02314 62531 62532

Introductory Programming, Development Methods for IT Systems og Version Control and Testing Methods

CDIO 2

Group 22

s235068 - Casper Kielstrup

s235108 - Emil K. Petersen-Dalum

s235124 - Mirza Zia Beg

s235079 - Mohammed Ismail

s235771 - Younes Hamadi

s235084 - Nico Laursen

Deadline:

Friday, 27 Oktober 2023

Indhold

Resume

Vores forrige kunde var vild med vores første terningspil, så IOouterActive har fået til opgave at lave et nyt og revolutionerende terningspil.

Timeregnskab

Hver person i gruppen har arbejdet 6 timer om ugen på Campus og mindst 5 timer om ugen i hjemmet.

Introduktion

Vi har videreudviklet på vores forrige terningspil fra IOouterActive. Vi har implementeret flere classes for at simplificere main koden. Dette program er blevet gjort mulig ved gruppearbejde og vore viden fra Introductory programming, development methods for IT systems og Version control and Test methods. Vores arbejde indeholder kravliste, passende UML-diagrammer og koden til spillet.

Projekt planlægning

Dette projekt var delt i flere underprojekter. Vi startede hver især at udarbejde et UML-diagram. Dette skulle danne grundlaget for nem og effektivt under udviklingsfasen. Derefter delte vi de forskellige elementer op af kodningen mellem gruppemedlemmerne.

Krav

Kravliste:

- Hvis spilleren slår 10, får man en ekstra tur.
- Spillerne starter med en pengebeholdning på 1000.
- Spillet er slut når en spiller når op på en pengebeholdning på 3000.
- Man kan ikke slå 1.
- Der skal være et output passende til feltlisten.
- Det skal kunne spilles på maskinerne i DTU's databarer.
- Det skal være let at skifte til andre terninger.
- Det skal være let at skifte til andre sprog.
- Spilleren kan ikke have en negativ pengebeholdning.
- Spillerne slår plat eller krone om startkastet i spillet.
- Skal indeholde en klasse "spiller". Som indeholder en spillers attributter og funktioner.
- Skal indeholde en klasse "konto". Som beskriver spillerens pengeholdning.

Design

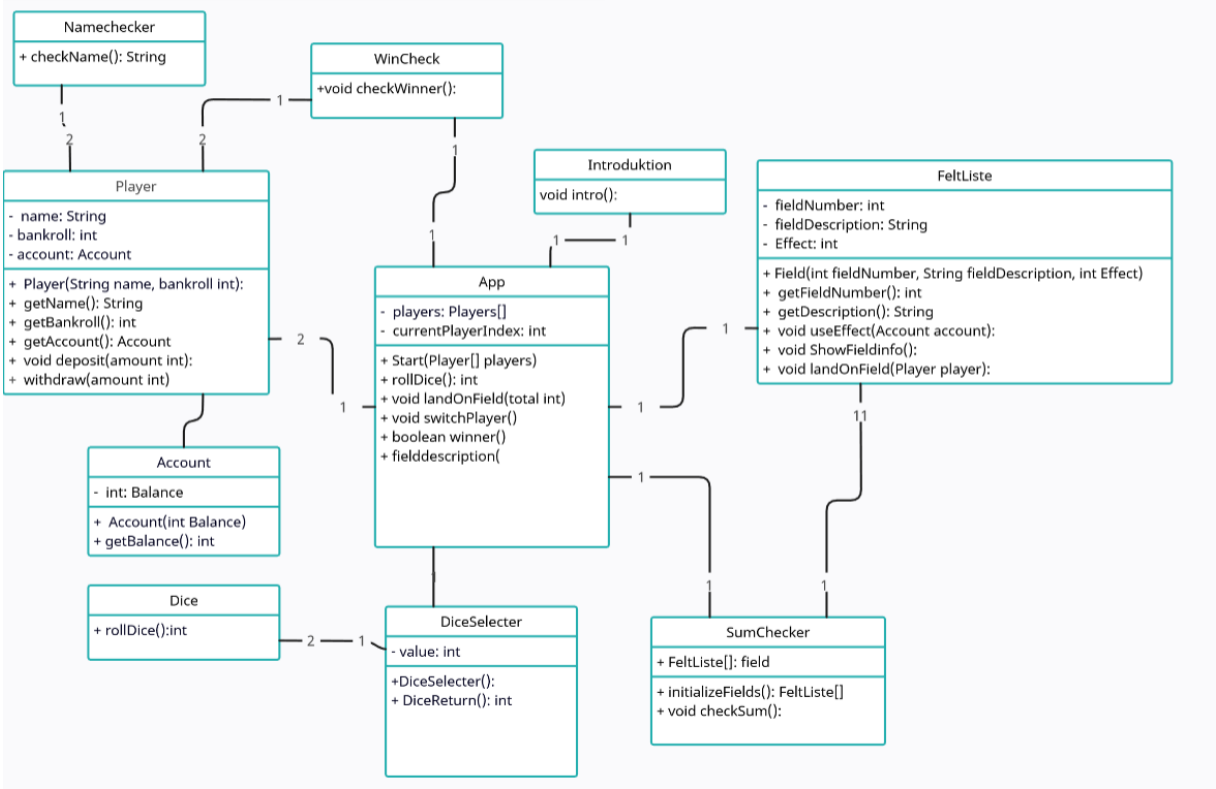
Use cases og beskrivelse:

Use case ID	Spille spillet
Kort beskrivelse	Use case 1 giver to spillere mulighed for at spille spillet på DTU's databar-maskiner. Spillerne skiftes til at kaste to terninger og lande på et felt, som er nummereret fra 2 til 12. Hvert felt har en positiv eller negativ effekt på spillernes pengesaldo, og en tilsvarende tekstbesked vises. Spillet starter med at begge spillere har en pengesaldo på 1000, og det slutter når én spiller når 3000.
Actors	Spiller 1, Spiller 2
Prerequisites	Spillet er klar til start.
Main flow	1. Spillet starter med at begge spillere har en startsaldo på 1000. 2. Spiller 1 kaster to terninger.

	3. Spillet beregner summen af terningkastet og flytter spilleren til det tilsvarende felt. 4. Spillet opdaterer spiller nummer 1's pengesaldo baseret på effekten af feltet. 5. Spillet viser en tekstmeddelelse relateret til feltet. 6. Hvis spiller nummer 1 når en pengesaldo på 3000, slutter spillet, og spiller 1 vinder. 7. Hvis ikke, bliver det spiller nummer 2's tur. 8. Spiller 2 kaster to terninger og gentager trin 3 til 7. 9. Spillet fortsætter, indtil en spiller når 3000 og vinder.

Use case ID	Felteffekter
Kort beskrivelse	Use case 2 beskriver de forskellige effekter af at lande på forskellige felter på spillepladen. Hvert felt i spillet har en specifik indflydelse på spillerens pengesaldo.
Actors	Spiller
Prerequisites	Spilleren er landet på en bane.
Main flow	1. Spillet tjekker feltnummeret, hvor spilleren er landet. 2. Baseret på feltnummeret bliver der anvendt den tilsvarende effekt på spillerens pengesaldo. 3. Spillet viser så en tekstmeddelelse, som beskriver effekten af feltet. 4. Hvis feltet er "The Werewall", får spilleren en ekstra tur.

Class diagram:



Forklaring:

Player Class:

Fields

String name: Navnet på spilleren

Int Bankroll: Mængden af penge på spillers konto

Account account: Instance af account-klassen forbundet til spilleren

Methods

Player(String name, Int Bankroll): Contructor af spiller-klassen som har argumenterne name for navnet på spilleren og Bankroll, som er beløbet spillerens konto starter på.

String getName(): Return spillerens navn

Int getBankroll(): Return nuværende beløb på spillerens konto.

Account getAccount: Return account tilhørende nuværende spiller.

Void deposit(int amount): indsæt beløb på spillerens konto

Void withdraw(int amount): fjern beløb fra spillerens konto

Account Class

Fields

Int balance: Nuværende beløb på spillers konto

Methods

Account(int balance): Constructor til Account-klassen

GetBalance(): return nuværende beløb på kontoen

Spil

Fields

Players Players[]: array af alle spillerne

CurrentPlayerIndex: Return nummeret på nuværende spiller(bruges til at identificere hvilken spiller som skal slå med terningerne)

Methods

Start(Player[]players): Vores main funktion, fungerer ved at der bliver dannet et array udfra de to spillers navne man taster ind i the command-line-interface.

Int rollDice(): Skal return et tilfældigt nr mellem 1 og 6.

Void landOnField(total int): Den skal tage summen af to terningekast som argument og finde det felt på mappet, der er tilsvarende summen af de to kast.

Void switchPlayer(): Skifter hvilken spiller der skal slå med terningerne

Boolean isWinner():Tjekker om en af de to spillere har penge tilsvarende eller større end beløbet man skal have for at vinde

String fielddescription(): skal return en string som beskriver ved det felt man landede på gør

Field/Map

Fields

Int fieldNumber: Nummeret på given felt

Int Effect: Den effekt feltet udløser, eksempelvis at spiller mister 50 point

String fieldDescription: beskrivelse af given felt og feltets effekter

Methods

Field(int fieldNumber, String fieldDescription, int Effect): Contructor til given felt, hvert felt skal sættes op separat med denne contstructor.

Int getFieldNumber(): Return nummeret på given felt

String getDescription(): Return en string som beskriver nuværende felt.

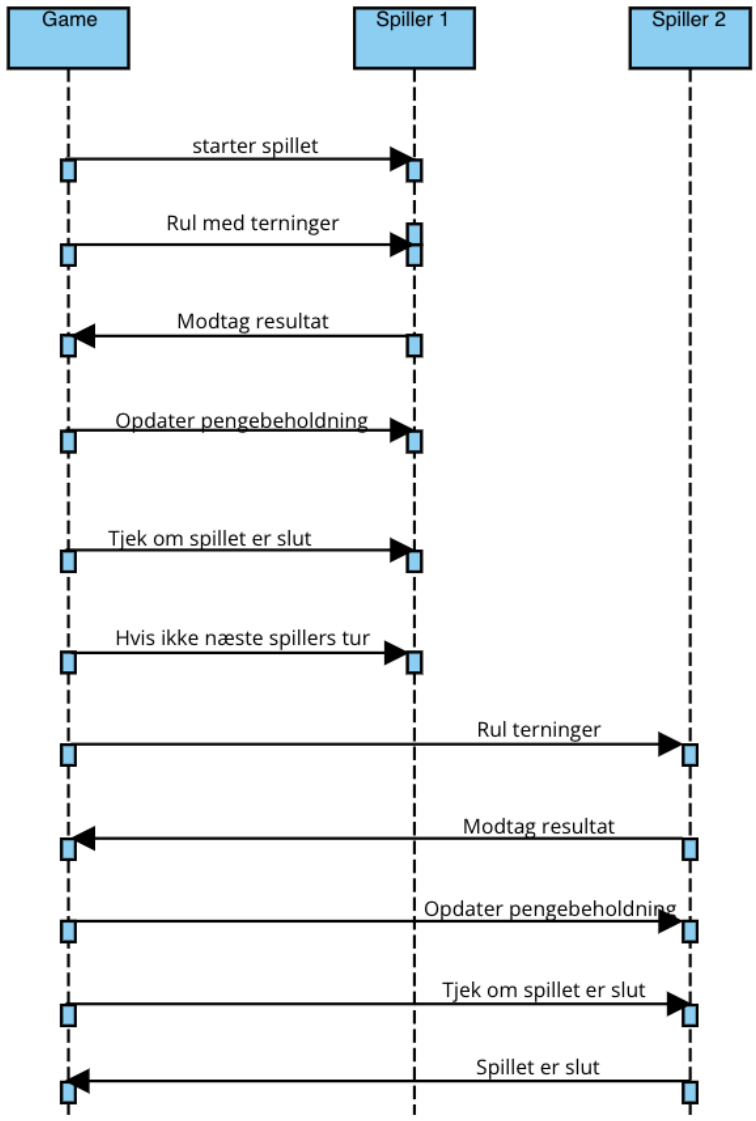
Int getEffect(): Return en integer som beskriver hvor mange point spilleren har mistet eller fået.

Void useEffect(Account account): Bruger effekten på given konto, eksempelvis fjerner den 30 point.

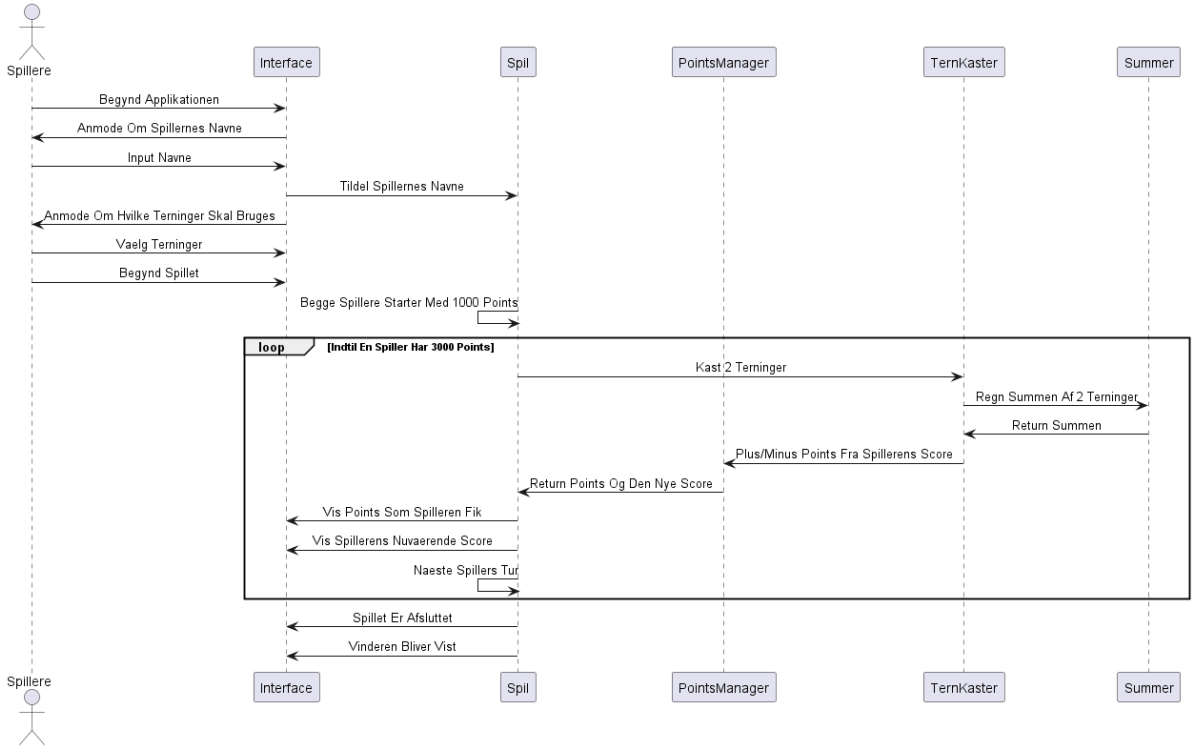
Void showFieldInfo(): Bruger getDescription til at printe en string som forklarer given felt

Void useEffectOnPlayer(Player player): Hvis et felt har effekter udover at ændre pointmængden på en konto, bruger vi denne funktion, eksempelvis får man en ekstra tur når man lander på werewall feltet

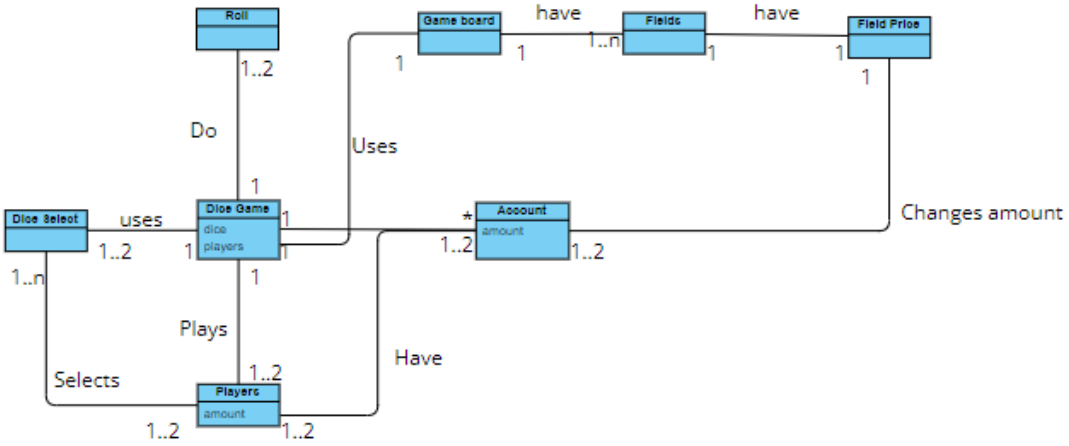
System-sekvensediagram:



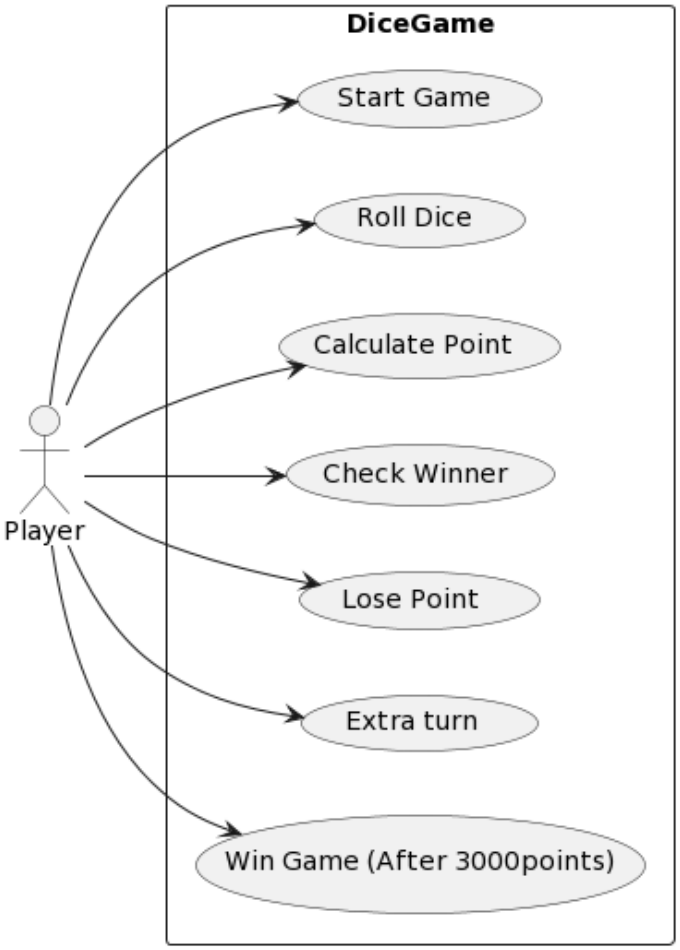
Sekvensdiagram:



Domænemodel:



Use case diagram:



Implementering

Da det er et krav at programmet skal kunne køres på computerne i databarene på DTU, har vi valgt at implementere den ved brug af terminalen. Databarene hos DTU har Java 13 installeret, så programmet kan køres ved at kompilere filerne og så køre App. F.eks. hvis alle .java filerne er i én mappe, skal man åbne en terminal i den mappe, og skrive “javac *.java”, og så “java App” efterfølgende.

Testning

Vi har lavet en jUnit test på vores account-klasse. Koden til testen ligger i en separat test mappe, da det´ gav problemer med at kompilere med de andre javafiler. For at køre testen, skal man kopiere den ind i src mappen, og det er et krav at have “extention pack for java” installeret på Visual Studio Code.

Der bliver testet for at balancen i account-klassen ikke kan blive negativ. Derudover har vi også testet for at deposit og withdraw metoderne fungerer som de skal. Alle tre test giver det forventede output.

Konklusion

Da vores tests giver os de forventede resultater, kan vi derved konkludere at koden virker som forventet, og at alle de givne krav fra IOOuterActive er opfyldt. Efter at have ladet andre teste programmet kan vi også se at det ikke er nødvendigt at kende alle funktionerne af koden.
Vores funktion ”extra turn” har vi ikke givet sin egen klasse, selvom dette normalt ville være ideelt. Vi har gjort dette, for at gøre udviklingsprocessen hen mod slutningen nemmere for os selv, da det ikke er nødvendigt at give funktionen sin egen klasse for at programmet virker som det skal.

Bilag

Literatur:

- Dice klassen fra forrige DiceGame.

Kode:

- https://github.com/ZabZZZZZ/22_CDIO_2