

# Systemudvikling (1. sem.)

Hvor er vi nu?



Hvad:

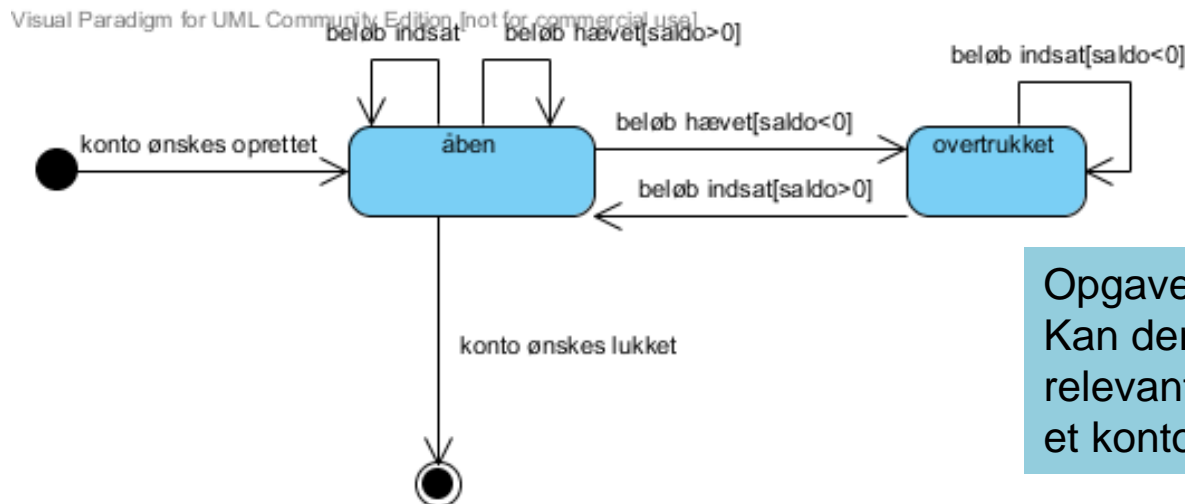
- Krav ✓
- Use case diagram ✓
- Use case beskrivelse ✓
- Analyseklasse ✓
- Analyseklasse generalisering ✓
- Analyseklassediagram ✓
- Adfærdsmønstre
- System sekvensdiagrammer(SSD)
- osv.

Hvordan:

- Designklasse ✓
- Design ✓
- Designklassediagram ✓
- Designkriterier og arkitektur ✓
- Brugergrænsefladedesign ✓
- Design af adfærd
- osv.

# Adfærd

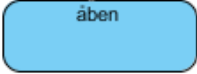



- **Def. adfærdsmønster (notation tilstandsdiagram)**
  - Abstrakt beskrivelse af de mulige hændelsesforløb og tilstande for objekter af en klasse



Opgave:  
Kan der være andre  
relevante tilstande på  
et konto objekt?

Eksempel på adfærdsmønster/tilstandsdiagram for objekter af  
klassen Konto

# Tilstandsdiagram(notation)

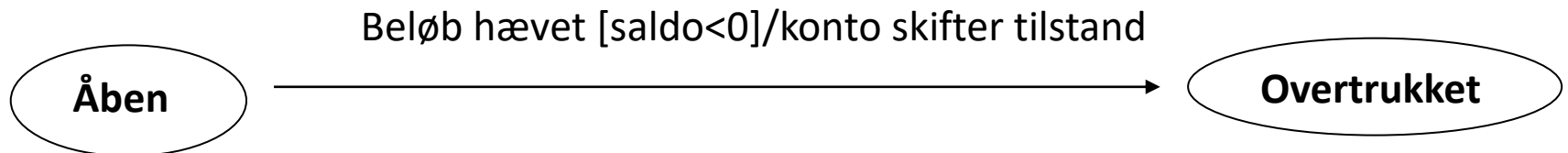
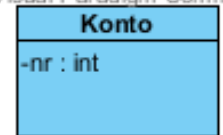
En tilstand	<p>Visual Paradigm for UML Com</p> 
En starttilstand	<p>Visual Par</p> 
En sluttilstand	<p>Visual Par</p> 
En hændelse	Konto ønskes oprettet
En transition	

# Syntax transition

- Transition har 3 frivillige elementer:
  - **Hændelse**: xxx
  - **Betingelse** (guard) : [xxx]
  - **Aktion** : /xxx (hvad der trigges af hændelsen)

Eks. transition på tilstandsdiagram for klassen Konto

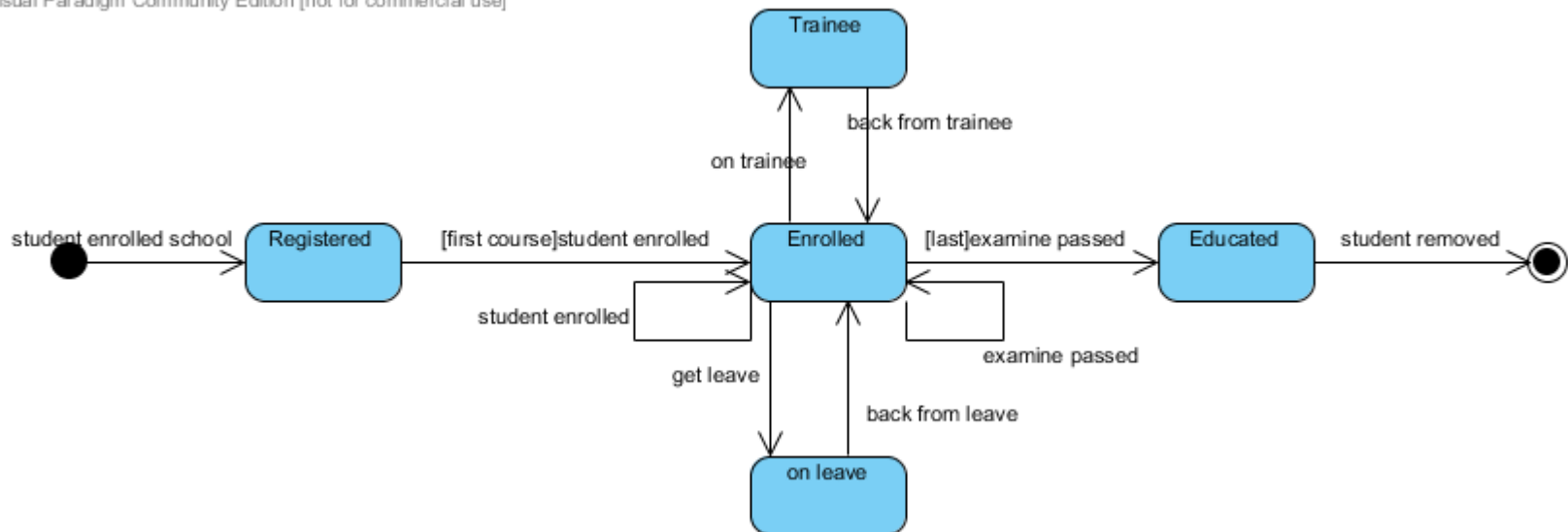
Visual Paradigm Commu



# Opgave

Se på figur 2.18 s. 38 i noten, diskuter og forstå diagrammet!  
Noget I ville tilføje?

Visual Paradigm Community Edition [not for commercial use]



Figur 2.18 Adfærdsmønster/tilstandsdiagram for objekter af klassen Student



# Hvornår beskrives adfærd

- Når opførsel/adfærd af klassens objekter er **tilstrækkelig kompliceret !**  
(beskrives altså ikke for alle klasser)
- Der er et **abstrakt mønster** af hændelsesforløb
- **Fælles forløb** for alle objekter i en klasse
- Afspejle **lovlige hændelsesforløb**

# Opgave

- Overvej om nogle af klasserne i KAS har noget opførsel/adfærd af klassens objekter, der er **tilstrækkelig kompliceret?**
- Efter 10 min. vender vi lige, hvad I er kommet frem til i første punkt, for at sikre, alle har noget at tegne i næste punkt
- Hvis ja så prøv i Visual Paradigm (State Machine Diagram) at tegne adfærdsmønsteret for objekter af klassen!

# Opgave

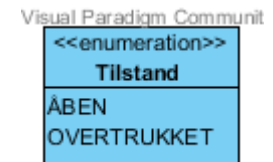
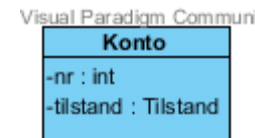
- Opgave 1 på opgavesedlen – 20 min.



# Design adfærd

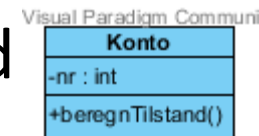
- Realisering af adfærd, flere måder

- Tilstandsattribut på klassen



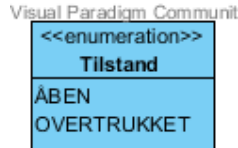
eller

- Metode der beregner tilstand



# Kode adfærd

```
public enum Tilstand {
    ÅBEN, OVERTRUKKET;
}
```



```
public class Konto {

    ....

    private KontoType kontoType;

    private Tilstand tilstand;

    public Konto(KontoType kontoType)
    {
        .....
        this.tilstand = Tilstand.ÅBEN;
        .....
    }

    .
    .
    .
}
```

```
public Transaktion hævEllerIndsætBeløb(double
beløb) {
    Transaktion transaktion = null;
    if (!(tilstand == Tilstand.OVERTRUKKET &&
        beløb < 0)) {

        transaktion = new Transaktion(beløb);
        transaktioner.add(transaktion);

        if (beregnsaldo() < 0) {
            tilstand = Tilstand.OVERTRUKKET;
        } else {
            tilstand = Tilstand.ÅBEN;
        }

    } else
        throw new RuntimeException(
            "Du forsøger at hæve på en
            overtrukket konto!");

    return transaktion;
}

....
```

# Matematisk Logik

```
if (!(tilstand == Tilstand.OVERTRUKKET &&  
    beløb < 0)) {
```

```
...  
}
```

de Morgans love ( $p$  og  $q$  er sand eller falsk (booleans)):

$$\neg(p \vee q) \equiv (\neg p) \wedge (\neg q)$$

$$\neg(\mathbf{p} \wedge \mathbf{q}) \equiv (\neg \mathbf{p}) \vee (\neg \mathbf{q})$$

Så:

$$\begin{aligned} &\neg((t = to) \wedge (b < 0)) \\ &\equiv ((t \neq to) \vee (b \geq 0)) \end{aligned}$$

Så vi kan fjerne "!":

```
if ((tilstand != Tilstand.OVERTRUKKET //  
    beløb >= 0)) {
```

```
...  
}
```

# Opgave

1. Importer koden med bankeksamplet. Læs og forstå koden ved evt. at tage en snak med dem i Breakoutroom'et. Kør testklassen nogle gange, ved at lave nogle nye eksempler på beløb, der **indsættes og hæves!**
2. **Udvid koden** med en **metode** i hhv. **Controller** og **Konto** til at **lukke** en konto med. **Derudover hvilke ændringer i metoden *createTransaktion* i Konto kræver det?**  
Lav ændringerne og test det af.

# Opgave

- opgave 2 eller 3 på opgavesedlen

# Opgave

- Opgave 4 på opgavesedlen