Technical University of Denmark

DTU

CDIO 1

1. Oktober 2020

Gruppe 20

Lucas Schoubye
S215801

Ismail Ali
S190201

Mark Nielsen
S204434

Mads Sørensen
S215805

Lucas Scoubye
Sxxxxxx

Indholdsfortegnelse

Resume

Timeregnskab

Indledning

Denne rapport er dokumentation af vores arbejder, herunder analyse, design, implementering og test, af en opgave stillet af en kunde gennem IOOuterActive. Opgaven er et terningespil, hvor 2 spillere skiftevis slår med 2 terninger. En spiller vinder ved at nå over 40 point og slå et par. Kunden har afgivet deres vision og kravspecifikation af spillet, som vi har søgt at forstå og afklare til fulde, så vi kan aflevere det bedste mulige produkt til kunden. Vores produkt er udviklet i Java, og vi har anvendt en Git-repository til produktets ændringshistorik.

Projekt-planlægning

Krav

Functionality

- Spillet foregår mellem 2 personer
- Spilleren skal slå med 2 terninger
- Spillet skal gå på tur
- Spillet skal vise resultatet af terningekastet
- Spillet vindes efter 40 point
- Spillet skal regne point korrekt
- Spillet skal fungere på en databar computer
- Spillet skal have private attributter
- Spillet bør have et GUI element
- Spilleren bør vinde ved at slå 2 ens efter 40 point
- Spilleren bør miste alle sine point hvis der slås to 1'ere
- Spilleren bør få en ekstra tur hvis der slås to ens
- Spilleren bør vinde spillet hvis der to ture i træk slås to 6'ere

Usability

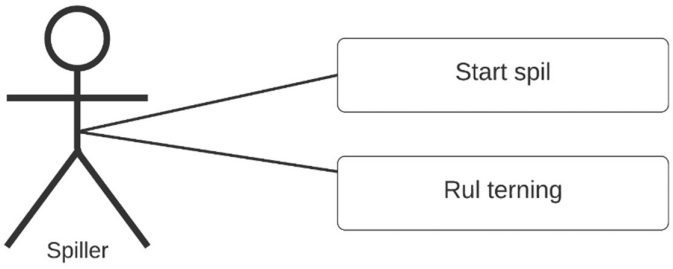
- Der bør være en knap til at slå med terningerne
- Spillet skal være dokumenteret
- Spillet skal vise hvis tur det er
- Spillet skal kunne anvendes uden en brugsanvisning

Reliability

- Raflebægeret skal fungere korrekt over 1000 kast

Analyse

Use Case Diagram



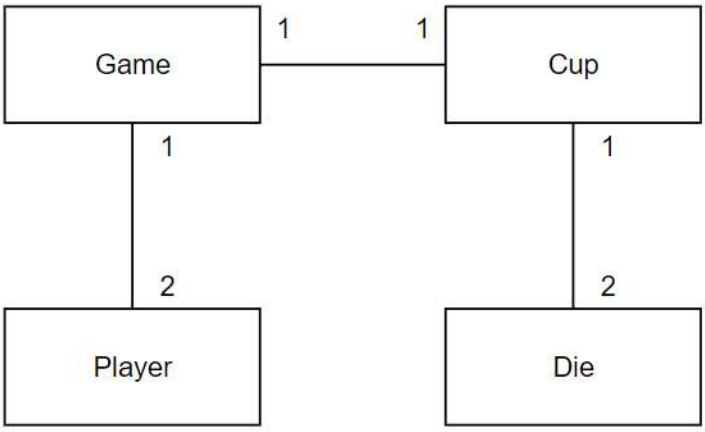
Figur 1. Use Case Diagram

Use case beskrivelse

- Start spil
  - Spillerne starter spillet.
- Rul terning
  - Spillerne ruller terningen og får værdierne vist, hvorefter point adderes til pågældende spillers samlet point.

Domænemodel

Herunder kan vi se de klasser som ville opstå i spillet. Domænemodellen er lavet ud fra kundens vision.



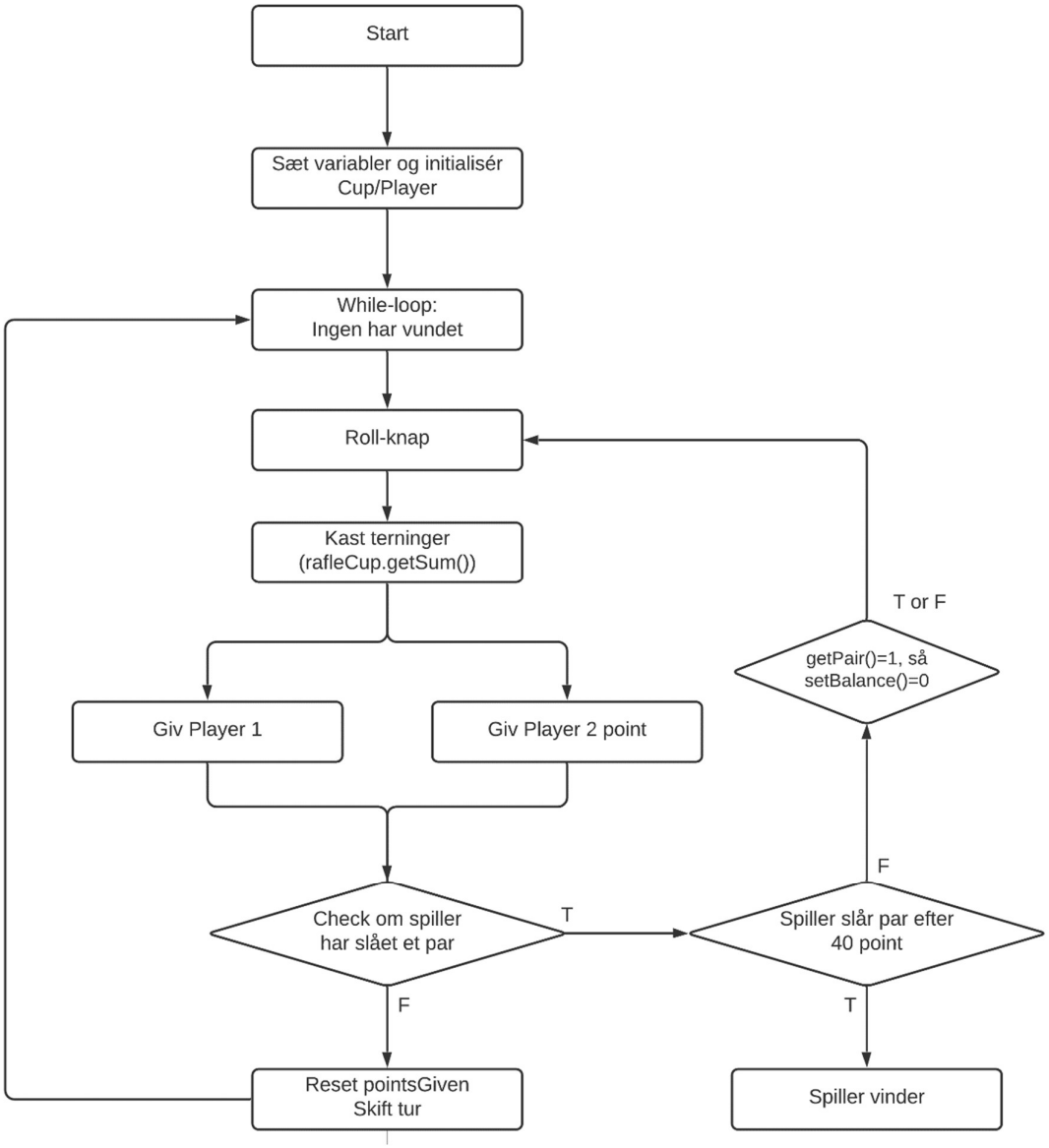
Figur 2, Domænemodel

Flowdiagram

Flowdiagrammer kan bruges til at visuelt repræsentere hvordan et program skal forløbe. Herunder har vi lavet 2 flowcharts som viser spillet med minimum mængde af regler og med tilføjet ekstra regler.

I venstre side af figur 2 kan et tidligt udkast til programmet ses. Her vil spillet automatisk skifte tur uden input fra spillerne. Der er ingen ekstra regler.

I højre side af figur 2 kan programmet ses i den nuværende version. Her er der blevet implementeret.



Figur 1, Flowdiagram

Design

Game

Game Klassen bruges til spil logikken. Her er vores ”public static void main” som kører vores loop. Her bliver alle objekter initialiseret. Den kalder de andre klasser og deres metoder så spillet kan opnå det flow som kan ses på figur 2. Hvis en spiller når over 40 point med et par vinder de spillet.

Die

Die Klassen er en simpel klasse. Den bliver brugt til at producere et tilfældigt tal fra 0-6. Den information skal kunne gemmes og kaldes fra andre metoder. For eksempel da der skal også tjekkes om de 2 terninger er et par. De tilfældige tal bliver produceret ved hjælp af ”Java.util.Random”, der bliver givet en konstant kaldt ”MAXVALUE”. Værdien af terningen bliver gemt i ”facevalue”. Klassen har en enkelt metode ”getFacevalue” som både slår med terningen og returnere værdien.

Cup

Klassen Cup sørger for at initialisere de to terningobjekter (Die1, Die2), hvorefter vi gør brug af metoden getSum() som returnerer den sammenlagte værdi af de to terninger. Derudover indeholder klassen også metoderne getFacevalue1() og getFacevalue2() som returnerer den enkelte ternings værdi. Metoden getPair() benyttes til at overholde reglerne ”hvis man slår to ens må man slå igen” og ”hvis man slår to ens over 40 vinder man spillet”, hvilket fremgår og er implementeret i if-sætningerne i klassen Game.

Player

Klassen Player er en simpel klasse. Den bliver brugt til at holde styr på og kontrollere hver spillers point. Vi kommer til at skulle have metoder der kan forøge scoren, returnere scoren og sætte scoren lig med nul hvis der bliver rullet et par 0.

Implementering

Game

Spillet blev implementeret med 3 variabler. Sum er sat til at starte spillet, hvor det senere definerer rafleCup til at rulle terningerne. Så bruges pointsgiven som har værdien 0 til når der skiftes tur til brug af loop, som giver spilleren et point ad gangen, og playerTurn er True for at kunne køre if statement til at skifte turen mellem spillerne.

Die

Terningen blev implementeret med 2 variabler og en 1 metode. MAXVALUE konstanten bestemmer mængden af sider på terningen, facevalue er hvilken værdi terningen har på det givne tidspunkt. Metoden getFacevalue() sætter facevalue til et en værdi mellem 1-6 og returnere værdien.

Cup

Cup blev implementeret med 3 variabler. QUANTITY som fortæller os hvor mange terninger der er i Cup, sum der fortæller hvor meget de to terninger giver lagt sammen og pair der fortæller om terningerne har slået et par. Her bruges igen den ovennævnte facevalue1 og 2, til at finde de ovennævnte variabler sum og pair på følgende måde: getSum() addere de to facevalues, getPair() kigger på de to facevalues og ser om de er ens eller ikke ens, altså par eller ikke par.

Player

Player klassen blev implementeret med 3 metoder og 1 variabel. increaseScore tilføjer et point til scoren, dette bliver brugt i et loop af game klassen. Vi har også en getScore og en metode setScoreZero som bliver brugt hvis spilleren slår et par 1.

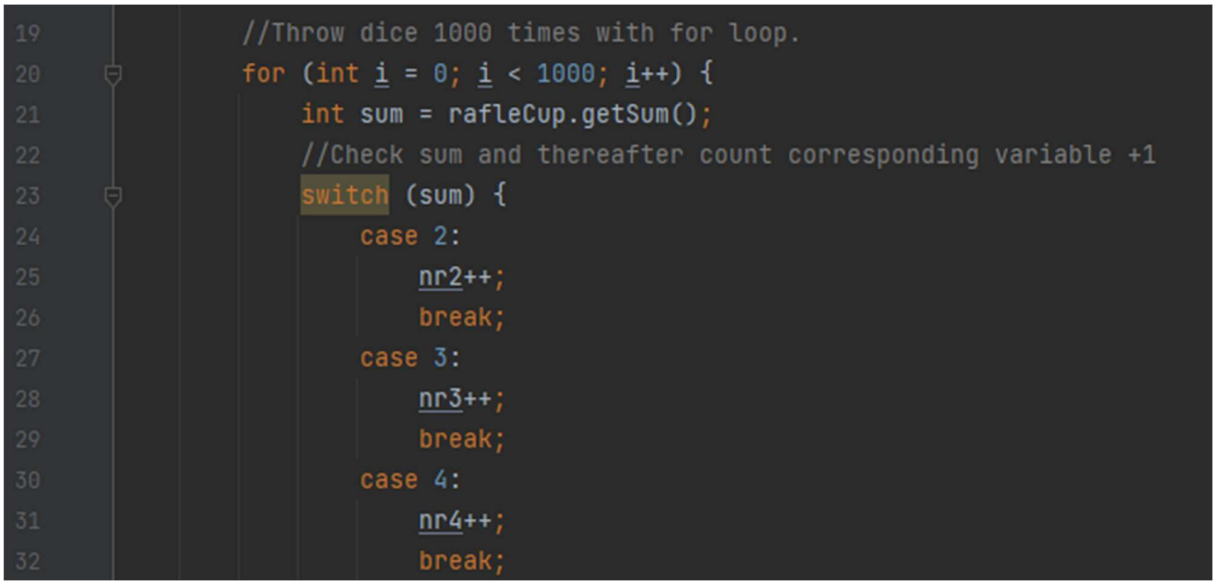
GUI

Den valgte GUI er implementeret efter import fra udgivet guide. Vi har her benyttet også af de forskellige metoder og interfaces og integreret det med vores terningespil.

Test

Raflebægeret er testet ved at kaste 2 terninger 1000 gange, og optalt alle værdier af summen af 2 terninger fra 2-12. Vi har oprettet en Test class til dette formål.

Raflebægeret er blevet kastet ved brug af et for loop, der looper 1000 gange, og har derefter kaldt funktionen getSum(), hvis værdi bliver tjekket i et switch-statement. Værdien af summen inkrementerer den tilsvarende variabel med 1.



```
19          //Throw dice 1000 times with for loop.
20          for (int i = 0; i < 1000; i++) {
21              int sum = rafleCup.getSum();
22              //Check sum and thereafter count corresponding variable +1
23              switch (sum) {
24                  case 2:
25                      nr2++;
26                      break;
27                  case 3:
28                      nr3++;
29                      break;
30                  case 4:
31                      nr4++;
32                      break;
```

Variablerne printes til konsollen sammen med den procentvise andel af 1000 kast, og der fås følgende værdier:

2: 23.0 (2.3%)

3: 60.0 (6.0%)

4: 90.0 (9.0%)

5: 114.0 (11.4%)

6: 122.0 (12.2%)

7: 160.0 (16.0%)

8: 141.0 (14.1%)

9: 121.0 (12.1%)

10: 79.0 (7.9%)

11: 54.0 (5.4%)

12: 36.0 (3.6%)

Vi valgte at bruge floats til vores variabler, da vi undgår afrunding i procentregningen, der sikrer større præcision.

Denne testmetode virker ikke med GUI-elementer. Dette gik først op for os efter vi havde implementeret GUI'en, derfor oprettede vi en noGUI-branch, hvor vi gik et par commits baglæns i vores program, og dermed kunne teste terningen uden GUI.

De statistiske sandsynligheder for at slå en given sum af 2 terninger er:

Total	Number of combinations	Probability
2	1	2.78%
3	2	5.56%
4	3	8.33%
5	4	11.11%
6	5	13.89%
7	6	16.67%
8	5	13.89%
9	4	11.11%
10	3	8.33%
11	2	5.56%
12	1	2.78%
Total	36	100%

[1]

Generelt kan vi konkludere, at vores terning er indenfor ± 1 procentpoint af de teoretiske sandsynligheder.

Konklusion

[1] Kilde: Susan Holmes (2000) https://statweb.stanford.edu/~susan/courses/s60/split/node65.html

Bilag

Generelle spørgsmål til specifikationer af krav:

1. Skal spillet have et GUI element? Det vil sige om spillet udelukkende skal spillet ved at få tekst i kommandofeltet eller om der skal være et visuelt element som traditionelle computerspil.
2. "Ser resultatet med det samme", hvilken målbart tidsinterval snakker vi om? Hvor stort et tidsinterval er der tale om?
3. "Opnår 40 point", Vil en spiller vinde hvis de opnår over 40 point med et par? Hvad sker der hvis en spiller kommer over 40 point? Hvad skal forstås med at vinde? Hvordan skal det indikeres?
4. "Slå 2 ens for at vinde" hvad hvis spilleren slår par 1? vinder spilleren eller mister de deres point?
5. "Slå med et raflebæger", skal der vises et raflebæger visuelt på skærmen? Skal der være en knap eller skal der klikkes med musen på et billede?
6. Hvor mange sider skal terningen have?
7. "Få en ekstra tur", menes der at spilleren får et ekstra kast?

Learnings til næste gang

1. Starte tidligere på analyse & design.
2. Være bedre til at lave atomic commits og mere sigende kommentarer til commits.
3. Være skarpere på definitioner af hårde krav til f.eks. test (hvornår "virker" raflebægeret korrekt?)
4. Få afklaret og identificeret upræcise krav og definitioner tidligere med "kunde" og "projektledere".