02314 62531 62532

Indledende programmering, Udviklingsmetoder til IT-systemer og Versionsstyring og testmetoder

---

CDIO delopgave 1

Gruppe 17

Jakob Skov Agergaard
s224570

Philip Muff Førrisdahl
s224566

Mads Fogelberg Hansen
s224563

Esben Skovmand Elnegaard
s224555

Jarl Boyd Roest
s224556

30. september 2022

Resumé

Denne rapport indeholder et terningespil, som er blevet udviklet på baggrund af kundens vision. Vi har analyseret kravene ved at lave en Use Case beskrivelse. Under arbejdet med terningespillet har vi hele tiden testet programmet og overvejet om det opfylder kravene fra kunden. Vi har forsøgt at bygge udviklingen op gennem en iterativ udviklingsprocces. Under processen har vi testet de 2 terninger som skal indgå i terningespillet, for at se om de er normaltfordelt rigtigt. Afslutningsvis er der blevet konkluderet at at vi har opfyldt kravene fra kunden og de er blevet implementerede i Intellij.

Indhold

1 Timeregnskab **3**

2 Indledning **3**

3 Projekt-planlægning **3**

4 Krav & Analyse **3**
4.1 Use case beskrivelse 4

5 Design **5**

6 Implementering **5**

7 Test **6**

8 Reflektion **7**

9 Konklusion **7**

10 Bilag **7**
10.1 Litteratur 7
10.2 Kode 7

1 Timeregnskab

- Arbejdsdag 1 (19-09-2022): 2,5 timer
- Arbejdsdag 2 (21-09-2022): 3 timer
- Arbejdsdag 3 (23-09-2022): 2 timer
- Arbejdsdag 4 (26-09-2022): 2,5 timer
- Arbejdsdag 5 (27-09-2022): 2 timer
- Arbejdsdag 6 (28-09-2022): 2,5 time
- Arbejdsdag 7 (30-09-2022): 3 timer

2 Indledning

Denne rapport dokumenterer udviklingen af spillet Terningespil, der er udviklet på baggrund af kundens vision, kundens krav, samt vejledningen af projektlederen. Vi har forsøgt at følge (R)UP - metoden ved at være risikodrevet og fleksible i udviklingsprocessen. Vi har løbende opdateret vores opgaveprioritering og sat nye mål for de næste arbejdsdage. Vi har lavet forskellige artifakter til at hjælpe os i at forme det produkt der opfylder kundens krav bedst.

3 Projekt-planlægning

Begyndelsen: Vi strukturerede, umiddelbart efter feedback fra projektlederen, en plan for udviklingen af projektet og programmet jf. kravspecifikationerne. Vi så på opgaverne og kravene til spillet, der var var slået på plads dag 1, og prioriterede opgaverne som følgende

Uge 1: Opgaver prioriteret som udkast (19-09-2022).
Versionsstyringen af programmet samt dokumentation for alle gruppemedlemmer Use case(s), evt. diagrammer Random nr generator samt test + terningkast og test. Interface relaterede funktioner (navn, pointtæller, turskift, antal ture)

Uge 2: Opdaterede opgaver (26-09-2022)
Opgaver løst: Versionering af programmet Diagram (sekvens) test af terninger interface kode: navn, pointtæller, turskift, antal ture
Nye Opgaver Prioriteret: Færdigudiviklen koden uden redepmtionfaktor Opdatering af sekvens diagram samt klassediagram Rapportere udvikling i rapporten. Non-funktionelle krav skal implementeres (bla. coinflip)

27-09-2022: Opgaver løst: Alt kode til terningekastspillet inkl. redepmtionfaktor.
Nye opgaver prioriteret: Opdatering og commit af sekvens- samt domæne diagram. Fully dressed Use-case beskrivelse af main use case + alt flow (redemption). Implementering af GUI.

28-09-2022: Opgaver færdige: Fejl i redemption loop fikset. Introduktion, dokumentation til test-af-terninger, design-klassediagram, samt use-case-beskrivelse færdiggjort.
Nye opgaver: Sidste opdatering af sekvensdiagram og færdiggørelse af rapport

4 Krav & Analyse

For at kunne lave en mere præcis kravsspecifikation startede vi med at spørge vores projektleder om følgende spørgsmål:

Q: hvilken version af Java har maskinerne installeret?

A: Den nyeste version (18).

Q: Hvilke input-metoder er tilgængelig?

A: Både mus og tastatur.

Q: Skal de 2 spillere have samme input-metode?

A: De skal ikke, men kan godt.

Q: Skal pointene være synlige for alle spillere eller blot nogle?

A: Ja, synlige for alle.

Q: Kan spillerne ende med mere end 40 point?

A: Ja

Q: Er vinderen den der når 40 point først? Hvad hvis de rammer 40 point på samme tur?

A: Det er antallet af ture der tæller. Hvis begge spillere får 40 eller over, i samme tur, er vinderen den der slår højest i næste slag. Hvis lige fortsættes dette.

Q: Hvad er et almindeligt menneske?

A: Alle der besidder over 8. klasse pc færdighed, er op til 80 år gamle og har et almindeligt helbred.

Q: Skal spillerne have navne?

A: Ja

Q: Hvem skal starte?

A: Dette bestemmes ved et "coin flip". Altså helt tilfældigt.

Q: Hvad er formålet med testen. Hvad ønskes der testet udover systemet funktionalitet som helhed?

A: At de virtuelle terninger, statistisk opfører sig som normale terninger.

Baseret på dette har vi produceret en use case beskrivelse for use casen: "Spil terningespil"

4.1 Use case beskrivelse

Scope: Spil terningespil
Level: Spillere
Primary actor: Spiller 1 og spiller 2
Stakeholders and interests:
-Spillere: Vil have et underholdene spil der virker hurtigt og korrekt.
-Projektlederne: Vil gerne kunne tilfredsstille kundens ønsker.
-kunden: Vil gerne have et spil, der kan spilles på DTU's databarer.
Preconditions: Der er styresystemet "Windows"på maskinerne og den nyeste version af Java.
Succes Guarantee: Spillet afsluttes og angiver en vinder.
Main Succes Scenario:

1. Spillerne indtaster deres navne en af gangen
2. Systemet bestemmer vha. et coinflip hvilken af de to spillere der starter
3. Vinderen af coinflippet slår med terningerne
4. Systemet lægger øjnene på terningerne sammen, tæller dem som points og informere spilleren om dennes samlede point
5. Den anden spiller slår nu med terningerne
6. Systemet lægger øjnene på terningerne sammen, tæller dem som points og informere spilleren om dennes samlede point
   Punkterne 3-6 gentages indtil en af spillerne opnår 40 eller flere points
7. Systemet fortæller spillerne hvem der har vundet

Extensions:

7a. Begge spillere opnår 40 point i samme runde

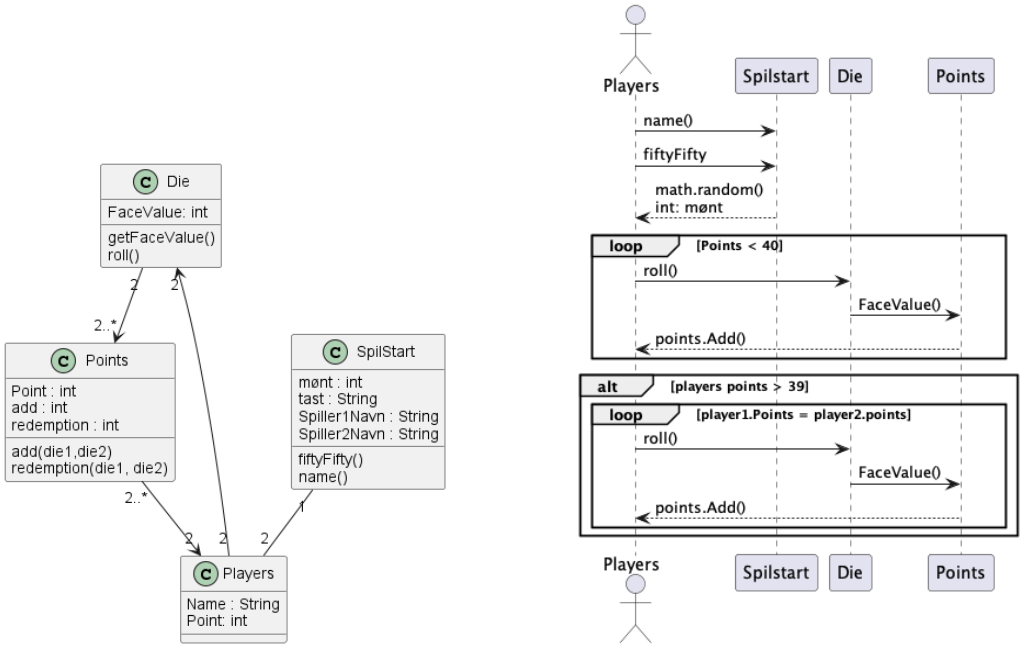
1. Systemet giver nu mulighed for at hver spiller kan kaste terningerne igen. Spilleren der slår højest med dette ene kast vinder spillet.

2. Hvis de to spilleres slag har samme værdi starter de igen fra punkt 7a.1

Special Requirements: Spillet vil kræve at spillerne kan skiftes til at benytte den samme input-metode
Frequency of Occurence: Spillet skal kunne gentages igen og igen
Open Issues:
-Skal spillerne have mulighed for at fortsætte et nyt spil med samme navne efter det første?

5 Design

For at finde frem til vores samlede løsning startede vi med at med at lave et groft sekvensdiagram. så vi kunne få et overblik over klasser, objekter og metoder.



Figur 1: Design klassediagram & Sekvensdiagram

Herefter var planen at starte med at implementere det funktionelle krav som enten bar mest risiko eller var vigtigst for spillet. Vi prioriterede derfor kravene således

1. Terninger - 2 stk. der kan kastes for at ændre deres øjenværdi
2. Points - Tæller øjenværdierne fra de to terninger sammen
3. Spillere - 2 stk. der hver har sine egne points og kan tildeles navne
4. Afslutning - erklærer en vinder baseret på points og slutter spillet
5. Redemption - Håndterer situation hvor begge spillere når 40 points eller over

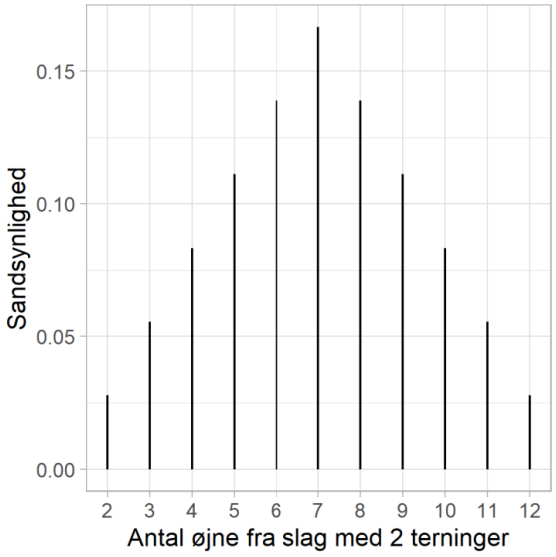
6 Implementering

Vi havde fra starten af projektet en lille forståelse for classes, objekter og metoder da vi ikke var nået til disse emner i indledende programmering. Derfor skrev vi store dele af programmet uden. Vi blev klogere senere i projektet, og implementerede derfor classes hvorfra vi kaldte metoder, mhp. at følge den object orienterede programmeringsmetode. Dette til trods for at det er et relativt lille projekt. Vi har løbende opdateret diagrammer og kode når vi opdagede en smartere løsning eller funktionalitet som vi manglede. På den måde har vi arbejdet iterativt og ikke været bange for at ændre retning i vores implementering.
Vi kan efterfølgende sige at vi ikke nødvendigvis har benyttet de konventions der er i Java ift. navngivning osv. Dette skyldes at vi har prøvet os frem og søgt på nettet for at finde ud af hvordan classes, objekter og metoder virker, da undervisningen i programmering ikke er nået så langt.

7 Test

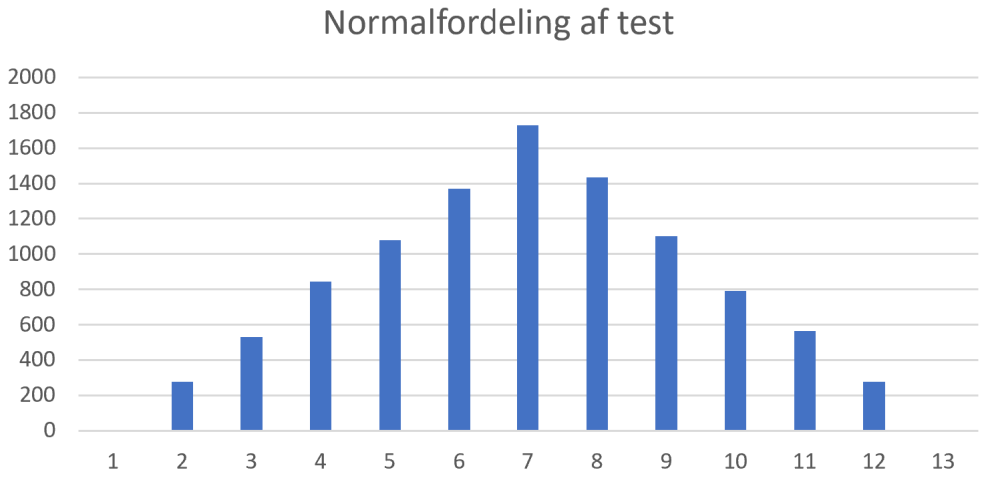
Der er blevet bedt om en test af de to terninger, som indgår i vores terningespil. Hensigten ved at teste terningerne er at se på sandsynligheden for de forskellige udfald. [1] De forskellige udfald skal stemmer overens med normalfordelingen for to terninger. En normal fordeling er karakteriserede ved at have en "klokke"formede kurve.
Der vil automatisk være kast, som er hyppigere end andre, da der er flere kombinationer som kan give det samme. Det vil for eksempel være tallene 6, 7 og 8, som vil være de mest hyppige kast. Tallene 6, 7 og 8 er de mest hyppige kast, fordi der er flest forskellige kombinationer, der fremkommer af disse tal og dermed er sandsynligheden for at kast disse kast størst.

Normal fordelingen for 2 terninger kan ses på [2]figur 2.



Figur 2: Sandsynlighed for at slå antal øjne

Vores test af terningerne er lavet i IntelliJ, som tager udgangspunkt i i de to terninger, som skal indgå i terningespillet. Terninger er blevet slået 10.000 gange for at få et mere præcis resultat. Man kan få et endnu mere præcis resultat, hvis man lavede flere terningkast, da det vil give flere observationer, og dermed en bedre normalfordeling. Derudover vil man få en pænere "klokke"formede kurve, hvis man havde et større interval af tal, da dette vil skabe en større spredning Testen af vores 2 terninger kan ses i følgende figur 3.



Figur 3: Normalfordelingen for terningekast med to terninger

Ved at sammenligne figur 2 og 3, kan man hurtigt se, at de som udgangs er meget ens. Begge

[1]https://emu.dk/sites/default/files/2020-06/normalfordelingen.pdf, side 7
[2]https://emu.dk/sites/default/files/2020-06/normalfordelingen.pdf, side 19

figurer har en det som man kan karakterisere som en "klokke"formede kurve, som også var målet. Der er nogle små ud svingninger i vores kurve, som gør at den ikke er helt præcis og dermed ikke ligner normalfordelinger af 2 terninger. Dette vil kunne løses sig ved at lave flere kast, der vil give flere observationer, som vil gøre det mere præcist.

Yderligere har det været et krav at teste antal slag hvor begge øjne er ens. Statistisk set burde dette være 1/6 eller 16,66..procent. Dette er fordi der er 6 muligheder for dobbeltøjne ud af 36 kast, ergo 1/6. Vores test giver et resultat på 16,6 procent over 10.000 slag, så vi mener det er tilfredsstillende.

8 Reflektion

Hvad har vi lært og hvad kan vi gøre bedre til næste gang?

1. Vi ville gerne lære at lave en 'rigtig' test, der rent faktisk kører programmet En test der kan teste hele spillet, så man ikke manuelt skal gennemgå programmet, hver gang man skal teste en ny linje kode
2. Undgå at bruge for lang tid på at lave diagrammer og modeller.
3. Sørg for at comitte oftere løbende
4. Mere struktur i arbejdsfordelingen og fremgangsmåden for hver arbejdsdag
5. Kommunikation i gruppen
6. Opgavefordeling, så alle får lavet på lidt af hvert

9 Konklusion

Vi har lavet et spil, der opfylder kundens krav. Dog er vi bevidste om at vi ikke har fundet den mest elegante løsning, da meget af vores kode kunne optimeres og da meget af koden er genbrugt igennem programmet. Samtidig kunne vi have valgt at implementere koden i en GUI, men da det ikke var et krav og tiden var knap, undlod vi dette.

10 Bilag

10.1 Litteratur

- https://emu.dk/sites/default/files/2020-06/normalfordelingen.pdf, Bo Markussen, Københavns Universitet 11/03/2020

10.2 Kode

Vores IntelliJ-projekt er pakket i .zip-filen der er afleveret. Derudover er der adgang til vores fjern-repo gennem følgende link: https://github.com/Jakob-SA/CDIO-1