

(02312 | 02314) 02313 02315

INDLEDENDE PROGRAMMERING, UDVIKLINGSMETODER TIL IT-SYSTEMER  
OG VERSIONSSTYRING OG TESTMETODER

---

# CDIO delopgave 1

---

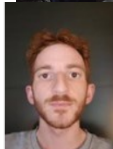
HOLD 12

Christine Reichenbach Nielsen  
s141361



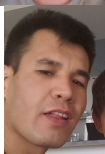
Theis Dahl Villumsen  
s195461

Amer Abdulrahman Biro  
s176356



Christian Francesco Notarmaso Pone  
s195465

Zahra Soleimani  
s170422



Mohammad Javad Zamani  
s176485

4. oktober 2019

## Indholdsfortegnelse

<b>1</b>	<b>Introduktion</b>	<b>1</b>
<b>2</b>	<b>Krav</b>	<b>2</b>
<b>3</b>	<b>Analyse</b>	<b>3</b>
<b>4</b>	<b>Design</b>	<b>5</b>
<b>5</b>	<b>Implementering</b>	<b>7</b>
<b>6</b>	<b>Test</b>	<b>8</b>
<b>7</b>	<b>Projektplanlægning</b>	<b>9</b>
<b>8</b>	<b>Konklusion</b>	<b>10</b>
	<b>Figurer</b>	<b>11</b>
	<b>Tabeller</b>	<b>12</b>

# 1 Introduktion

Dette er en rapport til programmet udviklet i forbindelse med projektet *CDIO delopgave 1*. Rapporten dækker krav, analyse, design, implementering og en smule test.

Gruppen er blevet bedt om, som en del af IOOuterActive, at udvikle et simpelt spil, hvor to spillere skiftes til at kaste to terninger. Summen af terningerne ligges til spillerens score og den første til at nå en score på 40 eller mere vinder.

Hvis Github stadig er online/finde, mens rapporten læses, vil koden være at finde på følgende link <https://github.com/notarp1/CDIO-1Gruppe12>.

## 2 Krav

Der skal udvikles et simpelt spil, så krav er der ikke mange af.  
Her er en liste med de hårde krav.

1. Spillet skal køre på Windows maskinerne i databarerne på DTU
2. Der skal være to spillere
3. En spiller slår med to terninger af gangen
4. Værdien af de to terninger ligges til spillerens score
5. Den første spiller til at nå en score på 40 eller mere vinder
6. Der skal bruges IntelliJ til udvikling og den skal være indstillet til UTF-8
7. Koden skal offentliggøres gennem Github på master-branch

Der er følgende ekstra opgaver/krav, som kunden ønsker, men som ikke er en del af de indledende krav og ikke et krav før programmet kan vurderes som værdigt.

8. Man mister alle point hvis man slår to énere
9. Man får en ekstra tur, hvis man slår to ens
10. Man kan vinde ved at slå to seksere i streg
11. Man skal slå to ens for at vinde efter man har opnået de 40 point

Af bløde krav er der følgende.

12. Almindelige mennesker skal kunne spille spillet

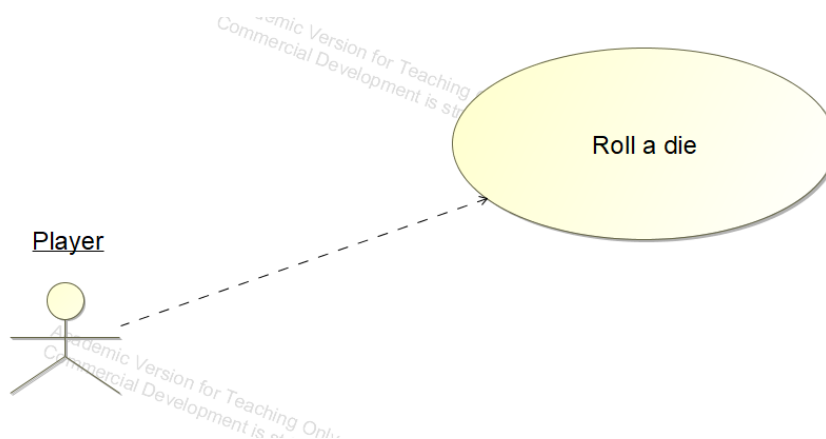
Til test er der følgende krav.

13. Der skal laves en test af rafflebægeret med 1000 kast, for at vise at de er tilfældige og normalfordelt

### 3 Analyse

Først skulle vi finde frem til om projektet overhovedet kunne realiseres. Vi vurderede vores projekt til at være næsten helt risikofrit, det bedømte vi ud fra flere faktorer. Først og fremmest fandt vi frem til at projektet var meget overskueligt og kunne i den grad færdiggøres inden for vores deadline. Det byggede vi på at vi fandt os selv faglærte nok til den nødvendige programmering, en tredje faktor er at spillet er offline og derfor behøver vi ikke at tænke på servere. Ud fra det fandt vi frem til at der ikke var grund til at lave en risk rating ud fra en PI matrix tabel, da vores eneste store risiko ville være at skuffe kunden ved ikke at få indført alle de valgfrie krav.

Da vi havde styr på kravene, kunne vi begynde at tænke på de første faser i analyse- og designprocessen. Vi brainstormede først hvad vores spil skulle kunne og oprettede ud fra det et use-case diagram (se figur 3.1).



Figur 3.1: Usecase-diagram "Play DiceGame" til programmet

Da vores program kun består af en use-case, tænkte vi at vi kunne beskrive den mere detaljeret med en casual use-case beskrivelse (se tabel 3.1).

<b>Use Case</b>	Play DiceGame
<b>ID</b>	1
<b>Brief description</b>	To spillere slår med et rafflebæger med to terninger og ser resultatet med det samme. Summen af terningernes værdier lægges til ens point. Vinderen er den der først når en score på 40 eller over.
<b>Primary Actors</b>	To spillere
<b>Secondary Actors</b>	spilfirmaet IOOuterActive
<b>Preconditions</b>	Der er en computer der fungerer
<b>Main Flow</b>	Spillere går ind på spillet og klikker “r” og derefter enter
<b>Postconditions</b>	Terningerne er kastet, terningernes værdier er vist og lagt til ens score
<b>Alternative flows</b>	Ingen

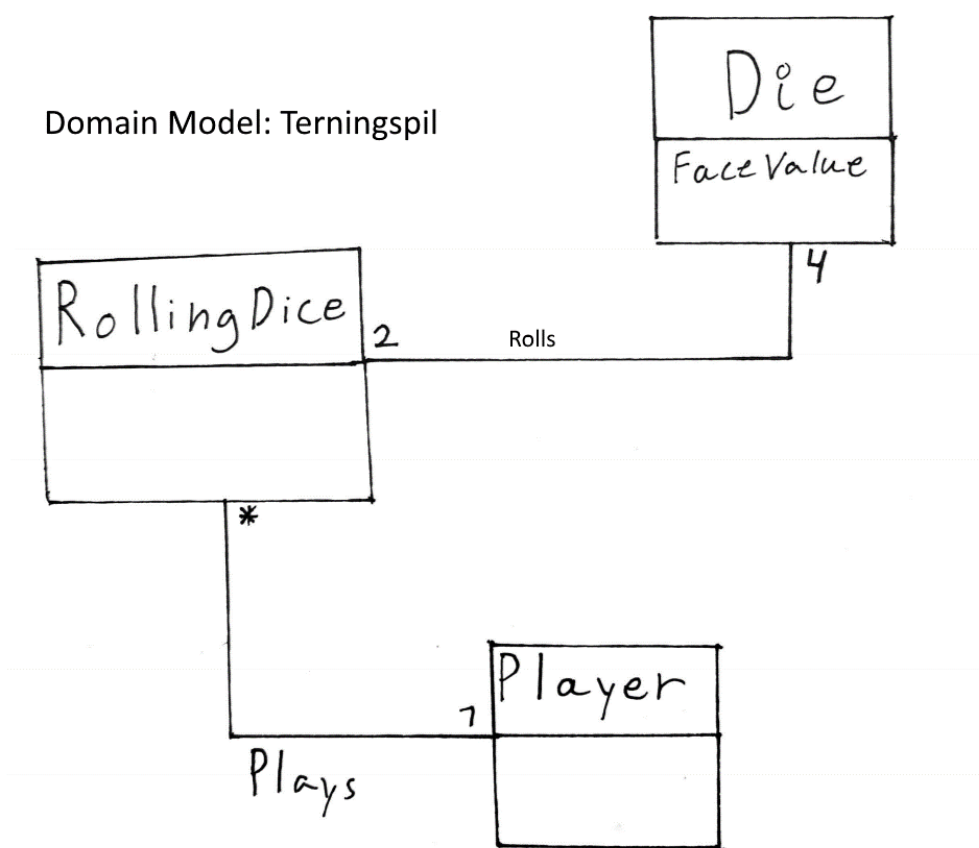
Tabel 3.1: Beskrivelse af Usecase-diagrammet i figur 3.1

## 4 Design

Da det er et krav at der skal udvikles i IntelliJ, er udviklings sproget allerede valgt; JAVA. Det betyder at der udvikles objektorienteret, hvorfor vi kan, og bør, opdele funktionalitet efter hvad det er. Da vi er firma som skal arbejde effektivt oprettede vi et fælles Github repository til at flette vores kode sammen. Ud fra vores usecase kunne vi nu begynde at tænke på klasser og kom frem til følgende:

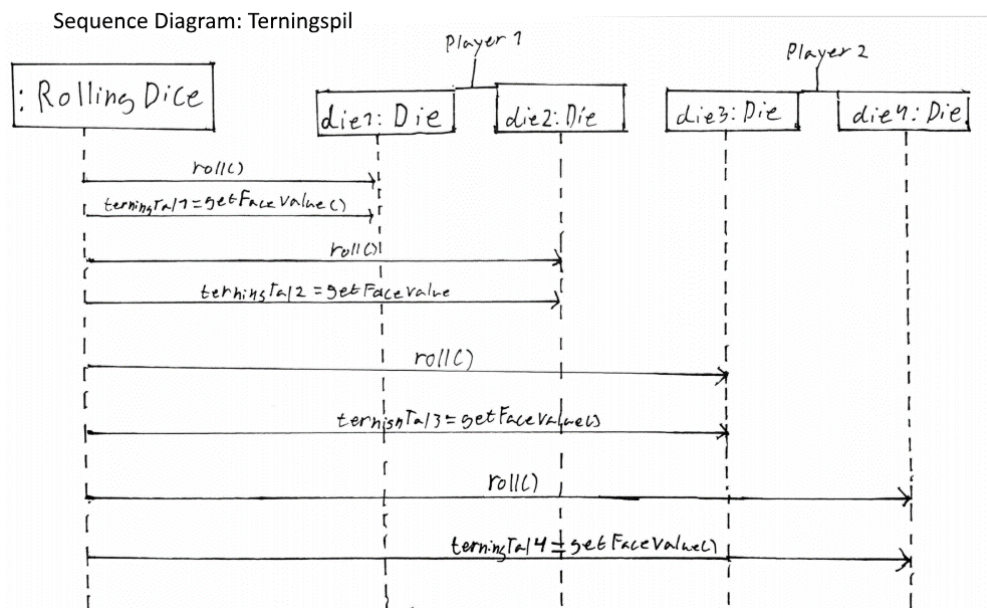
RollingDice: Vores hovedklasse der skal køre programmet og holde styr på hvad der er slået og finde en vinder. Die: En klasse der skal oprette et bestemt antal terninger, samt generere tilfældige tal imellem ét og seks. Player: En klasse der skal holde styr på spillerenes navn.

Ud fra ovenstående klasser kunne vi nu danne et overblik over hvordan vores klasser skulle opsættes via en domæne model (se figur 4.1).



Figur 4.1: Klasse-diagram til programmet

Vores domæne gav os derefter muligheden for at tegne et sekvensdiagram og få et bredt overblik over hvilke objekter og metoder vi skulle oprette (se figur 4.2).



Figur 4.2: Sekvens-diagram til programmet

Vi kunne nu konkludere at vi skal have oprettet nogle terninger, to til spiller 1 og to til spiller 2. Vores hovedklasse skulle benytte sig af en roll() metode for at generere et tilfældigt tal imellem 1 og 6. Vores hovedklasse skulle også benytte sig af en getFaceValue metode, med formål at supplere programmet med de tal som de rullede terninger viste.



## 5 Implementering

Programmet udvikles i JAVA og som IDE bruges IntelliJ, da det er disse holdet er blevet undervist i at bruge og IntelliJ har den feature, som gør at afleveringen vil overholde DTU's krav, i forhold til hvordan filerne ligger i den afleverede ZIP-fil.

Implementeringen af dataen fra vores, brainstorms, usecases og oprettede diagrammer skulle vise sig at blive mere kompliceret end forventet. Vi startede ud med at oprette vores mainclass og 'Die' class og ud fra det lavede vi en meget simpelt program der kunne slå to terninger samt, se deres faceValue og lægge summen sammen.

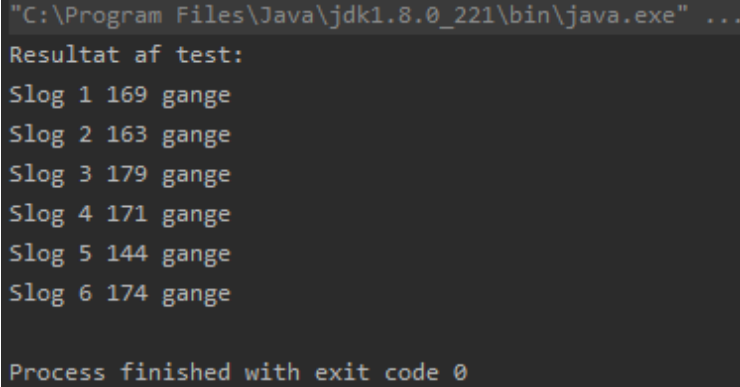
Da vi nu ville implementere en spiller2 begyndte diverse problemer at opstå med vores Player class. For at have et funktionelt program inden deadline var vi nød til at skrotte vores Player class. Ved ikke at have en Player klasse i vores program er vores kode nu mere uoverskuelig og samt sværere at modificere med nye features. Ved skrotningen af Player klassen medfulgte diverse konsekvenser, som betød vi ikke ville kunne fuldføre krav 8-10, samt kan man heller ikke bestemme sit spilnavn . At undvære krav 8-10 medførte til et spil der var forholdsvis nemt da man ikke mere skulle slå to ens efter de 40 point for at vinde. Derfor implementerede vi en ekstra regel; at hvis man slår enten to 1'ere, to 3'ere eller to 5'ere starter man forfra. For at gøre spille mere interaktivt har vi gjort det nødvendigt for spillerne at trykke på knappen 'r' for at slå terningerne, kontra at computeren ruller alle terningerne på en gang og finder vinderen på et splitsekund.

## 6 Test

Som givet i krav-spec, skal det testes at den udviklede terning er tilfældig (se krav 13).

Til dette formål er der udviklet en lille klasse *Test*, som laver 1000 kast og tæller hvor mange gange den slog de forskellige tal.

Resultatet af testen kan ses i figur 6.1 og det er tæt på at være jævnt fordelt.



```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...  
Resultat af test:  
Slog 1 169 gange  
Slog 2 163 gange  
Slog 3 179 gange  
Slog 4 171 gange  
Slog 5 144 gange  
Slog 6 174 gange  
  
Process finished with exit code 0
```

Figur 6.1: Test af programmet

## 7 Projektplanlægning

## 8 Konklusion

Holdet har udviklet programmet og skrevet en rapport efter de opstillede krav.

Programmet overholder alle vores obligatoriske krav, samt krav nr. 7 fra de valgfrie krav.

Dog er krav nr. 8-10 ikke blevet opfyldt. Vi konkluderer at vores projekt ikke har behov for at blive implementeret i UFP, fordi det er et lille projekt og vandfaldsmodel passer bedre at arbejde med. Vi konkluderer også at vi ikke behøvede at lave en Pi matrix tabel, fordi projektet er næsten risikofrit. Vi fandt ud af at vi er faglærte nok til at realisere projektet.

Vi har overholdt alle de obligatoriske krav og har opfyldt nogle af de ekstra krav. Spillet kan køre på alle Windows maskiner og kan spilles af almindelige mennesker.

## Figurer

3.1	Usecase-diagram "Play DiceGame" til programmet . . . . .	3
4.1	Klasse-diagram til programmet . . . . .	5
4.2	Sekvens-diagram til programmet . . . . .	6
6.1	Test af programmet . . . . .	8

## Tabeller

3.1	Beskrivelse af Usecase-diagrammet i figur 3.1 . . . . .	4
-----	---	---