02314 62531 62532

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

---

CDIO delopgave 2

Gruppe 17

Jakob Skov Agergaard
s224570

Philip Muff Førrisdahl
s224566

Mads Fogelberg Hansen
s224563

Esben Skovmand Elnegaard
s224555

Jarl Boyd Roest
s224556

28. oktober 2022

Resumé

Ud fra kundens vision er der blevet udarbejdet en rapport, som indeholder et terningespil som ønsket. Ved hjælp af krav og kundens vision, er der blevet fremstillet en analyse i form af en use case beskrivelse. Use case beskrivelsen beskriver de funktioner, som programmet skal indeholde for at kunne køre efter kundens vision og krav. Derudover er der lavet en test, der kontrollerer at en spillers pointbalance ikke kan blive negativ. Der er til slut blevet lavet en konklusion, som konkludere at der er blevet fremstillet et program, som opfylder kundens krav og vision.

Indhold

1 Timeregnskab

- Arbejdsdag 1: 2 timer
- Arbejdsdag 2: 2,5 timer
- Arbejdsdag 3: 2 timer
- Arbejdsdag 4: 1 time
- Arbejdsdag 5: 2 timer

2 Indledning

I denne rapport, giver vi et indblik i vores udviklingsproces af spillet terningspil V2. Vi har udviklet spillet, på baggrund af kundens vision, med vejledning af projektlederne for at præcisere krav. Arbejdsprocessen har forsøgt a følge Rational Unified Process. Dermed har arbejdet foregået i små iterationer med evaluering mellem.

3 Projekt-planlægning

Begyndelsen: På dag et, sørgede vi for at stille projektlederen relevante spørgsmål for at opnå en kravsspecifikation der var detaljeret og som levede op til forventninger fra alle parter.
Her udarbejdede vi også vores korte use case.

22-10-2022: Vi mødtes på discord for at få et overblik over hvor langt vi var i processen, samt for at arbejde videre på projektet. Her blev der startet på system-sekvensdiagram

Uge 1:

4 Krav & Analyse

4.1 Spørgsmål til og svar fra projektlederen

Q: Hvad vil det sige at "Det skal være let at skifte til andre terninger"?

A: Man skal let kunne ændre i klassen, så der kan skriftes til et andet antal sider.

Q: Hvad vil det sige at "Spillet skal let kunne oversættes til andre sprog."?

A: En oversætter skal kunne tilgå alt tekst, uden at skulle rode for meget rundt i koden.

Q: Er det nødvendigt at spillerne har andre navne end "spiller 1 og 2"

A: Det er op til os som udviklere.

Q: Hvad vil det sige at der skal "udskrives en tekst omhandlende det aktuelle felt"? Skal vi selv finde på en tekst?

A: Når der landes på et felt skal der udskrives en tekst, som vi selv kan finde på. Den skal som minimum give information om hvilket felt det drejer sig om.

Q: Kan man gå i negativ pengebeholdning?

A: Nej, det kan man ikke.

4.2 Kravsliste

- Skal kunne spilles på DTU's maskiner i databarer
- To terninger
- Hver spiller har en pengebeholdning, der starter på 1000
- Spil indeholder felter med numrene fra 2-12
- Felter påvirker spillernes pengebeholdning (se bilag 1 for værdiliste)
- Pengebeholdninger skal kunne benyttes i andre spil
- Udskrivning af tekst der matcher feltet
- Skal kunne oversættes til andre sprog (have alle prints i samme klasse)
- Antallet af sider på terningerne skal kunne ændres af kunden (ikke brugeren)

4.3 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. Desuden besidder brugeren over 8. klasse pc færdighed, er op til 80 år gamle og har et almindeligt helbred.
Succes Guarantee: Spillet afsluttes og angiver en vinder.[1]

Main Success Scenario

1. Første spiller slår med terningerne
2. Systemet lægger øjnene på terningerne sammen
3. Første spillers brik bliver nu rykket det antal felter som terningernes øjne viser
4. System fortæller spilleren hvilket felt de er landet på og udskriver en kort beskrivelse af feltet, herunder hvor mange point man får på det givne felt.
5. Pointene bliver tilføjet/fratrukket spillerens pengebeholdning
6. Systemet skifter tur til næste spiller.
   Punkterne 1-6 bliver nu gentaget for næste spiller
7. Alle steps bliver nu gentaget indtil en spiller når en pengebeholdning på 3000 eller derover.
8. Systemet udskriver en tekst der fortæller spillerne hvem der har vundet.

Extension (Alternate Succes Scenario):

6.a En spiller slår 10.

   1. Punkterne 1-5 bliver gentaget for denne spiller (Ekstra tur)

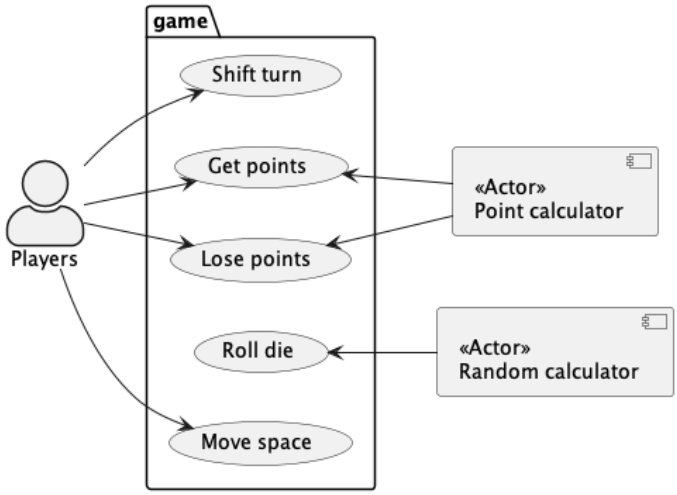
Sub use cases (beskrevet på 'brief format'):

- Ryk brik

[1] Gruppe 17 CDIO 1

- Slå terninger
- Få point
- Mist point
- Skift tur

Use case skrevet på 'fully dressed' format:
Første spiller trykker på enter for at starte spillet, og slår dermed også med begge terninger. Systemet udskriver dernæst hvilket felt spilleren er landet på, samt en kort beskrivelse af hvilken indflydelse feltet har på spillerens pengebeholdning.
Det er nu den anden spillers tur, og spilleren slår også med begge terninger. Systemet udskriver, på samme måde som før, hvilket felt spilleren er landet på, samt en kort beskrivelse af hvilken indflydelse feltet har på spillerens pengebeholdning.
Disse steps gentages indtil en spiller har en pengebeholdning på 3000. Når dette sker udskriver systemet hvilken spiller der har vundet, og spillet er dermed slut.
Use case diagram



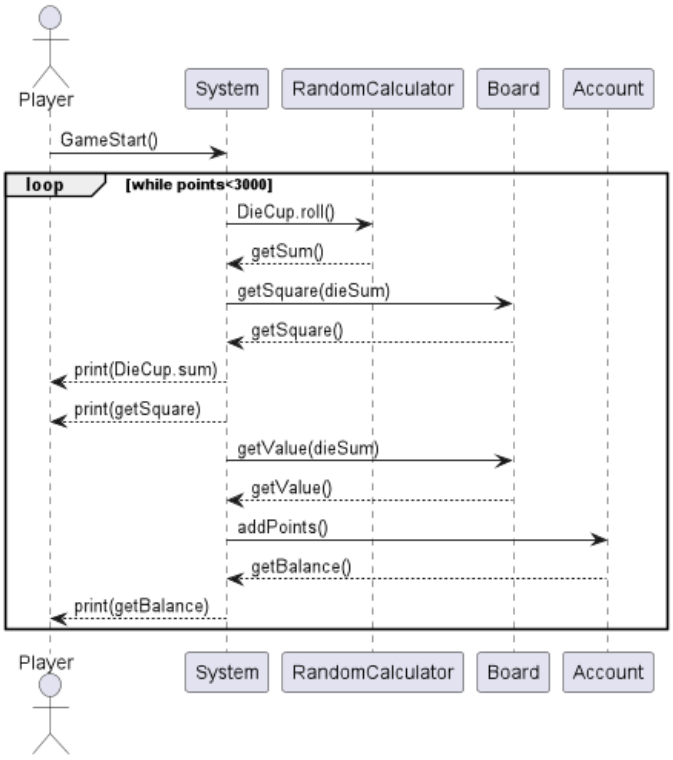
Figur 1: Use case diagram

5 Design

Vi har med henblik på at få lavet det mest optimale og overskuelige kode, lavet 2 former for design diagrammer, der danner et simpelt overblik på vores program. Med use case realization, har vi holdt den råde tråd, og påmindede os selv på kravene programmet skal have. vi har startet ud med at lave et use case diagram (fig 2) og et klasse diagram (fig 4). Heraf har vi udviklet et design klasse diagram (fig. 3) , og et systemsekvens diagram(fig. 1), der hver i sær viser spillet på deres måde. Med denne fremgangsmåde fik vi hurtigt sat klasserne op, med de defineret objekter og metoder.

6 Implementering

Vi har valgt at ligge alle Strings, der bliver printet, i én klasse. Dette giver mulighed for nemt at ændre teksten til et andet sprog.



Figur 2: System sequence diagram

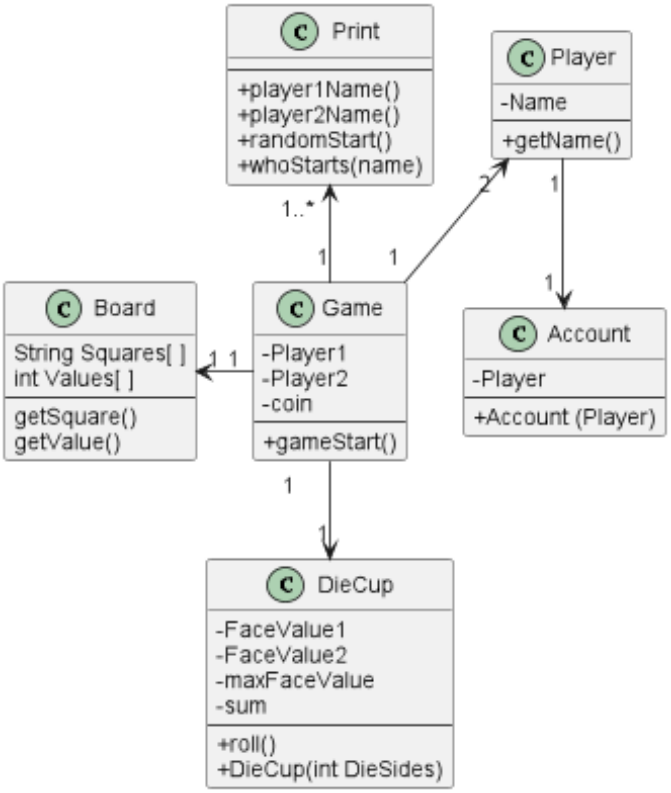
7 Test

Den primære årsag for at teste sit program er for at finde mulige fejl, men det kan også være for at kvalitetstjekke sit program eller forbedre det. Man bruger typisk iterativmetoden, og ikke vandfaldsmetoden, til at teste sit program, hvor man tester små bider af sit program løbende i programmerings processen. Dette gør at man løbende sikre sig at sit program fungere og lever op til kravene fra kunden.

Der er blevet efterspurgt en test af at en spillers balance ikke kan gå i negativ pengebeholdning.
Vores test er lavet på baggrund af af vores terningspil og tager derfor udgangspunkt der fra. Testen viser at pengebalancen ikke kan gå i negativ, som var det, der var efterspurgt.
Man kan derudover teste terningerne, for at se om de slag man kaster fordeler sig rigtig efter normalfordelingen af to terninger. Dette vil foreksempel kunne vise at terningerne ikke er snyde terninger, hvor der kan være større sandsynlighed for at slå et højere tal med de to terninger.

8 Konfiguration

Følgende beskrivelse udgør minimumskrav samt vejledning i hvordan kildekoden compiles, installeres og afvikles, samt hvordan koden importeres fra et git repository.
Som minimumskrav kræves det at styresystemet er af Windows version 10 eller mere. Computeren skal have java version 18 eller nyere.
Vejledning til kompilering, installering og afvikling:

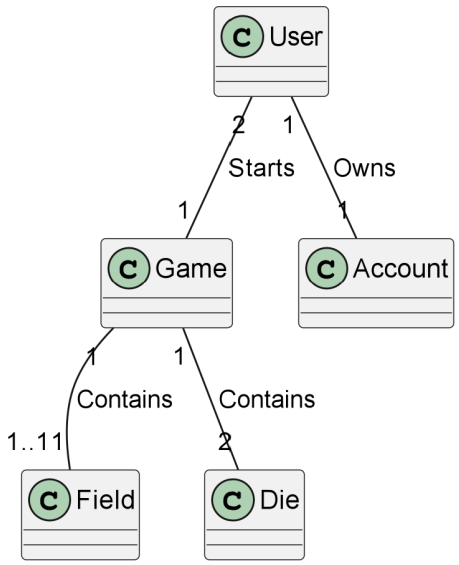


Figur 3: Design klasse diagram

1. Naviger til Github repo for projektet.
2. Tryk på "code"drop-downmenuen og download ZIP (vær sikker på du befinder dig i "master-branchen)
3. Udpak filen
4. Naviger til filen i en terminal og ind i mappen navngivet "src"
5. Her skriver du følgende i terminalen: *"javac Main.java"*. Dette kompilere programmet.
6. Herefter skriver du: *"java Main"* og programmet kører i kommandolinjen.

9 Konklusion

Ud fra testen, design, implementering og analysen, der er blevet fremstillet et terningespil. I analyse delen er der blevet analyserede på hvilke krav som kunden havde til sit spil, og hvordan man skulle tilgå dem. Derudover er der blevet lavet en Use Case beskrivelse, som beskriver hvilke funktioner der skal være tilstede i programmet, for at det fungere bedste muligt. Dernæst er programmet blevet illustrere i design diagrammer, som danner overblik over programmet, og gør det mere overskueligt og nemmere at kode selve programmet. Heraf kan konkluderes at der er blevet programmeret et terningespil Ud fra ovenstående tekst, kan det konkluderes at der er blevet fremstillet et terningespil med udgangspunkt i kundens krav og vision.



Figur 4: Klasse diagram

10 Bilag

10.0.1 Bilag 1: Liste med værdier for felter

Feltliste

1.	(Man kan ikke slå 1 med to terninger)	
2.	Tower	+250
3.	Crater	-100
4.	Palace gates	+100
5.	Cold Desert	-20
6.	Walled city	+180
7.	Monastery	0
8.	Black cave	-70
9.	Huts in the mountain	+60
10.	The Werewall (werewolf-wall)	-80, men spilleren får en ekstra tur.
11.	The pit	-50
12.	Goldmine	+650

10.0.2 Litteratur

1. Gruppe 17 - CDIO del 1
https://github.com/Jakob-SA/CDIO-1

10.0.3 Kode

Koden produceret i projektet kan findes på følgende link: https://github.com/Jakob-SA/CDIO-2