62531, 02312, 02314, 62532

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

CDIO-1

Gruppe 33 - SWT

Frederik Farsøe Bank

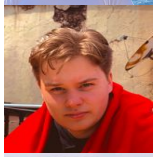
s225820

Peter Hannibal Hildorf

s224274

Nicolai Sandberg Rongsted

s221677



Haseeb Kamran

s224270

30. september 2022

2. Indholdsfortegnelse

3. Resume:

Problemstillingen i denne opgave var at identificerer, de objekter man skulle bruge til at lave koden, for at få det til at indgå under objekt orienteret programmering, derudover ville der naturligt ske merge-konflikter når man arbejdede med git. Det er vigtigt at benævne vigtigheden ved projekt-planlægningen.

Projektets forløb skete via. vandfaldsmetoden, hvilket vil sige alle opgaver blev løst som gruppe. Der blev lavet 3 objekter som benævnes senere i opgaven, som i sidste ende samledes til et endeligt spil, i mainen. Derudover er der en gennemgående analyse af risiko og en use-case i både casual form og fully dressed. Designet er blevet beskrevet i både diagrammer og med tekst.

4. Timeregnskab:

Vi har hhv. brugt både 6 timer på kodningen og 6 timer på at lave analyser og skrive rapporten.

5. Indledning:

Objektorienteret programmering er vigtigt at forstå og lære, da jo større projekter man skal jo mere kodninger er der. Dette kan derfor være en løsning, da det skaber overblik, organisering, og stor forståelse for hvordan koderne hænger sammen. Dette ser man i dette projekt hvor der

netop bliver lavet forskellige objekter til at få koderne til at hænge sammen. Projektet indeholder planlægningen, hvilke krav der er fra firmaet IOOuteractive, en analyse hvor man kigger på alt hvad IOOuteractive forventer, og hvordan spillet skal kodes, herefter et design af kodningen. Til sidst ser man hvordan det er testet og man ser en stokastisk variabel af tilfældigheden. Der er anvendt materiale fra:
Java Software Solutions (Foundations of Program Design) ninth edition - Lewis Loftus.
Applying UML And Patterns (Third edition) - Craig Larman

6. Projekt-planlægning:

I vores projekt har vi lavet, en risikoanalyse som kommer ind på hvor hvilke risiko vi vægter højest og lavest. Herefter har vi planlagt hvordan vores sekvensdiagram skal laves hvor vi så viser hvordan vi har kodet spillet på en visuelt plan.

Så har vi udnyttet vandfaldsmetoden, da vi alle i gruppen har svært ved at kode. Vi løser opgaverne sammen og en af gangen, sådan sørger vi alle har en fælles forståelse af hvordan vi har lavet vores projekt.

for at vi skal forstå projektet opgave, har derfor også lavet en navneords analyse, for at være helt sikker på, hvad er det vi skal lave til spillet. Derfor laver vi et klasse diagram, som viser os hvilke klasser vi skal have, og kode.

Herefter har vi lavet en stokastisk diagram som vil bevise vores spil er bygget efter elementet "tilfældigheden" da det er med to terninger man kaster.

7. Krav:

Et spil som kan køre i databarerne på DTU. Hvilket er java version 18
Det skal være et spil mellem 2 spillere.
Man skal kunne slå med to terninger og se resultatet med det samme.
Summen af terningerne skal lægges til ens point.
Vinderen er den der når 40 point først.

Ekstra krav hvis ressourcer haves:
Spilleren mister alle sine point hvis man slår to 1'ere.

Spilleren får en ekstra tur hvis man slår 2 ens.

Spilleren kan vinde spillet ved at slå to 6'ere, hvis spilleren også i forrige kast slog to 6'ere uanset om det er på ekstra kast eller i forrige tur.

Spilleren skal slå to ens for at vinde spillet, efter at man har opnået 40 point.

8. Analyse:

Vores primære aktører og superbrugere er spillerne, og vores sekundære er IOOuterActive der har bedt os om at lave spillet. Vi har også en ikke human aktør i DTU's hjemmeside, da spillet skulle kunne spilles fra deres databar, og derfor kræver det at DTU hjemmesiden fungerer for vores spil kan spilles. Så ville deres server så også fungere som vores passive aktør. Vores spil skal kunne fungere sammen med DTU's hjemmeside og Windows da det er et af kravene fra IOOuterActive. Kravet for at spillet skal kunne spilles på DTU's computere er så at compile vores java version til 18, da vores er up to date, hvilket er version 19.

Herefter har vi lavet en navneords analyse, hvor vi fik de 3 navneord: Spiller, terning og raflebæger, hvorefter vi lavede et klassediagram, og sat dem i relation til hinanden. Dette brugte vi til at opbygge vores tre objekter der bliver brugt i koden. (se bilag 1.2)

Use Case analyse:

Casual Form:
Spillet fungerer så 2 spillere slår med to terninger hver, derefter bliver hver spillers terninger lagt sammen, summen af de to terninger lægges til spillerens point. Spillerne bliver så ved med at slå med terningerne indtil, en af dem når 40 point. Derefter afsluttes spillet.

Fully Dressed

Use Case navn
- Terningspil

Scope
- Spillerne

Level
- at 2 spillere kan spille mod hinanden i et terningspil om hvem der først får 40 point.

Primære aktører
- Spillerne

Stakeholders and Interests
- Spillerne og IOOuterActive.

Preconditions
- At kunden skal kunne spille spillet med to spillere, hvor koden kaster to terninger for hver spiller, spillerne skal kunne dyste til 40 hvorefter den der først rammer 40 vinder.

Main success scenario

- Kunderne får deres spil, de spiller det uden konsekvenser.

Extensions

- Kunderne får deres spil, de spiller det uden konsekvenser, men vinderens skærm bliver grøn efter han vinder.

Special Requirements

- 2 spillere
- spillet vindes ved 40 point
- 2 terninger pr. spiller
- Den skal fungere på windows

Technology and Data Variations list

- Spillere skal kunne genstarte spillet og starte på ny

Frequency of Occurrence

-

Open issues

- Skal spillere selv kunne vælge hvad de spiller til og hvor mange runder de spiller?

Risikoanalyse

Denne risikoanalyse er lavet ud fra Likelihood x severity skemaet (Bilag 1.3)

Projekt	-//-		
Hvad kan gå galt?	Risiko S x K = R	Tiltag der reducerer sandsynligheden	Konsekvens x Sandsynlighed
Spillet slutter ikke ved 40 point, men bliver ved.	S= 1 K=1	Det er skrevet i kode, så da koden fungerer en gang skulle den gerne blive ved med at fungere	R=1
Spillerne kan ikke finde ud af at spille spillet	S= 1 K=3	Man skal trykke på en knap, sandsynligheden er meget lille for spillerne er dumme nok til ikke at kunne finde ud af det. Derefter ville de kunne lære det relativt nemt.	R=3
Terningerne dur ikke normalt	S= 5 K= 1		R= 5

Prioritering af risici

Projekt			
Hvad kan gå galt?	Konsekvens for projektet (0-5)	Sandsynlighed (%)	Konsekvens x Sandsynlighed
Spillet slutter ikke ved 40 point, men bliver ved.	1	0-20%	
Spillerne kan ikke finde ud af at spille spillet	3	0-20 %	
Terningerne dur ikke normalt	3	0-20%	

Vi er klar til acceptere alle vores risici, da sandsynligheden er næsten lig nul, vi har gennemført nok test af programmet uden fejl, eller afbrydelser.

Vi kan forebygge risikoen om, at spillerne ikke kan finde ud af spillet, ved at lave en manual til at forstå spillet.

I et særtilfælde hvor en af fejlene opstår ville vi henvise spillerne, om at genstarte koden, eller downloade den igen. Det skulle gerne genstarte alt fra start og dermed være korrekt.

9. Design:

I vores kodning har vi lavet 3 objekter, rafflebæger(raffle), terning, spiller(Player), vi har lavet disse 3 objekter da det var nævnt at det skulle projektet indeholde. I terning objektet, indeholder kode som genererer tilfældige værdier, for at demonstrere at man kaster terningen, og at det bliver tilfældig værdi man får, hvor minimum kan få 1 point og max. 6.

Rafflebæger objektet skal demonstrere at der er to terninger i objekter som den “roller”, og giver os de værdier som terningerne har givet os, og som derfra bliver samlet til en sum.

Spiller objektet har anvendt til at lave en profil for en spiller. Hvor spilleren starter med 0 point, og får sine point ved at “rolle” med rafflebægeren objektet.

I selve spillet har vi så anvendt disse 3 objekter, for at få det hele til at hænge sammen. Her har vi så sat reglerne for spillet, og hvordan spillet skal spilles. Reglerne vi har sat er at spillet stopper når enten spiller 1 eller spiller 2 får 40 point. Måden vi spiller spillet er at de skiftevis roller raffle bægeret og ser hvad terningerne bliver. Som der blev nævnt i projektet er der

også ekstraopgaver. Vi fik lavet ekstra opgave 2, som går ud på at hvis man slår to ens, får man en ekstra tur.

Sekvens diagrammet viser os hvordan vi har kodet vores projekt, og som man kan se viser det helt basalt hvordan et spil fungere med to terninger. Man kaster den ene terning og den anden terning, og de to terninger giver os to forskellige summe. Denne forståelse har vi også anvendt i vores kode, og fungere præcis på samme måde. (Bilag 1.5)

Ud Fra vores domænemodel af delsystemerne i terningspillet, kan vi se de forskellige relationer. Terninger kan give min. 2 point og max. 12 point og man kan min. få point af 1 terning og max 2 terninger. Det samme gælder for spillere. Spillere har max 2 terninger og min. 1. og der er min. 1 spiller der kaster terningerne eller 2 spillere. Så der relationen mellem spillere og point.
spillere kan min. få 1 til uendeligt, da de kan blive ved med kaste terninger og få point. Point kan max blive til 40, da det er hvad spillerne spiller til. relationen mellem point og runder, er at man max kan have 20 runder, før man får 40 point. (Bilag 1.4)

10.Test:

Vi skrev en java class, i intelliJ (TerningTest.java), hvor vi fik den til at køre en terning og kastede den 1000 gange. Hvorefter den opsamlede hvor mange gange den landede på hver side af terningen. Det givne data satte vi ind i Maple, hvor vi udregnede chancen for at ramme siden x antal gange. (se bilag 1.1)

11. Konklusion:

Efterfølgende opgaven har vi fået større forståelse for hvad det vil sige at lave objektorienteret programmering, da vi i løbet af de sidste par uger, har hoppet meget rundt i vores kode, fordi vi ikke forstod den givne problemstilling, til eftertanke, vil vi ligge langt større vægt på vores projekt planlægning. Vores tilgang til næste projekt vil klart være anderledes, da det har skabt os store problemer da vi ikke havde en klar tilgang til vores projekt, vi skulle have brugt en iterativ fremgangsmåde til vores kodning, frem for den vandfaldsmetode som vi brugte. I starten af projektet lavede vi kun et enkelt objekt som vi kaldte terningen, men efter hjælp fra hjælpelærere, og en navneords analyse forstod vi vores misforståelse og skulle arbejde forfra.
Dog er vi nået frem til et slutprojekt der er kodet korrekt, og med en ekstra opgave løst.

12. Bilag:

Bilag 1.1 (Maple beregning af sandsynlighed):

Sandsynlighed
Terning test 1000 slag med en terning:

$with(Gym)$:
Vores værdier er:
1'ere: 168
2'ere: 156
3'ere: 158
4'ere: 177
5'ere: 164
6'ere: 177

chancen for at få hvert tal er $\frac{1}{6}$

jeg bruger binompdf til at finde sandsynligheden for at slå eks. 168 1'ere ud af 1000 slag med $\frac{1}{6}$ chance

1'ere:

$$binpdf\left(1000, \frac{1}{6}, 168\right)$$

2'ere

0.03351177647

$$binpdf\left(1000, \frac{1}{6}, 156\right)$$

3'ere

0.02290386842

$$binpdf\left(1000, \frac{1}{6}, 158\right)$$

4'ere

0.02627738313

$$binpdf\left(1000, \frac{1}{6}, 177\right)$$

0.02263298944

5'ere

$$binpdf\left(1000, \frac{1}{6}, 164\right)$$

0.03318879220

6'ere

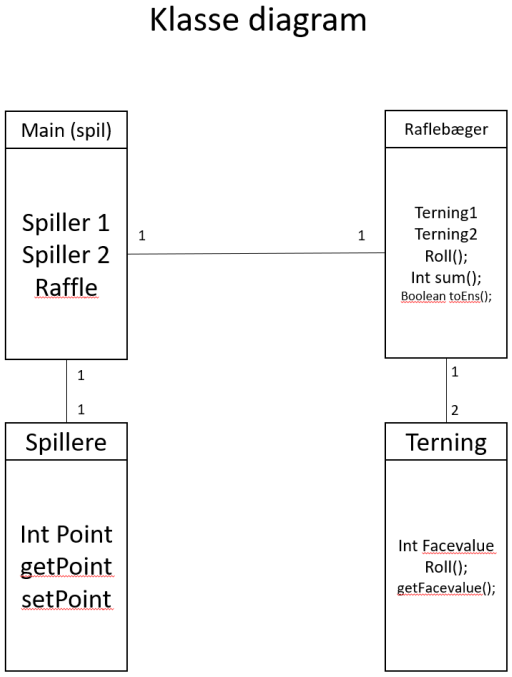
$$binpdf\left(1000, \frac{1}{6}, 177\right)$$

0.02263298944

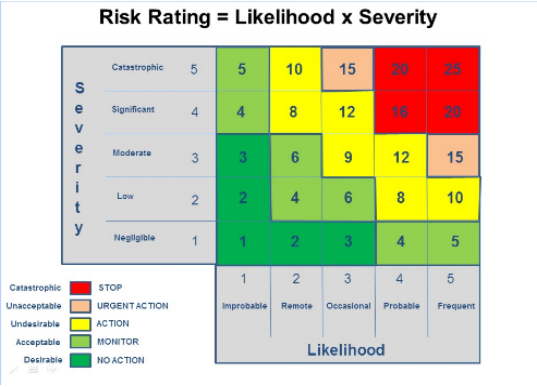
Da alle procent-værdierne er over 1%, kan vi godtage at vores terning er tilfældig.

flere bilag på næste side:

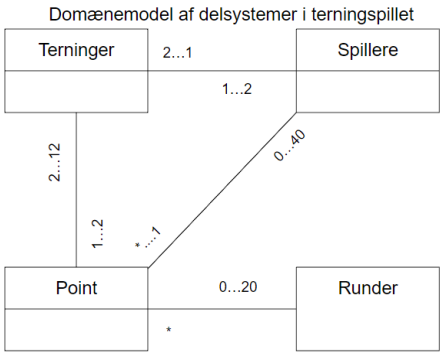
Bilag 1.2: (Klasse diagram fra navneords analyse)



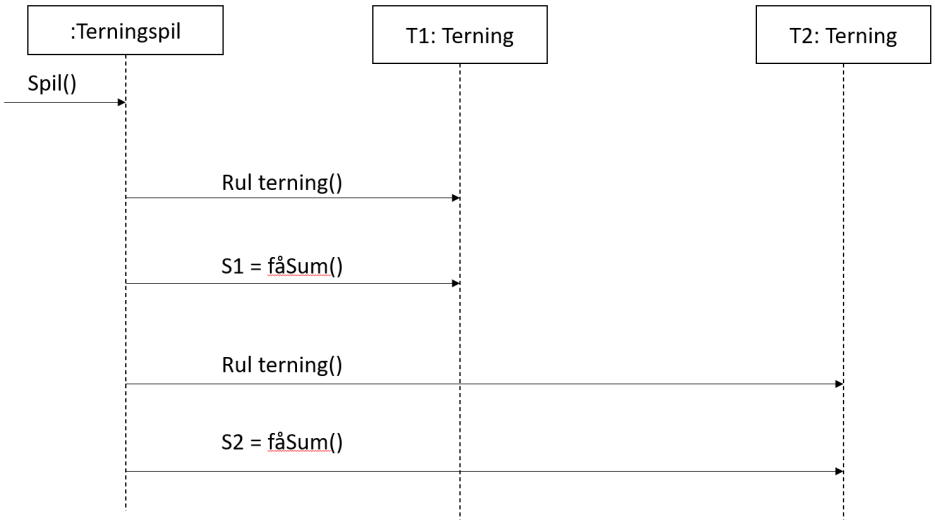
Bilag 1.3: (Likelihood x severity skema) (undervisningen: fra powerpoint slide: https://docs.google.com/presentation/d/1-DDDh2vJM59hJQDL07m32ETQ0sWL2VLKdAv-35PLGZg/edit#slide=id.gf0e2d33aef_0_518)



Bilag 1.4: (Domænemodel af delsystemer i terningspillet)



Bilag 1.5: **(**Lavet ud fra figur 1.4 kapitel 1 side 10 bog Applying UML and Patterns):



Bilag 1.6: Link til vores github repositorie:
https://github.com/ffbank/CDIO_1