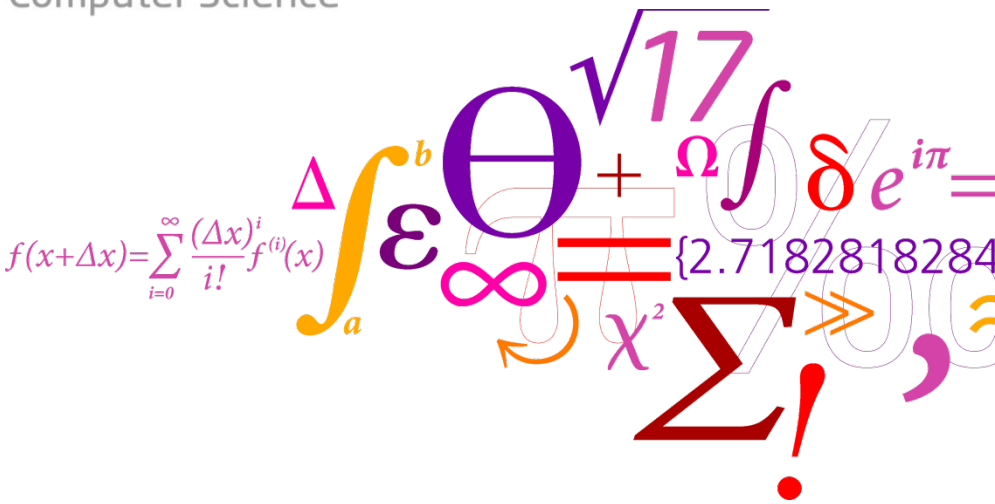
Projekt i software-udvikling (02362)
Forår 2022

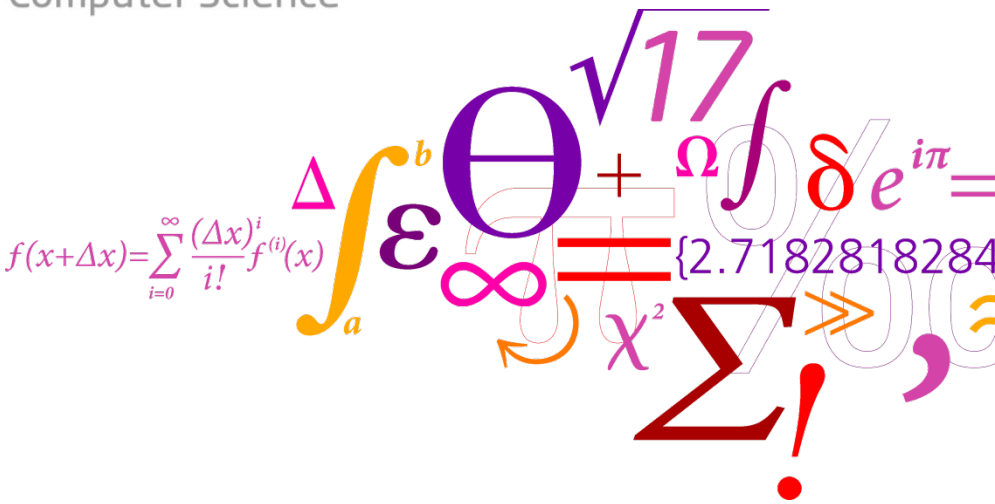
Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



I. Introduktion

DTU Compute
Department of Applied Mathematics and Computer Science



Kursus: Kort sagt

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

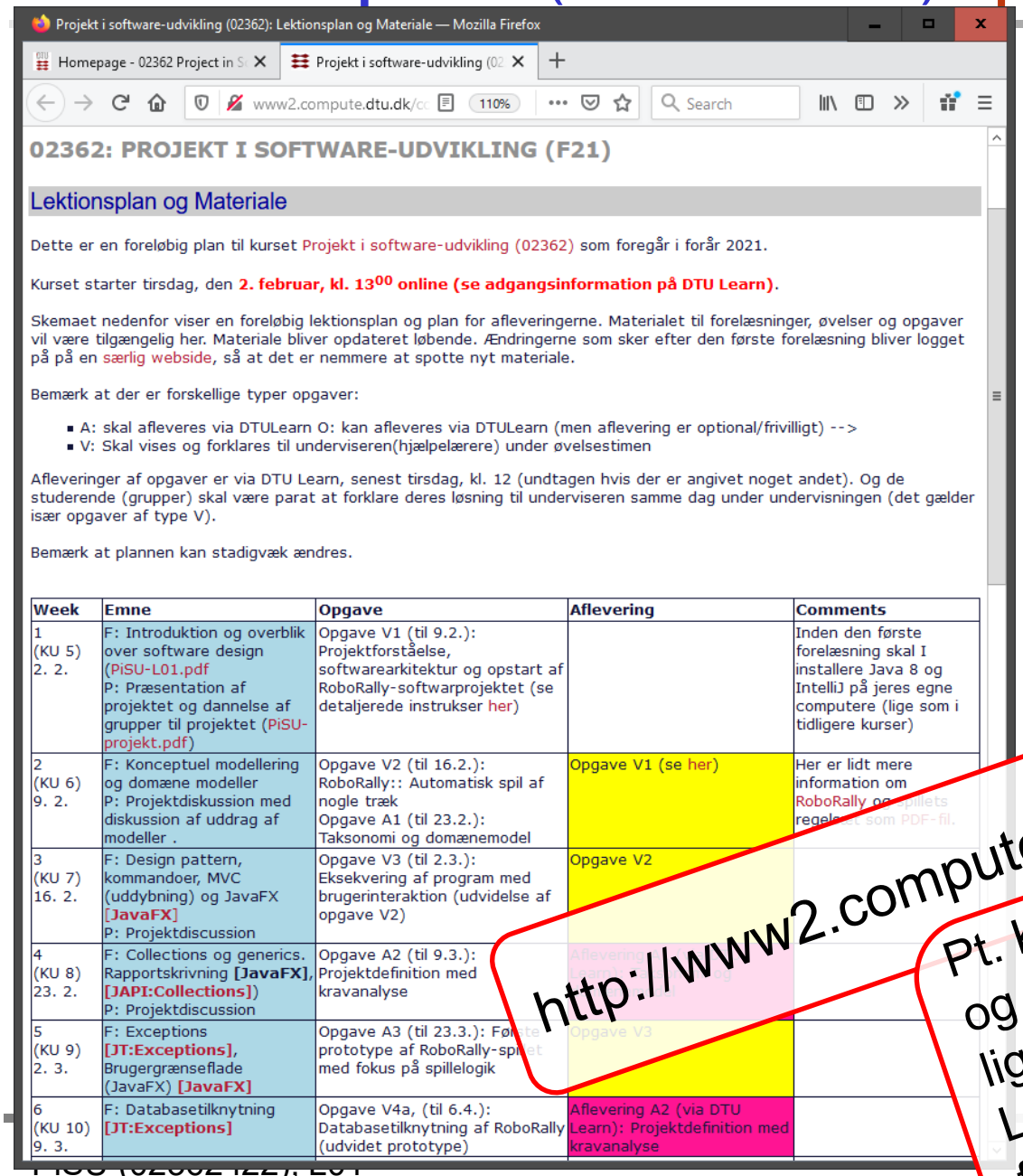
Indtil nu (1. semester):

- **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

Lektionsplan (fra sidste år)



Projekt i software-udvikling (02362): Lektionsplan og Materiale — Mozilla Firefox

Homepage - 02362 Project in S | Projekt i software-udvikling (02

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

02362: PROJEKT I SOFTWARE-UDVIKLING (F21)

Lektionsplan og Materiale

Dette er en foreløbig plan til kurset Projekt i software-udvikling (02362) som foregår i forår 2021.

Kurset starter tirsdag, den **2. februar, kl. 13^{00} online (se adgangsinformation på DTU Learn)**.

Skemaet nedenfor viser en foreløbig lektionsplan og plan for afleveringerne. Materialet til forelæsninger, øvelser og opgaver vil være tilgængelig her. Materiale bliver opdateret løbende. Ændringerne som sker efter den første forelæsning bliver logget på på en særlig webside, så at det er nemmere at spotte nyt materiale.

Bemærk at der er forskellige typer opgaver:

- A: skal afleveres via DTULearn O: kan afleveres via DTULearn (men aflevering er optional/frivilligt) -->
- V: Skal vises og forklares til underviseren(hjælpelærere) under øvelsestimen

Afleveringer af opgaver er via DTU Learn, senest tirsdag, kl. 12 (undtagen hvis der er angivet noget andet). Og de studerende (grupper) skal være parat at forklare deres løsning til underviseren samme dag under undervisningen (det gælder især opgaver af type V).

Bemærk at plannen kan stadigvæk ændres.

Week	Emne	Opgave	Aflevering	Comments
1 (KU 5) 2. 2.	F: Introduktion og overblik over software design (PiSU-L01.pdf P: Præsentation af projektet og dannelse af grupper til projektet (PiSU-projekt.pdf)	Opgave V1 (til 9.2.): Projektforståelse, softwarearkitektur og opstart af RoboRally-softwarprojektet (se detaljerede instrukser her)		Inden den første forelæsning skal I installere Java 8 og IntelliJ på jeres egne computere (lige som i tidligere kurser)
2 (KU 6) 9. 2.	F: Konceptuel modellering og domæne modeller P: Projektdiskussion med diskussion af uddrag af modeller .	Opgave V2 (til 16.2.): RoboRally:: Automatisk spil af nogle træk Opgave A1 (til 23.2.): Taksonomi og domænemodel	Opgave V1 (se her)	Her er lidt mere information om RoboRally og spillets regelsæt som PDF-fil.
3 (KU 7) 16. 2.	F: Design pattern, kommandoer, MVC (uddybning) og JavaFX [JavaFX] P: Projektdiscussion	Opgave V3 (til 2.3.): Eksekvering af program med brugerinteraktion (udvidelse af opgave V2)	Opgave V2	
4 (KU 8) 23. 2.	F: Collections og generics. Rapportskrivning [JavaFX], [JAPI:Collections]) P: Projektdiscussion	Opgave A2 (til 9.3.): Projektdefinition med kravanalyse	Aflevering [illegible]	
5 (KU 9) 2. 3.	F: Exceptions [JT:Exceptions], Brugergrænseflade (JavaFX) [JavaFX]	Opgave A3 (til 23.3.): Første prototype af RoboRally-spillet med fokus på spillelogik	Opgave V3	
6 (KU 10) 9. 3.	F: Databasetilknytning [JT:Exceptions]	Opgave V4a, (til 6.4.): Databasetilknytning af RoboRally (udvidet prototype)	Aflevering A2 (via DTU Learn): Projektdefinition med kravanalyse	

Der er videoer til forelæsninger og nogle demoer til hele kurset fra sidste år. Dem må i gerne, forløbet i år følger nogenlunde det fra sidste år.
Men koden (og nogle andet materiale senere), som I får udleveret i starten har ændret sig lidt.

http://www2.compute.dtu.dk/courses/02362/f22

Pt. Kan jeg desværre ikke kreere og opdatere websiderne; derfor ligger alt materiale kun på DTU Learn (også en plan Plan-02362-f22.v1.pdf). Vi følger næsten skema som sidste år.

Kursus i dag

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Forelæsnig
  - Introduktion og motivation
  - Software design (overblik)
- Projektdiskussion
- Øvelser: Opgave V1

Senere er der også nogle afleveringer (via DTU Learn med faste afleverings-datoer): **A-opgaver**

Bemærk at opgaven ikke skal afleveres, men **vises til underviseren** næste gang: **V-opgave**!

Kursets opbygning

- Forelæsninger
- Øvelser
  - på DTU og
  - hjemmearbejde
- Opgaver (afleveringer)

Bemærk at kurset 02362 er 5 ECTS point. Det betyder at der forventens ca. **5 timer arbejde per uge ved siden af** undervisningstimerne!

At blive en god softwareudvikler kræver erfaring!

- Projektarbejde
  - Vi bruger færdigheder fra Database-kursus (02327) i dette projekt (JDBC, SQL, …)

Emner

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

Vi starter med projektrelaterede opgaver fra uge 1.

Nogle ”basale emner” kommer senere eller ved siden af!

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)

+ nogle ad hoc emner: sig til, hvis I mangler noget eller har brug for nogle detaljer!

Opfriskning (eksempel)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
class Pos {
   int x;
   int y;
}
class PosName {
   String name;
   Pos pos;
   public PosName(String name, Pos pos) {
      this.name = name;
      this.pos = pos;
} }
```

Denne kode er ikke pæn!

Kun et teknisk eksempel, som opfrisking af nogle begreber og tænkemåder!

Opfriskning (eksempel)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
class Test {
public static void main(String[] args){
    Pos pos1 = new Pos();
    pos1.x = 1;
    pos1.y = 2;

    Pos pos2 = pos1;
    pos2.x = 3;
    pos2.y = 4;
```

Hvilke værdier har
`pos1.x`, `pos2.x`,
`pos1.y` og `pos2.y`
når programmet når her?
Og hvorfor?

→ Se diskussion på tavlen!

Opfriskning (eksempel)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
PosName posName1 = new PosName("position 1", pos1);
PosName posName2 = new PosName("position 2", pos2);
```

Hvordan ville værdierne se ud her?

```
Pos[] posArray = new Pos[10];
for (int i = 0; i < posArray.length; i++) {
  posArray[i].x = i;
  posArray[i].y = i;
}
```

Hvordan ville værdierne se ud her?

```
posArray[1] = pos1;
posArray[2] = pos2;
```

Og hvorfor?

- Tænk over det
- Dan jer en "mentalt model" hvad der foregår
- Diskussion!

Opfriskning (eksempel)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
PosName posName1 = new PosName("position 1", pos1);
PosName posName2 = new PosName("position 2", pos2);


Pos[] posArray = new Pos[10];
for (int i = 0; i < posArray.length; i++) {
  posArray[i] = new Pos();
  posArray[i].x = i;
  posArray[i].y = i;
}

posArray[1] = pos1;
posArray[2] = pos2;
```

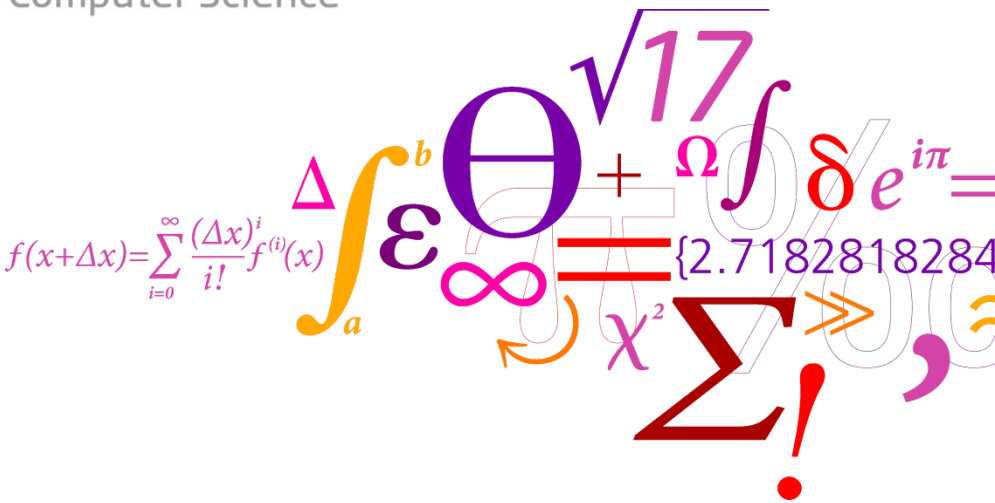
Hvordan ville værdierne se ud her?

→ Tavlediskussion

II. (Software) Design (Del 1)

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

DTU Compute
Department of Applied Mathematics and Computer Science



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

Design

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- **Design** (og implementering) taler om **HVORDAN** (**HOW**) software skal realiseres
- Design taler om det overordnede struktur af softwaren (dens **arkitektur**): hovedkomponenterne, deres interfaces og hvordan de spiller sammen
- Det kaldes på engelsk også for ”programming in the large”

Her kan vi kun tale om nogle enkelte aspekter og principper af design og arkitektur! Og vi kommer tilbage til det hele tiden – også rent praktisk i selve projektet.

Ekskursion: CDIO

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Conceive — Analyse

Design — Design

Implement — ”Programmering” (lidt enkelt sagt)

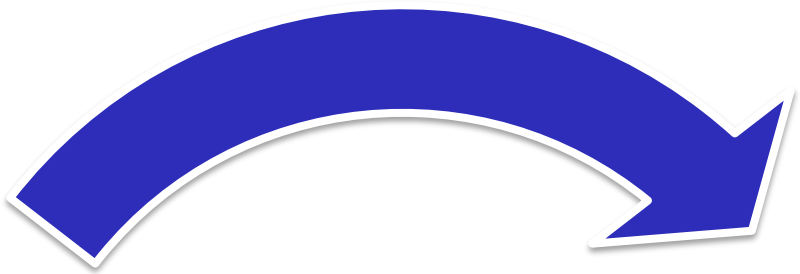
Operate

Co-evolution

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

"**Hvad**" skal softwaren gøre?

"**Hvordan**" er softwaren realiseret?



WHAT

HOW

*Realiteten er mere kompliceret, men at adskille "**what**" og "**how**" i jeres hoved er et første.*

2. Design patterns

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.

Freeman & Freeman kalder det “experience reuse” (genbrug af erfaring)!

Design Patterns i SU:

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

De kaldes ofte for “Gang of Four” (GoF / Go4).

- Gamma, Helm, Johnson, Vlissides: Design Patterns. Addison-Wesley 1995.
- Eric Freeman, Elisabeth Freeman: Head First Design Patterns. O’Reilly 2004 [FF]
- …

Eksempel: Observer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Name and classification

Observer, object, behavioural

Vi følger nogenlunde GoF

Intent

”Define a one-to-many dependency between objects so that when an object changes all its dependents are are notified and updated automatically” [GoF].

Also know as

Dependents, Publish-Subscribe, Listener

Eksempel: Observer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Motivation

[...] maintain consistency between related objects without introducing tight coupling (which increases reusability) [...]

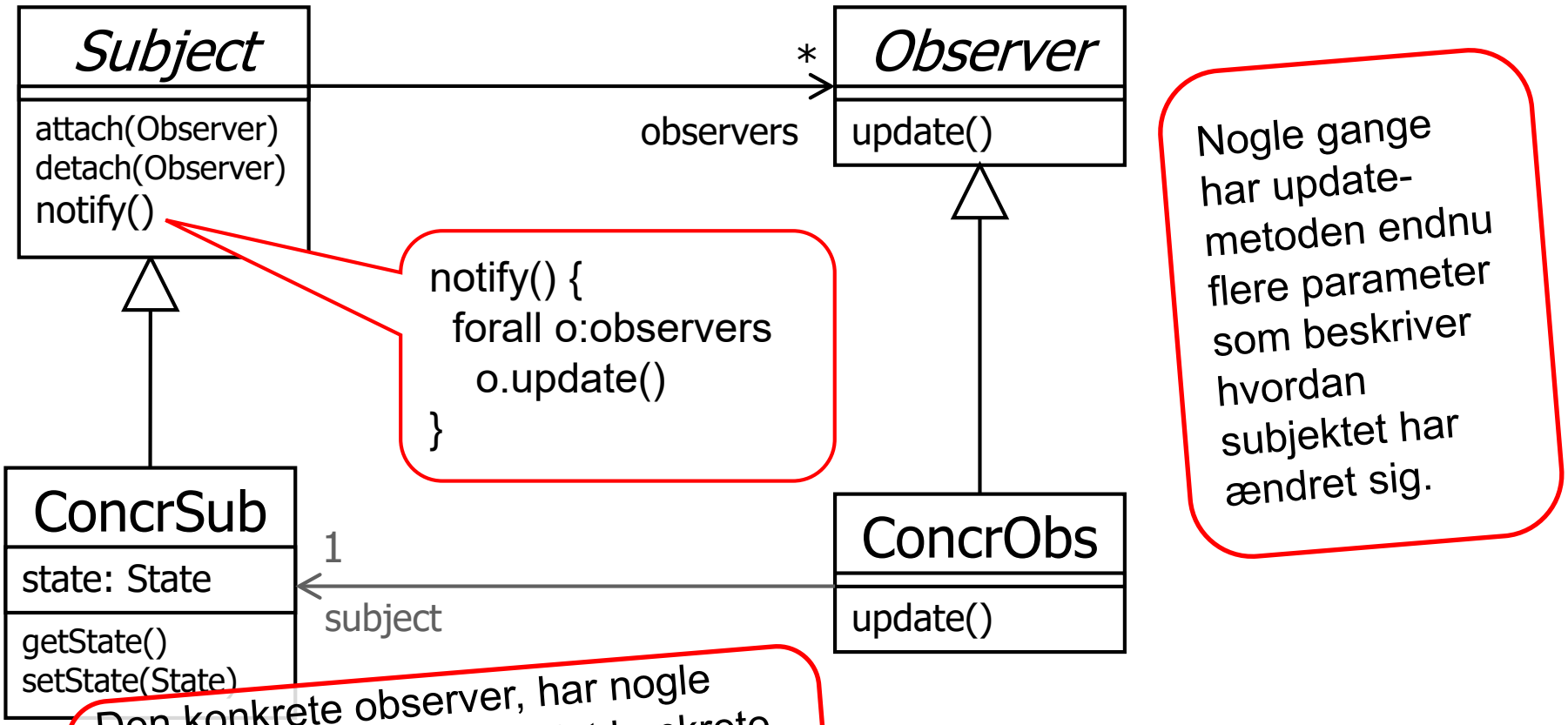
Typical Example

... update views when the underlying model changes ...

Se også MVC senere.

Eksempel: Observer

Structure



Eksempel: Observer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Participants (see structure)

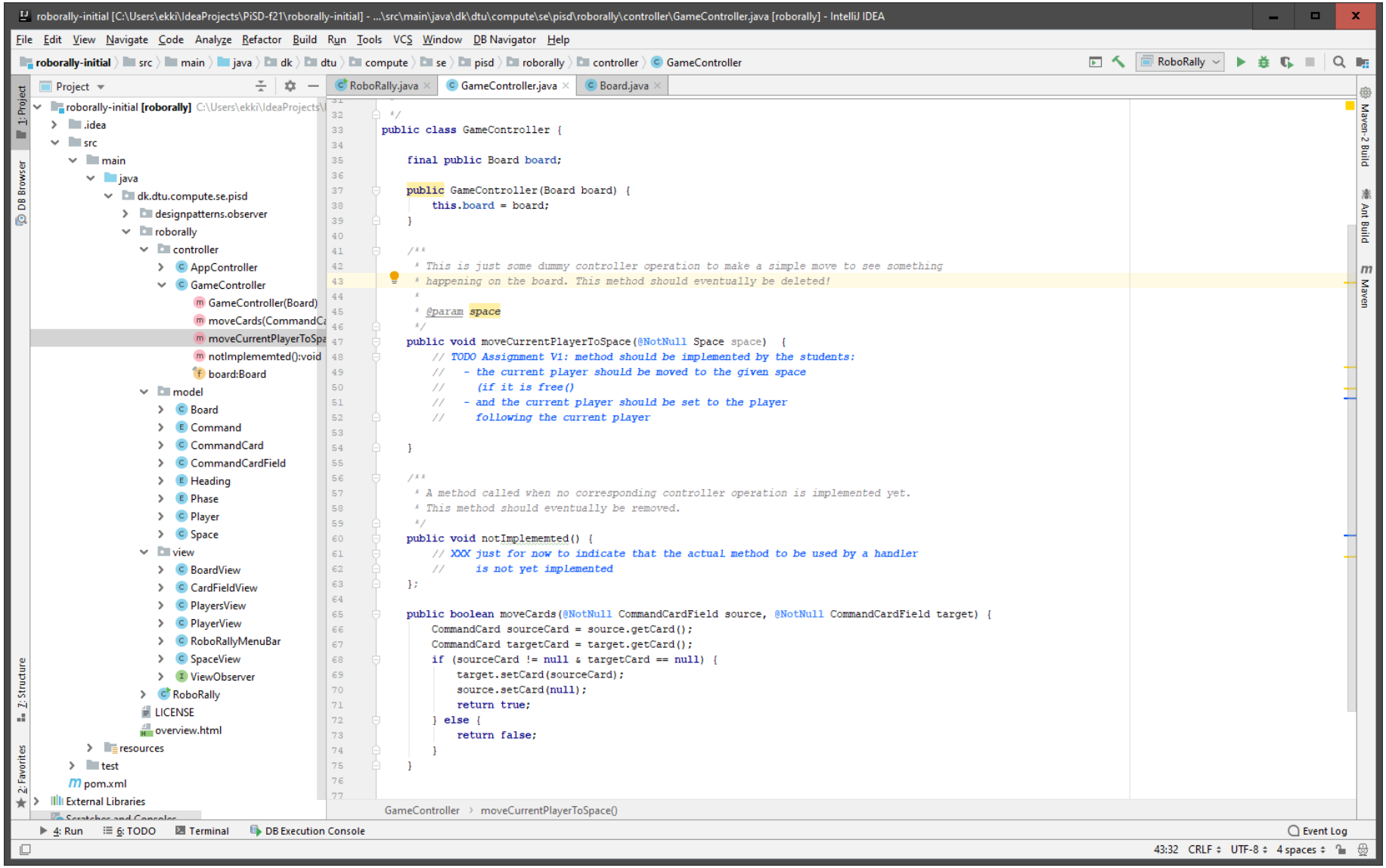
~ GoF

- **Subject**
  - knows its observers
  - provides an interface for attaching and detaching Observer objects
- **Observer**
  - defines the updating interface for being notified
- **ConreteSubject**
  - stores the state (of interest)
  - sends notifications
- **ConreteObserver**
  - Implements the Observer‘s updating interface to keep its state consistent

→ diskussion i selve RoboRally-implementering (senere under diskussion af selve software)

Diskussion: Eksempel

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Fortsættes …

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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 i software:

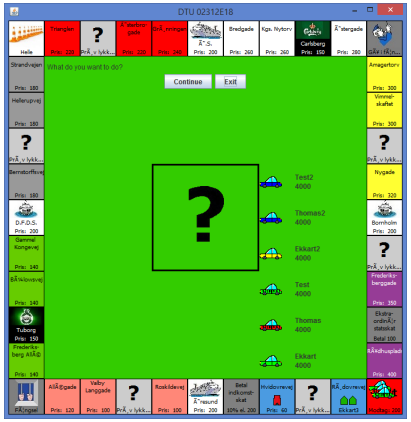
- Delen som viser modellens information til brugeren (**view**)
- Delen som koordinerer med brugeren og implementerer logikken bagved (**controller**); ”implementerer af aktivitetsdiagrammer”

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

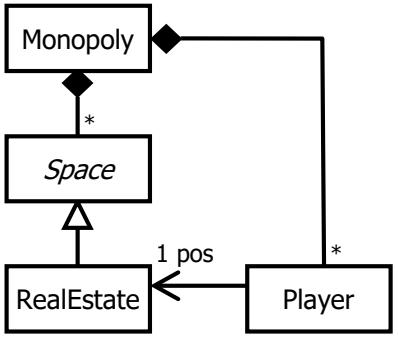
Model View Controller (MVC)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

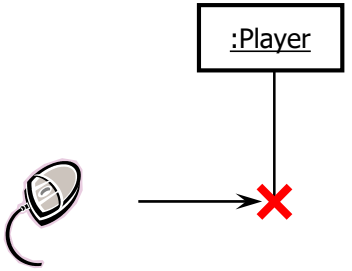
View



Model

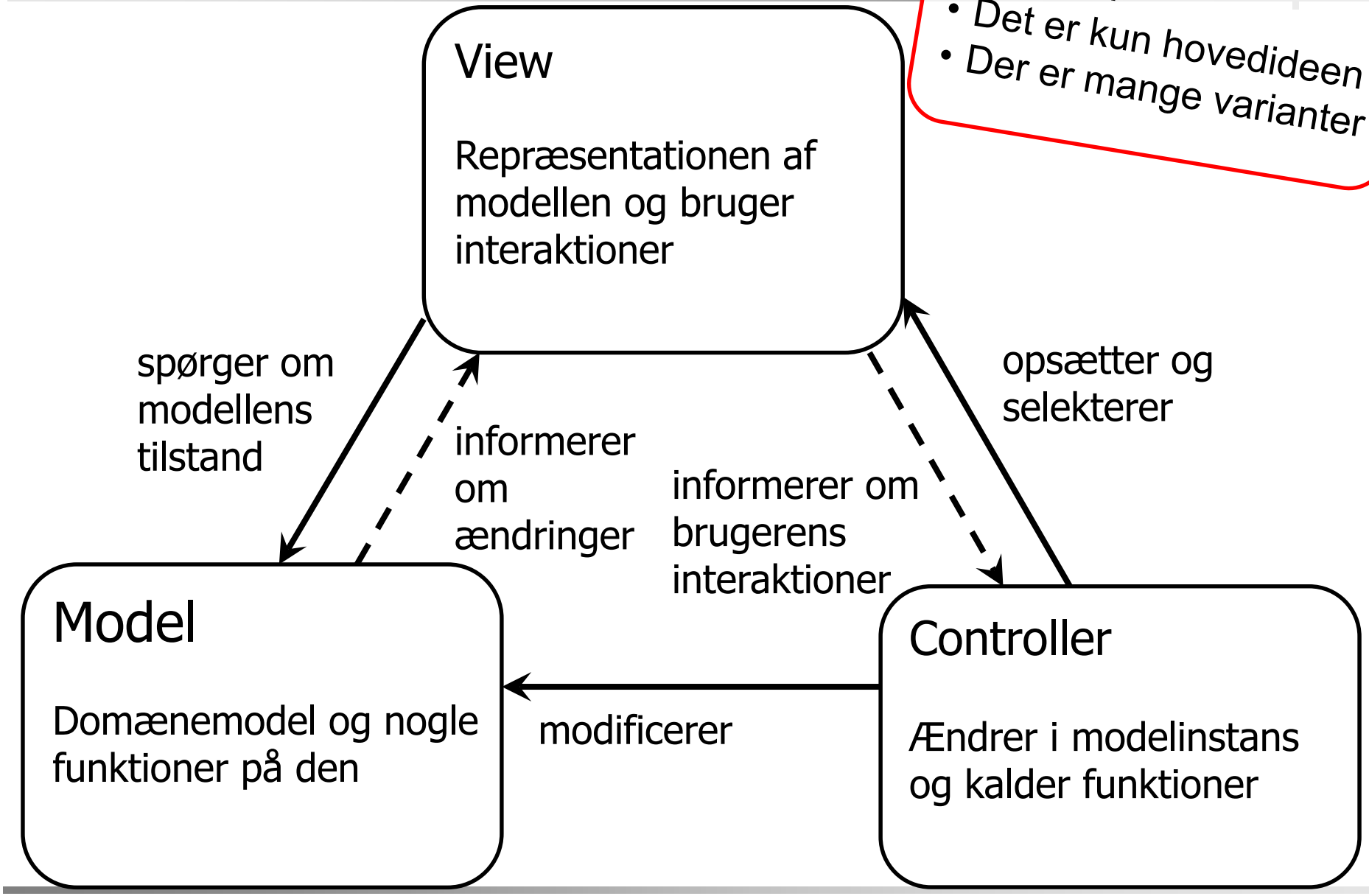


Controller



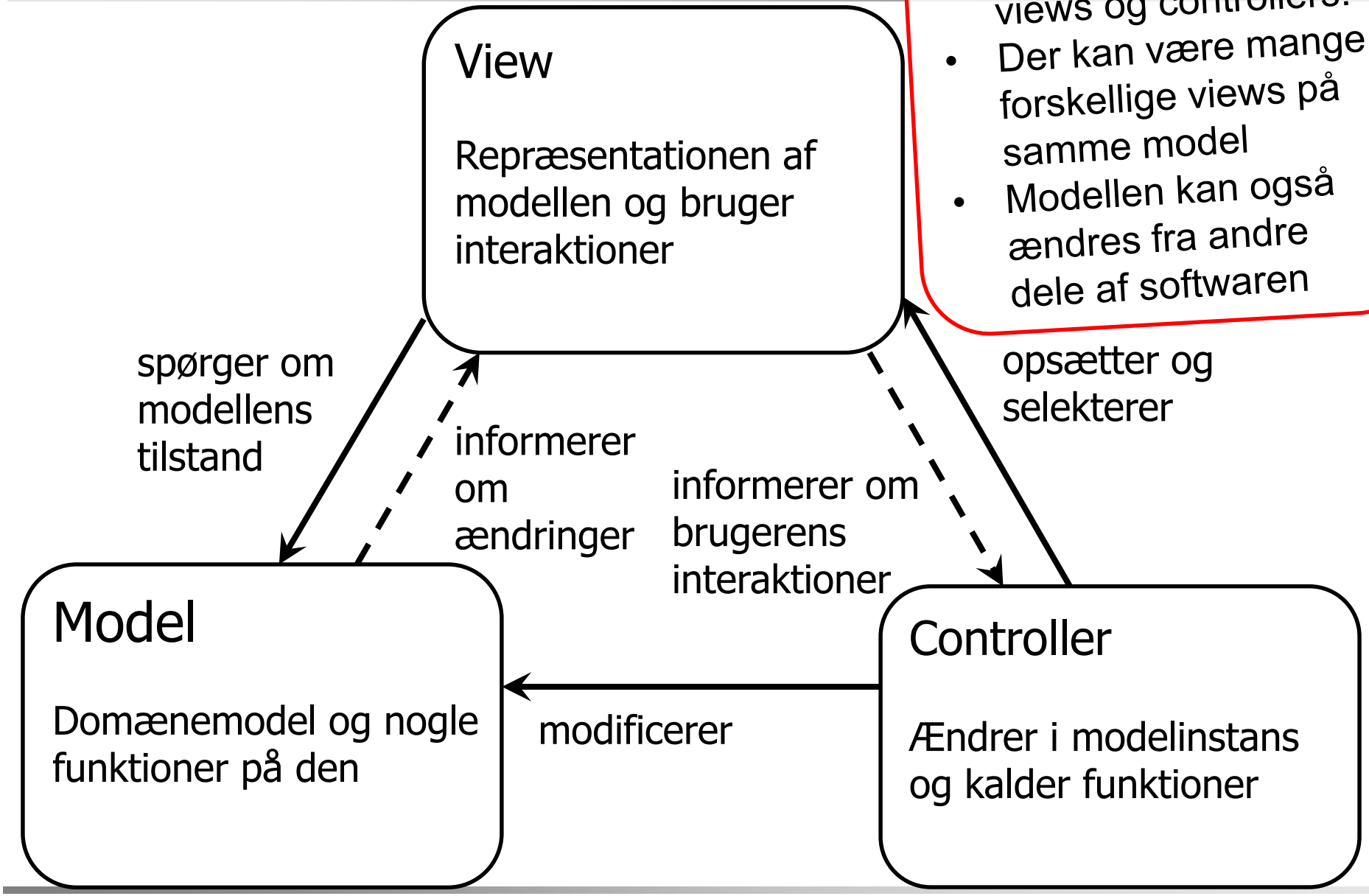
MVC

Bemærk:
- Det er kun hovedideen
- Der er mange varianter



MVC

- Model kender ikke views og controllers!
- Der kan være mange forskellige views på samme model
- Modellen kan også ændres fra andre dele af softwaren

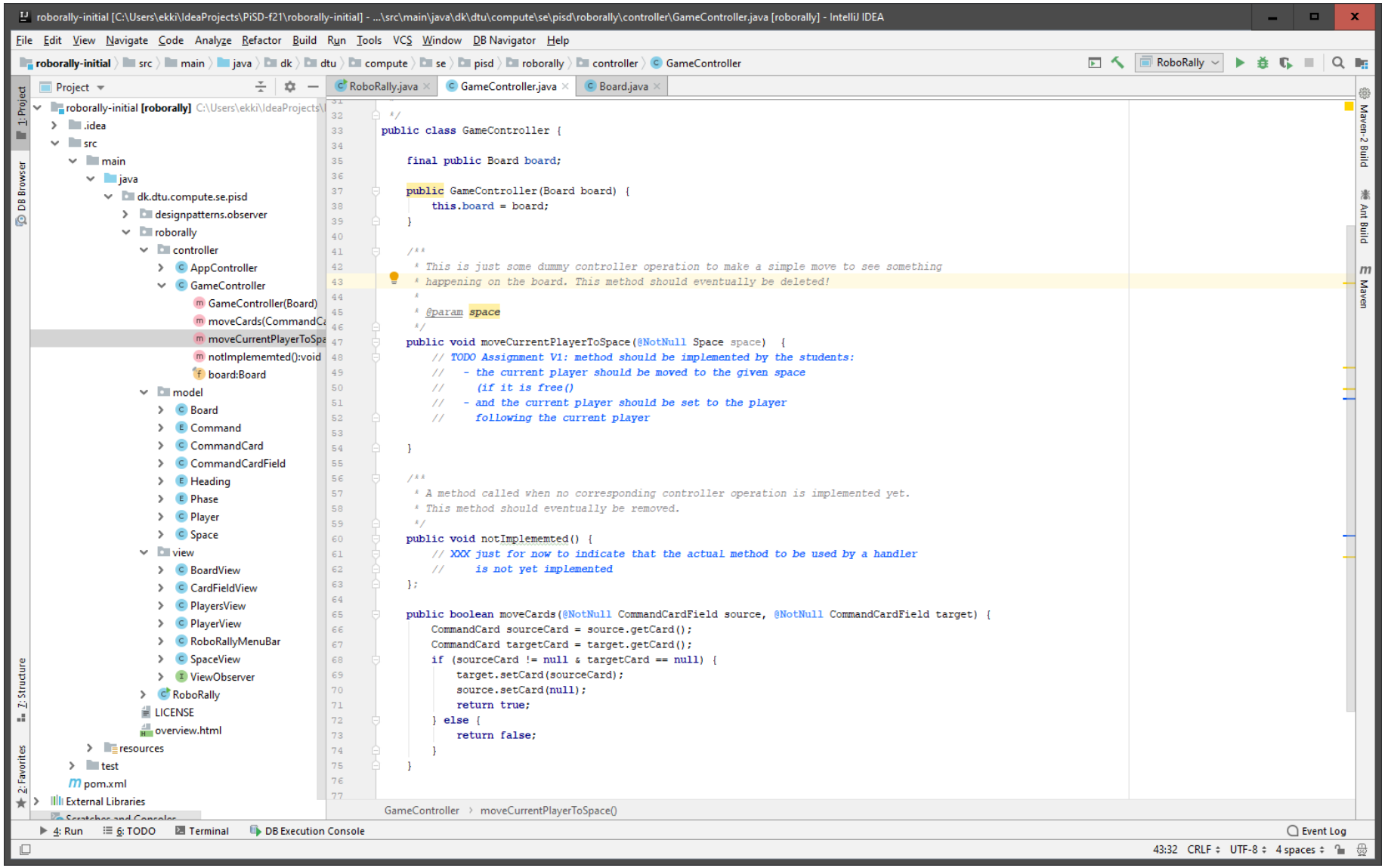


DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

MVC er et designprincip (pattern / arkitektur)
hvordan god software skal være struktureret

Diskussion: Eksempel

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Ledetråde til jeres design

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Misbrug ikke GUIen for at gemme information af selve spillet (all information om spillets tilstand skal gemmes i modellen)

→ Desværre ”inviterer” JavaFX til at gemme information i GUIen! Men det introducerer problemer …

- Spillets logik (reglerne) skal realiseres kun I softwarens controller (baseret på modellens metoder)
- Aktioner af aktivitetsdiagrammer kan fx. implementeres som metoder, som kan bruges af en eller flere kontroller for at starte disse aktiviteter
- Lav et klart interface for at gemme og indlæse modellen (spillets tilstand), som bliver udløst af softwarens kontroller: Data Access Layer / Data Access Objects

Det vender vi tilbage til mange gange!

Kursus i dag (uge 01)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Forelæsnig
  - Introduktion og motivation
  - Software design (overblik) ✓
- Øvelser: Opgave V1
- Projektdiskussion

Resten af opgave skal I klare som hjemmearbejde og vise det næste uge 8.2.).

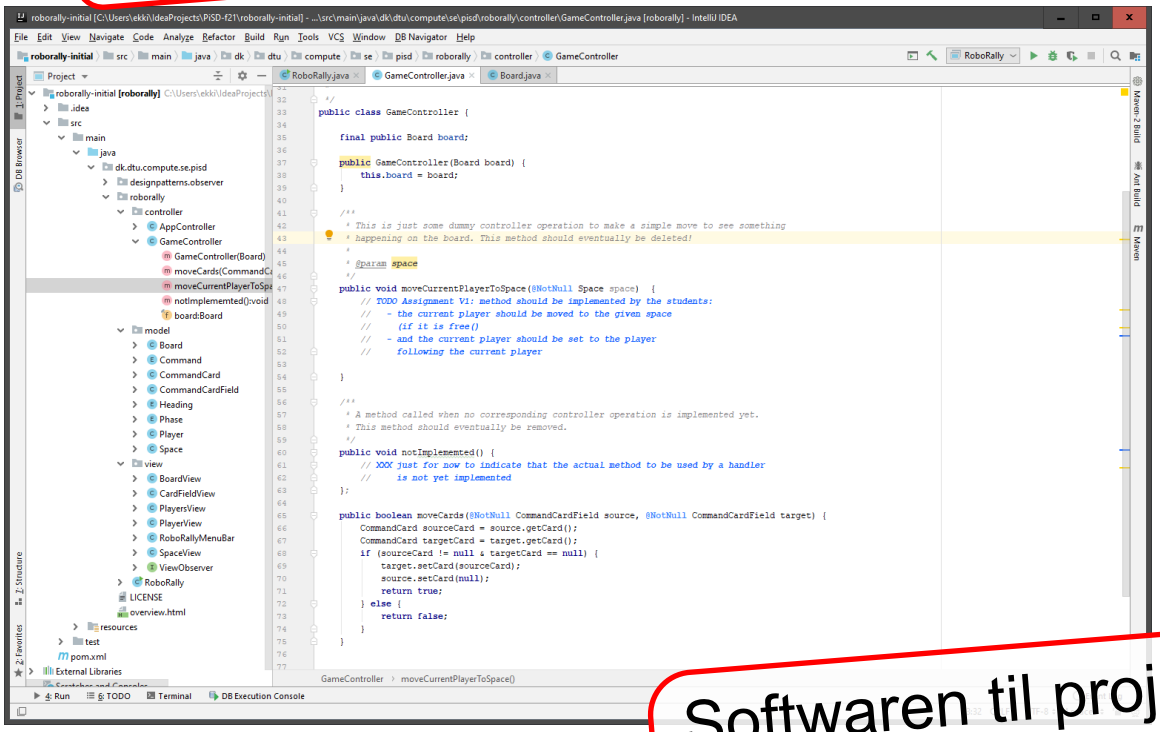
Bemærk at opgave V1 ikke skal afleveres, men **vises til underviseren/ hjælpelærere** næste gang: **V-opgave**!

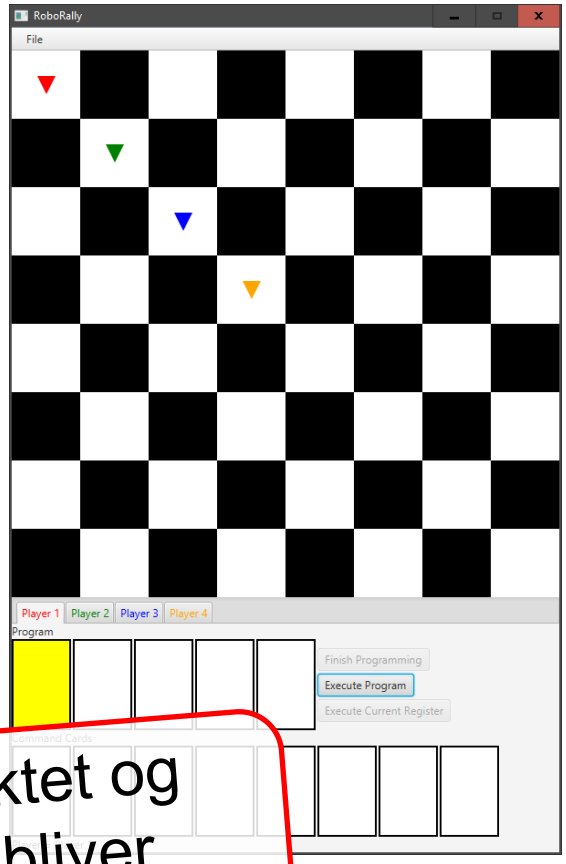
Opgave V1 (til 8. 2. 2022)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Se http://www2.compute.dtu.dk/courses/02362/f22/opgaver/01/

Pt. desværre kun på DTU Learn.





Softwaren til projektet og ideer til løsningen bliver diskuteret separat.

Opgave V1 (til 8. 2. 2022)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Meget kort sagt :

- Kig ind eksemplet ”roborally” (V.1.1.1) og prøv at forstå dets struktur.
- Som resultat, skriv JavaDoc kommentarer til alle pakker, klasser og offentlige metoder.
- Programmer funktionalitet så, at brugeren kan klikke på et tomt felt. Derpå flytter sig brikken af den aktuelle spiller til dette felt, og den næste spiller bliver den aktuelle spiller.
- …

Opgave V1 (til 8. 2. 2022)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Statusfeltet skal vise antallet af træk, som spillerne har lavet.
- Sæt jer ind i RoboRally-spillet og lav en liste med vigtige begreber (taksonomi).

Spillets regler finder man på:

- **`https://avalonhill.wizards.com/games/robo-rally`**
- **`http://media.wizards.com/2017/rules/roborally_rules.pdf`**