



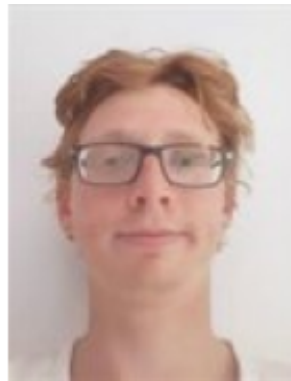
Danmarks  
Tekniske  
Universitet

62531 udviklingsmetoder til it-system  
62532 versionsstyring og testmetoder  
02312 indledende programmering

**Gruppe 23**



Hans Christian  
Leth-Nissen  
s205435



Henrik Lynggaard  
Skindhøj  
s205464



Kasper Falch  
Skov  
s205429



Christoffer Fink  
s205449



Phillip Igor Meyer  
Clausen  
s205468

Henrik Lynggaard Skindhøj, s205464  
Hans Christian Leth-Nissen, s205435  
Filip Igor Meyer Clausen, s205468

Christoffer Fink, s205449  
Kasper Falch Skov, s205429

## Github repo:

[https://github.com/Kageschwump/23\\_del3](https://github.com/Kageschwump/23_del3)

## Timeregnskab

Kasper Falch Skov, s205429	1 timer
Christoffer Fink, s205449	1 timer
Filip Igor Meyer Clausen, s205469	1 timer
Henrik Lynggaard Skindhøj, s205464	1 timer
Hans Christian Leth-Nissen, s 205435	1 timer

## Abstract

## Indholdsfortegnelse

### 1. Indledning


I opgavebeskrivelsen har vi fået til opgave at lave et Monopoly Junior spil. Vi har fået udleveret reglerne til spillet, og ud over dette har vi fået mere eller mindre frie rammer til at vurdere hvad vi mener der er det vigtigste i reglerne. I projektet vil vi udarbejde vores krav efter F.U.R.P.S modellen. Efterfølgende vil vi udarbejde nogle use cases, som vi beskriver fully dressed. Herefter vil vi finde frem til nogle passende klasser, som vil være relevante for programmet, og ud fra disse vil vi opstille en række artefakter. Vi vil også lave en række tests, som kunden kan køre, så vi sikrer os, at programmet kører som det skal. Vi har fået udleveret en GUI, som kunden gerne vil have vi bruger i spillet, så den gør vi selvfølgelig også brug af.

### 2. Kravspecifikation

Vi har valgt at dele vores krav op i FURPS-princippet

#### **Functionality:**

Alle krav som ligger her under functionality, har vi lagt her, fordi vi mener at disse krav har noget at gøre med hvordan spillet skal opføre sig, når det er færdigudviklet.

- F1: Spillet skal kunne spilles mellem 2-4 spillere.
- F2: Spilleren skal kunne lande på et felt og fortsætte fra det felt næste gang det er deres tur.
- F3: Den yngste spiller starter.
- F4: Alle spillere starter på feltet "Start"
- F5: Spillerne skal rykke sig med uret
- F6: Spilleren skal altid rykke fremad, aldrig tilbage.
- F7: En spiller rykker det antal felter, som terningen viser i øjne, efter de har slået med den.
- F8: Hver gang man passerer start, modtager man 2 Monopoly W- skejs. 
- F9: Alle spillere går på skift, så der bliver skiftet tur, hver gang en spiller har rykket et antal felter
- F10: Hvis en spiller lander på et ledigt felt, **skal** personen betale banken det beløb der står på feltet, og placere et solgt skilt på feltet.
- F11: Hvis en spiller lander på et felt, der er ejet af en anden spiller, skal de betale husleje til ejeren af feltet. Huslejen er det beløb der står skrevet på feltet.
- F12: Hvis en spiller lander på et felt som spilleren selv ejer, sker der ikke noget
- F13: Hvis en spiller ejer begge ejendomme i samme farve er huslejen det dobbelte af det beløb, som står på feltet.
- F14: Hvis en spiller lander på et "CHANCE" felt, skal de tage det øverste chance-kort fra bunken og følge instruktionerne, herefter skal de ligge kortet tilbage nederst i bunken
- F15: Hvis en spiller lander på et "GÅ I FÆNGSEL" felt, skal spilleren gå til "FÆNGSEL" feltet, hvor de ikke modtager 2 skejs for at passere start. I starten af den fængslede spillers næste tur, skal de betale 1 skejs, eller bruge "Du løslades uden omkostninger"-kortet hvis spilleren har det. Derefter skal spilleren kaste med terningen, og rykke normalt. En spiller kan godt modtage husleje, mens de er i fængsel.
- F16: Hvis en spiller lander på et "PÅ BESØG" felt, sker der ikke noget
- F17: Hvis en spiller lander på et "GRATIS PARKERING" felt, sker der ikke noget
- F18: Hvis en spiller ikke har nok penge til at betale husleje, købe en ejendom, som personen lander på, eller betale afgiften fra et chancekort, er personen gået fallit, og spillet slutter.
- F19: Den spiller der har flest penge, når spillet er slut, vinder.
- F20: Hvis 2 spillere har samme antal penge efter spillet er slut, optælles værdien af ejendomme og lægges til pengebeholdningen.

### Usability:

Usability beskriver for det meste en blanding af dokumentation, æstetisk og responsivt design.

U1: Der skal være tydelig sammenhæng mellem beskrivelser og diagrammer, samt implementering af kode.

U2: Projektet skal udarbejdes efter GRASP-mønstre.

### **Reliability:**

Reliability krav er krav der har noget at gøre med om programmet er stabilt, altså at det ikke crasher på underlige tidspunkter. Samtidig sikre vi også at programmet kører som det skal uden diverse bugs.

R1: Der skal laves mindst 3 testcases med tilhørende fremgangsmåder, testprocedure og testrapporter.

R2: Der skal laves mindst én JUnit test til centrale metoder.

R3: Der skal laves mindst en brugertest. Testen skal udføres med en der ikke har kendskab til kodning.

### **Performance:**

DTU's maskiner skal kunne køre kører programmet. Her tester vi programmets helhed ved indførelse af Junit tests, samt kørsel på DTU's maskiner. På DTU's maskiner tids tester vi vores kode.

P1: Spillet skal køre uden bemærkelsesværdige forsinkelser, på DTUs maskiner

### **Supportability:**

Under supportability har vi valgt at ligge krav, som vi mener gør det nemmere for kunden eller andre spiludviklere at følge vores projekt. Dette gør at kunden kan følge udviklingen af projektet, og hvis spillet skulle outsources til en anden udvikler, vil det også være nemmere at få et overblik over koden.

S1: Projektet skal udarbejdes i Git, da kunden gerne vil kunne følge med i udviklingen af spillet.

S2: Projektet skal indeholde en beskrivelse af hvordan man importerer projektet fra git til IntelliJ, og hvordan man derefter kører programmet.

S3: Koden skal commites hver gang en delopgave er løst, eller mindst en gang i timen

S4: Hvert commit skal indeholde en kort beskrivelse, der forklarer, hvad der er lavet.

### 3. Analyse

#### Use case diagram

#### Use cases

Vi mener ikke der er brug for mere end tre usecases når man snakker om et brætspil. Det eneste vi har brug for er at starte spillet, at slå med terningerne, og at trække et kort, når man lander på et chance-felt. Ud over dette er spillet rimelig selvkørende i og med at det følger et sæt regler, som allerede er defineret

Use case section:	Beskrivelse
Navn	UC 1.1 - Start game
Scope	System
Level	User Goal
Primær aktør	Player
Preconditions	Spillet er ikke allerede igang
Postconditions	Motherlode er sat op, og klar til at spille
Main Success Scenario	1. Spiller: Kører programmet 2. System: Opretter Motherlode 3. System: Opretter to spillere 1. System: Giver hver spiller 1000 point 5. Spiller kan spille spillet
Extensions	NaN
Frequency of occurrence	Hver gang der skal startes et nyt spil

Henrik Lynggaard Skindhøj, s205464  
Hans Christian Leth-Nissen, s205435  
Filip Igor Meyer Clausen, s205468

Christoffer Fink, s205449  
Kasper Falch Skov, s205429

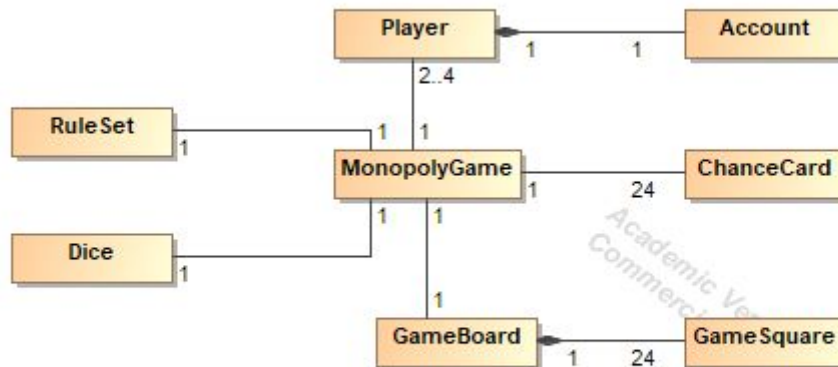
Use case section:	Beskrivelse
Navn	UC 1.2 - Roll dice
Scope	System
Level	User Goal
Primær aktør	Player
Preconditions	Motherlode er oprettet/startet
Postconditions	Der er rullet en terning
Main Success Scenario	<p>1. System: Spørger efter inputtet "enter" fra spilleren.</p> <p>2. Player: Trykker "enter" i programmet for at rulle med terningerne, og ruller terningerne.</p> <p>3. System: Generer to tal fra 1-6, lægger dem sammen, og rykker det antal øjne på brættet.</p> <p>4. System: Printer feltets tilhørende tekst ud</p> <p>5. System: Tildel det antal point af det felt Player lander på</p>
Extensions	5.1. Spilleren: Har fået 3000 point og vinder
Frequency of occurrence	Hver gang en af de to spillere skal kaste en terning, altså hver tur

Henrik Lynggaard Skindhøj, s205464  
Hans Christian Leth-Nissen, s205435  
Filip Igor Meyer Clausen, s205468

Christoffer Fink, s205449  
Kasper Falch Skov, s205429

Use case section:	Beskrivelse
Navn	UC 1.2 - Draw card
Scope	System
Level	User Goal
Primær aktør	Player
Preconditions	En player lander på et chance-felt
Postconditions	Chancekortets betingelser er udført
Main Success Scenario	1. System: Trækker et kort for spilleren 2. System: Printer chance-kortets indhold 3. System: Udfører kortets betingelser
Extensions	NaN
Frequency of occurrence	Hver gang en player lander på et chance-kort

## Domænemodel



Vores domænemodel forsøger at beskrive spillet og associationerne. I vores tilfælde har vi valgt at angive "MonopolyGame" som en klasse. Se klassen

## System Sekvensdiagram

### 4. Design

#### Design Klassediagram

#### Sekvensdiagram

#### GRASP mønstre

### 5. Implementering

- Lav passende konstruktører.
- Lav passende get og set metoder.



Henrik Lynggaard Skindhøj, s205464  
Hans Christian Leth-Nissen, s205435  
Filip Igor Meyer Clausen, s205468

Christoffer Fink, s205449  
Kasper Falch Skov, s205429

Lav passende toString metoder.  
Lav en klasse GameBoard der kan indeholde alle felterne i et array.  
Tilføj en toString metode der udskriver alle felterne i arrayet.  
Lav det spil kunden har bedt om med de klasser I nu har.  
Benyt GUI'en. Gui' skal importeres fra Maven: [Maven repository](#)

## 6. Dokumentation

Forklar hvad arv er.  
Forklar hvad abstract betyder.  
Fortæl hvad det hedder hvis alle fieldklasserne har en landOnField metode der gør noget forskelligt.  
Dokumentation for test med screenshots.  
Dokumentation for overholdt GRASP.

## 7. Test

## 8. Projektplanlægning

Versionsstyring

Konfigurationsstyring

## 9. Konklusion

## Bibliografi

Programmerings bogen

Lewis, J.. & Loftus, W.. (2017). *Java Software Solutions: Foundations of Program Design* (9.. udg.). Pearson Education Limited.

UML bogen

Larman, C.. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3.. udg.). Pearson Education.

Henrik Lynggaard Skindhøj, s205464  
Hans Christian Leth-Nissen, s205435  
Filip Igor Meyer Clausen, s205468

Christoffer Fink, s205449  
Kasper Falch Skov, s205429

CDIO 1  
CDIO 2

**Bilag**