CDIO 1

Udvikling af et terningspil

Rapporten må ikke bruges til undervisning

Medlemmer:	Studie nr.:	Gruppe nr.:
Mads Anton Schultz	s175123	36
Jacob Johan Jørgensen	s175132	
Monica Schmidt	s175130	
Nicolai Øyen Dam	s175131	
Nils David Rasamoel	s165526	

CDIO 1								
Time-regnskab	Ver. 2008-09-03							
Dato	Deltager	Analyse	Design	Impl.	Test	Dok.	Andet	Ialt
12-10-2017	Monica	4	1	0	2	2	0	9
12-10-2017	Nils	4	2	0	1	2	0	9
12-10-2017	Nicolai	4	1	0	2	2	0	9
12-10-2017	Mads	4	2	0	1	2	0	9
12-10-2017	Jacob	4	1	0	2	2	0	9

Indholdsfortegnelse

1. Indledning

Projektet har til formål at lave et terningespil mellem to personer, systemet skal kun kunne bruges på platforme med styresystemet Windows.
Terningespillet går ud på, at spillerne starter med have 0 point, spiller 1 starter med at slå med et raflebæger med to terninger, hvorefter terningernes værdi vises med det samme. Efter hvert slag lægges summen af terningernes værdi til den aktuelle spillers point. Man vinder spillet ved at opnå 40 point først. Vi vil løse denne opgave ved først at være sikre på at vi har forstået alle kravene, ved at lave en kravspecifikation. Vi vil først opstille nogle funktionelle - og ikke funktionelle krav. Herefter vil vi også kigge lidt på interessenter til vores system. For at være sikker på at vi har defineret kravene korrekt, finder vi nogle aktører af vores system, derefter opretter vi nogle use cases, med tilhørende diagrammer. Disse use cases bliver beskrevet, og sammenlignet med vores krav. Efterfølgende vil vi udarbejde en domænemodel. Før at finde på eventuelle klasser, udfører vi en navneordsanalyse, ud fra vores beskrivelser af use casene. Ud fra disse navneord opstiller vi nogle klasser, som bruges i vores domænemodel. Efter at vi har lavet vores domænemodel, vil vi udarbejde en design model, hvilket er en domænemodel tilføjet nogle metoder. Ud fra vores design model, har vi lavet vores kode. Efter at koden er programmeret, har vi udført nogle test, som viser om vores spil opfylder kravene. Vi har bl.a. benyttet positive test, og JUnit test.

2. Analyse

I analysen vil der blive foretaget forskellige artefakts som hjælper os med at specificerer kravene, og forstår hvilket område det er vi skal arbejde med. Til det har vi brugt FURPS+, use cases, og domænemodeller. Det er her, vi skal analysere os til en forståelse af det domæne vi skal udarbejde et system til.

2.1 Krav

Til at analysere krav, er der anvendt modellen FURPS+. FURPS+ bruges til at identificere de funktionelle krav og ikke-funktionelle krav.

Funktionelle krav:

1. Begge terninger skal kunne vise hele værdier mellem 1 til og med 6.
2. Systemet skal kunne oprette et spil til to personer, et spil kan ikke oprettes med færre eller flere personer.
3. Systemet skal vise spillernes point.
4. Systemet skal lægge spillernes point sammen.
5. Systemet skal oplyse når en spiller har vundet, ved at have opnået 40 point, eller derover.
6. Det skal være muligt at kunne genstarte terningespillet, når spillet er slut.

Ikke funktionelle krav (URPS+)
Usability: Menneskelige faktorer, hjælp, dokumentation

-

Reliability: Fejlfrekvens, fejl retning, forudsigelighed

-

Performance: Svar tider, nøjagtighed, ydeevne, ressourceforbrug

- Systemet skal have en lav responstid (< 0,333s)

Supportability: Anvendelighed, tilpasningsevne, vedlige holdbarhed

- Koden skal være på dansk

+:

Implementation: Ressource begrænsninger, sprog og værktøjer, hardware

- Spillet skal være på dansk
- Spillet skal kun fungere på computere
- Spillet skal ikke have en tidsbegrænsning

Interface: Begrænsninger forårsaget af kommunikation med eksterne systemer

- Systemet skal kun fungere på platforme hvor styresystemet Windows er installeret.

Operations: Systemstyring i dets operationelle ramme

Packaging: F.eks. en fysisk boks

- Programmet vil ikke have en fysisk indpakning

Legal: F.eks.licenser

2.2 Interessenter

I punktet interessenter vil der blive redegjort for hvilke interessenter, der er i forbindelse med terningespillet.

Interessenter:

- Kunden
- DTU

Vi har valgt følgende interessenter, da vi mener, at kunden har en stor interesse i spillet, da det er efter deres ønske, at spillet laves. Derfor er det også dem der bestemmer, hvad de vil have det til at kunne. Vi har også skrevet DTU på som interessent, fordi at vi i opgaven bliver informeret om, at det skal bruges på DTU's databarer. Derfor må DTU have en interesse i hvad det er for noget, de skal have på deres maskiner.

Sammenhæng mellem krav og interessent tabel:

	Kunden	DTU
Begge terninger skal kunne vise hele værdier mellem 1 til og med 6.		
Systemet skal kunne oprette et spil til to personer, et spil kan ikke oprettes med færre eller flere personer		
Systemet skal vise spillernes points		
Systemet skal lægge spillernes point sammen		
Systemet skal oplyse når en spiller har vundet, ved at have opnået 40 points		
Systemet skal have en lav responstid (< 0,3s)		
Spillet skal være på dansk		

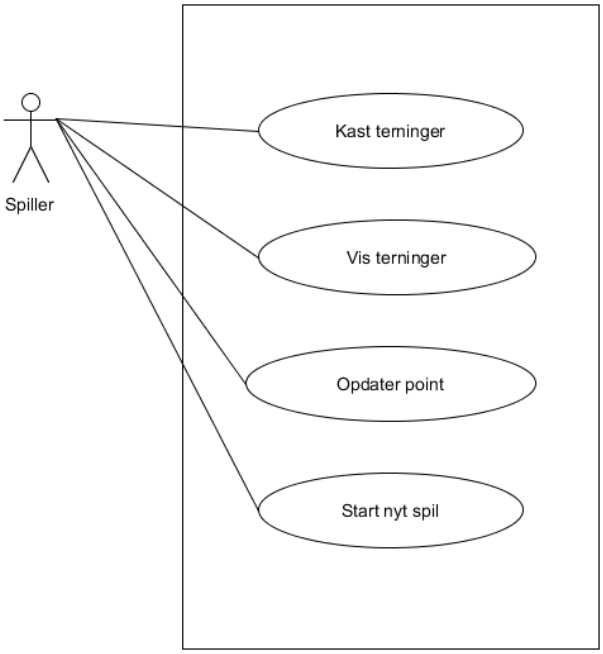
Spillet skal kun fungere på computere		
Systemet skal kun fungere med styresystemet Windows er installeret		

2.3 Use case:

Her har vi udarbejder et use case diagram, som beskriver hvordan vi regner med at aktøren (spiller 1 og 2) kan agere inde i vores spil.

Forudsætninger:

1. En forudsætning for denne Use Case er, at der kastes med to terninger, begge terninger kan vise hele værdier mellem 1 til og med 6.
2. En anden forudsætning er, at systemet selv viser vinderen af spillet, når en spiller har nået 40 point, eller derover.



Figur 1: Use-Case diagram over terningspil

Primær aktør: Spiller
Ved at bruge ordet spiller, menes der både spiller 1 og spiller 2.

Beskrivelse af aktør:
Spiller er en person, der har til formål at spille terningespillet og vinde spillet, ved at opnå 40 points først.

Aktør tabel:

	UC1	UC2	UC3	UC4
Spiller	X	X	x	x

Use Cases brief-beskrivelser
Her bliver vores 4 usecases beskrevet i en briefudgave, for at give en bedre forståelse.

Use case	Brief beskrivelse
Kast terninger	**Spiller** trykker på en **knap** for at kaste **terning**erne, terningerne "kastes" og **værdi**en beregnes.
Vis terninger	**Terning**ernes **værdi** vises frem til **spiller** med **GUI**'en.
Opdater point	**Point** beregnes efter **kast** og opdateres på **GUI**'en.
Start nyt spil	**Spiller** trykker på **knap** for at starte et nyt **spil**, **spillet** startes eller genstartes. **Point** er sat i nul og første **spiller** starter.

Krav diagram
Her laver vi et diagram hvor vi sammenholder vores funktionelle krav, men vores use cases. Det gøres for at finde eventuelle overflødelige krav eller use cases. Det kan også være vi har overset noget i en af delene, så bliver vi gjort opmærksom på det her, så fejlen kan rettes.

	UC1	UC2	UC3	UC4
K1	X	X		
K2				
K3			X	
K4				X
K5		X	X	
K6			X	X

K2 omtales under forudsætninger.

Fully dressed Use Case beskrivelse
Det er valgt at tage udgangspunkt i en Use Case beskrivelse og beskreve den ved hjælp af metoden, fully dressed Use case.

Use Case:
Kast terninger

Scoope:
Terningspillet

Level/Niveau:
Bruger mål (kast terningerne og få resultatet af kastet)

Primær aktør:
Spiller

Andre interessenter:
DTU (spillet skal spillet på DTU's computere)

Forudsætninger:
Det er en forudsætning, at man ved terningkastet med to terninger skal give en tilfældig værdi mellem 2 til 12, hvor begge værdier indgår. Terningerne har enkeltvis en værdi mellem 1 til og med 6.

Succeskriterier:
Terningerne kastes og man får resultatet med det samme.

Vigtigste succes scenarie:

1. Spilleren kaster terningerne.
2. Man får resultatet af kastet. (Tilfældigt tal mellem 2-12)

Udvidelser:

1a) Spilleren kan ikke kaste terningen.
2a) Der bliver ikke vist noget resultat.
2b) Resultat er ikke tilfældigt. Der vises eksempelvis flere 1'ere end 6'ere.

Specielle Krav:

- Når spilleren kaster terningerne skal resultatet gives inden for responstiden på 0,3 sekunder.
- Terningerne skal have hele værdier mellem 1 til 6.

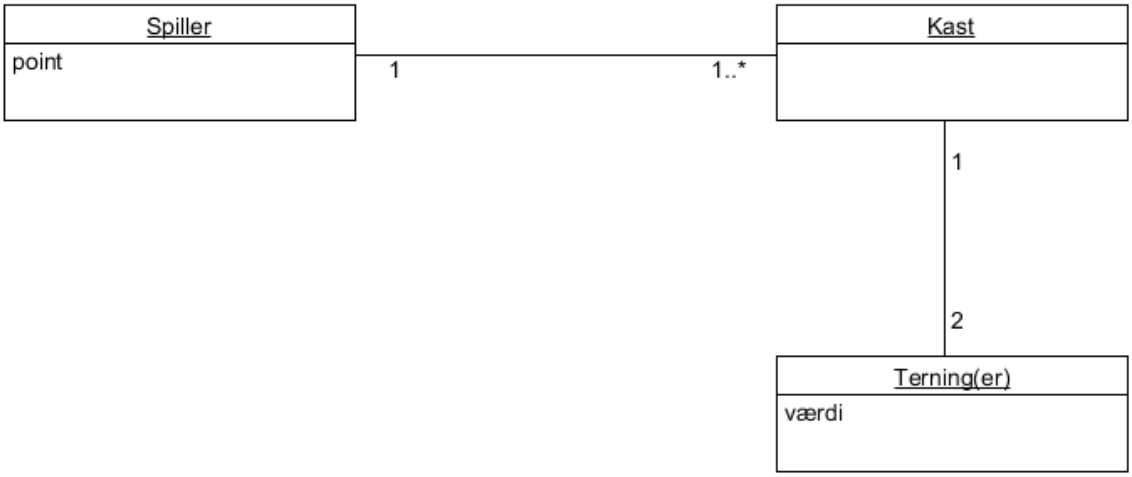
Teknologi og dataliste**:**

2a) Der skal bruges *Math.random()* metoden fra *Math* klassen i *java.lang* pakken.
2b) Der benyttes et GUI bibliotek, der har til formål at vise terningerne og deres værdi.

Frequency of Occurrence:
Use Case er en af de mest centrale ***MEST CENTRALE AF HVAD** i systemet, da spillet er afhængig af, at to spiller kaster to terninger. I et spil kastes der omkring 20-30 gange.

2.4 Domænemodel

Ud fra de tidligere brief Use Case, har vi udført en navneordsanalyse. Dette betyder at vi har kigget på samtlige navneord fra vores brief beskrivelse, for at få ideer om hvilke klasser der måske skal oprettes i vores kommende spil. Dette har medført at vi har lavet følgende domænemodel:



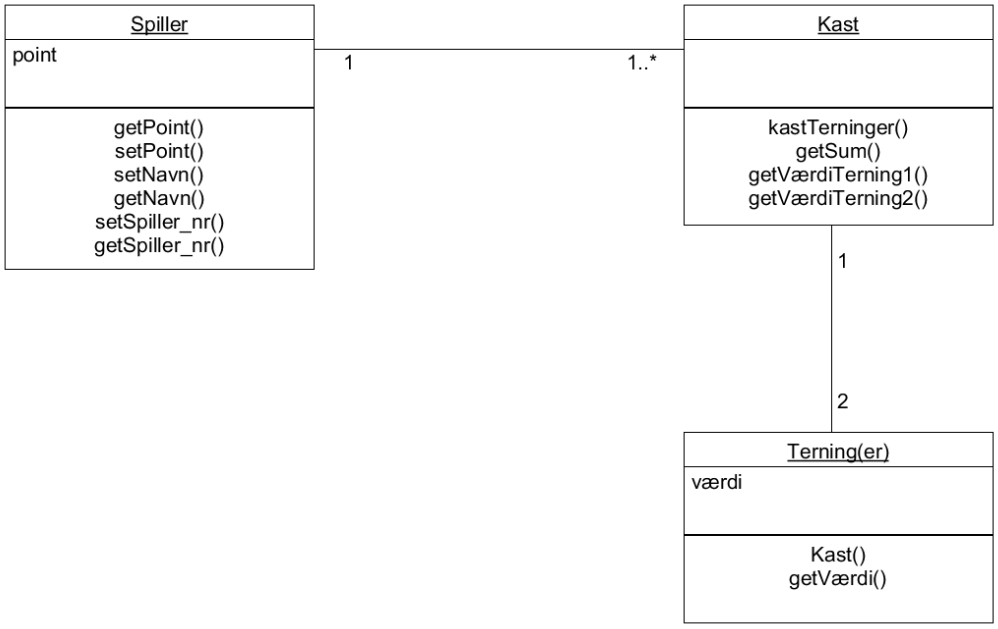
En **spiller** (med point), har en eller flere **kast**. Et **kast** har to **terninger** med værdi.

3.Design

Her vil der komme en beskrivelse af vores designmodel, som beskriver hvordan vi vil designe vores kode. Altså den softwaremæssige del.

3.1 Designmodel

Her har vi taget vores klasser fra vores domænemodel. Vi har kigget lidt på attributterne, for at se om ikke der skal bruges nogle set og get metoder for at angive og få oplyst de forskellige ting. Ud over det har vi også kommet i tanke om nogle andre ting, som vi tænker er vigtige at få med som metode i klassen.



4.Test

Her vil vi udføre en række test, som sikrer kvaliteten på vores kommende system. I opgaven var der nogle specifikke krav, til hvad der skulle testes for, de er foretaget i følgende afsnit. Samtidig har vi lavet nogle testcases, som tjekker om vores funktionelle krav er opfyldt.

4.1 Testcase

Her får vi et overblik over hvilke test, vi har udført for hvert krav. Det fremgår af tabellen at ingen test er fejlet, så det har været succesfuldt.

ID	Krav	Beskrivelse	Vejledning	Forventet output	Aktuelle output	Test date	Resultat
TC1	K1	Værdierne 1-6 skal vises på begge terninger	Indtast navn Spiller 1 -> Indtast navn Spiller 2 -> Start -> Kast terning	Begge terninger viser værdierne mellem 1-6	Terninger viste begge værdier mellem 1-6, som forventet.	10/10-2017	succes
TC2	K2	Spillet skal kun kunne spilles af to personer	Indtast navn Spiller 1 > Indtast navn Spiller 2 > Spillet starter	Det er kun muligt at spille med to oprettede spillere	Spillet kan kun startes med to oprettede spilere	10/10-2017	succes
TC3	K3	Spillet skal vise spillernes point	Indtast navn Spiller 1 -> Indtast navn Spiller 2 -> Start -> Kast terning	Spillernes point skal fremgå, ud fra et kast med terningerne	Spillernes point fremgår nede i højre hjørne som forventet	10/10-2017	succes
TC4	K4	Efter hvert slag skal pointene opdateres	Indtast navn -> indtast navn -> start -> kast terning	Det forventes at point går fra 0, til det antal øjne der er slået	Spillerens point blev opdateret, med den værdi øjnene viste	10/10-2017	succes
TC5	K5	Spillet skal oplyse når en spiller har vundet, ved at have opnået 40 point	Indtast navn -> indtast navn -> start -> 1. Spiller 1 kast terning -> 2. Spiller 2 kast terning Step 1 og 2 forsættes indtil en spiller har opnået 40	Det forventes at spillet oplyser spillerne, når en har opnået 40 point	Da den første spiller kom på over 40 point, oplyste spillet at vedkommende havde vundet	10/10-2017	succes
TC6	K6	Det skal være muligt at genstarte spillet,	Indtast navn -> indtast navn -> start -> 1. Spiller 1 kast terning ->	Det forventes, at når en spiller har vundet, så kommer der	Efter endt spil, kom en knap med genstart, som så kunne benyttes til at	10/10-2017	succes

		efter endt runde	2. Spiller 2 kast terning Step 1 og 2 forsættes indtil en spiller har opnået 40 -> Genstart	en knap med genstart, som kan benyttes	genstarte spillet		

4.2 Traceability matrix

Traceability matrixen er med til at give et overblik over, om der optræder en eller flere testcases som er irrelevante i forhold de tilhørende krav. Hvis det er er to eller flere testcases som opfylder de samme krav, kan det sammenfattes til en testcase.

	K1	K2	K3	K4	K5	K6
TC1	X					
TC2		X				
TC3			X			
TC4				X		
TC5					X	
TC6						X

4.3 JUnit test

For at test programmet, terningespil, er der foretaget to JUnit test.
Den første JUnit test havde til formål at undersøge, at værdierne af terningerne er forholdsvis lige fordelt, ved at foretage 1000 terningekast.
JUnit testen viser, at værdien 7, er den værdi som opstår hyppigst, da det er den værdi der er størst sandsynlighed for at slå. Det er der fordi at 7 er det tal, som kan fås med flest forskellige tal på de to terninger.

Den anden JUnit test havde til formål at undersøge hvor mange gange de to terninger viste det samme antal øjne. Vi udregnede at hvis der kastes med 2 terninger 36 gange, så vil der være en sandsynlig på 6 ud af 36, hvor de to terninger vil være ens. Så sagde vi at 36, cirka går 3 gange op i 100, og derfor ganger vi også 6 med tre. Derudover har vi tilføjet en lille fejlmargin på 2, da det ikke vil give mening at regne med fejlprocenter ved så lave tal.

Begge JUnit testene, kan findes i pakken test, i vores javakode.

5.Brugervejledning

For at komme i gang med spillet skal man først skrive navnet på Spiller 1, derefter skal man skrive navnet på Spiller 2, efterfølgende vil spillepladen komme op på skærmen.
For at komme i gang med spillet, trykkes der på knappen ”start”, som er placeret oppe i midten. Efter at der er blevet trykket på ”start”, kommer en besked der fortæller hvis tur det er.
Når spilleren er klar til at kaste terningerne, trykkes der på ”kast”. Knappen ”kast” er placeret samme sted som ”start” knappen. Efter at der er blevet trykket, kan man nu se værdien af terningerne, samtidig opdateres spillerens point, når der foretages et kast. Når spiller 2 er klar, kan han slå ved at trykke ”kast”.
Dette vil så fortsætte, indtil en spiller har opnået 40 point, eller derover, spillet fortælle herefter, at der er fundet en vinder, og hvem vinderen er. Herefter er det muligt at genstarte spillet, med nye spillere.

6.Konklusion

Projektet havde det formålet at programmere et terningespil til to personer, hvor de skiftevis skulle slå med to terninger, den spiller der først opnåede 40 point eller derover var vinderen.

Det konkluderes at programmet, terningespil er en succes, da spillet udfører de opstillede krav fra analysen. Yderligere er der foretaget test ved hjælp af JUnit. JUnit testene viste, at inden er værdierne lige fordelt.