



KURSER:

02312 02313 02315

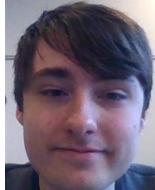
CDIO 2



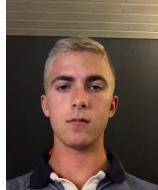
Jeppe Kaare Larsen



Anton N. Ranløv



Mathias Skøn Frimann



Nicklas Beyer Lydersen



Rasmus Søborg



Hans Krogh

9. november 2018

Ansvarsområder

Diverse	Navn
LaTeX-opsætning	Jeppe og Anton
Indledning	Jeppe
Korrektur	Jeppe

Analyse	Navn
MoSCoW-analayse	Nicklas
Use-case-diagram	Fælles
Systemsekvensdiagram	Fælles
Domænemodel	Fælles

Design	Navn
Arkitektur	Jeppe
Designklassediagram	Fælles
Sekvensdiagram	Fælles
GRASP-mønstre	Jeppe

Kode	Navn
CDIO2	Anton, Hans og Rasmus
DiceCup(CDIO1)	Rasmus
Dice(CDIO1)	Rasmus
View-klasse og smårettelser	Jeppe

Implementering	Navn
Dokumentation	Jeppe
Javadoc i koden	Nicklas og Jeppe

Test	Navn
Test	Rasmus

Indhold

Ansvarsområder	1
Indledning	3
Hoved afsnit	4
1 Analyse	4
1.1 MoSCoW Analyse	4
1.2 Krav	4
1.3 Use-case-diagram	6
1.4 System Sekvensdiagram	7
1.5 Domænemodel	8
2 Design	9
2.1 Arkitektur	9
2.2 Designklassediagram	10
2.3 Sekvensdiagram	10
2.4 GRASP-mønstre	11
2.5 Use-cases	12
3 Implementering	14
3.1 Dokumentation	14
4 Test	15
Bilag	16
Bilag 1 - Designklassediagram (roteret)	16

Indledning

Der er i udviklingsvirksomheden IOOuterActive stillet til opgave, at udvikle et spil mellem to personer, som skal spilles i DTU's databarer.

Kundens vision er som følger (kilde: opgavebeskrivelsen):

Spillerne slår på skift med 2 terninger og lander på et felt med numrene fra 2 - 12. At lande på hvert af disse felter har en positiv eller negativ effekt på spillernes pengebeholdning. (Se den følgende feltoversigt), derudover udskrives en tekst omhandlende det aktuelle felt. Når en spiller lander på Goldmine kan der f.eks. udskrives: "Du har fundet guld i bjergene og sælger det for 650, du er rig!".

Spillerne starter med en pengebeholdning på 1000. Spillet er slut når en spiller når 3000. Spillet skal let kunne oversættes til andre sprog. Det skal være let at skifte til andre terninger. Vi vil gerne have at I holder jer for øje at vi gerne vil kunne bruge spilleren og hans pengebeholdning i andre spil.

Følgende værktøjer vil blive benyttet:

- Overleaf - parallel tekstbehandling I LaTeX
- IntelliJ - Javaudvikling
- GitHub - Versionsstyring og parallel udvikling
 - (Repo: <https://github.com/Hanskrogh/CDIO2>)
- MagicDraw - UML-diagrammer
- Javadoc - dokumentering af koden
- JUnit - enhedstest

Hovedafsnit

1 Analyse

1.1 MoSCoW Analyse

Kundens krav deles op i fire underpunkter, for at få et overblik over, hvilke krav der vil blive implementeret i denne version, samt hvilke krav vi finder nødvendige for at programmet virker optimalt.

- Krav markeret med (M) er must krav. Dvs. at kunden har specificeret disse krav, og at de skal implementeres i programmet.
- Krav markeret med (S) er should krav. Dvs. at de specificerede krav burde implementeres, men de er ikke nødvendige for programmet.
- Krav markeret med (C) er could krav. Dvs. at de specificerede krav kunne blive implementeret i programmet, men de er kun idéer og er ikke en højt prioritet.
- Krav markeret med (W) er won't krav. Dvs. at det er krav, som ikke bliver implementeret i denne version af programmet.

1.2 Krav

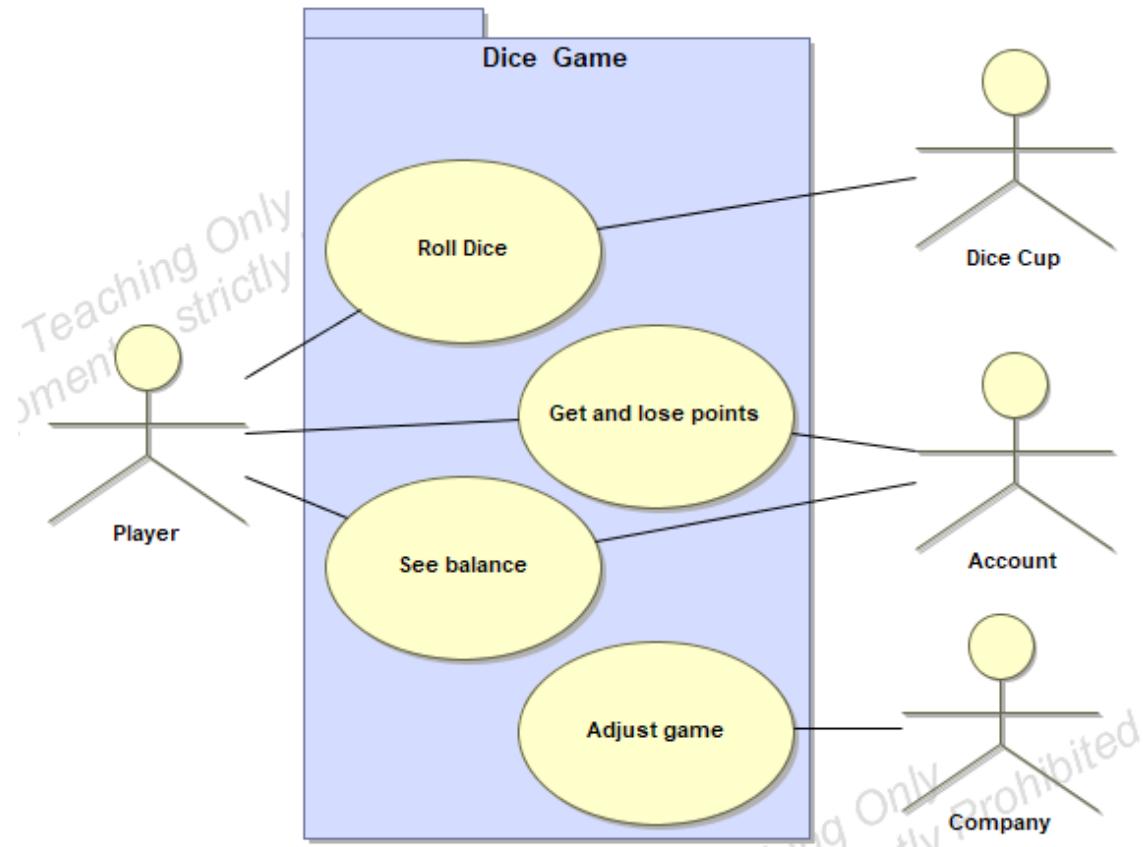
Her er kundens krav skrevet op, og der er lavet MoSCoW-analyse på hvert krav i forhold til, hvilke krav der findes nødvendige.

- (M) Spillet skal virke på DTU's databar.
- (M) Spillerne skal slå på skift med 2 terninger.
- (M) Terningernes øjne skal afgøre hvilket "felt" de lander på (2-12).
- (M) Spillerne skal starte med 1000 penge, og vinde når de har 3000 eller flere penge.
- (M) Pengene må ikke kunne gå i negativ.
- (M) Felterne afgører, hvilken ændring der sker til spillernes penge.

- (M) Felterne spillerne kan lande på skal være som følgende:

1. (Man kan ikke slå 1 med to terninger)	
2. Tower	+250
3. Crater	-100
4. Palace gates	+100
5. Cold Desert	-20
6. Walled city	+180
7. Monastery	0
8. Black cave	-70
9. Huts in the mountain	+60
10. The Werewall	-80, (men spilleren får en ekstra tur)
11. The pit	-50
12. Goldmine	+650
- (S) Spillet skal nemt kunne oversættes til andre sprog.
- (M) Spillerne og deres pengeholdning skal kunne blive brugt i andre spil.
- (S) Hvis muligt skal GUI benyttes.
- (M) Der skal efterfølgende kunne arbejdes videre på projektet fra kundens side.
- (M) Det skal være muligt at se hvem der har bidraget med hvad.
- (M) Der skal anvendes GIT.
- (M) Der skal committes, hver gang en ændring eller delopgave bliver lavet.
- (M) Der skal være en kort beskrivelse med hvert commit.
- (M) Terningerne fra sidste projekt skal anvendes.

1.3 Use-case-diagram

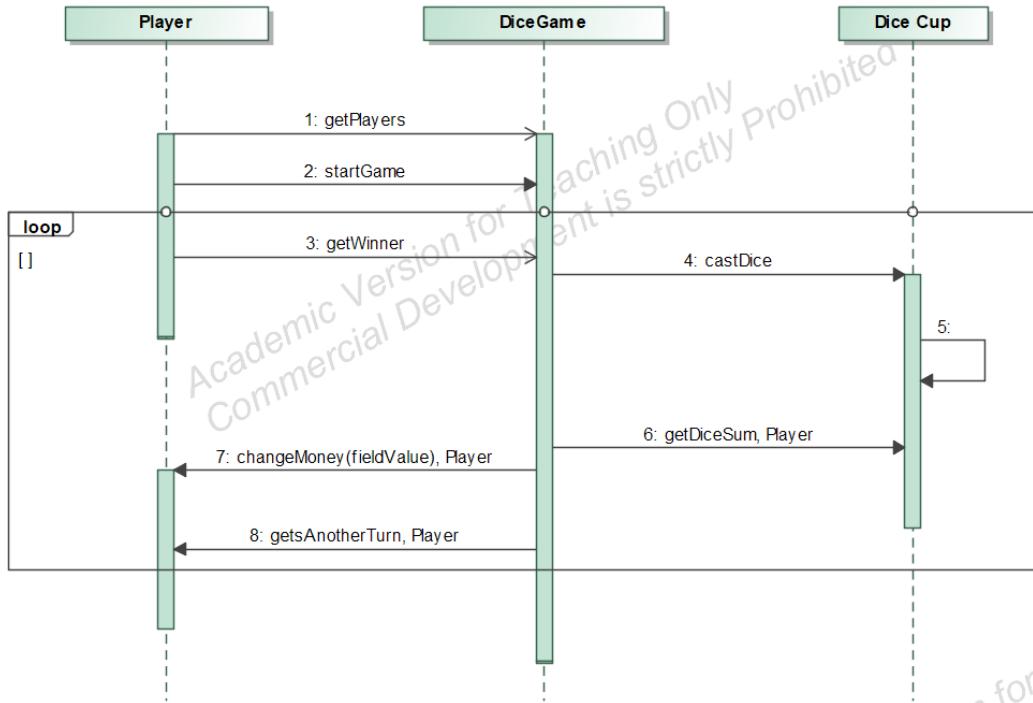


Følgende fire aktører er blevet valgt:

- Player
- Dice Cup
- Account
- Company

Opgaverne de har, er baseret ud fra viden fra sidste CDIO-projekt samt opgavebeskrivelsen fra denne.

1.4 System Sekvensdiagram

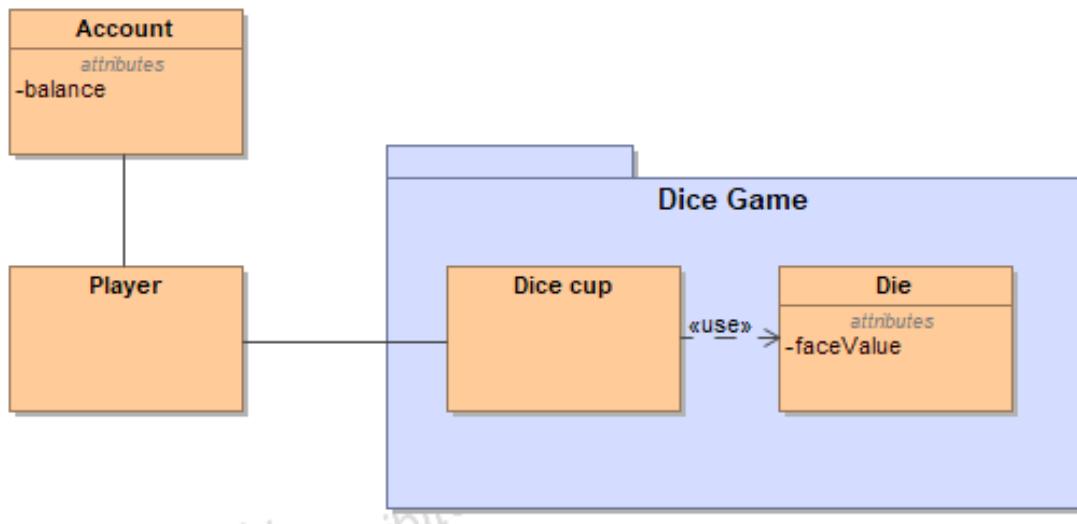


Navnene i diagrammet er baseret på et tidligt konceptstadi, så de stemmer ikke helt overens med den færdige version af produktet.

Overordnet er diagrammet tilsvarende løsningen, da spillet stadigvæk starter med at hente spillere, hvorefter spillet startes og kører et loop med terningkast på skift.

Der mangler et par funktioner, som blev tilføjet til sidst, såsom valg af sprog og antal af spillere.

1.5 Domænemodel



På domænemodellen ses, at en spiller (Player) deltager i spillet (Dice Game) med sin konto (Account).

Domænemodellen blev opstillet meget tidligt i de første faser af processen og fremstår derfor meget simpel i forhold til det endelige klassediagram.

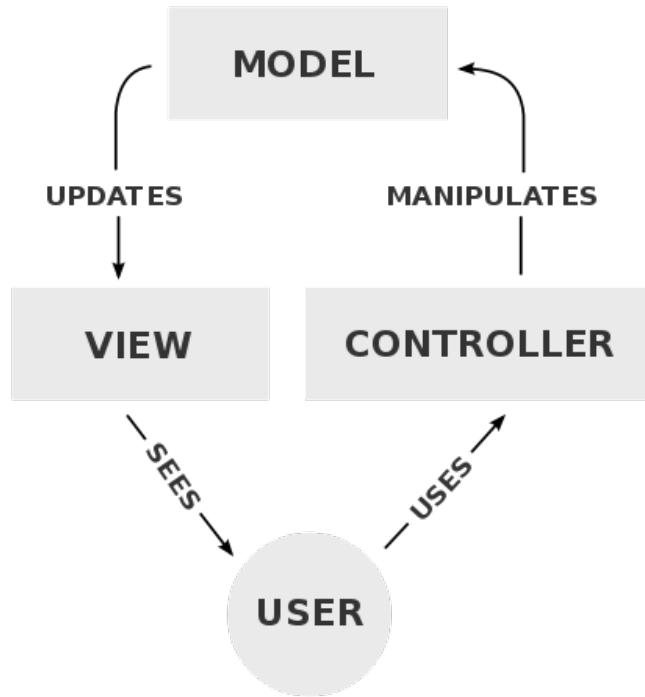
2 Design

2.1 Arkitektur

Det er blevet besluttet at basere programmets arkitektur på konceptet ”Model View Controller” (MVC).

Fordelen med denne arkitektur er i dette tilfælde, at al input og output fra/til brugeren går gennem View-klassen, således er det ikke så omfattende at implementere forskellige former for userinterface (UI). Det kræver f.eks. en begrænset indsats at skifte konsol-interfacet ud med et grafisk interface, da der kun skal ændres i View-klassen.

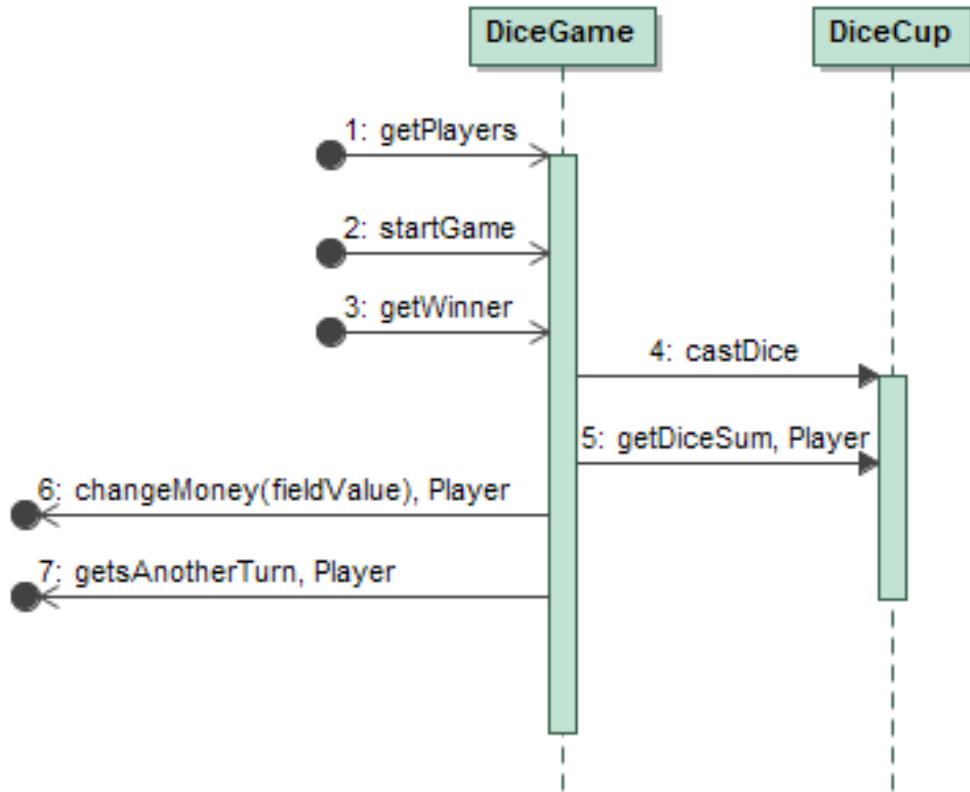
Det er yderligere muligt at have forskellige View-klasser, dermed kan man have forskellige UI's. Man kan f.eks. have et UI til anvendelse på en smartphone og et andet til anvendelse på PC.



2.2 Designklassediagram

Se bilag 1.

2.3 Sekvensdiagram



Sekvensdiagrammet er lavet ud fra klasserne DiceGame og DiceCup fra systemsekvensdiagrammet.

2.4 GRASP-mønstre

Koden er udviklet med henblik på at følge GRASP-princippet ”High Cohesion - Low Coupling” (Dansk: Høj samhørighed - lav kobling). Dette gælder for klasserne imellem, som skal være så uafhængigt fungerende som muligt, men alligevel samarbejde på bedst mulig måde. Udførelsen af dette kan ses på designklassediagrammet (bilag1), hvor ingen klasser er direkte afhængige af hinandens attributter.

Hvis man ændrer på en klasse, har det ikke effekt på de tilhørende samarbejdende klasser, mange af klasserne kan derfor ”tages ud” og genbruges i andre spil.

Koden overholder følgende andre GRASP-principper:

- Creator - Creatoren er i dette tilfælde Game-klassen, som opretter en ny instans af Controlleren (DiceGame-klassen), som så starter spillet.
- Controller - Controlleren er i dette tilfælde DiceGame-klassen, som ”styrer” de andre klasser og instanser heraf.
- Information Expert - Hver klasse indeholder kun informationer tilhørende deres ansvarsområde.

2.5 Use-cases

Der er tre use-cases til spillet: Valg af sprog, valg af antallet af spillere og navn, samt ”spil spillet”.

1. Spilleren vil spille spillet på x sprog
 - Main success
 - Ved prompten i starten af spillet, vælger spilleren enten engelsk eller dansk ved at indtaste hhv. 1 eller 2.
 - Spillet bliver sat til det valgte sprog og starter.
 - Fail case
 - Ved prompten i starten af spillet, vælger spilleren ingen af valgmulighederne, men trykker i stedet for på tilfældige knapper.
 - På et tal tryk får spilleren beskeden ”Invalid input! Try again.” og får muligheden for at vælge et nyt tal.
 - På et bogstav tryk crasher spillet.
2. Spilleren vælger antallet af spillere og giver dem et navn.
 - Main Succes
 - Spilleren vælger antallet af spillere ved hjælp af et tal.
 - Spilleren indtaster et negativt tal og bliver bedt om at prøve igen.
 - Spilleren indtaster en gyldig værdi og bliver bedt om at indtaster spillernes navne.
 - Spilleren kan indtaste lige det navn spilleren vil.
 - Fail case
 - Spilleren indtaster alt andet end et tal ved valg af antallet af spillere - spillet crasher.

3. Spilleren spiller spillet.

- Main Success
 - Efter at have valgt sprog og antal spillere, sætter spilleren spillet i gang.
 - Da der ikke er meget andet input end at trykke enter, kører spillet på skift som det skal, indtil en af spillerne når 3000 points, og den spiller så vinder.
- Fail Case
 - Spilleren spiller spillet, men istedet for bare at trykke enter, indtaster spilleren et eller flere bogstaver før enter bliver trykket
 - Antallet af bogstaver her, indikere antallet af ture der bliver taget (godt nok på skift mellem spillerne).
 - Det vil sige at hvis spilleren indtaster en lang nok række bogstaver, vil spillet slutte med det samme og give en vinder.
 - * Note: Denne bug kan ses som en feature, da man relativt hurtigt kan komme igennem spillet uden at skulle trykke enter mange gange, hvilket kan være smart, hvis man har travlt.

3 Implementering

3.1 Dokumentation

Programmet er løbende dokumenteret i Javadoc.

Den interaktive Javadoc-dokumentation er i HTML-format og fungerer som en webside. Den kan ses under mappen Javadoc i den afleverede zip-fil, hvor den åbnes ved at åbne index.html-filen i den foretrukne webbrowser.

I dokumentationen er alle klasser og metoder beskrevet på engelsk. Dermed kan der videreudvikles og vedligeholdes på koden af andre udviklingshold, danske såvel som internationale.

4 Test

Indholder citater fra CDIO1, Tests, Rasmus Søborg:

Der er udarbejdet unit-tests på klasserne Raflebæger og Terning. Der er anvendt testing-frameworket (JUnit 4). Testkoden kan ses i bilag 5 og 6.

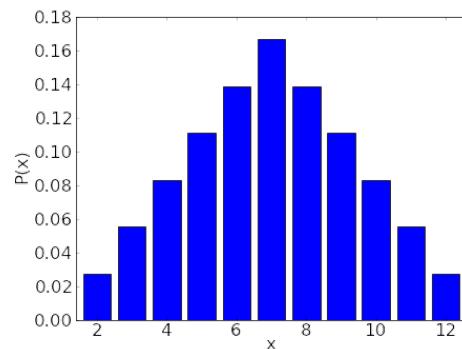
Til at teste Terning-klassen kaldes metoden “kast” 60.000 gange. Ud af de 60.000 kast kontrolleres at terningens værdi aldrig er mindre end 1 eller større end n (hvor n er antal sider som terningen har). Det kontrolleres yderligere at terningen ikke er vægtet mod enkelte numre (altså at der er en ligelig fordeling over alle numre). Det kontrolleres derfor at ud af de 60.000 kast, skal der være 10.000 ± 400 af hver øje værdi på terningen når $n=6$ hvor n er antal sider. Helt generelt må der være $k/n \pm (k/n)/25$ af hvert side af terningen, hvor n er antal sider på terningen og k er antal kast. Derfor kan vi opstille en ny test hvor $n=8$, ud fra 60.000 kast må vi så have 7500 ± 300 af hvert kast.

I Raflebæger-klassen anvendes 2 terninger. Når man kaster to terninger, må der være en teoretisk fordeling, som gør at der er 136chance for at summen af terningerne er 2, 236chance for at summen af terningerne er 3, 336chance for at summen af terningerne er 4 m.v. Til sidst er der 636chance for at summen af terningerne er 7 (størst chance). Altså kan summen af to n-sidet terningen beskrives ved funktionen:

$$Pr(X = x) = \frac{n - |x - (n + 1)|}{n^2}, x \in 2, 3, \dots, 2n \quad (1)$$

Det antages at n er et fast nummer i funktionen (i dette tilfælde 6 sider, altså $n=6$). $Pr(X=x)$ er lineær i intervallet $[2, n+1]$ og igen lineær i intervallet $[n+1, 2n]$

Herunder ses distributionen af tal for 2 terninger, hvor $n=6$. (Kilde: Nykamp DQ, “The idea of a probability distribution.” Fra Math Insight. http://mathinsight.org/probability_distribution_idea, set. d. 08/10-2018, kl 11:44)



Bilag

Bilag 1 - Designklassediagram (roteret)

