CDIO 1

02312 62531 62532
Indledende programmering, Udviklingsmetoder til IT-systemer og versionsstyring og testmetoder.

Emil Krøldrup
s224276

Matthias Bruun Kobbernagel
s214662

Filip Merlung
s180436

Anders Nørgaard
s224305

Noah Ravn Rasmussen Sewerin
s225822

30. September 2022

GRUPPE 5
DIPLOM SOFTWARETEKNOLOGI

1 Resumé

2 Timeregnskab

Indhold

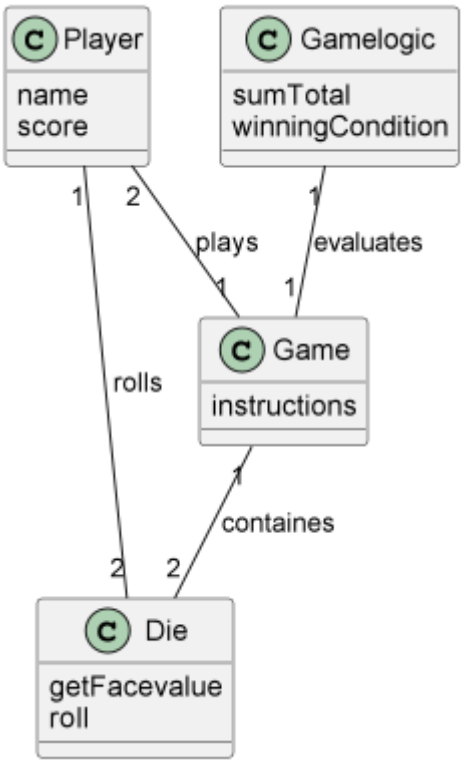
3 Indledning

Dette er en kort rapport om et system der kan spille et simpelt terningspil. I denne rapport er der kort beskrevet hvilke metoder fra UML vi har brugt i vores analysefase. Dertil har vi udarbejdet en kravspecifikation i samarbejde med projektleder og kunde. Der er også gjort brug af UML i vores designfase for at kunne påbegynde vores implementering. Dertil er der udarbejdet en test til projektet.

4 Projektplanlægning

5 Krav/Analyse

Her ses vores konceptuelle analyse-klassediagram, vi har udarbejdet for at få et overblik over hvad hvilke objekter vores terningspil skal indeholde(Se Figur 1).



Figur 1. Analyse-klassediagram

Dertil er der lavet en ekstrem simpel use case som gælder for vores system:
Spil spillet.

Kravspecifikation:

Et system, der kan bruges på maskinerne(computerne som bruger Windows og har Java version 17) i data-baren på DTU.
Det skal være et spil mellem 2 personer.
Spillet skal gå ud på at man slår med et raflebæger med to terninger og ser resultatet med det samme.
Terningerne har 6 sider hver.
Summen af terningernes værdier lægges til ens point.
Vinderen er den, der opnår 40 point eller derover.

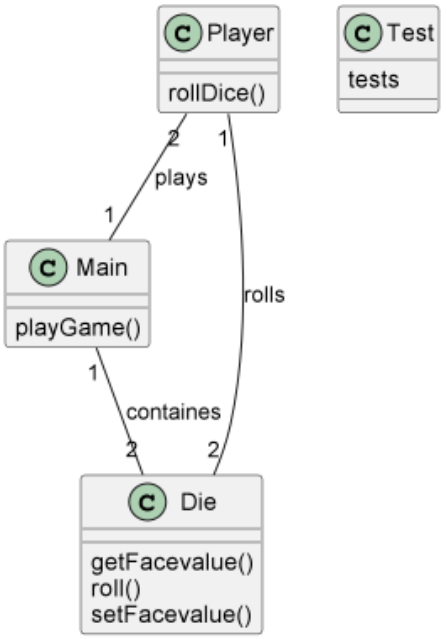
En person der kan operere en pc, skal kunne spille spillet uden en brugsanvisning(Systemet giver dig instruktioner undervejs.)

Der skal være en test som beviser at raflebægeret virker korrekt hen over 1000 kast.

Dokumentation og skal være på dansk eller engelsk, dog skal fagudtryk være naturlige.

6 Design

Her har vi efterfølgende redigeret i vores konceptuelle analyse klassediagram og lavet den om til en design klassediagram(Se figur 2). Dette klassediagram, findes også inde i selve vores projekt, i form af en .puml.



Figur 2. Design-klassediagram.

7 Implementering

Vi har udarbejdet en Die-klasse der ruller terningen og viser et tal mellem 1 og 6. Derudover har vi lavet en Player-klasse der holder styr på point og kan lægge terningernes sum sammen og lægge dem til pointene. Yderligere har vi samlet det i en Main med en playGame funktion som kører vores spil. Dertil har vi lavet en separat pakke med en test-klasse.

8 Test

Vi udarbejder en test der ruller terningen 1000 gange og gemmer de 1000 forskellige summer. Derefter bliver summen delt ud i de forskellige hyppigheder(2-12) og frekvenser målt i %.

9 Konklusion

Distributionen passer meget fint, ift vores test, der er en smule afvigelse ift til forventet fordeling, men dette må skyldes at testsample er for lille. Den skulle formentlig være markant større end 1000 rul, for at vi fik en frekvens der lå tættere op ad den forventede fordeling.

10 Bilag

10.1 Litteraturliste

Larman, C. (2005). *Applying Uml and patterns: An introduction to object-oriented analysis and design and the unified process* (3rd ed.). Prentice-Hall.

Lewis, J., & Loftus, W. (2018). *Java Software Solutions: Foundations of Program Design* (9th ed.). Pearson Education Limited.

10.2 Kode

https://github.com/andelsbolig/o5_del1.git