eksamen på DTU Inside her.

Zoom-mødet til eksamenen bliver konfigureret med et "venteværelse", hvor alle studerende af en gruppe skal være logge på imens gruppens eksamen foregår. Alle studerende skal logges på i god tid op tid eksamenen. På deres tidspunkter bliver de forskellige studerende så "kaldt ind" i mødelokalet (i starten er det hele gruppen til gruppepræsentation).

For at afprøve og teste opsætningen og udstyret afholdes der gruppemøderne den 5. maj med samme opsætning og på samme Zoom-mødet, som også eksamenen ville foregår. Tidsplanen til prøveeksamen/gruppemøderne bliver bekendtgjort på websiden med materiale og informationerne om uge 13 på kursets websider.

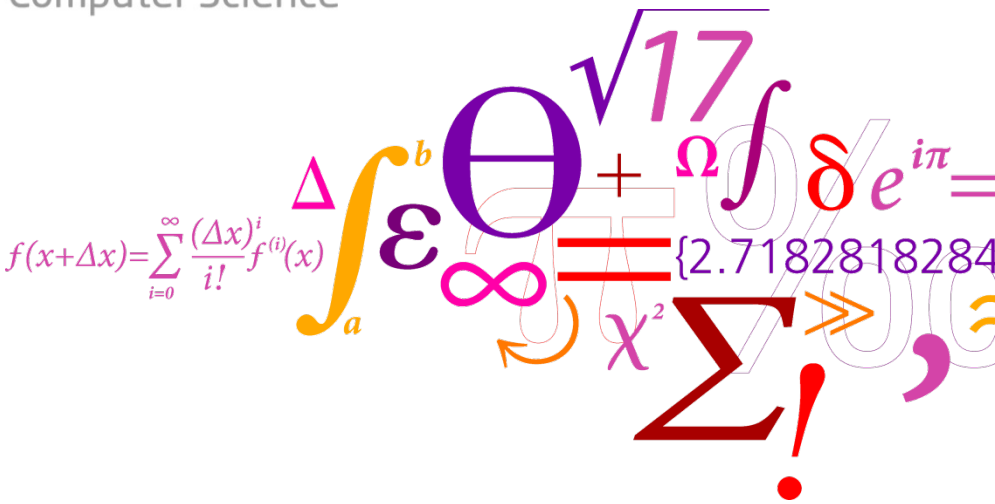
Tidsplanen til eksaminer til de forskellige grupper er:

Mandag, den 18. maj 2020:

- 9^{30} - ca. 12^{00}: Gruppe A
- 13^{30} - ca. 16^{15}: Gruppe B

XI. Afslutning

DTU Compute
Department of Applied Mathematics and Computer Science



Kursus: Kort sagt

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Indtil nu (1. semester):

- **Kendskab** til nogle programmerkonstrukter
- Basale **færdigheder** i programmering

Efter dette kursus

- **Programmererfaring** med et lidt større projekt
- **Kendskab** til nogle yderlige programmerkonstrukter
- **Erfaring** med god programmeringsstil
- **Erfaring** med at debugge projekter
- **Lidt erfaring** med databaser

Emner

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- (Java) Interfaces
- Datatyper
  - egne
  - brug af indbyggede datatyper i Java
- Rekursion
- Exceptions
- Generics
- Tests

Emner

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Modellering (domæne og designmodeller)
- God stil og skik i programmering
- Java docs
- Design pattern
- Model-View-Controller-princippet (MVC)
- Brugerdialog og inputvalidering (regulære udtryk)
- Filer
- Threads
- Tilknytning til database (→ 02327 og ekstra forelæsning uge 6 / uge 7)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Motivation

- Implementeringer i programmer er ofte meget teknisk
- Der er mange dataer som kan ødelægges, når man ikke bruger dem rigtigt
- Man ved ikke hvad der er vigtigt og ikke så vigtigt

- Programmet er svært at forstå
- Programmet kan nemt ødelægges

Interface `Stack`

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public interface Stack {

  void clear();

  Integer pop();

  Integer top();

  void push(Integer value);

  int size();

  default boolean isEmpty() {
    return size() == 0;
  }
}
```

Implementering (eks.)

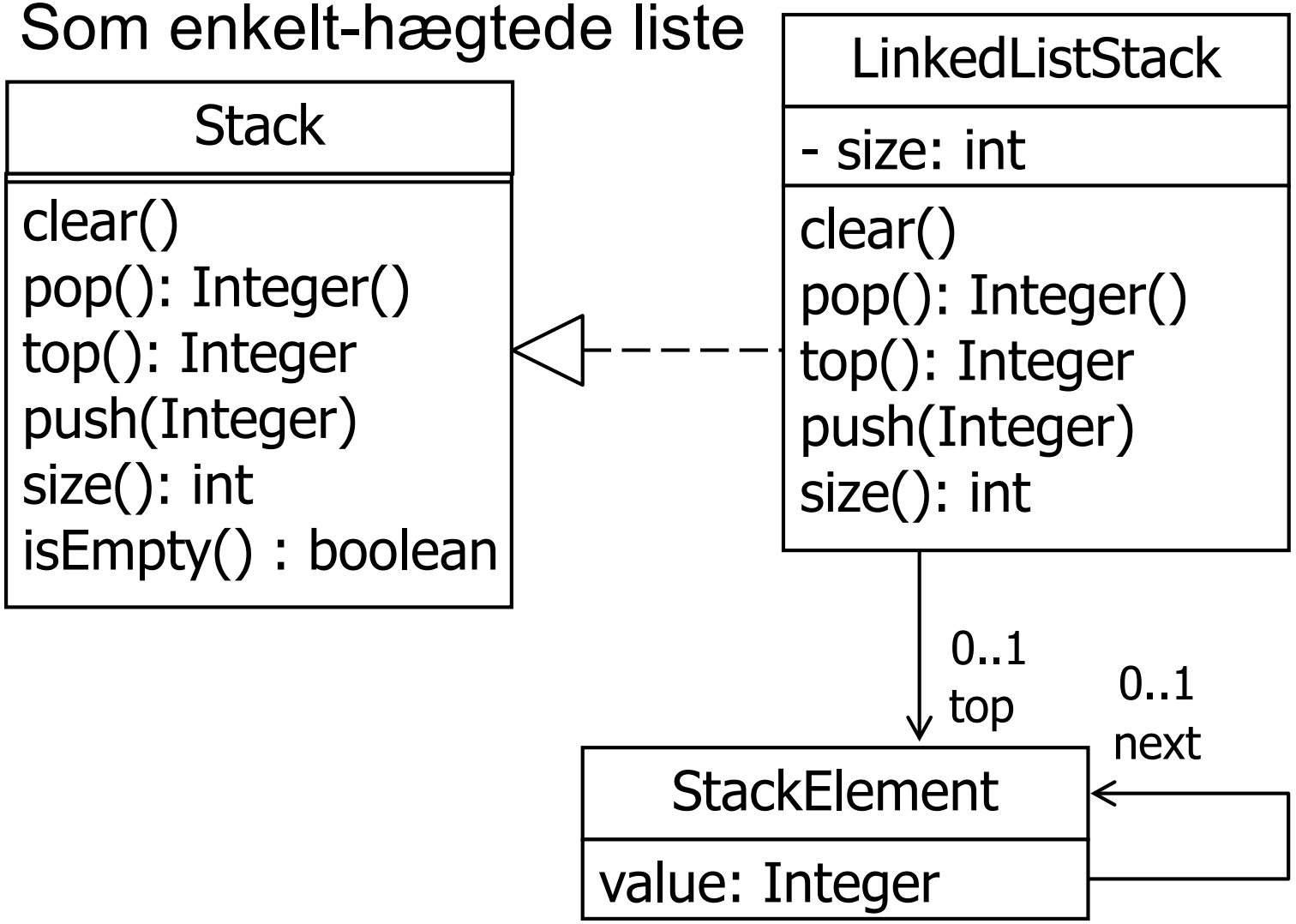
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public class LinkedListStack implements Stack {

   @Override
   public void clear() {
      ...
   }

   @Override
   public Integer pop() {
      ...;
   }
   ...
}
```

Implementeringer

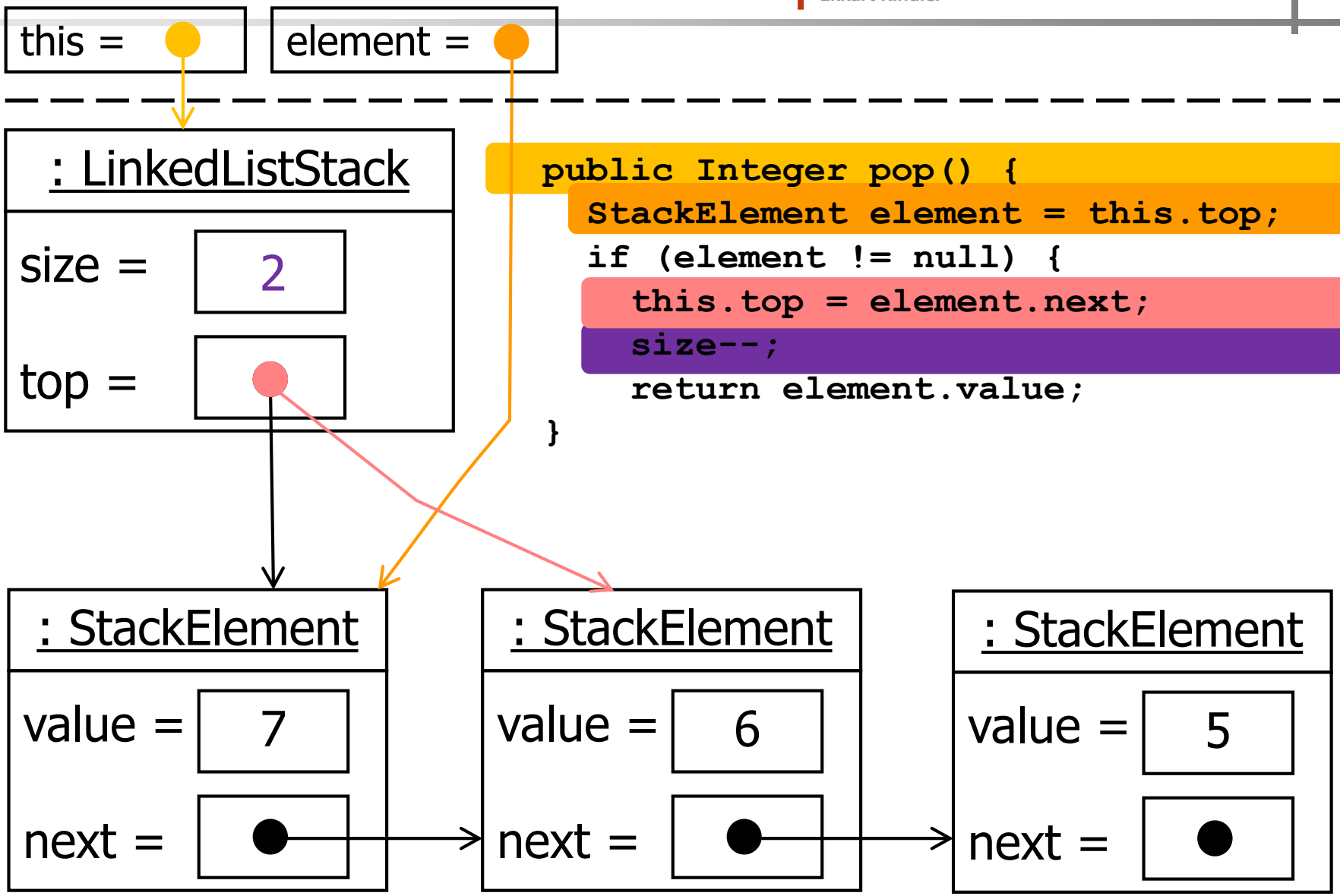
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Som enkelt-hægtede liste



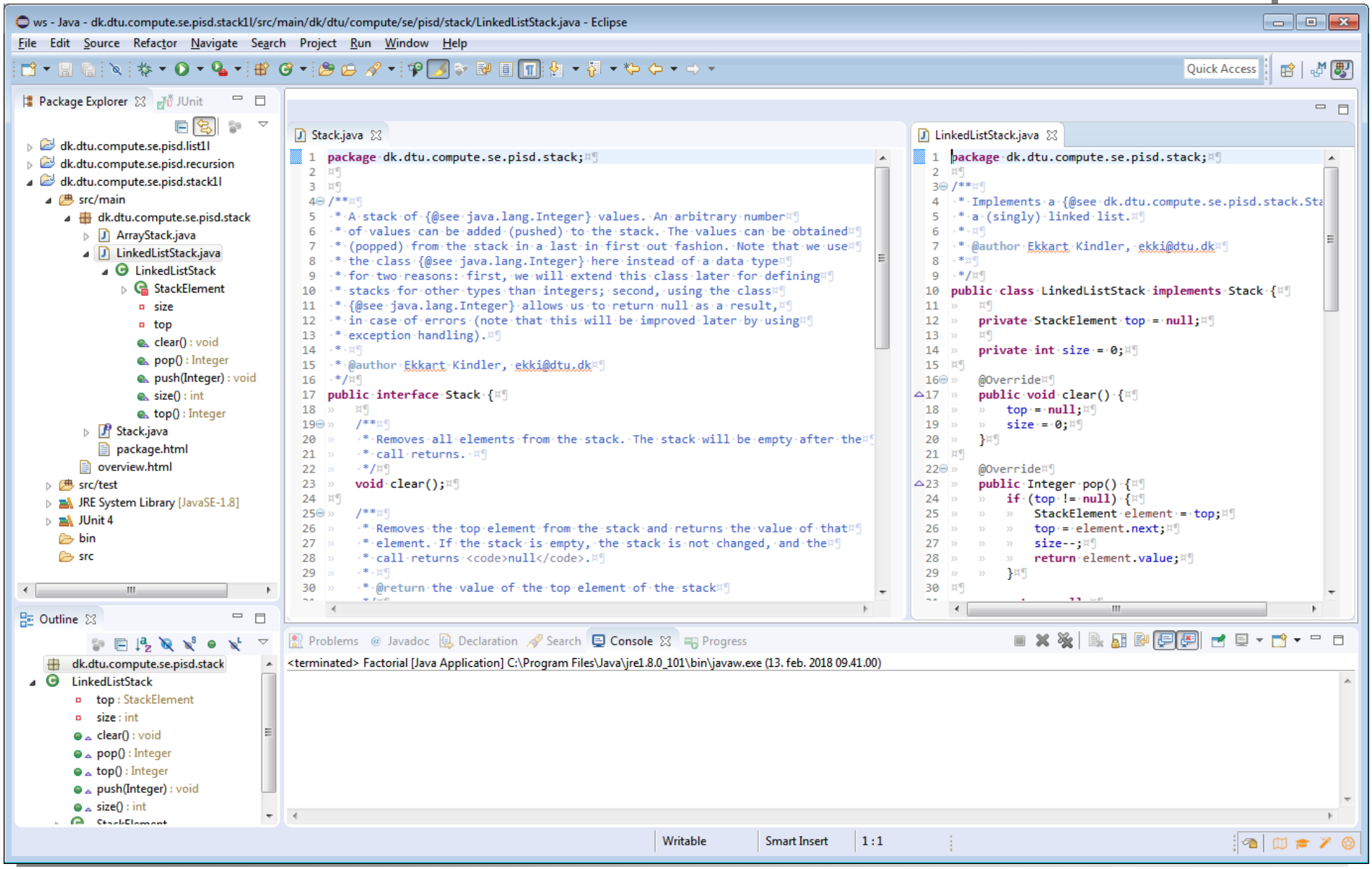
pop()

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



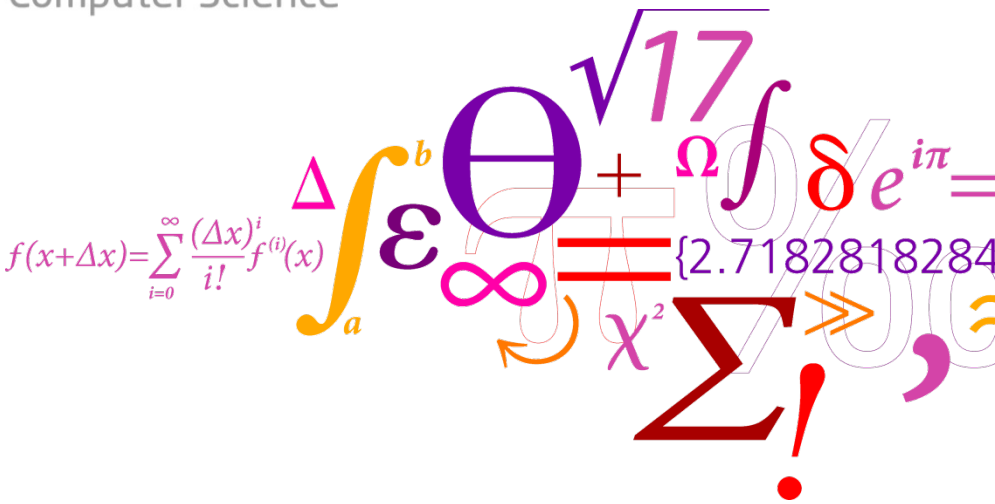
JavaDoc

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



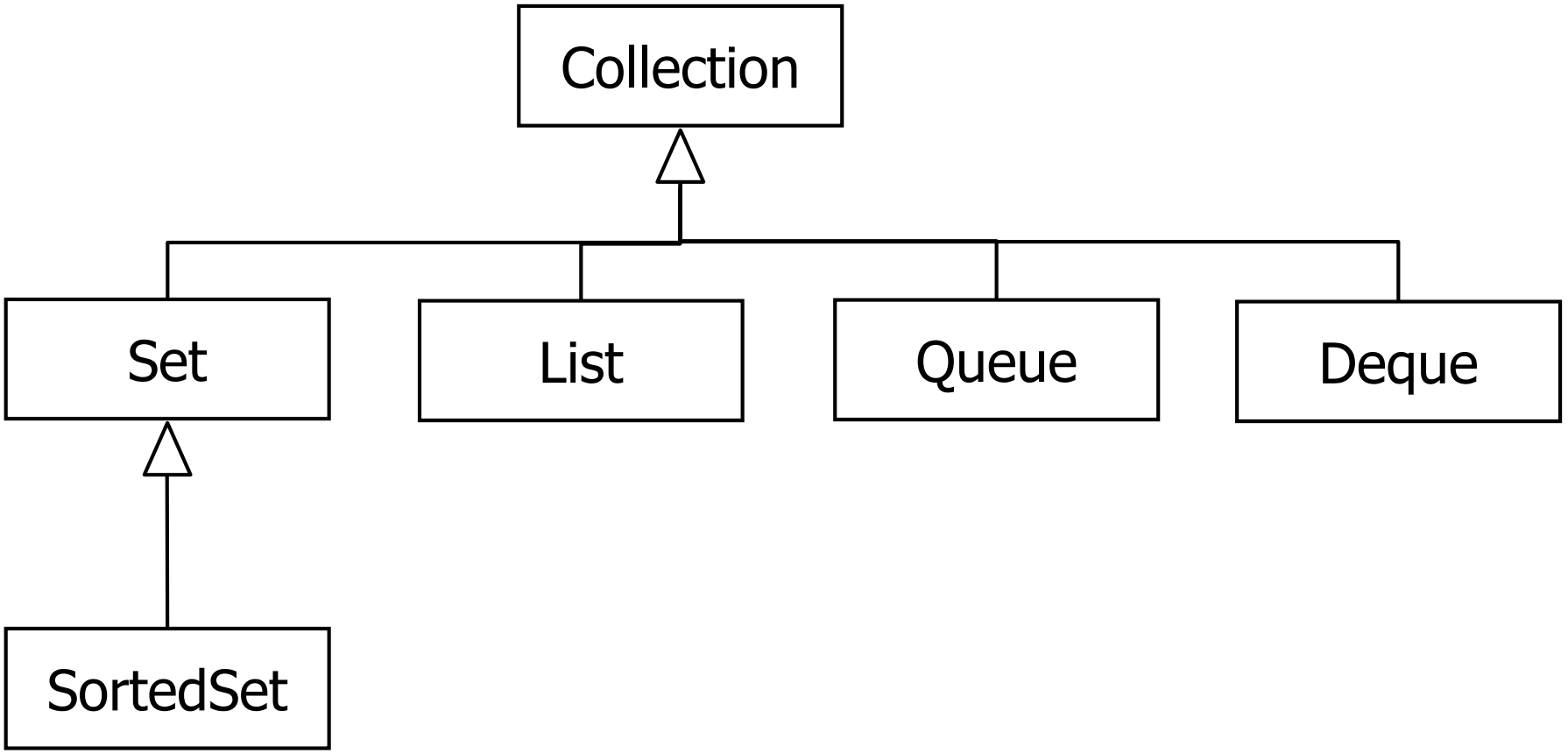
Javas indbyggede datastrukturer

DTU Compute
Department of Applied Mathematics and Computer Science



Collections overview

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Comparable

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public interface Comparable<T> {

    public int  compareTo(T o);

}
```

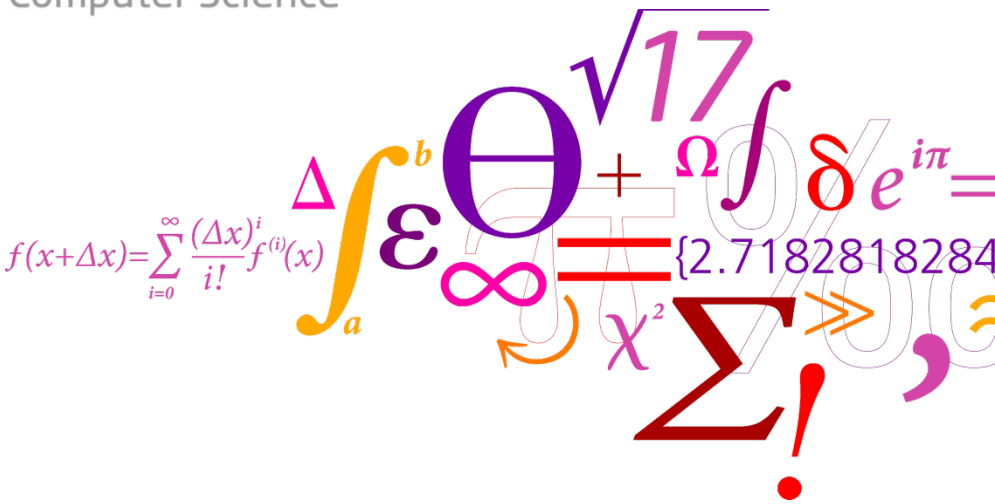
`o1.compareTo(o2) < 0` hvis object `o1` er mindre end object `o2`

`o1.compareTo(o2)== 0` hvis object `o1` og object `o2` er lige

`o1.compareTo(o2) > 0` hvis object `o1` er større end object `o2`

Lister, Sortering, og Søgning

DTU Compute
Department of Applied Mathematics and Computer Science



Bubble sort

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Lidt mere systematisk:

Vi antager at listen ligger i et array

```
int[] list = ...
```

og

```
int size = ...
```

er den aktuelle længde af listen

Bubble sort (1st version)

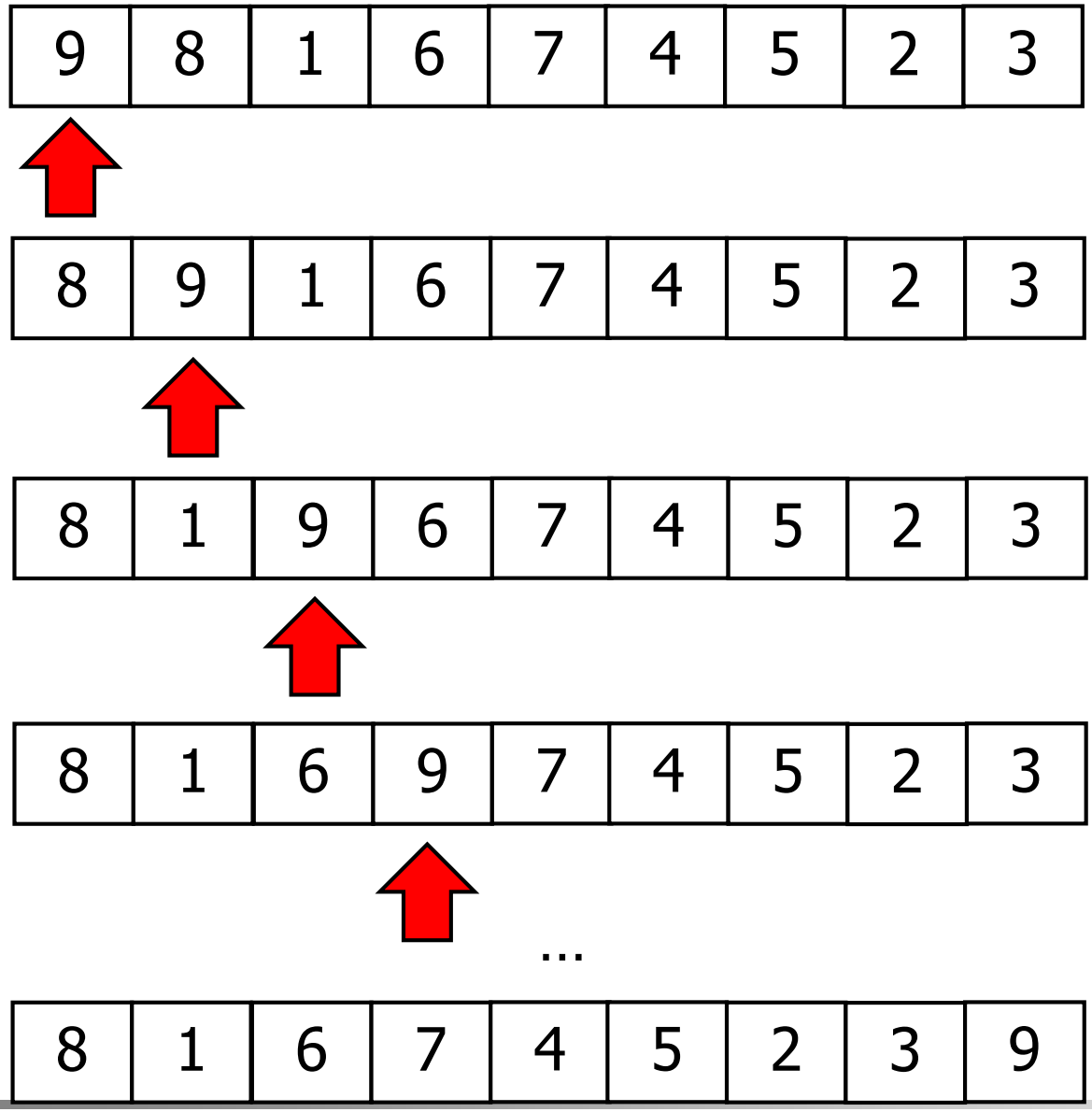
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
boolean swapped;
do {
   swapped = false;
   for (int i=0; i+1<size; i++) {
      if (values[i] > values[i+1]) {
         int v = values[i];
         values[i] = values[i+1];
         values[i+1] = v;
         swapped = true;
   } }
} while (swapped);
```

Byt systematisk, så længe der er elementer ved siden af hinanden, som ikke er sorteret.

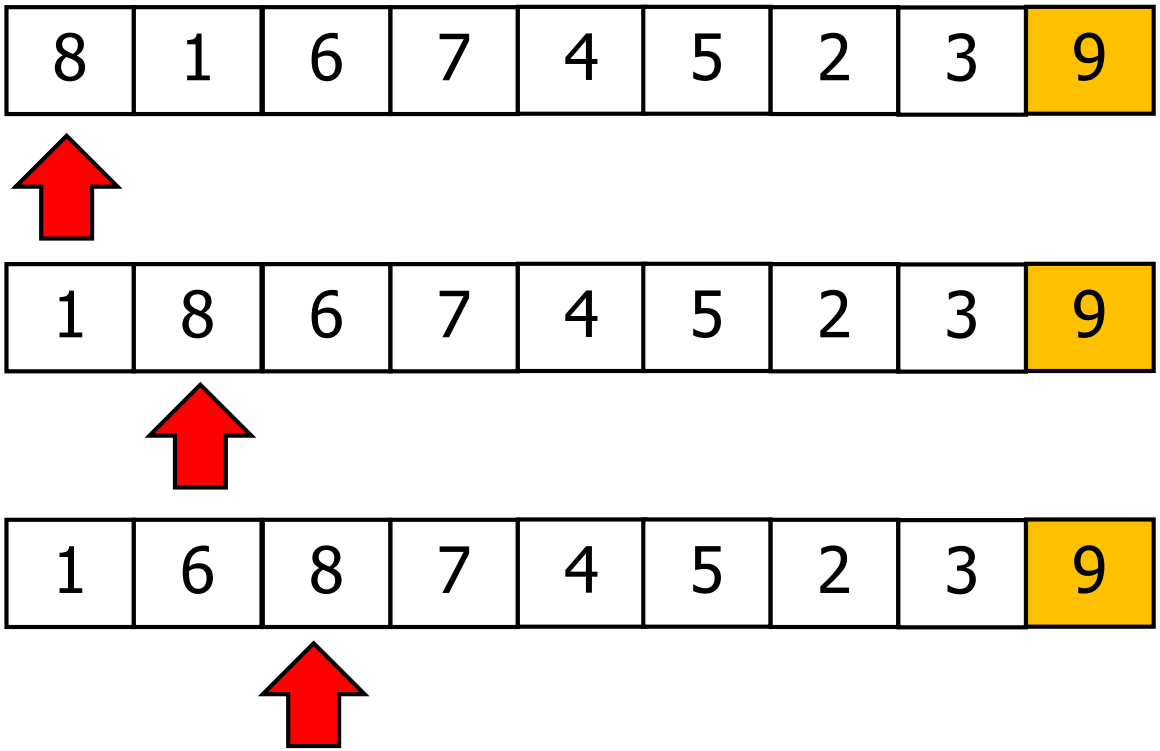
Første iteration

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



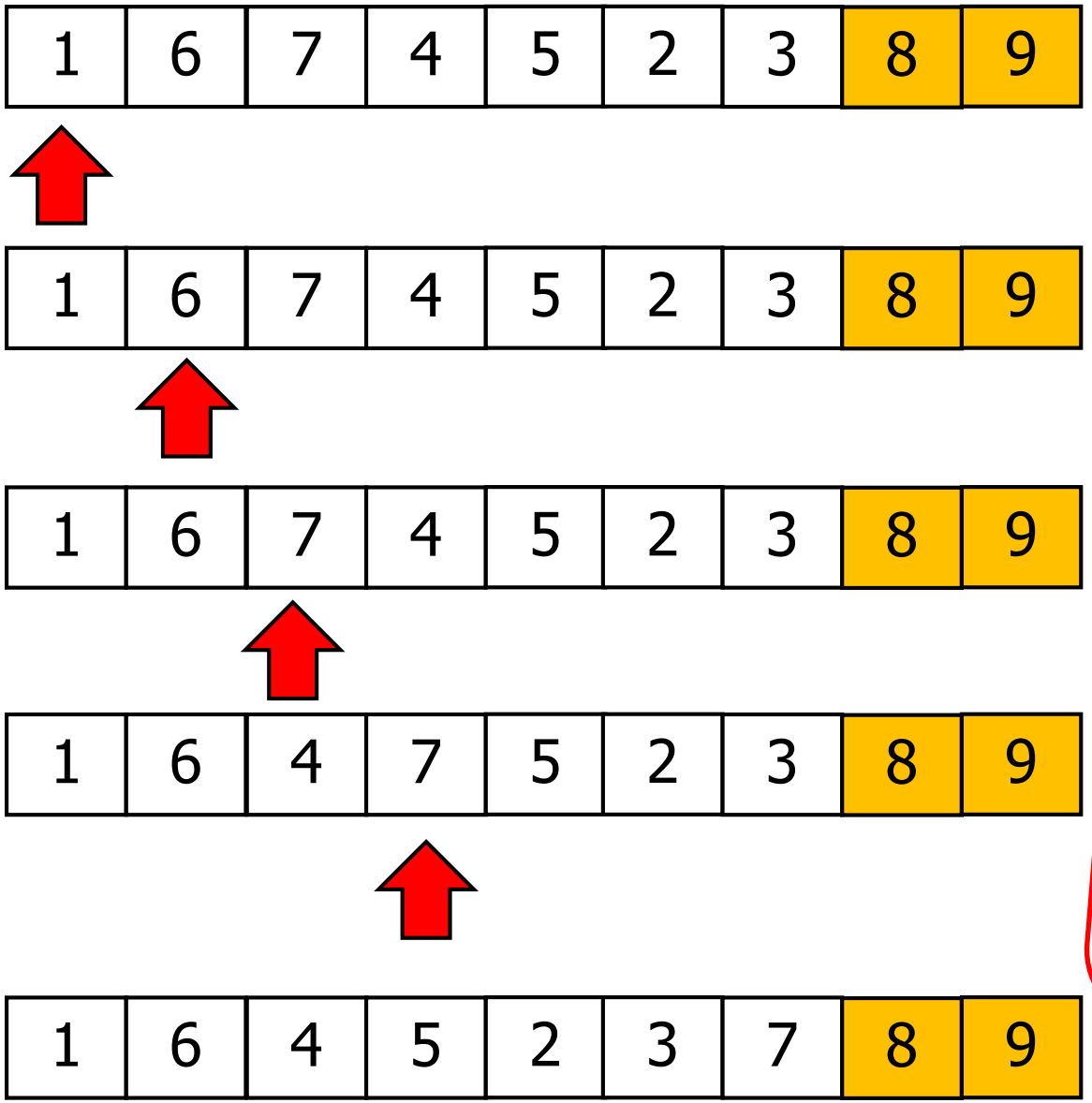
Anden iteration

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Tredje iteration

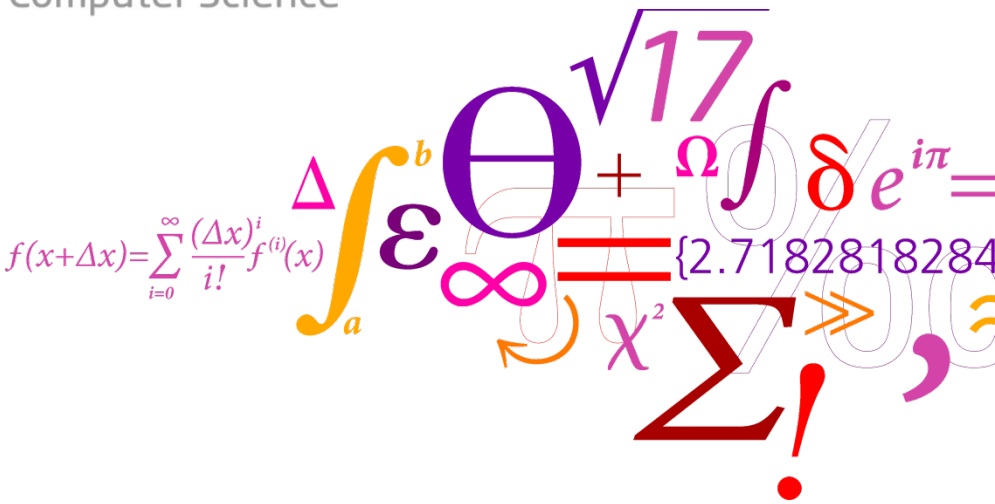
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

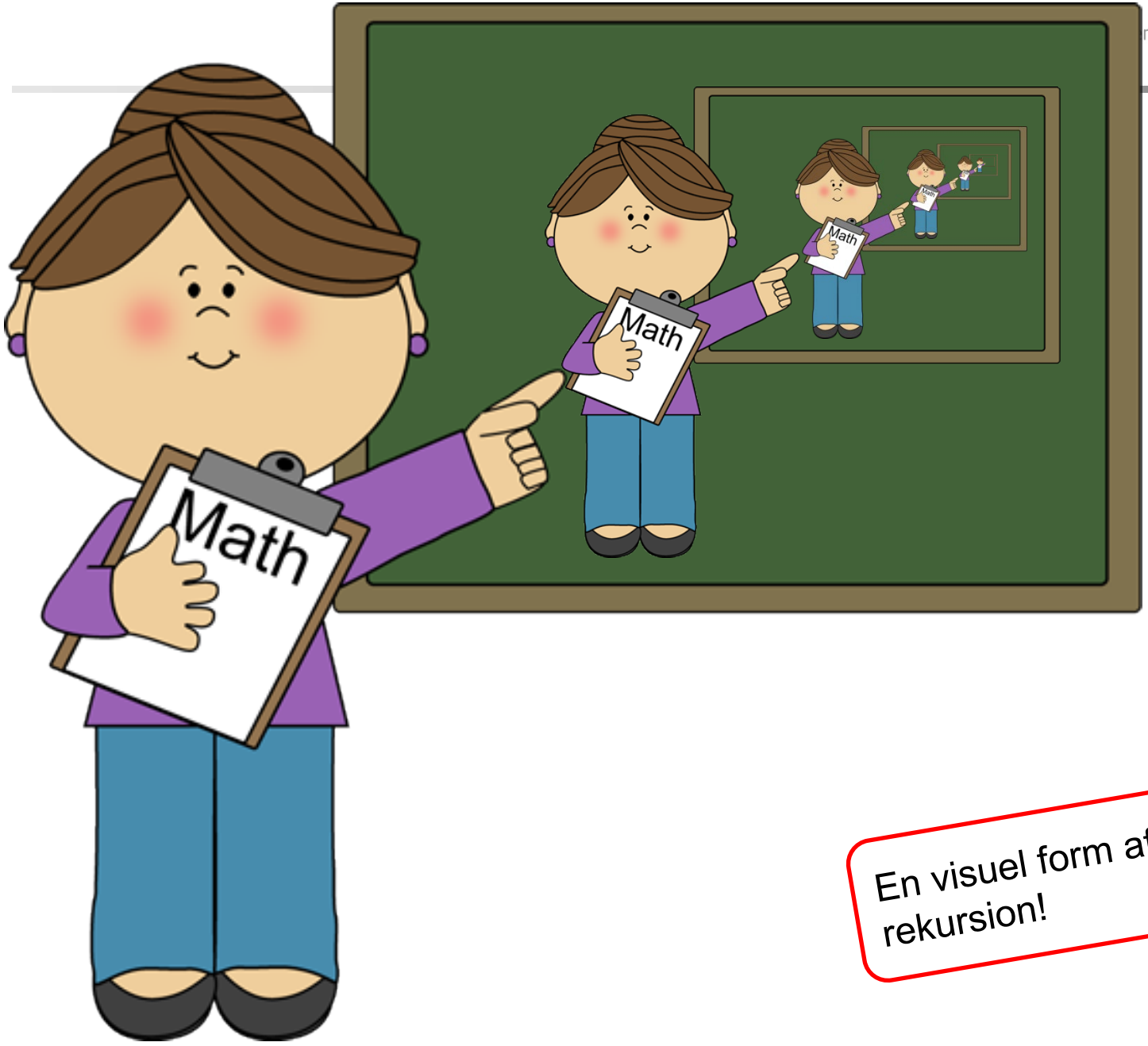


I hver iteration kan vi stoppe en position tidligere!

Rekursion

DTU Compute
Department of Applied Mathematics and Computer Science





En visuel form af en rekursion!

Rekursiv definition *n!*

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

$$n! = \begin{cases} 1 & \text{hvis } n=0 \\ n * (n-1)! & \text{hvis } n>0 \end{cases}$$

Her er der **rekursion** igen

Fakultet-funktion i Java

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
private static int factorial(int n) {
   if (n == 0) {
      return 1;
   } else if (n > 0) {
      return n * factorial(n-1);
   } else {
      throw IllegalArgumentException(
        "f(" + n + ")" );
   }
}
```

Fakultet-funktion i Java

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
private static double factorial(int n) {
   if (n == 0) {
      return 1;
   } else if (n > 0
      return n * fact          (n-1);
   } else {
      throw Ille
      "f(" + n +
   }
}
```

Nogle detaljer om **primitive datatyper** i Java:
`int` og `long`: når tal bliver for store/små, bliver de forkerte (uden advarsel)!

Det starter allerede, at gå galt med `int` på `factorial(14)`!

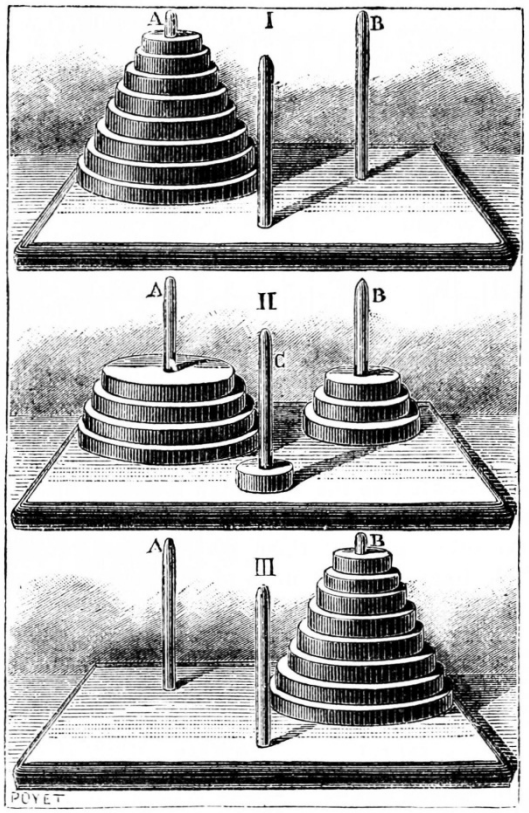
Når man bruger `double` og `float` eksisterer værdier som `Infinity` og `NaN` (Not a Number)), som viser, at resultatet er for stort eller forkert.

Andet Eksempel

Engl. Towers of Hanoi

Hanois tårne: Matematisk ”puslespil”:

- Stativ med tre pinde (*A*, *B* og *C*)
- På pind *A* er der *n* skiver med aftagende størrelse (set nedfra)
- **Opgave**:
  Flyt alle *n* skiver fra *A* til *B*, men
  - Kun en skive ad gangen
  - En større skive må aldrig ligge på en skive som er mindre

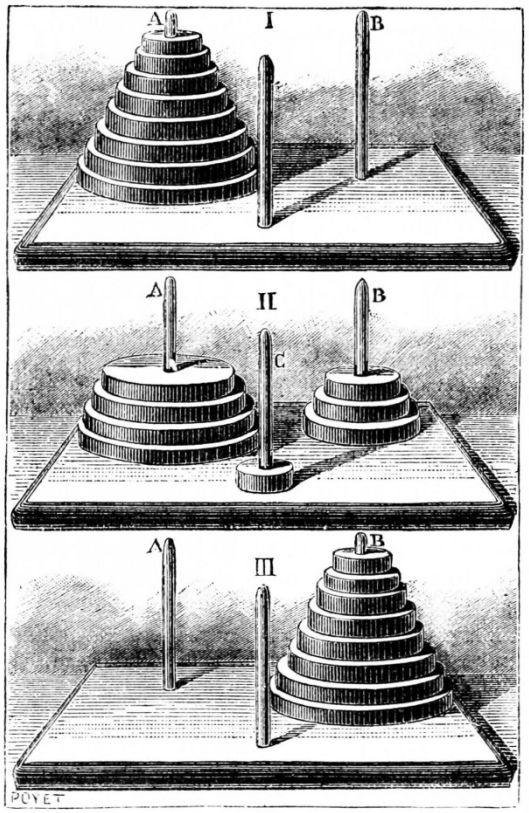


Løsningsidé:

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Flyt *n* skiver fra *A* til *B*

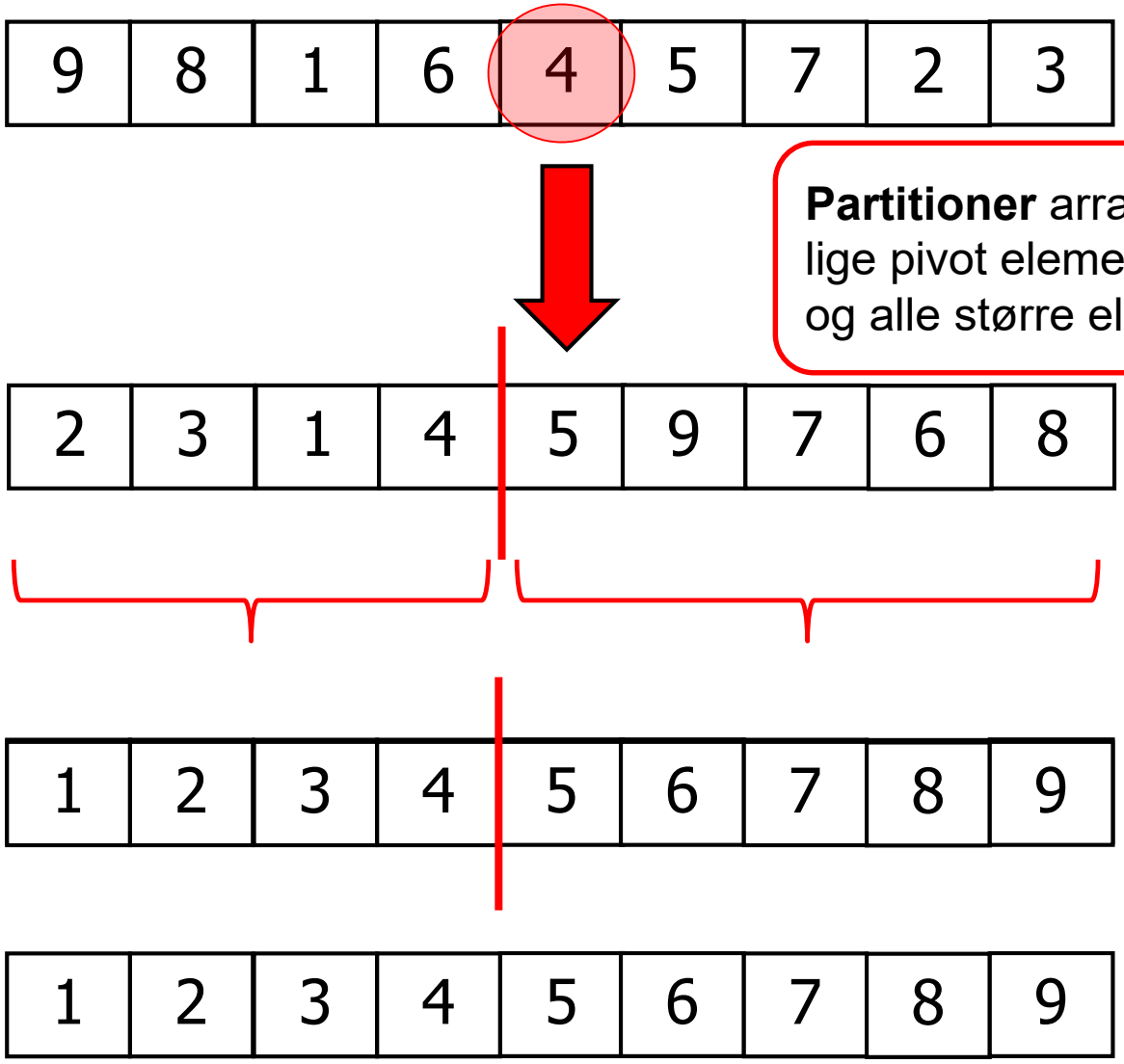
1. Flyt de øverste *n-1* skiver fra *A* til *C* (iføge reglerne)
2. Så fly den nedereste (størrste) skive *n* fra *A* til *B* (som er OK ifølge reglerne)
3. Nu flyt de *n-1* skiver fra *C* til *B*



Sammen med rekursion er det løsningen fordi 1. og 3. er samme problemstilling bare for (n-1) skiver

Quicksort (→ Hoare 1959)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Vælg et tilfældigt ***pivot*** element

Partitioner arrayet, så at alle tal lavere eller lige pivot elementet ligger på venstre side og alle større eller lige på højre side

Sorter partitionerne uafhængigt af hinanden **rekursiv**

Så er hele arrayet sorteret

Quicksort

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

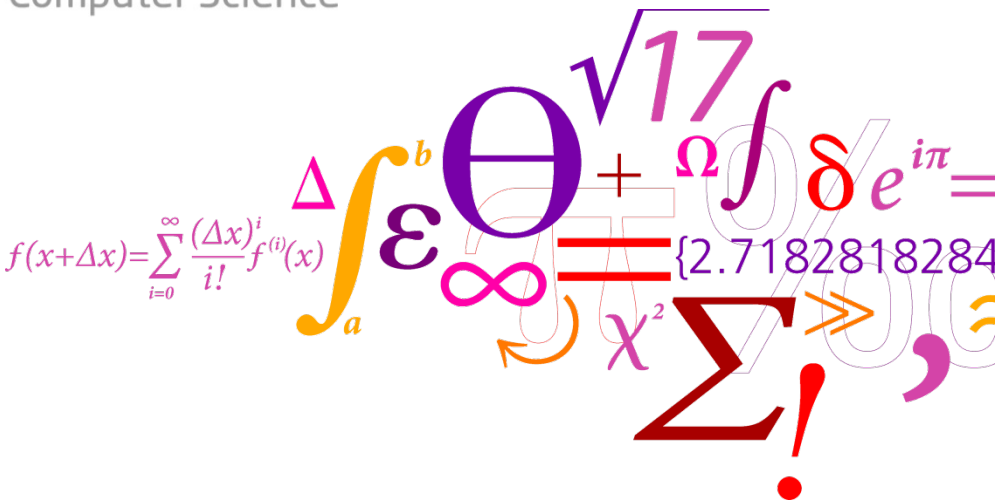
```
private void quicksort(int lower, int upper) {
  if (lower >= upper) {
    return;
  }

  int partition = partition(lower, upper);

  quicksort(lower, partition);

  quicksort(partition+1, upper);
}
```

Bemærk at det er vigtigt at begge partitioner bliver mindre! Begge skal have mindst et element

API Design

Indeholder også tekniske emner som *generiske typer* og *exceptions*!

DTU Compute
Department of Applied Mathematics and Computer Science



Eks.: Stak (interface)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public interface Stack<E> {

        void clear();

        E pop();

        E top();

        void push(E value);

        ...

        int size();

}
```

E er en parameter, som er den type stakken skal senere have: generisk datatype

Eks.: Stak (interface)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public interface Stack<E> {

  void clear();

  E pop() throws IllegalStateException;

  E top() throws IllegalStateException;

  void push(E value) throws IllegalArgumentException;

  ...

  int size();

}
```

Eks.: Stak (implement.)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public class LinkedListStack<E> implements Stack<E> {

    ...

    public E pop() throws IllegalStateException {
        if (top != null) {
            StackElement element = top;
            top = element.next;
            size--;
            return element.value;
        }
        throw new IllegalStateException();
    }

    ...
```

throw betyder at en exception bliver kastet her!

Eks.: Stak (implement.)

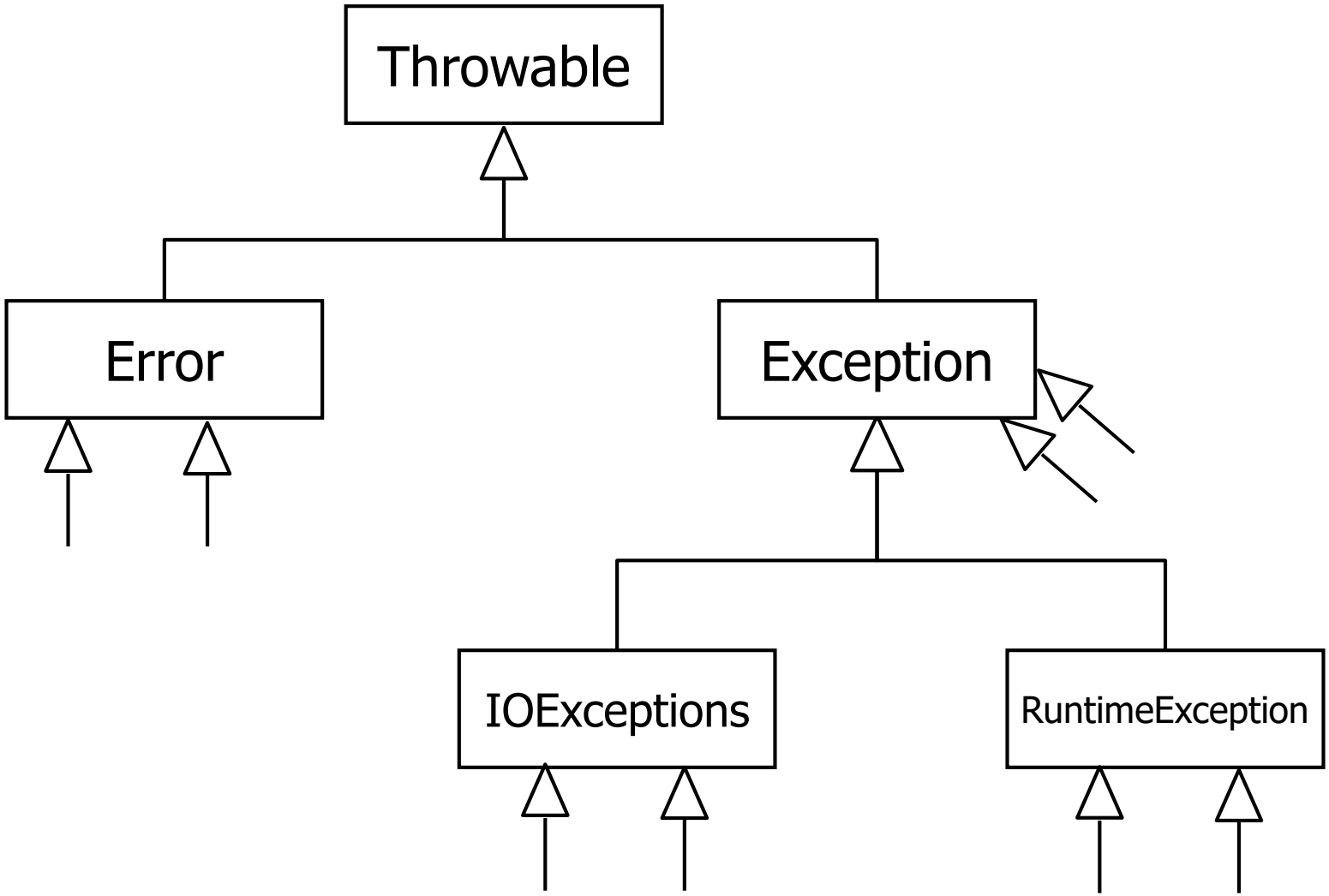
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public void push(E value)
   throws IllegalStateException {
        StackElement newElement =
              new StackElement(value, top);
        size++;
        top = newElement;
}
```

Implementeringen af `push()` kaster ikke en exception, men `new StackElement()` gør; og da `push()` ikke fanger den, ryger den videre op til kalderen af `push()`.

Exception: Typehierarki

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Overvågede exceptions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- er exceptions, hvor den, som kalder metoden (*engl. caller*), kan gøre noget ved

  typisk: forkert brug af metoden (kalde med de rigtige parameter), IOException (prøve igen),

- skal specificeres I metodens deklaration eller fanges I metodens implementering

Uovervågede exceptions

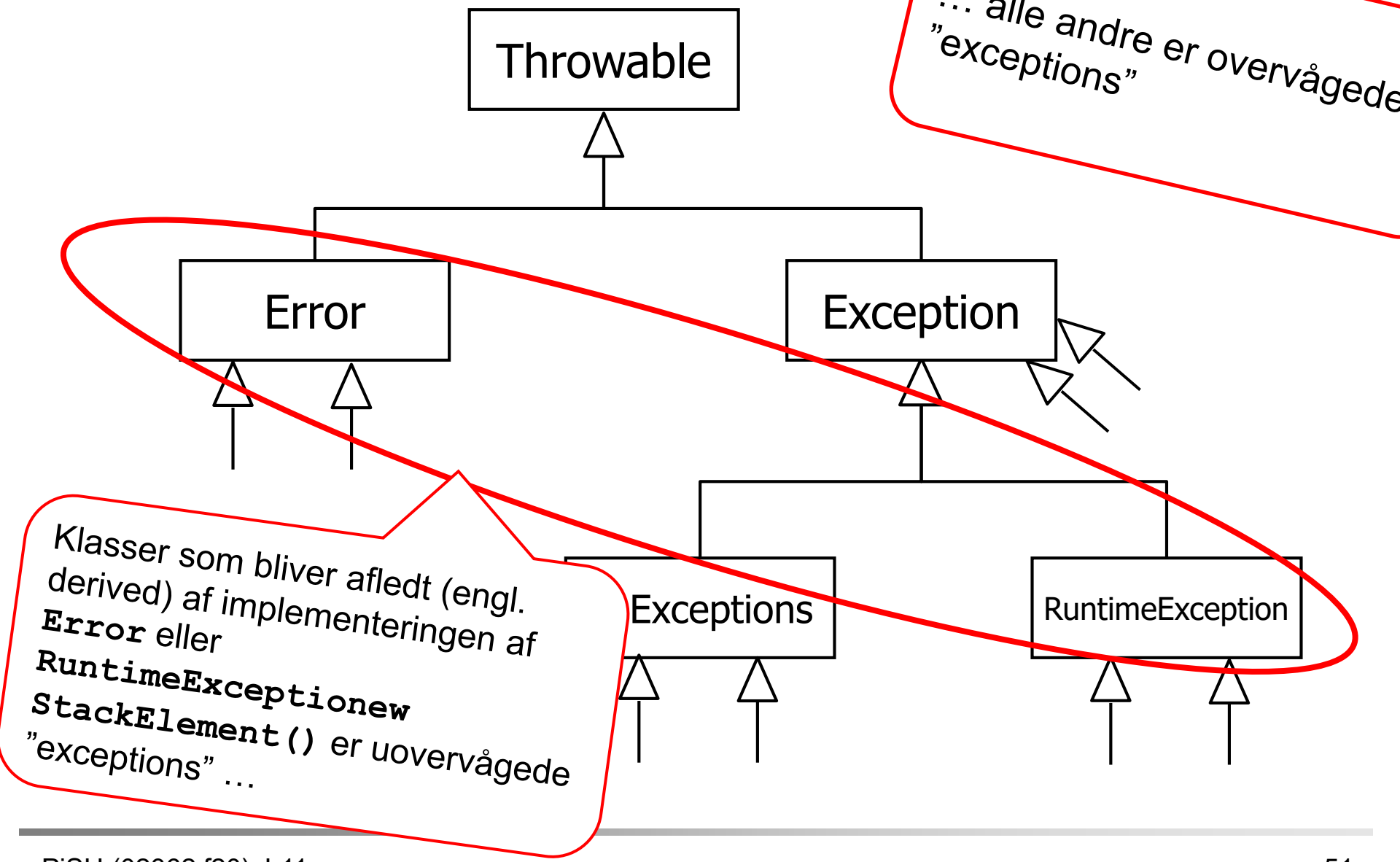
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- er exceptions, hvor den, som kalder metoden (*engl. caller*), ikke kan gøre noget ved
  typisk: programmeringsfejl i implementering
  (NullPointerException, ClassCastException, ArrayIndexOutOfBounds, StackOverflowError, IOError, …)

- er man ikke nødt til at specificere i metodens deklaration, selvom de bliver kastet eller ikke fanget I metodens implementering
- ryger typisk op til main-tråden og stopper hele programmet, når de optræder

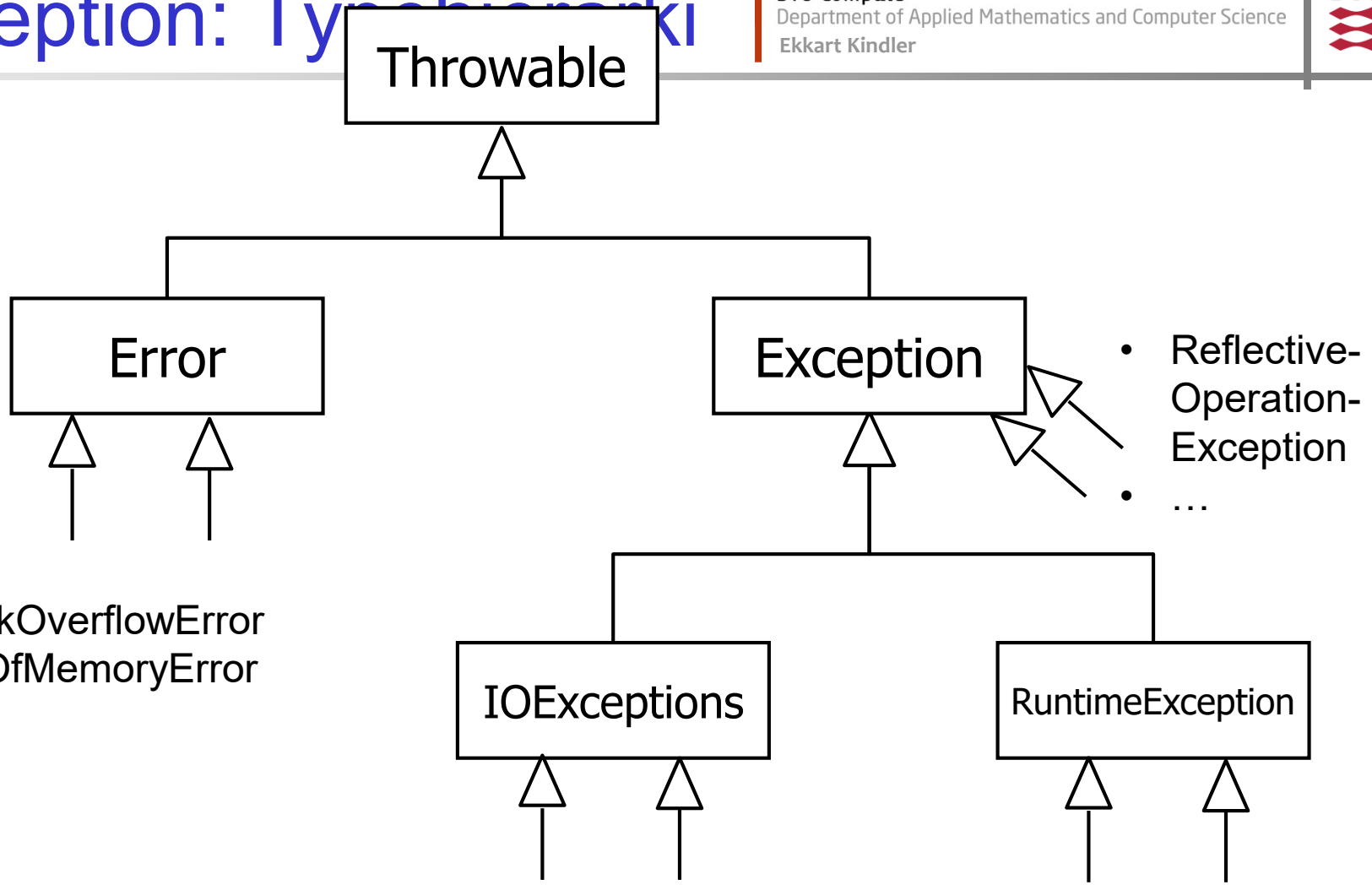
Exception: Typehierarki

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Exception: Typehierarki

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



- StackOverflowError
- OutOfMemoryError
- …

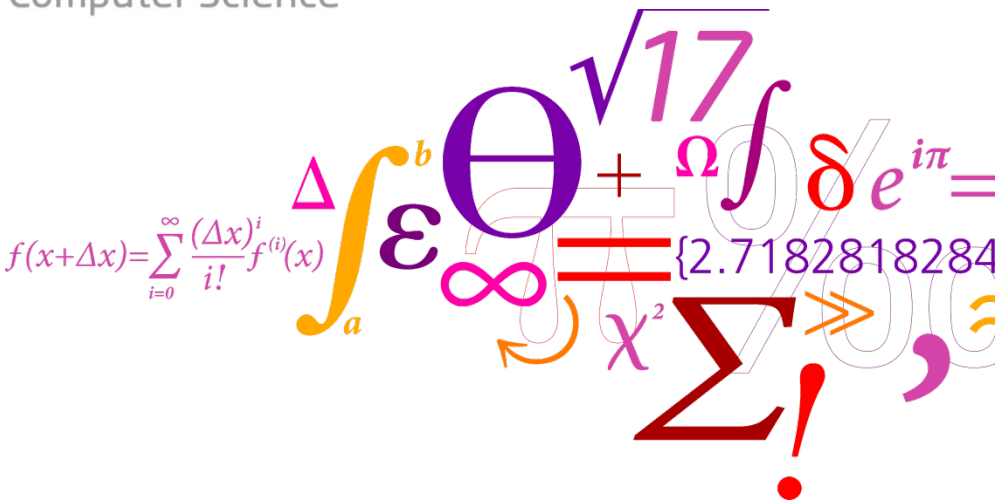
- Reflective-Operation-Exception
- …

- FileNotFoundException
- EOFException
- …

- ClassCastException
- IllegalArgumentException
- NullPointerException
- IndexOutOfBoundsException
- …

Analyse og konceptuelle modeller

DTU Compute
Department of Applied Mathematics and Computer Science



Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Inden man kan implementere et stykke software er man nødt til af finde ud af **hvad** selve software skal gøre og formulere det fra brugerens synsvinkel:

- Hvilke brugere er der?
- Hvilke begreber (engl. concepts) spiller en rolle
- Hvordan hænger begreberne sammen?
- Hvilke funktioner (use-cases/aktiviteter) er der? Hvad indebærer de og hvornår kan man gennemføre dem?

I vores projekt er det mest af alt: RoboRally-spillets regler

Hvordan gør man det

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

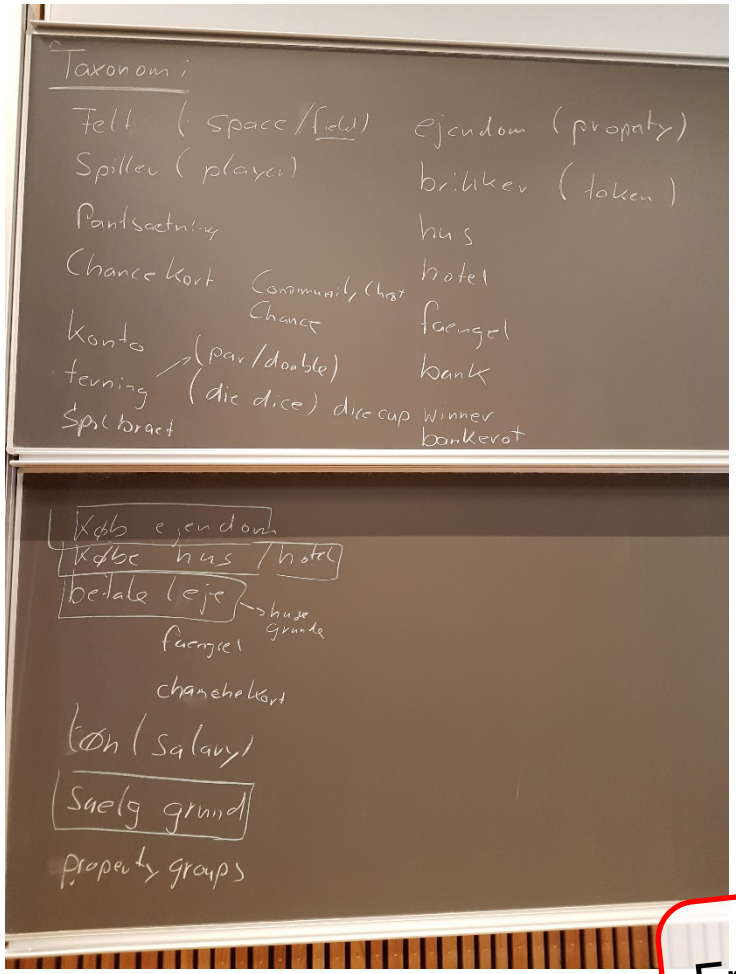
- Find skriftlig dokumentation om området (domænet) og skriv det ud
- Marker alle begreber som synes relevante
- Tal med nogle eksperter fra domænet

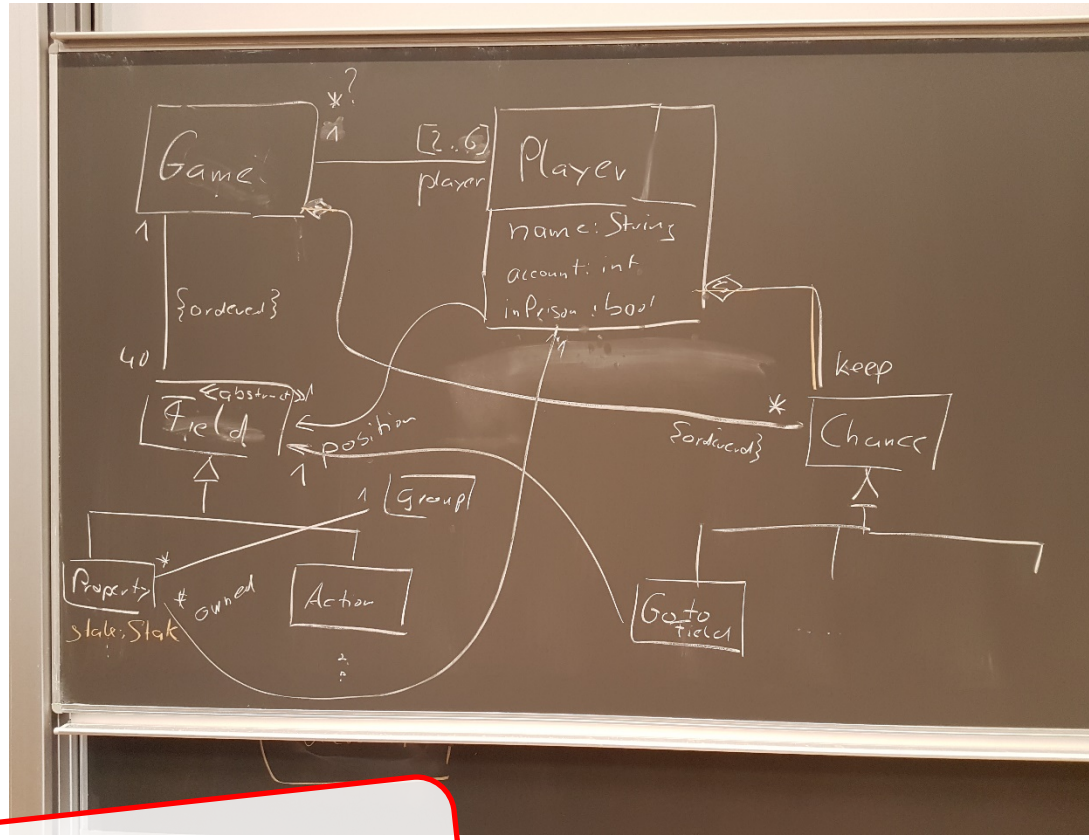
→ **Taksonomi**: liste af begreber (ord)

→ **Glossar**: Beskrivelse af begreber deres egenskaber og hvordan de forholder sig til hinanden

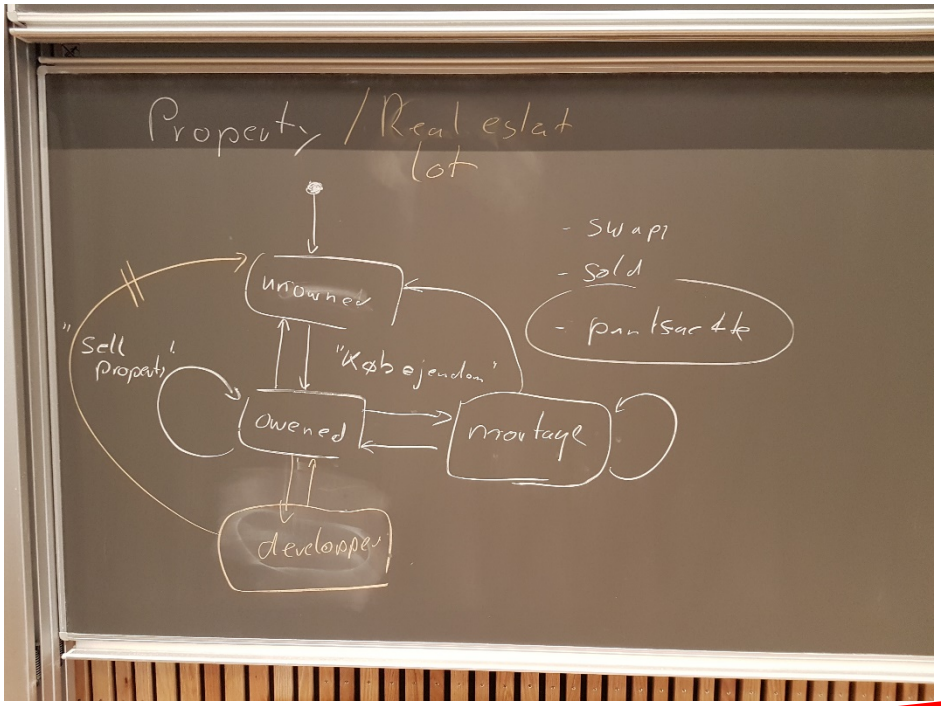
→ **Domænemodel**: Begreber, deres attributter og deres relation repræsenteret som klassediagram

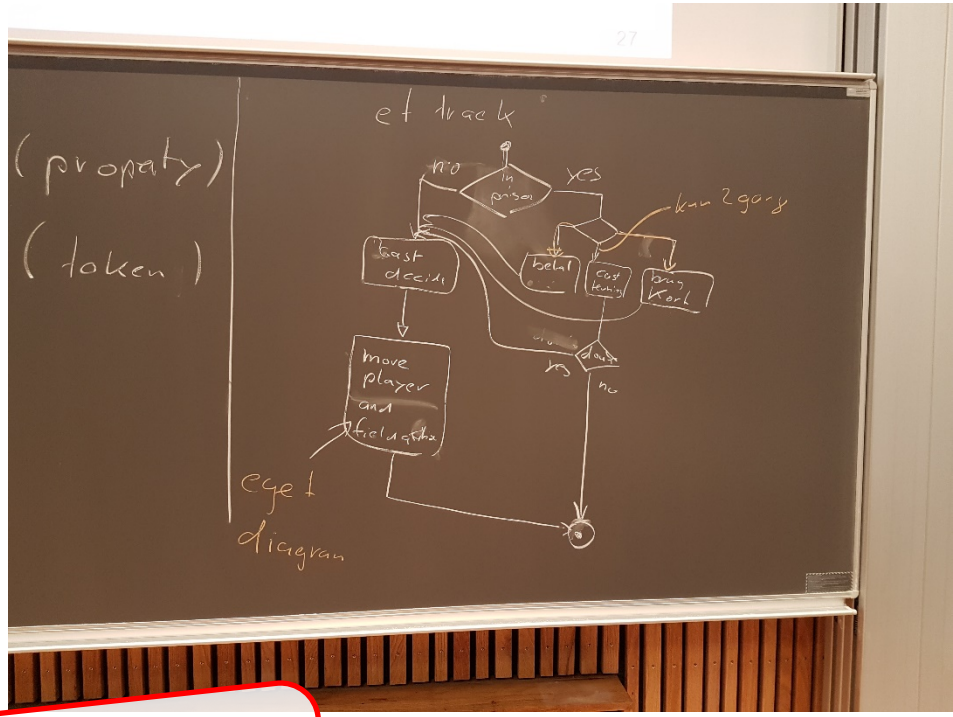
Dette klassediagram har ikke noget med programmering at gøre (endnu). Tænk kun på begreber og deres relation!





Fra sidste år:
Monopoly/Matador!



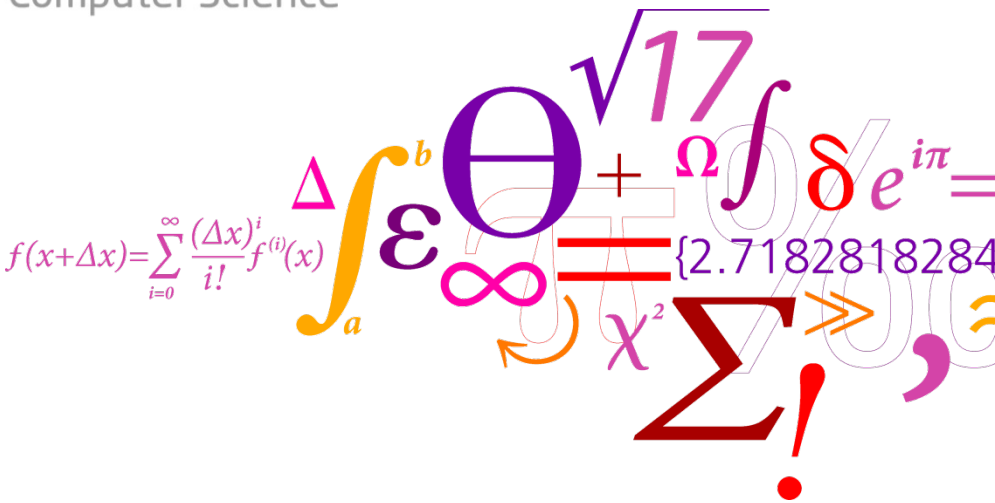


Fra sidste år:
Monopoly/Matador!

(Software) Design

Bare for at være sikker på at vi ikke taler om GUI layout design.

DTU Compute
Department of Applied Mathematics and Computer Science



1. Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

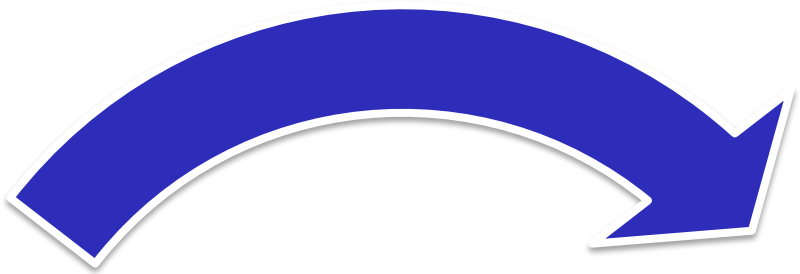
Analysen drejer sig **kun** om **HVAD** (engl. **WHAT**) softwaren skal gøre (ud fra brugerens synsvinkel)

Analysen siger **ikke noget** om **HVORDAN** (engl. **HOW**) softwaren skal realiseres og implementeres

Co-evolution

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

"**Hvad**" skal softwaren gøre?



WHAT HOW

"**Hvordan**" er softwaren realiseret?

Design patterns

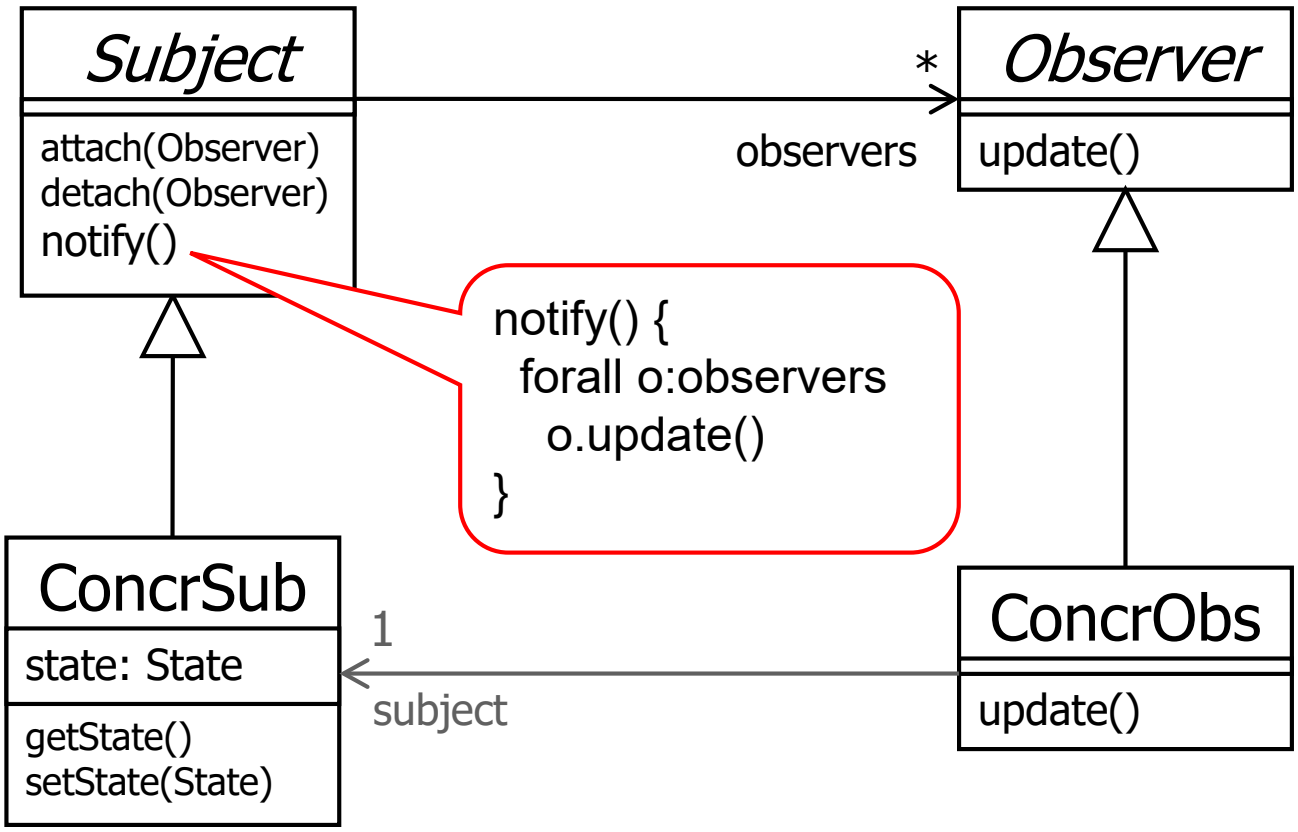
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Oprindeligt, kom begrebet fra: Alexander et al. 1977.

Design patterns (i softwareudvikling) er den destillerede erfaring af softwareudviklings-eksperter hvordan man løser standardproblemer I software-design.

Eksempel: Observer

Structure



Model View Controller (MVC)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Hvis man gøre det rigtigt er domæne modeller (**model**) og deres implementeringer en fundamental del af den udviklede software

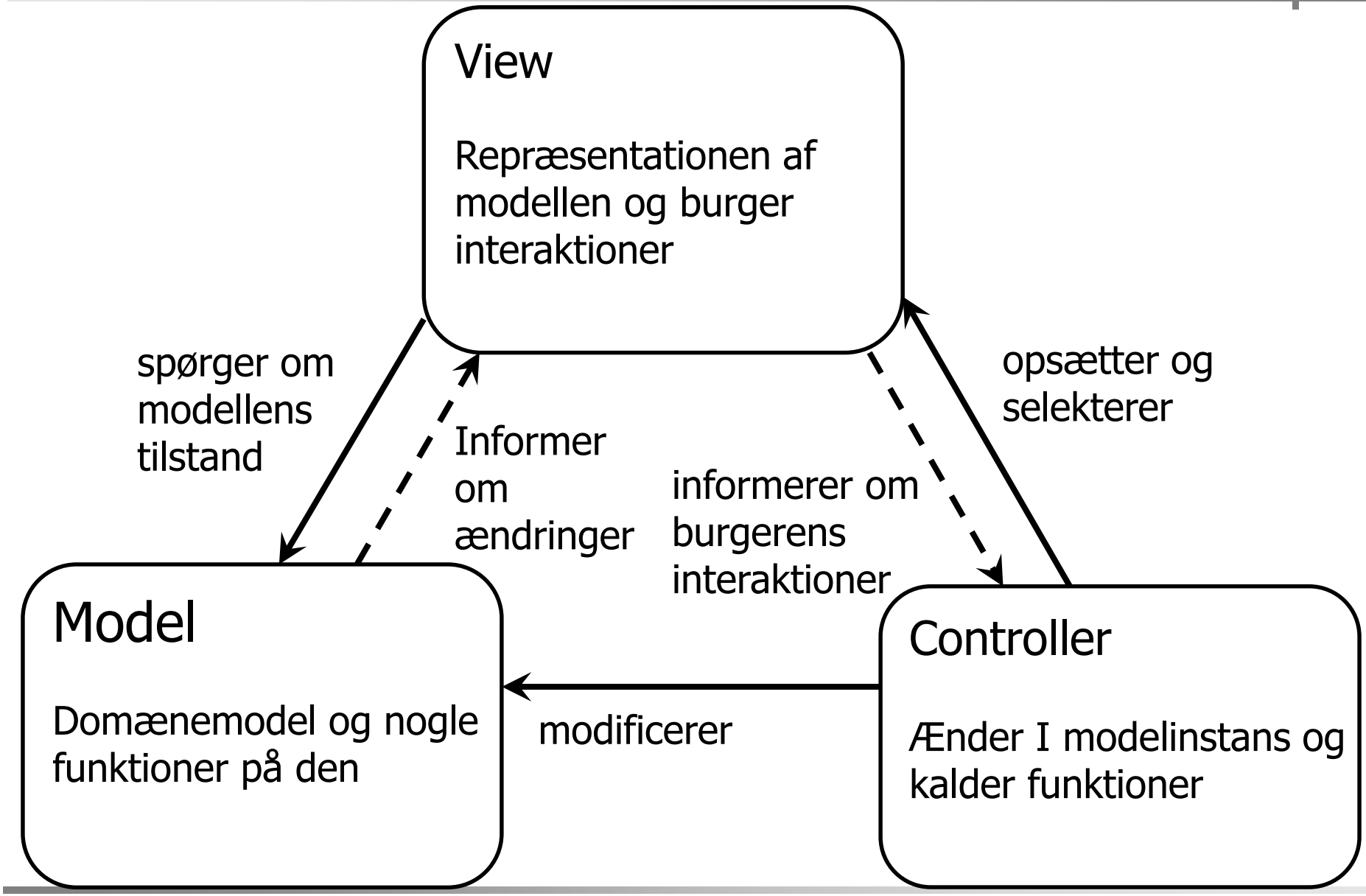
Men der ville være yderlige dele software:

- Delen som viser modellens information til brugeren (**view**)
- Delen som koordinerer med brugeren og implementerer logikken bagved (**controller**)

Bemærk: Disse dele af softwaren kan også modelleres, men de skal ikke forveksles med model forstand af domæne modellen (**hvad**). De kaldes for "software modeller" (**hvordan**) eller "design modeller".

MVC

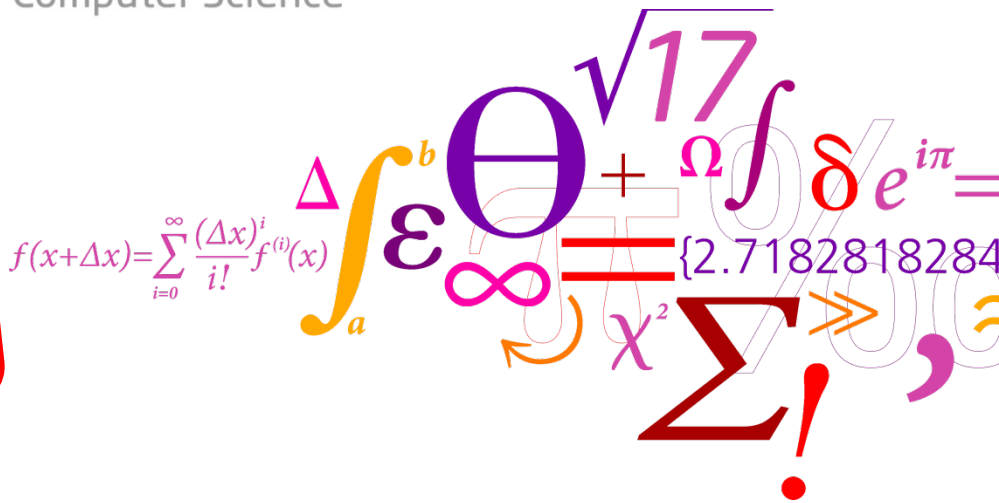
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Læse og skrive filer, Gson

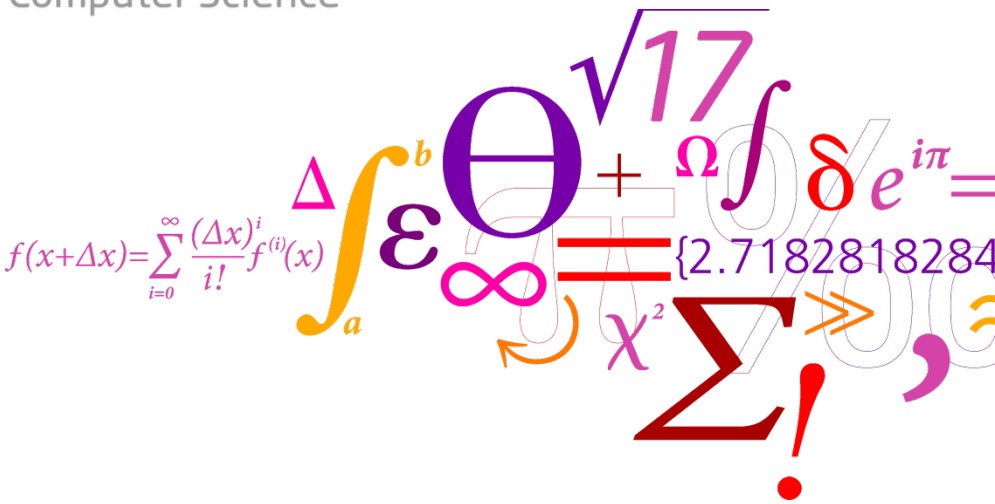
DTU Compute
Department of Applied Mathematics and Computer Science

Uge 7, video PiSU 07.4



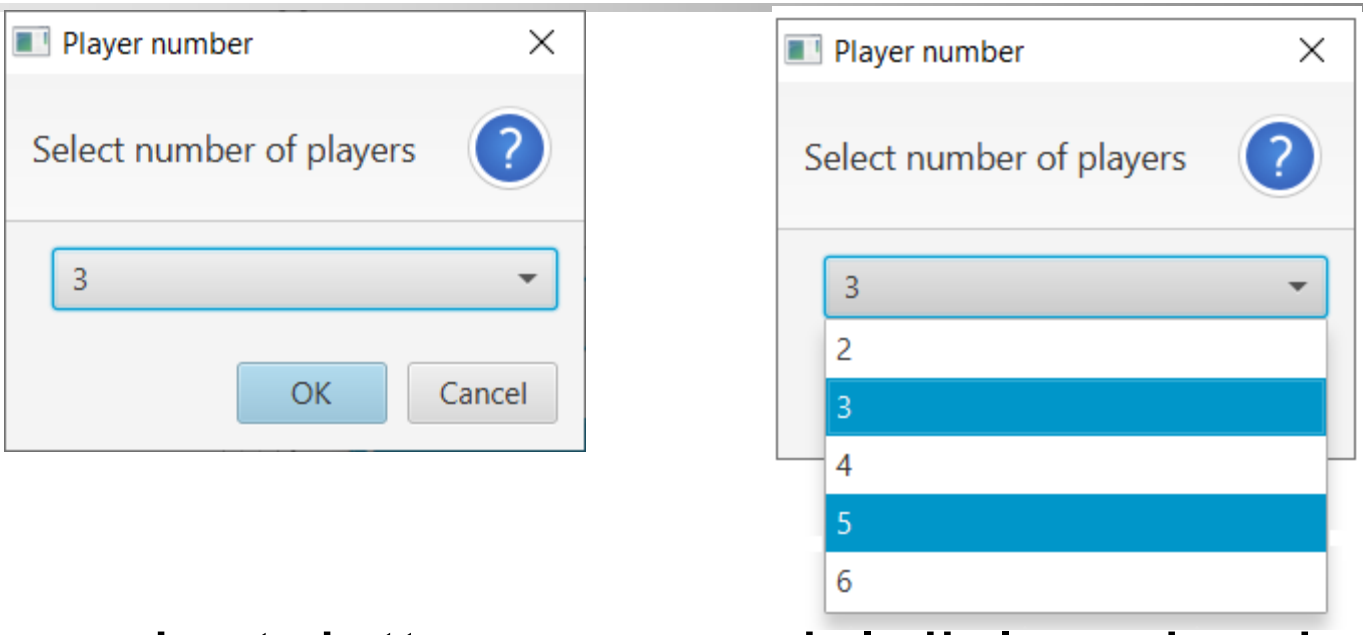
Dialoger og inputvalidering

DTU Compute
Department of Applied Mathematics and Computer Science



Eksempel: ChoiceDialog

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



- Bemærk at dette er en modal dialog, dvs. brugeren kan først interagere med resten af GUIen (eller forældre-dialog), når denne her dialog er afsluttet.

I `GameController` skal I helst ikke bruge (modale) dialoger! Se opgave V3, hvordan man kan undgå dette!

Regulære Udtryk

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

I nogle situationer vil man sikre sig, at brugeren indtaster informationer, som overholder en hvis struktur:

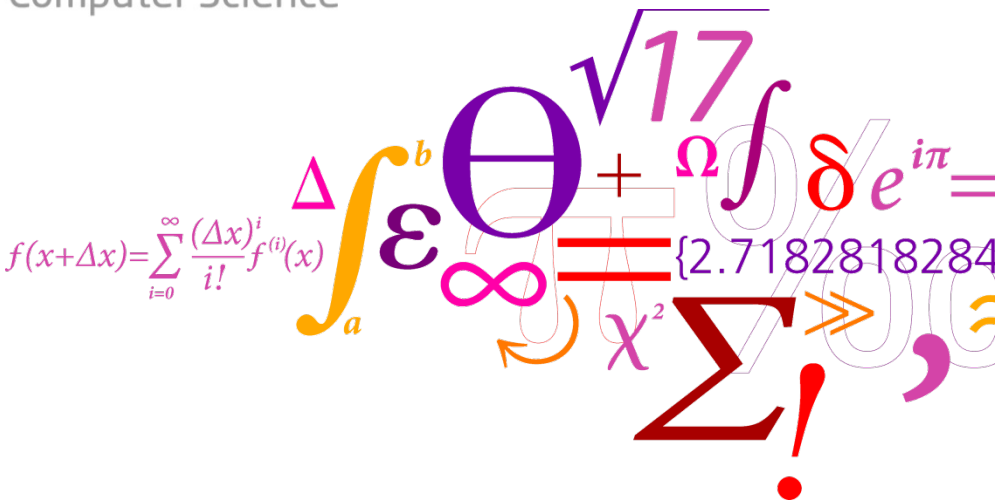
- Adresser
- Email-addresser
- Telefonnumre
- Datoer

Regulære udtryk er et meget gammelt koncept i datalogi – meget ældre en Java. Men her bruger vi især notationen fra Java (*,+, ?, | er i generel brug også i andre kontekster)
→ Uge 8 video PiSU L08

Mange af disse strukturer kan defineres med såkaldte **regulære udtryk**

DAL & DAO (V4/A4)

DTU Compute
Department of Applied Mathematics and Computer Science



```
public interface IRepository {

  boolean createGame(Board game);

  boolean updateGame(Board game);

  Board loadGame(int id);

  List<GameInDB> getGameIds();

}
```

- Prepared statements
- Brug af opdaterbare ResultSets
- Transaktioner:
  - `connection.setAutoCommit(false)`
  - `connection.commit()  // explicit commit`
  - `connection.rollback() // explicit rollback`