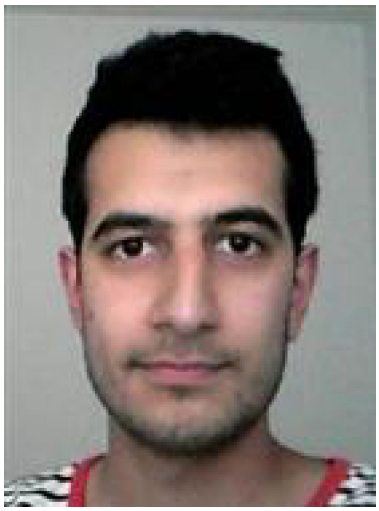
Projektnavn: CDIO 1: Terningespil
Gruppenummer: *18*
Afleveringsfrist: *Fredag den 7/9 2016 Kl. 23:59*
Fag og retning: Diplom Softwareteknologi
Denne rapport er afleveret via Campusnet (der skrives ikke under).
Denne rapport indeholder x sider inklusiv denne.

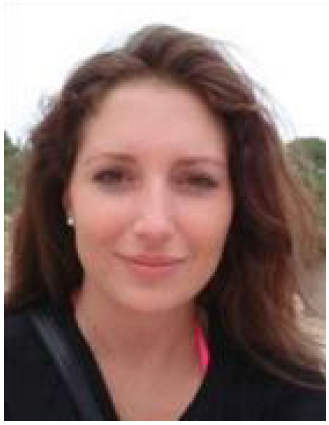
Kasper Leiszner s165218

Bijan Negari s144261



Frederik von Scholten s145005

Helene Zgaya s104745



Troels Lund s161791

Timeregnskab:

Dato	Deltager	Design	Implementering	Test	Dokumentation	Andet	I alt
07-10-2016	Helene	5	6	3	9	0.5	23.5
07-10-2016	Kasper	1	10	6	5	0	22
07-10-2016	Troels	6	8	2	4	2	22
07-10-2016	Frederik	8	3	3	9	0.5	23.5
07-10-2016	Bijan	5	7	4	6	0	22
	Sum	28	34	18	31	1	113

Resumé

Vores projekt går ud på at udvikle et spil for to spillere der på skift skal slå med to terninger, og se hvem der først der kommer til 40 point. Vi formåede at lave alle opgaver med CDIO-metoden, men ikke ekstra opgaverne.

Indhold

Indledning

Denne rapport er udarbejdet på det grundlag at udvikle et spil efter forespørgsel fra kunde. Spillet foregå mellem to spillere hvor man på skift kaster med to terninger via et raflebæger. Vinderen er den der først når 40 point. Flere funktioner for dette spil er beskrevet senere i rapporten og vil blive implementeret såfremt der er ressourcer til det.

På baggrund af ovenstående har vi udviklet en kravspecifikation som er dannet via kommunikation mellem kunden, projektlederen og os, så vores program kan leve op til kundens forventninger.

Vi har først og fremmest ønsket at danne et overblik over hvordan spillet skal udvikles, hvorfor der er inddraget flere modeller der beskriver fremgangsmetoden for programmet. Disse er beskrevet længere nede i rapporten. Vi skriver i det objektorienteret programmeringssprog java.

1. *Vores primære mål er altså at udvikle spillet med dets basale funktioner der SKAL (på engelsk shall) være med i programmet.*
   Til udførelse af punkt 1 ønsker vi at lave modeller og diagrammer der visualiserer udviklingen af spillet.
   Vi vil desuden designe vores egen brugerflade (GUI).
2. *Sekundært implementeres ekstra funktioner til spillet (udføres KUN så længe der er ressourcer til det)*

Hovedafsnittet vil gennemgå ovenstående i detaljer. Der vil til sidst fremlægges en diskussion om hvad vi har formået at opfylde og hvad der eventuelt kunne være gjort bedre. Rapporten afsluttes med en konklusion på hvorvidt vores mål for dette projekt er nået.

Hovedafsnit

Det væsentligste for denne opgave er at klargøre hvad vores program skal kunne ifølge kunden. Dette er udformet i nedenstående:

Kravspecifikation

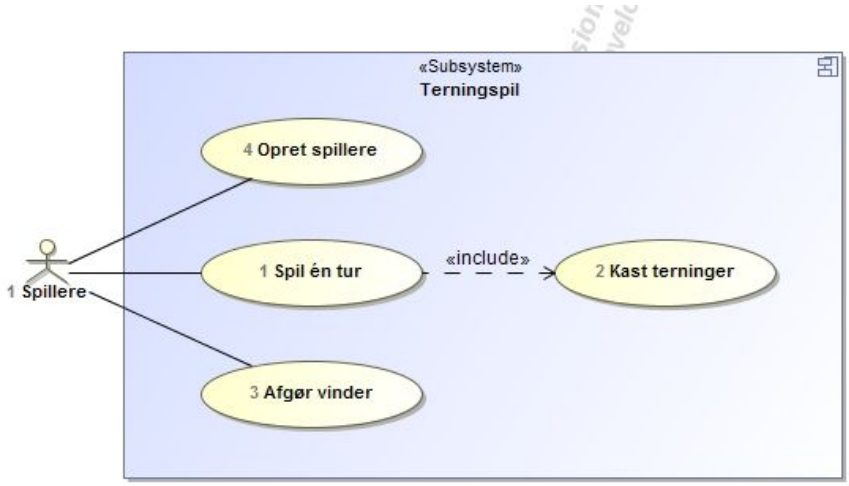
Ud fra Kundens vision er specifik kravspecifikation blevet udviklet for at undgår misforståelser samt have en klar forventningsafstemning til systemets opbygning og funktionalitet.

- Systemet skal kunne bruges på maskinerne i databarerne på DTU.
- Systemet skal være et spil mellem to spillere.
- Spillet skal kunne slå/rafle/kaste med to terninger, udregne summen af de to terninger, og lægge summen af terningerne til henholdsvis spiller 1 og spiller 2's samlede point.
- Undervejs i spillet, skal spillernes slag og point vises på GUI'en gennem hele spillet.
- Spillerne skiftes til at slå med terningerne.
- Hvis en spiller slår to 1'ere, mister den spiller alle sine point.
- Slår en spiller to ens, får den spiller en ekstra tur.
- Slår en spiller to 6'ere, to gange i træk, vinder den spiller spillet med det samme.
- Spillet kan kun vindes ved at slå to ens, efter at man har opnået 40 point.
    - Man kan dog ikke vinde på 2 ens 1'ere, men man får en ekstra tur.
- Spillet skal være nemt at spille. DVS. at enhver skal kunne spille spillet uden at læse en brugsanvisning.
- Terningerne skal virke korrekt, dvs. at fordelingen af slag skal være fuldstændig tilfældige, på en almindelig terning med 6 sider.
- Ved siden af spillet skal det bevises ved test at terningerne virker korrekt.

Diagrammer

For at danne overblik har vi udformet flere diagrammer med tilhørende beskrivelser:

Use Case diagram



I diagrammet ses tre primære usecases (opret spillere, Spil én tur, Afgør vindere). "Kast terninger" er inkluderet i "Spil én tur".

Use case beskrivelser

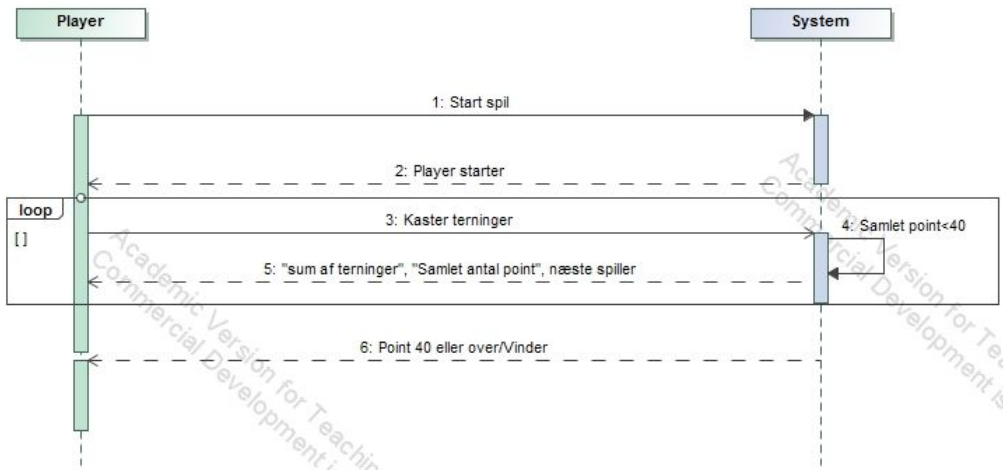
Use case: Spil én tur
ID: 01
Beskrivelse: Spillerne spiller én tur i terningespillet.
Primær aktør: Spillere
Sekundær aktør: Ingen
Precondition: 1. Start nyt spil
Main flow: 1. SpillerX har tur 2. Include (Kaster terninger) 3. Terninger returner værdi for hver terning 4. Summen af de to terninger udskrives 5. Summen af terningerne lægges til Spillers point score 6. Venter på næste tur

Postcondition: 1. Spillers tur afsluttet
Alternativ flows: None

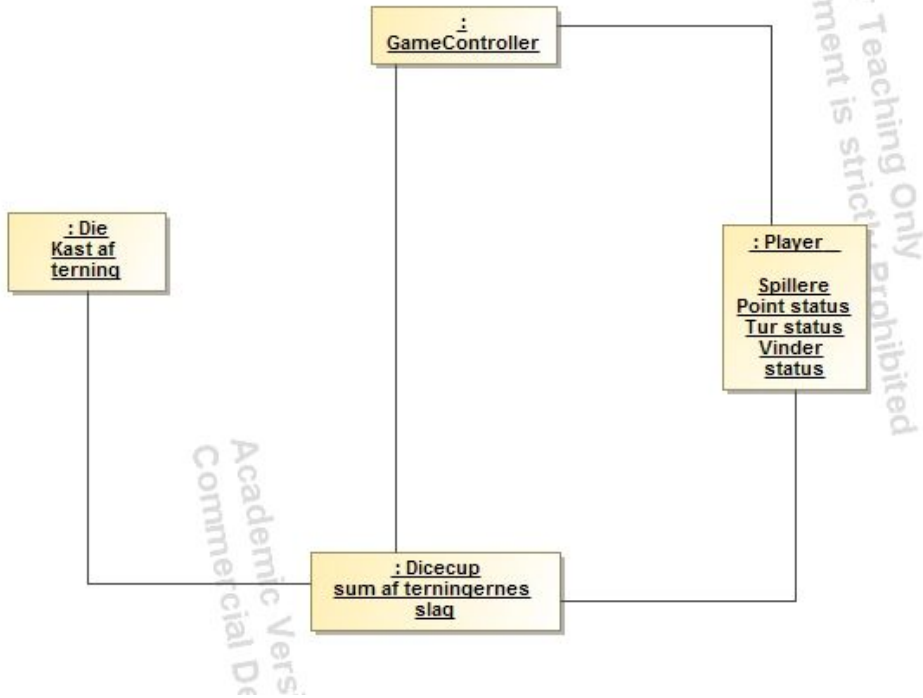
Use case: Kaster terningerne
ID: 02
Beskrivelse: Terninger returnere tilfældig værdi for terning 1 og 2 og returnere værdierne
Primær aktør: Spillere
Sekundær aktør: Ingen
Precondition: 1. SpillerX tur
Main flow: 1. Terning 1 returnere tilfældigt tal mellem 1 og 6 2. Terning 2 returnere tilfældigt tal mellem 1 og 6
Postcondition: 1. Terninger returner værdi for hver terning
Alternativ flows: None

Use case: Afgør vinder
ID: 03
Beskrivelse: Afgør om der er en spiller der har vundet. Spillet afsluttes hvis der findes en vinder.
Primær aktør: Spillere
Sekundær aktør: Ingen
Precondition: Spillers tur afsluttet
Main flow: 1. Undersøger om en spiller har 40 eller flere point 2. Hvis så afgøres vinder
Postcondition: 1. Vinder afgjort sandt eller falsk
Alternativ flows: None

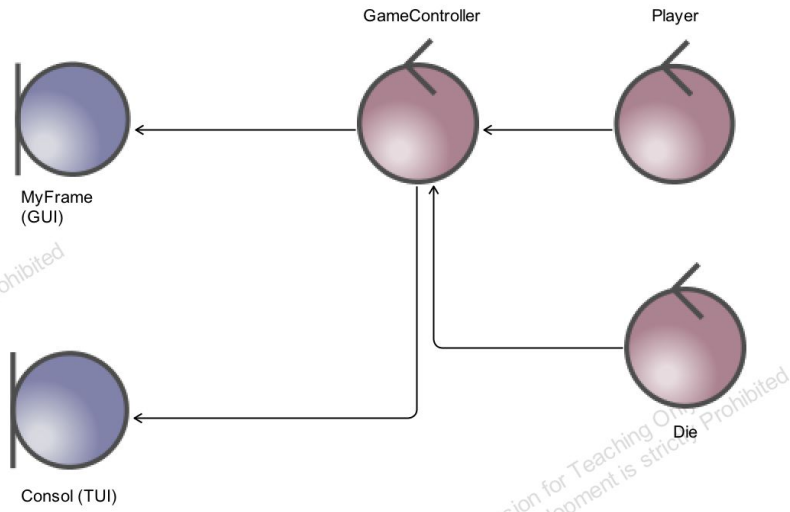
System sekvens analyse



Domænemodel

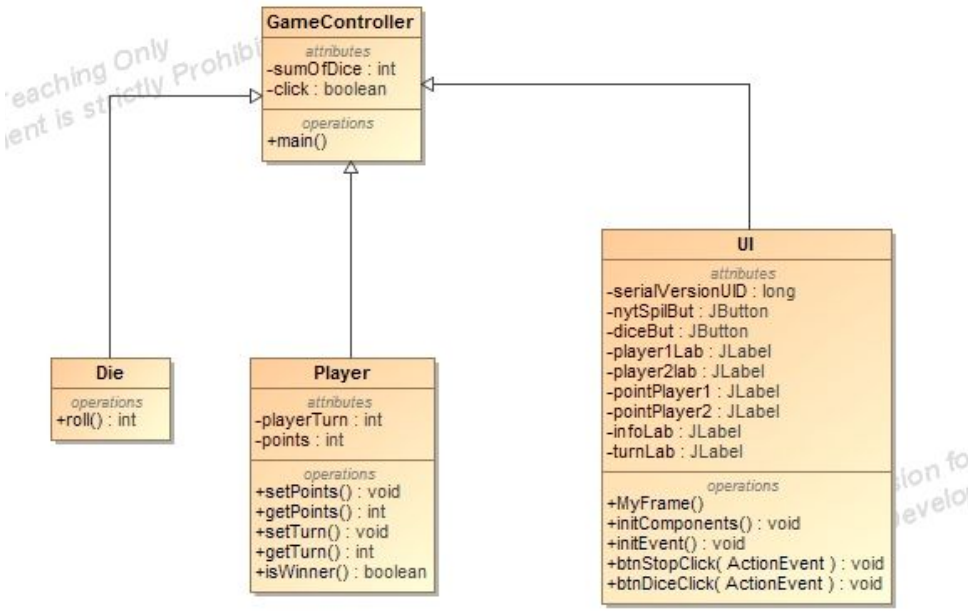


BCE model



Vores GUI, MyFrame samt udskriften i Konsollen står for at håndterer interaktion omgivelser. Game Controlleren fordeler ansvaret videre ud på henholdsvis Die og Player Objekterne.

Klasse-diagram



Test

Vi tester for at dokumentere at programmet fungere efter hensigten og sikre kvaliteten af koden. Til at teste programmet bruger vi JUnit test. JUnit

Test af terningens ægthed samt mulige værdier.

Vi vil teste at terningen kun kan returnere værdier mellem 1 og 6, da det er en normal 6 sidet terning. Derfor returner vores JUnitet test "false" ved værdier uden for intervallet. Testen returnere dog "true", og vi må dermed konkludere at terningen kun returnere 1 til 6.

Vi vil undersøge ægtheden af terningerne. Altså at fordelingen af terningens værdier er fordelt ligeligt. For at undersøge ægtheden af terningen er der brugt en JUnit test for fordelingen af de seks mulige værdier (terningens 6 sider). vi "ruller" terningen 1000 gange for at få et fornuftigt billed af om vores terning inden for den statistiske unøjagtighed, har lige stor sandsynlighed for at lande på 1-6. Dette gøres ved at forventet 166 gange på hver, hvor vi accepterer en afvigelse på +-40.

JUnitet testen returnerer true, og derved har alle værdierne inden for intervallet [126,206] og derved kan vi konkludere at vores terning er ægte.

Konklusion

Vi har udviklet det efterspurgte spil med alle basis kravene opfyldt. De ekstra valgfrie funktioner er ikke blevet implementeret i programmet. Havde vi planlagt vores tid bedre, ville programmet have været skrevet pænere og alt i alt have været bedre, og indeholde de ekstra funktioner og dimensioner til spillet.

Litteraturliste

GUI (MyFrame) - https://gist.github.com/arbo77/3318971

Applying UML and Patterns by Craig Larman, Pub Date: October 20, 2004, tredje udgave. Publisher Addison Wesley Professional.