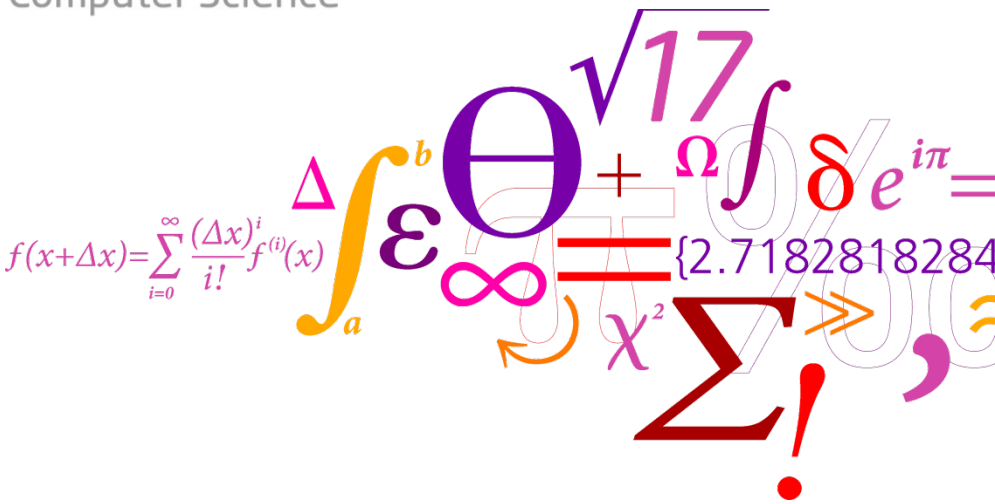
Videregående Programmering (02324)
Projekt i software-udvikling (02362)
Forår 2021

Ekkart Kindler

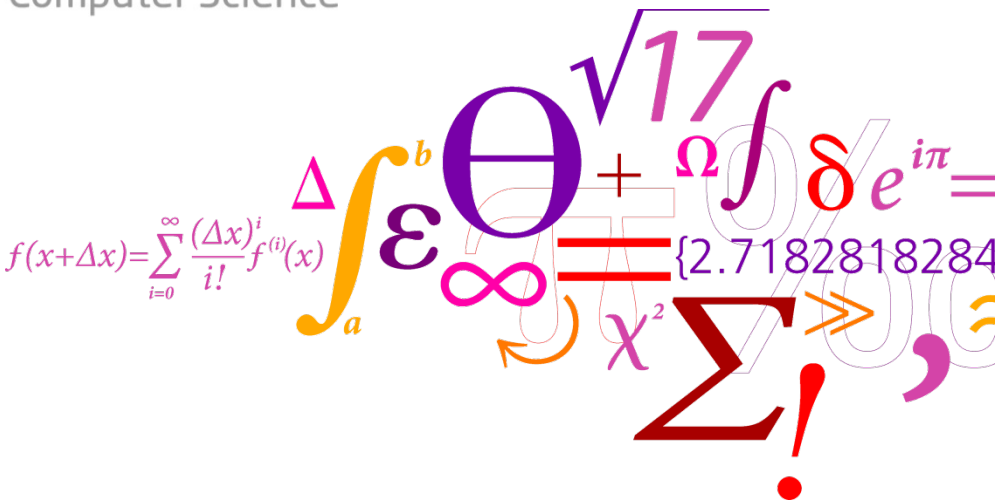
DTU Compute
Department of Applied Mathematics and Computer Science



III. Analyse og konceptuelle modeller

Kort rekapitulation fra sidste gang og live-diskussion (igen)

DTU Compute
Department of Applied Mathematics and Computer Science



Motivation (fra L01)

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

Hvordan gøre 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)

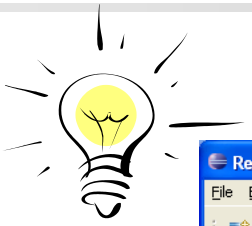
→ **Glossar**: Beskrivelse af begreber deres egenskaber og hvordan de forholder sig til hinanden

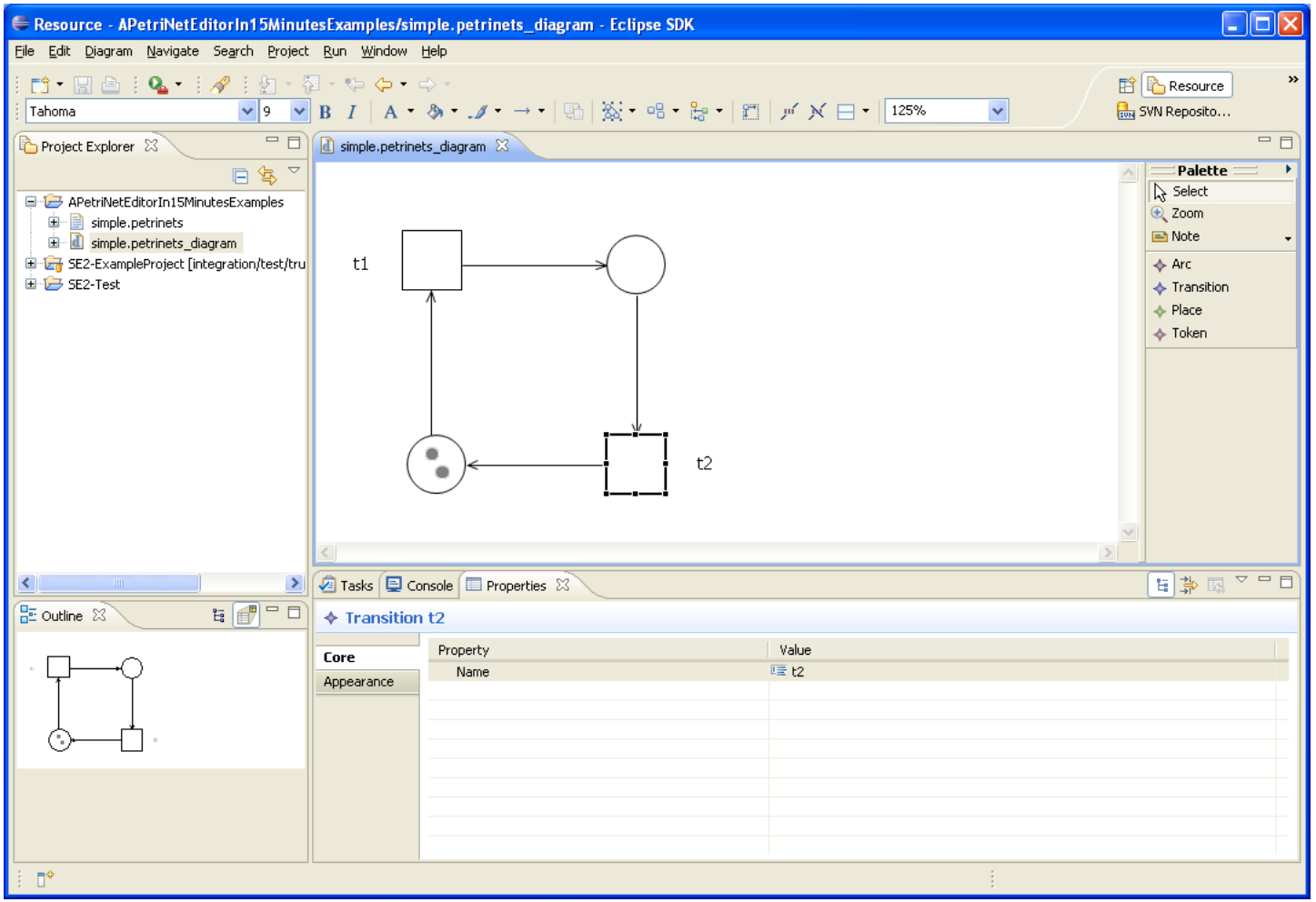
→ **Domænemodel**: Begreber, deres attributter og deres relation repræsenteret som klassediagram og beskrivelse af aktiviteter og begrebernes tilstande (aktivitets- og tilstandsdiagrammer)

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

2. Eksempel: Petrinet

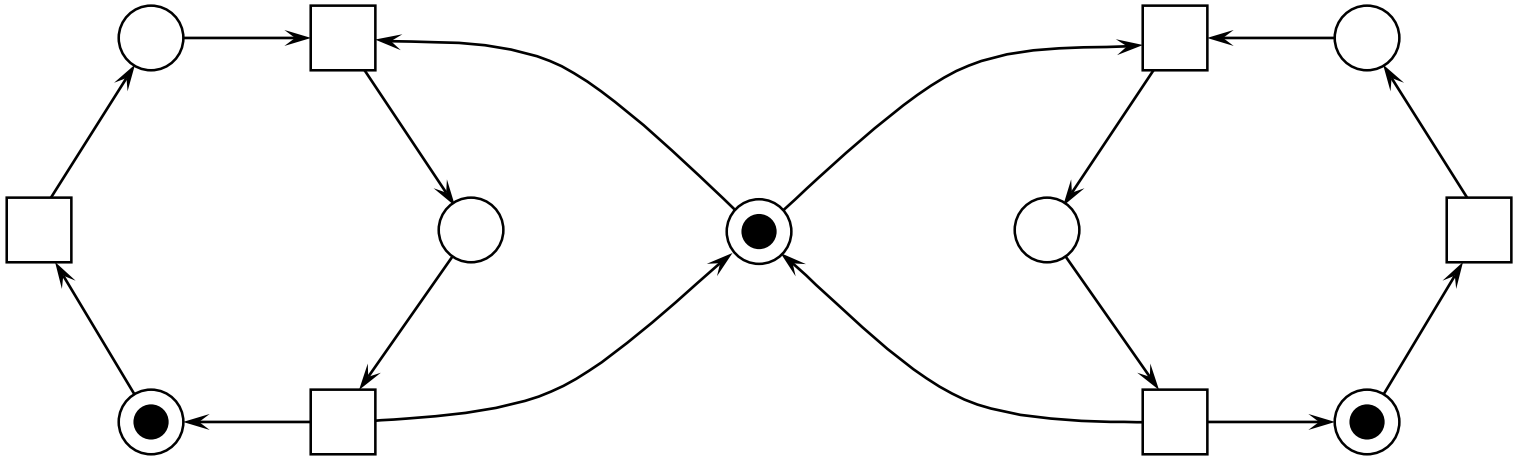
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler





Petrinet

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Trin

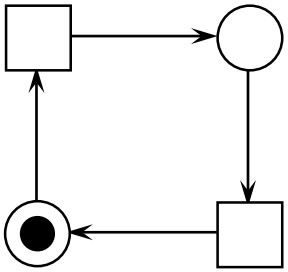
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

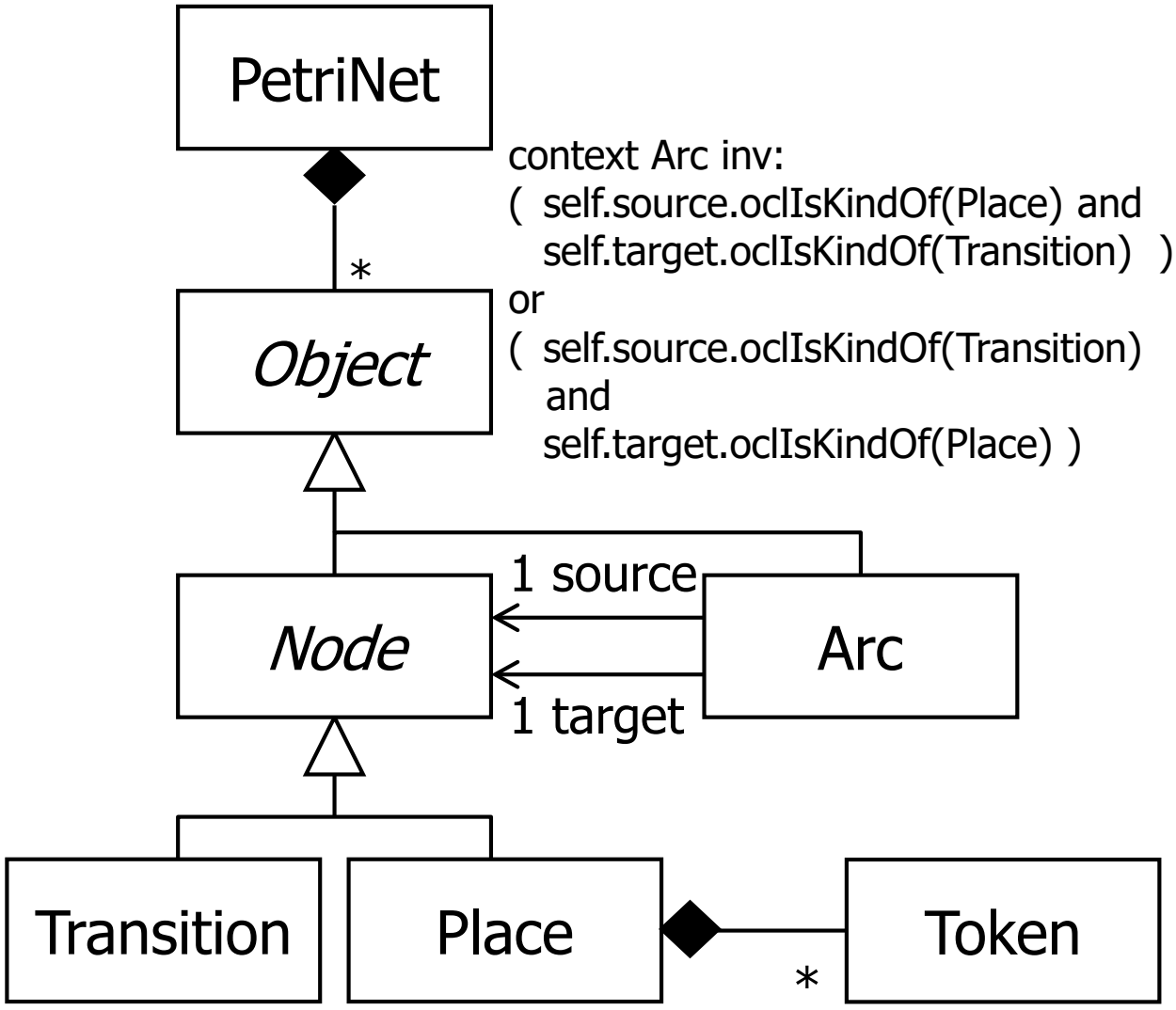
- Eksempler
- Taksonomi (på tavlen)
- Glossar
- Model (på tavlen)

Regel: I skal først starte med at lave et UML diagram, hvis I har kigget på nogle eksempler først og har en liste med hovedbegreberne (taksonomi)!

Model (Meta Model)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler





Meta model for Petri nets

3. Eksempel: RoboRally

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Vi gennemgår processen med hjælp af udtræk af vores RoboRally-projekt

Diskussion i grupper og under live-delen igen.

1. Find beskrivelsen af RoboRally frem
   (se projektslides og især links til RoboRallys regler)
2. Taksonomi
3. Glossar
4. Modeller (klassediagrammer, tilstandsdiagrammer, aktivitetsdiagrammer)

Navneord!

RoboRally-Taksonomi....

DTU | Inser | Desig | Layo | Refer | Maili | Revie | View | Deve | Acro

Templafy | Clipboard | Font | Paragraph | Styles | Editing | Adobe Acrobat

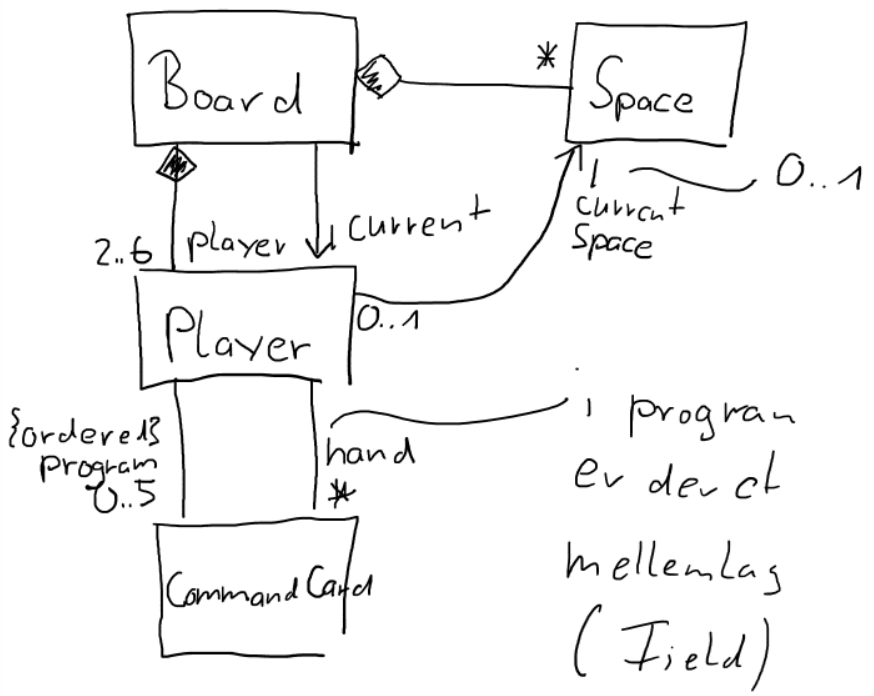
RoboRally: Taksonomi

Hovedord (nouns)

- Robot (player)
- Kommadokort
- Felter (Space)
- Bræt (Board) / Game
- Antenna
- Energy cubes
- Hand card
- Program

Udsagnsord (verbs)

- Start game
- Ryk frem, ... (kommandoer)
- Power-up (upgrade)
- Aktivate programs (robots)
- Program robots



Livscyklus af udvalgte objekter

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Vi starter med livscyklus af udvalgte objekter, da de er typisk nemmere og tætter på klassediagrammet:

- spil
- spiller (== robot ?)
- felt
- kort
- …

Diskussion i grupper og på tavlen; vi bruger tilstandsdiagrammer (UML state machines) til det

Faktisk kunne nogle af de der tilstande være et attribut af klasser i klassediagrammet (eller de kan afledes af nogle af objektets andre attributter.

Aktiviteter

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

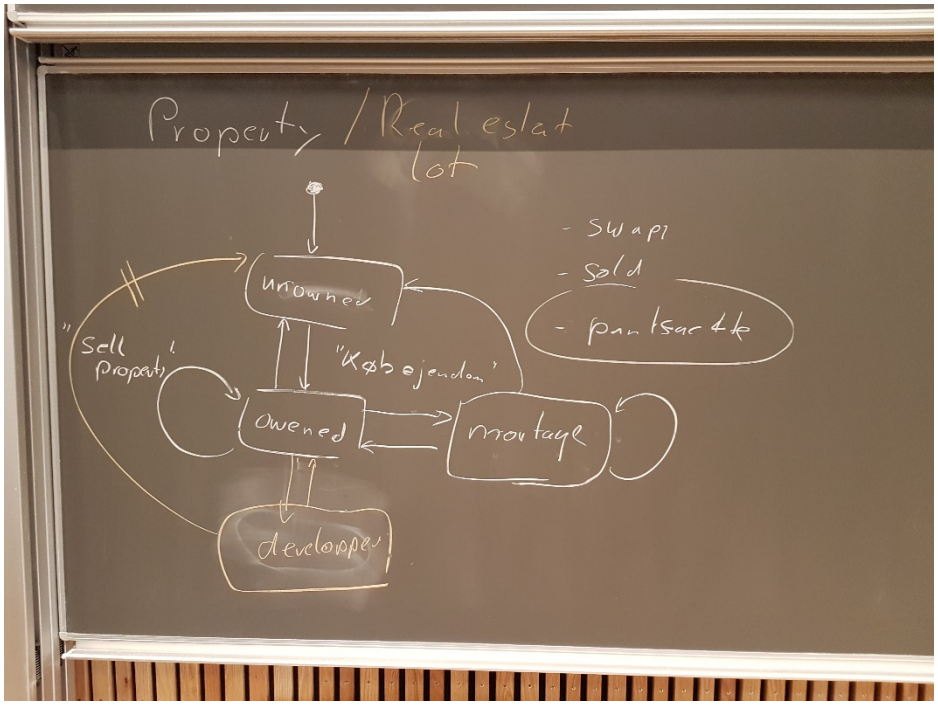
Vi diskuterer udtræk af nogle udvalgte aktiviteter:

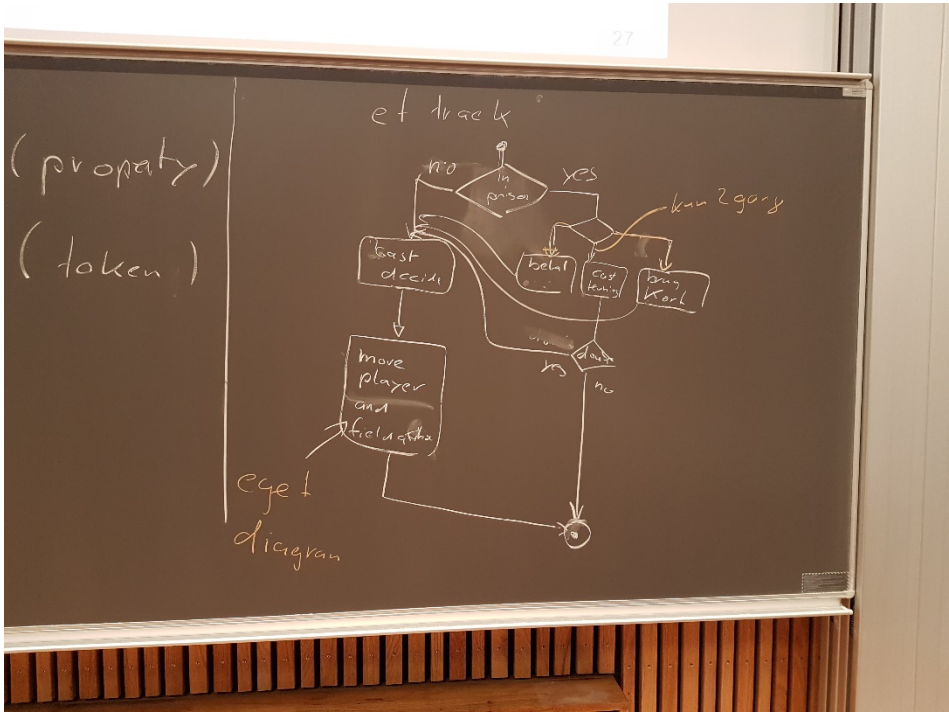
- lav et træk???
- …

Diskussion i grupper og på tavlen; vi bruger aktivitetsdiagrammer (UML activity diagrams) til det

Bemærk at nogle aktiviteter bliver udført som del af nogle andre aktiviteter; og aktiviteter kan køre også sideløbende med andre aktiviteter

Fra sidste år:
Monopoly/Matador

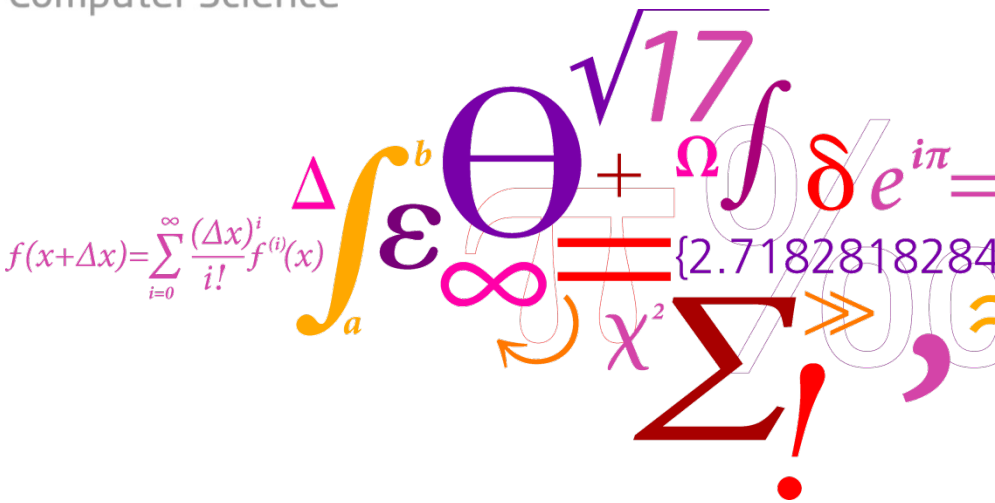




II. (Software) Design

Med nogle flere design patterns

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

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 den 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!

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]
- …

Ansvarsfraskrivelse

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Design patterns er et emne i sig selv og kunne være indhold af et separate kursus eller seminar.
- Her gives der bare en kort overblik over ideen bagved og nogle meget vigtige design patterns, som er vigtige at forstå good design.

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

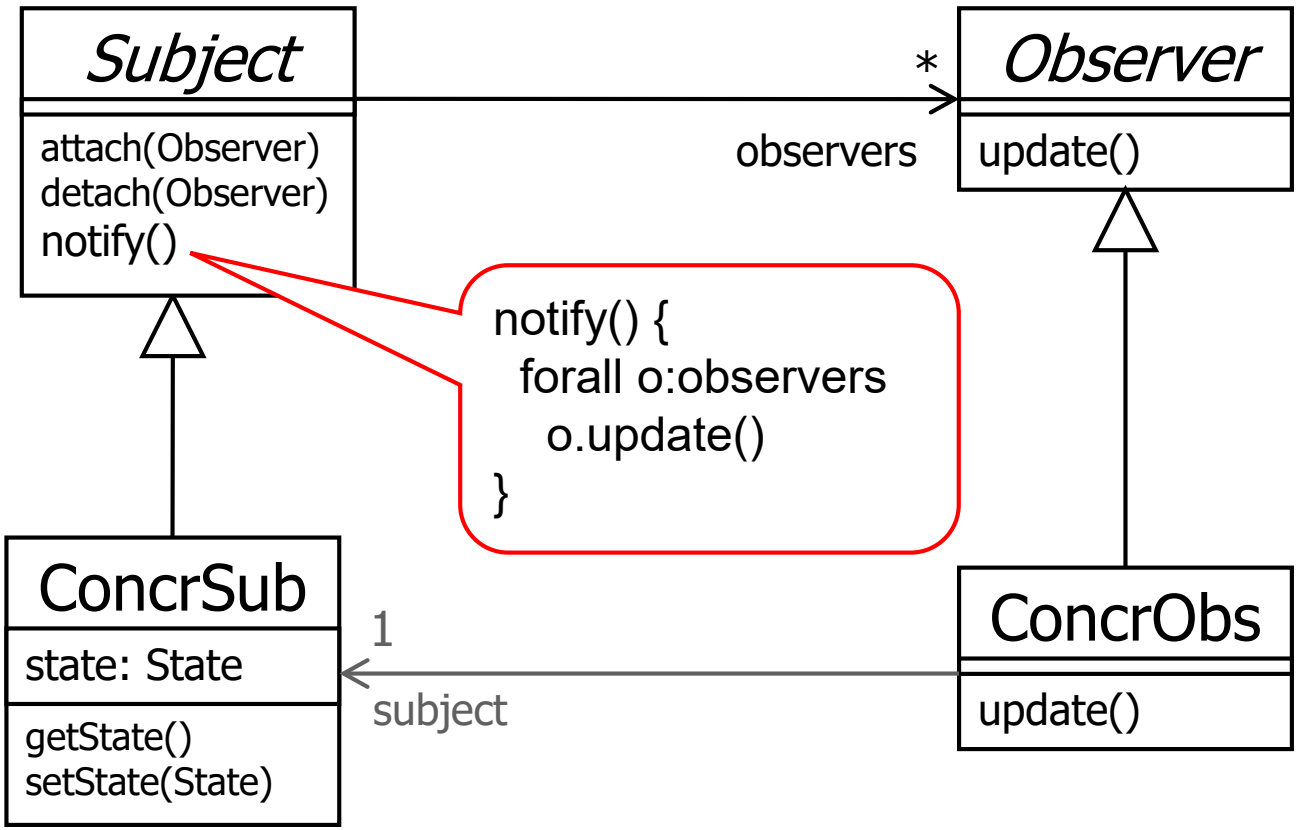
... update views when the underlying model changes ...

→ MVC!

Eksempel: Observer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

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

Example: Singleton (GoF)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Se [GoF] eller [FF] for detaljer

Name and classification

Singleton, object-based, creational

Intent

Ensure that a class has only one instance, and provide a global point of access to it [GoF]

Motivation

…

Bemærk: Tænk jer godt om, inden I burger singletons (eller statiske attributter). Meget ofte bliver singletons brugt som "hack"!

Example: Command

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Name and classification

Command, behavioural

Intent

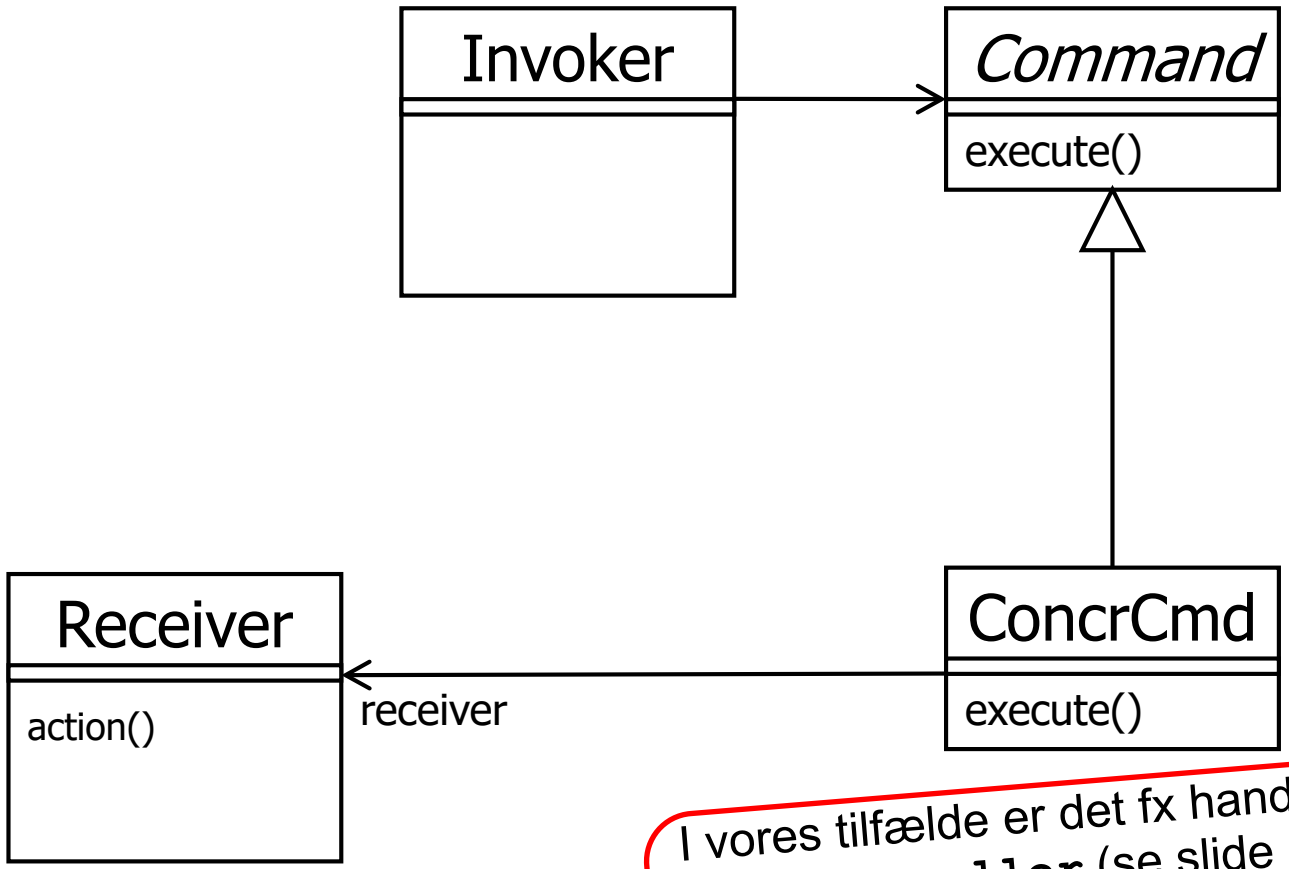
Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations [GoF]

Motivation

Use of different implementations in different contexts with easy portability …

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Structure



I vores tilfælde er det fx handlere der kalder **`GameController`** (se slide 36)
Generel er i Java **`Runnable`** som indkapsler kode som skal eksekveres af andre (det vender vi tilbage til).

Example: Command

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- **Variation / extension**
  A command can also have `undo()` and `redo()` methods. This can be used to store executed commands on a undo-stack; by executing the `undo()` method in reverse order, earlier commands can be undo (and redone).

Denne slags kommandoer er kernen i undo/redo-mekanismen af mange editorer!

Eksempel: Abstract Factory

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Name and classification

Abstract factory, object, creational

Intent

Provide an interface for creating families of related or dependent objects without specifying their concrete classes [GoF]

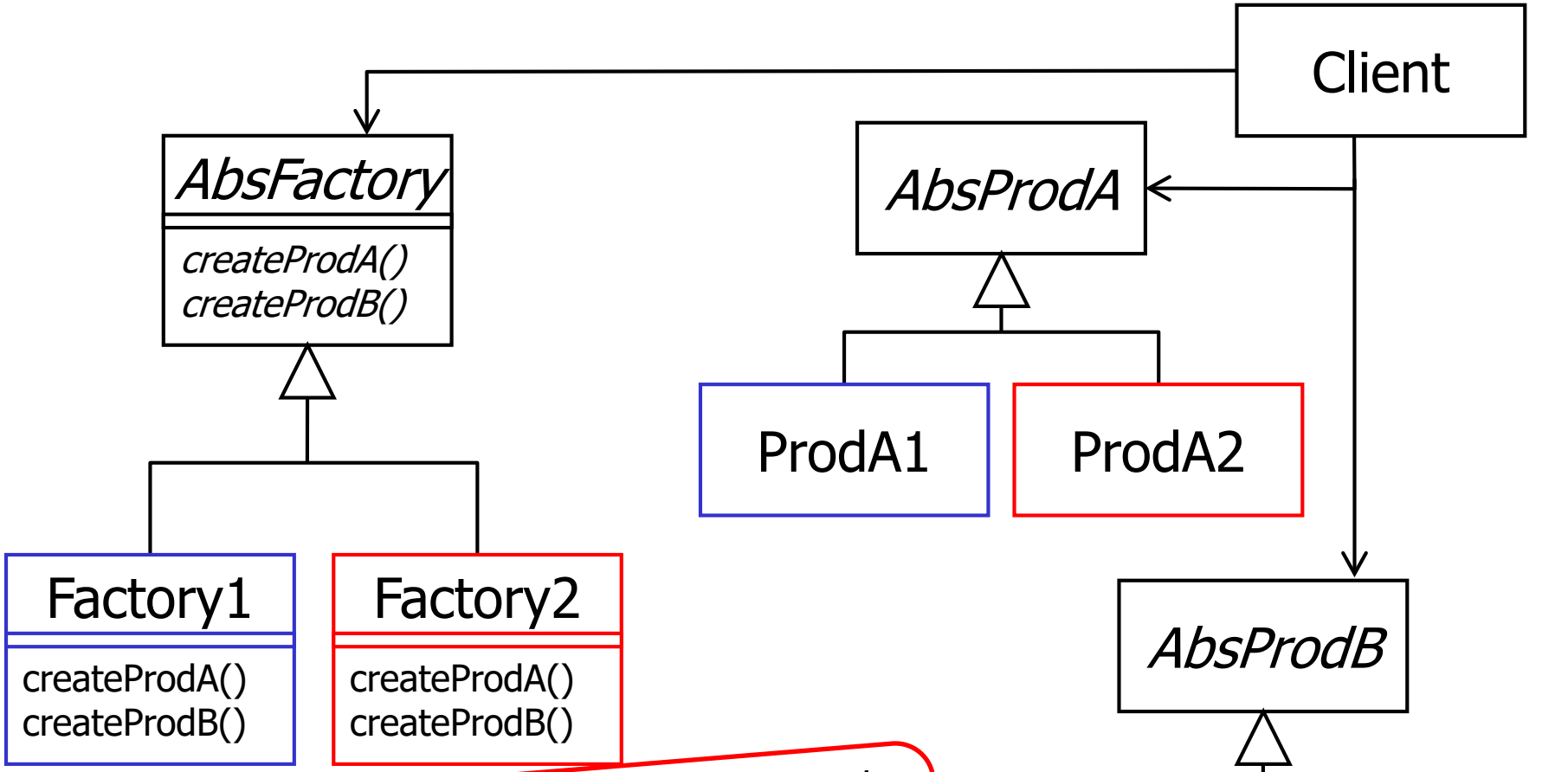
Motivation

Use of different implementations in different contexts with easy portability …

Det vender vi måske tilbage til senere!

Eksempel: Abstract Factory

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Det bruges ofte med automatisk generet kode hvor den genererede kode ligger bagved et interface; brugeren skal ikke referere til implementeringen, men kun til interfaces.

Design Patterns: Resumé

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- GoF præsenterer 23 design patterns
- Der eksisterer mange flere (og nogle komplekse kombinationer af de enkle mønstre, fx. MVC)

- "Pattern terminologien" bruges ofte til at kommunikere designet! Derfor er det vigtigt at kende design patterns!
- Design patterns hjælper at forstå det overordne design af et stykke software

- Design patterns skal **ikke bruges** alt for skematiske

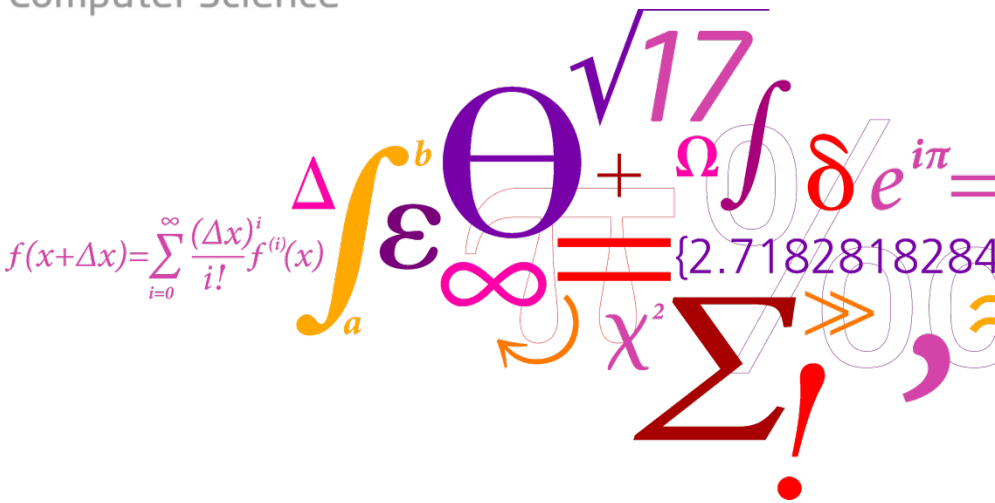
- Der eksisterer også **anti-patterns** (→ bad smells): hvordan man ikke skal programmeres

IV. Java Praksis

Det vil blive vist med praktiske eksempler fra vores projekt i IntelliJ senere.

Her er der kun nogle referencer!

DTU Compute
Department of Applied Mathematics and Computer Science



1. JavaDoc-Kommentarer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- I jeres endelige projekt skal alle offentlige klasser, metoder og attributter have JavaDoc-kommentarer
- Se fx. pakken **`designpatterns`** i koden til jeres opgaver!
- Ud fra disse kommentarer kan der genereres HTML dokumentation af jeres kode
- See
  [JD:HowTo] *The Javadoc Homepage: How to*:
  http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html
  [IJ:JavaDoc] *IntelliJ IDEA: Documenting code*:
  https://www.jetbrains.com/help/idea/working-with-code-documentation.html#generate-javadoc

Eksempel

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

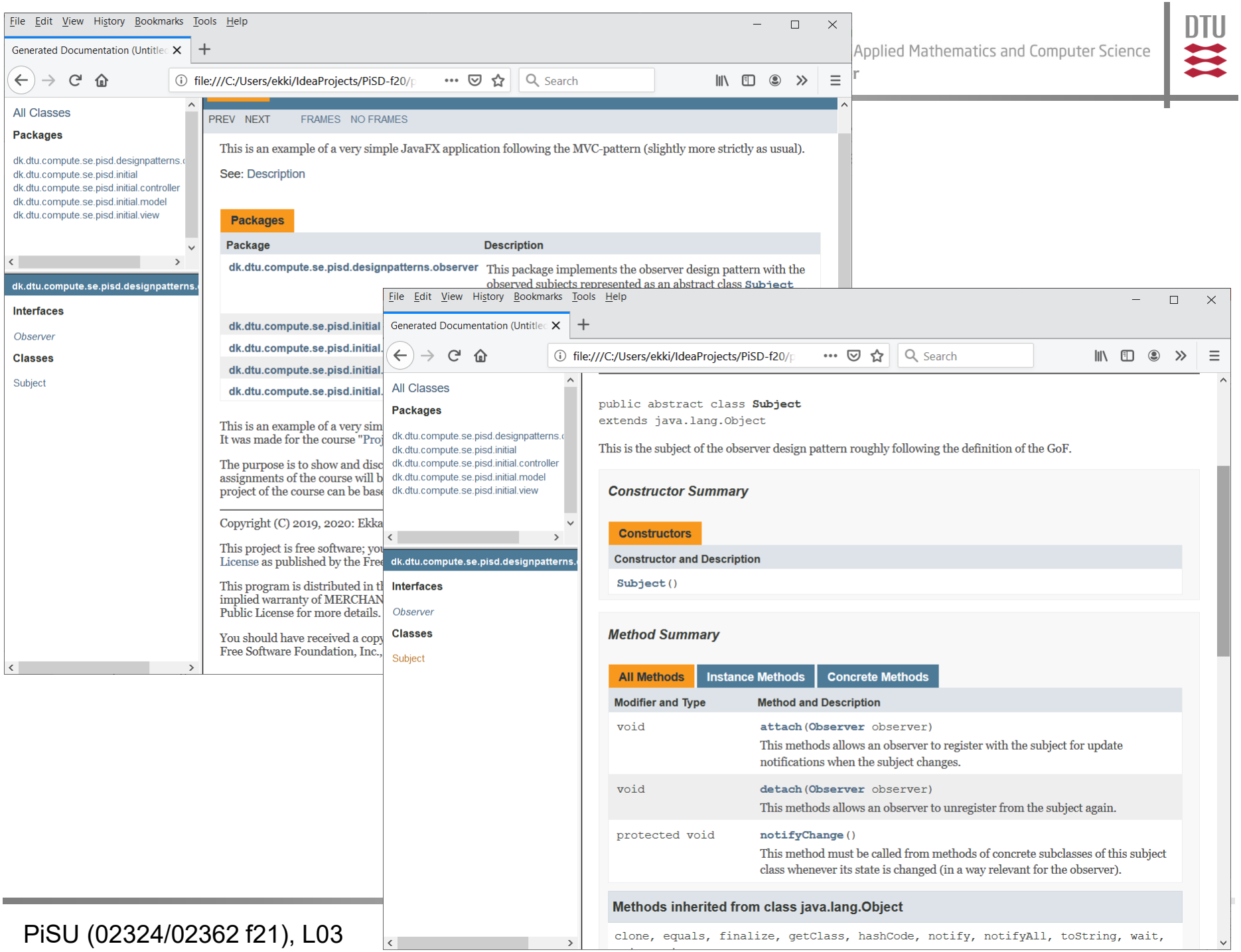
`@xxx` kaldes for JavaDoc tags!

```
/**
 * This is the subject of the observer design pattern roughly
 * following the definition of the GoF.
 *
 * @author Ekkart Kindler, ekki@dtu.dk
 */
public abstract class Subject { ...
```

Java-metoder har typisk ikke `@author` tags! Men i jeres projekt skal I angive autorer til individuelle metoder, hvis ikke en autor stor for hele klassen.

```
    /**
     * This methods allows an observer to register with the subject
     * for update notifications when the subject changes.
     *
     * @param observer the observer who registers
     */
    final public void attach(Observer observer) {

...
```


File Edit View History Bookmarks Tools Help
Generated Documentation (Untitled)
file:///C:/Users/ekki/IdeaProjects/PiSD-f20/p
Search
All Classes
Packages
dk.dtu.compute.se.pisd.designpatterns
dk.dtu.compute.se.pisd.initial
dk.dtu.compute.se.pisd.initial.controller
dk.dtu.compute.se.pisd.initial.model
dk.dtu.compute.se.pisd.initial.view
PREV NEXT FRAMES NO FRAMES
This is an example of a very simple JavaFX application following the MVC-pattern (slightly more strictly as usual).
See: Description
Packages
Package
Description
dk.dtu.compute.se.pisd.designpatterns.observer
This package implements the observer design pattern with the observed subjects represented as an abstract class Subject
dk.dtu.compute.se.pisd.designpatterns.
Interfaces
Observer
Classes
Subject
dk.dtu.compute.se.pisd.initial
dk.dtu.compute.se.pisd.initial.
dk.dtu.compute.se.pisd.initial.
dk.dtu.compute.se.pisd.initial.
This is an example of a very sim
It was made for the course "Proj
The purpose is to show and disc
assignments of the course will b
project of the course can be base
Copyright (C) 2019, 2020: Ekka
This project is free software; yo
License as published by the Free
This program is distributed in th
implied warranty of MERCHAN
Public License for more details.
You should have received a copy
Free Software Foundation, Inc.,
File Edit View History Bookmarks Tools Help
Generated Documentation (Untitled)
file:///C:/Users/ekki/IdeaProjects/PiSD-f20/p
Search
All Classes
Packages
dk.dtu.compute.se.pisd.designpatterns.
dk.dtu.compute.se.pisd.initial
dk.dtu.compute.se.pisd.initial.controller
dk.dtu.compute.se.pisd.initial.model
dk.dtu.compute.se.pisd.initial.view
dk.dtu.compute.se.pisd.designpatterns.
Interfaces
Observer
Classes
Subject
public abstract class Subject
extends java.lang.Object
This is the subject of the observer design pattern roughly following the definition of the GoF.
Constructor Summary
Constructors
Constructor and Description
Subject()
Method Summary
All Methods
Instance Methods
Concrete Methods
Modifier and Type
Method and Description
void
attach(Observer observer)
This methods allows an observer to register with the subject for update notifications when the subject changes.
void
detach(Observer observer)
This methods allows an observer to unregister from the subject again.
protected void
notifyChange()
This method must be called from methods of concrete subclasses of this subject class whenever its state is changed (in a way relevant for the observer).
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait,

2. JavaFX Programmering

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Til at programmere GUIen burger vi JavaFX
- Her er der nogle informationer om JavaFX programmering:
- JavaFX Applikationen:
  - https://docs.oracle.com/javafx/2/get_started/hello_world.htm
  - https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
- JavaFX LayoutPanes:
  - https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm#
- JavaFX UI Controls:
  - https://docs.oracle.com/javafx/2/ui_controls/overview.htm#
  - https://docs.oracle.com/javafx/2/ui_controls/text-field.htm#
  - https://docs.oracle.com/javafx/2/ui_controls/button.htm#
- JavaFX Dialogs:
  - https://examples.javacodegeeks.com/desktop-java/javafx/dialog-javafx/javafx-dialog-example/

Knapper og aktioner

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
Button optionButton = new Button("Option 1");


optionButton.setOnAction(
  e -> gameController.notImplemented() );

optionButton.setDisable(false);


playerInteractionPanel.getChildren().add(optionButton);
```

Kreerer et nyt knap-objekt med label "Option 1"

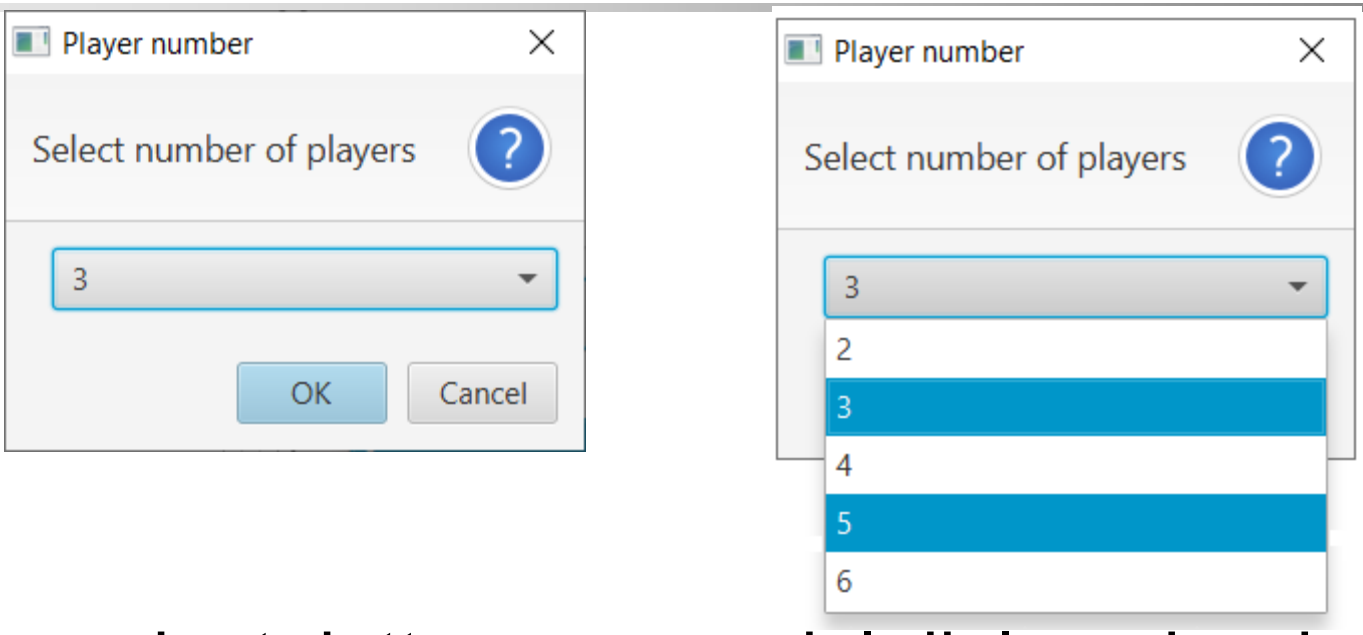
Tilknytter en handling (aktion) til knappen, som bliver eksekveret, når brugeren trykker på knappen.

Siger om knappen skal vises disabled eller enabled.

Tilføjer knappen til et andet GUI element (her spilleret panel). Først her bliver knappen synligt (på dette panel)!

Eksempel: ChoiceDialog

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

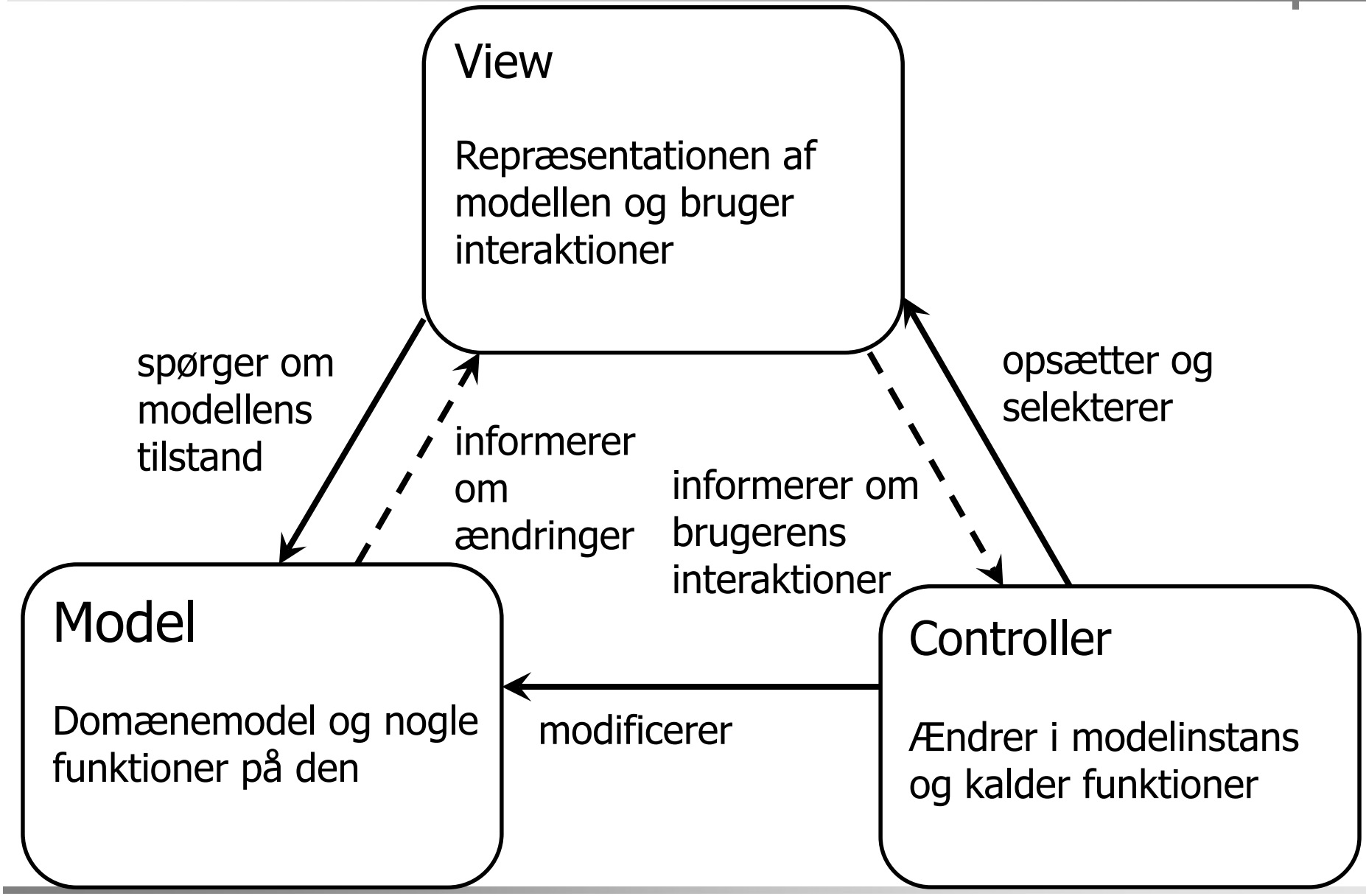


- Bemærk at dette er en modal dialog, dvs. brugeren kan først interagere med resten af GUIen (eller forældre-dialog), når denne her dialog er afsluttet.

I **`GameController`** skal I ikke bruge (modale) dialoger! Se opgave V3, hvordan man kan undgå dette! Men under selve starten af spillet (**`AppController`**) er det OK!

MVC

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



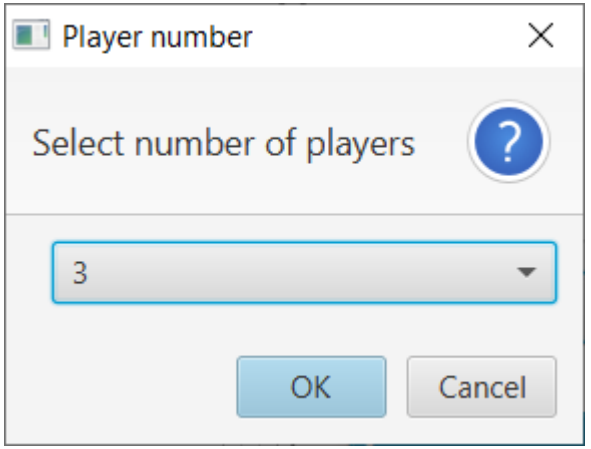
Eksempel: ChoiceDialog

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
final private List<Integer> PLAYER_NUMBER_OPTIONS =
    Arrays.asList(2, 3, 4, 5, 6);
...

Optional<Integer> result = null;
do {
    ChoiceDialog dialog =
      new ChoiceDialog(PLAYER_NUMBER_OPTIONS.get(0),
                       PLAYER_NUMBER_OPTIONS);
    dialog.setTitle("Player number");
    dialog.setHeaderText("Select number of players");
    result = dialog.showAndWait();
} while (!result.isPresent());

int no = result.get();
```



Opgave V3 (til 1./2. 3.)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

http://www2.compute.dtu.dk/courses/02362/f21/opgaver/V03/

- I får udleveret en udvidet implementering af kommandoer, som også indeholder et *interaktivt kommando*! Det kræver, at spilleren interagerer med softwaren, når sådan et kort bliver eksekveret (for at vælge hvad der skal ske).
- I skal udvide **`GameController og PlayerView`**, så at der vises knapper med de mulige optioner, når kortet bliver eksekveret. Og når spilleren trykker på en af disse knapper, skal **`GameControlleren`** kaldes og det valgte kommando skal eksekveres for den aktuelle spiller (og eksekvering a robotternes programmer skal fortsættes).
- Derudover skal i dokumentere programmet med JavaDocs!

Opgave A1 (til 22./23. 2.)

Men husk aflevering A1 (næste uge)!

Detaljerede analyse/konceptuelle modeller til RoboRally (**afleveringsopgave**):

- Taksonomi (tilstandsbegreber / aktionsbegreber)
- Klassediagrammer

- Tilstandsdiagrammer til relevante klasser (frivilligt)
- Aktivitetsdiagrammer til relevante aktiviteter (frivilligt)