Danmarks Tekniske Universitet

02312 02313 02315

Indledende Programmering
Udviklingsmetoder til IT-systemer
Versionsstyring og Testmetoder

Gruppe 20

4. oktober 2019

CDIO 1


Mads Storgaard-Nielsen
s180076

Nikolaj Morgen
s170182


Martin Mårtensson
s195469

Frederik Verne Henriksen
s173394


Mohamad Abdulfatah Ashmar
s176492

Daniel Styrbæk-Petersen
s143861

Intro

Resumé

- Denne opgave beskriver konstruktionen af et terningespil til en fiktiv kunde. Opgaven er konstrueret efter CDIO konceptet, hvilket har gjort det muligt at lave projektet ud fra blot en række krav stillet af denne kunde. Opgaven introduceres ved en specificering af krav hvori der beskrives at spillet blandt andet skal være for to spillere, og at hver spiller skal skiftes til at slå to terninger. Derefter beskrives nogle use cases der opfylder disse krav, altså use casesne AntalSpillere og KastTerning. Derefter opstilles diverse diagrammer der viser hvordan koden struktureres. Kildekoden til projektet er skrevet i Java 8, og der er brugt flere klasser for at gøre det lettere at læse og eventuelt ændre i programmet. Efterfølgende udføres der tests, og der konkluderes i disse, at afvigelsen på resultaterne er lavere ved flere tests. Afslutningsvist konluderes der, at det lykkedes at lave et spil der opfyldte samtlige krav fra kunden, men at der kunne have været implementeret flere af de optionelle krav for et mere avanceret terningespil.

Indledning

- Opgaven handler om at opbygge et system der skal fungere som et terninge spil. Reglerne og systemet skal være simpelt og yderst brugervenligt. Når en værdi på 40 point er opnået, skal spillet som udgangspunkt afsluttes. Hvis muligt, skal der tilføjes lidt ekstra til regelsættet. Det fremgår i opgaveformuleringen, at spillet ønskes nemt og uden brugermanual.

Timeregnskab

Områder/Navne	Mads	Frederik	Mohamad	Nikolaj	Martin	Daniel	I alt
Projektplanlægning	1	1	1	1	1	1	6
Rapportskrivning	3	2	3	4	2	5	19
Kravspecifikation	0	1	1	1	1	0	4
Use cases	1	0	0.5	3	4	0	8.5
Programmering	8	2	0	2	0	2	14
Versionsstyring	1	1	1	1	1	1	6
Testing	1.5	2	0	1.5	0	1	6
Gennemlæsning	2	2	2	2	2	2	12
Samlet tid i timer	**17.5**	**11**	**8.5**	**15.5**	**11**	**12**	**75.5**

Tabel : Tabel over timeregnskab

Indholdsfortegnelse

Intro **1**

Resumé . . . 1

Indledning . . . 1

Timeregnskab . . . 1

Krav **3**

Vision . . . 3

Navneordsanalyse . . . 4

Kravspecifikation . . . 4

Funktionelle krav, FURPS+ . . . 5

Non-funktionelle krav, FURPS+ . . . 5

Domænemodel . . . 6

Analyse **7**

Usecase diagram . . . 7

Usecase beskrivelser . . . 8

Sekvens diagram . . . 9

Design klasse diagram . . . 10

Implementering **11**

Kode eksempel . . . 11

Gennemgang af koden . . . 12

Testing **13**

Test cases . . . 13

Reflektion **15**

Konklusion . . . 15

Projektforløb . . . 15

Bilag **16**

Bilag A . . . 16

Bilag B . . . 16

Krav

Vision

- Et af kravene fra kunden er, at spillet skal være simpelt. Det har vi tolket som i, at det skal kunne spilles med en knap, efter indtastningen af spillerenes navne. Vi tænker, at udforme den printede tekst. Det skal ikke herske tvivl om hverken score eller hvis tur det er.

- Her kommer en beskrivelse af umiddelbare første tanker for vist tekst på linjeform, med tilhørende beskrivelse:

Indtast navnet på spiller 1:
Indtast navnet på spiller 2:
(Hvorefter navnene fremgår før hvert kast)

I skal slå om hvem der starter, tast enter for at slå

Spiller 1 slog 6 og spiller 2 slog 4
Spiller 1 starter
(Der er nu styr på hvilken spiller der starter)

Spiller 1 starter med terningerne
(Dette skaber tydelighed om hvem der starter)
Spiller 1 slog : 1 og 5 hvilket giver 6 points!
Spiller 1 har nu: 6 points!
(Der er nu hele tiden styr på, hvem der slår, og overblik på score.)

Og så fremdeles med points frem til 40

Spiller 1 vinder med 42 points mod spiller 2's 38 points
(Spillet er nu afsluttet, grundet spiller 1's værdi over 40.)

Navneordsanalyse

- Nedenstående er en liste over navneord til vores navnenordsanalyse baseret på vores udvalgte "Ekstratur"Use Case.
  - Spiller, Dobbelt slag, Sum af to, Point, Samme værdi.
  - Vi har overvejet behovet for yderligere tilføjelser af navneord, men vi mener de eksisterende navneord er fyldestgørende.

Kravspecifikation

- Krav (R)
  - R1 2 Spillere.
  - R2 Terninger kastes i hver tur.
  - R3 Point beregnes i hver tur.
  - R4 Spil afsluttes når en spiller rammer 40 point.
  - R5 Nulstil Point ved to 1'ere.
  - R6 Ekstra tur, ved dobbelt slag.
  - R7 Ved dobbelt 6, to gange i træk, vindes spillet(*)

- Use Cases (U)
  - U1 AntalSpillere
  - U2 KastTerning
  - U3 VælgVinder
  - U4 NulstilPoint
  - U5 GivEkstratur
  - U6 AfslutSpilMed12(*)
  - U7 AfslutSpilMedTo6(**)

U6(*) U7(**) er ikke blevet implementeret grundet resourcemangel

Krav (R) Use Case (U)	U1	U2	U3	U4	U5	U6	U7
R1	■		■				
R2		■		■			
R3	■						
R4			■				
R5						■	
R6					■		
R7							■

Tabel : Krav -og Usecase tabel: Vores sikkerhed for, at use cases dækker alle ønskede krav

Funktionelle krav, FURPS+

- Functionality
  - Spiller1 og spiller2 slår med terninger om hvem der starter.
  - Når der tastes enter, kastes terningerne.
  - Et spil for 2 personer
  - Hver spiller kaster 2 terninger.
  - Summen af terningernes øjne bliver lagt til spillerens point ved hvert kast.
  - Vinderen kåres når den første spiller rammer 40 point.
  - Spillerne får en ekstra tur hvis de slår et dobbeltslag.
  - Hvis en spiller slår to 1'ere, sættes spillerens point til 0.

- Useability
  - Det skal kunne spilles af alle, uden en introduktion.

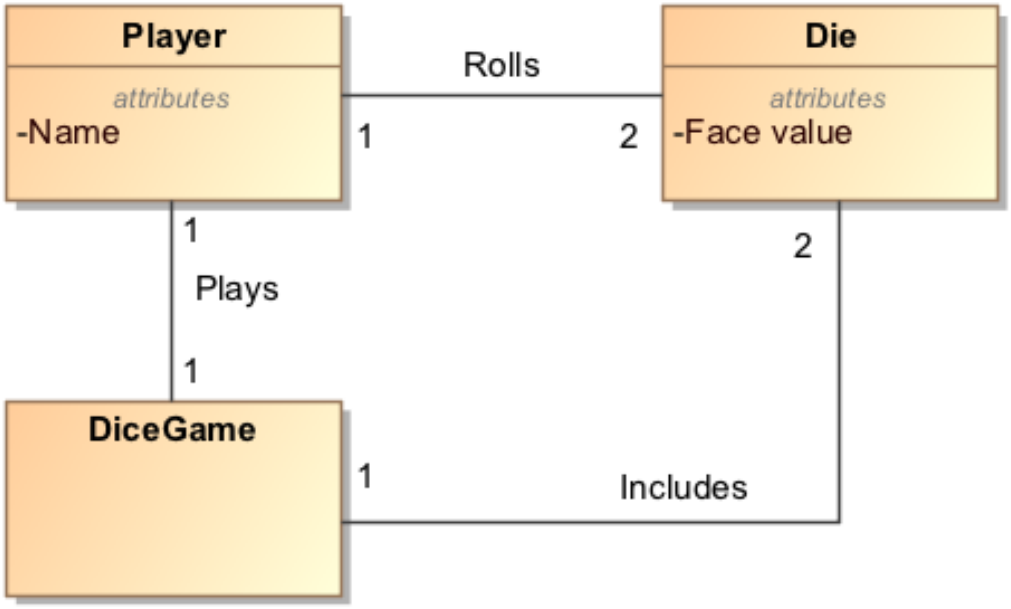
Non-funktionelle krav, FURPS+

- Reliability
  - Spillet skal kunne fejlfrit imens spillet er igang.
  - Terningerne skal virke korrekt.

- Performance
  - Spillerne skal se terningekastet maks 333 ms. efter de er kastet

- Supportability
  - Koden skal være kommenteret, og med fortællende variablen og metodenavne.
  - Koden skal være bygget op fornuftigt.

Domænemodel



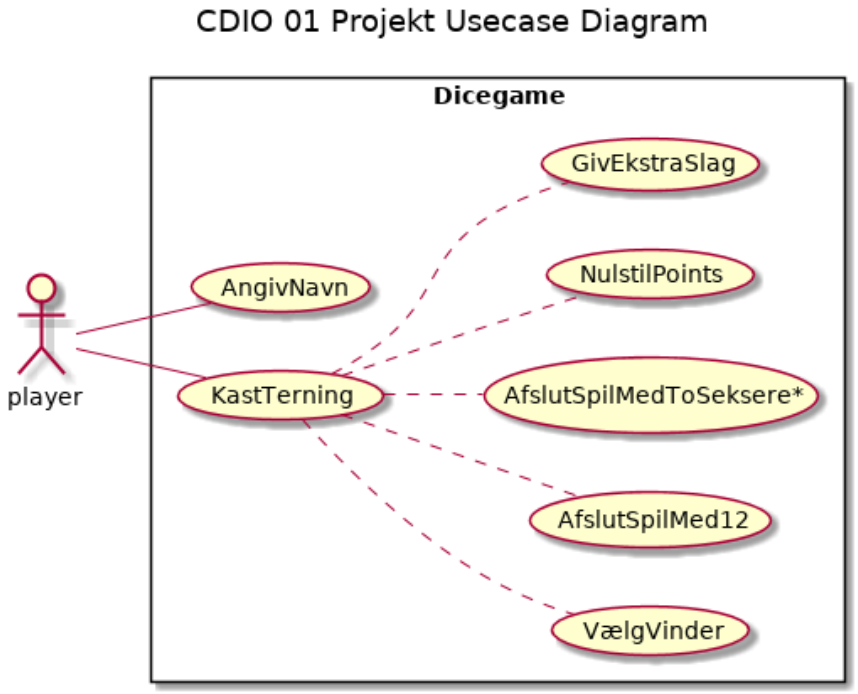
Figur : Domænemodel lavet i Magic Draw

- Hvis vi læser (DiceGame) fra venstre:
  Terningespillet har to spillere, og spillerne ruller terningerne, og terningerne bliver inkluderert i terningespillet.

- Hvis vi læser (DiceGame) fra højre:
  Terningspillet har to terninger, terningerne blev rulles af spilleren, spilleren spiller terningspil.

Analyse

Usecase diagram



Figur : Usecasediagram lavet i PlantUML

- I diagrammet ovenfor vises vores use cases, vores aktører samt hvad der sker efter øjnenes værdi er blevet evalueret.

Usecase beskrivelser

- **AngivNavne**
  Successkriterie: 2 spillere registreres i starten af spiller, og undervejs i spiller hersker der ingen tvivl om, hvilken spillsers tur det er.
  Spillet designes til 2 spillere, som intro til spillet registreres spillernes navne, Det er derfor ikke muligt at spille kun 1 spiller. Dog kan man spille 1 person, da man bare kan oprette 2 fiktive navne og derved spille mod sig selv. Der spilles på en computer og der anvendes kun "return knappen".

- **KastTerning**
  Successkriterie: Terninger kastes tilfældigt, uden systematik for værdiernes udfald. Spillet er tænkt sådan, at man har 2 terninger. Terningerne er traditionelle 6 kantede terninger, med værdierne 1-6. Man kan derfor med to terninger opnå værdier fra 2 - 12. Terningerne kastes tilfældigt, så der er ingen garanti for udfaldet af kast.

- **VælgVinder**
  Succeskriterie: Mellem hvert kast opgøres der status på point, så spillerne undervejs kan holde øje med scorer.
  VælgVinder fungerer sådan, at summen af terningernes værdier lagres. Første spiller der opnår 40 point, har vundet. Undervejs opgøres point mellem hver kast, så der konstant er vished om score.
  Systemet kan ydermere vise direkte sejr, nederlag eller miste point, hvis gældende regler, markeret med (*) træder i kraft undervejs.

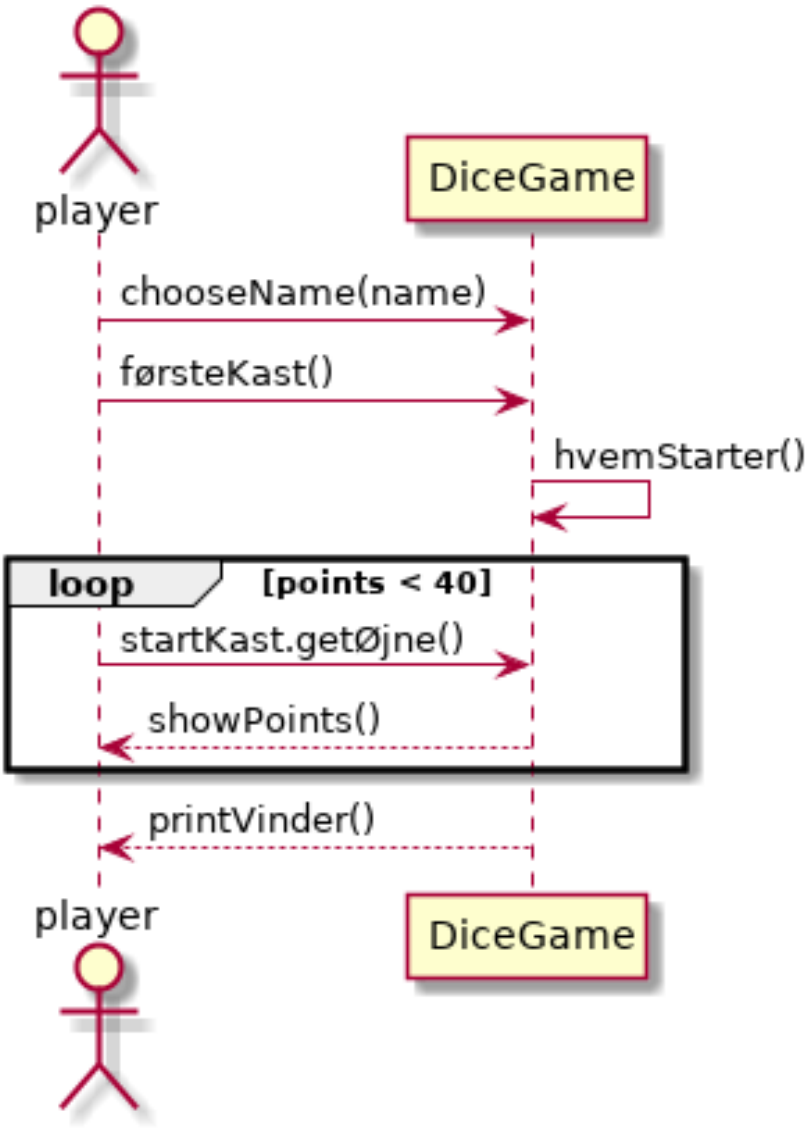
- **NulstilPoint** Succeskriterie: Spillerens point nulstilles, ligegyldig antal, hvis der bliver slået dobbelt 1. Hvis reglen indføres i spillet. (Dette afhænger af projektets tidsplan)
  PointNulstilles ideén fungerer således, at man får sat sin point til 0, hvis man slår 2 1'ere i et kast. Dette har til formål, at gøre spillet ekstra uforudsigeligt.

- **GivEkstraTur** Succeskriterie: Spiller får lov at slå igen, hvis man slår et dobbeltslag. Dette gælder også selvom du rammer en sum af 2, og derved mister alle point.
  Hvis en spiller slår samme værdi på begge terninger, så får spilleren lov at kaste igen. Dette har til formål, at gøre spillet ekstra uforudsigeligt.

- **AfslutSpilMed12(*)** Succeskriterie: Spillet vindes hvis spiller slår 12 point to gange i træk, ligegyldig tidspunkt i spil.
  Spillet afsluttes øjeblikkeligt med sejr, hvis en spiller slår værdien 12 to gange i træk. Ligegyldig om det er spillets to første kast. Dette er fordi, at sandsynligheden for dobbelt 12 er lille og dermed sjælden.

- **AfslutSpilMedToSeksere(**)** Succeskriterie: Når man har nået ønsket limit, 40 point, så vindes spil når spilleren har slået værdien 12.
  Efter at spiller har opnået ønskede 40 point, slår spilleren til der opnås værdien 12. Hvis spilleren ikke slår værdien 12 er det den andens spillers tur, medmindre det er dobbelt værdi af en anden. Dette har til formål at trække spillet lidt ud, da det ellers kan afsluttes "forholdsvis" hurtigt.

Sekvens diagram

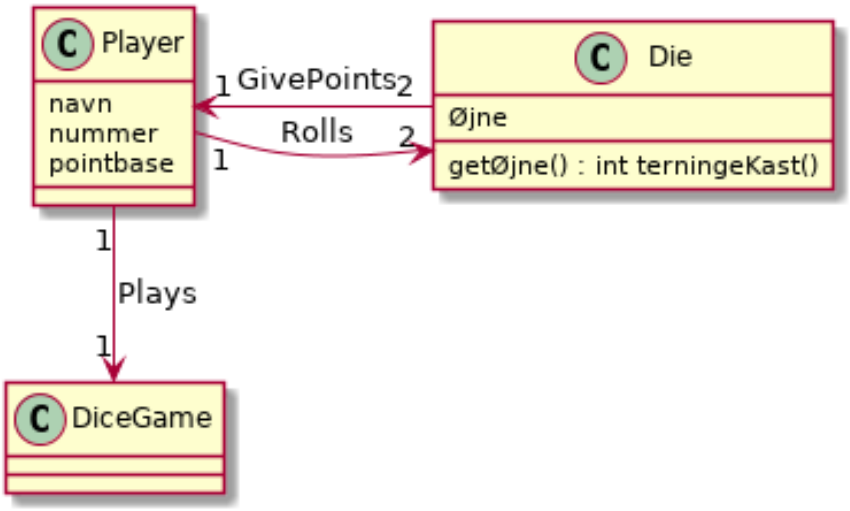


Figur : Sekvensdiagram lavet i PlantUML

- I diagrammet ovenfor vises et sekvens diagram over terningespillets forløb.

- Player vælger navn og kaster første kast, derefter evaluerer DiceGame hvilken player som skal starte.
  Så begynder loopet hvor Player slår med terningen og ser hvor mange points han har opnået ind til at han når 40 eller flere points, hvorefter han ryger ud af loopet og DiceGame viser hvem som har vundet spillet.

Design klasse diagram



Figur : Design class diagram lavet i PlantUML

- I design class diagrammet ses 3 klasser som hver har nogen atributter og funtioner. Man kan se at Spilleren ruller terningen og terningen giver Spilleren points, imens at spilleren spiller Spillet.

Implementering

Kode eksempel

- Vi valgte at spillerne skulle slå om at starte, vi lavede derfor en klasse der håndterede dette, hver spiller får tildelt et kast og metoden evaluere herefter slagene og sætter "tur"variablen til en værdi, der derefter bliver returneret, den værdi afgør så hvem der får lov at starte når do while løkken evaluere "tur"s værdi.

HvemStarter klassen:

```
package com.company;
class HvemStarter {

    //opretter to terningekast vha. TerningeKast klassen
    TerningeKast spiller1slag = new TerningeKast();
    TerningeKast spiller2slag = new TerningeKast();

    //Løkke der finder ud af hvilken en af spillerne der får lov at kaste først
    public int førsteKast() {
        int tur = 2, startslag1,startslag2;;

        //Terningerne kastes og gemmes i startslag1 og startslag2
        do {
            startslag1 = spiller1slag.getØjne();
            startslag2 = spiller2slag.getØjne();

            //Slagene udskrives
            System.out.println("Spiller 1 slog: " + startslag1 + "\nSpiller 2 slog: " +
            ↪  startslag2);

            //Hvis slagene er ens udskrives der Omslag, og der bliver slået igen, indtil slagene
            ↪  ikke er ens
            if (startslag1 == startslag2) {
                System.out.println("Omslag!");
            }
        } while (startslag1 == startslag2);

        //Det højeste slag bliver fundet, og enten spiller 1 eller spiller 2 starter, alt efter om
        ↪  tur går en op eller ej.
        if (startslag1 > startslag2) {
            System.out.println("\nSpiller 1 starter!");
        } else {
            tur++;
            System.out.println("\nSpiller 2 starter!");
        }
        //Turs nye værdi bliver returneret så den rigtige spiller får lov at starte.
        return tur;
    }
}
```

Gennemgang af koden

- Det følgende er en kort gennemgang af programmets hoved funktioner:
  - Når spiller startes bliver spiller1 og spiller2 bedt om at indtaste deres navne (Linje 24-27)
  - Spillerne slår om hvem der får lov at begynde spillet, spillet håndterer dette i HvemStarter klassen, variablen "tur" får tildelt en værdi på 2 eller 3 vha. førsteKast metoden. Tur har værdien 0 til at starte med, men efter førsteKast metoden har evalueret de to slag (linje 39) bliver tur sat til en værdi på linje 42 og do-while løkken evaluere hvem der starter spillet linje 47(spiller1) og linje 90(spiller2) alt efter om værdien af tur er delelig med 2 eller ej.
  - I hver tur får spilleren tildelt to terningekast vha. getØjne metoden i TerningeKast klassen (linje 50+51(spiller1)) og (linje 93+94(spiller2), summen af de to terninger bliver lagt sammen på hhv. linje 64(spiller1) + 107(spiller2), og derefter lagt til spillerens point på hhv. linje 65(spiller1) + 108(spiller2)
  - Når spillet er igang skiftes spiller1 og spiller2 til at kaste med terningerne, efter hver spillers tur bliver der lagt én til tur tælleren linje 80(spiller1)+123(spiller2) dog ikke hvis der er blevet slået et dobbeltslag, hvor man får en ekstra tur, dette bliver evaluaere af et if-statements på linje 77-80(spiller1) og linje 120-123(spiller2).
  - Spillet afsluttes når en spiller rammer 40 point, det sker ved at do-while løkken bliver brudt vha. If-statements i linje 83-85(spiller1) og 126-128(spiller2)
  - Efter løkken bliver brudt, evaluere et if-statement på linje 133-136 hvem der vandt spillet og der bliver udskrevet en besked med hvem vinderen er, vinderens point, samt taberen og taberens point (linje 133-136).

Testing

Test cases

- Vi designer et program der kaster 1000 kast i træk, gemmer data for hvert kast og derefter analyserer det. Derefter kan vi se om der er systematik i kastene. Forudsætningen for en acceptabel test vil være, at tallene ikke har nogen form for systematik, altså at kastene er fuldstændig tilfældige. Vi vil vurdere data sådan, at vores 1000 kast med to terninger vil ca give 333 af hver værdi på hhv. 1,2,3,4,5 og 6. Vi vil finde ud af, hvor stor afvigelse der er og vurdere herefter.

```
Hver terningen bliver kastet det valgte antal kast
- Hvor mange kast vil du teste over?: 1000
Antal kast med to terninger: 1000

Terning 1:
1'ere: 166 - hvilket svarer til ca. : 16.62% af kastene

2'ere: 150 - hvilket svarer til ca. : 15.02% af kastene

3'ere: 158 - hvilket svarer til ca. : 15.82% af kastene

4'ere: 171 - hvilket svarer til ca. : 17.12% af kastene

5'ere: 174 - hvilket svarer til ca. : 17.42% af kastene

6'ere: 181 - hvilket svarer til ca. : 18.12% af kastene

Procent i alt: 100.10


Terning 2:
1'ere: 172 - hvilket svarer til ca. : 17.22% af kastene

2'ere: 167 - hvilket svarer til ca. : 16.72% af kastene

3'ere: 164 - hvilket svarer til ca. : 16.42% af kastene

4'ere: 172 - hvilket svarer til ca. : 17.22% af kastene

5'ere: 147 - hvilket svarer til ca. : 14.71% af kastene

6'ere: 178 - hvilket svarer til ca. : 17.82% af kastene

Procent i alt: 100.10


Antallet af dobbeltslag: 169 hvilket svarer til: 16.90% af kastene.
```

Figur : Her ses programmet testet over 1000 simulerede kast

```
Hver terningen bliver kastet det valgte antal kast
- Hvor mange kast vil du teste over?: 1000000
Antal kast med to terninger: 1000000

Terning 1:
1'ere: 167183 - hvilket svarer til ca. : 16.72% af kastene

2'ere: 167555 - hvilket svarer til ca. : 16.76% af kastene

3'ere: 166477 - hvilket svarer til ca. : 16.65% af kastene

4'ere: 166013 - hvilket svarer til ca. : 16.60% af kastene

5'ere: 166419 - hvilket svarer til ca. : 16.64% af kastene

6'ere: 166353 - hvilket svarer til ca. : 16.64% af kastene

Procent i alt: 100.00


Terning 2:
1'ere: 167009 - hvilket svarer til ca. : 16.70% af kastene

2'ere: 166497 - hvilket svarer til ca. : 16.65% af kastene

3'ere: 166865 - hvilket svarer til ca. : 16.69% af kastene

4'ere: 166473 - hvilket svarer til ca. : 16.65% af kastene

5'ere: 166545 - hvilket svarer til ca. : 16.65% af kastene

6'ere: 166611 - hvilket svarer til ca. : 16.66% af kastene

Procent i alt: 100.00


Antallet af dobbeltslag: 167491 hvilket svarer til: 16.75% af kastene.
```

Figur : Her ses programmet testet over 1000000 simulerede kast

- Vi må forvente 333 af hver type over de 1000 kast
  - 1'ere: 166+172 = 338. Hvilket er en afvigelse på 5 ift det forventede, hvilket er 0.5%
  - 2'ere: 158+167= 325. Hvilket er en afvigelse på 8 ift det forventede, hvilket er 0.8%
  - 3'ere: 158+164= 322. Hvilket er en afvigelse på 11 ift det forventede, hvilket er 1.1%
  - 4'ere: 171+172= 343. Hvilket er en afvigelse på 10 ift det forventede, hvilket er 1.0%
  - 5'ere: 174+147= 321. Hvilket er en afvigelse på 12 ift det forventede, hvilket er 1.2%
  - 6'ere: 181+178= 359. Hvilket er en afvigelse på 26 ift det forventede, hvilket er 2,6%
  - Dobbeltslag: 169. Hvilket er en afvigelse på 3 ift de forventede 166

Konklusion af test:

- Vi kan konkludere, at over tusind kast er den største afvigelse på 2.6 % jf. Fig 1 og på Fig. 2, samme test over en million kast, ses det at den største afvigelse er >0.5%. Hvilket må betyde at math.random metoden i Java kaster fuldstændig tilfældigt. Vi antager, at var testen var kørt over uendelige kast, ville resultatet være det forventede.

Unit tests

- En anden måde hvorpå, man kunne have testet programmet efter kundens specifikationer var ved at gøre brug af Unit tests. Mere specifikt J-unit tests, der går ud på at teste java programmer ved brug af assertion statements. Gruppen har dog ikke modtaget undervisning i teorien bag eller implementation af disse. Derfor har vi valgt ikke at gøre brug af dette værktøj under denne øvelse.

Reflektion

Konklusion

- I denne opgave lykkedes det os at lave et spil som opfyldte alle krav fra opgavebeskrivelsen. Vi kunne have givet os i kast med at bruge det GUI der blev foreslået at vi kunne bruge, hvilket ville have givet en bedre brugeroplevelse til spillere der ikke er vant til at bruge terminalbaserede kommandoer. I forbindelse med projektet, havde vi mulighed for at videreudvikle på programmet i form af tillægende use cases som kunden havde. Vi har valgt ikke at gøre brug af dette tilbud, men nogle af disse funktioner kunne have være implementeret ved at arbejde videre med spillere som klasser i stedet for simple int værdier. Man kunne også have opbevaret alle terning slagende i arrays af last values, så man kunne holde styr på hvorvidt om en spiller har slået to seksere to gange i træk.

Projektforløb

- Vi valgte at strukturere projektet ved at først danne os et overblik over hvad der skulle laves, og derefter uddelegerede vi dele af projektet imellem gruppemedlemmer. Dette gjorde at vi kunne arbejde mere effektivt med projektet, og undgik dobbeltarbejde ved at flere lavede på det samme. Vi havde derefter en deadline der hed, at alle skulle være færdige med deres ansvarsområder dagen inden vi skulle aflevere, så vi kunne læse rapporten igennem hver især. Vi rettede derefter de sidste ting til den efterfølgende dag.

  Kildekoden blev skrevet i én klasse til at starte med, og først derefter delt ud i andre klasser. Dette fungerede da projektet ikke var så stort, og der derfor ikke skulle bruges så mange avancerede klasser, men havde det været et større projekt skulle vi have delt koden op i klasser til at begynde med.

  Vi valgte ikke at arbejde i iterationer i dette projekt, da vi vurderede at projektet var for småt til at få glæde af at arbejde med mange af de fordele som Unified Process giver ved større projekter. Selvom vi ikke arbejdede i iterationer, fandt vi dog at vi måtte ændre på vores umiddelbare plan. Vi havde som udgangspunkt tænkt os at undlade at lave tests, og så blot skrive det på som en antagelse af at vi kunne noget vi ikke havde lært. Vi blev dog enige om at vi godt kunne lave en test alligevel, og dermed udførte vi vores tests alligevel.

Bilag

Bilag A

- Link til GitHub med Terningespillet's sourcecode
  - `https://github.com/madsstorgaardnielsen/CDIO1_20_del1/tree/dev/src/com/company`

Bilag B

- Link til GitHub med sourcecode til statestik over terninger
  - `https://github.com/madsstorgaardnielsen/TestAfTerningkast/blob/master/src/com/company/Main.java`